# Garfield

0.3

Generated by Doxygen 1.14.0

1 Garfield++ 1
1.1 Building the project . . . 1
1.1.1 Options . . . 1
1.2 Building the documentation . . . 1
1.3 Running the examples . . . 1
1.4 Building and running examples using an installed version of Garfield . . . 2
1.5 Inclusion into other projects . . . 2

2 Namespace Index 2
2.1 Namespace List . . . 2

3 Hierarchical Index 2
3.1 Class Hierarchy . . . 2

4 Class Index 7
4.1 Class List . . . 7

5 File Index 13
5.1 File List . . . 13

6 Namespace Documentation 16
6.1 Garfield Namespace Reference . . . 16
6.1.1 Typedef Documentation . . . 20
6.1.2 Enumeration Type Documentation . . . 21
6.1.3 Function Documentation . . . 22
6.1.4 Variable Documentation . . . 56
6.2 Garfield::Degrade Namespace Reference . . . 56
6.2.1 Function Documentation . . . 57
6.3 Garfield::Magboltz Namespace Reference . . . 61
6.3.1 Function Documentation . . . 65
6.3.2 Variable Documentation . . . 66
6.4 Garfield::Numerics Namespace Reference . . . 86
6.4.1 Detailed Description . . . 87
6.4.2 Function Documentation . . . 88
6.5 Garfield::Numerics::CERNLIB Namespace Reference . . . 92
6.5.1 Detailed Description . . . 93
6.5.2 Function Documentation . . . 93
6.6 Garfield::Numerics::QUADPACK Namespace Reference . . . 95
6.6.1 Detailed Description . . . 95
6.6.2 Function Documentation . . . 96
6.7 Garfield::Polygon Namespace Reference . . . 98
6.7.1 Function Documentation . . . 98
6.8 Heed Namespace Reference . . . 99

7 Class Documentation 99

7.1 Garfield::AvalancheGrid Class Reference . . . 99
7.1.1 Detailed Description . . . 101
7.1.2 Constructor & Destructor Documentation . . . 101
7.1.3 Member Function Documentation . . . 101
7.1.4 Member Data Documentation . . . 105
7.2 Garfield::AvalancheGridSpaceCharge Class Reference . . . 108
7.2.1 Detailed Description . . . 112
7.2.2 Constructor & Destructor Documentation . . . 112
7.2.3 Member Function Documentation . . . 112
7.2.4 Member Data Documentation . . . 120
7.3 Garfield::AvalancheMC Class Reference . . . 128
7.3.1 Detailed Description . . . 133
7.3.2 Member Enumeration Documentation . . . 133
7.3.3 Constructor & Destructor Documentation . . . 133
7.3.4 Member Function Documentation . . . 134
7.3.5 Member Data Documentation . . . 148
7.4 Garfield::AvalancheMicroscopic Class Reference . . . 154
7.4.1 Detailed Description . . . 159
7.4.2 Constructor & Destructor Documentation . . . 159
7.4.3 Member Function Documentation . . . 160
7.4.4 Friends And Related Symbol Documentation . . . 177
7.4.5 Member Data Documentation . . . 177
7.5 Garfield::AvalancheGrid::AvalancheNode Struct Reference . . . 186
7.5.1 Detailed Description . . . 186
7.5.2 Member Data Documentation . . . 186
7.6 Garfield::MediumSilicon::Band Struct Reference . . . 188
7.6.1 Detailed Description . . . 188
7.6.2 Member Data Documentation . . . 189
7.7 Garfield::ComponentAnalyticField::Charge3d Struct Reference . . . 191
7.7.1 Detailed Description . . . 191
7.7.2 Member Data Documentation . . . 191
7.8 Garfield::TrackBichsel::Cluster Struct Reference . . . 192
7.8.1 Detailed Description . . . 192
7.8.2 Member Data Documentation . . . 192
7.9 Garfield::TrackDegrade::Cluster Struct Reference . . . 193
7.9.1 Detailed Description . . . 193
7.9.2 Member Data Documentation . . . 194
7.10 Garfield::TrackElectron::Cluster Struct Reference . . . 195
7.10.1 Detailed Description . . . 195
7.10.2 Member Data Documentation . . . 195
7.11 Garfield::TrackHeed::Cluster Struct Reference . . . 196
7.11.1 Detailed Description . . . 196
7.11.2 Member Data Documentation . . . 196

7.12 Garfield::TrackPAI::Cluster Struct Reference . . . 198
7.12.1 Detailed Description . . . 198
7.12.2 Member Data Documentation . . . 198
7.13 Garfield::TrackSimple::Cluster Struct Reference . . . 199
7.13.1 Detailed Description . . . 199
7.13.2 Member Data Documentation . . . 199
7.14 Garfield::TrackSrim::Cluster Struct Reference . . . 200
7.14.1 Detailed Description . . . 200
7.14.2 Member Data Documentation . . . 200
7.15 Garfield::TrackTrim::Cluster Struct Reference . . . 202
7.15.1 Detailed Description . . . 202
7.15.2 Member Data Documentation . . . 202
7.16 Garfield::ComponentAnalyticField Class Reference . . . 203
7.16.1 Detailed Description . . . 213
7.16.2 Member Enumeration Documentation . . . 213
7.16.3 Constructor & Destructor Documentation . . . 214
7.16.4 Member Function Documentation . . . 215
7.16.5 Member Data Documentation . . . 250
7.17 Garfield::ComponentAnsys121 Class Reference . . . 261
7.17.1 Detailed Description . . . 266
7.17.2 Constructor & Destructor Documentation . . . 266
7.17.3 Member Function Documentation . . . 266
7.18 Garfield::ComponentAnsys123 Class Reference . . . 268
7.18.1 Detailed Description . . . 273
7.18.2 Constructor & Destructor Documentation . . . 273
7.18.3 Member Function Documentation . . . 273
7.19 Garfield::ComponentComsol Class Reference . . . 274
7.19.1 Detailed Description . . . 280
7.19.2 Constructor & Destructor Documentation . . . 280
7.19.3 Member Function Documentation . . . 280
7.19.4 Member Data Documentation . . . 283
7.20 Garfield::ComponentConstant Class Reference . . . 284
7.20.1 Detailed Description . . . 285
7.20.2 Constructor & Destructor Documentation . . . 285
7.20.3 Member Function Documentation . . . 286
7.20.4 Member Data Documentation . . . 289
7.21 Garfield::ComponentCST Class Reference . . . 291
7.21.1 Detailed Description . . . 298
7.21.2 Constructor & Destructor Documentation . . . 298
7.21.3 Member Function Documentation . . . 298
7.21.4 Member Data Documentation . . . 304
7.22 Garfield::ComponentElmer Class Reference . . . 306
7.22.1 Detailed Description . . . 311

7.22.2 Constructor & Destructor Documentation . . . 311
7.22.3 Member Function Documentation . . . 311
7.23 Garfield::ComponentElmer2d Class Reference . . . 312
7.23.1 Detailed Description . . . 317
7.23.2 Constructor & Destructor Documentation . . . 317
7.23.3 Member Function Documentation . . . 317
7.24 Garfield::ComponentFieldMap Class Reference . . . 318
7.24.1 Detailed Description . . . 323
7.24.2 Member Enumeration Documentation . . . 323
7.24.3 Constructor & Destructor Documentation . . . 323
7.24.4 Member Function Documentation . . . 323
7.24.5 Member Data Documentation . . . 335
7.25 Garfield::ComponentGrid Class Reference . . . 338
7.25.1 Detailed Description . . . 342
7.25.2 Member Enumeration Documentation . . . 342
7.25.3 Constructor & Destructor Documentation . . . 342
7.25.4 Member Function Documentation . . . 343
7.25.5 Member Data Documentation . . . 351
7.26 Garfield::ComponentNeBem2d Class Reference . . . 354
7.26.1 Detailed Description . . . 357
7.26.2 Member Enumeration Documentation . . . 357
7.26.3 Constructor & Destructor Documentation . . . 357
7.26.4 Member Function Documentation . . . 357
7.26.5 Member Data Documentation . . . 364
7.27 Garfield::ComponentNeBem3d Class Reference . . . 366
7.27.1 Detailed Description . . . 371
7.27.2 Member Enumeration Documentation . . . 371
7.27.3 Constructor & Destructor Documentation . . . 371
7.27.4 Member Function Documentation . . . 371
7.27.5 Member Data Documentation . . . 382
7.28 Garfield::ComponentNeBem3dMap Class Reference . . . 386
7.28.1 Detailed Description . . . 388
7.28.2 Constructor & Destructor Documentation . . . 388
7.28.3 Member Function Documentation . . . 389
7.28.4 Member Data Documentation . . . 393
7.29 Garfield::ComponentParallelPlate Class Reference . . . 395
7.29.1 Detailed Description . . . 397
7.29.2 Member Enumeration Documentation . . . 398
7.29.3 Constructor & Destructor Documentation . . . 398
7.29.4 Member Function Documentation . . . 398
7.29.5 Member Data Documentation . . . 404
7.30 Garfield::ComponentTcad2d Class Reference . . . 407
7.30.1 Detailed Description . . . 411

7.30.2 Constructor & Destructor Documentation . . . 411
7.30.3 Member Function Documentation . . . 411
7.30.4 Member Data Documentation . . . 414
7.31 Garfield::ComponentTcad3d Class Reference . . . 414
7.31.1 Detailed Description . . . 418
7.31.2 Constructor & Destructor Documentation . . . 418
7.31.3 Member Function Documentation . . . 419
7.31.4 Member Data Documentation . . . 422
7.32 Garfield::ComponentTcadBase< N > Class Template Reference . . . 422
7.32.1 Detailed Description . . . 425
7.32.2 Constructor & Destructor Documentation . . . 425
7.32.3 Member Function Documentation . . . 426
7.32.4 Member Data Documentation . . . 435
7.33 Garfield::ComponentUser Class Reference . . . 439
7.33.1 Detailed Description . . . 441
7.33.2 Constructor & Destructor Documentation . . . 441
7.33.3 Member Function Documentation . . . 441
7.33.4 Member Data Documentation . . . 445
7.34 Garfield::ComponentVoxel Class Reference . . . 446
7.34.1 Detailed Description . . . 449
7.34.2 Constructor & Destructor Documentation . . . 449
7.34.3 Member Function Documentation . . . 449
7.34.4 Member Data Documentation . . . 455
7.35 Garfield::MediumMagboltz::Deexcitation Struct Reference . . . 457
7.35.1 Detailed Description . . . 457
7.35.2 Member Data Documentation . . . 458
7.36 Garfield::ComponentTcadBase< N >::Defect Struct Reference . . . 459
7.36.1 Detailed Description . . . 459
7.36.2 Member Data Documentation . . . 459
7.37 Garfield::DriftLineRKF Class Reference . . . 459
7.37.1 Detailed Description . . . 463
7.37.2 Member Enumeration Documentation . . . 463
7.37.3 Constructor & Destructor Documentation . . . 463
7.37.4 Member Function Documentation . . . 463
7.37.5 Member Data Documentation . . . 472
7.38 Garfield::ComponentParallelPlate::Electrode Struct Reference . . . 474
7.38.1 Detailed Description . . . 475
7.38.2 Member Data Documentation . . . 475
7.39 Garfield::AvalancheMicroscopic::Electron Struct Reference . . . 476
7.39.1 Detailed Description . . . 476
7.39.2 Member Data Documentation . . . 476
7.40 Garfield::TrackDegrade::Electron Struct Reference . . . 476
7.40.1 Detailed Description . . . 477

7.40.2 Member Data Documentation . . . 477
7.41 Garfield::TrackHeed::Electron Struct Reference . . . 477
7.41.1 Detailed Description . . . 478
7.41.2 Member Data Documentation . . . 478
7.42 Garfield::ComponentFieldMap::Element Struct Reference . . . 478
7.42.1 Detailed Description . . . 478
7.42.2 Member Data Documentation . . . 479
7.43 Garfield::ComponentNeBem2d::Element Struct Reference . . . 479
7.43.1 Detailed Description . . . 479
7.43.2 Member Data Documentation . . . 479
7.44 Garfield::ComponentNeBem3d::Element Struct Reference . . . 480
7.44.1 Detailed Description . . . 481
7.44.2 Member Data Documentation . . . 481
7.45 Garfield::ComponentNeBem3dMap::Element Struct Reference . . . 482
7.45.1 Detailed Description . . . 482
7.45.2 Member Data Documentation . . . 482
7.46 Garfield::ComponentTcadBase< N >::Element Struct Reference . . . 483
7.46.1 Detailed Description . . . 483
7.46.2 Member Data Documentation . . . 483
7.47 Garfield::ComponentVoxel::Element Struct Reference . . . 484
7.47.1 Detailed Description . . . 484
7.47.2 Member Data Documentation . . . 484
7.48 Garfield::AvalancheMC::EndPoint Struct Reference . . . 484
7.48.1 Detailed Description . . . 485
7.48.2 Member Data Documentation . . . 485
7.49 Garfield::TrackDegrade::Excitation Struct Reference . . . 485
7.49.1 Detailed Description . . . 485
7.49.2 Member Data Documentation . . . 485
7.50 Garfield::MediumGas::ExcLevel Struct Reference . . . 486
7.50.1 Detailed Description . . . 486
7.50.2 Member Data Documentation . . . 486
7.51 Garfield::Geometry Class Reference . . . 487
7.51.1 Detailed Description . . . 487
7.51.2 Constructor & Destructor Documentation . . . 487
7.51.3 Member Function Documentation . . . 488
7.51.4 Member Data Documentation . . . 489
7.52 Garfield::GeometryRoot Class Reference . . . 489
7.52.1 Detailed Description . . . 490
7.52.2 Constructor & Destructor Documentation . . . 490
7.52.3 Member Function Documentation . . . 490
7.52.4 Member Data Documentation . . . 492
7.53 Garfield::GeometrySimple Class Reference . . . 492
7.53.1 Detailed Description . . . 493

7.53.2 Constructor & Destructor Documentation . . . 494
7.53.3 Member Function Documentation . . . 494
7.53.4 Member Data Documentation . . . 496
7.54 Garfield::AvalancheGridSpaceCharge::GridNode Struct Reference . . . 496
7.54.1 Detailed Description . . . 497
7.54.2 Member Data Documentation . . . 497
7.55 Garfield::MediumGas::IonLevel Struct Reference . . . 499
7.55.1 Detailed Description . . . 500
7.55.2 Member Data Documentation . . . 500
7.56 Garfield::KDTree Class Reference . . . 500
7.56.1 Detailed Description . . . 501
7.56.2 Constructor & Destructor Documentation . . . 501
7.56.3 Member Function Documentation . . . 501
7.56.4 Friends And Related Symbol Documentation . . . 503
7.56.5 Member Data Documentation . . . 503
7.57 Garfield::KDTreeNode Class Reference . . . 503
7.57.1 Detailed Description . . . 504
7.57.2 Constructor & Destructor Documentation . . . 504
7.57.3 Member Function Documentation . . . 504
7.57.4 Friends And Related Symbol Documentation . . . 505
7.57.5 Member Data Documentation . . . 505
7.58 Garfield::KDTreeResult Struct Reference . . . 506
7.58.1 Detailed Description . . . 506
7.58.2 Member Data Documentation . . . 506
7.59 Garfield::ComponentFieldMap::Material Struct Reference . . . 506
7.59.1 Detailed Description . . . 507
7.59.2 Member Data Documentation . . . 507
7.60 Garfield::MediumCdTe Class Reference . . . 507
7.60.1 Detailed Description . . . 508
7.60.2 Constructor & Destructor Documentation . . . 508
7.60.3 Member Function Documentation . . . 508
7.60.4 Member Data Documentation . . . 510
7.61 Garfield::MediumConductor Class Reference . . . 511
7.61.1 Detailed Description . . . 511
7.61.2 Constructor & Destructor Documentation . . . 511
7.61.3 Member Function Documentation . . . 511
7.62 Garfield::MediumDiamond Class Reference . . . 512
7.62.1 Detailed Description . . . 513
7.62.2 Constructor & Destructor Documentation . . . 513
7.62.3 Member Function Documentation . . . 513
7.62.4 Member Data Documentation . . . 515
7.63 Garfield::MediumGaAs Class Reference . . . 516
7.63.1 Detailed Description . . . 517

7.63.2 Constructor & Destructor Documentation . . . 517
7.63.3 Member Function Documentation . . . 517
7.63.4 Member Data Documentation . . . 519
7.64 Garfield::MediumGaN Class Reference . . . 520
7.64.1 Detailed Description . . . 521
7.64.2 Constructor & Destructor Documentation . . . 521
7.64.3 Member Function Documentation . . . 521
7.64.4 Member Data Documentation . . . 523
7.65 Garfield::MediumGas Class Reference . . . 524
7.65.1 Detailed Description . . . 527
7.65.2 Constructor & Destructor Documentation . . . 527
7.65.3 Member Function Documentation . . . 527
7.65.4 Member Data Documentation . . . 536
7.66 Garfield::MediumMagboltz Class Reference . . . 538
7.66.1 Detailed Description . . . 545
7.66.2 Constructor & Destructor Documentation . . . 545
7.66.3 Member Function Documentation . . . 546
7.66.4 Member Data Documentation . . . 554
7.67 Garfield::MediumPlastic Class Reference . . . 560
7.67.1 Detailed Description . . . 560
7.67.2 Constructor & Destructor Documentation . . . 560
7.67.3 Member Function Documentation . . . 561
7.68 Garfield::MediumSilicon Class Reference . . . 561
7.68.1 Detailed Description . . . 565
7.68.2 Member Enumeration Documentation . . . 565
7.68.3 Constructor & Destructor Documentation . . . 566
7.68.4 Member Function Documentation . . . 566
7.68.5 Member Data Documentation . . . 575
7.69 Garfield::ComponentFieldMap::Node Struct Reference . . . 581
7.69.1 Detailed Description . . . 581
7.69.2 Member Data Documentation . . . 581
7.70 Garfield::ComponentGrid::Node Struct Reference . . . 581
7.70.1 Detailed Description . . . 581
7.70.2 Member Data Documentation . . . 581
7.71 Garfield::OpticalData Class Reference . . . 582
7.71.1 Detailed Description . . . 583
7.71.2 Constructor & Destructor Documentation . . . 583
7.71.3 Member Function Documentation . . . 583
7.71.4 Member Data Documentation . . . 584
7.72 Garfield::Panel Struct Reference . . . 585
7.72.1 Detailed Description . . . 585
7.72.2 Member Data Documentation . . . 585
7.73 Garfield::TrackElectron::Parameters Struct Reference . . . 586

7.73.1 Detailed Description . . . 586
7.73.2 Member Data Documentation . . . 586
7.74 Garfield::AvalancheGrid::Path Struct Reference . . . 587
7.74.1 Detailed Description . . . 587
7.74.2 Member Data Documentation . . . 587
7.75 Garfield::AvalancheMicroscopic::Photon Struct Reference . . . 588
7.75.1 Detailed Description . . . 588
7.75.2 Member Data Documentation . . . 588
7.76 Garfield::TrackHeed::Photon Struct Reference . . . 589
7.76.1 Detailed Description . . . 589
7.76.2 Member Data Documentation . . . 589
7.77 Garfield::ComponentAnalyticField::Pixel Struct Reference . . . 590
7.77.1 Detailed Description . . . 590
7.77.2 Member Data Documentation . . . 590
7.78 Garfield::ComponentAnalyticField::Plane Struct Reference . . . 591
7.78.1 Detailed Description . . . 592
7.78.2 Member Data Documentation . . . 592
7.79 Garfield::PlottingEngine Class Reference . . . 592
7.79.1 Detailed Description . . . 593
7.79.2 Constructor & Destructor Documentation . . . 593
7.79.3 Member Function Documentation . . . 593
7.79.4 Member Data Documentation . . . 594
7.80 Garfield::AvalancheGridSpaceCharge::Point Struct Reference . . . 594
7.80.1 Detailed Description . . . 594
7.80.2 Member Data Documentation . . . 594
7.81 Garfield::AvalancheMC::Point Struct Reference . . . 595
7.81.1 Detailed Description . . . 595
7.81.2 Member Data Documentation . . . 595
7.82 Garfield::AvalancheMicroscopic::Point Struct Reference . . . 595
7.82.1 Detailed Description . . . 596
7.82.2 Member Data Documentation . . . 596
7.83 Garfield::ComponentNeBem3d::Primitive Struct Reference . . . 597
7.83.1 Detailed Description . . . 597
7.83.2 Member Data Documentation . . . 597
7.84 Garfield::QuadTree Class Reference . . . 599
7.84.1 Detailed Description . . . 599
7.84.2 Constructor & Destructor Documentation . . . 599
7.84.3 Member Function Documentation . . . 600
7.84.4 Member Data Documentation . . . 600
7.85 Garfield::Random Class Reference . . . 602
7.85.1 Detailed Description . . . 602
7.85.2 Constructor & Destructor Documentation . . . 602
7.85.3 Member Function Documentation . . . 602

7.85.4 Member Data Documentation . . . 602
7.86 Garfield::RandomEngine< Engine, SeedType > Class Template Reference . . . 602
7.86.1 Detailed Description . . . 603
7.86.2 Member Typedef Documentation . . . 603
7.86.3 Constructor & Destructor Documentation . . . 603
7.86.4 Member Function Documentation . . . 604
7.86.5 Member Data Documentation . . . 604
7.87 Garfield::RandomEngineRoot Class Reference . . . 604
7.87.1 Detailed Description . . . 605
7.87.2 Constructor & Destructor Documentation . . . 605
7.87.3 Member Function Documentation . . . 606
7.87.4 Member Data Documentation . . . 606
7.88 Garfield::RandomEngineSTL Class Reference . . . 606
7.88.1 Detailed Description . . . 607
7.88.2 Constructor & Destructor Documentation . . . 607
7.88.3 Member Function Documentation . . . 607
7.88.4 Member Data Documentation . . . 608
7.89 Garfield::ComponentComsol::Range Struct Reference . . . 608
7.89.1 Detailed Description . . . 608
7.89.2 Member Data Documentation . . . 608
7.90 Garfield::ComponentNeBem2d::Region Struct Reference . . . 609
7.90.1 Detailed Description . . . 609
7.90.2 Member Data Documentation . . . 609
7.91 Garfield::ComponentTcadBase< N >::Region Struct Reference . . . 610
7.91.1 Detailed Description . . . 610
7.91.2 Member Data Documentation . . . 610
7.92 Garfield::AvalancheMC::Seed Struct Reference . . . 611
7.92.1 Detailed Description . . . 611
7.92.2 Member Data Documentation . . . 611
7.93 Garfield::AvalancheMicroscopic::Seed Struct Reference . . . 611
7.93.1 Detailed Description . . . 612
7.93.2 Member Data Documentation . . . 612
7.94 Garfield::ComponentNeBem2d::Segment Struct Reference . . . 612
7.94.1 Detailed Description . . . 612
7.94.2 Member Data Documentation . . . 613
7.95 Garfield::Shaper Class Reference . . . 613
7.95.1 Detailed Description . . . 614
7.95.2 Member Enumeration Documentation . . . 614
7.95.3 Constructor & Destructor Documentation . . . 614
7.95.4 Member Function Documentation . . . 615
7.95.5 Member Data Documentation . . . 616
7.96 Garfield::Solid Class Reference . . . 616
7.96.1 Detailed Description . . . 620

7.96.2 Member Enumeration Documentation . . . 620
7.96.3 Constructor & Destructor Documentation . . . 620
7.96.4 Member Function Documentation . . . 621
7.96.5 Member Data Documentation . . . 628
7.97 Garfield::SolidBox Class Reference . . . 630
7.97.1 Detailed Description . . . 633
7.97.2 Constructor & Destructor Documentation . . . 634
7.97.3 Member Function Documentation . . . 634
7.97.4 Member Data Documentation . . . 636
7.98 Garfield::SolidExtrusion Class Reference . . . 636
7.98.1 Detailed Description . . . 640
7.98.2 Constructor & Destructor Documentation . . . 640
7.98.3 Member Function Documentation . . . 641
7.98.4 Member Data Documentation . . . 643
7.99 Garfield::SolidHole Class Reference . . . 643
7.99.1 Detailed Description . . . 648
7.99.2 Constructor & Destructor Documentation . . . 648
7.99.3 Member Function Documentation . . . 648
7.99.4 Member Data Documentation . . . 651
7.100 Garfield::SolidRidge Class Reference . . . 652
7.100.1 Detailed Description . . . 656
7.100.2 Constructor & Destructor Documentation . . . 656
7.100.3 Member Function Documentation . . . 657
7.100.4 Member Data Documentation . . . 659
7.101 Garfield::SolidSphere Class Reference . . . 660
7.101.1 Detailed Description . . . 663
7.101.2 Constructor & Destructor Documentation . . . 664
7.101.3 Member Function Documentation . . . 664
7.101.4 Member Data Documentation . . . 666
7.102 Garfield::SolidTube Class Reference . . . 667
7.102.1 Detailed Description . . . 671
7.102.2 Constructor & Destructor Documentation . . . 672
7.102.3 Member Function Documentation . . . 673
7.102.4 Member Data Documentation . . . 676
7.103 Garfield::SolidWire Class Reference . . . 677
7.103.1 Detailed Description . . . 681
7.103.2 Constructor & Destructor Documentation . . . 681
7.103.3 Member Function Documentation . . . 682
7.103.4 Member Data Documentation . . . 683
7.104 Garfield::ComponentNeBem2d::SpaceCharge Struct Reference . . . 683
7.104.1 Detailed Description . . . 684
7.104.2 Member Data Documentation . . . 684
7.105 Garfield::AvalancheMicroscopic::Statistics Struct Reference . . . 684

7.105.1 Detailed Description . . . 685
7.105.2 Member Data Documentation . . . 685
7.106 Garfield::ComponentAnalyticField::Strip Struct Reference . . . 685
7.106.1 Detailed Description . . . 686
7.106.2 Member Data Documentation . . . 686
7.107 TGeoTet Class Reference . . . 686
7.107.1 Detailed Description . . . 687
7.107.2 Constructor & Destructor Documentation . . . 687
7.107.3 Member Function Documentation . . . 687
7.107.4 Member Data Documentation . . . 689
7.108 Garfield::Track Class Reference . . . 689
7.108.1 Detailed Description . . . 691
7.108.2 Constructor & Destructor Documentation . . . 691
7.108.3 Member Function Documentation . . . 692
7.108.4 Member Data Documentation . . . 695
7.109 Garfield::TrackBichsel Class Reference . . . 696
7.109.1 Detailed Description . . . 699
7.109.2 Constructor & Destructor Documentation . . . 699
7.109.3 Member Function Documentation . . . 699
7.109.4 Member Data Documentation . . . 700
7.110 Garfield::TrackDegrade Class Reference . . . 702
7.110.1 Detailed Description . . . 704
7.110.2 Constructor & Destructor Documentation . . . 704
7.110.3 Member Function Documentation . . . 705
7.110.4 Member Data Documentation . . . 707
7.111 Garfield::TrackElectron Class Reference . . . 708
7.111.1 Detailed Description . . . 710
7.111.2 Constructor & Destructor Documentation . . . 710
7.111.3 Member Function Documentation . . . 711
7.111.4 Member Data Documentation . . . 712
7.112 Garfield::TrackHeed Class Reference . . . 712
7.112.1 Detailed Description . . . 717
7.112.2 Constructor & Destructor Documentation . . . 717
7.112.3 Member Function Documentation . . . 717
7.112.4 Member Data Documentation . . . 726
7.113 Garfield::TrackPAI Class Reference . . . 729
7.113.1 Detailed Description . . . 732
7.113.2 Constructor & Destructor Documentation . . . 732
7.113.3 Member Function Documentation . . . 732
7.113.4 Member Data Documentation . . . 734
7.114 Garfield::TrackSimple Class Reference . . . 735
7.114.1 Detailed Description . . . 738
7.114.2 Constructor & Destructor Documentation . . . 738

7.114.3 Member Function Documentation . . . 738
7.114.4 Member Data Documentation . . . 739
7.115 Garfield::TrackSrim Class Reference . . . 739
7.115.1 Detailed Description . . . 743
7.115.2 Constructor & Destructor Documentation . . . 744
7.115.3 Member Function Documentation . . . 744
7.115.4 Member Data Documentation . . . 748
7.116 Garfield::TrackTrim Class Reference . . . 751
7.116.1 Detailed Description . . . 754
7.116.2 Constructor & Destructor Documentation . . . 754
7.116.3 Member Function Documentation . . . 754
7.116.4 Member Data Documentation . . . 757
7.117 Garfield::Vec1Impl< type > Class Template Reference . . . 758
7.117.1 Detailed Description . . . 758
7.117.2 Constructor & Destructor Documentation . . . 758
7.117.3 Member Function Documentation . . . 759
7.118 Garfield::Vec2Impl< type > Class Template Reference . . . 759
7.118.1 Detailed Description . . . 759
7.118.2 Constructor & Destructor Documentation . . . 760
7.118.3 Member Function Documentation . . . 760
7.119 Garfield::Vec3Impl< type > Class Template Reference . . . 760
7.119.1 Detailed Description . . . 761
7.119.2 Constructor & Destructor Documentation . . . 761
7.119.3 Member Function Documentation . . . 761
7.120 Garfield::Vector< type, Dimension > Class Template Reference . . . 762
7.120.1 Detailed Description . . . 763
7.120.2 Constructor & Destructor Documentation . . . 763
7.120.3 Member Function Documentation . . . 763
7.121 Garfield::ViewBase Class Reference . . . 764
7.121.1 Detailed Description . . . 766
7.121.2 Constructor & Destructor Documentation . . . 766
7.121.3 Member Function Documentation . . . 767
7.121.4 Member Data Documentation . . . 771
7.122 Garfield::ViewCell Class Reference . . . 773
7.122.1 Detailed Description . . . 775
7.122.2 Constructor & Destructor Documentation . . . 775
7.122.3 Member Function Documentation . . . 776
7.122.4 Member Data Documentation . . . 778
7.123 Garfield::ViewDrift Class Reference . . . 778
7.123.1 Detailed Description . . . 782
7.123.2 Constructor & Destructor Documentation . . . 782
7.123.3 Member Function Documentation . . . 782
7.123.4 Friends And Related Symbol Documentation . . . 786

7.123.5 Member Data Documentation . . . 786
7.124 Garfield::ViewFEMesh Class Reference . . . 788
7.124.1 Detailed Description . . . 792
7.124.2 Member Typedef Documentation . . . 792
7.124.3 Constructor & Destructor Documentation . . . 792
7.124.4 Member Function Documentation . . . 792
7.124.5 Member Data Documentation . . . 797
7.125 Garfield::ViewField Class Reference . . . 799
7.125.1 Detailed Description . . . 803
7.125.2 Member Enumeration Documentation . . . 803
7.125.3 Constructor & Destructor Documentation . . . 803
7.125.4 Member Function Documentation . . . 804
7.125.5 Member Data Documentation . . . 809
7.126 Garfield::ViewGeometry Class Reference . . . 810
7.126.1 Detailed Description . . . 813
7.126.2 Constructor & Destructor Documentation . . . 813
7.126.3 Member Function Documentation . . . 813
7.126.4 Member Data Documentation . . . 814
7.127 Garfield::ViewIsochrons Class Reference . . . 814
7.127.1 Detailed Description . . . 817
7.127.2 Constructor & Destructor Documentation . . . 817
7.127.3 Member Function Documentation . . . 817
7.127.4 Member Data Documentation . . . 819
7.128 Garfield::ViewMedium Class Reference . . . 820
7.128.1 Detailed Description . . . 825
7.128.2 Member Enumeration Documentation . . . 825
7.128.3 Constructor & Destructor Documentation . . . 826
7.128.4 Member Function Documentation . . . 827
7.128.5 Member Data Documentation . . . 834
7.129 Garfield::ViewSignal Class Reference . . . 837
7.129.1 Detailed Description . . . 840
7.129.2 Constructor & Destructor Documentation . . . 840
7.129.3 Member Function Documentation . . . 841
7.129.4 Member Data Documentation . . . 843
7.130 Garfield::ComponentFieldMap::WeightingFieldCopy Struct Reference . . . 844
7.130.1 Detailed Description . . . 844
7.130.2 Member Data Documentation . . . 844
7.131 Garfield::ComponentAnalyticField::Wire Struct Reference . . . 845
7.131.1 Detailed Description . . . 845
7.131.2 Member Data Documentation . . . 845
7.132 Garfield::ComponentNeBem2d::Wire Struct Reference . . . 846
7.132.1 Detailed Description . . . 847
7.132.2 Member Data Documentation . . . 847

8 File Documentation 847
8.1 /builds/garfield/docs/source/Include/Garfield/AvalancheGrid.hh File Reference . . . . . 847
8.2 AvalancheGrid.hh . . . . . 848
8.3 /builds/garfield/docs/source/Include/Garfield/AvalancheGridSpaceCharge.hh File Reference 849
8.4 AvalancheGridSpaceCharge.hh . . . . . 850
8.5 /builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh File Reference . . . . . 853
8.6 AvalancheMC.hh . . . . . 853
8.7 /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh File Reference . . . 856
8.8 AvalancheMicroscopic.hh . . . . . 857
8.9 /builds/garfield/docs/source/Include/Garfield/Component.hh File Reference . . . . . 861
8.10 Component.hh . . . . . 861
8.11 /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh File Reference 866
8.12 ComponentAnalyticField.hh . . . . . 866
8.13 /builds/garfield/docs/source/Include/Garfield/ComponentAnsys121.hh File Reference . . 874
8.14 ComponentAnsys121.hh . . . . . 874
8.15 /builds/garfield/docs/source/Include/Garfield/ComponentAnsys123.hh File Reference . . 875
8.16 ComponentAnsys123.hh . . . . . 875
8.17 /builds/garfield/docs/source/Include/Garfield/ComponentComsol.hh File Reference . . . 875
8.18 ComponentComsol.hh . . . . . 876
8.19 /builds/garfield/docs/source/Include/Garfield/ComponentConstant.hh File Reference . . . 877
8.20 ComponentConstant.hh . . . . . 877
8.21 /builds/garfield/docs/source/Include/Garfield/ComponentCST.hh File Reference . . . . . 878
8.22 ComponentCST.hh . . . . . 879
8.23 /builds/garfield/docs/source/Include/Garfield/ComponentElmer.hh File Reference . . . . 881
8.24 ComponentElmer.hh . . . . . 881
8.25 /builds/garfield/docs/source/Include/Garfield/ComponentElmer2d.hh File Reference . . . 881
8.26 ComponentElmer2d.hh . . . . . 882
8.27 /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh File Reference . . 882
8.28 ComponentFieldMap.hh . . . . . 883
8.29 /builds/garfield/docs/source/Include/Garfield/ComponentGrid.hh File Reference . . . . . 888
8.30 ComponentGrid.hh . . . . . 889
8.31 /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh File Reference . . 891
8.32 ComponentNeBem2d.hh . . . . . 891
8.33 /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3d.hh File Reference . . 893
8.34 ComponentNeBem3d.hh . . . . . 894
8.35 /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3dMap.hh File Reference 898
8.36 ComponentNeBem3dMap.hh . . . . . 898
8.37 /builds/garfield/docs/source/Include/Garfield/ComponentParallelPlate.hh File Reference 899
8.38 ComponentParallelPlate.hh . . . . . 900
8.39 /builds/garfield/docs/source/Include/Garfield/ComponentTcad2d.hh File Reference . . . . 903
8.40 ComponentTcad2d.hh . . . . . 903
8.41 /builds/garfield/docs/source/Include/Garfield/ComponentTcad3d.hh File Reference . . . . 904
8.42 ComponentTcad3d.hh . . . . . 905

8.43 /builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh File Reference . . 906
8.44 ComponentTcadBase.hh . . . . . 906
8.45 /builds/garfield/docs/source/Include/Garfield/ComponentUser.hh File Reference . . . . . 909
8.46 ComponentUser.hh . . . . . 910
8.47 /builds/garfield/docs/source/Include/Garfield/ComponentVoxel.hh File Reference . . . . . 911
8.48 ComponentVoxel.hh . . . . . 912
8.49 /builds/garfield/docs/source/Include/Garfield/DegradeInterface.hh File Reference . . . . . 913
8.50 DegradeInterface.hh . . . . . 914
8.51 /builds/garfield/docs/source/Include/Garfield/DriftLineRKF.hh File Reference . . . . . 915
8.52 DriftLineRKF.hh . . . . . 915
8.53 /builds/garfield/docs/source/Include/Garfield/FundamentalConstants.hh File Reference . 918
8.54 FundamentalConstants.hh . . . . . 918
8.55 /builds/garfield/docs/source/Include/Garfield/GarfieldConstants.hh File Reference . . . . 919
8.56 GarfieldConstants.hh . . . . . 919
8.57 /builds/garfield/docs/source/Include/Garfield/Geometry.hh File Reference . . . . . 920
8.58 Geometry.hh . . . . . 920
8.59 /builds/garfield/docs/source/Include/Garfield/GeometryRoot.hh File Reference . . . . . 920
8.60 GeometryRoot.hh . . . . . 921
8.61 /builds/garfield/docs/source/Include/Garfield/GeometrySimple.hh File Reference . . . . . 921
8.62 GeometrySimple.hh . . . . . 922
8.63 /builds/garfield/docs/source/Include/Garfield/HelperMacros.hh File Reference . . . . . 922
8.63.1 Macro Definition Documentation . . . . . 923
8.64 HelperMacros.hh . . . . . 923
8.65 /builds/garfield/docs/source/Include/Garfield/KDTree.hh File Reference . . . . . 923
8.66 KDTree.hh . . . . . 923
8.67 /builds/garfield/docs/source/Include/Garfield/LinkDef.h File Reference . . . . . 925
8.68 LinkDef.h . . . . . 925
8.69 /builds/garfield/docs/source/Include/Garfield/MagboltzInterface.hh File Reference . . . . 926
8.70 MagboltzInterface.hh . . . . . 931
8.71 /builds/garfield/docs/source/Include/Garfield/Medium.hh File Reference . . . . . 933
8.72 Medium.hh . . . . . 934
8.73 /builds/garfield/docs/source/Include/Garfield/MediumCdTe.hh File Reference . . . . . 942
8.74 MediumCdTe.hh . . . . . 942
8.75 /builds/garfield/docs/source/Include/Garfield/MediumConductor.hh File Reference . . . . 943
8.76 MediumConductor.hh . . . . . 943
8.77 /builds/garfield/docs/source/Include/Garfield/MediumDiamond.hh File Reference . . . . 944
8.78 MediumDiamond.hh . . . . . 944
8.79 /builds/garfield/docs/source/Include/Garfield/MediumGaAs.hh File Reference . . . . . 945
8.80 MediumGaAs.hh . . . . . 945
8.81 /builds/garfield/docs/source/Include/Garfield/MediumGaN.hh File Reference . . . . . 946
8.82 MediumGaN.hh . . . . . 946
8.83 /builds/garfield/docs/source/Include/Garfield/MediumGas.hh File Reference . . . . . 947
8.84 MediumGas.hh . . . . . 947

8.85 /builds/garfield/docs/source/Include/Garfield/MediumMagboltz.hh File Reference . . . . 950
8.86 MediumMagboltz.hh . . . . . 951
8.87 /builds/garfield/docs/source/Include/Garfield/MediumPlastic.hh File Reference . . . . . 955
8.88 MediumPlastic.hh . . . . . 955
8.89 /builds/garfield/docs/source/Include/Garfield/MediumSilicon.hh File Reference . . . . . 955
8.90 MediumSilicon.hh . . . . . 956
8.91 /builds/garfield/docs/source/Include/Garfield/MultiProcessInterface.hh File Reference . . 959
8.92 MultiProcessInterface.hh . . . . . 960
8.93 /builds/garfield/docs/source/Include/Garfield/Numerics.hh File Reference . . . . . 960
8.94 Numerics.hh . . . . . 962
8.95 /builds/garfield/docs/source/Include/Garfield/OpticalData.hh File Reference . . . . . 964
8.96 OpticalData.hh . . . . . 964
8.97 /builds/garfield/docs/source/Include/Garfield/Plotting.hh File Reference . . . . . 965
8.98 Plotting.hh . . . . . 965
8.99 /builds/garfield/docs/source/Include/Garfield/PlottingEngine.hh File Reference . . . . . 965
8.100 PlottingEngine.hh . . . . . 965
8.101 /builds/garfield/docs/source/Include/Garfield/Polygon.hh File Reference . . . . . 966
8.102 Polygon.hh . . . . . 966
8.103 /builds/garfield/docs/source/Include/Garfield/QuadTree.hh File Reference . . . . . 967
8.104 QuadTree.hh . . . . . 967
8.105 /builds/garfield/docs/source/Include/Garfield/Random.hh File Reference . . . . . 968
8.106 Random.hh . . . . . 969
8.107 /builds/garfield/docs/source/Include/Garfield/RandomEngine.hh File Reference . . . . . 970
8.108 RandomEngine.hh . . . . . 970
8.109 /builds/garfield/docs/source/Include/Garfield/RandomEngineRoot.hh File Reference . . 971
8.110 RandomEngineRoot.hh . . . . . 971
8.111 /builds/garfield/docs/source/Include/Garfield/RandomEngineSTL.hh File Reference . . 971
8.112 RandomEngineSTL.hh . . . . . 972
8.113 /builds/garfield/docs/source/Include/Garfield/Sensor.hh File Reference . . . . . 972
8.113.1 Macro Definition Documentation . . . . . 972
8.114 Sensor.hh . . . . . 972
8.115 /builds/garfield/docs/source/Include/Garfield/Shaper.hh File Reference . . . . . 977
8.116 Shaper.hh . . . . . 977
8.117 /builds/garfield/docs/source/Include/Garfield/Solid.hh File Reference . . . . . 977
8.118 Solid.hh . . . . . 978
8.119 /builds/garfield/docs/source/Include/Garfield/SolidBox.hh File Reference . . . . . 980
8.120 SolidBox.hh . . . . . 980
8.121 /builds/garfield/docs/source/Include/Garfield/SolidExtrusion.hh File Reference . . . . . 981
8.122 SolidExtrusion.hh . . . . . 981
8.123 /builds/garfield/docs/source/Include/Garfield/SolidHole.hh File Reference . . . . . 982
8.124 SolidHole.hh . . . . . 982
8.125 /builds/garfield/docs/source/Include/Garfield/SolidRidge.hh File Reference . . . . . 983
8.126 SolidRidge.hh . . . . . 983

8.127 /builds/garfield/docs/source/Include/Garfield/SolidSphere.hh File Reference . . . . . 984
8.128 SolidSphere.hh . . . . . 984
8.129 /builds/garfield/docs/source/Include/Garfield/SolidTube.hh File Reference . . . . . 985
8.130 SolidTube.hh . . . . . 985
8.131 /builds/garfield/docs/source/Include/Garfield/SolidWire.hh File Reference . . . . . 986
8.132 SolidWire.hh . . . . . 987
8.133 /builds/garfield/docs/source/Include/Garfield/TetrahedralTree.hh File Reference . . . . 987
8.134 TetrahedralTree.hh . . . . . 988
8.135 /builds/garfield/docs/source/Include/Garfield/TGeoTet.hh File Reference . . . . . 989
8.136 TGeoTet.hh . . . . . 989
8.137 /builds/garfield/docs/source/Include/Garfield/Track.hh File Reference . . . . . 990
8.138 Track.hh . . . . . 990
8.139 /builds/garfield/docs/source/Include/Garfield/TrackBichsel.hh File Reference . . . . . 991
8.140 TrackBichsel.hh . . . . . 991
8.141 /builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh File Reference . . . . . 992
8.142 TrackDegrade.hh . . . . . 993
8.143 /builds/garfield/docs/source/Include/Garfield/TrackElectron.hh File Reference . . . . . 994
8.144 TrackElectron.hh . . . . . 994
8.145 /builds/garfield/docs/source/Include/Garfield/TrackHeed.hh File Reference . . . . . 995
8.146 TrackHeed.hh . . . . . 995
8.147 /builds/garfield/docs/source/Include/Garfield/TrackPAI.hh File Reference . . . . . 998
8.148 TrackPAI.hh . . . . . 998
8.149 /builds/garfield/docs/source/Include/Garfield/TrackSimple.hh File Reference . . . . . 999
8.150 TrackSimple.hh . . . . . 1000
8.151 /builds/garfield/docs/source/Include/Garfield/TrackSrim.hh File Reference . . . . . 1000
8.152 TrackSrim.hh . . . . . 1001
8.153 /builds/garfield/docs/source/Include/Garfield/TrackTrim.hh File Reference . . . . . 1002
8.154 TrackTrim.hh . . . . . 1002
8.155 /builds/garfield/docs/source/Include/Garfield/Utilities.hh File Reference . . . . . 1003
8.156 Utilities.hh . . . . . 1004
8.157 /builds/garfield/docs/source/Include/Garfield/Vector.hh File Reference . . . . . 1004
8.157.1 Macro Definition Documentation . . . . . 1005
8.157.2 Typedef Documentation . . . . . 1005
8.158 Vector.hh . . . . . 1005
8.159 /builds/garfield/docs/source/Include/Garfield/ViewBase.hh File Reference . . . . . 1006
8.160 ViewBase.hh . . . . . 1007
8.161 /builds/garfield/docs/source/Include/Garfield/ViewCell.hh File Reference . . . . . 1008
8.162 ViewCell.hh . . . . . 1009
8.163 /builds/garfield/docs/source/Include/Garfield/ViewDrift.hh File Reference . . . . . 1009
8.164 ViewDrift.hh . . . . . 1010
8.165 /builds/garfield/docs/source/Include/Garfield/ViewFEMesh.hh File Reference . . . . . 1011
8.166 ViewFEMesh.hh . . . . . 1011
8.167 /builds/garfield/docs/source/Include/Garfield/ViewField.hh File Reference . . . . . 1013

8.168 ViewField.hh . . . 1013
8.169 /builds/garfield/docs/source/Include/Garfield/ViewGeometry.hh File Reference . . . 1015
8.170 ViewGeometry.hh . . . 1015
8.171 /builds/garfield/docs/source/Include/Garfield/ViewIsochrons.hh File Reference . . . 1016
8.172 ViewIsochrons.hh . . . 1016
8.173 /builds/garfield/docs/source/Include/Garfield/ViewMedium.hh File Reference . . . 1017
8.174 ViewMedium.hh . . . 1017
8.175 /builds/garfield/docs/source/Include/Garfield/ViewSignal.hh File Reference . . . 1020
8.176 ViewSignal.hh . . . 1020
8.177 /builds/garfield/docs/source/README.md File Reference . . . 1021

# 1 Garfield++

Garfield++ is a toolkit for the detailed simulation of detectors which use gases or semi-conductors as sensitive medium. The main area of application is currently in micropattern gaseous detectors.

Garfield++ shares functionality with Garfield. The main differences are the more up-to-date treatment of electron transport in gases, the inclusion of semiconductor detectors and the user interface, which is derived from ROOT.

More...

## 1.1 Building the project

### 1.1.1 Options

GARFIELD_WITH_CUDA: Build CUDA code if possible OFF by default

GARFIELD_CUDA_USE_FLOAT: Use float with CUDA code (for some performances issues on some hardware)" <tt>double</tt> by default <tt>GARFIELD_WITH_DEGRADE</tt>: Build Degrade <tt>ON</tt> by default <tt>GARFIELD_WITH_EXAMPLES</tt>: Build Garfield++ examples <tt>ON</tt> by default <tt>GARFIELD_WITH_TESTS</tt>: Build Garfield++ tests <tt>ON</tt> by default For simplicity, we define an environment variable <tt>\$GARFIELD_↩ HOME</tt> pointing to the directory to which we cloned the repository. Assuming that <tt>\$↩ GARFIELD_HOME</tt> is your current working directory, you can build and install Garfield++ as follows: @icode{bash} mkdir build cd build cmake [-DCMAKE_INSTALL_PREFIX=<installdir>] [-↩ DGARFIELD_WITH_DOC=ON] [-DGARFIELD_WITH_EXAMPLES=ON] <path to sources> make -j<number of cores on your machine> make install @endicode If <tt>CMAKE_INSTALL_PREFIX</tt> is not provided in the configuration command, <tt>\$GARFIELD_HOME/install</tt> will be used as installation prefix. The <tt>GARFIELD_WITH_DOC</tt> variable is optional, and should be passed if you wish to build the Doxygen based API documentation. Please note that this requires an existing installation of <a href="http://www.doxygen.org/index.html" >Doxygen</a>. If CMake cannot locate Doxygen, its install location should be added into <tt>CMAKE_PREFIX_PATH</tt>. For further details please have a look at <a href="http://www.cmake.org/cmake-tutorial/" >the CMake tutorial.

## 1.2 Building the documentation

The documentation of the project is based on doxygen. To build the documentation, the project must have been configured with GARFIELD_WITH_DOC enabled, as described earlier. It can then be built and installed:

```
make doc
make install
```

By default, this installs the documentation into <installdir>/share/doc/HSFTEMPLATE/share/doc.

### 1.3 Running the examples

You can run the examples from the build directory (if WITH_EXAMPLES has been turned on) but you need to setup a running environment defining some variables, in particular for the HEED database.

In the following lines we use the GEM example (assuming that the build folder is our current working directory):

```
source setupGarfield.sh
cd Examples/Gem
./gem
```

Alternatively, thanks to PyROOT, you can use the Garfield++ classes from Python`. Assuming again that the build folder is our current working directory we can run the GEM example like this:

```
source setupGarfield.sh
cd Examples/Gem
python3 -i gem.py
```

### 1.4 Building and running examples using an installed version of Garfield

Make sure that all required environment variables are set by sourcing the script setupGarfield.sh:

```
source $GARFIELD_HOME/install/share/Garfield/setupGarfield.sh
```

To get started, it can be useful to copy one of the examples to a local directory, modify it, build it against an installed version of Garfield and run it.

```
cp -r $GARFIELD_HOME/Examples/Gem .
mkdir Gem/build; cd Gem/build
cmake ..
make
./gem
```

### 1.5 Inclusion into other projects

If you want to build your own project against Garfield, CMake may be the best option for you. Just add its location to CMAKE_PREFIX_PATH and call find_package(Garfield) within your CMakeLists.txt.

```
cmake_minimum_required(VERSION 3.12 FATAL_ERROR)
project(test)

if(NOT TARGET Garfield::Garfield)
  find_package(Garfield REQUIRED)
endif()

add_executable(test test.C)
target_link_libraries(test Garfield::Garfield)
```

## 2 Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Garfield 16

Garfield::Degrade 56

Garfield::Magboltz 61

Garfield::Numerics
Collection of numerical routines 86

Garfield::Numerics::CERNLIB
Linear algebra routines from CERNLIB 92

Garfield::Numerics::QUADPACK
Functions for performing numerical integration (quadrature) 95

Garfield::Polygon 98

Heed 99

# 3 Hierarchical Index

## 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

std::array< T, D >

Garfield::Vector< type, 1 > 762

Garfield::Vec1Impl< type > 758

Garfield::Vector< type, 2 > 762

Garfield::Vec2Impl< type > 759

Garfield::Vector< type, 3 > 762

Garfield::Vec3Impl< type > 760

Garfield::Vector< type, Dimension > 762

Garfield::Vec3Impl< double > 760

Garfield::Vec3Impl< float > 760

Garfield::AvalancheGrid 99

Garfield::AvalancheGridSpaceCharge 108

Garfield::AvalancheMC 128

Garfield::AvalancheMicroscopic 154

Garfield::AvalancheGrid::AvalancheNode 186

Garfield::MediumSilicon::Band 188

Garfield::ComponentAnalyticField::Charge3d 191

Garfield::TrackBichsel::Cluster 192

Garfield::TrackDegrade::Cluster 193

Garfield::TrackElectron::Cluster 195

Garfield::TrackHeed::Cluster 196

Garfield::TrackPAI::Cluster 198

Garfield::TrackSimple::Cluster 199

Garfield::TrackSrim::Cluster 200

Garfield::TrackTrim::Cluster 202

Component

Garfield::ComponentTcadBase< 2 > 422

Garfield::ComponentTcad2d 407

Garfield::ComponentTcadBase< 3 > 422

Garfield::ComponentTcad3d 414

Garfield::ComponentAnalyticField 203

Garfield::ComponentConstant 284

Garfield::ComponentFieldMap 318

Garfield::ComponentAnsys121 261

Garfield::ComponentAnsys123 268

Garfield::ComponentCST 291

Garfield::ComponentComsol 274

Garfield::ComponentElmer 306

Garfield::ComponentElmer2d 312

Garfield::ComponentGrid 338

Garfield::ComponentNeBem2d 354

Garfield::ComponentNeBem3d 366

Garfield::ComponentNeBem3dMap 386

Garfield::ComponentParallelPlate 395

Garfield::ComponentTcadBase< N > 422

Garfield::ComponentUser 439

Garfield::ComponentVoxel 446

Garfield::MediumMagboltz::Deexcitation 457

Garfield::ComponentTcadBase< N >::Defect 459

Garfield::DriftLineRKF 459

Garfield::ComponentParallelPlate::Electrode 474

Garfield::AvalancheMicroscopic::Electron 476

Garfield::TrackDegrade::Electron 476

Garfield::TrackHeed::Electron 477

Garfield::ComponentFieldMap::Element 478

Garfield::ComponentNeBem2d::Element 479

Garfield::ComponentNeBem3d::Element 480

Garfield::ComponentNeBem3dMap::Element 482

Garfield::ComponentTcadBase< N >::Element 483

Garfield::ComponentVoxel::Element 484

Garfield::AvalancheMC::EndPoint 484

Garfield::TrackDegrade::Excitation 485

Garfield::MediumGas::ExcLevel 486

Garfield::Geometry 487

- Garfield::GeometryRoot 489
- Garfield::GeometrySimple 492

Garfield::AvalancheGridSpaceCharge::GridNode 496

Garfield::MediumGas::IonLevel 499

Garfield::KDTree 500

Garfield::KDTreeNode 503

Garfield::KDTreeResult 506

Garfield::ComponentFieldMap::Material 506
Medium

- Garfield::MediumCdTe 507
- Garfield::MediumConductor 511
- Garfield::MediumDiamond 512
- Garfield::MediumGaAs 516
- Garfield::MediumGaN 520
- Garfield::MediumGas 524
  - Garfield::MediumMagboltz 538
- Garfield::MediumPlastic 560
- Garfield::MediumSilicon 561

Garfield::ComponentFieldMap::Node 581

Garfield::ComponentGrid::Node 581

Garfield::OpticalData 582

Garfield::Panel 585

Garfield::TrackElectron::Parameters 586

Garfield::AvalancheGrid::Path 587

Garfield::AvalancheMicroscopic::Photon 588

Garfield::TrackHeed::Photon 589

Garfield::ComponentAnalyticField::Pixel 590

Garfield::ComponentAnalyticField::Plane 591

Garfield::PlottingEngine 592

Garfield::AvalancheGridSpaceCharge::Point 594

Garfield::AvalancheMC::Point 595

Garfield::AvalancheMicroscopic::Point 595

Garfield::ComponentNeBem3d::Primitive 597

Garfield::QuadTree 599

Garfield::Random 602

Garfield::RandomEngine< Engine, SeedType > 602

Garfield::RandomEngine< RandomEngineRoot, UInt_t > 602

Garfield::RandomEngineRoot 604

Garfield::RandomEngine< RandomEngineSTL, std::mt19937::result_type > 602

Garfield::RandomEngineSTL 606

Garfield::ComponentComsol::Range 608

Garfield::ComponentNeBem2d::Region 609

Garfield::ComponentTcadBase< N >::Region 610

Garfield::AvalancheMC::Seed 611

Garfield::AvalancheMicroscopic::Seed 611

Garfield::ComponentNeBem2d::Segment 612

Garfield::Shaper 613

Garfield::Solid 616

Garfield::SolidBox 630

Garfield::SolidExtrusion 636

Garfield::SolidHole 643

Garfield::SolidRidge 652

Garfield::SolidSphere 660

Garfield::SolidTube 667

Garfield::SolidWire 677

Garfield::ComponentNeBem2d::SpaceCharge 683

Garfield::AvalancheMicroscopic::Statistics 684

Garfield::ComponentAnalyticField::Strip 685
TGeoBBox

TGeoTet 686

Garfield::Track 689

Garfield::TrackBichsel 696

Garfield::TrackDegrade 702

Garfield::TrackElectron 708

Garfield::TrackHeed 712

Garfield::TrackPAI 729

Garfield::TrackSimple 735

Garfield::TrackSrim 739

Garfield::TrackTrim 751

Garfield::ViewBase 764

Garfield::ViewCell 773

Garfield::ViewDrift 778

Garfield::ViewFEMesh 788

Garfield::ViewField 799

Garfield::ViewGeometry 810

Garfield::ViewIsochrons 814

Garfield::ViewMedium 820

Garfield::ViewSignal 837

Garfield::ComponentFieldMap::WeightingFieldCopy 844

Garfield::ComponentAnalyticField::Wire 845

Garfield::ComponentNeBem2d::Wire 846

# 4 Class Index

## 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


Garfield::AvalancheGrid
Calculate avalanches in a uniform electric field using avalanche statistics 99

Garfield::AvalancheGridSpaceCharge
Propagates avalanches with the 2d (axi-symmetric) space-charge routine from Lippmann, Riegler (2004) in uniform background fields 108

Garfield::AvalancheMC
Calculate drift lines and avalanches based on macroscopic transport coefficients, using Monte Carlo integration 128

Garfield::AvalancheMicroscopic
Calculate electron drift lines and avalanches using microscopic tracking 154

Garfield::AvalancheGrid::AvalancheNode 186

Garfield::MediumSilicon::Band 188

Garfield::ComponentAnalyticField::Charge3d 191

Garfield::TrackBichsel::Cluster 192

Garfield::TrackDegrade::Cluster 193

Garfield::TrackElectron::Cluster 195

Garfield::TrackHeed::Cluster 196

Garfield::TrackPAI::Cluster 198

Garfield::TrackSimple::Cluster 199

Garfield::TrackSrim::Cluster 200

Garfield::TrackTrim::Cluster 202

Garfield::ComponentAnalyticField
Semi-analytic calculation of two-dimensional configurations consisting of wires, planes, and tubes 203

Garfield::ComponentAnsys121
Component for importing and interpolating two-dimensional ANSYS field maps 261

Garfield::ComponentAnsys123
Component for importing and interpolating three-dimensional ANSYS field maps 268

Garfield::ComponentComsol
Component for importing and interpolating Comsol field maps 274

Garfield::ComponentConstant
Component with constant electric field 284

Garfield::ComponentCST
Component for importing and interpolating field maps from CST 291

Garfield::ComponentElmer
Component for importing field maps computed by Elmer 306

Garfield::ComponentElmer2d
Component for importing two-dimensional field maps computed by Elmer 312

Garfield::ComponentFieldMap
Base class for components based on finite-element field maps 318

Garfield::ComponentGrid
Component for interpolating field maps on a regular mesh 338

Garfield::ComponentNeBem2d
Two-dimensional implementation of the nearly exact Boundary Element Method 354

Garfield::ComponentNeBem3d
Interface to neBEM 366

Garfield::ComponentNeBem3dMap
Component for interpolating field maps stored in a mesh generated by neBEM 386

Garfield::ComponentParallelPlate
Component for parallel-plate geometries 395

Garfield::ComponentTcad2d
Interpolation in a two-dimensional field map created by Sentaurus Device 407

Garfield::ComponentTcad3d
Interpolation in a three-dimensional field map created by Sentaurus Device 414

Garfield::ComponentTcadBase< N >
Interpolation in a field map created by Sentaurus Device 422

Garfield::ComponentUser
Simple component with electric field given by a user function 439

Garfield::ComponentVoxel
Component for interpolating field maps stored in a regular mesh 446

Garfield::MediumMagboltz::Deexcitation 457

Garfield::ComponentTcadBase< N >::Defect 459

Garfield::DriftLineRKF
Calculation of drift lines based on macroscopic transport coefficients using Runge-Kutta-↩
Fehlberg integration 459

Garfield::ComponentParallelPlate::Electrode
Structure that captures the information of the electrodes under study 474

Garfield::AvalancheMicroscopic::Electron 476

Garfield::TrackDegrade::Electron 476

Garfield::TrackHeed::Electron 477

Garfield::ComponentFieldMap::Element 478

Garfield::ComponentNeBem2d::Element 479

Garfield::ComponentNeBem3d::Element 480

Garfield::ComponentNeBem3dMap::Element 482

Garfield::ComponentTcadBase< N >::Element 483

Garfield::ComponentVoxel::Element 484

Garfield::AvalancheMC::EndPoint 484

Garfield::TrackDegrade::Excitation 485

Garfield::MediumGas::ExcLevel 486

Garfield::Geometry
Abstract base class for geometry classes 487

Garfield::GeometryRoot
Use a geometry defined using the ROOT TGeo package 489

Garfield::GeometrySimple
"Native" geometry, using simple shapes 492

Garfield::AvalancheGridSpaceCharge::GridNode 496

Garfield::MediumGas::IonLevel 499

Garfield::KDTree
Main k-d tree class 500

Garfield::KDTreeNode
A node in the k-d tree 503

Garfield::KDTreeResult
Search result 506

Garfield::ComponentFieldMap::Material 506

Garfield::MediumCdTe
Cadmium-Telluride 507

Garfield::MediumConductor
Conducting medium 511

Garfield::MediumDiamond
Diamond 512

Garfield::MediumGaAs
Gallium-Arsenide 516

Garfield::MediumGaN
Gallium-Nitride 520

Garfield::MediumGas
Base class for gas media 524

Garfield::MediumMagboltz
Interface to Magboltz (version 11) 538

Garfield::MediumPlastic
Plastic medium 560

Garfield::MediumSilicon
Solid crystalline silicon 561

Garfield::ComponentFieldMap::Node 581

Garfield::ComponentGrid::Node 581

Garfield::OpticalData
Photoabsorption cross-sections for some gases 582

Garfield::Panel
Surface panel 585

Garfield::TrackElectron::Parameters 586

Garfield::AvalancheGrid::Path 587

Garfield::AvalancheMicroscopic::Photon 588

Garfield::TrackHeed::Photon 589

Garfield::ComponentAnalyticField::Pixel 590

Garfield::ComponentAnalyticField::Plane 591

Garfield::PlottingEngine
Plotting style 592

Garfield::AvalancheGridSpaceCharge::Point 594

Garfield::AvalancheMC::Point 595

Garfield::AvalancheMicroscopic::Point 595

Garfield::ComponentNeBem3d::Primitive 597

Garfield::QuadTree
Quadtree search 599

Garfield::Random 602

Garfield::RandomEngine< Engine, SeedType >
Abstract base class for random number generators 602

Garfield::RandomEngineRoot 604

Garfield::RandomEngineSTL 606

Garfield::ComponentComsol::Range 608

Garfield::ComponentNeBem2d::Region 609

Garfield::ComponentTcadBase< N >::Region 610

Garfield::AvalancheMC::Seed 611

Garfield::AvalancheMicroscopic::Seed 611

Garfield::ComponentNeBem2d::Segment 612

Garfield::Shaper
Class for signal processing 613

Garfield::Solid
Abstract base class for solids 616

Garfield::SolidBox
Box 630

Garfield::SolidExtrusion
Extrusion 636

Garfield::SolidHole
Box with a cylindrical hole 643

Garfield::SolidRidge
Triangular prism (Toblerone bar) 652

Garfield::SolidSphere
Sphere 660

Garfield::SolidTube
Cylindrical tube 667

Garfield::SolidWire
Wire 677

Garfield::ComponentNeBem2d::SpaceCharge 683

Garfield::AvalancheMicroscopic::Statistics 684

Garfield::ComponentAnalyticField::Strip 685

TGeoTet 686

Garfield::Track
Abstract base class for track generation 689

Garfield::TrackBichsel
Generate tracks using differential cross-sections for silicon computed by Hans Bichsel 696

Garfield::TrackDegrade
Interface to Degrade 702

Garfield::TrackElectron
[WIP] Ionization calculation based on MIP program (S. Biagi) 708

Garfield::TrackHeed
Generate tracks using Heed++ 712

Garfield::TrackPAI 729

Garfield::TrackSimple
Generate tracks based on a cluster density given by the user 735

Garfield::TrackSrim
Generate tracks based on SRIM energy loss, range and straggling tables 739

Garfield::TrackTrim
Generate tracks based on TRIM output files 751

Garfield::Vec1Impl< type > 758

Garfield::Vec2Impl< type > 759

Garfield::Vec3Impl< type > 760

Garfield::Vector< type, Dimension > 762

Garfield::ViewBase
Base class for visualization classes 764

Garfield::ViewCell
Visualize the ”cell” defined in an analytic-field component 773

Garfield::ViewDrift
Visualize drift lines and tracks 778

Garfield::ViewFEMesh
Draw the mesh of a field-map component 788

Garfield::ViewField
Visualize the potential or electric field of a component or sensor 799

Garfield::ViewGeometry
Visualize a geometry defined using the ”native” shapes 810

Garfield::ViewIsochrons
Draw equal time contour lines 814

Garfield::ViewMedium
Plot transport coefficients as function of electric and magnetic field 820

Garfield::ViewSignal
Plot the signal computed by a sensor as a ROOT histogram 837

Garfield::ComponentFieldMap::WeightingFieldCopy 844

Garfield::ComponentAnalyticField::Wire 845

Garfield::ComponentNeBem2d::Wire 846

# 5 File Index

## 5.1 File List

Here is a list of all files with brief descriptions:

/builds/garfield/docs/source/Include/Garfield/AvalancheGrid.hh 847

/builds/garfield/docs/source/Include/Garfield/AvalancheGridSpaceCharge.hh 849

/builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh 853

/builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh 856

/builds/garfield/docs/source/Include/Garfield/Component.hh 861

/builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh 866

/builds/garfield/docs/source/Include/Garfield/ComponentAnsys121.hh 874

/builds/garfield/docs/source/Include/Garfield/ComponentAnsys123.hh 875

/builds/garfield/docs/source/Include/Garfield/ComponentComsol.hh 875

/builds/garfield/docs/source/Include/Garfield/ComponentConstant.hh 877

/builds/garfield/docs/source/Include/Garfield/ComponentCST.hh 878

/builds/garfield/docs/source/Include/Garfield/ComponentElmer.hh 881

/builds/garfield/docs/source/Include/Garfield/ComponentElmer2d.hh 881

/builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh 882

/builds/garfield/docs/source/Include/Garfield/ComponentGrid.hh 888

/builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh 891

/builds/garfield/docs/source/Include/Garfield/ComponentNeBem3d.hh 893

/builds/garfield/docs/source/Include/Garfield/ComponentNeBem3dMap.hh 898

/builds/garfield/docs/source/Include/Garfield/ComponentParallelPlate.hh 899

/builds/garfield/docs/source/Include/Garfield/ComponentTcad2d.hh 903

/builds/garfield/docs/source/Include/Garfield/ComponentTcad3d.hh 904

/builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh 906

/builds/garfield/docs/source/Include/Garfield/ComponentUser.hh 909

/builds/garfield/docs/source/Include/Garfield/ComponentVoxel.hh 911

/builds/garfield/docs/source/Include/Garfield/DegradeInterface.hh 913

/builds/garfield/docs/source/Include/Garfield/DriftLineRKF.hh 915

/builds/garfield/docs/source/Include/Garfield/FundamentalConstants.hh 918

/builds/garfield/docs/source/Include/Garfield/GarfieldConstants.hh 919

/builds/garfield/docs/source/Include/Garfield/Geometry.hh 920

/builds/garfield/docs/source/Include/Garfield/GeometryRoot.hh 920

/builds/garfield/docs/source/Include/Garfield/GeometrySimple.hh 921

/builds/garfield/docs/source/Include/Garfield/HelperMacros.hh 922

/builds/garfield/docs/source/Include/Garfield/KDTree.hh 923

/builds/garfield/docs/source/Include/Garfield/LinkDef.h 925

/builds/garfield/docs/source/Include/Garfield/MagboltzInterface.hh 926

/builds/garfield/docs/source/Include/Garfield/Medium.hh 933

/builds/garfield/docs/source/Include/Garfield/MediumCdTe.hh 942

/builds/garfield/docs/source/Include/Garfield/MediumConductor.hh 943

/builds/garfield/docs/source/Include/Garfield/MediumDiamond.hh 944

/builds/garfield/docs/source/Include/Garfield/MediumGaAs.hh 945

/builds/garfield/docs/source/Include/Garfield/MediumGaN.hh 946

/builds/garfield/docs/source/Include/Garfield/MediumGas.hh 947

/builds/garfield/docs/source/Include/Garfield/MediumMagboltz.hh 950

/builds/garfield/docs/source/Include/Garfield/MediumPlastic.hh 955

/builds/garfield/docs/source/Include/Garfield/MediumSilicon.hh 955

/builds/garfield/docs/source/Include/Garfield/MultiProcessInterface.hh 959

/builds/garfield/docs/source/Include/Garfield/Numerics.hh 960

/builds/garfield/docs/source/Include/Garfield/OpticalData.hh 964

/builds/garfield/docs/source/Include/Garfield/Plotting.hh 965

/builds/garfield/docs/source/Include/Garfield/PlottingEngine.hh 965

/builds/garfield/docs/source/Include/Garfield/Polygon.hh 966

/builds/garfield/docs/source/Include/Garfield/QuadTree.hh 967

/builds/garfield/docs/source/Include/Garfield/Random.hh 968

/builds/garfield/docs/source/Include/Garfield/RandomEngine.hh 970

/builds/garfield/docs/source/Include/Garfield/RandomEngineRoot.hh 971

/builds/garfield/docs/source/Include/Garfield/RandomEngineSTL.hh 971

/builds/garfield/docs/source/Include/Garfield/Sensor.hh 972

/builds/garfield/docs/source/Include/Garfield/Shaper.hh 977

/builds/garfield/docs/source/Include/Garfield/Solid.hh 977

/builds/garfield/docs/source/Include/Garfield/SolidBox.hh 980

/builds/garfield/docs/source/Include/Garfield/SolidExtrusion.hh 981

/builds/garfield/docs/source/Include/Garfield/SolidHole.hh 982

/builds/garfield/docs/source/Include/Garfield/SolidRidge.hh 983

/builds/garfield/docs/source/Include/Garfield/SolidSphere.hh 984

/builds/garfield/docs/source/Include/Garfield/SolidTube.hh 985

/builds/garfield/docs/source/Include/Garfield/SolidWire.hh 986

/builds/garfield/docs/source/Include/Garfield/TetrahedralTree.hh 987

/builds/garfield/docs/source/Include/Garfield/TGeoTet.hh 989

/builds/garfield/docs/source/Include/Garfield/Track.hh 990

/builds/garfield/docs/source/Include/Garfield/TrackBichsel.hh 991

/builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh 992

/builds/garfield/docs/source/Include/Garfield/TrackElectron.hh 994

/builds/garfield/docs/source/Include/Garfield/TrackHeed.hh 995

/builds/garfield/docs/source/Include/Garfield/TrackPAI.hh 998

/builds/garfield/docs/source/Include/Garfield/TrackSimple.hh 999

/builds/garfield/docs/source/Include/Garfield/TrackSrim.hh 1000

/builds/garfield/docs/source/Include/Garfield/TrackTrim.hh 1002

/builds/garfield/docs/source/Include/Garfield/Utilities.hh 1003

/builds/garfield/docs/source/Include/Garfield/Vector.hh 1004

/builds/garfield/docs/source/Include/Garfield/ViewBase.hh 1006

/builds/garfield/docs/source/Include/Garfield/ViewCell.hh 1008

/builds/garfield/docs/source/Include/Garfield/ViewDrift.hh 1009

/builds/garfield/docs/source/Include/Garfield/ViewFEMesh.hh 1011

/builds/garfield/docs/source/Include/Garfield/ViewField.hh 1013

/builds/garfield/docs/source/Include/Garfield/ViewGeometry.hh 1015

/builds/garfield/docs/source/Include/Garfield/ViewIsochrons.hh 1016

/builds/garfield/docs/source/Include/Garfield/ViewMedium.hh 1017

/builds/garfield/docs/source/Include/Garfield/ViewSignal.hh 1020

# 6 Namespace Documentation

## 6.1 Garfield Namespace Reference

Namespaces

- namespace Degrade
- namespace Magboltz
- namespace Numerics

  Collection of numerical routines.
- namespace Polygon

Classes

- class AvalancheGrid

  Calculate avalanches in a uniform electric field using avalanche statistics.
- class AvalancheGridSpaceCharge

  Propagates avalanches with the 2d (axi-symmetric) space-charge routine from Lippmann, Riegler (2004) in uniform background fields.
- class AvalancheMC

  Calculate drift lines and avalanches based on macroscopic transport coefficients, using Monte Carlo integration.
- class AvalancheMicroscopic

  Calculate electron drift lines and avalanches using microscopic tracking.
- class ComponentAnalyticField

  Semi-analytic calculation of two-dimensional configurations consisting of wires, planes, and tubes.

- class ComponentAnsys121

  Component for importing and interpolating two-dimensional ANSYS field maps.
- class ComponentAnsys123

  Component for importing and interpolating three-dimensional ANSYS field maps.
- class ComponentComsol

  Component for importing and interpolating Comsol field maps.
- class ComponentConstant

  Component with constant electric field.
- class ComponentCST

  Component for importing and interpolating field maps from CST.
- class ComponentElmer

  Component for importing field maps computed by Elmer.
- class ComponentElmer2d

  Component for importing two-dimensional field maps computed by Elmer.
- class ComponentFieldMap

  Base class for components based on finite-element field maps.
- class ComponentGrid

  Component for interpolating field maps on a regular mesh.
- class ComponentNeBem2d

  Two-dimensional implementation of the nearly exact Boundary Element Method.
- class ComponentNeBem3d

  Interface to neBEM.
- class ComponentNeBem3dMap

  Component for interpolating field maps stored in a mesh generated by neBEM.
- class ComponentParallelPlate

  Component for parallel-plate geometries.
- class ComponentTcad2d

  Interpolation in a two-dimensional field map created by Sentaurus Device.
- class ComponentTcad3d

  Interpolation in a three-dimensional field map created by Sentaurus Device.
- class ComponentTcadBase

  Interpolation in a field map created by Sentaurus Device.
- class ComponentUser

  Simple component with electric field given by a user function.
- class ComponentVoxel

  Component for interpolating field maps stored in a regular mesh.
- class DriftLineRKF

  Calculation of drift lines based on macroscopic transport coefficients using Runge-Kutta-Fehlberg integration.
- class Geometry

  Abstract base class for geometry classes.
- class GeometryRoot

  Use a geometry defined using the ROOT TGeo package.
- class GeometrySimple

  "Native" geometry, using simple shapes.
- class KDTree

  Main k-d tree class.
- class KDTreeNode

  A node in the k-d tree.
- struct KDTreeResult

  Search result.

- class MediumCdTe
  Cadmium-Telluride.
- class MediumConductor
  Conducting medium.
- class MediumDiamond
  Diamond.
- class MediumGaAs
  Gallium-Arsenide.
- class MediumGaN
  Gallium-Nitride.
- class MediumGas
  Base class for gas media.
- class MediumMagboltz
  Interface to Magboltz (version 11).
- class MediumPlastic
  Plastic medium.
- class MediumSilicon
  Solid crystalline silicon
- class OpticalData
  Photoabsorption cross-sections for some gases.
- struct Panel
  Surface panel.
- class PlottingEngine
  Plotting style.
- class QuadTree
  Quadtree search.
- class Random
- class RandomEngine
  Abstract base class for random number generators.
- class RandomEngineRoot
- class RandomEngineSTL
- class Shaper
  Class for signal processing.
- class Solid
  Abstract base class for solids.
- class SolidBox
  Box.
- class SolidExtrusion
  Extrusion.
- class SolidHole
  Box with a cylindrical hole.
- class SolidRidge
  Triangular prism (Toblerone bar).
- class SolidSphere
  Sphere.
- class SolidTube
  Cylindrical tube.
- class SolidWire
  Wire.
- class Track
  Abstract base class for track generation.

- class TrackBichsel

  Generate tracks using differential cross-sections for silicon computed by Hans Bichsel.
- class TrackDegrade

  Interface to Degrade.
- class TrackElectron

  [WIP] Ionization calculation based on MIP program (S. Biagi).
- class TrackHeed

  Generate tracks using Heed++.
- class TrackPAI
- class TrackSimple

  Generate tracks based on a cluster density given by the user.
- class TrackSrim

  Generate tracks based on SRIM energy loss, range and straggling tables.
- class TrackTrim

  Generate tracks based on TRIM output files.
- class Vec1Impl
- class Vec2Impl
- class Vec3Impl
- class Vector
- class ViewBase

  Base class for visualization classes.
- class ViewCell

  Visualize the "cell" defined in an analytic-field component.
- class ViewDrift

  Visualize drift lines and tracks.
- class ViewFEMesh

  Draw the mesh of a field-map component.
- class ViewField

  Visualize the potential or electric field of a component or sensor.
- class ViewGeometry

  Visualize a geometry defined using the "native" shapes.
- class ViewIsochrons

  Draw equal time contour lines.
- class ViewMedium

  Plot transport coefficients as function of electric and magnetic field.
- class ViewSignal

  Plot the signal computed by a sensor as a ROOT histogram.

Typedefs

- typedef std::vector< std::vector< double > > KDTreeArray
- using Vec3 = Vec3Impl<double>
- using Vec3D = Vec3Impl<double>
- using Vec3F = Vec3Impl<float>
- using ViewMesh = ViewFEMesh

Enumerations

- enum class Particle {
  Electron = 0 , Ion , Hole , Positron ,
  NegativeIon , Photon }
- enum class MPRunMode { Normal = 0 , GPUExclusive }

Functions

- class GARFIELD_CLASS_NAME (Component)

  Abstract base class for components.
- class GARFIELD_CLASS_NAME (Medium)

  Abstract base class for components.
- void SetDefaultStyle ()
- void SetSerif ()
- void SetSansSerif ()
- double RndmUniform ()

  Draw a random number uniformly distributed in the range [0, 1).
- double RndmUniformPos ()

  Draw a random number uniformly distributed in the range (0, 1).
- double RndmGaussian ()

  Draw a Gaussian random variate with mean zero and standard deviation one.
- double RndmGaussian (const double mu, const double sigma)

  Draw a Gaussian random variate with mean mu and standard deviation sigma.
- double RndmLorentzian (const double mu, const double gamma)

  Draw a Lorentzian random variate with mean mu and half-width at half maximum gamma.
- double RndmVoigt (const double mu, const double sigma, const double gamma)

  Draw a random number according to a Voigt function with mean mu.
- unsigned int RndmYuleFurry (const double mean)

  Draw a random number from a geometric distribution.
- double RndmPolya (const double theta)

  Draw a Polya distributed random number.
- double RndmLandau ()

  Draw a random number from a Landau distribution.
- double RndmVavilov (const double rkappa, const double beta2)

  Draw a random number from a Vavilov distribution.
- int RndmPoisson (const double mean)

  Draw a random number from a Poisson distribution.
- double RndmHeedWF (const double w, const double f)

  Draw a random energy needed to create a single electron in a material asymptotic work function W and Fano factor F, according to Igor Smirnov's phenomenological model.
- void RndmDirection (double &dx, double &dy, double &dz, const double length=1.)

  Draw a random (isotropic) direction vector.
- class GARFIELD_CLASS_NAME (Sensor)

  Sensor
- class GARFIELD_CLASS_NAME (TetrahedralTree)

  Helper class for searches in field maps.
- void ltrim (std::string &line)
- void rtrim (std::string &line)
- std::vector< std::string > tokenize (const std::string &line)
- bool startsWith (const std::string &line, const std::string &s)

Variables

- ComponentNeBem3d * gComponentNeBem3d
- PlottingEngine plottingEngine

### 6.1.1 Typedef Documentation

#### KDTreeArray

typedef std::vector<std::vector<double> > Garfield::KDTreeArray

Definition at line 19 of file KDTree.hh.

#### Vec3

using Garfield::Vec3 = Vec3Impl<double>

Definition at line 27 of file TetrahedralTree.hh.

#### Vec3D

using Garfield::Vec3D = Vec3Impl<double>

Definition at line 98 of file Vector.hh.

#### Vec3F

using Garfield::Vec3F = Vec3Impl<float>

Definition at line 99 of file Vector.hh.

#### ViewMesh

using Garfield::ViewMesh = ViewFEMesh

Definition at line 163 of file ViewFEMesh.hh.

### 6.1.2 Enumeration Type Documentation

#### MPRunMode

enum class Garfield::MPRunMode [strong]

Enumerator

| | |
|---|---|
| Normal | |
| GPUExclusive | |

Definition at line 7 of file MultiProcessInterface.hh.

```
00007 { Normal = 0, GPUExclusive };
```

#### Particle

enum class Garfield::Particle [strong]

Enumerator

| | |
|---|---|
| Electron | |
| Ion | |
| Hole | |
| Positron | |
| NegativeIon | |
| Photon | |

Definition at line 7 of file GarfieldConstants.hh.

00007 { Electron = 0, Ion, Hole, Positron, NegativeIon, Photon };

### 6.1.3 Function Documentation

GARFIELD_CLASS_NAME() [1/4]

class Garfield::GARFIELD_CLASS_NAME (
Component )

Abstract base class for components.

Default constructor.

Constructor

Destructor

Define the geometry.

Reset.

Get the medium at a given location (x, y, z).

Calculate the drift field at given point.

Parameters

| | |
|---|---|
| x,y,z | coordinates [cm]. |
| ex,ey,ez | components of the electric field [V/cm]. |
| m | pointer to the medium at this location. |
| status | status flag |

Status flags:

0: Inside an active medium
> 0: Inside a wire of type X
-4 ... -1: On the side of a plane where no wires are
-5: Inside the mesh but not in an active medium
-6: Outside the mesh
-10: Unknown potential type (should not occur)
other: Other cases (should not occur)

Calculate the drift field [V/cm] and potential [V] at (x, y, z).

Calculate the drift field [V/cm] at (x, y, z).

Calculate the (drift) electrostatic potential [V] at (x, y, z).

Calculate the voltage range [V].

Calculate the weighting field at a given point and for a given electrode.

Parameters

| | |
|---|---|
| x,y,z | coordinates [cm]. |
| wx,wy,wz | components of the weighting field [1/cm]. |
| label | name of the electrode |

Calculate the weighting potential at a given point.

Parameters

| | |
|---|---|
| x,y,z | coordinates [cm]. |
| label | name of the electrode. |

Returns

weighting potential [dimensionless].

Return the time steps at which the delayed weighting potential/field are stored/evaluated.

Calculate the delayed weighting field at a given point and time and for a given electrode.

Parameters

| | |
|---|---|
| x,y,z | coordinates [cm]. |
| t | time [ns]. |
| wx,wy,wz | components of the weighting field [1/cm]. |
| label | name of the electrode |

Calculate the delayed weighting potential at a given point and time and for a given electrode.

Parameters

| | |
|---|---|
| x,y,z | coordinates [cm]. |
| t | time [ns]. |
| label | name of the electrode |

Calculate the delayed weighting potentials at a given point and for a given electrode, for a set of pre-defined times.

Calculate the magnetic field at a given point.

Parameters

| | |
|---|---|
| x,y,z | coordinates [cm]. |
| bx,by,bz | components of the magnetic field [Tesla]. |
| status | status flag. |

---

Set a constant magnetic field.

Ready for use?

Does the component have a 3D field (map)?

Get the bounding box coordinates.

Get the coordinates of the elementary cell.

Return the number of mesh elements.

Get the indices of the nodes constituting a given element.

Get the region/material of a mesh element and a flag whether it is associated to an active medium.

Return the number of mesh nodes.

Get the coordinates of a mesh node.

Integrate the normal component of the electric field over a circle.

Parameters

| | |
|---|---|
| xc,yc | centre of the circle [cm] |
| r | radius [cm] |
| nI | number of intervals for the integration |

Returns

charge enclosed in the circle [fC / cm]

Integrate the normal component of the electric field over a sphere.

Parameters

| | |
|---|---|
| xc,yc,zc | centre of the sphere [cm] |
| r | radius of the sphere [cm] |
| nI | number of integration intervals in phi and theta |

Returns

charge enclosed in the sphere [fC]

Integrate the normal component of the electric field over a parallelogram.

Parameters

| | |
|---|---|
| x0,y0,z0 | coordinates of one of the corners [cm] |
| dx1,dy1,dz1 | vector to one of the adjacent corners [cm] |
| dx2,dy2,dz2 | vector to the other adjacent corner [cm] |
| nU,nV | number of integration points in the two directions |

Returns

flux [V cm]

Integrate the normal component of the weighting field over a parallelogram.

Integrate the electric field flux through a line from (x0,y0,z0) to (x1,y1,z1) along a direction (xp,yp,zp).

Parameters

| | |
|---|---|
| x0,y0,z0 | coordinates of the starting point |
| x1,y1,z1 | coordinates of the end point |
| xp,yp,zp | normal vector |
| nI | number of intervals for the integration |
| isign | include both negative and positive contributions (0) or only contributions with a given polarity (+1,-1) |

Determine whether the line between two points crosses a wire.

Parameters

| | |
|---|---|
| x0,y0,z0 | first point [cm]. |
| x1,y1,z1 | second point [cm] |
| xc,yc,zc | point [cm] where the line crosses the wire or the coordinates of the wire centre. |
| centre | flag whether to return the coordinates of the line-wire crossing point or of the wire centre. |
| rc | radius [cm] of the wire. |

Determine whether a particle is inside the trap radius of a wire.

Parameters

| | |
|---|---|
| q0 | charge of the particle [in elementary charges]. |
| x0,y0,z0 | position [cm] of the particle. |
| xw,yw | coordinates of the wire (if applicable). |
| rw | radius of the wire (if applicable). |

Determine whether the line between two points crosses a plane.

Enable simple periodicity in the $x$ direction.

Enable simple periodicity in the $y$ direction.

Enable simple periodicity in the $z$ direction.

Return periodicity flags.

Enable mirror periodicity in the $x$ direction.

Enable mirror periodicity in the $y$ direction.

Enable mirror periodicity in the $y$ direction.

Return mirror periodicity flags.

Enable axial periodicity in the $x$ direction.

Enable axial periodicity in the $y$ direction.

Enable axial periodicity in the $z$ direction.

Return axial periodicity flags.

Enable rotation symmetry around the $x$ axis.

Enable rotation symmetry around the $y$ axis.

Enable rotation symmetry around the $z$ axis.

Return rotation symmetry flags.

Enable triangular periodicity in the $xy$ plane.

Enable triangular periodicity in the $xz$ plane.

Enable triangular periodicity in the $yz$ plane.

Switch on debugging messages.

Switch off debugging messages.

Does the component have a non-zero magnetic field?

Does the component have maps of the Townsend coefficient?

Does the component have maps of the attachment coefficient?

Does the component have maps of the low-field mobility?

Does the component have velocity maps?

Get the electron attachment coefficient.

Get the hole attachment coefficient.

Get the hole Mobility coefficient.

Get the electron Townsend coefficient.

Get the hole Townsend coefficient.

Get the electron drift velocity.

Get the hole drift velocity.

Create and initialise GPU Transfer class

Class name.

Pointer to the geometry.

Constant magnetic field.

Ready for use?

Switch on/off debugging messages

Simple periodicity in x, y, z.

Mirror periodicity in x, y, z.

Axial periodicity in x, y, z.

Rotation symmetry around x-axis, y-axis, z-axis.

Triangle symmetric in the xy, xz, and yz plane.

Triangle symmetric octant of imported map (0 < phi < Pi/4 --> octant 1).

Octants where |x| >= |y|

Time steps at which the delayed weighting potentials/fields are stored.

Reset the component.

Verify periodicities.

Definition at line 1 of file Component.hh.

```
00031                                   {
00032  public:
00033 #ifdef ___GPUCOMPILE___
00034   GARFIELD_CLASS_NAME(Component)() = default;
00035 #else
00037   GARFIELD_CLASS_NAME(Component)() = delete;
00039   GARFIELD_CLASS_NAME(Component)(const std::string& name);
00040 #endif
00042   virtual ~GARFIELD_CLASS_NAME(Component)() {};
00043
00044 #ifndef ___GPUCOMPILE___
00046   virtual void SetGeometry(Geometry* geo);
00048   virtual void Clear();
00049
00051   virtual Medium* GetMedium(const double x, const double y, const double z);
00052
00070   virtual void ElectricField(const double x, const double y, const double z,
00071                              double& ex, double& ey, double& ez, Medium*& m,
00072                              int& status) = 0;
00074   virtual void ElectricField(const double x, const double y, const double z,
00075                              double& ex, double& ey, double& ez, double& v,
00076                              Medium*& m, int& status) = 0;
00077 #else
00078   ___device___ void ElectricField(const cuda_t xin, const cuda_t yin,
00079                                   const cuda_t zin, cuda_t& ex, cuda_t& ey,
00080                                   cuda_t& ez, MediumGPU*& m, int& status);
00081 #endif
00082
00083 #ifndef ___GPUCOMPILE___
00085   std::array<double, 3> ElectricField(const double x, const double y,
00086                                       const double z);
00088   virtual double ElectricPotential(const double x, const double y,
00089                                    const double z);
00091   virtual bool GetVoltageRange(double& vmin, double& vmax) = 0;
00092
00098   virtual void WeightingField(const double x, const double y, const double z,
00099                               double& wx, double& wy, double& wz,
00100                               const std::string& label);
00106   virtual double WeightingPotential(const double x, const double y,
00107                                     const double z, const std::string& label);
00108
00112   virtual const std::vector<double>& DelayedSignalTimes(
00113       const std::string& /*label*/) {
00114     return m_wdtimes;
00115   }
00123   virtual void DelayedWeightingField(const double x, const double y,
00124                                      const double z, const double t, double& wx,
00125                                      double& wy, double& wz,
00126                                      const std::string& label);
00127
00134   virtual double DelayedWeightingPotential(const double x, const double y,
00135                                            const double z, const double t,
00136                                            const std::string& label);
00137
00141   virtual void DelayedWeightingPotentials(const double x, const double y,
00142                                           const double z,
00143                                           const std::string& label,
00144                                           std::vector<double>& dwp);
00151   virtual void MagneticField(const double x, const double y, const double z,
00152                              double& bx, double& by, double& bz, int& status);
00154   void SetMagneticField(const double bx, const double by, const double bz);
00155
```

```
00157   virtual bool IsReady() { return m_ready; }
00159   virtual bool Is3d() { return true; }
00160
00162   virtual bool GetBoundingBox(double& xmin, double& ymin, double& zmin,
00163                               double& xmax, double& ymax, double& zmax);
00164
00166   virtual bool GetElementaryCell(double& xmin, double& ymin, double& zmin,
00167                                  double& xmax, double& ymax, double& zmax);
00168   // Get the x-length of the elementary cell.
00169   double CellSizeX();
00170   // Get the y-length of the elementary cell.
00171   double CellSizeY();
00172   // Get the z-length of the elementary cell.
00173   double CellSizeZ();
00174
00176   virtual size_t GetNumberOfElements() const { return 0; }
00178   virtual bool GetElementNodes(const size_t /*i*/,
00179                                std::vector<size_t>& /*nodes*/) const {
00180     return false;
00181   }
00184   virtual bool GetElementRegion(const size_t /*i*/, size_t& /*mat*/,
00185                                 bool& /*drift*/) const {
00186     return false;
00187   }
00189   virtual size_t GetNumberOfNodes() const { return 0; }
00191   virtual bool GetNode(const size_t i, double& x, double& y, double& z) const;
00192
00200   double IntegrateFluxCircle(const double xc, const double yc, const double r,
00201                              const unsigned int nI = 50);
00209   double IntegrateFluxSphere(const double xc, const double yc, const double zc,
00210                              const double r, const unsigned int nI = 20);
00211
00220   double IntegrateFluxParallelogram(
00221       const double x0, const double y0, const double z0, const double dx1,
00222       const double dy1, const double dz1, const double dx2, const double dy2,
00223       const double dz2, const unsigned int nU = 20, const unsigned int nV = 20);
00224
00227   double IntegrateWeightingFluxParallelogram(
00228       const std::string& label, const double x0, const double y0,
00229       const double z0, const double dx1, const double dy1, const double dz1,
00230       const double dx2, const double dy2, const double dz2,
00231       const unsigned int nU = 20, const unsigned int nV = 20);
00232
00242   double IntegrateFluxLine(const double x0, const double y0, const double z0,
00243                            const double x1, const double y1, const double z1,
00244                            const double xp, const double yp, const double zp,
00245                            const unsigned int nI, const int isign = 0);
00246
00256   virtual bool CrossedWire(const double x0, const double y0, const double z0,
00257                            const double x1, const double y1, const double z1,
00258                            double& xc, double& yc, double& zc,
00259                            const bool centre, double& rc);
00266   virtual bool InTrapRadius(const double q0, const double x0, const double y0,
00267                             const double z0, double& xw, double& yw,
00268                             double& rw);
00271   virtual bool CrossedPlane(const double x0, const double y0, const double z0,
00272                             const double x1, const double y1, const double z1,
00273                             double& xc, double& yc, double& zc);
00274
00276   void EnablePeriodicityX(const bool on = true) {
00277     m_periodic[0] = on;
00278     UpdatePeriodicity();
00279   }
00281   void EnablePeriodicityY(const bool on = true) {
00282     m_periodic[1] = on;
00283     UpdatePeriodicity();
00284   }
00286   void EnablePeriodicityZ(const bool on = true) {
00287     m_periodic[2] = on;
00288     UpdatePeriodicity();
00289   }
00291   void IsPeriodic(bool& perx, bool& pery, bool& perz) {
00292     perx = m_periodic[0];
00293     pery = m_periodic[1];
00294     perz = m_periodic[2];
00295   }
00296
00298   void EnableMirrorPeriodicityX(const bool on = true) {
00299     m_mirrorPeriodic[0] = on;
00300     UpdatePeriodicity();
00301   }
00303   void EnableMirrorPeriodicityY(const bool on = true) {
00304     m_mirrorPeriodic[1] = on;
00305     UpdatePeriodicity();
00306   }
00308   void EnableMirrorPeriodicityZ(const bool on = true) {
00309     m_mirrorPeriodic[2] = on;
```

```
00310   UpdatePeriodicity();
00311 }
00313 void IsMirrorPeriodic(bool& perx, bool& pery, bool& perz) {
00314   perx = m_mirrorPeriodic[0];
00315   pery = m_mirrorPeriodic[1];
00316   perz = m_mirrorPeriodic[2];
00317 }
00318
00320 void EnableAxialPeriodicityX(const bool on = true) {
00321   m_axiallyPeriodic[0] = on;
00322   UpdatePeriodicity();
00323 }
00325 void EnableAxialPeriodicityY(const bool on = true) {
00326   m_axiallyPeriodic[1] = on;
00327   UpdatePeriodicity();
00328 }
00330 void EnableAxialPeriodicityZ(const bool on = true) {
00331   m_axiallyPeriodic[2] = on;
00332   UpdatePeriodicity();
00333 }
00335 void IsAxiallyPeriodic(bool& perx, bool& pery, bool& perz) {
00336   perx = m_axiallyPeriodic[0];
00337   pery = m_axiallyPeriodic[1];
00338   perz = m_axiallyPeriodic[2];
00339 }
00340
00342 void EnableRotationSymmetryX(const bool on = true) {
00343   m_rotationSymmetric[0] = on;
00344   UpdatePeriodicity();
00345 }
00347 void EnableRotationSymmetryY(const bool on = true) {
00348   m_rotationSymmetric[1] = on;
00349   UpdatePeriodicity();
00350 }
00352 void EnableRotationSymmetryZ(const bool on = true) {
00353   m_rotationSymmetric[2] = on;
00354   UpdatePeriodicity();
00355 }
00357 void IsRotationSymmetric(bool& rotx, bool& roty, bool& rotz) {
00358   rotx = m_rotationSymmetric[0];
00359   roty = m_rotationSymmetric[1];
00360   rotz = m_rotationSymmetric[2];
00361 }
00362
00364 void EnableTriangleSymmetricXY(const bool on = true,
00365                                const bool oct = 2) {
00366   m_triangleSymmetric[0] = on;
00367   m_triangleSymmetricOct = oct;
00368   m_mirrorPeriodic[0] = on;
00369   m_mirrorPeriodic[1] = on;
00370 }
00372 void EnableTriangleSymmetricXZ(const bool on = true,
00373                                const bool oct = 2) {
00374   m_triangleSymmetric[1] = on;
00375   m_triangleSymmetricOct = oct;
00376   m_mirrorPeriodic[0] = on;
00377   m_mirrorPeriodic[2] = on;
00378 }
00380 void EnableTriangleSymmetricYZ(const bool on = true,
00381                                const bool oct = 2) {
00382   m_triangleSymmetric[2] = on;
00383    m_triangleSymmetricOct = oct;
00384   m_mirrorPeriodic[1] = on;
00385   m_mirrorPeriodic[2] = on;
00386 }
00387
00389 void EnableDebugging(const bool on = true) { m_debug = on; }
00391 void DisableDebugging() { m_debug = false; }
00392
00394 virtual bool HasMagneticField() const;
00395
00397 virtual bool HasTownsendMap() const { return false; }
00399 virtual bool HasAttachmentMap() const { return false; }
00401 virtual bool HasMobilityMap() const { return false; }
00403 virtual bool HasVelocityMap() const { return false; }
00404
00406 virtual bool ElectronAttachment(const double /*x*/, const double /*y*/,
00407                                 const double /*z*/, double& eta) {
00408   eta = 0;
00409   return false;
00410 }
00412 virtual bool HoleAttachment(const double /*x*/, const double /*y*/,
00413                             const double /*z*/, double& eta) {
00414   eta = 0;
00415   return false;
00416 }
00417
```

```
00418   // Get the electron mobility coefficient.
00419   virtual bool ElectronMobility(const double /*x*/, const double /*y*/,
00420                                 const double /*z*/, double& mu) {
00421     mu = 0;
00422     return false;
00423   }
00425   virtual bool HoleMobility(const double /*x*/, const double /*y*/,
00426                             const double /*z*/, double& mu) {
00427     mu = 0;
00428     return false;
00429   }
00430
00432   virtual bool ElectronTownsend(const double /*x*/, const double /*y*/,
00433                                 const double /*z*/, double& alpha) {
00434     alpha = 0;
00435     return false;
00436   }
00438   virtual bool HoleTownsend(const double /*x*/, const double /*y*/,
00439                             const double /*z*/, double& alpha) {
00440     alpha = 0;
00441     return false;
00442   }
00444   virtual bool ElectronVelocity(const double /*x*/, const double /*y*/,
00445                                 const double /*z*/, double& vx, double& vy,
00446                                 double& vz) {
00447     vx = vy = vz = 0;
00448     return false;
00449   }
00451   virtual bool HoleVelocity(const double /*x*/, const double /*y*/,
00452                             const double /*z*/, double& vx, double& vy,
00453                             double& vz) {
00454     vx = vy = vz = 0;
00455     return false;
00456   }
00457
00458   virtual double StepSizeHint() { return -1.; }
00459
00461   virtual double CreateGPUTransferObject(ComponentGPU*& comp_gpu);
00462
00463  protected:
00465   std::string m_className = "Component";
00466
00468   Geometry* m_geometry = nullptr;
00469
00471   std::array<double, 3> m_b0 = {{0., 0., 0.}};
00472 #endif
00474   bool m_ready = false;
00475
00476 #ifdef ___GPUCOMPILE___
00478   bool m_periodic[3] = {false, false, false};
00480   bool m_mirrorPeriodic[3] = {false, false, false};
00482   bool m_axiallyPeriodic[3] = {false, false, false};
00484   bool m_rotationSymmetric[3] = {false, false, false};
00486   bool m_triangleSymmetric[3] = {false, false, false};
00488   int m_triangleSymmetricOct = 0;
00490   const int m_triangleOctRules[4] = {1, 4, 5, 8};
00491   bool m_outsideCone = false;
00492 #else
00494   bool m_debug = false;
00495
00497   std::array<bool, 3> m_periodic = {{false, false, false}};
00499   std::array<bool, 3> m_mirrorPeriodic = {{false, false, false}};
00501   std::array<bool, 3> m_axiallyPeriodic = {{false, false, false}};
00503   std::array<bool, 3> m_rotationSymmetric = {{false, false, false}};
00505   std::array<bool, 3> m_triangleSymmetric = {{false, false, false}};
00507   int m_triangleSymmetricOct = 0;
00509   const std::array<int, 4> m_triangleOctRules = {1, 4, 5, 8};
00510   bool m_outsideCone = false;
00511 #endif
00512
00513 #ifndef ___GPUCOMPILE___
00515   std::vector<double> m_wdtimes;
00516
00518   virtual void Reset() = 0;
00520   virtual void UpdatePeriodicity() = 0;
00521
00522  private:
00523   double IntegrateFluxParallelogram(const double x0, const double y0,
00524                                     const double z0, const double dx1,
00525                                     const double dy1, const double dz1,
00526                                     const double dx2, const double dy2,
00527                                     const double dz2, const unsigned int nU,
00528                                     const unsigned int nV, const bool wfield,
00529                                     const std::string& label);
00530 #else
00531
00532 // include parts from derived class due to big performance hit from using
```

```
00533 // virtual methods
00534 // TODO GPU TN: It isn't clear to me that we actually need (at least) the Ansys
00535 // include - all the code is protected by an ifndef ___GPUCOMPILE___, whereas it
00536 // will be defined when these includes are made
00537 #include "ComponentAnsys123.hh"
00538 #include "ComponentFieldMap.hh"
00539
00540   friend class ComponentAnsys123;
00541   friend class ComponentComsol;
00542   friend class ComponentElmer;
00543   friend class ComponentFieldMap;
00544   friend class Component;
00545
00546   // enum to mimic polymorphism
00547   enum class ComponentType {
00548     Component = 0,
00549     ComponentFieldMap,
00550     ComponentAnsys123,
00551     ComponentComsol,
00552     ComponentElmer
00553   };
00554
00555   ComponentType m_ComponentType{ComponentType::Component};
00556
00557 #endif
00558 };
```

## GARFIELD_CLASS_NAME() [2/4]

class Garfield::GARFIELD_CLASS_NAME (
Medium )

Abstract base class for components.

Constructor

Destructor

Return the id number of the class instance.

Get the medium name/identifier.

Is this medium a gas?

Is this medium a semiconductor?

Is this medium a conductor?

Set the temperature [K].

Get the temperature [K].

Set the relative static dielectric constant.

Get the relative static dielectric constant.

Get number of components of the medium.

Get the name and fraction of a given component.

Set the effective atomic number.

Get the effective atomic number.

Set the effective atomic weight.

Get the effective atomic weight.

Set the number density [cm-3].

Get the number density [cm-3].

Set the mass density [g/cm3].

Get the mass density [g/cm3].

Switch electron/ion/hole transport on/off.

Make the medium ionisable or non-ionisable.

Is charge carrier transport enabled in this medium?

Does the medium have electron scattering rates?

Is charge deposition by charged particles/photon enabled in this medium?

Set the W value (average energy to produce an electron/ion or e/h pair).

Get the W value.

Set the Fano factor.

Get the Fano factor.

Plot the drift velocity as function of the electric field.

Plot the diffusion coefficients as function of the electric field.

Plot the Townsend coefficient(s) as function of the electric field.

Plot the attachment coefficient(s) as function of the electric field.

Plot Townsend and attachment coefficients.

Drift velocity [cm / ns]

Flux (mean velocity; shorthand: wv) and bulk (center of mass velocity; shorthand: wr) drift velocity [cm / ns]

Longitudinal and transverse diffusion coefficients [cm1/2]

Diffusion tensor: diagonal elements are the diffusion coefficients [cm] along e, btrans, e x b, off-diagonal elements are the covariances

Ionisation coefficient [cm-1]

Attachment coefficient [cm-1]

TOF Ionisation rate [ns-1]

TOF Attachment Rate [ns-1]

Lorentz angle

Low-field mobility [cm2 V-1 ns-1]

Dispersion relation (energy vs. wave vector)

Sample the momentum vector for a given energy (only meaningful in semiconductors).

Null-collision rate [ns-1]

Collision rate [ns-1] for given electron energy

Sample the collision type. Update energy and direction vector.

Drift velocity [cm / ns]

Longitudinal and transverse diffusion coefficients [cm1/2]

Diffusion tensor

Ionisation coefficient [cm-1]

Attachment coefficient [cm-1]

Low-field mobility [cm2 V-1 ns-1]

Ion drift velocity [cm / ns]

Longitudinal and transverse diffusion coefficients [cm1/2]

Dissociation coefficient

Low-field ion mobility [cm2 V-1 ns-1]

Negative ion drift velocity [cm / ns]

Low-field negative ion mobility [cm2 V-1 ns-1]

Set the range of fields to be covered by the transport tables.

Set the fields and E-B angles to be used in the transport tables.

Get the fields and E-B angles used in the transport tables.

Set an entry in the table of drift speeds along E.

Get an entry in the table of drift speeds along E.

Set an entry in the table of drift speeds along ExB.

Get an entry in the table of drift speeds along ExB.

Set an entry in the table of drift speeds along Btrans.

Get an entry in the table of drift speeds along Btrans.

Set an entry in the table of flux drift speeds.

Get an entry in the table of flux drift speeds.

Set an entry in the table of bulk drift speeds.

Get an entry in the table of bulk drift speeds.

Set an entry in the table of longitudinal diffusion coefficients.

Get an entry in the table of longitudinal diffusion coefficients.

Set an entry in the table of transverse diffusion coefficients.

Get an entry in the table of transverse diffusion coefficients.

Set an entry in the table of Townsend coefficients.

Get an entry in the table of Townsend coefficients.

Set an entry in the table of attachment coefficients.

Get an entry in the table of attachment coefficients.

Set an entry in the table of ionization rate of TOF.

Get an entry in the table of ionization rate of TOF.

Set an entry in the table of attachment rate of TOF.

Get an entry in the table of attachment rate of TOF.

Set an entry in the table of Lorentz angles.

Get an entry in the table of Lorentz angles.

Set an entry in the table of drift speeds along E.

Get an entry in the table of drift speeds along E.

Set an entry in the table of drift speeds along ExB.

Get an entry in the table of drift speeds along ExB.

Set an entry in the table of drift speeds along Btrans.

Get an entry in the table of drift speeds along Btrans.

Set an entry in the table of longitudinal diffusion coefficients.

Get an entry in the table of longitudinal diffusion coefficients.

Set an entry in the table of transverse diffusion coefficients.

Get an entry in the table of transverse diffusion coefficients.

Set an entry in the table of Townsend coefficients.

Get an entry in the table of Townsend coefficients.

Set an entry in the table of attachment coefficients.

Get an entry in the table of attachment coefficients.

Initialise the table of ion mobilities from a list of electric fields and corresponding mobilities. The mobilities will be interpolated at the electric fields of the currently set grid.

Set an entry in the table of ion mobilities.

Get an entry in the table of ion mobilities.

Set an entry in the table of longitudinal diffusion coefficients.

Get an entry in the table of longitudinal diffusion coefficients.

Set an entry in the table of transverse diffusion coefficients.

Get an entry in the table of transverse diffusion coefficients.

Set an entry in the table of dissociation coefficients.

Get an entry in the table of dissociation coefficients.

Set an entry in the table of negative ion mobilities.

Get an entry in the table of negative ion mobilities.

Reset all tables of transport parameters.

Select the extrapolation method for fields below/above the table range. Possible options are ”constant”, ”linear”, and ”exponential”.

Set the degree of polynomial interpolation (usually 2).

Get the energy range [eV] of the available optical data.

Get the complex dielectric function at a given energy.

Switch on/off debugging messages

Create and initialise GPU Transfer class

Definition at line 1 of file Medium.hh.

```
00035                                 {
00036  public:
00037 #ifdef ___GPUCOMPILE___
00039   GARFIELD_CLASS_NAME(Medium)() = default;
00041   ~GARFIELD_CLASS_NAME(Medium)() {};
00042 #else
00044   GARFIELD_CLASS_NAME(Medium)();
00046   virtual ~GARFIELD_CLASS_NAME(Medium)();
00047 #endif
00048
00050   ___DEVICE___ int GetId() const { return m_id; }
00051
00052 #ifndef ___GPUCOMPILE___
00054   const std::string& GetName() const { return m_name; }
00056   virtual bool IsGas() const { return false; }
00058   virtual bool IsSemiconductor() const { return false; }
00060   virtual bool IsConductor() const { return false; }
00061
00063   void SetTemperature(const double t);
00065   double GetTemperature() const { return m_temperature; }
00066   // Set the pressure [Torr].
00067   void SetPressure(const double p);
00068   // Get the pressure [Torr].
00069   double GetPressure() const { return m_pressure; }
00071   void SetDielectricConstant(const double eps);
00073   double GetDielectricConstant() const { return m_epsilon; }
00074
00076   unsigned int GetNumberOfComponents() const { return m_nComponents; }
00078   virtual void GetComponent(const unsigned int i, std::string& label,
00079                             double& f);
00081   virtual void SetAtomicNumber(const double z);
00083   virtual double GetAtomicNumber() const { return m_z; }
00085   virtual void SetAtomicWeight(const double a);
00087   virtual double GetAtomicWeight() const { return m_a; }
00089   virtual void SetNumberDensity(const double n);
00091   virtual double GetNumberDensity() const { return m_density; }
00093   virtual void SetMassDensity(const double rho);
```

```
00095   virtual double GetMassDensity() const;
00096
00098   virtual void EnableDrift(const bool on = true) { m_driftable = on; }
00100   virtual void EnablePrimaryIonisation(const bool on = true) {
00101     m_ionisable = on;
00102   }
00103 #endif
00104
00106   ___DEVICE___ bool IsDriftable() const { return m_driftable; }
00108   ___DEVICE___ bool IsMicroscopic() const { return m_microscopic; }
00109
00110 #ifndef ___GPUCOMPILE___
00112   bool IsIonisable() const { return m_ionisable; }
00113
00115   void SetW(const double w) { m_w = w; }
00117   double GetW() const { return m_w; }
00119   void SetFanoFactor(const double f) { m_fano = f; }
00121   double GetFanoFactor() const { return m_fano; }
00122
00124   void PlotVelocity(const std::string& carriers, TPad* pad);
00126   void PlotDiffusion(const std::string& carriers, TPad* pad);
00128   void PlotTownsend(const std::string& carriers, TPad* pad);
00130   void PlotAttachment(const std::string& carriers, TPad* pad);
00132   void PlotAlphaEta(const std::string& carriers, TPad* pad);
00133
00134   // Transport parameters for electrons
00136   virtual bool ElectronVelocity(const double ex, const double ey,
00137                                 const double ez, const double bx,
00138                                 const double by, const double bz, double& vx,
00139                                 double& vy, double& vz);
00142   virtual bool ElectronVelocityFluxBulk(const double ex, const double ey,
00143                                         const double ez, const double bx,
00144                                         const double by, const double bz,
00145                                         double& wv, double& wr);
00147   virtual bool ElectronDiffusion(const double ex, const double ey,
00148                                  const double ez, const double bx,
00149                                  const double by, const double bz, double& dl,
00150                                  double& dt);
00154   virtual bool ElectronDiffusion(const double ex, const double ey,
00155                                  const double ez, const double bx,
00156                                  const double by, const double bz,
00157                                  double cov[3][3]);
00159   virtual bool ElectronTownsend(const double ex, const double ey,
00160                                 const double ez, const double bx,
00161                                 const double by, const double bz,
00162                                 double& alpha);
00164   virtual bool ElectronAttachment(const double ex, const double ey,
00165                                   const double ez, const double bx,
00166                                   const double by, const double bz,
00167                                   double& eta);
00169   virtual bool ElectronTOFIonisation(const double ex, const double ey,
00170                                      const double ez, const double bx,
00171                                      const double by, const double bz,
00172                                      double& riontof);
00174   virtual bool ElectronTOFAttachment(const double ex, const double ey,
00175                                      const double ez, const double bx,
00176                                      const double by, const double bz,
00177                                      double& ratttof);
00179   virtual bool ElectronLorentzAngle(const double ex, const double ey,
00180                                     const double ez, const double bx,
00181                                     const double by, const double bz,
00182                                     double& lor);
00184   virtual double ElectronMobility();
00185   // Microscopic electron transport properties
00186
00188   virtual double GetElectronEnergy(const double px, const double py,
00189                                    const double pz, double& vx, double& vy,
00190                                    double& vz, const int band = 0);
00193   virtual void GetElectronMomentum(const double e, double& px, double& py,
00194                                    double& pz, int& band);
00195
00197   virtual double GetElectronNullCollisionRate(const int band = 0);
00198 #endif
00199
00200 #ifdef ___GPUCOMPILE___
00201
00202   ___device___ cuda_t GetElectronCollisionRate(const cuda_t e, const int band);
00203
00204   ___device___ bool ElectronCollision(const cuda_t e, int& type, int& level,
00205                                       cuda_t& e1, cuda_t& dx, cuda_t& dy,
00206                                       cuda_t& dz, Particle* secondaries_type,
00207                                       cuda_t* secondaries_energy,
00208                                       int& num_secondaries, int& ndxc, int& band);
00209 #else
00211   virtual double GetElectronCollisionRate(const double e, const int band = 0);
00212   struct Secondary {
00213     Particle type = Particle::Electron;
```

```
00214     double energy = 0.;
00215     double time = 0.;
00216     double distance = 0.;
00217   };
00219   virtual bool ElectronCollision(const double e, int& type, int& level,
00220                                  double& e1, double& dx, double& dy, double& dz,
00221                                  std::vector<Secondary>& secondaries,
00222                                  int& band);
00223 #endif
00224
00225 #ifndef ___GPUCOMPILE___
00226   // Transport parameters for holes
00228   virtual bool HoleVelocity(const double ex, const double ey, const double ez,
00229                             const double bx, const double by, const double bz,
00230                             double& vx, double& vy, double& vz);
00232   virtual bool HoleDiffusion(const double ex, const double ey, const double ez,
00233                              const double bx, const double by, const double bz,
00234                              double& dl, double& dt);
00236   virtual bool HoleDiffusion(const double ex, const double ey, const double ez,
00237                              const double bx, const double by, const double bz,
00238                              double cov[3][3]);
00240   virtual bool HoleTownsend(const double ex, const double ey, const double ez,
00241                             const double bx, const double by, const double bz,
00242                             double& alpha);
00244   virtual bool HoleAttachment(const double ex, const double ey, const double ez,
00245                               const double bx, const double by, const double bz,
00246                               double& eta);
00248   virtual double HoleMobility();
00249
00250   // Transport parameters for ions
00252   virtual bool IonVelocity(const double ex, const double ey, const double ez,
00253                            const double bx, const double by, const double bz,
00254                            double& vx, double& vy, double& vz);
00255   bool HasIonVelocity() const { return !(m_iVel.empty() && m_iMob.empty()); }
00257   virtual bool IonDiffusion(const double ex, const double ey, const double ez,
00258                             const double bx, const double by, const double bz,
00259                             double& dl, double& dt);
00261   virtual bool IonDissociation(const double ex, const double ey,
00262                                const double ez, const double bx,
00263                                const double by, const double bz, double& diss);
00265   virtual double IonMobility();
00266
00268   virtual bool NegativeIonVelocity(const double ex, const double ey,
00269                                    const double ez, const double bx,
00270                                    const double by, const double bz, double& vx,
00271                                    double& vy, double& vz);
00273   virtual double NegativeIonMobility();
00274
00276   void SetFieldGrid(double emin, double emax, const size_t ne, bool logE,
00277                     double bmin = 0., double bmax = 0., const size_t nb = 1,
00278                     double amin = HalfPi, double amax = HalfPi,
00279                     const size_t na = 1);
00281   void SetFieldGrid(const std::vector<double>& efields,
00282                     const std::vector<double>& bfields,
00283                     const std::vector<double>& angles);
00285   void GetFieldGrid(std::vector<double>& efields, std::vector<double>& bfields,
00286                     std::vector<double>& angles);
00287
00289   bool SetElectronVelocityE(const size_t ie, const size_t ib, const size_t ia,
00290                             const double v) {
00291     return SetEntry(ie, ib, ia, "ElectronVelocityE", m_eVelE, v);
00292   }
00294   bool GetElectronVelocityE(const size_t ie, const size_t ib, const size_t ia,
00295                             double& v) {
00296     return GetEntry(ie, ib, ia, "ElectronVelocityE", m_eVelE, v);
00297   }
00299   bool SetElectronVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00300                               const double v) {
00301     return SetEntry(ie, ib, ia, "ElectronVelocityExB", m_eVelX, v);
00302   }
00304   bool GetElectronVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00305                               double& v) {
00306     return GetEntry(ie, ib, ia, "ElectronVelocityExB", m_eVelX, v);
00307   }
00309   bool SetElectronVelocityB(const size_t ie, const size_t ib, const size_t ia,
00310                             const double v) {
00311     return SetEntry(ie, ib, ia, "ElectronVelocityB", m_eVelB, v);
00312   }
00314   bool GetElectronVelocityB(const size_t ie, const size_t ib, const size_t ia,
00315                             double& v) {
00316     return GetEntry(ie, ib, ia, "ElectronVelocityB", m_eVelB, v);
00317   }
00319   bool SetElectronFluxVelocity(const size_t ie, const size_t ib,
00320                                const size_t ia, const double v) {
00321     return SetEntry(ie, ib, ia, "ElectronFluxVelocity", m_eVelWv, v);
00322   }
00324   bool GetElectronFluxVelocity(const size_t ie, const size_t ib,
```

```
00325                               const size_t ia, double& v) {
00326   return GetEntry(ie, ib, ia, "ElectronFluxVelocity", m_eVelWv, v);
00327 }
00329 bool SetElectronBulkVelocity(const size_t ie, const size_t ib,
00330                               const size_t ia, const double v) {
00331   return SetEntry(ie, ib, ia, "ElectronBulkVelocity", m_eVelWr, v);
00332 }
00334 bool GetElectronBulkVelocity(const size_t ie, const size_t ib,
00335                               const size_t ia, double& v) {
00336   return GetEntry(ie, ib, ia, "ElectronBulkVelocity", m_eVelWr, v);
00337 }
00339 bool SetElectronLongitudinalDiffusion(const size_t ie, const size_t ib,
00340                                       const size_t ia, const double dl) {
00341   return SetEntry(ie, ib, ia, "ElectronLongitudinalDiffusion", m_eDifL, dl);
00342 }
00344 bool GetElectronLongitudinalDiffusion(const size_t ie, const size_t ib,
00345                                       const size_t ia, double& dl) {
00346   return GetEntry(ie, ib, ia, "ElectronLongitudinalDiffusion", m_eDifL, dl);
00347 }
00349 bool SetElectronTransverseDiffusion(const size_t ie, const size_t ib,
00350                                     const size_t ia, const double dt) {
00351   return SetEntry(ie, ib, ia, "ElectronTransverseDiffusion", m_eDifT, dt);
00352 }
00354 bool GetElectronTransverseDiffusion(const size_t ie, const size_t ib,
00355                                     const size_t ia, double& dt) {
00356   return GetEntry(ie, ib, ia, "ElectronTransverseDiffusion", m_eDifT, dt);
00357 }
00359 bool SetElectronTownsend(const size_t ie, const size_t ib, const size_t ia,
00360                          const double alpha) {
00361   return SetEntry(ie, ib, ia, "ElectronTownsend", m_eAlp, alpha);
00362 }
00364 bool GetElectronTownsend(const size_t ie, const size_t ib, const size_t ia,
00365                          double& alpha) {
00366   return GetEntry(ie, ib, ia, "ElectronTownsend", m_eAlp, alpha);
00367 }
00369 bool SetElectronAttachment(const size_t ie, const size_t ib, const size_t ia,
00370                            const double eta) {
00371   return SetEntry(ie, ib, ia, "ElectronAttachment", m_eAtt, eta);
00372 }
00374 bool GetElectronAttachment(const size_t ie, const size_t ib, const size_t ia,
00375                            double& eta) {
00376   return GetEntry(ie, ib, ia, "ElectronAttachment", m_eAtt, eta);
00377 }
00379 bool SetElectronTOFIonisation(const size_t ie, const size_t ib,
00380                               const size_t ia, const double v) {
00381   return SetEntry(ie, ib, ia, "ElectronTOFIonisation", m_eRIon, v);
00382 }
00384 bool GetElectronTOFIonisation(const size_t ie, const size_t ib,
00385                               const size_t ia, double& v) {
00386   return GetEntry(ie, ib, ia, "ElectronTOFIonisation", m_eRIon, v);
00387 }
00389 bool SetElectronTOFAttachment(const size_t ie, const size_t ib,
00390                               const size_t ia, const double v) {
00391   return SetEntry(ie, ib, ia, "ElectronTOFAttachment", m_eRAtt, v);
00392 }
00394 bool GetElectronTOFAttachment(const size_t ie, const size_t ib,
00395                               const size_t ia, double& v) {
00396   return GetEntry(ie, ib, ia, "ElectronTOFAttachment", m_eRAtt, v);
00397 }
00398
00400 bool SetElectronLorentzAngle(const size_t ie, const size_t ib,
00401                              const size_t ia, const double lor) {
00402   return SetEntry(ie, ib, ia, "ElectronLorentzAngle", m_eLor, lor);
00403 }
00405 bool GetElectronLorentzAngle(const size_t ie, const size_t ib,
00406                              const size_t ia, double& lor) {
00407   return GetEntry(ie, ib, ia, "ElectronLorentzAngle", m_eLor, lor);
00408 }
00409
00411 bool SetHoleVelocityE(const size_t ie, const size_t ib, const size_t ia,
00412                       const double v) {
00413   return SetEntry(ie, ib, ia, "HoleVelocityE", m_hVelE, v);
00414 }
00416 bool GetHoleVelocityE(const size_t ie, const size_t ib, const size_t ia,
00417                       double& v) {
00418   return GetEntry(ie, ib, ia, "HoleVelocityE", m_hVelE, v);
00419 }
00421 bool SetHoleVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00422                         const double v) {
00423   return SetEntry(ie, ib, ia, "HoleVelocityExB", m_hVelX, v);
00424 }
00426 bool GetHoleVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00427                         double& v) {
00428   return GetEntry(ie, ib, ia, "HoleVelocityExB", m_hVelX, v);
00429 }
00431 bool SetHoleVelocityB(const size_t ie, const size_t ib, const size_t ia,
00432                       const double v) {
```

```
00433   return SetEntry(ie, ib, ia, "HoleVelocityB", m_hVelB, v);
00434 }
00436 bool GetHoleVelocityB(const size_t ie, const size_t ib, const size_t ia,
00437                       double& v) {
00438   return GetEntry(ie, ib, ia, "HoleVelocityB", m_hVelB, v);
00439 }
00440
00442 bool SetHoleLongitudinalDiffusion(const size_t ie, const size_t ib,
00443                                   const size_t ia, const double dl) {
00444   return SetEntry(ie, ib, ia, "HoleLongitudinalDiffusion", m_hDifL, dl);
00445 }
00447 bool GetHoleLongitudinalDiffusion(const size_t ie, const size_t ib,
00448                                   const size_t ia, double& dl) {
00449   return GetEntry(ie, ib, ia, "HoleLongitudinalDiffusion", m_hDifL, dl);
00450 }
00452 bool SetHoleTransverseDiffusion(const size_t ie, const size_t ib,
00453                                 const size_t ia, const double dt) {
00454   return SetEntry(ie, ib, ia, "HoleTransverseDiffusion", m_hDifT, dt);
00455 }
00457 bool GetHoleTransverseDiffusion(const size_t ie, const size_t ib,
00458                                 const size_t ia, double& dt) {
00459   return GetEntry(ie, ib, ia, "HoleTransverseDiffusion", m_hDifT, dt);
00460 }
00462 bool SetHoleTownsend(const size_t ie, const size_t ib, const size_t ia,
00463                      const double alpha) {
00464   return SetEntry(ie, ib, ia, "HoleTownsend", m_hAlp, alpha);
00465 }
00467 bool GetHoleTownsend(const size_t ie, const size_t ib, const size_t ia,
00468                      double& alpha) {
00469   return GetEntry(ie, ib, ia, "HoleTownsend", m_hAlp, alpha);
00470 }
00472 bool SetHoleAttachment(const size_t ie, const size_t ib, const size_t ia,
00473                        const double eta) {
00474   return SetEntry(ie, ib, ia, "HoleAttachment", m_hAtt, eta);
00475 }
00477 bool GetHoleAttachment(const size_t ie, const size_t ib, const size_t ia,
00478                        double& eta) {
00479   return GetEntry(ie, ib, ia, "HoleAttachment", m_hAtt, eta);
00480 }
00481
00486 bool SetIonMobility(const std::vector<double>& fields,
00487                     const std::vector<double>& mobilities,
00488                     const bool negativeIons = false);
00490 bool SetIonMobility(const size_t ie, const size_t ib, const size_t ia,
00491                     const double mu);
00493 bool GetIonMobility(const size_t ie, const size_t ib, const size_t ia,
00494                     double& mu) {
00495   return GetEntry(ie, ib, ia, "IonMobility", m_iMob, mu);
00496 }
00497
00499 bool SetIonLongitudinalDiffusion(const size_t ie, const size_t ib,
00500                                  const size_t ia, const double dl) {
00501   return SetEntry(ie, ib, ia, "IonLongitudinalDiffusion", m_iDifL, dl);
00502 }
00504 bool GetIonLongitudinalDiffusion(const size_t ie, const size_t ib,
00505                                  const size_t ia, double& dl) {
00506   return GetEntry(ie, ib, ia, "IonLongitudinalDiffusion", m_iDifL, dl);
00507 }
00509 bool SetIonTransverseDiffusion(const size_t ie, const size_t ib,
00510                                const size_t ia, const double dt) {
00511   return SetEntry(ie, ib, ia, "IonTransverseDiffusion", m_iDifT, dt);
00512 }
00514 bool GetIonTransverseDiffusion(const size_t ie, const size_t ib,
00515                                const size_t ia, double& dt) {
00516   return GetEntry(ie, ib, ia, "IonTransverseDiffusion", m_iDifT, dt);
00517 }
00519 bool SetIonDissociation(const size_t ie, const size_t ib, const size_t ia,
00520                         const double diss) {
00521   return SetEntry(ie, ib, ia, "IonDissociation", m_iDis, diss);
00522 }
00524 bool GetIonDissociation(const size_t ie, const size_t ib, const size_t ia,
00525                         double& diss) {
00526   return GetEntry(ie, ib, ia, "IonDissociation", m_iDis, diss);
00527 }
00528
00530 bool SetNegativeIonMobility(const size_t ie, const size_t ib, const size_t ia,
00531                             const double mu);
00533 bool GetNegativeIonMobility(const size_t ie, const size_t ib, const size_t ia,
00534                             double& mu) {
00535   return GetEntry(ie, ib, ia, "NegativeIonMobility", m_nMob, mu);
00536 }
00537
00539 virtual void ResetTables();
00540
00541 void ResetElectronVelocity() {
00542   m_eVelE.clear();
00543   m_eVelB.clear();
```

```
00544     m_eVelX.clear();
00545     m_eVelWv.clear();
00546     m_eVelWr.clear();
00547   }
00548   void ResetElectronDiffusion() {
00549     m_eDifL.clear();
00550     m_eDifT.clear();
00551     m_eDifM.clear();
00552   }
00553   void ResetElectronTownsend() { m_eAlp.clear(); }
00554   void ResetElectronAttachment() { m_eAtt.clear(); }
00555   void ResetElectronTOFRates() {
00556     m_eRIon.clear();
00557     m_eRAtt.clear();
00558   }
00559   void ResetElectronLorentzAngle() { m_eLor.clear(); }
00560
00561   void ResetHoleVelocity() {
00562     m_hVelE.clear();
00563     m_hVelB.clear();
00564     m_hVelX.clear();
00565   }
00566   void ResetHoleDiffusion() {
00567     m_hDifL.clear();
00568     m_hDifT.clear();
00569     m_hDifM.clear();
00570   }
00571   void ResetHoleTownsend() { m_hAlp.clear(); }
00572   void ResetHoleAttachment() { m_hAtt.clear(); }
00573
00574   void ResetIonMobility() {
00575     m_iMob.clear();
00576     m_iVel.clear();
00577   }
00578   void ResetIonDiffusion() {
00579     m_iDifL.clear();
00580     m_iDifT.clear();
00581   }
00582   void ResetIonDissociation() { m_iDis.clear(); }
00583   void ResetNegativeIonMobility() {
00584     m_nMob.clear();
00585     m_nVel.clear();
00586   }
00587
00588   void VelocityFromMobility(
00589       const std::vector<std::vector<std::vector<double> > >& mob,
00590       std::vector<std::vector<std::vector<double> > >& vel);
00591
00594   void SetExtrapolationMethodVelocity(const std::string& extrLow,
00595                                       const std::string& extrHigh);
00596   void SetExtrapolationMethodDiffusion(const std::string& extrLow,
00597                                        const std::string& extrHigh);
00598   void SetExtrapolationMethodTownsend(const std::string& extrLow,
00599                                       const std::string& extrHigh);
00600   void SetExtrapolationMethodAttachment(const std::string& extrLow,
00601                                         const std::string& extrHigh);
00602   void SetExtrapolationMethodIonMobility(const std::string& extrLow,
00603                                          const std::string& extrHigh);
00604   void SetExtrapolationMethodIonDissociation(const std::string& extrLow,
00605                                              const std::string& extrHigh);
00606
00608   void SetInterpolationMethodVelocity(const unsigned int intrp);
00609   void SetInterpolationMethodDiffusion(const unsigned int intrp);
00610   void SetInterpolationMethodTownsend(const unsigned int intrp);
00611   void SetInterpolationMethodAttachment(const unsigned int intrp);
00612   void SetInterpolationMethodIonMobility(const unsigned int intrp);
00613   void SetInterpolationMethodIonDissociation(const unsigned int intrp);
00614
00615   // Scaling of fields and transport parameters.
00616   virtual double ScaleElectricField(const double e) const { return e; }
00617   virtual double UnScaleElectricField(const double e) const { return e; }
00618   virtual double ScaleVelocity(const double v) const { return v; }
00619   virtual double ScaleDiffusion(const double d) const { return d; }
00620   virtual double ScaleDiffusionTensor(const double d) const { return d; }
00621   virtual double ScaleTownsend(const double alpha) const { return alpha; }
00622   virtual double ScaleAttachment(const double eta) const { return eta; }
00623   virtual double ScaleLorentzAngle(const double lor) const { return lor; }
00624   virtual double ScaleDissociation(const double diss) const { return diss; }
00625
00626   // Optical properties
00628   virtual bool GetOpticalDataRange(double& emin, double& emax,
00629                                    const unsigned int i = 0);
00631   virtual bool GetDielectricFunction(const double e, double& eps1, double& eps2,
00632                                      const unsigned int i = 0);
00633   // Get the photoabsorption cross-section [cm2] at a given energy.
00634   virtual bool GetPhotoAbsorptionCrossSection(const double e, double& sigma,
00635                                               const unsigned int i = 0);
```

```
00636   virtual double GetPhotonCollisionRate(const double e);
00637   virtual bool PhotonCollision(const double e, int& type, int& level,
00638                                double& e1, double& ctheta,
00639                                std::vector<Secondary>& secondaries);
00640
00642   void EnableDebugging() { m_debug = true; }
00643   void DisableDebugging() { m_debug = false; }
00644
00646   virtual double CreateGPUTransferObject(MediumGPU*& med_gpu);
00647
00648  protected:
00649   std::string m_className = "Medium";
00650
00651   static int m_idCounter;
00652
00653   // Number of components
00654   unsigned int m_nComponents = 1;
00655   // Name
00656   std::string m_name = "";
00657   // Temperature [K]
00658   double m_temperature = 293.15;
00659   // Pressure [Torr]
00660   double m_pressure = 760.;
00661   // Static dielectric constant
00662   double m_epsilon = 1.;
00663   // (Effective) atomic number Z
00664   double m_z = 1.;
00665   // Atomic weight A
00666   double m_a = 0.;
00667   // Number density [cm-3]
00668   double m_density = 0.;
00669 #endif
00670
00671   // Id number
00672   int m_id;
00673
00674   // Transport flags
00675   bool m_driftable = false;
00676   bool m_microscopic = false;
00677   bool m_ionisable = false;
00678
00679 #ifndef ___GPUCOMPILE___
00680
00681   // W value
00682   double m_w = 0.;
00683   // Fano factor
00684   double m_fano = 0.;
00685
00686   // Update flag
00687   bool m_isChanged = true;
00688
00689   // Switch on/off debugging messages
00690   bool m_debug = false;
00691
00692   // Tables of transport parameters
00693   bool m_tab2d = false;
00694
00695   // Field grids
00696   std::vector<double> m_eFields;
00697   std::vector<double> m_bFields;
00698   std::vector<double> m_bAngles;
00699
00700   // Electrons
00701   std::vector<std::vector<std::vector<double> > > m_eVelE;
00702   std::vector<std::vector<std::vector<double> > > m_eVelX;
00703   std::vector<std::vector<std::vector<double> > > m_eVelB;
00704   std::vector<std::vector<std::vector<double> > > m_eDifL;
00705   std::vector<std::vector<std::vector<double> > > m_eDifT;
00706   std::vector<std::vector<std::vector<double> > > m_eAlp;
00707   std::vector<std::vector<std::vector<double> > > m_eAtt;
00708   std::vector<std::vector<std::vector<double> > > m_eLor;
00709   std::vector<std::vector<std::vector<double> > > m_eVelWv;
00710   std::vector<std::vector<std::vector<double> > > m_eVelWr;
00711   std::vector<std::vector<std::vector<double> > > m_eRIon;
00712   std::vector<std::vector<std::vector<double> > > m_eRAtt;
00713
00714   std::vector<std::vector<std::vector<std::vector<double> > > > m_eDifM;
00715
00716   // Holes
00717   std::vector<std::vector<std::vector<double> > > m_hVelE;
00718   std::vector<std::vector<std::vector<double> > > m_hVelX;
00719   std::vector<std::vector<std::vector<double> > > m_hVelB;
00720   std::vector<std::vector<std::vector<double> > > m_hDifL;
00721   std::vector<std::vector<std::vector<double> > > m_hDifT;
00722   std::vector<std::vector<std::vector<double> > > m_hAlp;
00723   std::vector<std::vector<std::vector<double> > > m_hAtt;
00724
```

```
00725   std::vector<std::vector<std::vector<std::vector<double> > > > m_hDifM;
00726
00727   // Ions
00728   std::vector<std::vector<std::vector<double> > > m_iMob;
00729   std::vector<std::vector<std::vector<double> > > m_iVel;
00730   std::vector<std::vector<std::vector<double> > > m_iDifL;
00731   std::vector<std::vector<std::vector<double> > > m_iDifT;
00732   std::vector<std::vector<std::vector<double> > > m_iDis;
00733   // Negative ions
00734   std::vector<std::vector<std::vector<double> > > m_nMob;
00735   std::vector<std::vector<std::vector<double> > > m_nVel;
00736
00737   // Thresholds for Townsend, attachment and dissociation coefficients.
00738   unsigned int m_eThrAlp = 0;
00739   unsigned int m_eThrAtt = 0;
00740   unsigned int m_hThrAlp = 0;
00741   unsigned int m_hThrAtt = 0;
00742   unsigned int m_iThrDis = 0;
00743
00744   // Extrapolation methods (TODO: enum).
00745   std::pair<unsigned int, unsigned int> m_extrVel = {0, 1};
00746   std::pair<unsigned int, unsigned int> m_extrDif = {0, 1};
00747   std::pair<unsigned int, unsigned int> m_extrAlp = {0, 1};
00748   std::pair<unsigned int, unsigned int> m_extrAtt = {0, 1};
00749   std::pair<unsigned int, unsigned int> m_extrLor = {0, 1};
00750   std::pair<unsigned int, unsigned int> m_extrMob = {0, 1};
00751   std::pair<unsigned int, unsigned int> m_extrDis = {0, 1};
00752
00753   // Interpolation methods
00754   unsigned int m_intpVel = 2;
00755   unsigned int m_intpDif = 2;
00756   unsigned int m_intpAlp = 2;
00757   unsigned int m_intpAtt = 2;
00758   unsigned int m_intpLor = 2;
00759   unsigned int m_intpMob = 2;
00760   unsigned int m_intpDis = 2;
00761
00762   bool Velocity(const double ex, const double ey, const double ez,
00763                 const double bx, const double by, const double bz,
00764                 const std::vector<std::vector<std::vector<double> > >& velE,
00765                 const std::vector<std::vector<std::vector<double> > >& velB,
00766                 const std::vector<std::vector<std::vector<double> > >& velX,
00767                 const double q, double& vx, double& vy, double& vz) const;
00768   bool VelocityFluxBulk(
00769       const double ex, const double ey, const double ez, const double bx,
00770       const double by, const double bz,
00771       const std::vector<std::vector<std::vector<double> > >& velWv,
00772       const std::vector<std::vector<std::vector<double> > >& velWr,
00773       double& wv, double& wr) const;
00774   static void Langevin(const double ex, const double ey, const double ez,
00775                        double bx, double by, double bz, const double mu,
00776                        double& vx, double& vy, double& vz);
00777   static void Langevin(const double ex, const double ey, const double ez,
00778                        double bx, double by, double bz, const double mu,
00779                        const double muH, double& vx, double& vy, double& vz);
00780   bool Diffusion(const double ex, const double ey, const double ez,
00781                  const double bx, const double by, const double bz,
00782                  const std::vector<std::vector<std::vector<double> > >& difL,
00783                  const std::vector<std::vector<std::vector<double> > >& difT,
00784                  double& dl, double& dt) const;
00785   bool Diffusion(
00786       const double ex, const double ey, const double ez, const double bx,
00787       const double by, const double bz,
00788       const std::vector<std::vector<std::vector<std::vector<double> > > >& diff,
00789       double cov[3][3]) const;
00790   bool Alpha(const double ex, const double ey, const double ez, const double bx,
00791              const double by, const double bz,
00792              const std::vector<std::vector<std::vector<double> > >& tab,
00793              unsigned int intp, const unsigned int thr,
00794              const std::pair<unsigned int, unsigned int>& extr,
00795              double& alpha) const;
00796   double GetAngle(const double ex, const double ey, const double ez,
00797                   const double bx, const double by, const double bz,
00798                   const double e, const double b) const;
00799   bool Interpolate(const double e, const double b, const double a,
00800                    const std::vector<std::vector<std::vector<double> > >& table,
00801                    double& y, const unsigned int intp,
00802                    const std::pair<unsigned int, unsigned int>& extr,
00803                    const bool logval = false) const;
00804
00805   double Interpolate1D(const double e, const std::vector<double>& table,
00806                        const std::vector<double>& fields,
00807                        const unsigned int intpMeth,
00808                        const std::pair<unsigned int, unsigned int>& extr,
00809                        const bool logval = false) const;
00810
00811   bool SetEntry(const size_t i, const size_t j, const size_t k,
```

```
00812                 const std::string& fcn,
00813                 std::vector<std::vector<std::vector<double> > >& tab,
00814                 const double val);
00815   bool GetEntry(const size_t i, const size_t j, const size_t k,
00816                 const std::string& fcn,
00817                 const std::vector<std::vector<std::vector<double> > >& tab,
00818                 double& val) const;
00819
00820   void SetExtrapolationMethod(const std::string& low, const std::string& high,
00821                               std::pair<unsigned int, unsigned int>& extr,
00822                               const std::string& fcn);
00823   bool GetExtrapolationIndex(std::string str, unsigned int& nb) const;
00824   size_t SetThreshold(
00825       const std::vector<std::vector<std::vector<double> > >& tab) const;
00826
00827   void Clone(std::vector<std::vector<std::vector<double> > >& tab,
00828              const std::vector<double>& efields,
00829              const std::vector<double>& bfields,
00830              const std::vector<double>& angles, const unsigned int intp,
00831              const std::pair<unsigned int, unsigned int>& extr,
00832              const double init, const std::string& label);
00833   void Clone(std::vector<std::vector<std::vector<std::vector<double> > > >& tab,
00834              const size_t n, const std::vector<double>& efields,
00835              const std::vector<double>& bfields,
00836              const std::vector<double>& angles, const unsigned int intp,
00837              const std::pair<unsigned int, unsigned int>& extr,
00838              const double init, const std::string& label);
00839
00840   void Init(const size_t nE, const size_t nB, const size_t nA,
00841             std::vector<std::vector<std::vector<double> > >& tab,
00842             const double val);
00843   void Init(const size_t nE, const size_t nB, const size_t nA, const size_t nT,
00844             std::vector<std::vector<std::vector<std::vector<double> > > >& tab,
00845             const double val);
00846
00847 #else
00848
00849 #include "MediumMagboltz.hh"
00850
00851   friend class MediumGas;
00852   friend class MediumMagboltz;
00853
00854   // enum to mimic polymorphism
00855   enum class MediumType { Medium = 0, MediumGas, MediumMagboltz };
00856
00857   MediumType m_MediumType{MediumType::Medium};
00858
00859 #endif
00860 };
```

## GARFIELD_CLASS_NAME() [3/4]

class Garfield::GARFIELD_CLASS_NAME (
Sensor )

Sensor

Default constructor.

Destructor.

Constructor from a single component.

Add a component.

Get the number of components attached to the sensor.

Retrieve the pointer to a given component.

Activate/deactivate a given component.

Activate/deactivate use of the magnetic field of a given component.

Does the sensor have a non-zero magnetic field?

Add an electrode.

Get the number of electrodes attached to the sensor.

Remove all electrodes.

Remove all components, electrodes and reset the sensor.

Get the drift field and potential at (x, y, z).

Get the drift field at (x, y, z).

Get the magnetic field at (x, y, z).

Get the weighting field at (x, y, z).

Get the weighting potential at (x, y, z).

Get the delayed weighting potential at (x, y, z).

Get the medium at (x, y, z).

Switch debugging messages on/off.

Set the user area to the default.

Set the user area explicitly.

Return the current user area.

Check if a point is inside the user area.

Check if a point is inside an active medium and inside the user area.

Return the voltage range.

Start a new event, when computing the average signal over multiple events.

Reset signals and induced charges of all electrodes.

Set the time window and binning for the signal calculation.

Parameters

| | |
|---|---|
| tstart | start time [ns] |
| tstep | bin width [ns] |
| nsteps | number of bins |

---

Retrieve the time window and binning.

Compute the component of the signal due to the delayed weighting field.

Set the points in time at which to evaluate the delayed weighting field.

Set the number of points to be used when averaging the delayed signal vector over a time bin (default: 0). The averaging is done with a $2 \times navg + 1$ point Newton-Raphson integration.

Set/override the signal in a given time bin explicitly.

Set/override the signal.

Retrieve the total signal for a given electrode and time bin.

Retrieve the electron signal for a given electrode and time bin.

Retrieve the prompt signal for a given electrode and time bin.

Retrieve the delayed signal for a given electrode and time bin.

Retrieve the electron signal for a given electrode and time bin.

Retrieve the ion or hole signal for a given electrode and time bin.

Retrieve the delayed electron signal for a given electrode and time bin.

Retrieve the delayed ion/hole signal for a given electrode and time bin.

Calculated using the weighting potentials at the start and end points.

Set the function to be used for evaluating the transfer function.

Set the points to be used for interpolating the transfer function.

Set the transfer function using a Shaper object.

Evaluate the transfer function at a given time.

Print some information about the presently set transfer function.

Plot the presently set transfer function.

Cache integral and FFT of the transfer function instead of recomputing it at every call (default: on).

Convolute the induced current on a given electrode with the transfer function.

Convolute all induced currents with the transfer function.

Replace the signal on a given electrode by its integral.

Replace the signals on all electrodes curve by their integrals.

Return whether the signal has been integrated/convoluted.

Delay the signal and subtract an attenuated copy (modelling a constant fraction discriminator).

$$v_{out} = v_{in}\,(t - t_d) - f v_{in}.$$

Set the function to be used for evaluating the noise component.

Add noise to the induced signal.

Add white noise to the induced signal, given a desired output ENC.

Parameters

| | |
|---|---|
| label | name of the electrode |
| enc | Equivalent Noise Charge, in electrons. |
| poisson | flag to sample the number of noise pulses from a Poisson distribution. Otherwise the noise charge in each bin is sampled from a Gaussian distribution. |
| q0 | amplitude of the noise delta pulses (in electrons). |

Add white noise to the induced signals on all electrodes.

Determine the threshold crossings of the current signal curve.

Parameters

| | |
|---|---|
| thr | threshold value |
| label | electrode for which to compute the threshold crossings |
| n | number of threshold crossings |

Get the number of threshold crossings (after having called ComputeThresholdCrossings).

Retrieve the time and type of a given threshold crossing (after having called ComputeThresholdCrossings.

Parameters

| | |
|---|---|
| i | index |
| time | threshold crossing time [ns] |
| level | threshold (should correspond to the value requested). |
| rise | flag whether the crossing is on a rising or falling slope. |

Calculate the signal from a drift line using the weighting potential method.

Calculate the signal from an avalanche using the weighting potential method.

Calculate the signal from a drift line using the weighting field method.

Calculate the signal from a drift line using the weighting field method, given the drift velocities at each drift line point.

Plot the induced signal.

Exporting induced signal to a csv file.

Add the induced charge from a charge carrier drift between two points.

Determine whether a line between two points crosses a wire, calls Component::CrossedWire.

Determine whether a point is in the trap radius of a wire.

Determine whether a line between two points crosses a plane, calls Component::CrossedPlane.

Integrate the electric field flux through a line from (x0,y0,z0) to (x1,y1,z1) along a direction (xp,yp,zp).

Create and initialise GPU Transfer class

Mutex.

Components

Electrodes

Definition at line 1 of file Sensor.hh.

```
00044                                {
00045  public:
00047   GARFIELD_CLASS_NAME(Sensor)() = default;
00049   ~GARFIELD_CLASS_NAME(Sensor)() = default;
00050
00051 #ifndef ___GPUCOMPILE___
00053   Sensor(Component* comp);
00055   void AddComponent(Component* comp);
00057   size_t GetNumberOfComponents() const { return m_components.size(); }
00059   Component* GetComponent(const unsigned int i);
00061   void EnableComponent(const unsigned int i, const bool on);
00063   void EnableMagneticField(const unsigned int i, const bool on);
00065   bool HasMagneticField() const;
00066
00068   void AddElectrode(Component* comp, const std::string& label);
00070   size_t GetNumberOfElectrodes() const { return m_electrodes.size(); }
00072   void ClearElectrodes();
00074   void Clear();
00075
00077   void ElectricField(const double x, const double y, const double z, double& ex,
00078                      double& ey, double& ez, double& v, Medium*& medium,
00079                      int& status);
00080 #endif
00082   ___DEVICE___
00083   void ElectricField(const double x, const double y, const double z, double& ex,
00084                      double& ey, double& ez,
00085                      GARFIELD_CLASS_NAME(Medium) * &medium,
00086                      int& status) ___GPUCONST___;
00087
00088 #ifndef ___GPUCOMPILE___
00090   void MagneticField(const double x, const double y, const double z, double& bx,
00091                      double& by, double& bz, int& status);
00092
00094   void WeightingField(const double x, const double y, const double z,
00095                       double& wx, double& wy, double& wz,
00096                       const std::string& label);
00098   double WeightingPotential(const double x, const double y, const double z,
00099                             const std::string& label);
00100
00102   double DelayedWeightingPotential(const double x, const double y,
00103                                    const double z, const double t,
00104                                    const std::string& label);
00105
00107   Medium* GetMedium(const double x, const double y, const double z);
00108
00110   void EnableDebugging(const bool on = true) { m_debug = on; }
00111
00113   bool SetArea(const bool verbose = false);
00115   bool SetArea(const double xmin, const double ymin, const double zmin,
00116                const double xmax, const double ymax, const double zmax);
00118   bool GetArea(double& xmin, double& ymin, double& zmin, double& xmax,
00119                double& ymax, double& zmax);
00120 #endif
00122   ___DEVICE___
00123   bool IsInArea(const double x, const double y, const double z) ___GPUCONST___;
00124
00125 #ifndef ___GPUCOMPILE___
00127   bool IsInside(const double x, const double y, const double z);
00128
00130   bool GetVoltageRange(double& vmin, double& vmax);
00131
00133   void NewSignal() { ++m_nEvents; }
00135   void ClearSignal();
00136 #else
00138   ___device___ void AddSignal(const double q, const double t0, const double t1,
00139                               const double x0, const double y0, const double z0,
00140                               const double x1, const double y1, const double z1,
00141                               const bool integrateWeightingField,
00142                               const bool useWeightingPotential,
00143                               const int particle_idx);
00144 #endif
00145
00146 #ifndef ___GPUCOMPILE___
00152   void SetTimeWindow(const double tstart, const double tstep,
00153                      const unsigned int nsteps);
00155   void GetTimeWindow(double& tstart, double& tstep,
00156                      unsigned int& nsteps) const {
```

```
00157   tstart = m_tStart;
00158   tstep = m_tStep;
00159   nsteps = m_nTimeBins;
00160 }
00161
00163 void EnableDelayedSignal(const bool on = true) { m_delayedSignal = on; }
00165 void SetDelayedSignalTimes(const std::vector<double>& ts);
00170 void SetDelayedSignalAveragingOrder(const unsigned int navg) {
00171   m_nAvgDelayedSignal = navg;
00172 }
00173
00175 void SetSignal(const std::string& label, const unsigned int bin,
00176               const double signal);
00178 void SetSignal(const std::string& label, const std::vector<double>& ts,
00179               const std::vector<double>& is);
00181 double GetSignal(const std::string& label, const unsigned int bin);
00183 double GetSignal(const std::string& label, const unsigned int bin,
00184                 const int comp);
00186 double GetPromptSignal(const std::string& label, const unsigned int bin);
00188 double GetDelayedSignal(const std::string& label, const unsigned int bin);
00190 double GetElectronSignal(const std::string& label, const unsigned int bin);
00192 double GetIonSignal(const std::string& label, const unsigned int bin);
00194 double GetDelayedElectronSignal(const std::string& label,
00195                                 const unsigned int bin);
00197 double GetDelayedIonSignal(const std::string& label, const unsigned int bin);
00199 double GetInducedCharge(const std::string& label);
00200
00202 void SetTransferFunction(std::function<double(double)>);
00204 void SetTransferFunction(const std::vector<double>& times,
00205                          const std::vector<double>& values);
00207 void SetTransferFunction(Shaper& shaper);
00209 double GetTransferFunction(const double t);
00211 void PrintTransferFunction();
00213 void PlotTransferFunction();
00216 void EnableTransferFunctionCache(const bool on = true) {
00217   m_cacheTransferFunction = on;
00218 }
00221 bool ConvoluteSignal(const std::string& label, const bool fft = false);
00223 bool ConvoluteSignals(const bool fft = false);
00225 bool IntegrateSignal(const std::string& label);
00227 bool IntegrateSignals();
00229 bool IsIntegrated(const std::string& label) const;
00230
00236 bool DelayAndSubtractFraction(const double td, const double f);
00238 void SetNoiseFunction(double (*f)(double t));
00240 void AddNoise(const bool total = true, const bool electron = false,
00241              const bool ion = false);
00250 void AddWhiteNoise(const std::string& label, const double enc,
00251                   const bool poisson = true, const double q0 = 1.);
00253 void AddWhiteNoise(const double enc, const bool poisson = true,
00254                   const double q0 = 1.);
00255
00261 bool ComputeThresholdCrossings(const double thr, const std::string& label,
00262                                int& n);
00265 size_t GetNumberOfThresholdCrossings() const {
00266   return m_thresholdCrossings.size();
00267 }
00275 bool GetThresholdCrossing(const unsigned int i, double& time, double& level,
00276                           bool& rise) const;
00277
00280 void AddSignalWeightingPotential(
00281     const double q, const std::vector<double>& ts,
00282     const std::vector<std::array<double, 3> >& xs);
00285 void AddSignalWeightingPotential(
00286     const double q, const std::vector<double>& ts,
00287     const std::vector<std::array<double, 3> >& xs,
00288     const std::vector<double>& qs);
00291 void AddSignalWeightingField(const double q, const std::vector<double>& ts,
00292                              const std::vector<std::array<double, 3> >& xs,
00293                              const bool integrateWeightingField);
00297 void AddSignalWeightingField(const double q, const std::vector<double>& ts,
00298                              const std::vector<std::array<double, 3> >& xs,
00299                              const std::vector<std::array<double, 3> >& vs,
00300                              const std::vector<double>& ns, const int navg);
00301
00303 void PlotSignal(const std::string& label, TPad* pad,
00304                const std::string optTotal = "t",
00305                const std::string optPrompt = "",
00306                const std::string optDelayed = "");
00308 void ExportSignal(const std::string& label, const std::string& filename,
00309                  const bool chargeCariers = false) const;
00310
00312 void AddInducedCharge(const double q, const double x0, const double y0,
00313                       const double z0, const double x1, const double y1,
00314                       const double z1);
00315
00318 bool CrossedWire(const double x0, const double y0, const double z0,
```

```
00319                 const double x1, const double y1, const double z1,
00320                 double& xc, double& yc, double& zc, const bool centre,
00321                 double& rc);
00323   bool InTrapRadius(const double q0, const double x0, const double y0,
00324                     const double z0, double& xw, double& yw, double& rw);
00327   bool CrossedPlane(const double x0, const double y0, const double z0,
00328                     const double x1, const double y1, const double z1,
00329                     double& xc, double& yc, double& zc);
00330
00333   double IntegrateFluxLine(const double x0, const double y0, const double z0,
00334                            const double x1, const double y1, const double z1,
00335                            const double xp, const double yp, const double zp,
00336                            const unsigned int nI, const int isign = 0);
00337   // TODO!
00338   double GetTotalInducedCharge(const std::string& label);
00339
00340   double StepSizeHint();
00341
00343   double CreateGPUTransferObject(SensorGPU*& sensor_gpu);
00344 #if USEGPU
00346   void TransferGPUElectrodeSignals(SensorGPU*& sensor_gpu);
00347 #endif
00348
00349  private:
00350   std::string m_className = "Sensor";
00352   std::mutex m_mutex;
00353 #endif
00354
00356 #ifdef ___GPUCOMPILE___
00357   ComponentGPU** m_components = nullptr;
00358   size_t m_numComponents;
00359   friend class Sensor;
00360 #else
00361   std::vector<std::tuple<Component*, bool, bool> > m_components;
00362 #endif
00363
00364 #ifndef ___GPUCOMPILE___
00365   struct Electrode {
00366     Component* comp;
00367     std::string label;
00368     std::vector<double> signal;
00369     std::vector<double> delayedSignal;
00370     std::vector<double> electronSignal;
00371     std::vector<double> ionSignal;
00372     std::vector<double> delayedElectronSignal;
00373     std::vector<double> delayedIonSignal;
00374     double charge;
00375     bool integrated;
00376   };
00378   std::vector<Electrode> m_electrodes;
00379 #else
00380   struct ElectrodeGPU {
00381     ComponentGPU* comp;
00382     int label;
00383     double* signal;
00384   };
00385
00386   ElectrodeGPU* m_electrodes = nullptr;
00387   size_t m_numElectrodes;
00388 #endif
00389
00390   // Time window for signals
00391   double m_tStart = 0.;
00392   double m_tStep = 10.;
00393   unsigned int m_nTimeBins = 200;
00394   unsigned int m_nEvents = 0;
00395 #ifndef ___GPUCOMPILE___
00396   bool m_delayedSignal = false;
00397   std::vector<double> m_delayedSignalTimes;
00398   unsigned int m_nAvgDelayedSignal = 0;
00399
00400   // Transfer function
00401   std::function<double(double)> m_fTransfer;
00402   Shaper* m_shaper = nullptr;
00403   std::vector<std::pair<double, double> > m_fTransferTab;
00404   bool m_cacheTransferFunction = true;
00405   // Integral of the transfer function squared.
00406   double m_fTransferSq = -1.;
00407   // FFT of the transfer function.
00408   std::vector<double> m_fTransferFFT;
00409
00410   // Noise
00411   double (*m_fNoise)(double t) = nullptr;
00412
00413   std::vector<std::pair<double, bool> > m_thresholdCrossings;
00414   double m_thresholdLevel = 0.;
00415 #endif
```

```
00416
00417   // User bounding box
00418   double m_xMinUser = 0., m_yMinUser = 0., m_zMinUser = 0.;
00419   double m_xMaxUser = 0., m_yMaxUser = 0., m_zMaxUser = 0.;
00420   bool m_hasUserArea = false;
00421
00422 #ifndef ___GPUCOMPILE___
00423   // Switch on/off debugging messages
00424   bool m_debug = false;
00425
00426   // Return the current sensor size
00427   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00428                       double& ymax, double& zmax);
00429
00430   void FillSignal(Electrode& electrode, const double q,
00431                   const std::vector<double>& ts, const std::vector<double>& is,
00432                   const int navg, const bool delayed = false);
00433   void FillBin(Electrode& electrode, const unsigned int bin,
00434                const double signal, const bool electron, const bool delayed) {
00435     std::lock_guard<std::mutex> guard(m_mutex);
00436     electrode.signal[bin] += signal;
00437     if (delayed) electrode.delayedSignal[bin] += signal;
00438     if (electron) {
00439       electrode.electronSignal[bin] += signal;
00440       if (delayed) electrode.delayedElectronSignal[bin] += signal;
00441     } else {
00442       electrode.ionSignal[bin] += signal;
00443       if (delayed) electrode.delayedIonSignal[bin] += signal;
00444     }
00445   }
00446 #else
00447   ___device___ void FillBin(ElectrodeGPU& electrode, const unsigned int bin,
00448                             const double signal, const bool electron,
00449                             const bool delayed, const int particle_idx);
00450 #endif
00451
00452 #ifndef ___GPUCOMPILE___
00453   void IntegrateSignal(Electrode& electrode);
00454   void ConvoluteSignal(Electrode& electrode, const std::vector<double>& tab);
00455   bool ConvoluteSignalFFT();
00456   bool ConvoluteSignalFFT(const std::string& label);
00457   void ConvoluteSignalFFT(Electrode& electrode, const std::vector<double>& tab,
00458                           const unsigned int nn);
00459   // Evaluate the integral over the transfer function squared.
00460   double TransferFunctionSq();
00461   double InterpolateTransferFunctionTable(const double t) const;
00462   void MakeTransferFunctionTable(std::vector<double>& tab);
00463   void FFT(std::vector<double>& data, const bool inverse, const int nn);
00464 #endif
00465 };
```

## GARFIELD_CLASS_NAME() [4/4]

class Garfield::GARFIELD_CLASS_NAME (
TetrahedralTree )

Helper class for searches in field maps.

This class stores the mesh nodes and elements in an Octree data structure to optimize the element search operations

Author: Ali Sheharyar

Organization: Texas A&M University at Qatar Destructor

Insert a mesh element with given bounding box and index to the tree.

Create and initialise GPU Transfer class

Definition at line 1 of file TetrahedralTree.hh.

```
00042                                   {
00043  public:
00044 #ifdef ___GPUCOMPILE___
00045   // Constructor
00046   GARFIELD_CLASS_NAME(TetrahedralTree)() = default;
```

```
00047
00048   // Destructor
00049   ~GARFIELD_CLASS_NAME(TetrahedralTree)() = default;
00050 #else
00051   // Constructor
00052   GARFIELD_CLASS_NAME(TetrahedralTree)(const Vec3& origin,
00053                                        const Vec3& halfDimension);
00054
00056   ~GARFIELD_CLASS_NAME(TetrahedralTree)();
00057 #endif
00058
00059 #ifndef ___GPUCOMPILE___
00060   // Insert a mesh node (a vertex/point) to the tree
00061   void InsertMeshNode(Vec3 point, const int index);
00062
00064   void InsertMeshElement(const double bb[6], const int index);
00065
00067   double CreateGPUTransferObject(TetrahedralTreeGPU*& tree_gpu);
00068 #endif
00069
00070  private:
00071   static std::vector<int> emptyBlock;
00072
00073   // Physical centre of this tree node.
00074   Vec3 m_origin;
00075   // Half the width/height/depth of this tree node.
00076   Vec3 m_halfDimension;
00077   // Storing min and max points for convenience
00078   Vec3 m_min, m_max;
00079
00080   // The tree has up to eight children and can additionally store
00081   // a list of mesh nodes and mesh elements.
00082   // Pointers to child octants.
00083   GARFIELD_CLASS_NAME(TetrahedralTree) * children[8];
00084
00085   // Children follow a predictable pattern to make accesses simple.
00086   // Here, - means less than 'origin' in that dimension, + means greater than.
00087   // child: 0 1 2 3 4 5 6 7
00088   // x:     - - - - + + + +
00089   // y:     - - + + - - + +
00090   // z:     - + - + - + - +
00091
00092 #ifndef ___GPUCOMPILE___
00093   std::vector<std::pair<Vec3, int> > nodes;
00094 #endif
00095
00096 #ifdef ___GPUCOMPILE___
00097   int* elements{nullptr};
00098   int numelements{0};
00099 #else
00100   std::vector<int> elements;
00101 #endif
00102
00103   static const size_t BlockCapacity = 10;
00104
00105 #ifndef ___GPUCOMPILE___
00106   // Check if the given box overlaps with this tree node.
00107   bool DoesBoxOverlap(const double bb[6]) const;
00108 #endif
00109   // Check if this tree node is a leaf or intermediate node.
00110   ___DEVICE___ bool IsLeafNode() const;
00111
00112  public:
00113 // Get all tetrahedra linked to a block corresponding to the given point
00114 #ifdef ___GPUCOMPILE___
00115   ___device___ void GetElementsInBlock(const Vec3& point,
00116                                        const int*& tet_list_elems,
00117                                        int& num_elems) const;
00118 #else
00119   const std::vector<int>& GetElementsInBlock(const Vec3& point) const;
00120 #endif
00121
00122  private:
00123   ___DEVICE___ int GetOctantContainingPoint(const Vec3& point) const;
00124   // Get a block containing the input point
00125   ___DEVICE___ const GARFIELD_CLASS_NAME(TetrahedralTree) *
00126       GetBlockFromPoint(const Vec3& point) const;
00127   // A helper function used by the function above.
00128   // Called recursively on the child nodes.
00129   ___DEVICE___ const GARFIELD_CLASS_NAME(TetrahedralTree) *
00130       GetBlockFromPointHelper(const Vec3& point) const;
00131
00132 #ifdef ___GPUCOMPILE___
00133   friend class TetrahedralTree;
00134 #endif
00135 };
```

### ltrim()

```
void Garfield::ltrim (
                std::string & line)   [inline]
```

Definition at line 11 of file Utilities.hh.

```
00011                                {
00012   line.erase(line.begin(), std::find_if(line.begin(), line.end(), [](int ch) {
00013                return !std::isspace(ch);
00014              }));
00015 }
```

### RndmDirection()

```
void Garfield::RndmDirection (
                double & dx,
                double & dy,
                double & dz,
                const double length = 1.)   [inline]
```

Draw a random (isotropic) direction vector.

Definition at line 138 of file Random.hh.

```
00139                                                {
00140   const double phi = TwoPi * RndmUniform();
00141   const double ctheta = 2 * RndmUniform() - 1.;
00142   const double stheta = sqrt(1. - ctheta * ctheta);
00143   dx = length * cos(phi) * stheta;
00144   dy = length * sin(phi) * stheta;
00145   dz = length * ctheta;
00146 }
```

### RndmGaussian() [1/2]

```
double Garfield::RndmGaussian ()   [inline]
```

Draw a Gaussian random variate with mean zero and standard deviation one.

Definition at line 35 of file Random.hh.

```
00035                              {
00036   static bool cached = false;
00037   static double u = 0.;
00038   if (cached) {
00039     cached = false;
00040     return u;
00041   }
00042   // Box-Muller algorithm
00043   u = 2. * RndmUniform() - 1.;
00044   double v = 2. * RndmUniform() - 1.;
00045   double r2 = u * u + v * v;
00046   while (r2 > 1.) {
00047     u = 2. * RndmUniform() - 1.;
00048     v = 2. * RndmUniform() - 1.;
00049     r2 = u * u + v * v;
00050   }
00051   const double p = sqrt(-2. * log(r2) / r2);
00052   u *= p;
00053   cached = true;
00054   return v * p;
00055 }
```

RndmGaussian() [2/2]

```
double Garfield::RndmGaussian (
                const double mu,
                const double sigma)    [inline]
```

Draw a Gaussian random variate with mean mu and standard deviation sigma.

Definition at line 58 of file Random.hh.

```
00058                                                  {
00059   return mu + sigma * RndmGaussian();
00060 }
```

RndmHeedWF()

```
double Garfield::RndmHeedWF (
                const double w,
                const double f)
```

Draw a random energy needed to create a single electron in a material asymptotic work function W and Fano factor F, according to Igor Smirnov's phenomenological model.

RndmLandau()

```
double Garfield::RndmLandau ()
```

Draw a random number from a Landau distribution.

RndmLorentzian()

```
double Garfield::RndmLorentzian (
                const double mu,
                const double gamma)    [inline]
```

Draw a Lorentzian random variate with mean mu and half-width at half maximum gamma.

Definition at line 64 of file Random.hh.

```
00064                                                      {
00065   return mu + gamma * tan(Pi * (RndmUniform() - 0.5));
00066 }
```

RndmPoisson()

```
int Garfield::RndmPoisson (
                const double mean)
```

Draw a random number from a Poisson distribution.

## RndmPolya()

double Garfield::RndmPolya (
const double theta) [inline]

Draw a Polya distributed random number.

Definition at line 101 of file Random.hh.

```
00101                                       {
00102   // Algorithm from Review of Particle Physics
00103   // C. Amsler et al, Phys. Lett. B 667 (2008)
00104   if (theta <= 0.) return -log(RndmUniformPos());
00105   const double c = 3 * (theta + 1.) - 0.75;
00106   double u1, u2, v1, v2, v3;
00107   double x;
00108   while (1) {
00109     u1 = RndmUniformPos();
00110     v1 = u1 * (1. - u1);
00111     if (v1 == 0.) continue;
00112     v2 = (u1 - 0.5) * sqrt(c / v1);
00113     x = theta + v2;
00114     if (x <= 0.) continue;
00115     u2 = RndmUniformPos();
00116     v3 = 64 * u2 * u2 * pow(v1, 3);
00117     if (v3 <= 1. - 2 * v2 * v2 / x ||
00118         log(v3) <= 2 * (theta * log(x / theta) - v2)) {
00119       return x / (theta + 1.);
00120     }
00121   }
00122 }
```

## RndmUniform()

double Garfield::RndmUniform () [inline]

Draw a random number uniformly distributed in the range [0, 1).

Definition at line 25 of file Random.hh.

```
00025 { return Random::Draw(); }
```

## RndmUniformPos()

double Garfield::RndmUniformPos () [inline]

Draw a random number uniformly distributed in the range (0, 1).

Definition at line 28 of file Random.hh.

```
00028                                    {
00029   double r = RndmUniform();
00030   while (r <= 0.) r = RndmUniform();
00031   return r;
00032 }
```

## RndmVavilov()

double Garfield::RndmVavilov (
const double rkappa,
const double beta2)

Draw a random number from a Vavilov distribution.

### RndmVoigt()

```
double Garfield::RndmVoigt (
                const double mu,
                const double sigma,
                const double gamma)   [inline]
```

Draw a random number according to a Voigt function with mean mu.

The Voigt function is a convolution of a Gaussian (standard deviation sigma) and a Lorentzian (half width gamma).

Definition at line 72 of file Random.hh.

```
00073                                         {
00074   if (sigma <= 0.) return RndmLorentzian(mu, gamma);
00075   const double a = gamma / (Sqrt2 * sigma);
00076   const double x = RndmLorentzian(0., a) + RndmGaussian(0., 1. / Sqrt2);
00077   return mu + x * Sqrt2 * sigma;
00078 }
```

### RndmYuleFurry()

```
unsigned int Garfield::RndmYuleFurry (
                const double mean)   [inline]
```

Draw a random number from a geometric distribution.

Definition at line 81 of file Random.hh.

```
00081                                                 {
00082   if (mean <= 0.) return 0;
00083   return 1 + static_cast<unsigned int>(std::log(RndmUniformPos()) /
00084                                        std::log1p(-1. / mean));
00085   /*
00086   const double u = RndmUniform();
00087   double p = 1. / mean;
00088   const double q = 1. - p;
00089   double sum = p;
00090   unsigned int k = 1;
00091   while (sum < u) {
00092     ++k;
00093     p *= q;
00094     sum += p;
00095   }
00096   return k;
00097   */
00098 }
```

### rtrim()

```
void Garfield::rtrim (
                std::string & line)   [inline]
```

Definition at line 17 of file Utilities.hh.

```
00017                                   {
00018   line.erase(std::find_if(line.rbegin(), line.rend(),
00019                           [](int ch) { return !std::isspace(ch); })
00020                .base(),
00021              line.end());
00022 }
```

### SetDefaultStyle()

void Garfield::SetDefaultStyle () [inline]

Definition at line 10 of file Plotting.hh.

```
00010 { plottingEngine.SetDefaultStyle(); }
```

### SetSansSerif()

void Garfield::SetSansSerif () [inline]

Definition at line 14 of file Plotting.hh.

```
00014 { plottingEngine.SetSansSerif(); }
```

### SetSerif()

void Garfield::SetSerif () [inline]

Definition at line 12 of file Plotting.hh.

```
00012 { plottingEngine.SetSerif(); }
```

### startsWith()

bool Garfield::startsWith (
const std::string & line,
const std::string & s) [inline]

Definition at line 33 of file Utilities.hh.

```
00033                                                 {
00034   return (line.rfind(s, 0) == 0);
00035 }
```

### tokenize()

std::vector< std::string > Garfield::tokenize (
const std::string & line) [inline]

Definition at line 24 of file Utilities.hh.

```
00024                                                 {
00025   std::vector<std::string> words;
00026   std::istringstream ss(line);
00027   for (std::string word; ss » word;) {
00028     words.push_back(word);
00029   }
00030   return words;
00031 }
```

## 6.1.4 Variable Documentation

### gComponentNeBem3d

ComponentNeBem3d* Garfield::gComponentNeBem3d [extern]

plottingEngine

PlottingEngine Garfield::plottingEngine [extern]

## 6.2 Garfield::Degrade Namespace Reference

Functions

- void deginit (int64_t *ng, int64_t *nevt, int64_t *mip, int64_t *idvec, int32_t *iseed, double *e0, double *et, double *ec, int64_t *ngas1, int64_t *ngas2, int64_t *ngas3, int64_t *ngas4, int64_t *ngas5, int64_t *ngas6, double *frac1, double *frac2, double *frac3, double *frac4, double *frac5, double *frac6, double *t0, double *p0, double *etot, double *btot, double *bang, int64_t *jcmp, int64_t *jray, int64_t *jpap, int64_t *jbrm, int64_t *jecasc, int64_t *iverb)
- void getdedx (double *dedxi, double *cldensi)
- int64_t getie (double *e)
- double gettcf (int64_t *ie)
- double gettcfn (int64_t *ie)
- void getlevel (int64_t *ie, double *r1, int64_t *izbr, double *rgas, double *ein, int64_t *ia, double *wpl, int64_t *index, double *an, double *ps, double *wklm, int64_t *nc0, double *ec0, int64_t *ng1, double *eg1, int64_t *ng2, double *eg2, double *dstfl, int64_t *jpn, int64_t *kg1, int64_t *lg1, int64_t *igshel, int64_t *ionmdl, int64_t *ilvl)
- int64_t getgas (int64_t *ilvl)
- double ionpot ()
- void getebrem (int64_t *k, double *ee, double *xe, double *ye, double *ze, double *te, double *dxe, double *dye, double *dze, int64_t *iok)
- void getecasc (int64_t *k, double *ee, double *xe, double *ye, double *ze, double *te, double *dxe, double *dye, double *dze, int64_t *iok)
- void degrade ()
- void brems (int64_t *iz, double *ein, double *dx, double *dy, double *dz, double *eout, double *dxe, double *dye, double *dze, double *egamma, double *dxg, double *dyg, double *dzg)
- void bremscasc (int64_t *j11, double *egamma, double *x0, double *y0, double *z0, double *t0, double *gdcx, double *gdcy, double *gdcz, int64_t *ilow)
- void cascadee (int64_t *j11, int64_t *kgas, int64_t *lgas, double *x0, double *y0, double *z0, double *t0, double *einit, int64_t *ishell)
- void drcos (double *drx, double *dry, double *drz, double *theta, double *phi, double *drxx, double *dryy, double *drzz)
- void ionsplit (int64_t *i, double *e, double *ei, double *esec)

### 6.2.1 Function Documentation

brems()

void Garfield::Degrade::brems (
int64_t * iz,
double * ein,
double * dx,
double * dy,
double * dz,
double * eout,
double * dxe,
double * dye,
double * dze,
double * egamma,
double * dxg,
double * dyg,
double * dzg)

bremscasc()

```
void Garfield::Degrade::bremscasc (
                int64_t * j11,
                double * egamma,
                double * x0,
                double * y0,
                double * z0,
                double * t0,
                double * gdcx,
                double * gdcy,
                double * gdcz,
                int64_t * ilow)
```

cascadee()

```
void Garfield::Degrade::cascadee (
                int64_t * j11,
                int64_t * kgas,
                int64_t * lgas,
                double * x0,
                double * y0,
                double * z0,
                double * t0,
                double * einit,
                int64_t * ishell)
```

deginit()

```
void Garfield::Degrade::deginit (
                int64_t * ng,
                int64_t * nevt,
                int64_t * mip,
                int64_t * idvec,
                int32_t * iseed,
                double * e0,
                double * et,
                double * ec,
                int64_t * ngas1,
                int64_t * ngas2,
                int64_t * ngas3,
                int64_t * ngas4,
                int64_t * ngas5,
                int64_t * ngas6,
                double * frac1,
                double * frac2,
                double * frac3,
                double * frac4,
                double * frac5,
                double * frac6,
                double * t0,
                double * p0,
                double * etot,
                double * btot,
                double * bang,
```

int64_t * jcmp,
int64_t * jray,
int64_t * jpap,
int64_t * jbrm,
int64_t * jecasc,
int64_t * iverb)

### degrade()

void Garfield::Degrade::degrade ()

### drcos()

void Garfield::Degrade::drcos (
double * drx,
double * dry,
double * drz,
double * theta,
double * phi,
double * drxx,
double * dryy,
double * drzz)

### getdedx()

void Garfield::Degrade::getdedx (
double * dedxi,
double * cldensi)

### getebrem()

void Garfield::Degrade::getebrem (
int64_t * k,
double * ee,
double * xe,
double * ye,
double * ze,
double * te,
double * dxe,
double * dye,
double * dze,
int64_t * iok)

### getecasc()

void Garfield::Degrade::getecasc (
int64_t * k,
double * ee,
double * xe,
double * ye,
double * ze,
double * te,
double * dxe,
double * dye,
double * dze,
int64_t * iok)

### getgas()

```
int64_t Garfield::Degrade::getgas (
                int64_t * ilvl)
```

### getie()

```
int64_t Garfield::Degrade::getie (
                double * e)
```

### getlevel()

```
void Garfield::Degrade::getlevel (
                int64_t * ie,
                double * r1,
                int64_t * izbr,
                double * rgas,
                double * ein,
                int64_t * ia,
                double * wpl,
                int64_t * index,
                double * an,
                double * ps,
                double * wklm,
                int64_t * nc0,
                double * ec0,
                int64_t * ng1,
                double * eg1,
                int64_t * ng2,
                double * eg2,
                double * dstfl,
                int64_t * jpn,
                int64_t * kg1,
                int64_t * lg1,
                int64_t * igshel,
                int64_t * ionmdl,
                int64_t * ilvl)
```

### gettcf()

```
double Garfield::Degrade::gettcf (
                int64_t * ie)
```

### gettcfn()

```
double Garfield::Degrade::gettcfn (
                int64_t * ie)
```

### ionpot()

```
double Garfield::Degrade::ionpot ()
```

ionsplit()

```
void Garfield::Degrade::ionsplit (
                int64_t * i,
                double * e,
                double * ei,
                double * esec)
```

## 6.3 Garfield::Magboltz Namespace Reference

Functions

- void gasmix_ (std::int64_t *ngs, double *q, double *qin, std::int64_t *nin, double *e, double *ei, char *name, double *virl, double *eb, double *peqel, double *peqin, double *penfra, std::int64_t *kel, std::int64_t *kin, double *qion, double *peqion, double *eion, std::int64_t *nion, double *qatt, std::int64_t *natt, double *qnull, std::int64_t *nnull, double *scln, std::int64_t *nc0, double *ec0, double *wk, double *efl, std::int64_t *ng1, double *eg1, std::int64_t *ng2, double *eg2, char scrpt[nMaxLevelsPerComponent][nCharDescr], char scrptn[nMaxNullTerms][nCharDescr], short namelen, short scrpt_len, short scrptn_len)
- void colf_ (double *freq, double *freel, double *freion, double *freatt, double *frein, std::int64_t *ntotal)
- void colft_ (double *freq, double *freel, double *freion, double *freatt, double *frein, std::int64_t *ntotal)
- void magboltz_ ()

Variables

- constexpr unsigned int nEnergySteps = 4000
- constexpr unsigned int nMaxIonisationTerms = 30
- constexpr unsigned int nMaxInelasticTerms = 250
- constexpr unsigned int nMaxAttachmentTerms = 8
- constexpr unsigned int nMaxNullTerms = 10
- constexpr unsigned int nMaxLevelsPerComponent = 300
- constexpr unsigned int nCharName = 25
- constexpr unsigned int nCharDescr = 50
- constexpr unsigned int nMaxLevels = 960
- constexpr unsigned int nMaxComponents = 6
- struct {
  double Garfield::Magboltz::eovb
  double Garfield::Magboltz::wb
  double Garfield::Magboltz::btheta
  double Garfield::Magboltz::bmag
  } bfld_

- struct {
  std::int64_t Garfield::Magboltz::nGas
  std::int64_t Garfield::Magboltz::nStep
  std::int64_t Garfield::Magboltz::nAniso
  double Garfield::Magboltz::efinal
  double Garfield::Magboltz::estep
  double Garfield::Magboltz::akt
  double Garfield::Magboltz::ary
  double Garfield::Magboltz::tempc
  double Garfield::Magboltz::torr
  std::int64_t Garfield::Magboltz::ipen
  } inpt_

- struct {
  double Garfield::Magboltz::tmax
  double Garfield::Magboltz::small
  double Garfield::Magboltz::api
  double Garfield::Magboltz::estart
  double Garfield::Magboltz::theta
  double Garfield::Magboltz::phi
  double Garfield::Magboltz::rstart
  double Garfield::Magboltz::efield
  std::int64_t Garfield::Magboltz::nmax
  } setp_

- struct {
  double Garfield::Magboltz::amgas [6]
  double Garfield::Magboltz::vtmb [6]
  double Garfield::Magboltz::tcfmx
  double Garfield::Magboltz::tcfmxg [6]
  std::int64_t Garfield::Magboltz::ithrm
  } thrm_

- struct {
  double Garfield::Magboltz::echarg
  double Garfield::Magboltz::emass
  double Garfield::Magboltz::amu
  double Garfield::Magboltz::pir2
  } cnsts_

- struct {
  double Garfield::Magboltz::eg [nEnergySteps]
  double Garfield::Magboltz::eroot [nEnergySteps]
  double Garfield::Magboltz::qt1 [nEnergySteps]
  double Garfield::Magboltz::qt2 [nEnergySteps]
  double Garfield::Magboltz::qt3 [nEnergySteps]
  double Garfield::Magboltz::qt4 [nEnergySteps]
  } mix2_

- struct {
  double Garfield::Magboltz::den [nEnergySteps]
  } dens_

- struct {
  double Garfield::Magboltz::time [300]
  std::int64_t Garfield::Magboltz::icoll [30]
  double Garfield::Magboltz::spec [nEnergySteps]
  double Garfield::Magboltz::tmax1
  double Garfield::Magboltz::ave
  double Garfield::Magboltz::den
  double Garfield::Magboltz::xid
  double Garfield::Magboltz::x
  double Garfield::Magboltz::y
  double Garfield::Magboltz::z
  double Garfield::Magboltz::st
  std::int64_t Garfield::Magboltz::nnull
  std::int64_t Garfield::Magboltz::icoln [nMaxLevels]
  std::int64_t Garfield::Magboltz::icolnn [60]
  } outpt_

- struct {

double Garfield::Magboltz::time [300]
std::int64_t Garfield::Magboltz::icoll [5][nMaxComponents]
double Garfield::Magboltz::spec [nEnergySteps]
double Garfield::Magboltz::tmax1
double Garfield::Magboltz::ave
double Garfield::Magboltz::den
double Garfield::Magboltz::xid
double Garfield::Magboltz::x
double Garfield::Magboltz::y
double Garfield::Magboltz::z
double Garfield::Magboltz::st
std::int64_t Garfield::Magboltz::nnull
std::int64_t Garfield::Magboltz::icoln [290][nMaxComponents]
std::int64_t Garfield::Magboltz::icolnn [10][nMaxComponents]
} outptt_

- struct {
  char Garfield::Magboltz::dscrpt [nMaxLevels][nCharDescr]
  char Garfield::Magboltz::dscrptn [60][nCharDescr]
  } scrip_

- struct {
  char Garfield::Magboltz::dscrpt [nMaxLevelsPerComponent][nMaxComponents][nCharDescr]
  char Garfield::Magboltz::dscrptn [10][nMaxComponents][nCharDescr]
  } script_

- struct {
  double Garfield::Magboltz::cf [nMaxLevels][nEnergySteps]
  double Garfield::Magboltz::ein [nMaxLevels]
  double Garfield::Magboltz::tcf [nEnergySteps]
  std::int64_t Garfield::Magboltz::iarry [nMaxLevels]
  double Garfield::Magboltz::rgas [nMaxLevels]
  double Garfield::Magboltz::ipn [nMaxLevels]
  double Garfield::Magboltz::wpl [nMaxLevels]
  std::int64_t Garfield::Magboltz::last
  std::int64_t Garfield::Magboltz::isize
  double Garfield::Magboltz::penfra [nMaxLevels][3]
  double Garfield::Magboltz::tcfmax [8]
  } large_

- struct {
  double Garfield::Magboltz::cf [290][nEnergySteps][nMaxComponents]
  double Garfield::Magboltz::ein [290][nMaxComponents]
  double Garfield::Magboltz::tcf [nEnergySteps][nMaxComponents]
  std::int64_t Garfield::Magboltz::iarry [290][nMaxComponents]
  double Garfield::Magboltz::rgas [290][nMaxComponents]
  double Garfield::Magboltz::ipn [290][nMaxComponents]
  double Garfield::Magboltz::wpl [290][nMaxComponents]
  std::int64_t Garfield::Magboltz::last [nMaxComponents]
  std::int64_t Garfield::Magboltz::isize [nMaxComponents]
  double Garfield::Magboltz::penfra [290][3][nMaxComponents]
  double Garfield::Magboltz::tcfmax [nMaxComponents]
  } larget_

- struct {
  std::int64_t Garfield::Magboltz::ngasn [6]
  } gasn_

- struct {
  double Garfield::Magboltz::an1
  double Garfield::Magboltz::an2
  double Garfield::Magboltz::an3
  double Garfield::Magboltz::an4
  double Garfield::Magboltz::an5
  double Garfield::Magboltz::an6
  double Garfield::Magboltz::an
  double Garfield::Magboltz::frac [6]
  } ratio_

- struct {
  double Garfield::Magboltz::wx
  double Garfield::Magboltz::wy
  double Garfield::Magboltz::wz
  } vel_

- struct {
  double Garfield::Magboltz::dwx
  double Garfield::Magboltz::dwy
  double Garfield::Magboltz::dwz
  } velerr_

- struct {
  double Garfield::Magboltz::difxx
  double Garfield::Magboltz::difyy
  double Garfield::Magboltz::difzz
  double Garfield::Magboltz::difyz
  double Garfield::Magboltz::difxy
  double Garfield::Magboltz::difxz
  } diflab_

- struct {
  double Garfield::Magboltz::dxxer
  double Garfield::Magboltz::dyyer
  double Garfield::Magboltz::dzzer
  double Garfield::Magboltz::dyzer
  double Garfield::Magboltz::dxyer
  double Garfield::Magboltz::dxzer
  } diferb_

- struct {
  double Garfield::Magboltz::difln
  double Garfield::Magboltz::diftr
  } difvel_

- struct {
  double Garfield::Magboltz::dfler
  double Garfield::Magboltz::dfter
  } diferl_

- struct {
  double Garfield::Magboltz::alpha
  double Garfield::Magboltz::att
  } ctowns_

- struct {
  double Garfield::Magboltz::alper

double Garfield::Magboltz::atter
} ctwner_

- struct {
  double Garfield::Magboltz::ralpha
  double Garfield::Magboltz::ralper
  double Garfield::Magboltz::tofene
  double Garfield::Magboltz::tofener
  double Garfield::Magboltz::tofwv
  double Garfield::Magboltz::tofwver
  double Garfield::Magboltz::tofdl
  double Garfield::Magboltz::tofdler
  double Garfield::Magboltz::tofdt
  double Garfield::Magboltz::tofdter
  double Garfield::Magboltz::tofwr
  double Garfield::Magboltz::tofwrer
  double Garfield::Magboltz::rattof
  double Garfield::Magboltz::ratofer
  } tofout_

- struct {
  double Garfield::Magboltz::vdout
  double Garfield::Magboltz::vderr
  double Garfield::Magboltz::wsout
  double Garfield::Magboltz::wserr
  double Garfield::Magboltz::dlout
  double Garfield::Magboltz::dlerr
  double Garfield::Magboltz::dtout
  double Garfield::Magboltz::dterr
  double Garfield::Magboltz::alphasst
  double Garfield::Magboltz::alphaerr
  double Garfield::Magboltz::attsst
  double Garfield::Magboltz::atterr
  } sstout_

### 6.3.1 Function Documentation

colf_()

```
void Garfield::Magboltz::colf_ (
                double * freq,
                double * freel,
                double * freion,
                double * freatt,
                double * frein,
                std::int64_t * ntotal)
```

colft_()

```
void Garfield::Magboltz::colft_ (
                double * freq,
                double * freel,
                double * freion,
                double * freatt,
                double * frein,
                std::int64_t * ntotal)
```

gasmix_()

```
void Garfield::Magboltz::gasmix_ (
                std::int64_t * ngs,
                double * q,
                double * qin,
                std::int64_t * nin,
                double * e,
                double * ei,
                char * name,
                double * virl,
                double * eb,
                double * peqel,
                double * peqin,
                double * penfra,
                std::int64_t * kel,
                std::int64_t * kin,
                double * qion,
                double * peqion,
                double * eion,
                std::int64_t * nion,
                double * qatt,
                std::int64_t * natt,
                double * qnull,
                std::int64_t * nnull,
                double * scln,
                std::int64_t * nc0,
                double * ec0,
                double * wk,
                double * efl,
                std::int64_t * ng1,
                double * eg1,
                std::int64_t * ng2,
                double * eg2,
                char scrpt[nMaxLevelsPerComponent][nCharDescr],
                char scrptn[nMaxNullTerms][nCharDescr],
                short namelen,
                short scrpt_len,
                short scrptn_len)
```

magboltz_()

```
void Garfield::Magboltz::magboltz_ ()
```

### 6.3.2 Variable Documentation

akt

```
double Garfield::Magboltz::akt
```

Definition at line 42 of file MagboltzInterface.hh.

alper

double Garfield::Magboltz::alper

Definition at line 201 of file MagboltzInterface.hh.

alpha

double Garfield::Magboltz::alpha

Definition at line 198 of file MagboltzInterface.hh.

alphaerr

double Garfield::Magboltz::alphaerr

Definition at line 218 of file MagboltzInterface.hh.

alphasst

double Garfield::Magboltz::alphasst

Definition at line 218 of file MagboltzInterface.hh.

amgas

double Garfield::Magboltz::amgas[6]

Definition at line 61 of file MagboltzInterface.hh.

amu

double Garfield::Magboltz::amu

Definition at line 72 of file MagboltzInterface.hh.

an

double Garfield::Magboltz::an

Definition at line 167 of file MagboltzInterface.hh.

an1

double Garfield::Magboltz::an1

Definition at line 167 of file MagboltzInterface.hh.

an2

double Garfield::Magboltz::an2

Definition at line 167 of file MagboltzInterface.hh.

an3

double Garfield::Magboltz::an3

Definition at line 167 of file MagboltzInterface.hh.

an4

double Garfield::Magboltz::an4

Definition at line 167 of file MagboltzInterface.hh.

an5

double Garfield::Magboltz::an5

Definition at line 167 of file MagboltzInterface.hh.

an6

double Garfield::Magboltz::an6

Definition at line 167 of file MagboltzInterface.hh.

api

double Garfield::Magboltz::api

Definition at line 52 of file MagboltzInterface.hh.

ary

double Garfield::Magboltz::ary

Definition at line 43 of file MagboltzInterface.hh.

att

double Garfield::Magboltz::att

Definition at line 198 of file MagboltzInterface.hh.

atter

double Garfield::Magboltz::atter

Definition at line 201 of file MagboltzInterface.hh.

atterr

double Garfield::Magboltz::atterr

Definition at line 219 of file MagboltzInterface.hh.

attsst

double Garfield::Magboltz::attsst

Definition at line 219 of file MagboltzInterface.hh.

ave

double Garfield::Magboltz::ave

Definition at line 94 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::bfld_

bmag

double Garfield::Magboltz::bmag

Definition at line 33 of file MagboltzInterface.hh.

btheta

double Garfield::Magboltz::btheta

Definition at line 33 of file MagboltzInterface.hh.

cf

double Garfield::Magboltz::cf[290][nEnergySteps][nMaxComponents]

Definition at line 134 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::cnsts_

[struct]

struct { ... } Garfield::Magboltz::ctowns_

[struct]

struct { ... } Garfield::Magboltz::ctwner_

## den

double Garfield::Magboltz::den

Definition at line 86 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::dens_

## dfler

double Garfield::Magboltz::dfler

Definition at line 193 of file MagboltzInterface.hh.

## dfter

double Garfield::Magboltz::dfter

Definition at line 193 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::diferb_

[struct]

struct { ... } Garfield::Magboltz::diferl_

[struct]

struct { ... } Garfield::Magboltz::diflab_

### difln

double Garfield::Magboltz::difln

Definition at line 190 of file MagboltzInterface.hh.

### diftr

double Garfield::Magboltz::diftr

Definition at line 190 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::difvel_

### difxx

double Garfield::Magboltz::difxx

Definition at line 182 of file MagboltzInterface.hh.

### difxy

double Garfield::Magboltz::difxy

Definition at line 183 of file MagboltzInterface.hh.

### difxz

double Garfield::Magboltz::difxz

Definition at line 183 of file MagboltzInterface.hh.

### difyy

double Garfield::Magboltz::difyy

Definition at line 182 of file MagboltzInterface.hh.

### difyz

double Garfield::Magboltz::difyz

Definition at line 183 of file MagboltzInterface.hh.

### difzz

double Garfield::Magboltz::difzz

Definition at line 182 of file MagboltzInterface.hh.

### dlerr

double Garfield::Magboltz::dlerr

Definition at line 216 of file MagboltzInterface.hh.

### dlout

double Garfield::Magboltz::dlout

Definition at line 216 of file MagboltzInterface.hh.

### dscrpt

char Garfield::Magboltz::dscrpt[nMaxLevelsPerComponent][nMaxComponents][nCharDescr]

Definition at line 124 of file MagboltzInterface.hh.

### dscrptn

char Garfield::Magboltz::dscrptn[10][nMaxComponents][nCharDescr]

Definition at line 125 of file MagboltzInterface.hh.

### dterr

double Garfield::Magboltz::dterr

Definition at line 218 of file MagboltzInterface.hh.

### dtout

double Garfield::Magboltz::dtout

Definition at line 218 of file MagboltzInterface.hh.

dwx

double Garfield::Magboltz::dwx

Definition at line 177 of file MagboltzInterface.hh.

dwy

double Garfield::Magboltz::dwy

Definition at line 177 of file MagboltzInterface.hh.

dwz

double Garfield::Magboltz::dwz

Definition at line 177 of file MagboltzInterface.hh.

dxxer

double Garfield::Magboltz::dxxer

Definition at line 186 of file MagboltzInterface.hh.

dxyer

double Garfield::Magboltz::dxyer

Definition at line 187 of file MagboltzInterface.hh.

dxzer

double Garfield::Magboltz::dxzer

Definition at line 187 of file MagboltzInterface.hh.

dyyer

double Garfield::Magboltz::dyyer

Definition at line 186 of file MagboltzInterface.hh.

dyzer

double Garfield::Magboltz::dyzer

Definition at line 187 of file MagboltzInterface.hh.

dzzer

double Garfield::Magboltz::dzzer

Definition at line 186 of file MagboltzInterface.hh.

echarg

double Garfield::Magboltz::echarg

Definition at line 70 of file MagboltzInterface.hh.

efield

double Garfield::Magboltz::efield

Definition at line 56 of file MagboltzInterface.hh.

efinal

double Garfield::Magboltz::efinal

Definition at line 40 of file MagboltzInterface.hh.

eg

double Garfield::Magboltz::eg[nEnergySteps]

Definition at line 77 of file MagboltzInterface.hh.

ein

double Garfield::Magboltz::ein[290][nMaxComponents]

Definition at line 135 of file MagboltzInterface.hh.

emass

double Garfield::Magboltz::emass

Definition at line 71 of file MagboltzInterface.hh.

eovb

double Garfield::Magboltz::eovb

Definition at line 31 of file MagboltzInterface.hh.

eroot

double Garfield::Magboltz::eroot[nEnergySteps]

Definition at line 78 of file MagboltzInterface.hh.

estart

double Garfield::Magboltz::estart

Definition at line 53 of file MagboltzInterface.hh.

estep

double Garfield::Magboltz::estep

Definition at line 41 of file MagboltzInterface.hh.

frac

double Garfield::Magboltz::frac[6]

Definition at line 168 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::gasn_

iarry

std::int64_t Garfield::Magboltz::iarry[290][nMaxComponents]

Definition at line 137 of file MagboltzInterface.hh.

icoll

std::int64_t Garfield::Magboltz::icoll[5][nMaxComponents]

Definition at line 91 of file MagboltzInterface.hh.

icoln

std::int64_t Garfield::Magboltz::icoln[290][nMaxComponents]

Definition at line 102 of file MagboltzInterface.hh.

icolnn

std::int64_t Garfield::Magboltz::icolnn[10][nMaxComponents]

Definition at line 103 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::inpt_

ipen

std::int64_t Garfield::Magboltz::ipen

Definition at line 46 of file MagboltzInterface.hh.

ipn

double Garfield::Magboltz::ipn[290][nMaxComponents]

Definition at line 139 of file MagboltzInterface.hh.

isize

std::int64_t Garfield::Magboltz::isize[nMaxComponents]

Definition at line 142 of file MagboltzInterface.hh.

ithrm

std::int64_t Garfield::Magboltz::ithrm

Definition at line 65 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::large_

[struct]

struct { ... } Garfield::Magboltz::larget_

last

std::int64_t Garfield::Magboltz::last[nMaxComponents]

Definition at line 141 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::mix2_

### nAniso

std::int64_t Garfield::Magboltz::nAniso

Definition at line 39 of file MagboltzInterface.hh.

### nCharDescr

unsigned int Garfield::Magboltz::nCharDescr = 50 [constexpr]

Definition at line 21 of file MagboltzInterface.hh.

### nCharName

unsigned int Garfield::Magboltz::nCharName = 25 [constexpr]

Definition at line 20 of file MagboltzInterface.hh.

### nEnergySteps

unsigned int Garfield::Magboltz::nEnergySteps = 4000 [constexpr]

Definition at line 14 of file MagboltzInterface.hh.

### nGas

std::int64_t Garfield::Magboltz::nGas

Definition at line 37 of file MagboltzInterface.hh.

### ngasn

std::int64_t Garfield::Magboltz::ngasn[6]

Definition at line 163 of file MagboltzInterface.hh.

### nmax

std::int64_t Garfield::Magboltz::nmax

Definition at line 57 of file MagboltzInterface.hh.

### nMaxAttachmentTerms

unsigned int Garfield::Magboltz::nMaxAttachmentTerms = 8 [constexpr]

Definition at line 17 of file MagboltzInterface.hh.

### nMaxComponents

unsigned int Garfield::Magboltz::nMaxComponents = 6 [constexpr]

Definition at line 23 of file MagboltzInterface.hh.

### nMaxInelasticTerms

unsigned int Garfield::Magboltz::nMaxInelasticTerms = 250 [constexpr]

Definition at line 16 of file MagboltzInterface.hh.

### nMaxIonisationTerms

unsigned int Garfield::Magboltz::nMaxIonisationTerms = 30 [constexpr]

Definition at line 15 of file MagboltzInterface.hh.

### nMaxLevels

unsigned int Garfield::Magboltz::nMaxLevels = 960 [constexpr]

Definition at line 22 of file MagboltzInterface.hh.

### nMaxLevelsPerComponent

unsigned int Garfield::Magboltz::nMaxLevelsPerComponent = 300 [constexpr]

Definition at line 19 of file MagboltzInterface.hh.

### nMaxNullTerms

unsigned int Garfield::Magboltz::nMaxNullTerms = 10 [constexpr]

Definition at line 18 of file MagboltzInterface.hh.

### nnull

std::int64_t Garfield::Magboltz::nnull

Definition at line 101 of file MagboltzInterface.hh.

nStep

std::int64_t Garfield::Magboltz::nStep

Definition at line 38 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::outpt_

[struct]

struct { ... } Garfield::Magboltz::outptt_

penfra

double Garfield::Magboltz::penfra[290][3][nMaxComponents]

Definition at line 143 of file MagboltzInterface.hh.

phi

double Garfield::Magboltz::phi

Definition at line 54 of file MagboltzInterface.hh.

pir2

double Garfield::Magboltz::pir2

Definition at line 73 of file MagboltzInterface.hh.

qt1

double Garfield::Magboltz::qt1[nEnergySteps]

Definition at line 79 of file MagboltzInterface.hh.

qt2

double Garfield::Magboltz::qt2[nEnergySteps]

Definition at line 80 of file MagboltzInterface.hh.

qt3

double Garfield::Magboltz::qt3[nEnergySteps]

Definition at line 81 of file MagboltzInterface.hh.

qt4

double Garfield::Magboltz::qt4[nEnergySteps]

Definition at line 82 of file MagboltzInterface.hh.

ralper

double Garfield::Magboltz::ralper

Definition at line 206 of file MagboltzInterface.hh.

ralpha

double Garfield::Magboltz::ralpha

Definition at line 206 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::ratio_

ratofer

double Garfield::Magboltz::ratofer

Definition at line 210 of file MagboltzInterface.hh.

rattof

double Garfield::Magboltz::rattof

Definition at line 210 of file MagboltzInterface.hh.

rgas

double Garfield::Magboltz::rgas[290][nMaxComponents]

Definition at line 138 of file MagboltzInterface.hh.

rstart

double Garfield::Magboltz::rstart

Definition at line 55 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::scrip_

[struct]

struct { ... } Garfield::Magboltz::script_

[struct]

struct { ... } Garfield::Magboltz::setp_

small

double Garfield::Magboltz::small

Definition at line 51 of file MagboltzInterface.hh.

spec

double Garfield::Magboltz::spec[nEnergySteps]

Definition at line 92 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::sstout_

st

double Garfield::Magboltz::st

Definition at line 100 of file MagboltzInterface.hh.

tcf

double Garfield::Magboltz::tcf[nEnergySteps][nMaxComponents]

Definition at line 136 of file MagboltzInterface.hh.

tcfmax

double Garfield::Magboltz::tcfmax[nMaxComponents]

Definition at line 144 of file MagboltzInterface.hh.

tcfmx

double Garfield::Magboltz::tcfmx

Definition at line 63 of file MagboltzInterface.hh.

tcfmxg

double Garfield::Magboltz::tcfmxg[6]

Definition at line 64 of file MagboltzInterface.hh.

tempc

double Garfield::Magboltz::tempc

Definition at line 44 of file MagboltzInterface.hh.

theta

double Garfield::Magboltz::theta

Definition at line 54 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::thrm_

time

double Garfield::Magboltz::time[300]

Definition at line 90 of file MagboltzInterface.hh.

tmax

double Garfield::Magboltz::tmax

Definition at line 50 of file MagboltzInterface.hh.

tmax1

double Garfield::Magboltz::tmax1

Definition at line 93 of file MagboltzInterface.hh.

tofdl

double Garfield::Magboltz::tofdl

Definition at line 208 of file MagboltzInterface.hh.

tofdler

double Garfield::Magboltz::tofdler

Definition at line 208 of file MagboltzInterface.hh.

tofdt

double Garfield::Magboltz::tofdt

Definition at line 208 of file MagboltzInterface.hh.

tofdter

double Garfield::Magboltz::tofdter

Definition at line 208 of file MagboltzInterface.hh.

tofene

double Garfield::Magboltz::tofene

Definition at line 207 of file MagboltzInterface.hh.

tofener

double Garfield::Magboltz::tofener

Definition at line 207 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::tofout_

tofwr

double Garfield::Magboltz::tofwr

Definition at line 209 of file MagboltzInterface.hh.

tofwrer

double Garfield::Magboltz::tofwrer

Definition at line 209 of file MagboltzInterface.hh.

tofwv

double Garfield::Magboltz::tofwv

Definition at line 207 of file MagboltzInterface.hh.

tofwver

double Garfield::Magboltz::tofwver

Definition at line 207 of file MagboltzInterface.hh.

torr

double Garfield::Magboltz::torr

Definition at line 45 of file MagboltzInterface.hh.

vderr

double Garfield::Magboltz::vderr

Definition at line 215 of file MagboltzInterface.hh.

vdout

double Garfield::Magboltz::vdout

Definition at line 215 of file MagboltzInterface.hh.

[struct]

struct { ... } Garfield::Magboltz::vel_

[struct]

struct { ... } Garfield::Magboltz::velerr_

vtmb

double Garfield::Magboltz::vtmb[6]

Definition at line 62 of file MagboltzInterface.hh.

wb

double Garfield::Magboltz::wb

Definition at line 32 of file MagboltzInterface.hh.

wpl

double Garfield::Magboltz::wpl[290][nMaxComponents]

Definition at line 140 of file MagboltzInterface.hh.

wserr

double Garfield::Magboltz::wserr

Definition at line 216 of file MagboltzInterface.hh.

wsout

double Garfield::Magboltz::wsout

Definition at line 216 of file MagboltzInterface.hh.

wx

double Garfield::Magboltz::wx

Definition at line 174 of file MagboltzInterface.hh.

wy

double Garfield::Magboltz::wy

Definition at line 174 of file MagboltzInterface.hh.

wz

double Garfield::Magboltz::wz

Definition at line 174 of file MagboltzInterface.hh.

x

double Garfield::Magboltz::x

Definition at line 97 of file MagboltzInterface.hh.

xid

double Garfield::Magboltz::xid

Definition at line 96 of file MagboltzInterface.hh.

y

double Garfield::Magboltz::y

Definition at line 98 of file MagboltzInterface.hh.

z

double Garfield::Magboltz::z

Definition at line 99 of file MagboltzInterface.hh.

## 6.4 Garfield::Numerics Namespace Reference

Collection of numerical routines.

Namespaces

- namespace CERNLIB
  Linear algebra routines from CERNLIB.
- namespace QUADPACK
  Functions for performing numerical integration (quadrature).

Functions

- constexpr std::array< double, 2 > GaussLegendreNodes2 ()
- constexpr std::array< double, 2 > GaussLegendreWeights2 ()
- constexpr std::array< double, 3 > GaussLegendreNodes3 ()
- constexpr std::array< double, 3 > GaussLegendreWeights3 ()
- constexpr std::array< double, 4 > GaussLegendreNodes4 ()
- constexpr std::array< double, 4 > GaussLegendreWeights4 ()
- constexpr std::array< double, 5 > GaussLegendreNodes5 ()
- constexpr std::array< double, 5 > GaussLegendreWeights5 ()
- constexpr std::array< double, 6 > GaussLegendreNodes6 ()
- constexpr std::array< double, 6 > GaussLegendreWeights6 ()
- double Legendre (const unsigned int n, const double x)

  Legendre polynomials.
- double BesselI0S (const double xx)

  Modified Bessel functions.
- double BesselI1S (const double xx)
- double BesselK0S (const double xx)
- double BesselK0L (const double xx)
- double BesselK1S (const double xx)
- double BesselK1L (const double xx)
- double Divdif (const std::vector< double > &f, const std::vector< double > &a, int nn, double x, int mm)

  C++ version of DIVDIF (CERN program library E105) which performs tabular interpolation using symmetrically placed argument points.
- double LinearInterpolation (const std::vector< double > &ytab, const std::vector< double > &xtab, const double x)
- bool Boxin2 (const std::vector< std::vector< double > > &value, const std::vector< double > &xAxis, const std::vector< double > &yAxis, const int nx, const int ny, const double xx, const double yy, double &f, const int iOrder)

  Interpolation of order 1 and 2 in an irregular rectangular two-dimensional grid.
- bool Boxin3 (const std::vector< std::vector< std::vector< double > > > &value, const std::vector< double > &xAxis, const std::vector< double > &yAxis, const std::vector< double > &zAxis, const int nx, const int ny, const int nz, const double xx, const double yy, const double zz, double &f, const int iOrder)

  Interpolation of order 1 and 2 in an irregular rectangular three-dimensional grid.
- bool LeastSquaresFit (std::function< double(double, const std::vector< double > &)> f, std::vector< double > &par, std::vector< double > &epar, const std::vector< double > &x, const std::vector< double > &y, const std::vector< double > &ey, const unsigned int nMaxIter, const double diff, double &chi2, const double eps, const bool debug, const bool verbose)

  Least-squares minimisation.

### 6.4.1 Detailed Description

Collection of numerical routines.

### 6.4.2 Function Documentation

#### BesselI0S()

```
double Garfield::Numerics::BesselI0S (
                const double xx)   [inline]
```

Modified Bessel functions.

Series expansions from Abramowitz and Stegun.

Definition at line 174 of file Numerics.hh.

```
00174                                   {
00175   const double y = xx / 3.75;
00176   const double y2 = y * y;
00177   return 1. + 3.5156229 * y2 + 3.0899424 * y2 * y2 + 1.2067492 * pow(y2, 3) +
00178         0.2659732 * pow(y2, 4) + 0.0360768 * pow(y2, 5) +
00179         0.0045813 * pow(y2, 6);
00180 }
```

#### BesselI1S()

```
double Garfield::Numerics::BesselI1S (
                const double xx)   [inline]
```

Definition at line 182 of file Numerics.hh.

```
00182                                   {
00183   const double y = xx / 3.75;
00184   const double y2 = y * y;
00185   return xx * (0.5 + 0.87890594 * y2 + 0.51498869 * y2 * y2 +
00186               0.15084934 * pow(y2, 3) + 0.02658733 * pow(y2, 4) +
00187               0.00301532 * pow(y2, 5) + 0.00032411 * pow(y2, 6));
00188 }
```

#### BesselK0L()

```
double Garfield::Numerics::BesselK0L (
                const double xx)   [inline]
```

Definition at line 199 of file Numerics.hh.

```
00199                                   {
00200   const double y = 2. / xx;
00201   return (exp(-xx) / sqrt(xx)) *
00202         (1.25331414 - 0.07832358 * y + 0.02189568 * y * y -
00203          0.01062446 * pow(y, 3) + 0.00587872 * pow(y, 4) -
00204          0.00251540 * pow(y, 5) + 0.00053208 * pow(y, 6));
00205 }
```

#### BesselK0S()

```
double Garfield::Numerics::BesselK0S (
                const double xx)   [inline]
```

Definition at line 190 of file Numerics.hh.

```
00190                                   {
00191   const double y = xx / 2.;
00192   const double y2 = y * y;
00193   return -log(y) * BesselI0S(xx) - Gamma + 0.42278420 * y2 +
00194         0.23069756 * y2 * y2 + 0.03488590 * pow(y2, 3) +
00195         0.00262698 * pow(y2, 4) + 0.00010750 * pow(y2, 5) +
00196         0.00000740 * pow(y2, 6);
00197 }
```

### BesselK1L()

```
double Garfield::Numerics::BesselK1L (
                const double xx)   [inline]
```

Definition at line 216 of file Numerics.hh.

```
00216                                   {
00217   const double y = 2. / xx;
00218   return (exp(-xx) / sqrt(xx)) *
00219         (1.25331414 + 0.23498619 * y - 0.03655620 * y * y +
00220          0.01504268 * pow(y, 3) - 0.00780353 * pow(y, 4) +
00221          0.00325614 * pow(y, 5) - 0.00068245 * pow(y, 6));
00222 }
```

### BesselK1S()

```
double Garfield::Numerics::BesselK1S (
                const double xx)   [inline]
```

Definition at line 207 of file Numerics.hh.

```
00207                                   {
00208   const double y = xx / 2.;
00209   const double y2 = y * y;
00210   return log(y) * BesselI1S(xx) +
00211         (1. / xx) * (1. + 0.15443144 * y2 - 0.67278579 * y2 * y2 -
00212                      0.18156897 * pow(y2, 3) - 0.01919402 * pow(y2, 4) -
00213                      0.00110404 * pow(y2, 5) - 0.00004686 * pow(y2, 6));
00214 }
```

### Boxin2()

```
bool Garfield::Numerics::Boxin2 (
                const std::vector< std::vector< double > > & value,
                const std::vector< double > & xAxis,
                const std::vector< double > & yAxis,
                const int nx,
                const int ny,
                const double xx,
                const double yy,
                double & f,
                const int iOrder)
```

Interpolation of order 1 and 2 in an irregular rectangular two-dimensional grid.

### Boxin3()

```
bool Garfield::Numerics::Boxin3 (
                const std::vector< std::vector< std::vector< double > > > & value,
                const std::vector< double > & xAxis,
                const std::vector< double > & yAxis,
                const std::vector< double > & zAxis,
                const int nx,
                const int ny,
                const int nz,
                const double xx,
                const double yy,
                const double zz,
                double & f,
                const int iOrder)
```

Interpolation of order 1 and 2 in an irregular rectangular three-dimensional grid.

### Divdif()

```
double Garfield::Numerics::Divdif (
                const std::vector< double > & f,
                const std::vector< double > & a,
                int nn,
                double x,
                int mm)
```

C++ version of DIVDIF (CERN program library E105) which performs tabular interpolation using symmetrically placed argument points.

### GaussLegendreNodes2()

```
std::array< double, 2 > Garfield::Numerics::GaussLegendreNodes2 ()  [constexpr]
```

Definition at line 16 of file Numerics.hh.

```
00016                                       {
00017   return {-0.577350269189625765, 0.577350269189625765};
00018 }
```

### GaussLegendreNodes3()

```
std::array< double, 3 > Garfield::Numerics::GaussLegendreNodes3 ()  [constexpr]
```

Definition at line 20 of file Numerics.hh.

```
00020                                       {
00021   return {-0.774596669241483377, 0., 0.774596669241483377};
00022 }
```

### GaussLegendreNodes4()

```
std::array< double, 4 > Garfield::Numerics::GaussLegendreNodes4 ()  [constexpr]
```

Definition at line 26 of file Numerics.hh.

```
00026                                       {
00027   return {-0.861136311594052575, -0.339981043584856265, 0.339981043584856265,
00028           0.861136311594052575};
00029 }
```

### GaussLegendreNodes5()

```
std::array< double, 5 > Garfield::Numerics::GaussLegendreNodes5 ()  [constexpr]
```

Definition at line 34 of file Numerics.hh.

```
00034                                       {
00035   return {-0.906179845938663993, -0.538469310105683091, 0.,
00036           0.538469310105683091, 0.906179845938663993};
00037 }
```

### GaussLegendreNodes6()

std::array< double, 6 > Garfield::Numerics::GaussLegendreNodes6 () [constexpr]

Definition at line 42 of file Numerics.hh.

```
00042                                       {
00043   return {-0.932469514203152028, -0.661209386466264514, -0.238619186083196909,
00044           0.238619186083196909,  0.661209386466264514,  0.932469514203152028};
00045 }
```

### GaussLegendreWeights2()

std::array< double, 2 > Garfield::Numerics::GaussLegendreWeights2 () [constexpr]

Definition at line 19 of file Numerics.hh.

```
00019 { return {1., 1.}; }
```

### GaussLegendreWeights3()

std::array< double, 3 > Garfield::Numerics::GaussLegendreWeights3 () [constexpr]

Definition at line 23 of file Numerics.hh.

```
00023                                       {
00024   return {5. / 9., 8. / 9., 5. / 9.};
00025 }
```

### GaussLegendreWeights4()

std::array< double, 4 > Garfield::Numerics::GaussLegendreWeights4 () [constexpr]

Definition at line 30 of file Numerics.hh.

```
00030                                       {
00031   return {0.347854845137453857, 0.652145154862546143, 0.652145154862546143,
00032           0.347854845137453857};
00033 }
```

### GaussLegendreWeights5()

std::array< double, 5 > Garfield::Numerics::GaussLegendreWeights5 () [constexpr]

Definition at line 38 of file Numerics.hh.

```
00038                                       {
00039   return {0.236926885056189088, 0.478628670499366468, 128. / 225.,
00040           0.478628670499366468, 0.236926885056189088};
00041 }
```

### GaussLegendreWeights6()

std::array< double, 6 > Garfield::Numerics::GaussLegendreWeights6 () [constexpr]

Definition at line 46 of file Numerics.hh.

```
00046                                       {
00047   return {0.171324492379170345, 0.360761573048138608, 0.467913934572691047,
00048           0.467913934572691047, 0.360761573048138608, 0.171324492379170345};
00049 }
```

### LeastSquaresFit()

```
bool Garfield::Numerics::LeastSquaresFit (
                std::function< double(double, const std::vector< double > &)> f,
                std::vector< double > & par,
                std::vector< double > & epar,
                const std::vector< double > & x,
                const std::vector< double > & y,
                const std::vector< double > & ey,
                const unsigned int nMaxIter,
                const double diff,
                double & chi2,
                const double eps,
                const bool debug,
                const bool verbose)
```

Least-squares minimisation.

### Legendre()

```
double Garfield::Numerics::Legendre (
                const unsigned int n,
                const double x)   [inline]
```

Legendre polynomials.

Definition at line 159 of file Numerics.hh.

```
00159                                                   {
00160   if (std::abs(x) > 1.) return 0.;
00161   double p0 = 1.;
00162   double p1 = x;
00163   if (n == 0) return p0;
00164   if (n == 1) return p1;
00165   for (unsigned int k = 1; k < n; ++k) {
00166     p0 = ((2 * k + 1) * x * p1 - k * p0) / (k + 1);
00167     std::swap(p0, p1);
00168   }
00169   return p1;
00170 }
```

### LinearInterpolation()

```
double Garfield::Numerics::LinearInterpolation (
                const std::vector< double > & ytab,
                const std::vector< double > & xtab,
                const double x)
```

## 6.5 Garfield::Numerics::CERNLIB Namespace Reference

Linear algebra routines from CERNLIB.

Functions

- int deqn (const int n, std::vector< std::vector< double > > &a, std::vector< double > &b)

  Replaces b by the solution x of Ax = b, after which A is undefined.
- int deqinv (const int n, std::vector< std::vector< double > > &a, std::vector< double > &b)

  Replaces b by the solution x of Ax = b, and replace A by its inverse.
- void dfact (const int n, std::vector< std::vector< double > > &a, std::vector< int > &ir, int &ifail, double &det, int &jfail)
- void dfeqn (const int n, std::vector< std::vector< double > > &a, std::vector< int > &ir, std::vector< double > &b)
- void dfinv (const int n, std::vector< std::vector< double > > &a, std::vector< int > &ir)
- int dinv (const int n, std::vector< std::vector< double > > &a)

  Replace square matrix A by its inverse.
- void cfact (const int n, std::vector< std::vector< std::complex< double > > > &a, std::vector< int > &ir, int &ifail, std::complex< double > &det, int &jfail)
- void cfinv (const int n, std::vector< std::vector< std::complex< double > > > &a, std::vector< int > &ir)
- int cinv (const int n, std::vector< std::vector< std::complex< double > > > &a)

  Replace square matrix A by its inverse.
- void cfft (std::vector< std::complex< double > > &a, const int msign)

### 6.5.1 Detailed Description

Linear algebra routines from CERNLIB.

### 6.5.2 Function Documentation

cfact()

```
void Garfield::Numerics::CERNLIB::cfact (
                const int n,
                std::vector< std::vector< std::complex< double > > > & a,
                std::vector< int > & ir,
                int & ifail,
                std::complex< double > & det,
                int & jfail)
```

cfft()

```
void Garfield::Numerics::CERNLIB::cfft (
                std::vector< std::complex< double > > & a,
                const int msign)
```

cfinv()

```
void Garfield::Numerics::CERNLIB::cfinv (
                const int n,
                std::vector< std::vector< std::complex< double > > > & a,
                std::vector< int > & ir)
```

cinv()

```
int Garfield::Numerics::CERNLIB::cinv (
                const int n,
                std::vector< std::vector< std::complex< double > > > & a)
```

Replace square matrix A by its inverse.

deqinv()

```
int Garfield::Numerics::CERNLIB::deqinv (
                const int n,
                std::vector< std::vector< double > > & a,
                std::vector< double > & b)
```

Replaces b by the solution x of Ax = b, and replace A by its inverse.

deqn()

```
int Garfield::Numerics::CERNLIB::deqn (
                const int n,
                std::vector< std::vector< double > > & a,
                std::vector< double > & b)
```

Replaces b by the solution x of Ax = b, after which A is undefined.

Parameters

| | |
|---|---|
| n | order of the square matrix A. |
| a | n by n matrix. |
| b | right-hand side vector. |

Returns

0: normal exit, -1: singular matrix.

dfact()

```
void Garfield::Numerics::CERNLIB::dfact (
                const int n,
                std::vector< std::vector< double > > & a,
                std::vector< int > & ir,
                int & ifail,
                double & det,
                int & jfail)
```

dfeqn()

```
void Garfield::Numerics::CERNLIB::dfeqn (
                const int n,
                std::vector< std::vector< double > > & a,
                std::vector< int > & ir,
                std::vector< double > & b)
```

dfinv()

```
void Garfield::Numerics::CERNLIB::dfinv (
                const int n,
                std::vector< std::vector< double > > & a,
                std::vector< int > & ir)
```

dinv()

```
int Garfield::Numerics::CERNLIB::dinv (
                const int n,
                std::vector< std::vector< double > > & a)
```

Replace square matrix A by its inverse.

## 6.6 Garfield::Numerics::QUADPACK Namespace Reference

Functions for performing numerical integration (quadrature).

Functions

- void qagi (std::function< double(double)> f, double bound, const int inf, const double epsabs, const double epsrel, double &result, double &abserr, unsigned int &status)

  Estimates an integral over a semi-infinite or infinite interval.
- void qk15i (std::function< double(double)> f, double bound, const int inf, const double a, const double b, double &result, double &abserr, double &resabs, double &resasc)

  15-point Gauss-Kronrod integration with (semi-)infinite integration range.
- void qk15 (std::function< double(double)> f, const double a, const double b, double &result, double &abserr, double &resabs, double &resasc)

  15-point Gauss-Kronrod integration with finite integration range.

### 6.6.1 Detailed Description

Functions for performing numerical integration (quadrature).

Reimplemented from QUADPACK.

R. Piessens, E. de Doncker-Kapenger, C. Ueberhuber, D. Kahaner, QUADPACK, a Subroutine Package for Automatic Integration, Springer, 1983

### 6.6.2 Function Documentation

qagi()

```
void Garfield::Numerics::QUADPACK::qagi (
                std::function< double(double)> f,
                double bound,
                const int inf,
                const double epsabs,
                const double epsrel,
                double & result,
                double & abserr,
                unsigned int & status)
```

Estimates an integral over a semi-infinite or infinite interval.

Calculates an approximation to an integral

$$I = \int_a^\infty f(x)\,dx,$$

or

$$I = \int_{-\infty}^a f(x)\,dx,$$

or

$$I = \int_{-\infty}^\infty f(x)\,dx$$

hopefully satisfying

$$|I - \text{RESULT}| \leq \max(\varepsilon_{\text{abs}}, \varepsilon_{\text{rel}}\,|I|).$$

Parameters

| | |
|---|---|
| f | function to be integrated. |
| bound | value of the finite endpoint of the integration range (if any). |
| inf | indicates the type of integration range.<br>• 1: ( bound, +Infinity)<br>• -1: (-Infinity, bound)<br>• 2: (-Infinity, +Infinity) |
| epsabs | requested absolute accuracy |
| epsrel | requested relative accuracy |
| result | the estimated value of the integral |
| abserr | estimated error |

Parameters

| | |
|---|---|
| status | error flag<br>• 0: normal and reliable termination, requested accuracy has been achieved.<br>• > 0: abnormal termination, estimates for result and error are less reliable.<br>– 1: maximum number of subdivisions reached.<br>– 2: occurance of roundoff error prevents the requested tolerance from being achieved. Error may be underestimated.<br>– 3: extremely bad integrand behaviour at some points of the integration interval.<br>– 4: algorithm does not converge, roundoff error is detected in the extrapolation table. It is assumed that the requested tolerance cannot be achieved and that the returned result is the best that can be obtained.<br>– 5: integral is probably divergent, or slowly convergent.<br>– 6: invalid input. |

qk15()

```
void Garfield::Numerics::QUADPACK::qk15 (
                std::function< double(double)> f,
                const double a,
                const double b,
                double & result,
                double & abserr,
                double & resabs,
                double & resasc)
```

15-point Gauss-Kronrod integration with finite integration range.

qk15i()

```
void Garfield::Numerics::QUADPACK::qk15i (
                std::function< double(double)> f,
                double bound,
                const int inf,
                const double a,
                const double b,
                double & result,
                double & abserr,
                double & resabs,
                double & resasc)
```

15-point Gauss-Kronrod integration with (semi-)infinite integration range.

Parameters

| | |
|---|---|
| f | function to be integrated. |
| bound | finite bound of original integration range (0 if inf = 2). |
| inf | indicates the type of integration range. |
| a | lower limit for integration over subrange of (0, 1). |

Parameters

| | |
|---|---|
| b | upper limit for integration over subrange of (0, 1). |
| result | approximation to the integral. |
| abserr | estimate of the modulus of the absolute error. |
| resabs | approximation to the integral over $\|f\|$. |
| resasc | approximation to the integral of $\|f - I/(b-a)\|$ over $(a, b)$. |

## 6.7 Garfield::Polygon Namespace Reference

Functions

- void Inside (const std::vector< double > &xpl, const std::vector< double > &ypl, const double x, const double y, bool &inside, bool &edge)

  Determine whether the point (x, y) is located inside of the polygon (xpl, ypl).
- double Area (const std::vector< double > &xp, const std::vector< double > &yp)

  Determine the (signed) area of a polygon.
- bool NonTrivial (const std::vector< double > &xp, const std::vector< double > &yp)

  Check whether a set of points builds a non-trivial polygon.
- void EliminateButterflies (std::vector< double > &xp, std::vector< double > &yp, std::vector< double > &zp)

  Try to eliminate "butterflies" (crossing of two adjacent segments of a polygon), by point exchanges.

### 6.7.1 Function Documentation

Area()

```
double Garfield::Polygon::Area (
            const std::vector< double > & xp,
            const std::vector< double > & yp)
```

Determine the (signed) area of a polygon.

EliminateButterflies()

```
void Garfield::Polygon::EliminateButterflies (
            std::vector< double > & xp,
            std::vector< double > & yp,
            std::vector< double > & zp)
```

Try to eliminate "butterflies" (crossing of two adjacent segments of a polygon), by point exchanges.

Inside()

```
void Garfield::Polygon::Inside (
                const std::vector< double > & xpl,
                const std::vector< double > & ypl,
                const double x,
                const double y,
                bool & inside,
                bool & edge)
```

Determine whether the point (x, y) is located inside of the polygon (xpl, ypl).

NonTrivial()

```
bool Garfield::Polygon::NonTrivial (
                const std::vector< double > & xp,
                const std::vector< double > & yp)
```

Check whether a set of points builds a non-trivial polygon.

## 6.8 Heed Namespace Reference

# 7 Class Documentation

## 7.1 Garfield::AvalancheGrid Class Reference

Calculate avalanches in a uniform electric field using avalanche statistics.

#include <AvalancheGrid.hh>

Classes

- struct AvalancheNode
- struct Path

Public Member Functions

- AvalancheGrid ()
  Default constructor.
- AvalancheGrid (Sensor *sensor)
  Constructor.
- ~AvalancheGrid ()=default
  Destructor.
- void SetSensor (Sensor *sensor)
  Set the sensor.
- void StartGridAvalanche ()
  Start grid based avalanche simulation.
- void SetElectronVelocity (const double vx, const double vy, const double vz)
  Set the electron drift velocity (in cm / ns).

- void SetElectronTownsend (const double town)

  Set the electron Townsend coefficient (in 1 / cm).
- void SetElectronAttachment (const double att)

  Set the electron attachment coefficient (in 1 / cm).
- void SetMaxAvalancheSize (const double size)

  Set the maximum avalanche size (1e7 by default).
- void AddElectron (const double x, const double y, const double z, const double t=0, const int n=1)

  Add an electron to the initial configuration.
- void AddElectrons (AvalancheMicroscopic *avmc)

  Import electrons from AvalancheMicroscopic object.
- void SetGrid (const double xmin, const double xmax, const int xsteps, const double ymin, const double ymax, const int ysteps, const double zmin, const double zmax, const int zsteps)

  Set the grid.
- int GetAmountOfStartingElectrons ()

  Returns the initial number of electrons in the avalanche.
- int GetAvalancheSize ()

  Returns the final number of electrons in the avalanche.
- void EnableDebugging ()
- void Reset ()

Private Member Functions

- bool SnapToGrid (const double x, const double y, const double z, const double v, const int n=1)
- void NextAvalancheGridPoint ()
- bool GetParameters (AvalancheNode &node)
- void DeactivateNode (AvalancheNode &node)

Private Attributes

- bool m_debug = false
- double m_Townsend = -1
- double m_Attachment = -1
- double m_Velocity = 0.
- std::vector< int > m_velNormal = {0, 0, 0}
- double m_MaxSize = 1.6e7
- bool m_Saturated = false
- double m_SaturationTime = -1.
- int m_nestart = 0.
- std::vector< double > m_nLayer
- std::string m_className = "AvalancheGrid"
- Sensor * m_sensor = nullptr
- bool m_printPar = false
- std::vector< double > m_zgrid

  Grid points of z-coordinate.
- double m_zStepSize = 0.

  Distance between the grid points.
- std::vector< double > m_ygrid

  Grid points of y-coordinate.
- double m_yStepSize = 0.

  Distance between the grid points.
- std::vector< double > m_xgrid

Grid points of x-coordinate.

- double m_xStepSize = 0.

  Distance between the grid points.
- bool m_gridset = false

  Keeps track if the grid has been defined.
- int m_nTotal = 0

  Total amount of charge.
- double m_time = 0

  Clock.
- bool m_run = true

  Tracking if the charges are still in the drift gap.
- std::vector< AvalancheNode > m_activeNodes = {}

### 7.1.1 Detailed Description

Calculate avalanches in a uniform electric field using avalanche statistics.

Definition at line 16 of file AvalancheGrid.hh.

### 7.1.2 Constructor & Destructor Documentation

#### AvalancheGrid() [1/2]

Garfield::AvalancheGrid::AvalancheGrid () [inline]

Default constructor.

Definition at line 19 of file AvalancheGrid.hh.

```
00019 : AvalancheGrid(nullptr) {}
```

#### AvalancheGrid() [2/2]

```
Garfield::AvalancheGrid::AvalancheGrid (
            Sensor * sensor)
```

Constructor.

#### ~AvalancheGrid()

Garfield::AvalancheGrid::~AvalancheGrid () [default]

Destructor.

### 7.1.3 Member Function Documentation

#### AddElectron()

```
void Garfield::AvalancheGrid::AddElectron (
            const double x,
            const double y,
            const double z,
            const double t = 0,
            const int n = 1)
```

Add an electron to the initial configuration.

Parameters

| | |
|---|---|
| x | x-coordinate of initial electron. |
| y | y-coordinate of initial electron. |
| z | t-coordinate of initial electron. |
| t | starting time of avalanche. |
| n | number of electrons at this point. |

### AddElectrons()

```
void Garfield::AvalancheGrid::AddElectrons (
                AvalancheMicroscopic * avmc)
```

Import electrons from AvalancheMicroscopic object.

### DeactivateNode()

```
void Garfield::AvalancheGrid::DeactivateNode (
                AvalancheNode & node)   [private]
```

### EnableDebugging()

```
void Garfield::AvalancheGrid::EnableDebugging ()   [inline]
```

Definition at line 79 of file AvalancheGrid.hh.

```
00079 { m_debug = true; }
```

### GetAmountOfStartingElectrons()

```
int Garfield::AvalancheGrid::GetAmountOfStartingElectrons ()   [inline]
```

Returns the initial number of electrons in the avalanche.

Definition at line 75 of file AvalancheGrid.hh.

```
00075 { return m_nestart; }
```

### GetAvalancheSize()

```
int Garfield::AvalancheGrid::GetAvalancheSize ()   [inline]
```

Returns the final number of electrons in the avalanche.

Definition at line 77 of file AvalancheGrid.hh.

```
00077 { return m_nTotal; }
```

### GetParameters()

bool Garfield::AvalancheGrid::GetParameters (
AvalancheNode & node) [private]

### NextAvalancheGridPoint()

void Garfield::AvalancheGrid::NextAvalancheGridPoint () [private]

### Reset()

void Garfield::AvalancheGrid::Reset ()

### SetElectronAttachment()

void Garfield::AvalancheGrid::SetElectronAttachment (
const double att) [inline]

Set the electron attachment coefficient (in 1 / cm).

Definition at line 44 of file AvalancheGrid.hh.

```
00044 { m_Attachment = att; }
```

### SetElectronTownsend()

void Garfield::AvalancheGrid::SetElectronTownsend (
const double town) [inline]

Set the electron Townsend coefficient (in 1 / cm).

Definition at line 42 of file AvalancheGrid.hh.

```
00042 { m_Townsend = town; }
```

### SetElectronVelocity()

void Garfield::AvalancheGrid::SetElectronVelocity (
const double vx,
const double vy,
const double vz) [inline]

Set the electron drift velocity (in cm / ns).

Definition at line 31 of file AvalancheGrid.hh.

```
00031                                                       {
00032     double vel = std::sqrt(vx * vx + vy * vy + vz * vz);
00033     if (vel != std::abs(vx) && vel != std::abs(vy) && vel != std::abs(vz))
00034       return;
00035     int nx = (int)vx / vel;
00036     int ny = (int)vy / vel;
00037     int nz = (int)vz / vel;
00038     m_velNormal = {nx, ny, nz};
00039     m_Velocity = -std::abs(vel);
00040   }
```

### SetGrid()

```
void Garfield::AvalancheGrid::SetGrid (
                const double xmin,
                const double xmax,
                const int xsteps,
                const double ymin,
                const double ymax,
                const int ysteps,
                const double zmin,
                const double zmax,
                const int zsteps)
```

Set the grid.

Parameters

| | |
|---|---|
| xmin,xmax | x-coordinate range of the grid [cm]. |
| xsteps | number of x-coordinate points in the grid. |
| ymin,ymax | y-coordinate range of the grid [cm]. |
| ysteps | number of y-coordinate points in the grid. |
| zmin,zmax | z-coordinate range of the grid [cm]. |
| zsteps | number of z-coordinate points in the grid. |

### SetMaxAvalancheSize()

```
void Garfield::AvalancheGrid::SetMaxAvalancheSize (
                const double size)    [inline]
```

Set the maximum avalanche size (1e7 by default).

Definition at line 46 of file AvalancheGrid.hh.

```
00046 { m_MaxSize = size; }
```

### SetSensor()

```
void Garfield::AvalancheGrid::SetSensor (
                Sensor * sensor)    [inline]
```

Set the sensor.

Definition at line 26 of file AvalancheGrid.hh.

```
00026 { m_sensor = sensor; }
```

### SnapToGrid()

```
bool Garfield::AvalancheGrid::SnapToGrid (
                const double x,
                const double y,
                const double z,
                const double v,
                const int n = 1)    [private]
```

StartGridAvalanche()

void Garfield::AvalancheGrid::StartGridAvalanche ()

Start grid based avalanche simulation.

### 7.1.4 Member Data Documentation

m_activeNodes

std::vector<AvalancheNode> Garfield::AvalancheGrid::m_activeNodes = {} [private]

Definition at line 154 of file AvalancheGrid.hh.
00154 {};

m_Attachment

double Garfield::AvalancheGrid::m_Attachment = -1 [private]

Definition at line 88 of file AvalancheGrid.hh.

m_className

std::string Garfield::AvalancheGrid::m_className = "AvalancheGrid" [private]

Definition at line 104 of file AvalancheGrid.hh.

m_debug

bool Garfield::AvalancheGrid::m_debug = false [private]

Definition at line 84 of file AvalancheGrid.hh.

m_gridset

bool Garfield::AvalancheGrid::m_gridset = false [private]

Keeps track if the grid has been defined.

Definition at line 119 of file AvalancheGrid.hh.

m_MaxSize

double Garfield::AvalancheGrid::m_MaxSize = 1.6e7 [private]

Definition at line 94 of file AvalancheGrid.hh.

### m_nestart

int Garfield::AvalancheGrid::m_nestart = 0. [private]

Definition at line 100 of file AvalancheGrid.hh.

### m_nLayer

std::vector<double> Garfield::AvalancheGrid::m_nLayer [private]

Definition at line 102 of file AvalancheGrid.hh.

### m_nTotal

int Garfield::AvalancheGrid::m_nTotal = 0 [private]

Total amount of charge.

Definition at line 120 of file AvalancheGrid.hh.

### m_printPar

bool Garfield::AvalancheGrid::m_printPar = false [private]

Definition at line 108 of file AvalancheGrid.hh.

### m_run

bool Garfield::AvalancheGrid::m_run = true [private]

Tracking if the charges are still in the drift gap.

Definition at line 122 of file AvalancheGrid.hh.

### m_Saturated

bool Garfield::AvalancheGrid::m_Saturated = false [private]

Definition at line 96 of file AvalancheGrid.hh.

### m_SaturationTime

double Garfield::AvalancheGrid::m_SaturationTime = -1. [private]

Definition at line 98 of file AvalancheGrid.hh.

### m_sensor

Sensor* Garfield::AvalancheGrid::m_sensor = nullptr [private]

Definition at line 106 of file AvalancheGrid.hh.

### m_time

double Garfield::AvalancheGrid::m_time = 0 [private]

Clock.

Definition at line 121 of file AvalancheGrid.hh.

### m_Townsend

double Garfield::AvalancheGrid::m_Townsend = -1 [private]

Definition at line 86 of file AvalancheGrid.hh.

### m_velNormal

std::vector<int> Garfield::AvalancheGrid::m_velNormal = {0, 0, 0} [private]

Definition at line 92 of file AvalancheGrid.hh.
00092 {0, 0, 0};

### m_Velocity

double Garfield::AvalancheGrid::m_Velocity = 0. [private]

Definition at line 90 of file AvalancheGrid.hh.

### m_xgrid

std::vector<double> Garfield::AvalancheGrid::m_xgrid [private]

Grid points of x-coordinate.

Definition at line 116 of file AvalancheGrid.hh.

### m_xStepSize

double Garfield::AvalancheGrid::m_xStepSize = 0. [private]

Distance between the grid points.

Definition at line 117 of file AvalancheGrid.hh.

### m_ygrid

std::vector<double> Garfield::AvalancheGrid::m_ygrid [private]

Grid points of y-coordinate.

Definition at line 113 of file AvalancheGrid.hh.

### m_yStepSize

double Garfield::AvalancheGrid::m_yStepSize = 0. [private]

Distance between the grid points.

Definition at line 114 of file AvalancheGrid.hh.

### m_zgrid

std::vector<double> Garfield::AvalancheGrid::m_zgrid [private]

Grid points of z-coordinate.

Definition at line 110 of file AvalancheGrid.hh.

### m_zStepSize

double Garfield::AvalancheGrid::m_zStepSize = 0. [private]

Distance between the grid points.

Definition at line 111 of file AvalancheGrid.hh.

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheGrid.hh

## 7.2 Garfield::AvalancheGridSpaceCharge Class Reference

Propagates avalanches with the 2d (axi-symmetric) space-charge routine from Lippmann, Riegler (2004) in uniform background fields.

#include <AvalancheGridSpaceCharge.hh>

### Classes

- struct GridNode
- struct Point

Public Member Functions

- AvalancheGridSpaceCharge ()

  Default constructor.
- AvalancheGridSpaceCharge (Sensor *sensor)

  Constructor.
- ~AvalancheGridSpaceCharge ()=default

  Destructor.
- void Reset ()

  Reset the charges.
- void EnableDebugging (const bool option=true)

  Enable/disable debugging (log messages) (default off)
- void EnableStickyAnode (const bool option=true)

  Enable sticky anode (default on)
- void EnableTOF (const bool option=true)

  Enable to use TOF swarm parameters (default on)
- void EnableDiffusion (const bool option=true)

  Enable diffusion (default off)
- void EnableSpaceChargeEffect (const bool option=true)

  Enable space charge calculations (default on)
- void EnableAdaptiveTimeStepping (const bool option=true)

  Enable adaptive time stepping (default on)
- void EnableMC (const bool option=true)

  Enable Monte Carlo (gain/diffusion) up to a certain total number of electrons e.g.
- void SetNCrit (const long NCrit=1e8)

  Set electron saturation value applied for each gas gap if space charge effect is turned off (default 1e8)
- void SetFieldCalculation (const std::string &option="coulomb", const int nof_approx=1.)

  Sets the different options to calculate the space charge field coulomb: free field approximation (relaxation: fdm-relaxation methode (with a condition to stop)) mirror: symmetric 3 layer single gap rpc with metal - resistive layer - gas gap - r.
- void SetK (float option=0.95)

  Set the streamer-inception criterion constant K in the interval (0, ∞) s.t.
- void SetStopAtK (bool option=true)

  Stop the avalanche if K % field is reached.
- void SetSensor (Sensor *sensor)

  Set the sensor (and determine if it includes a parallel-plate component).
- void Set2dGrid (double zmin, double zmax, int zsteps, double rmax, int rsteps)
- void AddElectrons (AvalancheMicroscopic *avmc)

  Import electron (no ions) data from AvalancheMicroscopic class to axi-symmetric grid.
- void AddElectron (double x, double y, double z, double t=0, int n=1)

  Set n electrons onto the grid.
- void AddExtraElectron (double y, int n=1)

  After calling AddElectron, add more electrons on the same transversal line (y-freedom).
- void StartGridAvalanche (double dtime=-1)

  Starts the simulation with the imported electrons for a time step dt (dt = -1 until there are no electrons left in the gap).
- long GetAvalancheSize () const

  Returns the total positive charge in the gap's.
- double GetMeanDistance ()

  Return current mean distance of the electrons on the grid.
- long ReachedKPercent () const

  Returns if 100 * K % background charge has been reached.

- const std::vector< std::pair< double, long > > GetElectronEvolution () const

  Returns the total electron number evolution.
- void ExportGrid (const std::string &filename)

  Export the current grid to a txt file (electron, ions numbers and field magnitude) Take care: only where electrons are located is the space-charge field evaluated!

Private Member Functions

- void PrepareElectronsFromMicroscopicAvalanche ()
- bool SnapTo2dGrid (double x, double y, double z, long n=1, int gasLayer=0)
- void Prepare2dMesh ()
- bool TransportTimeStep ()
- void DiffuseTimeStep (double dx, long nElectron, double nPosIon, double nNegIon, int iz, int ir, int gasGap)
- void DistributeCharges (long nElectron, double nPosIon, double nNegIon, int iz, int ir, double stepZ, double stepR, int gasGap)
- void GetLocalField (int iz, int ir, double &eFieldZ, double &eFieldR, const std::string &fieldOption, int gasGap)
- void GetFreeChargedRing (int iz, int ir, int fz, int fr, double &eFieldZ, double &eFieldR)
- void GetFreeChargedRing (double zi, double ri, double zf, double rf, double &eFieldZ, double &eFieldR)
- bool AddFieldFromChargeAt (int iz, int ir, int fz, int fr, double N, double &eFieldZ, double &eFieldR)
- bool AddFieldFromChargeAt (int iz, int ir, double zf, double rf, double N, double &eFieldZ, double &eFieldR)
- void GetSwarmParameters (double MagEField, double &alpha, double &eta, double &drift, double &dSigmaL, double &dSigmaT, double &wv, double &wr, double &alphaPT, double &etaPT, int gasGap)
- void GetGlobalCoordinates (double r, double z, double phi, double &xg, double &yg, double &zg, int gasGap)
- void ImportEllipticIntegralValues (const std::string &filename)
- void GetEllipticIntegrals (double x, double &K, double &E)
- int GetGasGapNumber (int layerIndex)

Private Attributes

- std::string m_className = "AvalancheGridSpaceCharge"
- bool m_bDebug = false
- bool m_bDiffusion = false
- bool m_bStick = true
- bool m_bDriftAvalanche = false
- bool m_bImportAvalanche = false
- bool m_bPreparedImportAvalanche = false
- long m_lNCrit = 1e8
- bool m_bSpaceCharge = true
- float m_fStreamerK = 0.95
- bool m_bStopAtK = false
- bool m_bFieldK = false
- long m_lElectronsK
- bool m_bAdaptiveTime = true
- bool m_bImportElliptic = false
- bool m_bUseTOF = true

  Flag if TOF parameters should be used, else Magboltz drift and SST spatial coefficients.

- bool m_bWrAvailable = true

  Flag if bulk drift velocity is available to the simulation.
- bool m_bRatesAvailable = true

  Flag if temporal rates are available to the simulation.
- bool m_bMC = true
- int m_iFieldApprox = 1
- double m_dMinGroups = 50
- ComponentParallelPlate * m_pp = nullptr
- Sensor * m_sensor = nullptr
- std::vector< double > m_zGrid

  Grid points of z-coordinate.
- int m_zSteps = 0

  Number of grid points.
- double m_zStepSize = 0.
- std::vector< double > m_rGrid

  Distance between the grid points.
- int m_rSteps = 0.

  Number of grid points.
- double m_rStepSize = 0.
- bool m_isgridset = false

  Distance between the grid points.
- long m_nTotElectron = 0

  Total amount of electrons at time step.
- long m_nTotPosIons = 0

  total amount of charge created
- double m_time = 0.

  Clock.
- double m_time0 = 0.

  Initial time.
- double m_dt = 0.

  Time step.
- bool m_run = true

  Tracking if the charges are still in the drift gap.
- std::vector< std::vector< int > > m_zGasGapBoundaries

  [k] -> {izLeft, ..., izRight}
- std::vector< std::vector< GridNode > > m_grid

  grid with nodes on it
- std::vector< std::vector< Point > > m_vElectrons

  Electrons to transfer onto grid.
- std::vector< std::pair< double, long > > m_vNElectronEvolution
- std::vector< long > m_vGroupSizes

  same values as Lippmann & Riegler
- std::vector< int > m_vIndexGasGaps = {0}

  Which layer indices are gas layers.
- std::vector< std::vector< double > > m_vCoNGasLayer {}

  Coordinates of center of electron number.
- std::vector< double > m_vYPointInGasGap {}

  Example point (y-coord) in each gas gap.
- std::vector< double > m_ezBkg = {0}

  Uniform background field in z direction, can be negative.
- std::vector< int > m_vSaturatedGaps {}

Which gas gaps are saturated if saturation is on.

- std::string m_sFieldOption = "coulomb"
- std::vector< double > m_vXElliptic
- std::vector< double > m_vKElliptic
- std::vector< double > m_vEElliptic

### 7.2.1 Detailed Description

Propagates avalanches with the 2d (axi-symmetric) space-charge routine from Lippmann, Riegler (2004) in uniform background fields.

Different options to calculate space-charge-fields can be chosen.

Definition at line 17 of file AvalancheGridSpaceCharge.hh.

### 7.2.2 Constructor & Destructor Documentation

#### AvalancheGridSpaceCharge() [1/2]

Garfield::AvalancheGridSpaceCharge::AvalancheGridSpaceCharge () [inline]

Default constructor.

Definition at line 20 of file AvalancheGridSpaceCharge.hh.

```
00020 : AvalancheGridSpaceCharge(nullptr) {}
```

#### AvalancheGridSpaceCharge() [2/2]

```
Garfield::AvalancheGridSpaceCharge::AvalancheGridSpaceCharge (
            Sensor * sensor)
```

Constructor.

#### ~AvalancheGridSpaceCharge()

Garfield::AvalancheGridSpaceCharge::~AvalancheGridSpaceCharge () [default]

Destructor.

### 7.2.3 Member Function Documentation

#### AddElectron()

```
void Garfield::AvalancheGridSpaceCharge::AddElectron (
            double x,
            double y,
            double z,
            double t = 0,
            int n = 1)
```

Set n electrons onto the grid.

### AddElectrons()

```
void Garfield::AvalancheGridSpaceCharge::AddElectrons (
            AvalancheMicroscopic * avmc)
```

Import electron (no ions) data from AvalancheMicroscopic class to axi-symmetric grid.

### AddExtraElectron()

```
void Garfield::AvalancheGridSpaceCharge::AddExtraElectron (
            double y,
            int n = 1)
```

After calling AddElectron, add more electrons on the same transversal line (y-freedom).

### AddFieldFromChargeAt() [1/2]

```
bool Garfield::AvalancheGridSpaceCharge::AddFieldFromChargeAt (
            int iz,
            int ir,
            double zf,
            double rf,
            double N,
            double & eFieldZ,
            double & eFieldR)   [private]
```

### AddFieldFromChargeAt() [2/2]

```
bool Garfield::AvalancheGridSpaceCharge::AddFieldFromChargeAt (
            int iz,
            int ir,
            int fz,
            int fr,
            double N,
            double & eFieldZ,
            double & eFieldR)   [private]
```

### DiffuseTimeStep()

```
void Garfield::AvalancheGridSpaceCharge::DiffuseTimeStep (
            double dx,
            long nElectron,
            double nPosIon,
            double nNegIon,
            int iz,
            int ir,
            int gasGap)   [private]
```

DistributeCharges()

```
void Garfield::AvalancheGridSpaceCharge::DistributeCharges (
                long nElectron,
                double nPosIon,
                double nNegIon,
                int iz,
                int ir,
                double stepZ,
                double stepR,
                int gasGap)   [private]
```

EnableAdaptiveTimeStepping()

```
void Garfield::AvalancheGridSpaceCharge::EnableAdaptiveTimeStepping (
                const bool option = true)   [inline]
```

Enable adaptive time stepping (default on)

Definition at line 47 of file AvalancheGridSpaceCharge.hh.

```
00047                                                {
00048     m_bAdaptiveTime = option;
00049   }
```

EnableDebugging()

```
void Garfield::AvalancheGridSpaceCharge::EnableDebugging (
                const bool option = true)   [inline]
```

Enable/disable debugging (log messages) (default off)

Definition at line 30 of file AvalancheGridSpaceCharge.hh.

```
00030 { m_bDebug = option; }
```

EnableDiffusion()

```
void Garfield::AvalancheGridSpaceCharge::EnableDiffusion (
                const bool option = true)   [inline]
```

Enable diffusion (default off)

Definition at line 39 of file AvalancheGridSpaceCharge.hh.

```
00039 { m_bDiffusion = option; }
```

EnableMC()

```
void Garfield::AvalancheGridSpaceCharge::EnableMC (
                const bool option = true)   [inline]
```

Enable Monte Carlo (gain/diffusion) up to a certain total number of electrons e.g.

(1e5) (default on) if disabled: Mean values are considered instead of the statistical processes

Definition at line 55 of file AvalancheGridSpaceCharge.hh.

```
00055 { m_bMC = option; }
```

EnableSpaceChargeEffect()

void Garfield::AvalancheGridSpaceCharge::EnableSpaceChargeEffect (
const bool option = true) [inline]

Enable space charge calculations (default on)

Definition at line 42 of file AvalancheGridSpaceCharge.hh.

```
00042                                          {
00043     m_bSpaceCharge = option;
00044   }
```

EnableStickyAnode()

void Garfield::AvalancheGridSpaceCharge::EnableStickyAnode (
const bool option = true) [inline]

Enable sticky anode (default on)

Definition at line 33 of file AvalancheGridSpaceCharge.hh.

```
00033 { m_bStick = option; }
```

EnableTOF()

void Garfield::AvalancheGridSpaceCharge::EnableTOF (
const bool option = true) [inline]

Enable to use TOF swarm parameters (default on)

Definition at line 36 of file AvalancheGridSpaceCharge.hh.

```
00036 { m_bUseTOF = option; }
```

ExportGrid()

void Garfield::AvalancheGridSpaceCharge::ExportGrid (
const std::string & filename)

Export the current grid to a txt file (electron, ions numbers and field magnitude) Take care: only where electrons are located is the space-charge field evaluated!

GetAvalancheSize()

long Garfield::AvalancheGridSpaceCharge::GetAvalancheSize () const [inline]

Returns the total positive charge in the gap's.

Definition at line 108 of file AvalancheGridSpaceCharge.hh.

```
00108 { return m_nTotPosIons; }
```

### GetElectronEvolution()

const std::vector< std::pair< double, long > > Garfield::AvalancheGridSpaceCharge::GetElectronEvolution () const [inline], [nodiscard]

Returns the total electron number evolution.

Definition at line 123 of file AvalancheGridSpaceCharge.hh.

```
00123                                {
00124     return m_vNElectronEvolution;
00125   }
```

### GetEllipticIntegrals()

void Garfield::AvalancheGridSpaceCharge::GetEllipticIntegrals (
double x,
double & K,
double & E) [private]

### GetFreeChargedRing() [1/2]

void Garfield::AvalancheGridSpaceCharge::GetFreeChargedRing (
double zi,
double ri,
double zf,
double rf,
double & eFieldZ,
double & eFieldR) [private]

### GetFreeChargedRing() [2/2]

void Garfield::AvalancheGridSpaceCharge::GetFreeChargedRing (
int iz,
int ir,
int fz,
int fr,
double & eFieldZ,
double & eFieldR) [private]

### GetGasGapNumber()

int Garfield::AvalancheGridSpaceCharge::GetGasGapNumber (
int layerIndex) [inline], [private]

Definition at line 244 of file AvalancheGridSpaceCharge.hh.

```
00244                                    {
00245     auto it =
00246         std::find(m_vIndexGasGaps.begin(), m_vIndexGasGaps.end(), layerIndex);
00247     return (it != m_vIndexGasGaps.end())
00248                ? std::distance(m_vIndexGasGaps.begin(), it)
00249                : -1;
00250   }
```

GetGlobalCoordinates()

```
void Garfield::AvalancheGridSpaceCharge::GetGlobalCoordinates (
                double r,
                double z,
                double phi,
                double & xg,
                double & yg,
                double & zg,
                int gasGap)   [private]
```

GetLocalField()

```
void Garfield::AvalancheGridSpaceCharge::GetLocalField (
                int iz,
                int ir,
                double & eFieldZ,
                double & eFieldR,
                const std::string & fieldOption,
                int gasGap)   [private]
```

GetMeanDistance()

```
double Garfield::AvalancheGridSpaceCharge::GetMeanDistance ()
```

Return current mean distance of the electrons on the grid.

GetSwarmParameters()

```
void Garfield::AvalancheGridSpaceCharge::GetSwarmParameters (
                double MagEField,
                double & alpha,
                double & eta,
                double & drift,
                double & dSigmaL,
                double & dSigmaT,
                double & wv,
                double & wr,
                double & alphaPT,
                double & etaPT,
                int gasGap)   [private]
```

ImportEllipticIntegralValues()

```
void Garfield::AvalancheGridSpaceCharge::ImportEllipticIntegralValues (
                const std::string & filename)   [private]
```

Prepare2dMesh()

```
void Garfield::AvalancheGridSpaceCharge::Prepare2dMesh ()   [private]
```

### PrepareElectronsFromMicroscopicAvalanche()

void Garfield::AvalancheGridSpaceCharge::PrepareElectronsFromMicroscopicAvalanche () [private]

### ReachedKPercent()

long Garfield::AvalancheGridSpaceCharge::ReachedKPercent () const [inline], [nodiscard]

Returns if 100 ∗ K % background charge has been reached.

Definition at line 114 of file AvalancheGridSpaceCharge.hh.

```
00114                                       {
00115     if (m_bFieldK)
00116       return m_lElectronsK;
00117     else
00118       return -1;
00119   }
```

### Reset()

void Garfield::AvalancheGridSpaceCharge::Reset ()

Reset the charges.

### Set2dGrid()

void Garfield::AvalancheGridSpaceCharge::Set2dGrid (
double zmin,
double zmax,
int zsteps,
double rmax,
int rsteps)

Parameters

| | |
|---|---|
| zmin | coordinate along direction of background field |
| zmax | should more or less be at the end of the gap |
| zsteps | number of steps |
| rmax | coordinate perpendicular to background field |
| rsteps | number of steps |

### SetFieldCalculation()

void Garfield::AvalancheGridSpaceCharge::SetFieldCalculation (
const std::string & option = ”coulomb”,
const int nof_approx = 1.) [inline]

Sets the different options to calculate the space charge field coulomb: free field approximation (relaxation: fdm-relaxation methode (with a condition to stop)) mirror: symmetric 3 layer single gap rpc with metal - resistive layer - gas gap - r.

l. - m.

Definition at line 66 of file AvalancheGridSpaceCharge.hh.

```
00067                                                {
00068     m_sFieldOption = std::move(option);
00069     m_iFieldApprox = std::move(nof_approx);
00070   }
```

SetK()

void Garfield::AvalancheGridSpaceCharge::SetK (
float option = 0.95) [inline]

Set the streamer-inception criterion constant K in the interval (0, $\infty$) s.t.

1 = 100%

Definition at line 74 of file AvalancheGridSpaceCharge.hh.
00074 { m_fStreamerK = option; }

SetNCrit()

void Garfield::AvalancheGridSpaceCharge::SetNCrit (
const long NCrit = 1e8) [inline]

Set electron saturation value applied for each gas gap if space charge effect is turned off (default 1e8)

Definition at line 59 of file AvalancheGridSpaceCharge.hh.
00059 { m_lNCrit = NCrit; }

SetSensor()

void Garfield::AvalancheGridSpaceCharge::SetSensor (
Sensor * sensor)

Set the sensor (and determine if it includes a parallel-plate component).

SetStopAtK()

void Garfield::AvalancheGridSpaceCharge::SetStopAtK (
bool option = true) [inline]

Stop the avalanche if K % field is reached.

Definition at line 77 of file AvalancheGridSpaceCharge.hh.
00077 { m_bStopAtK = option; }

SnapTo2dGrid()

bool Garfield::AvalancheGridSpaceCharge::SnapTo2dGrid (
double x,
double y,
double z,
long n = 1,
int gasLayer = 0) [private]

StartGridAvalanche()

void Garfield::AvalancheGridSpaceCharge::StartGridAvalanche (
double dtime = -1)

Starts the simulation with the imported electrons for a time step dt (dt = -1 until there are no electrons left in the gap).

TransportTimeStep()

bool Garfield::AvalancheGridSpaceCharge::TransportTimeStep () [private]

### 7.2.4 Member Data Documentation

m_bAdaptiveTime

bool Garfield::AvalancheGridSpaceCharge::m_bAdaptiveTime = true [private]

Definition at line 271 of file AvalancheGridSpaceCharge.hh.

m_bDebug

bool Garfield::AvalancheGridSpaceCharge::m_bDebug = false [private]

Definition at line 255 of file AvalancheGridSpaceCharge.hh.

m_bDiffusion

bool Garfield::AvalancheGridSpaceCharge::m_bDiffusion = false [private]

Definition at line 256 of file AvalancheGridSpaceCharge.hh.

m_bDriftAvalanche

bool Garfield::AvalancheGridSpaceCharge::m_bDriftAvalanche = false [private]

Definition at line 259 of file AvalancheGridSpaceCharge.hh.

m_bFieldK

bool Garfield::AvalancheGridSpaceCharge::m_bFieldK = false [private]

Definition at line 268 of file AvalancheGridSpaceCharge.hh.

### m_bImportAvalanche

bool Garfield::AvalancheGridSpaceCharge::m_bImportAvalanche = false [private]

Definition at line 261 of file AvalancheGridSpaceCharge.hh.

### m_bImportElliptic

bool Garfield::AvalancheGridSpaceCharge::m_bImportElliptic = false [private]

Definition at line 272 of file AvalancheGridSpaceCharge.hh.

### m_bMC

bool Garfield::AvalancheGridSpaceCharge::m_bMC = true [private]

Definition at line 281 of file AvalancheGridSpaceCharge.hh.

### m_bPreparedImportAvalanche

bool Garfield::AvalancheGridSpaceCharge::m_bPreparedImportAvalanche = false [private]

Definition at line 262 of file AvalancheGridSpaceCharge.hh.

### m_bRatesAvailable

bool Garfield::AvalancheGridSpaceCharge::m_bRatesAvailable = true [private]

Flag if temporal rates are available to the simulation.

Definition at line 279 of file AvalancheGridSpaceCharge.hh.

### m_bSpaceCharge

bool Garfield::AvalancheGridSpaceCharge::m_bSpaceCharge = true [private]

Definition at line 264 of file AvalancheGridSpaceCharge.hh.

### m_bStick

bool Garfield::AvalancheGridSpaceCharge::m_bStick = true [private]

Definition at line 257 of file AvalancheGridSpaceCharge.hh.

### m_bStopAtK

bool Garfield::AvalancheGridSpaceCharge::m_bStopAtK = false [private]

Definition at line 267 of file AvalancheGridSpaceCharge.hh.

### m_bUseTOF

bool Garfield::AvalancheGridSpaceCharge::m_bUseTOF = true [private]

Flag if TOF parameters should be used, else Magboltz drift and SST spatial coefficients.

Definition at line 275 of file AvalancheGridSpaceCharge.hh.

### m_bWrAvailable

bool Garfield::AvalancheGridSpaceCharge::m_bWrAvailable = true [private]

Flag if bulk drift velocity is available to the simulation.

Definition at line 277 of file AvalancheGridSpaceCharge.hh.

### m_className

std::string Garfield::AvalancheGridSpaceCharge::m_className = "AvalancheGridSpaceCharge" [private]

Definition at line 253 of file AvalancheGridSpaceCharge.hh.

### m_dMinGroups

double Garfield::AvalancheGridSpaceCharge::m_dMinGroups = 50 [private]

Definition at line 284 of file AvalancheGridSpaceCharge.hh.

### m_dt

double Garfield::AvalancheGridSpaceCharge::m_dt = 0. [private]

Time step.

Definition at line 303 of file AvalancheGridSpaceCharge.hh.

### m_ezBkg

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_ezBkg = {0} [private]

Uniform background field in z direction, can be negative.

Definition at line 328 of file AvalancheGridSpaceCharge.hh.
00328 {0};

### m_fStreamerK

float Garfield::AvalancheGridSpaceCharge::m_fStreamerK = 0.95 [private]

Definition at line 266 of file AvalancheGridSpaceCharge.hh.

### m_grid

std::vector<std::vector<GridNode> > Garfield::AvalancheGridSpaceCharge::m_grid [private]

grid with nodes on it

Definition at line 310 of file AvalancheGridSpaceCharge.hh.

### m_iFieldApprox

int Garfield::AvalancheGridSpaceCharge::m_iFieldApprox = 1 [private]

Definition at line 283 of file AvalancheGridSpaceCharge.hh.

### m_isgridset

bool Garfield::AvalancheGridSpaceCharge::m_isgridset = false [private]

Distance between the grid points.

Keeps track if the grid has been defined.

Definition at line 297 of file AvalancheGridSpaceCharge.hh.

### m_lElectronsK

long Garfield::AvalancheGridSpaceCharge::m_lElectronsK [private]

Definition at line 269 of file AvalancheGridSpaceCharge.hh.

### m_lNCrit

long Garfield::AvalancheGridSpaceCharge::m_lNCrit = 1e8 [private]

Definition at line 263 of file AvalancheGridSpaceCharge.hh.

### m_nTotElectron

long Garfield::AvalancheGridSpaceCharge::m_nTotElectron = 0 [private]

Total amount of electrons at time step.

Definition at line 298 of file AvalancheGridSpaceCharge.hh.

### m_nTotPosIons

long Garfield::AvalancheGridSpaceCharge::m_nTotPosIons = 0 [private]

total amount of charge created

Definition at line 299 of file AvalancheGridSpaceCharge.hh.

### m_pp

ComponentParallelPlate* Garfield::AvalancheGridSpaceCharge::m_pp = nullptr [private]

Definition at line 286 of file AvalancheGridSpaceCharge.hh.

### m_rGrid

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_rGrid [private]

Distance between the grid points.

Grid points of r-coordinate.

Definition at line 293 of file AvalancheGridSpaceCharge.hh.

### m_rSteps

int Garfield::AvalancheGridSpaceCharge::m_rSteps = 0. [private]

Number of grid points.

Definition at line 294 of file AvalancheGridSpaceCharge.hh.

### m_rStepSize

double Garfield::AvalancheGridSpaceCharge::m_rStepSize = 0. [private]

Definition at line 295 of file AvalancheGridSpaceCharge.hh.

### m_run

bool Garfield::AvalancheGridSpaceCharge::m_run = true [private]

Tracking if the charges are still in the drift gap.

Definition at line 305 of file AvalancheGridSpaceCharge.hh.

### m_sensor

Sensor* Garfield::AvalancheGridSpaceCharge::m_sensor = nullptr [private]

Definition at line 287 of file AvalancheGridSpaceCharge.hh.

### m_sFieldOption

std::string Garfield::AvalancheGridSpaceCharge::m_sFieldOption = "coulomb" [private]

Definition at line 332 of file AvalancheGridSpaceCharge.hh.

### m_time

double Garfield::AvalancheGridSpaceCharge::m_time = 0. [private]

Clock.

Definition at line 301 of file AvalancheGridSpaceCharge.hh.

### m_time0

double Garfield::AvalancheGridSpaceCharge::m_time0 = 0. [private]

Initial time.

Definition at line 302 of file AvalancheGridSpaceCharge.hh.

### m_vCoNGasLayer

std::vector<std::vector<double> > Garfield::AvalancheGridSpaceCharge::m_vCoNGasLayer {} [private]

Coordinates of center of electron number.

Required: y in [zmin, zmax]

Definition at line 323 of file AvalancheGridSpaceCharge.hh.
00323 {};

### m_vEElliptic

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_vEElliptic [private]

Definition at line 335 of file AvalancheGridSpaceCharge.hh.

### m_vElectrons

std::vector<std::vector<Point> > Garfield::AvalancheGridSpaceCharge::m_vElectrons [private]

Electrons to transfer onto grid.

Definition at line 312 of file AvalancheGridSpaceCharge.hh.

### m_vGroupSizes

std::vector<long> Garfield::AvalancheGridSpaceCharge::m_vGroupSizes [private]

Initial value:
```
= {
    1500, 800, 400, 200, 100,
    50,   20,  10,  5,   2}
```

same values as Lippmann & Riegler

Definition at line 315 of file AvalancheGridSpaceCharge.hh.
```
00315                                {
00316       1500, 800, 400, 200, 100,
00317       50,   20,  10,  5,   2};
```

### m_vIndexGasGaps

std::vector<int> Garfield::AvalancheGridSpaceCharge::m_vIndexGasGaps = {0} [private]

Which layer indices are gas layers.

Definition at line 319 of file AvalancheGridSpaceCharge.hh.
```
00319 {0};
```

### m_vKElliptic

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_vKElliptic [private]

Definition at line 334 of file AvalancheGridSpaceCharge.hh.

### m_vNElectronEvolution

std::vector<std::pair<double, long> > Garfield::AvalancheGridSpaceCharge::m_vNElectronEvolution [private]

Definition at line 314 of file AvalancheGridSpaceCharge.hh.

### m_vSaturatedGaps

std::vector<int> Garfield::AvalancheGridSpaceCharge::m_vSaturatedGaps {} [private]

Which gas gaps are saturated if saturation is on.

Definition at line 330 of file AvalancheGridSpaceCharge.hh.
```
00330 {};
```

### m_vXElliptic

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_vXElliptic [private]

Definition at line 333 of file AvalancheGridSpaceCharge.hh.

### m_vYPointInGasGap

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_vYPointInGasGap {} [private]

Example point (y-coord) in each gas gap.

Definition at line 325 of file AvalancheGridSpaceCharge.hh.
00325 {};

### m_zGasGapBoundaries

std::vector<std::vector<int> > Garfield::AvalancheGridSpaceCharge::m_zGasGapBoundaries [private]

[k] -> {izLeft, ..., izRight}

Definition at line 308 of file AvalancheGridSpaceCharge.hh.

### m_zGrid

std::vector<double> Garfield::AvalancheGridSpaceCharge::m_zGrid [private]

Grid points of z-coordinate.

Definition at line 289 of file AvalancheGridSpaceCharge.hh.

### m_zSteps

int Garfield::AvalancheGridSpaceCharge::m_zSteps = 0 [private]

Number of grid points.

Definition at line 290 of file AvalancheGridSpaceCharge.hh.

### m_zStepSize

double Garfield::AvalancheGridSpaceCharge::m_zStepSize = 0. [private]

Definition at line 291 of file AvalancheGridSpaceCharge.hh.

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheGridSpaceCharge.hh

## 7.3 Garfield::AvalancheMC Class Reference

Calculate drift lines and avalanches based on macroscopic transport coefficients, using Monte Carlo integration.

#include <AvalancheMC.hh>

Classes

- struct EndPoint
- struct Point
- struct Seed

Public Member Functions

- AvalancheMC ()

  Default constructor.
- AvalancheMC (Sensor *sensor)

  Constructor.
- ~AvalancheMC ()

  Destructor.
- void SetSensor (Sensor *s)

  Set the sensor.
- bool DriftElectron (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of an electron from a given starting point.
- bool DriftHole (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of a hole from a given starting point.
- bool DriftIon (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of an ion from a given starting point.
- bool DriftNegativeIon (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of a negative ion from a given starting point.
- bool AvalancheElectron (const double x, const double y, const double z, const double t, const bool hole=false, const size_t w=1)

  Simulate an avalanche initiated by an electron at a given starting point.
- bool AvalancheHole (const double x, const double y, const double z, const double t, const bool electron=false, const size_t w=1)

  Simulate an avalanche initiated by a hole at a given starting point.
- bool AvalancheElectronHole (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate an avalanche initiated by an electron-hole pair.
- void AddElectron (const double x, const double y, const double z, const double t, const size_t w=1)

  Add an electron to the list of particles to be transported.
- void AddHole (const double x, const double y, const double z, const double t, const size_t w=1)

  Add a hole to the list of particles to be transported.
- void AddIon (const double x, const double y, const double z, const double t, const size_t w=1)

  Add an ion to the list of particles to be transported.
- void AddNegativeIon (const double x, const double y, const double z, const double t, const size_t w=1)

  Add an negative ion to the list of particles to be transported.

- bool ResumeAvalanche (const bool electron=true, const bool hole=true)

  Resume the simulation from the current set of charge carriers.
- const std::vector< EndPoint > & GetElectrons () const
- const std::vector< EndPoint > & GetHoles () const
- const std::vector< EndPoint > & GetIons () const
- const std::vector< EndPoint > & GetNegativeIons () const
- size_t GetNumberOfElectronEndpoints () const

  Return the number of electron trajectories in the last simulated avalanche (including captured electrons).
- size_t GetNumberOfIonEndpoints () const

  Return the number of ion trajectories.
- void GetElectronEndpoint (const size_t i, double &x0, double &y0, double &z0, double &t0, double &x1, double &y1, double &z1, double &t1, int &status) const

  Return the coordinates and time of start and end point of a given electron drift line.
- void GetIonEndpoint (const size_t i, double &x0, double &y0, double &z0, double &t0, double &x1, double &y1, double &z1, double &t1, int &status) const
- void GetNegativeIonEndpoint (const size_t i, double &x0, double &y0, double &z0, double &t0, double &x1, double &y1, double &z1, double &t1, int &status) const
- void EnablePlotting (ViewDrift *view)

  Switch on drift line plotting.
- void DisablePlotting ()

  Switch off drift line plotting.
- void EnableDriftLines (const bool on=true)

  Switch on storage of drift lines (default: off).
- void EnableSignalCalculation (const bool on=true)

  Switch calculation of induced currents on or off (default: enabled).
- void SetSignalAveragingOrder (const unsigned int navg)

  Set the number of points to be used when averaging the signal vector over a time bin in the Sensor class.
- void UseWeightingPotential (const bool on=true)

  Use the weighting potential (as opposed to the weighting field) for calculating the induced signal.
- void EnableInducedChargeCalculation (const bool on=true)

  Switch on calculation of induced charge (default: disabled).
- void EnableRKFSteps (const bool on=true)

  Switch on Runge-Kutta-Fehlberg stepping (as opposed to simple straight-line steps.
- void EnableProjectedPathIntegration (const bool on=true)

  Switch on equilibration of multiplication and attachment over the drift line (default: enabled).
- void EnableDiffusion ()

  Switch on diffusion (default: enabled).
- void DisableDiffusion ()

  Switch off diffusion.
- void EnableAttachment ()

  Switch on attachment (default: enabled).
- void DisableAttachment ()

  Switch off attachment and multiplication.
- void EnableMultiplication (const bool on)

  Switch multiplication on/off.
- void EnableTownsendMap (const bool on=true)

  Retrieve the Townsend coefficient from the component.
- void EnableAttachmentMap (const bool on=true)

  Retrieve the attachment coefficient from the component.
- void EnableMobilityMap (const bool on=true)

  Retrieve the (low-field) mobility from the component.

- void EnableVelocityMap (const bool on=true)

  Retrieve the drift velocity from the component.
- void EnableAvalancheSizeLimit (const unsigned int size)

  Set a maximum avalanche size (ignore further multiplication once this size has been reached).
- void DisableAvalancheSizeLimit ()

  Do not limit the maximum avalanche size.
- int GetAvalancheSizeLimit () const

  Return the currently set avalanche size limit.
- void SetTimeSteps (const double d=0.02)

  Use fixed-time steps (default 20 ps).
- void SetDistanceSteps (const double d=0.001)

  Use fixed distance steps (default 10 um).
- void SetCollisionSteps (const unsigned int n=100)

  Use exponentially distributed time steps with mean equal to the specified multiple of the collision time (default model).
- void SetStepDistanceFunction (double(*f)(double x, double y, double z))

  Retrieve the step distance from a user-supplied function.
- void SetTimeWindow (const double t0, const double t1)

  Define a time interval (only carriers inside the interval are drifted).
- void UnsetTimeWindow ()

  Do not limit the time interval within which carriers are drifted.
- void SetElectronSignalScalingFactor (const double scale)

  Set multiplication factor for the signal induced by electrons.
- void SetHoleSignalScalingFactor (const double scale)

  Set multiplication factor for the signal induced by holes.
- void SetIonSignalScalingFactor (const double scale)

  Set multiplication factor for the signal induced by ions.
- void GetAvalancheSize (unsigned int &ne, unsigned int &ni) const

  Return the number of electrons and ions/holes in the avalanche.
- std::pair< unsigned int, unsigned int > GetAvalancheSize () const

  Return the number of electrons and ions/holes in the avalanche.
- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off (default: off).

Private Types

- enum class StepModel { FixedTime , FixedDistance , CollisionTime , UserDistance }

Private Member Functions

- int DriftLine (const Seed &seed, std::vector< Point > &path, std::vector< Seed > &secondaries, const bool aval, const bool signal) const

  Compute a single drift line.
- bool TransportParticles (std::vector< Seed > &stack, const bool withElectrons, const bool withHoles, const bool aval)

  Compute an avalanche.
- int GetField (const std::array< double, 3 > &x, std::array< double, 3 > &e, std::array< double, 3 > &b, Medium *&medium) const

  Compute electric and magnetic field at a given position.

- double GetMobility (const Particle particle, Medium *medium, const std::array< double, 3 > &x) const

  Retrieve the low-field mobility.
- bool GetVelocity (const Particle particle, Medium *medium, const std::array< double, 3 > &x, const std::array< double, 3 > &e, const std::array< double, 3 > &b, std::array< double, 3 > &v) const

  Compute the drift velocity.
- bool GetDiffusion (const Particle particle, Medium *medium, const std::array< double, 3 > &e, const std::array< double, 3 > &b, double &dl, double &dt) const

  Compute the diffusion coefficients.
- double GetAttachment (const Particle particle, Medium *medium, const std::array< double, 3 > &x, const std::array< double, 3 > &e, const std::array< double, 3 > &b) const

  Compute the attachment coefficient.
- double GetTownsend (const Particle particle, Medium *medium, const std::array< double, 3 > &x, const std::array< double, 3 > &e, const std::array< double, 3 > &b) const

  Compute the Townsend coefficient.
- void StepRKF (const Particle particle, const std::array< double, 3 > &x0, const std::array< double, 3 > &v0, const double dt, std::array< double, 3 > &xf, std::array< double, 3 > &vf, int &status) const

  Compute end point and effective velocity for a step.
- void AddDiffusion (const double step, const double dl, const double dt, std::array< double, 3 > &x, const std::array< double, 3 > &v) const

  Add a diffusion step.
- void Terminate (const std::array< double, 3 > &x0, const double t0, std::array< double, 3 > &x, double &t) const

  Terminate a drift line close to the boundary.
- bool ComputeGainLoss (const Particle ptype, const size_t w, std::vector< Point > &path, int &status, std::vector< Seed > &secondaries, const bool semiconductor=false) const

  Compute multiplication and losses along the current drift line.
- bool ComputeAlphaEta (const Particle ptype, std::vector< Point > &path, std::vector< double > &alphas, std::vector< double > &etas) const

  Compute Townsend and attachment coefficients along the current drift line.
- bool Equilibrate (std::vector< double > &alphas) const
- void ComputeSignal (const double q, const std::vector< Point > &path) const

  Compute the induced signal for the current drift line.
- void ComputeInducedCharge (const double q, const std::vector< Point > &path) const

  Compute the induced charge for the current drift line.
- void PrintError (const std::string &fcn, const std::string &par, const Particle particle, const std::array< double, 3 > &x) const

Private Attributes

- std::string m_className = "AvalancheMC"
- Sensor * m_sensor = nullptr
- StepModel m_stepModel = StepModel::CollisionTime

  Step size model.
- double m_tMc = 0.02

  Fixed time step.
- double m_dMc = 0.001

  Fixed distance step.
- int m_nMc = 100

  Sample step size according to collision time.

- double(* m_fStep )(double x, double y, double z) = nullptr
  User function returning the step size.
- bool m_hasTimeWindow = false
  Flag whether a time window should be used.
- double m_tMin = 0.
  Lower limit of the time window.
- double m_tMax = 0.
  Upper limit of the time window.
- unsigned int m_sizeCut = 0
  Max. avalanche size.
- unsigned int m_nElectrons = 0
  Number of electrons produced.
- unsigned int m_nHoles = 0
  Number of holes produced.
- unsigned int m_nIons = 0
  Number of ions produced.
- unsigned int m_nNegativeIons = 0
  Number of negative ions produced.
- std::vector< EndPoint > m_electrons
  Start/end points of all electrons in the avalanche (including captured ones).
- std::vector< EndPoint > m_holes
  Start/end points of all holes in the avalanche (including captured ones).
- std::vector< EndPoint > m_ions
  Start/end points of all positive ions in the avalanche.
- std::vector< EndPoint > m_negativeIons
  Start/end points of all negative ions in the avalanche.
- ViewDrift * m_viewer = nullptr
- bool m_storeDriftLines = false
- bool m_doSignal = true
- unsigned int m_navg = 1
- bool m_useWeightingPotential = true
- bool m_doInducedCharge = false
- bool m_doEquilibration = true
- bool m_doRKF = false
- bool m_useDiffusion = true
- bool m_useAttachment = true
- bool m_useMultiplication = true
- double m_scaleE = 1.
  Scaling factor for electron signals.
- double m_scaleH = 1.
  Scaling factor for hole signals.
- double m_scaleI = 1.
  Scaling factor for ion signals.
- bool m_useTownsendMap = false
  Take Townsend coefficients from the component.
- bool m_useAttachmentMap = false
  Take attachment coefficients from the component.
- bool m_useMobilityMap = false
  Take mobility coefficients from the component.
- bool m_useVelocityMap = false
  Take the drift velocities from the component.
- bool m_debug = false

### 7.3.1 Detailed Description

Calculate drift lines and avalanches based on macroscopic transport coefficients, using Monte Carlo integration.

Definition at line 18 of file AvalancheMC.hh.

### 7.3.2 Member Enumeration Documentation

#### StepModel

enum class Garfield::AvalancheMC::StepModel [strong], [private]

Enumerator

| | |
|---|---|
| FixedTime | |
| FixedDistance | |
| CollisionTime | |
| UserDistance | |

Definition at line 224 of file AvalancheMC.hh.

```
00224                   {
00225     FixedTime,
00226     FixedDistance,
00227     CollisionTime,
00228     UserDistance
00229   };
```

### 7.3.3 Constructor & Destructor Documentation

#### AvalancheMC() [1/2]

Garfield::AvalancheMC::AvalancheMC () [inline]

Default constructor.

Definition at line 21 of file AvalancheMC.hh.

```
00021 : AvalancheMC(nullptr) {}
```

#### AvalancheMC() [2/2]

Garfield::AvalancheMC::AvalancheMC (
Sensor * sensor)

Constructor.

#### ~AvalancheMC()

Garfield::AvalancheMC::~AvalancheMC () [inline]

Destructor.

Definition at line 25 of file AvalancheMC.hh.

```
00025 {}
```

### 7.3.4 Member Function Documentation

#### AddDiffusion()

```
void Garfield::AvalancheMC::AddDiffusion (
                const double step,
                const double dl,
                const double dt,
                std::array< double, 3 > & x,
                const std::array< double, 3 > & v) const    [private]
```

Add a diffusion step.

#### AddElectron()

```
void Garfield::AvalancheMC::AddElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Add an electron to the list of particles to be transported.

#### AddHole()

```
void Garfield::AvalancheMC::AddHole (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Add a hole to the list of particles to be transported.

#### AddIon()

```
void Garfield::AvalancheMC::AddIon (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Add an ion to the list of particles to be transported.

#### AddNegativeIon()

```
void Garfield::AvalancheMC::AddNegativeIon (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Add an negative ion to the list of particles to be transported.

AvalancheElectron()

```
bool Garfield::AvalancheMC::AvalancheElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const bool hole = false,
                const size_t w = 1)
```

Simulate an avalanche initiated by an electron at a given starting point.

Parameters

| | |
|---|---|
| x,y,z,t | coordinates and time of the initial electron |
| hole | simulate the hole component of the avalanche or not |
| w | multiplicity of the initial electron |

AvalancheElectronHole()

```
bool Garfield::AvalancheMC::AvalancheElectronHole (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Simulate an avalanche initiated by an electron-hole pair.

AvalancheHole()

```
bool Garfield::AvalancheMC::AvalancheHole (
                const double x,
                const double y,
                const double z,
                const double t,
                const bool electron = false,
                const size_t w = 1)
```

Simulate an avalanche initiated by a hole at a given starting point.

ComputeAlphaEta()

```
bool Garfield::AvalancheMC::ComputeAlphaEta (
                const Particle ptype,
                std::vector< Point > & path,
                std::vector< double > & alphas,
                std::vector< double > & etas) const    [private]
```

Compute Townsend and attachment coefficients along the current drift line.

### ComputeGainLoss()

```
bool Garfield::AvalancheMC::ComputeGainLoss (
                const Particle ptype,
                const size_t w,
                std::vector< Point > & path,
                int & status,
                std::vector< Seed > & secondaries,
                const bool semiconductor = false) const    [private]
```

Compute multiplication and losses along the current drift line.

### ComputeInducedCharge()

```
void Garfield::AvalancheMC::ComputeInducedCharge (
                const double q,
                const std::vector< Point > & path) const    [private]
```

Compute the induced charge for the current drift line.

### ComputeSignal()

```
void Garfield::AvalancheMC::ComputeSignal (
                const double q,
                const std::vector< Point > & path) const    [private]
```

Compute the induced signal for the current drift line.

### DisableAttachment()

```
void Garfield::AvalancheMC::DisableAttachment ()    [inline]
```

Switch off attachment and multiplication.

Definition at line 164 of file AvalancheMC.hh.

```
00164 { m_useAttachment = false; }
```

### DisableAvalancheSizeLimit()

```
void Garfield::AvalancheMC::DisableAvalancheSizeLimit ()    [inline]
```

Do not limit the maximum avalanche size.

Definition at line 181 of file AvalancheMC.hh.

```
00181 { m_sizeCut = 0; }
```

### DisableDiffusion()

```
void Garfield::AvalancheMC::DisableDiffusion ()    [inline]
```

Switch off diffusion.

Definition at line 159 of file AvalancheMC.hh.

```
00159 { m_useDiffusion = false; }
```

### DisablePlotting()

void Garfield::AvalancheMC::DisablePlotting () [inline]

Switch off drift line plotting.

Definition at line 123 of file AvalancheMC.hh.

```
00123 { m_viewer = nullptr; }
```

### DriftElectron()

```
bool Garfield::AvalancheMC::DriftElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Simulate the drift line of an electron from a given starting point.

### DriftHole()

```
bool Garfield::AvalancheMC::DriftHole (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Simulate the drift line of a hole from a given starting point.

### DriftIon()

```
bool Garfield::AvalancheMC::DriftIon (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Simulate the drift line of an ion from a given starting point.

### DriftLine()

```
int Garfield::AvalancheMC::DriftLine (
                const Seed & seed,
                std::vector< Point > & path,
                std::vector< Seed > & secondaries,
                const bool aval,
                const bool signal) const   [private]
```

Compute a single drift line.

### DriftNegativeIon()

```
bool Garfield::AvalancheMC::DriftNegativeIon (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w = 1)
```

Simulate the drift line of a negative ion from a given starting point.

### EnableAttachment()

```
void Garfield::AvalancheMC::EnableAttachment ()  [inline]
```

Switch on attachment (default: enabled).

Definition at line 162 of file AvalancheMC.hh.

```
00162 { m_useAttachment = true; }
```

### EnableAttachmentMap()

```
void Garfield::AvalancheMC::EnableAttachmentMap (
                const bool on = true)  [inline]
```

Retrieve the attachment coefficient from the component.

Definition at line 171 of file AvalancheMC.hh.

```
00171 { m_useAttachmentMap = on; }
```

### EnableAvalancheSizeLimit()

```
void Garfield::AvalancheMC::EnableAvalancheSizeLimit (
                const unsigned int size)  [inline]
```

Set a maximum avalanche size (ignore further multiplication once this size has been reached).

Definition at line 179 of file AvalancheMC.hh.

```
00179 { m_sizeCut = size; }
```

### EnableDebugging()

```
void Garfield::AvalancheMC::EnableDebugging (
                const bool on = true)  [inline]
```

Switch debugging messages on/off (default: off).

Definition at line 217 of file AvalancheMC.hh.

```
00217 { m_debug = on; }
```

### EnableDiffusion()

void Garfield::AvalancheMC::EnableDiffusion () [inline]

Switch on diffusion (default: enabled).

Definition at line 157 of file AvalancheMC.hh.

```
00157 { m_useDiffusion = true; }
```

### EnableDriftLines()

void Garfield::AvalancheMC::EnableDriftLines (
const bool on = true) [inline]

Switch on storage of drift lines (default: off).

Definition at line 126 of file AvalancheMC.hh.

```
00126 { m_storeDriftLines = on; }
```

### EnableInducedChargeCalculation()

void Garfield::AvalancheMC::EnableInducedChargeCalculation (
const bool on = true) [inline]

Switch on calculation of induced charge (default: disabled).

Definition at line 142 of file AvalancheMC.hh.

```
00142                                                 {
00143     m_doInducedCharge = on;
00144   }
```

### EnableMobilityMap()

void Garfield::AvalancheMC::EnableMobilityMap (
const bool on = true) [inline]

Retrieve the (low-field) mobility from the component.

Definition at line 173 of file AvalancheMC.hh.

```
00173 { m_useMobilityMap = on; }
```

### EnableMultiplication()

void Garfield::AvalancheMC::EnableMultiplication (
const bool on) [inline]

Switch multiplication on/off.

Definition at line 166 of file AvalancheMC.hh.

```
00166 { m_useMultiplication = on; }
```

### EnablePlotting()

```
void Garfield::AvalancheMC::EnablePlotting (
            ViewDrift * view)
```

Switch on drift line plotting.

### EnableProjectedPathIntegration()

```
void Garfield::AvalancheMC::EnableProjectedPathIntegration (
            const bool on = true)   [inline]
```

Switch on equilibration of multiplication and attachment over the drift line (default: enabled).

Definition at line 152 of file AvalancheMC.hh.

```
00152                                                {
00153     m_doEquilibration = on;
00154   }
```

### EnableRKFSteps()

```
void Garfield::AvalancheMC::EnableRKFSteps (
            const bool on = true)   [inline]
```

Switch on Runge-Kutta-Fehlberg stepping (as opposed to simple straight-line steps.

Definition at line 148 of file AvalancheMC.hh.

```
00148 { m_doRKF = on; }
```

### EnableSignalCalculation()

```
void Garfield::AvalancheMC::EnableSignalCalculation (
            const bool on = true)   [inline]
```

Switch calculation of induced currents on or off (default: enabled).

Definition at line 129 of file AvalancheMC.hh.

```
00129 { m_doSignal = on; }
```

### EnableTownsendMap()

```
void Garfield::AvalancheMC::EnableTownsendMap (
            const bool on = true)   [inline]
```

Retrieve the Townsend coefficient from the component.

Definition at line 169 of file AvalancheMC.hh.

```
00169 { m_useTownsendMap = on; }
```

### EnableVelocityMap()

```
void Garfield::AvalancheMC::EnableVelocityMap (
                const bool on = true)   [inline]
```

Retrieve the drift velocity from the component.

Definition at line 175 of file AvalancheMC.hh.

```
00175 { m_useVelocityMap = on; }
```

### Equilibrate()

```
bool Garfield::AvalancheMC::Equilibrate (
                std::vector< double > & alphas) const   [private]
```

### GetAttachment()

```
double Garfield::AvalancheMC::GetAttachment (
                const Particle particle,
                Medium * medium,
                const std::array< double, 3 > & x,
                const std::array< double, 3 > & e,
                const std::array< double, 3 > & b) const   [private]
```

Compute the attachment coefficient.

### GetAvalancheSize() [1/2]

```
std::pair< unsigned int, unsigned int > Garfield::AvalancheMC::GetAvalancheSize () const   [inline]
```

Return the number of electrons and ions/holes in the avalanche.

Definition at line 213 of file AvalancheMC.hh.

```
00213                                                 {
00214     return std::make_pair(m_nElectrons, std::max(m_nIons, m_nHoles));
00215   }
```

### GetAvalancheSize() [2/2]

```
void Garfield::AvalancheMC::GetAvalancheSize (
                unsigned int & ne,
                unsigned int & ni) const   [inline]
```

Return the number of electrons and ions/holes in the avalanche.

Definition at line 208 of file AvalancheMC.hh.

```
00208                                                 {
00209     ne = m_nElectrons;
00210     ni = std::max(m_nIons, m_nHoles);
00211   }
```

### GetAvalancheSizeLimit()

int Garfield::AvalancheMC::GetAvalancheSizeLimit () const [inline]

Return the currently set avalanche size limit.

Definition at line 183 of file AvalancheMC.hh.

```
00183 { return m_sizeCut; }
```

### GetDiffusion()

bool Garfield::AvalancheMC::GetDiffusion (
const Particle particle,
Medium * medium,
const std::array< double, 3 > & e,
const std::array< double, 3 > & b,
double & dl,
double & dt) const [private]

Compute the diffusion coefficients.

### GetElectronEndpoint()

void Garfield::AvalancheMC::GetElectronEndpoint (
const size_t i,
double & x0,
double & y0,
double & z0,
double & t0,
double & x1,
double & y1,
double & z1,
double & t1,
int & status) const

Return the coordinates and time of start and end point of a given electron drift line.

Parameters

| | |
|---|---|
| i | index of the drift line |
| x0,y0,z0,t0 | coordinates and time of the starting point |
| x1,y1,z1,t1 | coordinates and time of the end point |
| status | status code (see GarfieldConstants.hh) |

### GetElectrons()

const std::vector< EndPoint > & Garfield::AvalancheMC::GetElectrons () const [inline]

Definition at line 90 of file AvalancheMC.hh.

```
00090 { return m_electrons; }
```

### GetField()

```
int Garfield::AvalancheMC::GetField (
                const std::array< double, 3 > & x,
                std::array< double, 3 > & e,
                std::array< double, 3 > & b,
                Medium *& medium) const    [private]
```

Compute electric and magnetic field at a given position.

### GetHoles()

```
const std::vector< EndPoint > & Garfield::AvalancheMC::GetHoles () const    [inline]
```

Definition at line 91 of file AvalancheMC.hh.

```
00091 { return m_holes; }
```

### GetIonEndpoint()

```
void Garfield::AvalancheMC::GetIonEndpoint (
                const size_t i,
                double & x0,
                double & y0,
                double & z0,
                double & t0,
                double & x1,
                double & y1,
                double & z1,
                double & t1,
                int & status) const
```

### GetIons()

```
const std::vector< EndPoint > & Garfield::AvalancheMC::GetIons () const    [inline]
```

Definition at line 92 of file AvalancheMC.hh.

```
00092 { return m_ions; }
```

### GetMobility()

```
double Garfield::AvalancheMC::GetMobility (
                const Particle particle,
                Medium * medium,
                const std::array< double, 3 > & x) const    [private]
```

Retrieve the low-field mobility.

### GetNegativeIonEndpoint()

```
void Garfield::AvalancheMC::GetNegativeIonEndpoint (
                const size_t i,
                double & x0,
                double & y0,
                double & z0,
                double & t0,
                double & x1,
                double & y1,
                double & z1,
                double & t1,
                int & status) const
```

### GetNegativeIons()

```
const std::vector< EndPoint > & Garfield::AvalancheMC::GetNegativeIons () const   [inline]
```

Definition at line 93 of file AvalancheMC.hh.

```
00093                                          {
00094     return m_negativeIons;
00095   }
```

### GetNumberOfElectronEndpoints()

```
size_t Garfield::AvalancheMC::GetNumberOfElectronEndpoints () const   [inline]
```

Return the number of electron trajectories in the last simulated avalanche (including captured electrons).

Definition at line 99 of file AvalancheMC.hh.

```
00099 { return m_electrons.size(); }
```

### GetNumberOfIonEndpoints()

```
size_t Garfield::AvalancheMC::GetNumberOfIonEndpoints () const   [inline]
```

Return the number of ion trajectories.

Definition at line 101 of file AvalancheMC.hh.

```
00101 { return m_ions.size(); }
```

### GetTownsend()

```
double Garfield::AvalancheMC::GetTownsend (
                const Particle particle,
                Medium * medium,
                const std::array< double, 3 > & x,
                const std::array< double, 3 > & e,
                const std::array< double, 3 > & b) const   [private]
```

Compute the Townsend coefficient.

GetVelocity()

```
bool Garfield::AvalancheMC::GetVelocity (
                const Particle particle,
                Medium * medium,
                const std::array< double, 3 > & x,
                const std::array< double, 3 > & e,
                const std::array< double, 3 > & b,
                std::array< double, 3 > & v) const    [private]
```

Compute the drift velocity.

PrintError()

```
void Garfield::AvalancheMC::PrintError (
                const std::string & fcn,
                const std::string & par,
                const Particle particle,
                const std::array< double, 3 > & x) const    [private]
```

ResumeAvalanche()

```
bool Garfield::AvalancheMC::ResumeAvalanche (
                const bool electron = true,
                const bool hole = true)
```

Resume the simulation from the current set of charge carriers.

SetCollisionSteps()

```
void Garfield::AvalancheMC::SetCollisionSteps (
                const unsigned int n = 100)
```

Use exponentially distributed time steps with mean equal to the specified multiple of the collision time (default model).

SetDistanceSteps()

```
void Garfield::AvalancheMC::SetDistanceSteps (
                const double d = 0.001)
```

Use fixed distance steps (default 10 um).

SetElectronSignalScalingFactor()

```
void Garfield::AvalancheMC::SetElectronSignalScalingFactor (
                const double scale)    [inline]
```

Set multiplication factor for the signal induced by electrons.

Definition at line 201 of file AvalancheMC.hh.

```
00201 { m_scaleE = scale; }
```

SetHoleSignalScalingFactor()

void Garfield::AvalancheMC::SetHoleSignalScalingFactor (
const double scale) [inline]

Set multiplication factor for the signal induced by holes.

Definition at line 203 of file AvalancheMC.hh.
00203 { m_scaleH = scale; }

SetIonSignalScalingFactor()

void Garfield::AvalancheMC::SetIonSignalScalingFactor (
const double scale) [inline]

Set multiplication factor for the signal induced by ions.

Definition at line 205 of file AvalancheMC.hh.
00205 { m_scaleI = scale; }

SetSensor()

void Garfield::AvalancheMC::SetSensor (
Sensor * s)

Set the sensor.

SetSignalAveragingOrder()

void Garfield::AvalancheMC::SetSignalAveragingOrder (
const unsigned int navg) [inline]

Set the number of points to be used when averaging the signal vector over a time bin in the Sensor class.

The averaging is done with a $2 \times navg + 1$ point Newton-Raphson integration. Default: 1.

Definition at line 134 of file AvalancheMC.hh.
00134 { m_navg = navg; }

SetStepDistanceFunction()

void Garfield::AvalancheMC::SetStepDistanceFunction (
double(* f )(double x, double y, double z))

Retrieve the step distance from a user-supplied function.

SetTimeSteps()

void Garfield::AvalancheMC::SetTimeSteps (
const double d = 0.02)

Use fixed-time steps (default 20 ps).

SetTimeWindow()

```
void Garfield::AvalancheMC::SetTimeWindow (
                const double t0,
                const double t1)
```

Define a time interval (only carriers inside the interval are drifted).

StepRKF()

```
void Garfield::AvalancheMC::StepRKF (
                const Particle particle,
                const std::array< double, 3 > & x0,
                const std::array< double, 3 > & v0,
                const double dt,
                std::array< double, 3 > & xf,
                std::array< double, 3 > & vf,
                int & status) const    [private]
```

Compute end point and effective velocity for a step.

Terminate()

```
void Garfield::AvalancheMC::Terminate (
                const std::array< double, 3 > & x0,
                const double t0,
                std::array< double, 3 > & x,
                double & t) const    [private]
```

Terminate a drift line close to the boundary.

TransportParticles()

```
bool Garfield::AvalancheMC::TransportParticles (
                std::vector< Seed > & stack,
                const bool withElectrons,
                const bool withHoles,
                const bool aval)    [private]
```

Compute an avalanche.

UnsetTimeWindow()

```
void Garfield::AvalancheMC::UnsetTimeWindow ()    [inline]
```

Do not limit the time interval within which carriers are drifted.

Definition at line 198 of file AvalancheMC.hh.

```
00198 { m_hasTimeWindow = false; }
```

UseWeightingPotential()

void Garfield::AvalancheMC::UseWeightingPotential (
const bool on = true) [inline]

Use the weighting potential (as opposed to the weighting field) for calculating the induced signal.

Definition at line 137 of file AvalancheMC.hh.

```
00137                                       {
00138     m_useWeightingPotential = on;
00139   }
```

### 7.3.5 Member Data Documentation

m_className

std::string Garfield::AvalancheMC::m_className = "AvalancheMC" [private]

Definition at line 220 of file AvalancheMC.hh.

m_debug

bool Garfield::AvalancheMC::m_debug = false [private]

Definition at line 300 of file AvalancheMC.hh.

m_dMc

double Garfield::AvalancheMC::m_dMc = 0.001 [private]

Fixed distance step.

Definition at line 236 of file AvalancheMC.hh.

m_doEquilibration

bool Garfield::AvalancheMC::m_doEquilibration = true [private]

Definition at line 279 of file AvalancheMC.hh.

m_doInducedCharge

bool Garfield::AvalancheMC::m_doInducedCharge = false [private]

Definition at line 278 of file AvalancheMC.hh.

### m_doRKF

bool Garfield::AvalancheMC::m_doRKF = false [private]

Definition at line 280 of file AvalancheMC.hh.

### m_doSignal

bool Garfield::AvalancheMC::m_doSignal = true [private]

Definition at line 275 of file AvalancheMC.hh.

### m_electrons

std::vector<EndPoint> Garfield::AvalancheMC::m_electrons [private]

Start/end points of all electrons in the avalanche (including captured ones).

Definition at line 263 of file AvalancheMC.hh.

### m_fStep

double(* Garfield::AvalancheMC::m_fStep) (double x, double y, double z) = nullptr [private]

User function returning the step size.

Definition at line 240 of file AvalancheMC.hh.

### m_hasTimeWindow

bool Garfield::AvalancheMC::m_hasTimeWindow = false [private]

Flag whether a time window should be used.

Definition at line 243 of file AvalancheMC.hh.

### m_holes

std::vector<EndPoint> Garfield::AvalancheMC::m_holes [private]

Start/end points of all holes in the avalanche (including captured ones).

Definition at line 266 of file AvalancheMC.hh.

### m_ions

std::vector<EndPoint> Garfield::AvalancheMC::m_ions [private]

Start/end points of all positive ions in the avalanche.

Definition at line 268 of file AvalancheMC.hh.

### m_navg

unsigned int Garfield::AvalancheMC::m_navg = 1 [private]

Definition at line 276 of file AvalancheMC.hh.

### m_negativeIons

std::vector<EndPoint> Garfield::AvalancheMC::m_negativeIons [private]

Start/end points of all negative ions in the avalanche.

Definition at line 270 of file AvalancheMC.hh.

### m_nElectrons

unsigned int Garfield::AvalancheMC::m_nElectrons = 0 [private]

Number of electrons produced.

Definition at line 253 of file AvalancheMC.hh.

### m_nHoles

unsigned int Garfield::AvalancheMC::m_nHoles = 0 [private]

Number of holes produced.

Definition at line 255 of file AvalancheMC.hh.

### m_nIons

unsigned int Garfield::AvalancheMC::m_nIons = 0 [private]

Number of ions produced.

Definition at line 257 of file AvalancheMC.hh.

### m_nMc

int Garfield::AvalancheMC::m_nMc = 100 [private]

Sample step size according to collision time.

Definition at line 238 of file AvalancheMC.hh.

### m_nNegativeIons

unsigned int Garfield::AvalancheMC::m_nNegativeIons = 0 [private]

Number of negative ions produced.

Definition at line 259 of file AvalancheMC.hh.

### m_scaleE

double Garfield::AvalancheMC::m_scaleE = 1. [private]

Scaling factor for electron signals.

Definition at line 285 of file AvalancheMC.hh.

### m_scaleH

double Garfield::AvalancheMC::m_scaleH = 1. [private]

Scaling factor for hole signals.

Definition at line 287 of file AvalancheMC.hh.

### m_scaleI

double Garfield::AvalancheMC::m_scaleI = 1. [private]

Scaling factor for ion signals.

Definition at line 289 of file AvalancheMC.hh.

### m_sensor

Sensor* Garfield::AvalancheMC::m_sensor = nullptr [private]

Definition at line 222 of file AvalancheMC.hh.

### m_sizeCut

unsigned int Garfield::AvalancheMC::m_sizeCut = 0 [private]

Max. avalanche size.

Definition at line 250 of file AvalancheMC.hh.

### m_stepModel

StepModel Garfield::AvalancheMC::m_stepModel = StepModel::CollisionTime [private]

Step size model.

Definition at line 231 of file AvalancheMC.hh.

### m_storeDriftLines

bool Garfield::AvalancheMC::m_storeDriftLines = false [private]

Definition at line 274 of file AvalancheMC.hh.

### m_tMax

double Garfield::AvalancheMC::m_tMax = 0. [private]

Upper limit of the time window.

Definition at line 247 of file AvalancheMC.hh.

### m_tMc

double Garfield::AvalancheMC::m_tMc = 0.02 [private]

Fixed time step.

Definition at line 234 of file AvalancheMC.hh.

### m_tMin

double Garfield::AvalancheMC::m_tMin = 0. [private]

Lower limit of the time window.

Definition at line 245 of file AvalancheMC.hh.

### m_useAttachment

bool Garfield::AvalancheMC::m_useAttachment = true [private]

Definition at line 282 of file AvalancheMC.hh.

### m_useAttachmentMap

bool Garfield::AvalancheMC::m_useAttachmentMap = false [private]

Take attachment coefficients from the component.

Definition at line 294 of file AvalancheMC.hh.

### m_useDiffusion

bool Garfield::AvalancheMC::m_useDiffusion = true [private]

Definition at line 281 of file AvalancheMC.hh.

### m_useMobilityMap

bool Garfield::AvalancheMC::m_useMobilityMap = false [private]

Take mobility coefficients from the component.

Definition at line 296 of file AvalancheMC.hh.

### m_useMultiplication

bool Garfield::AvalancheMC::m_useMultiplication = true [private]

Definition at line 283 of file AvalancheMC.hh.

### m_useTownsendMap

bool Garfield::AvalancheMC::m_useTownsendMap = false [private]

Take Townsend coefficients from the component.

Definition at line 292 of file AvalancheMC.hh.

### m_useVelocityMap

bool Garfield::AvalancheMC::m_useVelocityMap = false [private]

Take the drift velocities from the component.

Definition at line 298 of file AvalancheMC.hh.

### m_useWeightingPotential

bool Garfield::AvalancheMC::m_useWeightingPotential = true [private]

Definition at line 277 of file AvalancheMC.hh.

### m_viewer

ViewDrift* Garfield::AvalancheMC::m_viewer = nullptr [private]

Definition at line 272 of file AvalancheMC.hh.

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh

## 7.4 Garfield::AvalancheMicroscopic Class Reference

Calculate electron drift lines and avalanches using microscopic tracking.

#include <AvalancheMicroscopic.hh>

Classes

- struct Electron
- struct Photon
- struct Point
- struct Seed
- struct Statistics

Public Member Functions

- AvalancheMicroscopic ()
  Default constructor.
- AvalancheMicroscopic (Sensor *sensor)
  Constructor.
- ~AvalancheMicroscopic ()
  Destructor.
- void SetSensor (Sensor *sensor)
  Set the sensor.
- void EnablePlotting (ViewDrift *view, const std::size_t nColl=100)
  Switch on drift line plotting.
- void DisablePlotting ()
  Switch off drift line plotting.
- void EnableExcitationMarkers (const bool on=true)
  Draw a marker at every excitation or not.
- void EnableIonisationMarkers (const bool on=true)
  Draw a marker at every ionising collision or not.
- void EnableAttachmentMarkers (const bool on=true)
  Draw a marker at every attachment or not.
- void EnableSignalCalculation (const bool on=true)
  Switch calculation of induced currents on or off (default: enabled).
- void UseWeightingPotential (const bool on=true)
  Use the weighting potential (as opposed to the weighting field) for calculating the induced current.
- void EnableWeightingFieldIntegration (const bool on=true)
  Integrate the weighting field over a drift line step when calculating the induced current (default: off).
- void UseInducedCharge (const bool on=true)
  Switch on calculation of the total induced charge (default: off).
- void EnablePathLengthComputation (const bool on=true)
  Compute and store the path length of each trajectory (default: off).
- void EnableElectronEnergyHistogramming (TH1 *histo)
  Fill a histogram with the electron energy distribution.
- void DisableElectronEnergyHistogramming ()
  Stop histogramming the electron energy distribution.
- void EnableHoleEnergyHistogramming (TH1 *histo)
  Fill a histogram with the hole energy distribution.

- void DisableHoleEnergyHistogramming ()

  Stop histogramming the hole energy distribution.
- void SetDistanceHistogram (TH1 *histo, const char opt='r')

  Fill histograms of the distance between successive collisions.
- void EnableDistanceHistogramming (const int type)

  Fill distance distribution histograms for a given collision type.
- void DisableDistanceHistogramming (const int type)

  Stop filling distance distribution histograms for a given collision type.
- void DisableDistanceHistogramming ()

  Stop filling distance distribution histograms.
- void EnableSecondaryEnergyHistogramming (TH1 *histo)

  Fill histograms of the energy of electrons emitted in ionising collisions.
- void DisableSecondaryEnergyHistogramming ()

  Stop histogramming the secondary electron energy distribution.
- void EnableDriftLines (const bool on=true)

  Switch on storage of drift lines (default: off).
- void EnablePhotonTransport (const bool on=true)

  Switch on photon transport.
- void EnableBandStructure (const bool on=true)

  Switch on stepping according to band structure E(k), for semiconductors.
- void EnableNullCollisionSteps (const bool on=true, const int nSteps=1)

  Switch on update of coordinates for null-collision steps (default: off).
- void EnableRKNSteps (const bool on=true)

  Switch on Runge-Kutta-Nystrom stepping (default: off).
- void SetRKNTolerance (const double sTol=1.e-10, const double sMinStep=1.e-5)

  Set error tolerance and minimum step size on Runge-Kutta-Nystrom method (default: 1.e-10 and 1.e-5).
- void SetElectronTransportCut (const double cut)

  Set a (lower) energy threshold for electron transport.
- double GetElectronTransportCut () const

  Retrieve the value of the energy threshold.
- void SetPhotonTransportCut (const double cut)

  Set an energy threshold for photon transport.
- double GetPhotonTransportCut () const

  Retrieve the energy threshold for transporting photons.
- void EnableAvalancheSizeLimit (const unsigned int size)

  Set a max.
- void DisableAvalancheSizeLimit ()

  Do not apply a limit on the avalanche size.
- int GetAvalancheSizeLimit () const

  Retrieve the currently set size limit.
- void EnableMagneticField (const bool on=true)

  Switch on/off using the magnetic field in the stepping algorithm.
- void SetCollisionSteps (const unsigned int n)

  Set number of collisions to be skipped for storing drift lines.
- void SetTimeWindow (const double t0, const double t1)

  Define a time interval (only carriers inside the interval are simulated).
- void UnsetTimeWindow ()

  Do not restrict the time interval within which carriers are simulated.
- void GetAvalancheSize (int &ne, int &ni) const

  Return the number of electrons and ions in the avalanche.
- std::pair< int, int > GetAvalancheSize () const

Return the number of electrons and ions in the avalanche.

- void GetAvalancheSize (int &ne, int &nh, int &ni) const
- void GetAvalancheSizeGPU (int &ne, int &ni) const

  Return the number of electrons and ions in the avalanche.
- const std::vector< Electron > & GetElectrons () const
- const std::vector< Electron > & GetHoles () const
- size_t GetNumberOfElectronEndpoints () const

  Return the number of electron trajectories in the last simulated avalanche (including captured electrons).
- unsigned int GetNumberOfElectronEndpointsGPU () const
- void GetElectronEndpoint (const size_t i, double &x0, double &y0, double &z0, double &t0, double &e0, double &x1, double &y1, double &z1, double &t1, double &e1, int &status) const

  Return the coordinates and time of start and end point of a given electron drift line.
- void GetElectronEndpointGPU (const size_t i, double &x0, double &y0, double &z0, double &t0, double &e0, double &x1, double &y1, double &z1, double &t1, double &e1, int &status) const
- size_t GetNumberOfElectronDriftLinePoints (const size_t i=0) const
- void GetElectronDriftLinePoint (double &x, double &y, double &z, double &t, const size_t ip, const size_t ie=0) const
- size_t GetNumberOfPhotons () const
- void GetPhoton (const size_t i, double &e, double &x0, double &y0, double &z0, double &t0, double &x1, double &y1, double &z1, double &t1, int &status) const
- bool DriftElectron (const double x, const double y, const double z, const double t, const double e, const double dx=0., const double dy=0., const double dz=0., const size_t w=1)

  Calculate an electron drift line.
- bool AvalancheElectron (const double x, const double y, const double z, const double t, const double e, const double dx=0., const double dy=0., const double dz=0., const size_t w=1)

  Calculate an avalanche initiated by a given electron.
- void AddElectron (const double x, const double y, const double z, const double t, const double e, const double dx=0., const double dy=0., const double dz=0., const size_t w=1)

  Add an electron to the list of particles to be transported.
- bool ResumeAvalanche ()

  Continue the avalanche simulation from the current set of electrons.
- void SetUserHandleStep (void(*f)(double x, double y, double z, double t, double e, double dx, double dy, double dz, bool hole))

  Set a callback function to be called at every step.
- void UnsetUserHandleStep ()

  Deactivate the user handle called at every step.
- void SetUserHandleCollision (void(*f)(double x, double y, double z, double t, int type, int level, Medium *m, double e0, double e1, double dx0, double dy0, double dz0, double dx1, double dy1, double dz1))

  Set a callback function to be called at every (real) collision.
- void UnsetUserHandleCollision ()

  Deactivate the user handle called at every collision.
- void SetUserHandleAttachment (void(*f)(double x, double y, double z, double t, int type, int level, Medium *m))

  Set a user handling procedure, to be called at every attachment.
- void UnsetUserHandleAttachment ()

  Deactivate the user handle called at every attachment.
- void SetUserHandleInelastic (void(*f)(double x, double y, double z, double t, int type, int level, Medium *m))

  Set a user handling procedure, to be called at every inelastic collision.
- void UnsetUserHandleInelastic ()

  Deactivate the user handle called at every inelastic collision.

- void SetUserHandleIonisation (void(*f)(double x, double y, double z, double t, int type, int level, Medium *m))

  Set a user handling procedure, to be called at every ionising collision or excitation followed by Penning transfer.
- void UnsetUserHandleIonisation ()

  Deactivate the user handle called at every ionisation.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()
- Statistics GetStatistics ()
- void SetRunModeOptions (MPRunMode mode, int device=-1)
- void SetMaxNumShowerLoops (int max_loops)
- void SetShowProgress (bool show_progress)
- void SetDebugShowerIterationAndElectronID (int iter_num, int elec_id)
- std::vector< Seed > GetStackOld ()
- std::vector< Electron > GetStackOldGPU ()

Private Member Functions

- bool TransportElectrons (std::vector< Seed > &stack, const bool aval)
- int TransportElectron (const Seed &seed, const bool signal, std::vector< double > &ts, std::vector< std::array< double, 3 > > &xs, std::vector< Point > &path, std::vector< Seed > &stack)
- int TransportElectronBfield (const Seed &seed, const bool signal, std::vector< double > &ts, std::vector< std::array< double, 3 > > &xs, std::vector< Point > &path, std::vector< Seed > &stack)
- int TransportElectronSc (const Seed &seed, const bool signal, std::vector< double > &ts, std::vector< std::array< double, 3 > > &xs, std::vector< Point > &path, std::vector< Seed > &stack)
- void TransportPhoton (const double x, const double y, const double z, const double t, const double e, const size_t w, std::vector< Seed > &stack)
- bool transportParticleStack (const bool aval, std::vector< Seed > &stack, std::vector< Seed > &newParticles, const bool signal, const bool useBfield, const bool sc)
- void Terminate (double x0, double y0, double z0, double t0, double &x1, double &y1, double &z1, double &t1) const
- void CreatePenningElectron (const double x, const double y, const double z, const double t, const size_t w, const double ds, const double dt, const double ep, const int level, std::vector< Seed > &stack) const
- void PlotCollision (const int cstype, const size_t did, const double x, const double y, const double z, size_t &nCollPlot) const
- void CallUserHandles (const int cstype, const double x, const double y, const double z, const double t, const int level, Medium *medium, const double en1, const double en, const double kx, const double ky, const double kz, const double kx1, const double ky1, const double kz1) const
- void FillDistanceHistogram (const int cstype, const double x, const double y, const double z, double &xLast, double &yLast, double &zLast) const

Private Attributes

- std::string m_className = "AvalancheMicroscopic"
- MPRunMode m_runMode {MPRunMode::Normal}
- int m_cudaDevice {-1}
- int m_maxNumShowerLoops {-1}
- int m_debugElectronID {-1}
- int m_debugShowerLoopNum {-1}
- bool m_showProgress {false}
- AvalancheMicroscopicGPU * m_gpuInterface {nullptr}
- Statistics m_stats

- Sensor * m_sensor = nullptr
- std::vector< Electron > m_electrons
- std::vector< Electron > m_electrons_gpu
- std::vector< Electron > m_holes
- std::vector< Seed > m_stackStoreCPU
- std::vector< Electron > m_stackStoreGPU
- std::vector< Photon > m_photons
- int m_nElectrons = 0

  Number of electrons produced.
- int m_nHoles = 0

  Number of holes produced.
- int m_nIons = 0

  Number of ions produced.
- int m_nElectronsGPU = 0

  Number of electrons produced.
- int m_nHolesGPU = 0

  Number of holes produced.
- int m_nIonsGPU = 0

  Number of ions produced.
- ViewDrift * m_viewer = nullptr
- bool m_plotExcitations = true
- bool m_plotIonisations = true
- bool m_plotAttachments = true
- TH1 * m_histElectronEnergy = nullptr
- TH1 * m_histHoleEnergy = nullptr
- TH1 * m_histDistance = nullptr
- char m_distanceOption = 'r'
- std::vector< int > m_distanceHistogramType
- TH1 * m_histSecondary = nullptr
- bool m_doSignal = true
- bool m_useWeightingPotential = true
- bool m_integrateWeightingField = false
- bool m_doInducedCharge = false
- bool m_computePathLength = false
- bool m_storeDriftLines = false
- bool m_usePhotons = false
- bool m_useBandStructure = true
- bool m_useNullCollisionSteps = false
- bool m_useBfieldAuto = true
- bool m_useBfield = false
- bool m_rknSteps = false
- double m_rknsteperrortol = 1.e-10
- double m_rknMinh = 1.e-5
- double m_nullCollScale = 1.
- double m_deltaCut = 0.
- double m_gammaCut = 0.
- unsigned int m_sizeCut = 0
- size_t m_nCollSkip = 100
- size_t m_nCollPlot = 100
- bool m_hasTimeWindow = false
- double m_tMin = 0.
- double m_tMax = 0.
- void(* m_userHandleStep )(double x, double y, double z, double t, double e, double dx, double dy, double dz, bool hole) = nullptr

- void(* m_userHandleCollision )(double x, double y, double z, double t, int type, int level, Medium *m, double e0, double e1, double dx0, double dy0, double dz0, double dx1, double dy1, double dz1) = nullptr
- void(* m_userHandleAttachment )(double x, double y, double z, double t, int type, int level, Medium *m) = nullptr
- void(* m_userHandleInelastic )(double x, double y, double z, double t, int type, int level, Medium *m) = nullptr
- void(* m_userHandleIonisation )(double x, double y, double z, double t, int type, int level, Medium *m) = nullptr
- bool m_debug = false

Friends

- class AvalancheMicroscopicGPU

### 7.4.1 Detailed Description

Calculate electron drift lines and avalanches using microscopic tracking.

Definition at line 24 of file AvalancheMicroscopic.hh.

### 7.4.2 Constructor & Destructor Documentation

AvalancheMicroscopic() [1/2]

Garfield::AvalancheMicroscopic::AvalancheMicroscopic () [inline]

Default constructor.

Definition at line 27 of file AvalancheMicroscopic.hh.
00027 : AvalancheMicroscopic(nullptr) {}

AvalancheMicroscopic() [2/2]

Garfield::AvalancheMicroscopic::AvalancheMicroscopic (
Sensor * sensor)

Constructor.

~AvalancheMicroscopic()

Garfield::AvalancheMicroscopic::~AvalancheMicroscopic () [inline]

Destructor.

Definition at line 31 of file AvalancheMicroscopic.hh.
00031 {}

### 7.4.3 Member Function Documentation

#### AddElectron()

```
void Garfield::AvalancheMicroscopic::AddElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const double e,
                const double dx = 0.,
                const double dy = 0.,
                const double dz = 0.,
                const size_t w = 1)
```

Add an electron to the list of particles to be transported.

#### AvalancheElectron()

```
bool Garfield::AvalancheMicroscopic::AvalancheElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const double e,
                const double dx = 0.,
                const double dy = 0.,
                const double dz = 0.,
                const size_t w = 1)
```

Calculate an avalanche initiated by a given electron.

#### CallUserHandles()

```
void Garfield::AvalancheMicroscopic::CallUserHandles (
                const int cstype,
                const double x,
                const double y,
                const double z,
                const double t,
                const int level,
                Medium * medium,
                const double en1,
                const double en,
                const double kx,
                const double ky,
                const double kz,
                const double kx1,
                const double ky1,
                const double kz1) const    [private]
```

### CreatePenningElectron()

```
void Garfield::AvalancheMicroscopic::CreatePenningElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const size_t w,
                const double ds,
                const double dt,
                const double ep,
                const int level,
                std::vector< Seed > & stack) const   [private]
```

### DisableAvalancheSizeLimit()

```
void Garfield::AvalancheMicroscopic::DisableAvalancheSizeLimit ()   [inline]
```

Do not apply a limit on the avalanche size.

Definition at line 135 of file AvalancheMicroscopic.hh.

```
00135 { m_sizeCut = 0; }
```

### DisableDebugging()

```
void Garfield::AvalancheMicroscopic::DisableDebugging ()   [inline]
```

Definition at line 287 of file AvalancheMicroscopic.hh.

```
00287 { m_debug = false; }
```

### DisableDistanceHistogramming() [1/2]

```
void Garfield::AvalancheMicroscopic::DisableDistanceHistogramming ()
```

Stop filling distance distribution histograms.

### DisableDistanceHistogramming() [2/2]

```
void Garfield::AvalancheMicroscopic::DisableDistanceHistogramming (
                const int type)
```

Stop filling distance distribution histograms for a given collision type.

### DisableElectronEnergyHistogramming()

```
void Garfield::AvalancheMicroscopic::DisableElectronEnergyHistogramming ()   [inline]
```

Stop histogramming the electron energy distribution.

Definition at line 70 of file AvalancheMicroscopic.hh.

```
00070 { m_histElectronEnergy = nullptr; }
```

### DisableHoleEnergyHistogramming()

void Garfield::AvalancheMicroscopic::DisableHoleEnergyHistogramming () [inline]

Stop histogramming the hole energy distribution.

Definition at line 74 of file AvalancheMicroscopic.hh.

```
00074 { m_histHoleEnergy = nullptr; }
```

### DisablePlotting()

void Garfield::AvalancheMicroscopic::DisablePlotting () [inline]

Switch off drift line plotting.

Definition at line 39 of file AvalancheMicroscopic.hh.

```
00039 { m_viewer = nullptr; }
```

### DisableSecondaryEnergyHistogramming()

void Garfield::AvalancheMicroscopic::DisableSecondaryEnergyHistogramming () [inline]

Stop histogramming the secondary electron energy distribution.

Definition at line 92 of file AvalancheMicroscopic.hh.

```
00092 { m_histSecondary = nullptr; }
```

### DriftElectron()

```
bool Garfield::AvalancheMicroscopic::DriftElectron (
                const double x,
                const double y,
                const double z,
                const double t,
                const double e,
                const double dx = 0.,
                const double dy = 0.,
                const double dz = 0.,
                const size_t w = 1)
```

Calculate an electron drift line.

Parameters

| | |
|---|---|
| x,y,z,t | starting point of the electron |
| e | initial energy of the electron |
| dx,dy,dz | initial direction vector of the electron If the initial direction is not specified, it is sampled randomly. |
| w | weight (multiplicity) of the electron Secondary electrons are not transported. |

EnableAttachmentMarkers()

void Garfield::AvalancheMicroscopic::EnableAttachmentMarkers (
const bool on = true) [inline]

Draw a marker at every attachment or not.

Definition at line 45 of file AvalancheMicroscopic.hh.
00045 { m_plotAttachments = on; }

EnableAvalancheSizeLimit()

void Garfield::AvalancheMicroscopic::EnableAvalancheSizeLimit (
const unsigned int size) [inline]

Set a max.

avalanche size (i. e. ignore ionising collisions once this size has been reached).

Definition at line 133 of file AvalancheMicroscopic.hh.
00133 { m_sizeCut = size; }

EnableBandStructure()

void Garfield::AvalancheMicroscopic::EnableBandStructure (
const bool on = true) [inline]

Switch on stepping according to band structure E(k), for semiconductors.

Definition at line 102 of file AvalancheMicroscopic.hh.
00102 { m_useBandStructure = on; }

EnableDebugging()

void Garfield::AvalancheMicroscopic::EnableDebugging () [inline]

Switch on debugging messages.

Definition at line 286 of file AvalancheMicroscopic.hh.
00286 { m_debug = true; }

EnableDistanceHistogramming()

void Garfield::AvalancheMicroscopic::EnableDistanceHistogramming (
const int type)

Fill distance distribution histograms for a given collision type.

### EnableDriftLines()

```
void Garfield::AvalancheMicroscopic::EnableDriftLines (
            const bool on = true)   [inline]
```

Switch on storage of drift lines (default: off).

Definition at line 95 of file AvalancheMicroscopic.hh.

```
00095 { m_storeDriftLines = on; }
```

### EnableElectronEnergyHistogramming()

```
void Garfield::AvalancheMicroscopic::EnableElectronEnergyHistogramming (
            TH1 * histo)
```

Fill a histogram with the electron energy distribution.

### EnableExcitationMarkers()

```
void Garfield::AvalancheMicroscopic::EnableExcitationMarkers (
            const bool on = true)   [inline]
```

Draw a marker at every excitation or not.

Definition at line 41 of file AvalancheMicroscopic.hh.

```
00041 { m_plotExcitations = on; }
```

### EnableHoleEnergyHistogramming()

```
void Garfield::AvalancheMicroscopic::EnableHoleEnergyHistogramming (
            TH1 * histo)
```

Fill a histogram with the hole energy distribution.

### EnableIonisationMarkers()

```
void Garfield::AvalancheMicroscopic::EnableIonisationMarkers (
            const bool on = true)   [inline]
```

Draw a marker at every ionising collision or not.

Definition at line 43 of file AvalancheMicroscopic.hh.

```
00043 { m_plotIonisations = on; }
```

### EnableMagneticField()

```
void Garfield::AvalancheMicroscopic::EnableMagneticField (
            const bool on = true)   [inline]
```

Switch on/off using the magnetic field in the stepping algorithm.

Definition at line 140 of file AvalancheMicroscopic.hh.

```
00140                                       {
00141     m_useBfieldAuto = false;
00142     m_useBfield = on;
00143   }
```

EnableNullCollisionSteps()

```
void Garfield::AvalancheMicroscopic::EnableNullCollisionSteps (
                const bool on = true,
                const int nSteps = 1)   [inline]
```

Switch on update of coordinates for null-collision steps (default: off).

Definition at line 105 of file AvalancheMicroscopic.hh.

```
00105                                                   {
00106     m_useNullCollisionSteps = on;
00107     m_nullCollScale = nSteps <= 1 ? 1. : 1. / nSteps;
00108   }
```

EnablePathLengthComputation()

```
void Garfield::AvalancheMicroscopic::EnablePathLengthComputation (
                const bool on = true)   [inline]
```

Compute and store the path length of each trajectory (default: off).

Definition at line 64 of file AvalancheMicroscopic.hh.

```
00064                                           {
00065     m_computePathLength = on;
00066   }
```

EnablePhotonTransport()

```
void Garfield::AvalancheMicroscopic::EnablePhotonTransport (
                const bool on = true)   [inline]
```

Switch on photon transport.

Remarks

This feature has not been tested thoroughly.

Definition at line 99 of file AvalancheMicroscopic.hh.

```
00099 { m_usePhotons = on; }
```

EnablePlotting()

```
void Garfield::AvalancheMicroscopic::EnablePlotting (
                ViewDrift * view,
                const std::size_t nColl = 100)
```

Switch on drift line plotting.

EnableRKNSteps()

```
void Garfield::AvalancheMicroscopic::EnableRKNSteps (
                const bool on = true)   [inline]
```

Switch on Runge-Kutta-Nystrom stepping (default: off).

Definition at line 111 of file AvalancheMicroscopic.hh.

```
00111 { m_rknSteps = on; }
```

### EnableSecondaryEnergyHistogramming()

```
void Garfield::AvalancheMicroscopic::EnableSecondaryEnergyHistogramming (
            TH1 * histo)
```

Fill histograms of the energy of electrons emitted in ionising collisions.

### EnableSignalCalculation()

```
void Garfield::AvalancheMicroscopic::EnableSignalCalculation (
            const bool on = true)   [inline]
```

Switch calculation of induced currents on or off (default: enabled).

Definition at line 48 of file AvalancheMicroscopic.hh.

```
00048 { m_doSignal = on; }
```

### EnableWeightingFieldIntegration()

```
void Garfield::AvalancheMicroscopic::EnableWeightingFieldIntegration (
            const bool on = true)   [inline]
```

Integrate the weighting field over a drift line step when calculating the induced current (default: off).

Definition at line 56 of file AvalancheMicroscopic.hh.

```
00056                                                {
00057     m_integrateWeightingField = on;
00058   }
```

### FillDistanceHistogram()

```
void Garfield::AvalancheMicroscopic::FillDistanceHistogram (
            const int cstype,
            const double x,
            const double y,
            const double z,
            double & xLast,
            double & yLast,
            double & zLast) const   [private]
```

### GetAvalancheSize() [1/3]

```
std::pair< int, int > Garfield::AvalancheMicroscopic::GetAvalancheSize () const   [inline]
```

Return the number of electrons and ions in the avalanche.

Definition at line 159 of file AvalancheMicroscopic.hh.

```
00159                                          {
00160     return std::make_pair(m_nElectrons, m_nIons);
00161   }
```

GetAvalancheSize() [2/3]

```
void Garfield::AvalancheMicroscopic::GetAvalancheSize (
                int & ne,
                int & nh,
                int & ni) const    [inline]
```

Definition at line 162 of file AvalancheMicroscopic.hh.

```
00162                                                {
00163     ne = m_nElectrons;
00164     nh = m_nHoles;
00165     ni = m_nIons;
00166   }
```

GetAvalancheSize() [3/3]

```
void Garfield::AvalancheMicroscopic::GetAvalancheSize (
                int & ne,
                int & ni) const    [inline]
```

Return the number of electrons and ions in the avalanche.

Definition at line 154 of file AvalancheMicroscopic.hh.

```
00154                                          {
00155     ne = m_nElectrons;
00156     ni = m_nIons;
00157   }
```

GetAvalancheSizeGPU()

```
void Garfield::AvalancheMicroscopic::GetAvalancheSizeGPU (
                int & ne,
                int & ni) const    [inline]
```

Return the number of electrons and ions in the avalanche.

Definition at line 168 of file AvalancheMicroscopic.hh.

```
00168                                             {
00169     ne = m_nElectronsGPU;
00170     ni = m_nIonsGPU;
00171   }
```

GetAvalancheSizeLimit()

```
int Garfield::AvalancheMicroscopic::GetAvalancheSizeLimit () const    [inline]
```

Retrieve the currently set size limit.

Definition at line 137 of file AvalancheMicroscopic.hh.

```
00137 { return m_sizeCut; }
```

### GetElectronDriftLinePoint()

```
void Garfield::AvalancheMicroscopic::GetElectronDriftLinePoint (
                double & x,
                double & y,
                double & z,
                double & t,
                const size_t ip,
                const size_t ie = 0) const
```

### GetElectronEndpoint()

```
void Garfield::AvalancheMicroscopic::GetElectronEndpoint (
                const size_t i,
                double & x0,
                double & y0,
                double & z0,
                double & t0,
                double & e0,
                double & x1,
                double & y1,
                double & z1,
                double & t1,
                double & e1,
                int & status) const
```

Return the coordinates and time of start and end point of a given electron drift line.

Parameters

| | |
|---|---|
| i | index of the drift line |
| x0,y0,z0,t0 | coordinates and time of the starting point |
| x1,y1,z1,t1 | coordinates and time of the end point |
| e0,e1 | initial and final energy |
| status | status code (see GarfieldConstants.hh) |

### GetElectronEndpointGPU()

```
void Garfield::AvalancheMicroscopic::GetElectronEndpointGPU (
                const size_t i,
                double & x0,
                double & y0,
                double & z0,
                double & t0,
                double & e0,
                double & x1,
                double & y1,
                double & z1,
                double & t1,
                double & e1,
                int & status) const
```

### GetElectrons()

const std::vector< Electron > & Garfield::AvalancheMicroscopic::GetElectrons () const [inline]

Definition at line 194 of file AvalancheMicroscopic.hh.

```
00194 { return m_electrons; }
```

### GetElectronTransportCut()

double Garfield::AvalancheMicroscopic::GetElectronTransportCut () const [inline]

Retrieve the value of the energy threshold.

Definition at line 124 of file AvalancheMicroscopic.hh.

```
00124 { return m_deltaCut; }
```

### GetHoles()

const std::vector< Electron > & Garfield::AvalancheMicroscopic::GetHoles () const [inline]

Definition at line 195 of file AvalancheMicroscopic.hh.

```
00195 { return m_holes; }
```

### GetNumberOfElectronDriftLinePoints()

size_t Garfield::AvalancheMicroscopic::GetNumberOfElectronDriftLinePoints (
const size_t i = 0) const

### GetNumberOfElectronEndpoints()

size_t Garfield::AvalancheMicroscopic::GetNumberOfElectronEndpoints () const [inline]

Return the number of electron trajectories in the last simulated avalanche (including captured electrons).

Definition at line 198 of file AvalancheMicroscopic.hh.

```
00198 { return m_electrons.size(); }
```

### GetNumberOfElectronEndpointsGPU()

unsigned int Garfield::AvalancheMicroscopic::GetNumberOfElectronEndpointsGPU () const [inline]

Definition at line 199 of file AvalancheMicroscopic.hh.

```
00199                                          {
00200     return m_electrons_gpu.size();
00201   }
```

### GetNumberOfPhotons()

size_t Garfield::AvalancheMicroscopic::GetNumberOfPhotons () const [inline]

Definition at line 222 of file AvalancheMicroscopic.hh.

```
00222 { return m_photons.size(); }
```

### GetPhoton()

void Garfield::AvalancheMicroscopic::GetPhoton (
const size_t i,
double & e,
double & x0,
double & y0,
double & z0,
double & t0,
double & x1,
double & y1,
double & z1,
double & t1,
int & status) const

### GetPhotonTransportCut()

double Garfield::AvalancheMicroscopic::GetPhotonTransportCut () const [inline]

Retrieve the energy threshold for transporting photons.

Definition at line 129 of file AvalancheMicroscopic.hh.

```
00129 { return m_gammaCut; }
```

### GetStackOld()

std::vector< Seed > Garfield::AvalancheMicroscopic::GetStackOld () [inline]

Definition at line 458 of file AvalancheMicroscopic.hh.

```
00458                                {
00459     return m_stackStoreCPU;
00460   }
```

### GetStackOldGPU()

std::vector< Electron > Garfield::AvalancheMicroscopic::GetStackOldGPU () [inline]

Definition at line 461 of file AvalancheMicroscopic.hh.

```
00461 { return m_stackStoreGPU; }
```

### GetStatistics()

Statistics Garfield::AvalancheMicroscopic::GetStatistics () [inline]

Definition at line 300 of file AvalancheMicroscopic.hh.

```
00300 { return m_stats; }
```

### PlotCollision()

```
void Garfield::AvalancheMicroscopic::PlotCollision (
                const int cstype,
                const size_t did,
                const double x,
                const double y,
                const double z,
                size_t & nCollPlot) const    [private]
```

### ResumeAvalanche()

```
bool Garfield::AvalancheMicroscopic::ResumeAvalanche ()
```

Continue the avalanche simulation from the current set of electrons.

### SetCollisionSteps()

```
void Garfield::AvalancheMicroscopic::SetCollisionSteps (
                const unsigned int n)    [inline]
```

Set number of collisions to be skipped for storing drift lines.

Definition at line 146 of file AvalancheMicroscopic.hh.

```
00146 { m_nCollSkip = n; }
```

### SetDebugShowerIterationAndElectronID()

```
void Garfield::AvalancheMicroscopic::SetDebugShowerIterationAndElectronID (
                int iter_num,
                int elec_id)    [inline]
```

Definition at line 305 of file AvalancheMicroscopic.hh.

```
00305                                                  {
00306     m_debugShowerLoopNum = iter_num;
00307     m_debugElectronID = elec_id;
00308   }
```

### SetDistanceHistogram()

```
void Garfield::AvalancheMicroscopic::SetDistanceHistogram (
                TH1 * histo,
                const char opt = 'r')
```

Fill histograms of the distance between successive collisions.

Parameters

| | |
|---|---|
| histo | pointer to the histogram to be filled |
| opt | direction ('x', 'y', 'z', 'r') |

SetElectronTransportCut()

```
void Garfield::AvalancheMicroscopic::SetElectronTransportCut (
                const double cut)   [inline]
```

Set a (lower) energy threshold for electron transport.

This can be useful for simulating delta electrons.

Definition at line 122 of file AvalancheMicroscopic.hh.

```
00122 { m_deltaCut = cut; }
```

SetMaxNumShowerLoops()

```
void Garfield::AvalancheMicroscopic::SetMaxNumShowerLoops (
                int max_loops)   [inline]
```

Definition at line 303 of file AvalancheMicroscopic.hh.

```
00303 { m_maxNumShowerLoops = max_loops; }
```

SetPhotonTransportCut()

```
void Garfield::AvalancheMicroscopic::SetPhotonTransportCut (
                const double cut)   [inline]
```

Set an energy threshold for photon transport.

Definition at line 127 of file AvalancheMicroscopic.hh.

```
00127 { m_gammaCut = cut; }
```

SetRKNTolerance()

```
void Garfield::AvalancheMicroscopic::SetRKNTolerance (
                const double sTol = 1.e-10,
                const double sMinStep = 1.e-5)   [inline]
```

Set error tolerance and minimum step size on Runge-Kutta-Nystrom method (default: 1.e-10 and 1.e-5).

Definition at line 114 of file AvalancheMicroscopic.hh.

```
00115                                           {
00116     m_rknsteperrortol = sTol;
00117     m_rknMinh = sMinStep;
00118   }
```

SetRunModeOptions()

```
void Garfield::AvalancheMicroscopic::SetRunModeOptions (
                MPRunMode mode,
                int device = -1)
```

SetSensor()

void Garfield::AvalancheMicroscopic::SetSensor (
Sensor * sensor)

Set the sensor.

SetShowProgress()

void Garfield::AvalancheMicroscopic::SetShowProgress (
bool show_progress) [inline]

Definition at line 304 of file AvalancheMicroscopic.hh.
00304 { m_showProgress = show_progress; }

SetTimeWindow()

void Garfield::AvalancheMicroscopic::SetTimeWindow (
const double t0,
const double t1)

Define a time interval (only carriers inside the interval are simulated).

SetUserHandleAttachment()

void Garfield::AvalancheMicroscopic::SetUserHandleAttachment (
void(* f )(double x, double y, double z, double t, int type, int level, Medium *m))

Set a user handling procedure, to be called at every attachment.

SetUserHandleCollision()

void Garfield::AvalancheMicroscopic::SetUserHandleCollision (
void(* f )(double x, double y, double z, double t, int type, int level, Medium *m, double e0, double e1, double dx0, double dy0, double dz0, double dx1, double dy1, double dz1))

Set a callback function to be called at every (real) collision.

SetUserHandleInelastic()

void Garfield::AvalancheMicroscopic::SetUserHandleInelastic (
void(* f )(double x, double y, double z, double t, int type, int level, Medium *m))

Set a user handling procedure, to be called at every inelastic collision.

### SetUserHandleIonisation()

void Garfield::AvalancheMicroscopic::SetUserHandleIonisation (
void(* f )(double x, double y, double z, double t, int type, int level, Medium *m))

Set a user handling procedure, to be called at every ionising collision or excitation followed by Penning transfer.

### SetUserHandleStep()

void Garfield::AvalancheMicroscopic::SetUserHandleStep (
void(* f )(double x, double y, double z, double t, double e, double dx, double dy, double dz, bool hole))

Set a callback function to be called at every step.

### Terminate()

void Garfield::AvalancheMicroscopic::Terminate (
double x0,
double y0,
double z0,
double t0,
double & x1,
double & y1,
double & z1,
double & t1) const [private]

### TransportElectron()

int Garfield::AvalancheMicroscopic::TransportElectron (
const Seed & seed,
const bool signal,
std::vector< double > & ts,
std::vector< std::array< double, 3 > > & xs,
std::vector< Point > & path,
std::vector< Seed > & stack) [private]

### TransportElectronBfield()

int Garfield::AvalancheMicroscopic::TransportElectronBfield (
const Seed & seed,
const bool signal,
std::vector< double > & ts,
std::vector< std::array< double, 3 > > & xs,
std::vector< Point > & path,
std::vector< Seed > & stack) [private]

### TransportElectrons()

bool Garfield::AvalancheMicroscopic::TransportElectrons (
std::vector< Seed > & stack,
const bool aval) [private]

### TransportElectronSc()

```
int Garfield::AvalancheMicroscopic::TransportElectronSc (
                const Seed & seed,
                const bool signal,
                std::vector< double > & ts,
                std::vector< std::array< double, 3 > > & xs,
                std::vector< Point > & path,
                std::vector< Seed > & stack)    [private]
```

### transportParticleStack()

```
bool Garfield::AvalancheMicroscopic::transportParticleStack (
                const bool aval,
                std::vector< Seed > & stack,
                std::vector< Seed > & newParticles,
                const bool signal,
                const bool useBfield,
                const bool sc)    [private]
```

### TransportPhoton()

```
void Garfield::AvalancheMicroscopic::TransportPhoton (
                const double x,
                const double y,
                const double z,
                const double t,
                const double e,
                const size_t w,
                std::vector< Seed > & stack)    [private]
```

### UnsetTimeWindow()

```
void Garfield::AvalancheMicroscopic::UnsetTimeWindow ()    [inline]
```

Do not restrict the time interval within which carriers are simulated.

Definition at line 151 of file AvalancheMicroscopic.hh.

```
00151 { m_hasTimeWindow = false; }
```

### UnsetUserHandleAttachment()

```
void Garfield::AvalancheMicroscopic::UnsetUserHandleAttachment ()    [inline]
```

Deactivate the user handle called at every attachment.

Definition at line 272 of file AvalancheMicroscopic.hh.

```
00272 { m_userHandleAttachment = nullptr; }
```

### UnsetUserHandleCollision()

void Garfield::AvalancheMicroscopic::UnsetUserHandleCollision () [inline]

Deactivate the user handle called at every collision.

Definition at line 267 of file AvalancheMicroscopic.hh.

```
00267 { m_userHandleCollision = nullptr; }
```

### UnsetUserHandleInelastic()

void Garfield::AvalancheMicroscopic::UnsetUserHandleInelastic () [inline]

Deactivate the user handle called at every inelastic collision.

Definition at line 277 of file AvalancheMicroscopic.hh.

```
00277 { m_userHandleInelastic = nullptr; }
```

### UnsetUserHandleIonisation()

void Garfield::AvalancheMicroscopic::UnsetUserHandleIonisation () [inline]

Deactivate the user handle called at every ionisation.

Definition at line 283 of file AvalancheMicroscopic.hh.

```
00283 { m_userHandleIonisation = nullptr; }
```

### UnsetUserHandleStep()

void Garfield::AvalancheMicroscopic::UnsetUserHandleStep () [inline]

Deactivate the user handle called at every step.

Definition at line 259 of file AvalancheMicroscopic.hh.

```
00259 { m_userHandleStep = nullptr; }
```

### UseInducedCharge()

void Garfield::AvalancheMicroscopic::UseInducedCharge (
const bool on = true) [inline]

Switch on calculation of the total induced charge (default: off).

Definition at line 61 of file AvalancheMicroscopic.hh.

```
00061 { m_doInducedCharge = on; }
```

UseWeightingPotential()

void Garfield::AvalancheMicroscopic::UseWeightingPotential (
const bool on = true) [inline]

Use the weighting potential (as opposed to the weighting field) for calculating the induced current.

Definition at line 51 of file AvalancheMicroscopic.hh.

```
00051                                          {
00052     m_useWeightingPotential = on;
00053   }
```

### 7.4.4 Friends And Related Symbol Documentation

AvalancheMicroscopicGPU

friend class AvalancheMicroscopicGPU [friend]

Definition at line 463 of file AvalancheMicroscopic.hh.

### 7.4.5 Member Data Documentation

m_className

std::string Garfield::AvalancheMicroscopic::m_className = "AvalancheMicroscopic" [private]

Definition at line 311 of file AvalancheMicroscopic.hh.

m_computePathLength

bool Garfield::AvalancheMicroscopic::m_computePathLength = false [private]

Definition at line 372 of file AvalancheMicroscopic.hh.

m_cudaDevice

int Garfield::AvalancheMicroscopic::m_cudaDevice {-1} [private]

Definition at line 314 of file AvalancheMicroscopic.hh.

```
00314 {-1};
```

m_debug

bool Garfield::AvalancheMicroscopic::m_debug = false [private]

Definition at line 415 of file AvalancheMicroscopic.hh.

### m_debugElectronID

int Garfield::AvalancheMicroscopic::m_debugElectronID {-1} [private]

Definition at line 316 of file AvalancheMicroscopic.hh.
00316 {-1};

### m_debugShowerLoopNum

int Garfield::AvalancheMicroscopic::m_debugShowerLoopNum {-1} [private]

Definition at line 317 of file AvalancheMicroscopic.hh.
00317 {-1};

### m_deltaCut

double Garfield::AvalancheMicroscopic::m_deltaCut = 0. [private]

Definition at line 386 of file AvalancheMicroscopic.hh.

### m_distanceHistogramType

std::vector<int> Garfield::AvalancheMicroscopic::m_distanceHistogramType [private]

Definition at line 363 of file AvalancheMicroscopic.hh.

### m_distanceOption

char Garfield::AvalancheMicroscopic::m_distanceOption = 'r' [private]

Definition at line 362 of file AvalancheMicroscopic.hh.

### m_doInducedCharge

bool Garfield::AvalancheMicroscopic::m_doInducedCharge = false [private]

Definition at line 370 of file AvalancheMicroscopic.hh.

### m_doSignal

bool Garfield::AvalancheMicroscopic::m_doSignal = true [private]

Definition at line 367 of file AvalancheMicroscopic.hh.

### m_electrons

std::vector<Electron> Garfield::AvalancheMicroscopic::m_electrons [private]

Definition at line 325 of file AvalancheMicroscopic.hh.

### m_electrons_gpu

std::vector<Electron> Garfield::AvalancheMicroscopic::m_electrons_gpu [private]

Definition at line 326 of file AvalancheMicroscopic.hh.

### m_gammaCut

double Garfield::AvalancheMicroscopic::m_gammaCut = 0. [private]

Definition at line 387 of file AvalancheMicroscopic.hh.

### m_gpuInterface

AvalancheMicroscopicGPU* Garfield::AvalancheMicroscopic::m_gpuInterface {nullptr} [private]

Definition at line 319 of file AvalancheMicroscopic.hh.
00319 {nullptr};

### m_hasTimeWindow

bool Garfield::AvalancheMicroscopic::m_hasTimeWindow = false [private]

Definition at line 395 of file AvalancheMicroscopic.hh.

### m_histDistance

TH1* Garfield::AvalancheMicroscopic::m_histDistance = nullptr [private]

Definition at line 361 of file AvalancheMicroscopic.hh.

### m_histElectronEnergy

TH1* Garfield::AvalancheMicroscopic::m_histElectronEnergy = nullptr [private]

Definition at line 359 of file AvalancheMicroscopic.hh.

### m_histHoleEnergy

TH1* Garfield::AvalancheMicroscopic::m_histHoleEnergy = nullptr [private]

Definition at line 360 of file AvalancheMicroscopic.hh.

### m_histSecondary

TH1* Garfield::AvalancheMicroscopic::m_histSecondary = nullptr [private]

Definition at line 365 of file AvalancheMicroscopic.hh.

### m_holes

std::vector<Electron> Garfield::AvalancheMicroscopic::m_holes [private]

Definition at line 327 of file AvalancheMicroscopic.hh.

### m_integrateWeightingField

bool Garfield::AvalancheMicroscopic::m_integrateWeightingField = false [private]

Definition at line 369 of file AvalancheMicroscopic.hh.

### m_maxNumShowerLoops

int Garfield::AvalancheMicroscopic::m_maxNumShowerLoops {-1} [private]

Definition at line 315 of file AvalancheMicroscopic.hh.
00315 {-1};

### m_nCollPlot

size_t Garfield::AvalancheMicroscopic::m_nCollPlot = 100 [private]

Definition at line 393 of file AvalancheMicroscopic.hh.

### m_nCollSkip

size_t Garfield::AvalancheMicroscopic::m_nCollSkip = 100 [private]

Definition at line 392 of file AvalancheMicroscopic.hh.

### m_nElectrons

int Garfield::AvalancheMicroscopic::m_nElectrons = 0 [private]

Number of electrons produced.

Definition at line 341 of file AvalancheMicroscopic.hh.

### m_nElectronsGPU

int Garfield::AvalancheMicroscopic::m_nElectronsGPU = 0 [private]

Number of electrons produced.

Definition at line 348 of file AvalancheMicroscopic.hh.

### m_nHoles

int Garfield::AvalancheMicroscopic::m_nHoles = 0 [private]

Number of holes produced.

Definition at line 343 of file AvalancheMicroscopic.hh.

### m_nHolesGPU

int Garfield::AvalancheMicroscopic::m_nHolesGPU = 0 [private]

Number of holes produced.

Definition at line 350 of file AvalancheMicroscopic.hh.

### m_nIons

int Garfield::AvalancheMicroscopic::m_nIons = 0 [private]

Number of ions produced.

Definition at line 345 of file AvalancheMicroscopic.hh.

### m_nIonsGPU

int Garfield::AvalancheMicroscopic::m_nIonsGPU = 0 [private]

Number of ions produced.

Definition at line 352 of file AvalancheMicroscopic.hh.

### m_nullCollScale

double Garfield::AvalancheMicroscopic::m_nullCollScale = 1. [private]

Definition at line 383 of file AvalancheMicroscopic.hh.

### m_photons

std::vector<Photon> Garfield::AvalancheMicroscopic::m_photons [private]

Definition at line 338 of file AvalancheMicroscopic.hh.

### m_plotAttachments

bool Garfield::AvalancheMicroscopic::m_plotAttachments = true [private]

Definition at line 357 of file AvalancheMicroscopic.hh.

### m_plotExcitations

bool Garfield::AvalancheMicroscopic::m_plotExcitations = true [private]

Definition at line 355 of file AvalancheMicroscopic.hh.

### m_plotIonisations

bool Garfield::AvalancheMicroscopic::m_plotIonisations = true [private]

Definition at line 356 of file AvalancheMicroscopic.hh.

### m_rknMinh

double Garfield::AvalancheMicroscopic::m_rknMinh = 1.e-5 [private]

Definition at line 382 of file AvalancheMicroscopic.hh.

### m_rknsteperrortol

double Garfield::AvalancheMicroscopic::m_rknsteperrortol = 1.e-10 [private]

Definition at line 381 of file AvalancheMicroscopic.hh.

### m_rknSteps

bool Garfield::AvalancheMicroscopic::m_rknSteps = false [private]

Definition at line 380 of file AvalancheMicroscopic.hh.

### m_runMode

MPRunMode Garfield::AvalancheMicroscopic::m_runMode {MPRunMode::Normal} [private]

Definition at line 313 of file AvalancheMicroscopic.hh.

```
00313 {MPRunMode::Normal};
```

### m_sensor

Sensor* Garfield::AvalancheMicroscopic::m_sensor = nullptr [private]

Definition at line 323 of file AvalancheMicroscopic.hh.

### m_showProgress

bool Garfield::AvalancheMicroscopic::m_showProgress {false} [private]

Definition at line 318 of file AvalancheMicroscopic.hh.
00318 {false};

### m_sizeCut

unsigned int Garfield::AvalancheMicroscopic::m_sizeCut = 0 [private]

Definition at line 390 of file AvalancheMicroscopic.hh.

### m_stackStoreCPU

std::vector<Seed> Garfield::AvalancheMicroscopic::m_stackStoreCPU [private]

Definition at line 329 of file AvalancheMicroscopic.hh.

### m_stackStoreGPU

std::vector<Electron> Garfield::AvalancheMicroscopic::m_stackStoreGPU [private]

Definition at line 330 of file AvalancheMicroscopic.hh.

### m_stats

Statistics Garfield::AvalancheMicroscopic::m_stats [private]

Definition at line 321 of file AvalancheMicroscopic.hh.

### m_storeDriftLines

bool Garfield::AvalancheMicroscopic::m_storeDriftLines = false [private]

Definition at line 373 of file AvalancheMicroscopic.hh.

### m_tMax

double Garfield::AvalancheMicroscopic::m_tMax = 0. [private]

Definition at line 397 of file AvalancheMicroscopic.hh.

### m_tMin

double Garfield::AvalancheMicroscopic::m_tMin = 0. [private]

Definition at line 396 of file AvalancheMicroscopic.hh.

### m_useBandStructure

bool Garfield::AvalancheMicroscopic::m_useBandStructure = true [private]

Definition at line 375 of file AvalancheMicroscopic.hh.

### m_useBfield

bool Garfield::AvalancheMicroscopic::m_useBfield = false [private]

Definition at line 378 of file AvalancheMicroscopic.hh.

### m_useBfieldAuto

bool Garfield::AvalancheMicroscopic::m_useBfieldAuto = true [private]

Definition at line 377 of file AvalancheMicroscopic.hh.

### m_useNullCollisionSteps

bool Garfield::AvalancheMicroscopic::m_useNullCollisionSteps = false [private]

Definition at line 376 of file AvalancheMicroscopic.hh.

### m_usePhotons

bool Garfield::AvalancheMicroscopic::m_usePhotons = false [private]

Definition at line 374 of file AvalancheMicroscopic.hh.

### m_userHandleAttachment

void(* Garfield::AvalancheMicroscopic::m_userHandleAttachment) (double x, double y, double z, double t, int type, int level, Medium *m) = nullptr [private]

Definition at line 407 of file AvalancheMicroscopic.hh.

### m_userHandleCollision

void(* Garfield::AvalancheMicroscopic::m_userHandleCollision) (double x, double y, double z, double t, int type, int level, Medium *m, double e0, double e1, double dx0, double dy0, double dz0, double dx1, double dy1, double dz1) = nullptr [private]

Definition at line 403 of file AvalancheMicroscopic.hh.

### m_userHandleInelastic

void(* Garfield::AvalancheMicroscopic::m_userHandleInelastic) (double x, double y, double z, double t, int type, int level, Medium *m) = nullptr [private]

Definition at line 409 of file AvalancheMicroscopic.hh.

### m_userHandleIonisation

void(* Garfield::AvalancheMicroscopic::m_userHandleIonisation) (double x, double y, double z, double t, int type, int level, Medium *m) = nullptr [private]

Definition at line 411 of file AvalancheMicroscopic.hh.

### m_userHandleStep

void(* Garfield::AvalancheMicroscopic::m_userHandleStep) (double x, double y, double z, double t, double e, double dx, double dy, double dz, bool hole) = nullptr [private]

Definition at line 400 of file AvalancheMicroscopic.hh.

### m_useWeightingPotential

bool Garfield::AvalancheMicroscopic::m_useWeightingPotential = true [private]

Definition at line 368 of file AvalancheMicroscopic.hh.

### m_viewer

ViewDrift* Garfield::AvalancheMicroscopic::m_viewer = nullptr [private]

Definition at line 354 of file AvalancheMicroscopic.hh.

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh

## 7.5 Garfield::AvalancheGrid::AvalancheNode Struct Reference

Public Attributes

- double ix = 0
- double iy = 0
- double iz = 0
- int n = 1
- int layer = 0
- double townsend = 0
- double attachment = 0
- double velocity = 0
- double stepSize = 0
- std::vector< int > velNormal = {0, 0, 0}
- double time = 0.

  Clock.
- double dt = -1.

  time step.
- bool active = true
- Path path

### 7.5.1 Detailed Description

Definition at line 130 of file AvalancheGrid.hh.

### 7.5.2 Member Data Documentation

active

bool Garfield::AvalancheGrid::AvalancheNode::active = true

Definition at line 149 of file AvalancheGrid.hh.

attachment

double Garfield::AvalancheGrid::AvalancheNode::attachment = 0

Definition at line 140 of file AvalancheGrid.hh.

dt

double Garfield::AvalancheGrid::AvalancheNode::dt = -1.

time step.

Definition at line 147 of file AvalancheGrid.hh.

ix

double Garfield::AvalancheGrid::AvalancheNode::ix = 0

Definition at line 131 of file AvalancheGrid.hh.

iy

double Garfield::AvalancheGrid::AvalancheNode::iy = 0

Definition at line 132 of file AvalancheGrid.hh.

iz

double Garfield::AvalancheGrid::AvalancheNode::iz = 0

Definition at line 133 of file AvalancheGrid.hh.

layer

int Garfield::AvalancheGrid::AvalancheNode::layer = 0

Definition at line 137 of file AvalancheGrid.hh.

n

int Garfield::AvalancheGrid::AvalancheNode::n = 1

Definition at line 135 of file AvalancheGrid.hh.

path

Path Garfield::AvalancheGrid::AvalancheNode::path

Definition at line 151 of file AvalancheGrid.hh.

stepSize

double Garfield::AvalancheGrid::AvalancheNode::stepSize = 0

Definition at line 143 of file AvalancheGrid.hh.

time

double Garfield::AvalancheGrid::AvalancheNode::time = 0.

Clock.

Definition at line 146 of file AvalancheGrid.hh.

townsend

double Garfield::AvalancheGrid::AvalancheNode::townsend = 0

Definition at line 139 of file AvalancheGrid.hh.

velNormal

std::vector<int> Garfield::AvalancheGrid::AvalancheNode::velNormal = {0, 0, 0}

Definition at line 144 of file AvalancheGrid.hh.
00144 {0, 0, 0};

velocity

double Garfield::AvalancheGrid::AvalancheNode::velocity = 0

Definition at line 141 of file AvalancheGrid.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheGrid.hh

## 7.6 Garfield::MediumSilicon::Band Struct Reference

Public Attributes

- int nEnergySteps = 2000
- double eStep = 0.
- double invStep = 0.
- double eFinal
- double eMin = 0.
- int iMin = 0
- std::vector< double > dos
- int nValleys = 1
- double mL = 1.
- double mT = 1.
- double mC = 1.
- double alpha = 0.
- double cfNull
- std::vector< double > cfTot
- std::vector< std::vector< double > > cf
- std::vector< double > energyLoss
- std::vector< int > scatType
- int nLevels = 0

### 7.6.1 Detailed Description

Definition at line 256 of file MediumSilicon.hh.

### 7.6.2 Member Data Documentation

#### alpha

double Garfield::MediumSilicon::Band::alpha = 0.

Definition at line 277 of file MediumSilicon.hh.

#### cf

std::vector<std::vector<double> > Garfield::MediumSilicon::Band::cf

Definition at line 283 of file MediumSilicon.hh.

#### cfNull

double Garfield::MediumSilicon::Band::cfNull

Definition at line 279 of file MediumSilicon.hh.

#### cfTot

std::vector<double> Garfield::MediumSilicon::Band::cfTot

Definition at line 281 of file MediumSilicon.hh.

#### dos

std::vector<double> Garfield::MediumSilicon::Band::dos

Definition at line 267 of file MediumSilicon.hh.

#### eFinal

double Garfield::MediumSilicon::Band::eFinal

Definition at line 261 of file MediumSilicon.hh.

#### eMin

double Garfield::MediumSilicon::Band::eMin = 0.

Definition at line 263 of file MediumSilicon.hh.

### energyLoss

std::vector<double> Garfield::MediumSilicon::Band::energyLoss

Definition at line 284 of file MediumSilicon.hh.

### eStep

double Garfield::MediumSilicon::Band::eStep = 0.

Definition at line 258 of file MediumSilicon.hh.

### iMin

int Garfield::MediumSilicon::Band::iMin = 0

Definition at line 265 of file MediumSilicon.hh.

### invStep

double Garfield::MediumSilicon::Band::invStep = 0.

Definition at line 259 of file MediumSilicon.hh.

### mC

double Garfield::MediumSilicon::Band::mC = 1.

Definition at line 275 of file MediumSilicon.hh.

### mL

double Garfield::MediumSilicon::Band::mL = 1.

Definition at line 271 of file MediumSilicon.hh.

### mT

double Garfield::MediumSilicon::Band::mT = 1.

Definition at line 273 of file MediumSilicon.hh.

### nEnergySteps

int Garfield::MediumSilicon::Band::nEnergySteps = 2000

Definition at line 257 of file MediumSilicon.hh.

nLevels

int Garfield::MediumSilicon::Band::nLevels = 0

Definition at line 288 of file MediumSilicon.hh.

nValleys

int Garfield::MediumSilicon::Band::nValleys = 1

Definition at line 269 of file MediumSilicon.hh.

scatType

std::vector<int> Garfield::MediumSilicon::Band::scatType

Definition at line 286 of file MediumSilicon.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumSilicon.hh

## 7.7 Garfield::ComponentAnalyticField::Charge3d Struct Reference

Public Attributes

- double x
- double y
- double z
  Coordinates.
- double e
  Charge.

### 7.7.1 Detailed Description

Definition at line 518 of file ComponentAnalyticField.hh.

### 7.7.2 Member Data Documentation

e

double Garfield::ComponentAnalyticField::Charge3d::e

Charge.

Definition at line 520 of file ComponentAnalyticField.hh.

x

double Garfield::ComponentAnalyticField::Charge3d::x

Definition at line 519 of file ComponentAnalyticField.hh.

y

double Garfield::ComponentAnalyticField::Charge3d::y

Definition at line 519 of file ComponentAnalyticField.hh.

z

double Garfield::ComponentAnalyticField::Charge3d::z

Coordinates.

Definition at line 519 of file ComponentAnalyticField.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh

## 7.8 Garfield::TrackBichsel::Cluster Struct Reference

#include <TrackBichsel.hh>

Public Attributes

- double x
- double y
- double z
- double t
- double energy

### 7.8.1 Detailed Description

Definition at line 20 of file TrackBichsel.hh.

### 7.8.2 Member Data Documentation

energy

double Garfield::TrackBichsel::Cluster::energy

Definition at line 22 of file TrackBichsel.hh.

t

double Garfield::TrackBichsel::Cluster::t

Definition at line 21 of file TrackBichsel.hh.

x

double Garfield::TrackBichsel::Cluster::x

Definition at line 21 of file TrackBichsel.hh.

y

double Garfield::TrackBichsel::Cluster::y

Definition at line 21 of file TrackBichsel.hh.

z

double Garfield::TrackBichsel::Cluster::z

Definition at line 21 of file TrackBichsel.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackBichsel.hh

## 7.9 Garfield::TrackDegrade::Cluster Struct Reference

#include <TrackDegrade.hh>

Public Attributes

- double x
- double y
- double z
- double t
- std::vector< Electron > deltaElectrons
- std::vector< Electron > electrons
- std::vector< Excitation > excitations

### 7.9.1 Detailed Description

Definition at line 35 of file TrackDegrade.hh.

### 7.9.2 Member Data Documentation

#### deltaElectrons

std::vector<Electron> Garfield::TrackDegrade::Cluster::deltaElectrons

Definition at line 37 of file TrackDegrade.hh.

#### electrons

std::vector<Electron> Garfield::TrackDegrade::Cluster::electrons

Definition at line 38 of file TrackDegrade.hh.

#### excitations

std::vector<Excitation> Garfield::TrackDegrade::Cluster::excitations

Definition at line 39 of file TrackDegrade.hh.

#### t

double Garfield::TrackDegrade::Cluster::t

Definition at line 36 of file TrackDegrade.hh.

#### x

double Garfield::TrackDegrade::Cluster::x

Definition at line 36 of file TrackDegrade.hh.

#### y

double Garfield::TrackDegrade::Cluster::y

Definition at line 36 of file TrackDegrade.hh.

#### z

double Garfield::TrackDegrade::Cluster::z

Definition at line 36 of file TrackDegrade.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh

## 7.10 Garfield::TrackElectron::Cluster Struct Reference

#include <TrackElectron.hh>

Public Attributes

- double x
- double y
- double z
- double t
- double esec

### 7.10.1 Detailed Description

Definition at line 16 of file TrackElectron.hh.

### 7.10.2 Member Data Documentation

esec

double Garfield::TrackElectron::Cluster::esec

Definition at line 18 of file TrackElectron.hh.

t

double Garfield::TrackElectron::Cluster::t

Definition at line 17 of file TrackElectron.hh.

x

double Garfield::TrackElectron::Cluster::x

Definition at line 17 of file TrackElectron.hh.

y

double Garfield::TrackElectron::Cluster::y

Definition at line 17 of file TrackElectron.hh.

z

double Garfield::TrackElectron::Cluster::z

Definition at line 17 of file TrackElectron.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackElectron.hh

## 7.11 Garfield::TrackHeed::Cluster Struct Reference

#include <TrackHeed.hh>

Public Attributes

- double x
- double y
- double z
- double t
- double energy
- double extra
- std::vector< Photon > photons
- std::vector< Electron > electrons
- std::vector< Electron > ions

### 7.11.1 Detailed Description

Definition at line 58 of file TrackHeed.hh.

### 7.11.2 Member Data Documentation

electrons

std::vector<Electron> Garfield::TrackHeed::Cluster::electrons

Definition at line 63 of file TrackHeed.hh.

energy

double Garfield::TrackHeed::Cluster::energy

Definition at line 60 of file TrackHeed.hh.

extra

double Garfield::TrackHeed::Cluster::extra

Definition at line 61 of file TrackHeed.hh.

ions

std::vector<Electron> Garfield::TrackHeed::Cluster::ions

Definition at line 64 of file TrackHeed.hh.

photons

std::vector<Photon> Garfield::TrackHeed::Cluster::photons

Definition at line 62 of file TrackHeed.hh.

t

double Garfield::TrackHeed::Cluster::t

Definition at line 59 of file TrackHeed.hh.

x

double Garfield::TrackHeed::Cluster::x

Definition at line 59 of file TrackHeed.hh.

y

double Garfield::TrackHeed::Cluster::y

Definition at line 59 of file TrackHeed.hh.

z

double Garfield::TrackHeed::Cluster::z

Definition at line 59 of file TrackHeed.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackHeed.hh

## 7.12 Garfield::TrackPAI::Cluster Struct Reference

#include <TrackPAI.hh>

Public Attributes

- double x
- double y
- double z
- double t
- double energy

### 7.12.1 Detailed Description

Definition at line 16 of file TrackPAI.hh.

### 7.12.2 Member Data Documentation

energy

double Garfield::TrackPAI::Cluster::energy

Definition at line 18 of file TrackPAI.hh.

t

double Garfield::TrackPAI::Cluster::t

Definition at line 17 of file TrackPAI.hh.

x

double Garfield::TrackPAI::Cluster::x

Definition at line 17 of file TrackPAI.hh.

y

double Garfield::TrackPAI::Cluster::y

Definition at line 17 of file TrackPAI.hh.

z

double Garfield::TrackPAI::Cluster::z

Definition at line 17 of file TrackPAI.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackPAI.hh

## 7.13 Garfield::TrackSimple::Cluster Struct Reference

#include <TrackSimple.hh>

Public Attributes

- double x
- double y
- double z
- double t
- double energy

### 7.13.1 Detailed Description

Definition at line 14 of file TrackSimple.hh.

### 7.13.2 Member Data Documentation

energy

double Garfield::TrackSimple::Cluster::energy

Definition at line 16 of file TrackSimple.hh.

t

double Garfield::TrackSimple::Cluster::t

Definition at line 15 of file TrackSimple.hh.

x

double Garfield::TrackSimple::Cluster::x

Definition at line 15 of file TrackSimple.hh.

y

double Garfield::TrackSimple::Cluster::y

Definition at line 15 of file TrackSimple.hh.

z

double Garfield::TrackSimple::Cluster::z

Definition at line 15 of file TrackSimple.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackSimple.hh

## 7.14 Garfield::TrackSrim::Cluster Struct Reference

#include <TrackSrim.hh>

Public Attributes

- double x
- double y
- double z
- double t
  Cluster location and time.
- double energy
  Energy spent to make the cluster.
- double kinetic
  Ion energy when cluster was created.
- int n
  Number of electrons in this cluster.

### 7.14.1 Detailed Description

Definition at line 90 of file TrackSrim.hh.

### 7.14.2 Member Data Documentation

energy

double Garfield::TrackSrim::Cluster::energy

Energy spent to make the cluster.

Definition at line 92 of file TrackSrim.hh.

kinetic

double Garfield::TrackSrim::Cluster::kinetic

Ion energy when cluster was created.

Definition at line 93 of file TrackSrim.hh.

n

int Garfield::TrackSrim::Cluster::n

Number of electrons in this cluster.

Definition at line 94 of file TrackSrim.hh.

t

double Garfield::TrackSrim::Cluster::t

Cluster location and time.

Definition at line 91 of file TrackSrim.hh.

x

double Garfield::TrackSrim::Cluster::x

Definition at line 91 of file TrackSrim.hh.

y

double Garfield::TrackSrim::Cluster::y

Definition at line 91 of file TrackSrim.hh.

z

double Garfield::TrackSrim::Cluster::z

Definition at line 91 of file TrackSrim.hh.

The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackSrim.hh

## 7.15 Garfield::TrackTrim::Cluster Struct Reference

#include <TrackTrim.hh>

Public Attributes

- double x
- double y
- double z

  Location.
- double t

  Time.
- double energy

  Energy spent to make the cluster.
- double ekin

  Ion energy when cluster was created.
- int n

  Number of electrons in this cluster.

### 7.15.1 Detailed Description

Definition at line 60 of file TrackTrim.hh.

### 7.15.2 Member Data Documentation

ekin

double Garfield::TrackTrim::Cluster::ekin

Ion energy when cluster was created.

Definition at line 64 of file TrackTrim.hh.

energy

double Garfield::TrackTrim::Cluster::energy

Energy spent to make the cluster.

Definition at line 63 of file TrackTrim.hh.

n

int Garfield::TrackTrim::Cluster::n

Number of electrons in this cluster.

Definition at line 65 of file TrackTrim.hh.

t

double Garfield::TrackTrim::Cluster::t

Time.

Definition at line 62 of file TrackTrim.hh.

x

double Garfield::TrackTrim::Cluster::x

Definition at line 61 of file TrackTrim.hh.

y

double Garfield::TrackTrim::Cluster::y

Definition at line 61 of file TrackTrim.hh.

z

double Garfield::TrackTrim::Cluster::z

Location.

Definition at line 61 of file TrackTrim.hh.

The documentation for this struct was generated from the following file:

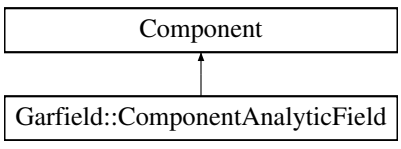
- /builds/garfield/docs/source/Include/Garfield/TrackTrim.hh

## 7.16 Garfield::ComponentAnalyticField Class Reference

Semi-analytic calculation of two-dimensional configurations consisting of wires, planes, and tubes.

#include <ComponentAnalyticField.hh>

Inheritance diagram for Garfield::ComponentAnalyticField:



Classes

- struct Charge3d
- struct Pixel
- struct Plane
- struct Strip
- struct Wire

Public Types

- enum Cell {
  A00 , B1X , B1Y , B2X ,
  B2Y , C10 , C2X , C2Y ,
  C30 , D10 , D20 , D30 ,
  D40 , Unknown }

Public Member Functions

- ComponentAnalyticField ()

  Constructor.
- ~ComponentAnalyticField ()

  Destructor.
- void SetMedium (Medium *medium)

  Set the medium inside the cell.
- void AddWire (const double x, const double y, const double diameter, const double voltage, const std::string &label="", const double length=100., const double tension=50., const double rho=19.3, const int ntrap=5)

  Add a wire at (x, y) .
- void AddTube (const double radius, const double voltage, const int nEdges, const std::string &label="")

  Add a tube.
- void AddPlaneX (const double x, const double voltage, const std::string &label="")

  Add a plane at constant x.
- void AddPlaneY (const double y, const double voltage, const std::string &label="")

  Add a plane at constant y.
- void AddPlaneR (const double r, const double voltage, const std::string &label="")

  Add a plane at constant radius.
- void AddPlanePhi (const double phi, const double voltage, const std::string &label="")

  Add a plane at constant phi.
- void AddStripOnPlaneX (const char direction, const double x, const double smin, const double smax, const std::string &label, const double gap=-1.)

  Add a strip in the y or z direction on an existing plane at constant x.
- void AddStripOnPlaneY (const char direction, const double y, const double smin, const double smax, const std::string &label, const double gap=-1.)

  Add a strip in the x or z direction on an existing plane at constant y.
- void AddStripOnPlaneR (const char direction, const double r, const double smin, const double smax, const std::string &label, const double gap=-1.)

  Add a strip in the phi or z direction on an existing plane at constant radius.
- void AddStripOnPlanePhi (const char direction, const double phi, const double smin, const double smax, const std::string &label, const double gap=-1.)

  Add a strip in the r or z direction on an existing plane at constant phi.

- void AddPixelOnPlaneX (const double x, const double ymin, const double ymax, const double zmin, const double zmax, const std::string &label, const double gap=-1., const double rot=0.)

  Add a pixel on an existing plane at constant x.
- void AddPixelOnPlaneY (const double y, const double xmin, const double xmax, const double zmin, const double zmax, const std::string &label, const double gap=-1., const double rot=0.)

  Add a pixel on an existing plane at constant y.
- void AddPixelOnPlaneR (const double r, const double phimin, const double phimax, const double zmin, const double zmax, const std::string &label, const double gap=-1.)

  Add a pixel on an existing plane at constant radius.
- void AddPixelOnPlanePhi (const double phi, const double rmin, const double rmax, const double zmin, const double zmax, const std::string &label, const double gap=-1.)

  Add a pixel on an existing plane at constant phi.
- void SetPeriodicityX (const double s)

  Set the periodic length [cm] in the x-direction.
- void SetPeriodicityY (const double s)

  Set the periodic length [cm] in the y-direction.
- void SetPeriodicityPhi (const double phi)

  Set the periodicity [degree] in phi.
- bool GetPeriodicityX (double &s)

  Get the periodic length in the x-direction.
- bool GetPeriodicityY (double &s)

  Get the periodic length in the y-direction.
- bool GetPeriodicityPhi (double &s)

  Get the periodicity [degree] in phi.
- void SetCartesianCoordinates ()

  Use Cartesian coordinates (default).
- void SetPolarCoordinates ()

  Use polar coordinates.
- bool IsPolar () const

  Are polar coordinates being used?
- void PrintCell ()

  Print all available information on the cell.
- void PlotCell (TPad *pad)

  Make a plot of the cell layout.
- void AddCharge (const double x, const double y, const double z, const double q)

  Add a point charge.
- void ClearCharges ()

  Remove all point charges.
- void PrintCharges () const

  Print a list of the point charges.
- std::string GetCellType ()

  Return the cell type.
- void AddReadout (const std::string &label, const bool silent=false)

  Request calculation of weighting field and potential for a given group of wires or planes.
- void SetNumberOfCellCopies (const unsigned int nfourier)
- bool MultipoleMoments (const unsigned int iw, const unsigned int order=4, const bool print=false, const bool plot=false, const double rmult=1., const double eps=1.e-4, const unsigned int nMax↩ Iter=20)

  Calculate multipole moments for a given wire.
- void EnableDipoleTerms (const bool on=true)

  Request dipole terms be included for each of the wires (default: off).

- void EnableChargeCheck (const bool on=true)

  Check the quality of the capacitance matrix inversion (default: off).
- unsigned int GetNumberOfWires () const

  Get the number of wires.
- bool GetWire (const unsigned int i, double &x, double &y, double &diameter, double &voltage, std::string &label, double &length, double &charge, int &ntrap) const

  Retrieve the parameters of a wire.
- unsigned int GetNumberOfPlanesX () const

  Get the number of equipotential planes at constant x.
- unsigned int GetNumberOfPlanesY () const

  Get the number of equipotential planes at constant y.
- unsigned int GetNumberOfPlanesR () const

  Get the number of equipotential planes at constant radius.
- unsigned int GetNumberOfPlanesPhi () const

  Get the number of equipotential planes at constant phi.
- bool GetPlaneX (const unsigned int i, double &x, double &voltage, std::string &label) const

  Retrieve the parameters of a plane at constant x.
- bool GetPlaneY (const unsigned int i, double &y, double &voltage, std::string &label) const

  Retrieve the parameters of a plane at constant y.
- bool GetPlaneR (const unsigned int i, double &r, double &voltage, std::string &label) const

  Retrieve the parameters of a plane at constant radius.
- bool GetPlanePhi (const unsigned int i, double &phi, double &voltage, std::string &label) const

  Retrieve the parameters of a plane at constant phi.
- bool GetTube (double &r, double &voltage, int &nEdges, std::string &label) const

  Retrieve the tube parameters.
- bool OptimiseOnTrack (const std::vector< std::string > &groups, const std::string &field_function, const double target, const double x0, const double y0, const double x1, const double y1, const unsigned int nP=20, const bool print=true)

  Vary the potential of selected electrodes to match (a function of) the potential or field along a given line to a target.
- bool OptimiseOnGrid (const std::vector< std::string > &groups, const std::string &field_function, const double target, const double x0, const double y0, const double x1, const double y1, const unsigned int nX=10, const unsigned int nY=10, const bool print=true)

  Vary the potential of selected electrodes to match (a function of) the potential or field on an x-y (or r-phi) grid of points.
- bool OptimiseOnWires (const std::vector< std::string > &groups, const std::string &field_function, const double target, const std::vector< unsigned int > &wires, const bool print=true)

  Vary the potential of selected electrodes to match (a function of) the potential or field on the surfaces of a set of wires.
- void SetOptimisationParameters (const double dist=1., const double eps=1.e-4, const unsigned int nMaxIter=10)

  Set the conditions at which to allow the iteration to stop.
- bool ElectricFieldAtWire (const unsigned int iw, double &ex, double &ey)

  Calculate the electric field at a given wire position, as if the wire itself were not there, but with the presence of its mirror images.
- void SetScanningGrid (const unsigned int nX, const unsigned int nY)

  Set the number of grid lines at which the electrostatic force as function of the wire displacement is computed.
- void SetScanningArea (const double xmin, const double xmax, const double ymin, const double ymax)

  Specify explicitly the boundaries of the the scanning area (i.
- void SetScanningAreaLargest ()

  Set the scanning area to the largest area around each wire which is free from other cell elements.

- void SetScanningAreaFirstOrder (const double scale=2.)

  Set the scanning area based on the zeroth-order estimates of the wire shift, enlarged by a scaling factor.
- void EnableExtrapolation (const bool on=true)

  Switch on/off extrapolation of electrostatic forces beyond the scanning area (default: off).
- void SetGravity (const double dx, const double dy, const double dz)

  Set the gravity orientation.
- void GetGravity (double &dx, double &dy, double &dz) const

  Get the gravity orientation.
- void EnableGravity (const bool on=true)

  Include gravity in the sag computation or not.
- void EnableElectrostaticForce (const bool on=true)

  Include the electrostatic force in the sag computation or not.
- bool ForcesOnWire (const unsigned int iw, std::vector< double > &xMap, std::vector< double > &yMap, std::vector< std::vector< double > > &fxMap, std::vector< std::vector< double > > &fyMap)

  Calculate a table of the forces acting on a wire.
- bool WireDisplacement (const unsigned int iw, const bool detailed, std::vector< double > &csag, std::vector< double > &xsag, std::vector< double > &ysag, double &stretch, const bool print=true)

  Compute the sag profile of a wire.
- void SetNumberOfShots (const unsigned int n)

  Set the number of shots used for numerically solving the wire sag differential equation.
- void SetNumberOfSteps (const unsigned int n)

  Set the number of integration steps within each shot (must be >= 1).
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- bool GetVoltageRange (double &pmin, double &pmax) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- bool GetBoundingBox (double &x0, double &y0, double &z0, double &x1, double &y1, double &z1) override
- bool GetElementaryCell (double &x0, double &y0, double &z0, double &x1, double &y1, double &z1) override
- bool CrossedWire (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, double &xc, double &yc, double &zc, const bool centre, double &rc) override
- bool InTrapRadius (const double q0, const double x0, const double y0, const double z0, double &xw, double &yx, double &rw) override
- bool CrossedPlane (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, double &xc, double &yc, double &zc) override
- double StepSizeHint () override

Private Types

- enum class ScanningRange { Largest = 0 , FirstOrder , User }

Private Member Functions

- void UpdatePeriodicity () override
- void Reset () override
- void CellInit ()
- bool Prepare ()
- bool CellCheck ()
- bool WireCheck () const
- bool CellType ()
- std::string GetCellType (const Cell) const
- bool PrepareStrips ()
- bool PrepareSignals ()
- bool SetupWireSignals ()
- bool SetupPlaneSignals ()
- bool Setup ()
- bool Update (const std::vector< double > &vw, const std::array< double, 5 > &vp)
- bool SetupA00 ()
- bool SetupB1X ()
- bool SetupB1Y ()
- bool SetupB2X ()
- bool SetupB2Y ()
- bool SetupC10 ()
- bool SetupC2X ()
- bool SetupC2Y ()
- bool SetupC30 ()
- bool SetupD10 ()
- bool SetupD20 ()
- bool SetupD30 ()
- bool IprA00 (const int mx, const int my, std::vector< std::vector< std::complex< double > > > &mat)
- bool IprB2X (const int my, std::vector< std::vector< std::complex< double > > > &mat)
- bool IprB2Y (const int mx, std::vector< std::vector< std::complex< double > > > &mat)
- bool IprC2X (std::vector< std::vector< std::complex< double > > > &mat)
- bool IprC2Y (std::vector< std::vector< std::complex< double > > > &mat)
- bool IprC30 (std::vector< std::vector< std::complex< double > > > &mat)
- bool IprD10 (std::vector< std::vector< std::complex< double > > > &mat)
- bool IprD30 (std::vector< std::vector< std::complex< double > > > &mat)
- bool SetupDipoleTerms ()
- bool Charge (std::vector< std::vector< double > > &mat)
- int Field (const double xin, const double yin, const double zin, double &ex, double &ey, double &ez, double &volt, const bool opt)
- void FieldA00 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldB1X (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldB1Y (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldB2X (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldB2Y (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldC10 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldC2X (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const

- void FieldC2Y (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldC30 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldD10 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldD20 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void FieldD30 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void Field3dA00 (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &volt) const
- void Field3dB2X (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &volt) const
- void Field3dB2Y (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &volt) const
- void Field3dD10 (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &volt) const
- bool Wfield (const double x, const double y, const double z, double &ex, double &ey, double &ez, const std::string &label) const
- void WfieldWireA00 (const double x, const double y, double &ex, double &ey, const int mx, const int my, const std::vector< double > &qw) const
- void WfieldWireB2X (const double x, const double y, double &ex, double &ey, const int my, const std::vector< double > &qw) const
- void WfieldWireB2Y (const double x, const double y, double &ex, double &ey, const int mx, const std::vector< double > &qw) const
- void WfieldWireC2X (const double x, const double y, double &ex, double &ey, const std::vector< double > &qw) const
- void WfieldWireC2Y (const double x, const double y, double &ex, double &ey, const std::vector< double > &qw) const
- void WfieldWireC30 (const double x, const double y, double &ex, double &ey, const std::vector< double > &qw) const
- void WfieldWireD10 (const double x, const double y, double &ex, double &ey, const std::vector< double > &qw) const
- void WfieldWireD30 (const double x, const double y, double &ex, double &ey, const std::vector< double > &qw) const
- void WfieldPlaneA00 (const double x, const double y, double &ex, double &ey, const int mx, const int my, const std::vector< double > &qp) const
- void WfieldPlaneB2X (const double x, const double y, double &ex, double &ey, const int my, const std::vector< double > &qp) const
- void WfieldPlaneB2Y (const double x, const double ypos, double &ex, double &ey, const int mx, const std::vector< double > &qp) const
- void WfieldPlaneC2X (const double x, const double y, double &ex, double &ey, const std::vector< double > &qp) const
- void WfieldPlaneC2Y (const double x, const double y, double &ex, double &ey, const std::vector< double > &qp) const
- void WfieldPlaneC30 (const double x, const double y, double &ex, double &ey, const std::vector< double > &qp) const
- void WfieldPlaneD10 (const double x, const double y, double &ex, double &ey, const std::vector< double > &qp) const
- void WfieldPlaneD30 (const double x, const double y, double &ex, double &ey, const std::vector< double > &qp) const
- void WfieldStripZ (const double x, const double y, double &ex, double &ey, const int ip, const Strip &strip) const
- void WfieldStripXy (const double x, const double y, const double z, double &ex, double &ey, double &ez, const int ip, const Strip &strip) const

- void WfieldStrip (const double x, const double y, const double g, const double w, double &fx, double &fy) const
- void WfieldPixel (const double x, const double y, const double z, double &ex, double &ey, double &ez, const int ip, const Pixel &pixel) const
- double Wpot (const double x, const double y, const double z, const std::string &label) const
- double WpotWireA00 (const double x, const double y, const int mx, const int my, const std::vector< double > &qw) const
- double WpotWireB2X (const double x, const double y, const int my, const std::vector< double > &qw) const
- double WpotWireB2Y (const double x, const double y, const int mx, const std::vector< double > &qw) const
- double WpotWireC2X (const double x, const double y, const std::vector< double > &qw) const
- double WpotWireC2Y (const double x, const double y, const std::vector< double > &qw) const
- double WpotWireC30 (const double x, const double y, const std::vector< double > &qw) const
- double WpotWireD10 (const double x, const double y, const std::vector< double > &qw) const
- double WpotWireD30 (const double x, const double y, const std::vector< double > &qw) const
- double WpotPlaneA00 (const double x, const double y, const int mx, const int my, const std::vector< double > &qp) const
- double WpotPlaneB2X (const double x, const double y, const int my, const std::vector< double > &qp) const
- double WpotPlaneB2Y (const double x, const double y, const int mx, const std::vector< double > &qp) const
- double WpotPlaneC2X (const double x, const double y, const std::vector< double > &qp) const
- double WpotPlaneC2Y (const double x, const double y, const std::vector< double > &qp) const
- double WpotPlaneC30 (const double x, const double y, const std::vector< double > &qp) const
- double WpotPlaneD10 (const double x, const double y, const std::vector< double > &qp) const
- double WpotPlaneD30 (const double x, const double y, const std::vector< double > &qp) const
- double WpotStripZ (const double x, const double y, const int ip, const Strip &strip) const
- double WpotStripXy (const double x, const double y, const double z, const int ip, const Strip &strip) const
- double WpotPixel (const double x, const double y, const double z, const int ip, const Pixel &pixel) const
- void FieldAtWireA00 (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireB1X (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireB1Y (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireB2X (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireB2Y (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireC10 (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireC2X (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireC2Y (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireC30 (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireD10 (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireD20 (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const
- void FieldAtWireD30 (const double xpos, const double ypos, double &ex, double &ey, const std::vector< bool > &cnalso) const

- void DipoleFieldA00 (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void DipoleFieldB1X (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void DipoleFieldB1Y (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void DipoleFieldB2X (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- void DipoleFieldB2Y (const double xpos, const double ypos, double &ex, double &ey, double &volt, const bool opt) const
- double Ph2 (const double xpos, const double ypos) const
- double Ph2Lim (const double radius) const
- void E2Sum (const double xpos, const double ypos, double &ex, double &ey) const
- void ConformalMap (const std::complex< double > &z, std::complex< double > &ww, std::complex< double > &wd) const
- bool SagDetailed (const Wire &wire, const std::vector< double > &xMap, const std::vector< double > &yMap, const std::vector< std::vector< double > > &fxMap, const std::vector< std::vector< double > > &fyMap, std::vector< double > &csag, std::vector< double > &xsag, std::vector< double > &ysag) const
- bool GetForceRatio (const Wire &wire, const double coor, const std::array< double, 2 > &bend, const std::array< double, 2 > &dbend, std::array< double, 2 > &f, const std::vector< double > &xMap, const std::vector< double > &yMap, const std::vector< std::vector< double > > &fxMap, const std::vector< std::vector< double > > &fyMap) const
- bool FindZeroes (const Wire &wire, const double h, std::vector< double > &x, const std::vector< double > &xMap, const std::vector< double > &yMap, const std::vector< std::vector< double > > &fxMap, const std::vector< std::vector< double > > &fyMap) const
- bool StepRKN (const Wire &wire, const double h, double &x, std::array< double, 2 > &y, std::array< double, 2 > &yp, const std::vector< double > &xMap, const std::vector< double > &yMap, const std::vector< std::vector< double > > &fxMap, const std::vector< std::vector< double > > &fyMap) const
- bool Trace (const Wire &wire, const double h, const std::vector< double > &xx, std::vector< double > &f, const std::vector< double > &xMap, const std::vector< double > &yMap, const std::vector< std::vector< double > > &fxMap, const std::vector< std::vector< double > > &fyMap) const
- size_t SignalLayer (const int mx, const int my) const
- void InitialiseFitParameters (const std::vector< std::string > &groups, std::vector< double > &vw0, std::array< double, 5 > &vp0, std::vector< double > &aFit, std::vector< std::vector< unsigned int > > &wiresInGroup, std::vector< std::vector< unsigned int > > &planesInGroup)

Static Private Member Functions

- static bool InTube (const double x0, const double y0, const double a, const int n)

Private Attributes

- std::mutex m_mutex
- Medium * m_medium = nullptr
- bool m_chargeCheck = false
- bool m_cellset = false
- bool m_sigset = false
- bool m_polar = false
- Cell m_cellType
- double m_xmin
- double m_xmax
- double m_ymin

- double m_ymax
- double m_zmin
- double m_zmax
- double m_vmin
- double m_vmax
- bool m_perx = false
- bool m_pery = false
- double m_sx
- double m_sy
- int m_nFourier = 1
- Cell m_cellTypeFourier = A00
- bool m_fperx = false
- bool m_fpery = false
- int m_mxmin = 0
- int m_mxmax = 0
- int m_mymin = 0
- int m_mymax = 0
- int m_mfexp = 0
- std::vector< std::string > m_readout
- unsigned int m_nWires
- std::vector< Wire > m_w
- bool m_dipole = false
- std::vector< double > m_cosph2
- std::vector< double > m_sinph2
- std::vector< double > m_amp2
- std::vector< double > m_b2sin
- int m_mode
- std::complex< double > m_zmult
- double m_p1
- double m_p2
- double m_c1
- std::vector< std::complex< double > > m_zw
- double m_kappa
- double m_v0
- double m_corvta
- double m_corvtb
- double m_corvtc
- bool m_ynplan [4]
- bool m_ynplax
- bool m_ynplay
- double m_coplan [4]
- double m_coplax
- double m_coplay
- double m_vtplan [4]
- std::array< Plane, 5 > m_planes
- bool m_tube = false
- int m_mtube = 0
- int m_ntube = 1
- double m_cotube = 1.
- double m_cotube2 = 1.
- double m_vttube = 0.
- double m_dmin = -1.
- std::vector< std::vector< std::vector< double > > > m_qwire
- std::vector< std::vector< std::vector< double > > > m_qplane
- std::vector< Charge3d > m_ch3d

- unsigned int m_nTermBessel = 10
- unsigned int m_nTermPoly = 100
- bool m_useElectrostaticForce = true
- bool m_useGravitationalForce = true
- std::array< double, 3 > m_down {{0, 0, 1}}
- unsigned int m_nShots = 2
- unsigned int m_nSteps = 20
- ScanningRange m_scanRange = ScanningRange::FirstOrder
- double m_xScanMin = 0.
- double m_xScanMax = 0.
- double m_yScanMin = 0.
- double m_yScanMax = 0.
- double m_scaleRange = 2.
- unsigned int m_nScanX = 11
- unsigned int m_nScanY = 11
- bool m_extrapolateForces = false
- double m_optDist = 1.
- double m_optEps = 1.e-4
- unsigned int m_optNitmax = 10

### 7.16.1 Detailed Description

Semi-analytic calculation of two-dimensional configurations consisting of wires, planes, and tubes.

Definition at line 21 of file ComponentAnalyticField.hh.

### 7.16.2 Member Enumeration Documentation

Cell

enum Garfield::ComponentAnalyticField::Cell

Enumerator

| | |
|---|---|
| A00 | |
| B1X | |
| B1Y | |
| B2X | |
| B2Y | |
| C10 | |
| C2X | |
| C2Y | |
| C30 | |
| D10 | |
| D20 | |
| D30 | |
| D40 | |
| Unknown | |

Definition at line 365 of file ComponentAnalyticField.hh.

```
00365           {
00366     A00,
00367     B1X,
00368     B1Y,
00369     B2X,
00370     B2Y,
00371     C10,
00372     C2X,
00373     C2Y,
00374     C30,
00375     D10,
00376     D20,
00377     D30,
00378     D40,
00379     Unknown
00380   };
```

ScanningRange

enum class Garfield::ComponentAnalyticField::ScanningRange [strong], [private]

Enumerator

| | |
|---|---|
| Largest | |
| FirstOrder | |
| User | |

Definition at line 536 of file ComponentAnalyticField.hh.

```
00536 { Largest = 0, FirstOrder, User };
```

### 7.16.3 Constructor & Destructor Documentation

ComponentAnalyticField()

Garfield::ComponentAnalyticField::ComponentAnalyticField ()

Constructor.

~ComponentAnalyticField()

Garfield::ComponentAnalyticField::~ComponentAnalyticField () [inline]

Destructor.

Definition at line 26 of file ComponentAnalyticField.hh.

```
00026 {}
```

### 7.16.4 Member Function Documentation

#### AddCharge()

```
void Garfield::ComponentAnalyticField::AddCharge (
                const double x,
                const double y,
                const double z,
                const double q)
```

Add a point charge.

#### AddPixelOnPlanePhi()

```
void Garfield::ComponentAnalyticField::AddPixelOnPlanePhi (
                const double phi,
                const double rmin,
                const double rmax,
                const double zmin,
                const double zmax,
                const std::string & label,
                const double gap = -1.)
```

Add a pixel on an existing plane at constant phi.

#### AddPixelOnPlaneR()

```
void Garfield::ComponentAnalyticField::AddPixelOnPlaneR (
                const double r,
                const double phimin,
                const double phimax,
                const double zmin,
                const double zmax,
                const std::string & label,
                const double gap = -1.)
```

Add a pixel on an existing plane at constant radius.

#### AddPixelOnPlaneX()

```
void Garfield::ComponentAnalyticField::AddPixelOnPlaneX (
                const double x,
                const double ymin,
                const double ymax,
                const double zmin,
                const double zmax,
                const std::string & label,
                const double gap = -1.,
                const double rot = 0.)
```

Add a pixel on an existing plane at constant x.

Parameters

| | |
|---|---|
| x | coordinate of the plane. |
| ymin | lower limit of the pixel cell in y, |
| ymax | upper limit of the pixel cell in y. |
| zmin | lower limit of the pixel cell in z. |
| zmax | upper limit of the pixel cell in z. |
| label | weighting field identifier. |
| gap | distance to the opposite plane (optional). |
| rot | rotation angle (rad) of the pixel (optional). |

### AddPixelOnPlaneY()

```
void Garfield::ComponentAnalyticField::AddPixelOnPlaneY (
                const double y,
                const double xmin,
                const double xmax,
                const double zmin,
                const double zmax,
                const std::string & label,
                const double gap = -1.,
                const double rot = 0.)
```

Add a pixel on an existing plane at constant y.

### AddPlanePhi()

```
void Garfield::ComponentAnalyticField::AddPlanePhi (
                const double phi,
                const double voltage,
                const std::string & label = "")
```

Add a plane at constant phi.

### AddPlaneR()

```
void Garfield::ComponentAnalyticField::AddPlaneR (
                const double r,
                const double voltage,
                const std::string & label = "")
```

Add a plane at constant radius.

### AddPlaneX()

```
void Garfield::ComponentAnalyticField::AddPlaneX (
                const double x,
                const double voltage,
                const std::string & label = "")
```

Add a plane at constant x.

AddPlaneY()

```
void Garfield::ComponentAnalyticField::AddPlaneY (
                const double y,
                const double voltage,
                const std::string & label = "")
```

Add a plane at constant y.

AddReadout()

```
void Garfield::ComponentAnalyticField::AddReadout (
                const std::string & label,
                const bool silent = false)
```

Request calculation of weighting field and potential for a given group of wires or planes.

AddStripOnPlanePhi()

```
void Garfield::ComponentAnalyticField::AddStripOnPlanePhi (
                const char direction,
                const double phi,
                const double smin,
                const double smax,
                const std::string & label,
                const double gap = -1.)
```

Add a strip in the r or z direction on an existing plane at constant phi.

AddStripOnPlaneR()

```
void Garfield::ComponentAnalyticField::AddStripOnPlaneR (
                const char direction,
                const double r,
                const double smin,
                const double smax,
                const std::string & label,
                const double gap = -1.)
```

Add a strip in the phi or z direction on an existing plane at constant radius.

AddStripOnPlaneX()

```
void Garfield::ComponentAnalyticField::AddStripOnPlaneX (
                const char direction,
                const double x,
                const double smin,
                const double smax,
                const std::string & label,
                const double gap = -1.)
```

Add a strip in the y or z direction on an existing plane at constant x.

Parameters

| | |
|---|---|
| direction | 'y' or 'z'. |
| x | coordinate of the plane. |
| smin | lower limit of the strip in y or z. |
| smax | upper limit of the strip in y or z. |
| label | weighting field identifier. |
| gap | distance to the opposite plane (optional). |

### AddStripOnPlaneY()

```
void Garfield::ComponentAnalyticField::AddStripOnPlaneY (
                const char direction,
                const double y,
                const double smin,
                const double smax,
                const std::string & label,
                const double gap = -1.)
```

Add a strip in the x or z direction on an existing plane at constant y.

### AddTube()

```
void Garfield::ComponentAnalyticField::AddTube (
                const double radius,
                const double voltage,
                const int nEdges,
                const std::string & label = "")
```

Add a tube.

### AddWire()

```
void Garfield::ComponentAnalyticField::AddWire (
                const double x,
                const double y,
                const double diameter,
                const double voltage,
                const std::string & label = "",
                const double length = 100.,
                const double tension = 50.,
                const double rho = 19.3,
                const int ntrap = 5)
```

Add a wire at (x, y) .

### CellCheck()

```
bool Garfield::ComponentAnalyticField::CellCheck ()   [private]
```

### CellInit()

void Garfield::ComponentAnalyticField::CellInit () [private]

### CellType()

bool Garfield::ComponentAnalyticField::CellType () [private]

### Charge()

bool Garfield::ComponentAnalyticField::Charge (
std::vector< std::vector< double > > & mat) [private]

### ClearCharges()

void Garfield::ComponentAnalyticField::ClearCharges ()

Remove all point charges.

### ConformalMap()

void Garfield::ComponentAnalyticField::ConformalMap (
const std::complex< double > & z,
std::complex< double > & ww,
std::complex< double > & wd) const [private]

### CrossedPlane()

bool Garfield::ComponentAnalyticField::CrossedPlane (
const double x0,
const double y0,
const double z0,
const double x1,
const double y1,
const double z1,
double & xc,
double & yc,
double & zc) [override]

### CrossedWire()

bool Garfield::ComponentAnalyticField::CrossedWire (
const double x0,
const double y0,
const double z0,
const double x1,
const double y1,
const double z1,
double & xc,
double & yc,
double & zc,
const bool centre,
double & rc) [override]

### DipoleFieldA00()

```
void Garfield::ComponentAnalyticField::DipoleFieldA00 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### DipoleFieldB1X()

```
void Garfield::ComponentAnalyticField::DipoleFieldB1X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### DipoleFieldB1Y()

```
void Garfield::ComponentAnalyticField::DipoleFieldB1Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### DipoleFieldB2X()

```
void Garfield::ComponentAnalyticField::DipoleFieldB2X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### DipoleFieldB2Y()

```
void Garfield::ComponentAnalyticField::DipoleFieldB2Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### E2Sum()

```
void Garfield::ComponentAnalyticField::E2Sum (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey) const    [private]
```

### ElectricField() [1/2]

```
void Garfield::ComponentAnalyticField::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)    [override]
```

### ElectricField() [2/2]

```
void Garfield::ComponentAnalyticField::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)    [override]
```

### ElectricFieldAtWire()

```
bool Garfield::ComponentAnalyticField::ElectricFieldAtWire (
                const unsigned int iw,
                double & ex,
                double & ey)
```

Calculate the electric field at a given wire position, as if the wire itself were not there, but with the presence of its mirror images.

### EnableChargeCheck()

```
void Garfield::ComponentAnalyticField::EnableChargeCheck (
                const bool on = true)    [inline]
```

Check the quality of the capacitance matrix inversion (default: off).

Definition at line 186 of file ComponentAnalyticField.hh.

```
00186 { m_chargeCheck = on; }
```

EnableDipoleTerms()

void Garfield::ComponentAnalyticField::EnableDipoleTerms (
const bool on = true)

Request dipole terms be included for each of the wires (default: off).

EnableElectrostaticForce()

void Garfield::ComponentAnalyticField::EnableElectrostaticForce (
const bool on = true) [inline]

Include the electrostatic force in the sag computation or not.

Definition at line 289 of file ComponentAnalyticField.hh.

```
00289                                                {
00290     m_useElectrostaticForce = on;
00291   }
```

EnableExtrapolation()

void Garfield::ComponentAnalyticField::EnableExtrapolation (
const bool on = true) [inline]

Switch on/off extrapolation of electrostatic forces beyond the scanning area (default: off).

Definition at line 280 of file ComponentAnalyticField.hh.

```
00280 { m_extrapolateForces = on; }
```

EnableGravity()

void Garfield::ComponentAnalyticField::EnableGravity (
const bool on = true) [inline]

Include gravity in the sag computation or not.

Definition at line 287 of file ComponentAnalyticField.hh.

```
00287 { m_useGravitationalForce = on; }
```

Field()

int Garfield::ComponentAnalyticField::Field (
const double xin,
const double yin,
const double zin,
double & ex,
double & ey,
double & ez,
double & volt,
const bool opt) [private]

### Field3dA00()

```
void Garfield::ComponentAnalyticField::Field3dA00 (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & volt) const   [private]
```

### Field3dB2X()

```
void Garfield::ComponentAnalyticField::Field3dB2X (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & volt) const   [private]
```

### Field3dB2Y()

```
void Garfield::ComponentAnalyticField::Field3dB2Y (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & volt) const   [private]
```

### Field3dD10()

```
void Garfield::ComponentAnalyticField::Field3dD10 (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & volt) const   [private]
```

### FieldA00()

```
void Garfield::ComponentAnalyticField::FieldA00 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### FieldAtWireA00()

```
void Garfield::ComponentAnalyticField::FieldAtWireA00 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

### FieldAtWireB1X()

```
void Garfield::ComponentAnalyticField::FieldAtWireB1X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

### FieldAtWireB1Y()

```
void Garfield::ComponentAnalyticField::FieldAtWireB1Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

### FieldAtWireB2X()

```
void Garfield::ComponentAnalyticField::FieldAtWireB2X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

### FieldAtWireB2Y()

```
void Garfield::ComponentAnalyticField::FieldAtWireB2Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

### FieldAtWireC10()

```
void Garfield::ComponentAnalyticField::FieldAtWireC10 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldAtWireC2X()

```
void Garfield::ComponentAnalyticField::FieldAtWireC2X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldAtWireC2Y()

```
void Garfield::ComponentAnalyticField::FieldAtWireC2Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldAtWireC30()

```
void Garfield::ComponentAnalyticField::FieldAtWireC30 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldAtWireD10()

```
void Garfield::ComponentAnalyticField::FieldAtWireD10 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldAtWireD20()

```
void Garfield::ComponentAnalyticField::FieldAtWireD20 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldAtWireD30()

```
void Garfield::ComponentAnalyticField::FieldAtWireD30 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                const std::vector< bool > & cnalso) const   [private]
```

FieldB1X()

```
void Garfield::ComponentAnalyticField::FieldB1X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldB1Y()

```
void Garfield::ComponentAnalyticField::FieldB1Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldB2X()

```
void Garfield::ComponentAnalyticField::FieldB2X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldB2Y()

```
void Garfield::ComponentAnalyticField::FieldB2Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldC10()

```
void Garfield::ComponentAnalyticField::FieldC10 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldC2X()

```
void Garfield::ComponentAnalyticField::FieldC2X (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldC2Y()

```
void Garfield::ComponentAnalyticField::FieldC2Y (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldC30()

```
void Garfield::ComponentAnalyticField::FieldC30 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldD10()

```
void Garfield::ComponentAnalyticField::FieldD10 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

FieldD20()

```
void Garfield::ComponentAnalyticField::FieldD20 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const   [private]
```

### FieldD30()

```
void Garfield::ComponentAnalyticField::FieldD30 (
                const double xpos,
                const double ypos,
                double & ex,
                double & ey,
                double & volt,
                const bool opt) const    [private]
```

### FindZeroes()

```
bool Garfield::ComponentAnalyticField::FindZeroes (
                const Wire & wire,
                const double h,
                std::vector< double > & x,
                const std::vector< double > & xMap,
                const std::vector< double > & yMap,
                const std::vector< std::vector< double > > & fxMap,
                const std::vector< std::vector< double > > & fyMap) const    [private]
```

### ForcesOnWire()

```
bool Garfield::ComponentAnalyticField::ForcesOnWire (
                const unsigned int iw,
                std::vector< double > & xMap,
                std::vector< double > & yMap,
                std::vector< std::vector< double > > & fxMap,
                std::vector< std::vector< double > > & fyMap)
```

Calculate a table of the forces acting on a wire.

Parameters

| | |
|---|---|
| iw | index of the wire |
| xMap | coordinates of the grid lines in x |
| yMap | coordinates of the grid lines in y |
| fxMap | x-components of the force at the grid points |
| fyMap | y-components of the force at the grid points |

### GetBoundingBox()

```
bool Garfield::ComponentAnalyticField::GetBoundingBox (
                double & x0,
                double & y0,
                double & z0,
                double & x1,
                double & y1,
                double & z1)    [override]
```

### GetCellType() [1/2]

std::string Garfield::ComponentAnalyticField::GetCellType () [inline]

Return the cell type.

Cells are classified according to the number and orientation of planes, the presence of periodicities and the location of the wires as one of the following types:

A non-periodic cells with at most 1 x- and 1 y-plane B1X x-periodic cells without x-planes and at most 1 y-plane B1Y y-periodic cells without y-planes and at most 1 x-plane B2X cells with 2 x-planes and at most 1 y-plane B2Y cells with 2 y-planes and at most 1 x-plane C1 doubly periodic cells without planes C2X doubly periodic cells with x-planes C2Y doubly periodic cells with y-planes C3 double periodic cells with x- and y-planes D1 round tubes without axial periodicity D2 round tubes with axial periodicity D3 polygonal tubes without axial periodicity

Definition at line 156 of file ComponentAnalyticField.hh.

```
00156                         {
00157     if (!m_cellset) {
00158       if (CellCheck()) CellType();
00159     }
00160     return GetCellType(m_cellType);
00161   }
```

### GetCellType() [2/2]

std::string Garfield::ComponentAnalyticField::GetCellType (
const Cell ) const [private]

### GetElementaryCell()

bool Garfield::ComponentAnalyticField::GetElementaryCell (
double & x0,
double & y0,
double & z0,
double & x1,
double & y1,
double & z1) [override]

### GetForceRatio()

bool Garfield::ComponentAnalyticField::GetForceRatio (
const Wire & wire,
const double coor,
const std::array< double, 2 > & bend,
const std::array< double, 2 > & dbend,
std::array< double, 2 > & f,
const std::vector< double > & xMap,
const std::vector< double > & yMap,
const std::vector< std::vector< double > > & fxMap,
const std::vector< std::vector< double > > & fyMap) const [private]

### GetGravity()

```
void Garfield::ComponentAnalyticField::GetGravity (
                double & dx,
                double & dy,
                double & dz) const
```

Get the gravity orientation.

### GetMedium()

```
Medium * Garfield::ComponentAnalyticField::GetMedium (
                const double x,
                const double y,
                const double z)   [override]
```

### GetNumberOfPlanesPhi()

```
unsigned int Garfield::ComponentAnalyticField::GetNumberOfPlanesPhi () const
```

Get the number of equipotential planes at constant phi.

### GetNumberOfPlanesR()

```
unsigned int Garfield::ComponentAnalyticField::GetNumberOfPlanesR () const
```

Get the number of equipotential planes at constant radius.

### GetNumberOfPlanesX()

```
unsigned int Garfield::ComponentAnalyticField::GetNumberOfPlanesX () const
```

Get the number of equipotential planes at constant x.

### GetNumberOfPlanesY()

```
unsigned int Garfield::ComponentAnalyticField::GetNumberOfPlanesY () const
```

Get the number of equipotential planes at constant y.

### GetNumberOfWires()

```
unsigned int Garfield::ComponentAnalyticField::GetNumberOfWires () const   [inline]
```

Get the number of wires.

Definition at line 189 of file ComponentAnalyticField.hh.

```
00189 { return m_nWires; }
```

GetPeriodicityPhi()

```
bool Garfield::ComponentAnalyticField::GetPeriodicityPhi (
                double & s)
```

Get the periodicity [degree] in phi.

GetPeriodicityX()

```
bool Garfield::ComponentAnalyticField::GetPeriodicityX (
                double & s)
```

Get the periodic length in the x-direction.

GetPeriodicityY()

```
bool Garfield::ComponentAnalyticField::GetPeriodicityY (
                double & s)
```

Get the periodic length in the y-direction.

GetPlanePhi()

```
bool Garfield::ComponentAnalyticField::GetPlanePhi (
                const unsigned int i,
                double & phi,
                double & voltage,
                std::string & label) const
```

Retrieve the parameters of a plane at constant phi.

GetPlaneR()

```
bool Garfield::ComponentAnalyticField::GetPlaneR (
                const unsigned int i,
                double & r,
                double & voltage,
                std::string & label) const
```

Retrieve the parameters of a plane at constant radius.

GetPlaneX()

```
bool Garfield::ComponentAnalyticField::GetPlaneX (
                const unsigned int i,
                double & x,
                double & voltage,
                std::string & label) const
```

Retrieve the parameters of a plane at constant x.

GetPlaneY()

```
bool Garfield::ComponentAnalyticField::GetPlaneY (
                const unsigned int i,
                double & y,
                double & voltage,
                std::string & label) const
```

Retrieve the parameters of a plane at constant y.

GetTube()

```
bool Garfield::ComponentAnalyticField::GetTube (
                double & r,
                double & voltage,
                int & nEdges,
                std::string & label) const
```

Retrieve the tube parameters.

GetVoltageRange()

```
bool Garfield::ComponentAnalyticField::GetVoltageRange (
                double & pmin,
                double & pmax)    [override]
```

GetWire()

```
bool Garfield::ComponentAnalyticField::GetWire (
                const unsigned int i,
                double & x,
                double & y,
                double & diameter,
                double & voltage,
                std::string & label,
                double & length,
                double & charge,
                int & ntrap) const
```

Retrieve the parameters of a wire.

InitialiseFitParameters()

```
void Garfield::ComponentAnalyticField::InitialiseFitParameters (
                const std::vector< std::string > & groups,
                std::vector< double > & vw0,
                std::array< double, 5 > & vp0,
                std::vector< double > & aFit,
                std::vector< std::vector< unsigned int > > & wiresInGroup,
                std::vector< std::vector< unsigned int > > & planesInGroup)    [private]
```

### InTrapRadius()

```
bool Garfield::ComponentAnalyticField::InTrapRadius (
                const double q0,
                const double x0,
                const double y0,
                const double z0,
                double & xw,
                double & yx,
                double & rw)   [override]
```

### InTube()

```
bool Garfield::ComponentAnalyticField::InTube (
                const double x0,
                const double y0,
                const double a,
                const int n)   [static], [private]
```

### IprA00()

```
bool Garfield::ComponentAnalyticField::IprA00 (
                const int mx,
                const int my,
                std::vector< std::vector< std::complex< double > > > & mat)   [private]
```

### IprB2X()

```
bool Garfield::ComponentAnalyticField::IprB2X (
                const int my,
                std::vector< std::vector< std::complex< double > > > & mat)   [private]
```

### IprB2Y()

```
bool Garfield::ComponentAnalyticField::IprB2Y (
                const int mx,
                std::vector< std::vector< std::complex< double > > > & mat)   [private]
```

### IprC2X()

```
bool Garfield::ComponentAnalyticField::IprC2X (
                std::vector< std::vector< std::complex< double > > > & mat)   [private]
```

### IprC2Y()

```
bool Garfield::ComponentAnalyticField::IprC2Y (
                std::vector< std::vector< std::complex< double > > > & mat)   [private]
```

### IprC30()

```
bool Garfield::ComponentAnalyticField::IprC30 (
                std::vector< std::vector< std::complex< double > > > & mat)    [private]
```

### IprD10()

```
bool Garfield::ComponentAnalyticField::IprD10 (
                std::vector< std::vector< std::complex< double > > > & mat)    [private]
```

### IprD30()

```
bool Garfield::ComponentAnalyticField::IprD30 (
                std::vector< std::vector< std::complex< double > > > & mat)    [private]
```

### IsPolar()

```
bool Garfield::ComponentAnalyticField::IsPolar () const    [inline]
```

Are polar coordinates being used?

Definition at line 122 of file ComponentAnalyticField.hh.

```
00122 { return m_polar; }
```

### MultipoleMoments()

```
bool Garfield::ComponentAnalyticField::MultipoleMoments (
                const unsigned int iw,
                const unsigned int order = 4,
                const bool print = false,
                const bool plot = false,
                const double rmult = 1.,
                const double eps = 1.e-4,
                const unsigned int nMaxIter = 20)
```

Calculate multipole moments for a given wire.

Parameters

| | |
|---|---|
| iw | Index of the wire. |
| order | Order of the highest multipole moment. |
| print | Print information about the fitting process. |
| plot | Plot the potential and multipole fit around the wire. |
| rmult | Distance in multiples of the wire radius at which the decomposition is to be carried out. |
| eps | Used in the fit for calculating the covariance matrix. |
| nMaxIter | Maximum number of iterations in the fit. |

OptimiseOnGrid()

```
bool Garfield::ComponentAnalyticField::OptimiseOnGrid (
                const std::vector< std::string > & groups,
                const std::string & field_function,
                const double target,
                const double x0,
                const double y0,
                const double x1,
                const double y1,
                const unsigned int nX = 10,
                const unsigned int nY = 10,
                const bool print = true)
```

Vary the potential of selected electrodes to match (a function of) the potential or field on an x-y (or r-phi) grid of points.

OptimiseOnTrack()

```
bool Garfield::ComponentAnalyticField::OptimiseOnTrack (
                const std::vector< std::string > & groups,
                const std::string & field_function,
                const double target,
                const double x0,
                const double y0,
                const double x1,
                const double y1,
                const unsigned int nP = 20,
                const bool print = true)
```

Vary the potential of selected electrodes to match (a function of) the potential or field along a given line to a target.

Parameters

| | |
|---|---|
| groups | Identifier of the electrodes for which to vary the potential. |
| field_function | A function of the coordinates (x, y or r, phi), the electrostatic field (ex, ey, e) and potential (v). |
| target | Value of the field function to be reproduced. |
| x0,y0 | Starting point of the line. |
| x1,y1 | End point of the line. |
| nP | Number of points on the line. |
| print | Flag to print out information during each fit cycle. |

OptimiseOnWires()

```
bool Garfield::ComponentAnalyticField::OptimiseOnWires (
                const std::vector< std::string > & groups,
                const std::string & field_function,
                const double target,
                const std::vector< unsigned int > & wires,
                const bool print = true)
```

Vary the potential of selected electrodes to match (a function of) the potential or field on the surfaces of a set of wires.

### Ph2()

double Garfield::ComponentAnalyticField::Ph2 (
const double xpos,
const double ypos) const [private]

### Ph2Lim()

double Garfield::ComponentAnalyticField::Ph2Lim (
const double radius) const [inline], [private]

Definition at line 778 of file ComponentAnalyticField.hh.

```
00778                                    {
00779     return -log(abs(m_zmult) * radius * (1. - 3. * m_p1 + 5. * m_p2));
00780   }
```

### PlotCell()

void Garfield::ComponentAnalyticField::PlotCell (
TPad * pad)

Make a plot of the cell layout.

### Prepare()

bool Garfield::ComponentAnalyticField::Prepare () [private]

### PrepareSignals()

bool Garfield::ComponentAnalyticField::PrepareSignals () [private]

### PrepareStrips()

bool Garfield::ComponentAnalyticField::PrepareStrips () [private]

### PrintCell()

void Garfield::ComponentAnalyticField::PrintCell ()

Print all available information on the cell.

### PrintCharges()

void Garfield::ComponentAnalyticField::PrintCharges () const

Print a list of the point charges.

### Reset()

void Garfield::ComponentAnalyticField::Reset () [inline], [override], [private]

Definition at line 561 of file ComponentAnalyticField.hh.

```
00561                       {
00562     CellInit();
00563     m_medium = nullptr;
00564   }
```

### SagDetailed()

bool Garfield::ComponentAnalyticField::SagDetailed (
const Wire & wire,
const std::vector< double > & xMap,
const std::vector< double > & yMap,
const std::vector< std::vector< double > > & fxMap,
const std::vector< std::vector< double > > & fyMap,
std::vector< double > & csag,
std::vector< double > & xsag,
std::vector< double > & ysag) const [private]

### SetCartesianCoordinates()

void Garfield::ComponentAnalyticField::SetCartesianCoordinates ()

Use Cartesian coordinates (default).

### SetGravity()

void Garfield::ComponentAnalyticField::SetGravity (
const double dx,
const double dy,
const double dz)

Set the gravity orientation.

### SetMedium()

void Garfield::ComponentAnalyticField::SetMedium (
Medium * medium) [inline]

Set the medium inside the cell.

Definition at line 29 of file ComponentAnalyticField.hh.

```
00029 { m_medium = medium; }
```

### SetNumberOfCellCopies()

void Garfield::ComponentAnalyticField::SetNumberOfCellCopies (
const unsigned int nfourier)

SetNumberOfShots()

void Garfield::ComponentAnalyticField::SetNumberOfShots (
const unsigned int n) [inline]

Set the number of shots used for numerically solving the wire sag differential equation.

Definition at line 319 of file ComponentAnalyticField.hh.
00319 { m_nShots = n; }

SetNumberOfSteps()

void Garfield::ComponentAnalyticField::SetNumberOfSteps (
const unsigned int n)

Set the number of integration steps within each shot (must be >= 1).

SetOptimisationParameters()

void Garfield::ComponentAnalyticField::SetOptimisationParameters (
const double dist = 1.,
const double eps = 1.e-4,
const unsigned int nMaxIter = 10)

Set the conditions at which to allow the iteration to stop.

Parameters

| | |
|---|---|
| dist | Maximum deviation among all points between target and field function. |
| eps | Relative change between iterations. |
| nMaxIter | Maximum number of iterations. |

SetPeriodicityPhi()

void Garfield::ComponentAnalyticField::SetPeriodicityPhi (
const double phi)

Set the periodicity [degree] in phi.

SetPeriodicityX()

void Garfield::ComponentAnalyticField::SetPeriodicityX (
const double s)

Set the periodic length [cm] in the x-direction.

SetPeriodicityY()

```
void Garfield::ComponentAnalyticField::SetPeriodicityY (
                const double s)
```

Set the periodic length [cm] in the y-direction.

SetPolarCoordinates()

```
void Garfield::ComponentAnalyticField::SetPolarCoordinates ()
```

Use polar coordinates.

SetScanningArea()

```
void Garfield::ComponentAnalyticField::SetScanningArea (
                const double xmin,
                const double xmax,
                const double ymin,
                const double ymax)
```

Specify explicitly the boundaries of the the scanning area (i.

e. the area in which the electrostatic force acting on a wire is computed).

SetScanningAreaFirstOrder()

```
void Garfield::ComponentAnalyticField::SetScanningAreaFirstOrder (
                const double scale = 2.)
```

Set the scanning area based on the zeroth-order estimates of the wire shift, enlarged by a scaling factor.

This is the default behaviour.

SetScanningAreaLargest()

```
void Garfield::ComponentAnalyticField::SetScanningAreaLargest ()  [inline]
```

Set the scanning area to the largest area around each wire which is free from other cell elements.

Definition at line 274 of file ComponentAnalyticField.hh.

```
00274 { m_scanRange = ScanningRange::Largest; }
```

SetScanningGrid()

```
void Garfield::ComponentAnalyticField::SetScanningGrid (
                const unsigned int nX,
                const unsigned int nY)
```

Set the number of grid lines at which the electrostatic force as function of the wire displacement is computed.

### Setup()

bool Garfield::ComponentAnalyticField::Setup () [private]

### SetupA00()

bool Garfield::ComponentAnalyticField::SetupA00 () [private]

### SetupB1X()

bool Garfield::ComponentAnalyticField::SetupB1X () [private]

### SetupB1Y()

bool Garfield::ComponentAnalyticField::SetupB1Y () [private]

### SetupB2X()

bool Garfield::ComponentAnalyticField::SetupB2X () [private]

### SetupB2Y()

bool Garfield::ComponentAnalyticField::SetupB2Y () [private]

### SetupC10()

bool Garfield::ComponentAnalyticField::SetupC10 () [private]

### SetupC2X()

bool Garfield::ComponentAnalyticField::SetupC2X () [private]

### SetupC2Y()

bool Garfield::ComponentAnalyticField::SetupC2Y () [private]

### SetupC30()

bool Garfield::ComponentAnalyticField::SetupC30 () [private]

### SetupD10()

bool Garfield::ComponentAnalyticField::SetupD10 () [private]

### SetupD20()

bool Garfield::ComponentAnalyticField::SetupD20 () [private]

### SetupD30()

bool Garfield::ComponentAnalyticField::SetupD30 () [private]

### SetupDipoleTerms()

bool Garfield::ComponentAnalyticField::SetupDipoleTerms () [private]

### SetupPlaneSignals()

bool Garfield::ComponentAnalyticField::SetupPlaneSignals () [private]

### SetupWireSignals()

bool Garfield::ComponentAnalyticField::SetupWireSignals () [private]

### SignalLayer()

```
size_t Garfield::ComponentAnalyticField::SignalLayer (
                const int mx,
                const int my) const    [private]
```

### StepRKN()

```
bool Garfield::ComponentAnalyticField::StepRKN (
                const Wire & wire,
                const double h,
                double & x,
                std::array< double, 2 > & y,
                std::array< double, 2 > & yp,
                const std::vector< double > & xMap,
                const std::vector< double > & yMap,
                const std::vector< std::vector< double > > & fxMap,
                const std::vector< std::vector< double > > & fyMap) const    [private]
```

### StepSizeHint()

double Garfield::ComponentAnalyticField::StepSizeHint () [override]

### Trace()

```
bool Garfield::ComponentAnalyticField::Trace (
                const Wire & wire,
                const double h,
                const std::vector< double > & xx,
                std::vector< double > & f,
                const std::vector< double > & xMap,
                const std::vector< double > & yMap,
                const std::vector< std::vector< double > > & fxMap,
                const std::vector< std::vector< double > > & fyMap) const    [private]
```

### Update()

```
bool Garfield::ComponentAnalyticField::Update (
                const std::vector< double > & vw,
                const std::array< double, 5 > & vp)    [private]
```

### UpdatePeriodicity()

```
void Garfield::ComponentAnalyticField::UpdatePeriodicity ()    [override], [private]
```

### WeightingField()

```
void Garfield::ComponentAnalyticField::WeightingField (
                const double x,
                const double y,
                const double z,
                double & wx,
                double & wy,
                double & wz,
                const std::string & label)    [inline], [override]
```

Definition at line 334 of file ComponentAnalyticField.hh.

```
00336                                                 {
00337     wx = wy = wz = 0.;
00338     if (!m_sigset) PrepareSignals();
00339     Wfield(x, y, z, wx, wy, wz, label);
00340   }
```

### WeightingPotential()

```
double Garfield::ComponentAnalyticField::WeightingPotential (
                const double x,
                const double y,
                const double z,
                const std::string & label)    [inline], [override]
```

Definition at line 341 of file ComponentAnalyticField.hh.

```
00342                                                 {
00343     if (!m_sigset) PrepareSignals();
00344     return Wpot(x, y, z, label);
00345   }
```

### Wfield()

```
bool Garfield::ComponentAnalyticField::Wfield (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                const std::string & label) const   [private]
```

### WfieldPixel()

```
void Garfield::ComponentAnalyticField::WfieldPixel (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                const int ip,
                const Pixel & pixel) const   [private]
```

### WfieldPlaneA00()

```
void Garfield::ComponentAnalyticField::WfieldPlaneA00 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const int mx,
                const int my,
                const std::vector< double > & qp) const   [private]
```

### WfieldPlaneB2X()

```
void Garfield::ComponentAnalyticField::WfieldPlaneB2X (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const int my,
                const std::vector< double > & qp) const   [private]
```

### WfieldPlaneB2Y()

```
void Garfield::ComponentAnalyticField::WfieldPlaneB2Y (
                const double x,
                const double ypos,
                double & ex,
                double & ey,
                const int mx,
                const std::vector< double > & qp) const   [private]
```

### WfieldPlaneC2X()

```
void Garfield::ComponentAnalyticField::WfieldPlaneC2X (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qp) const    [private]
```

### WfieldPlaneC2Y()

```
void Garfield::ComponentAnalyticField::WfieldPlaneC2Y (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qp) const    [private]
```

### WfieldPlaneC30()

```
void Garfield::ComponentAnalyticField::WfieldPlaneC30 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qp) const    [private]
```

### WfieldPlaneD10()

```
void Garfield::ComponentAnalyticField::WfieldPlaneD10 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qp) const    [private]
```

### WfieldPlaneD30()

```
void Garfield::ComponentAnalyticField::WfieldPlaneD30 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qp) const    [private]
```

### WfieldStrip()

```
void Garfield::ComponentAnalyticField::WfieldStrip (
                const double x,
                const double y,
                const double g,
                const double w,
                double & fx,
                double & fy) const    [private]
```

### WfieldStripXy()

```
void Garfield::ComponentAnalyticField::WfieldStripXy (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                const int ip,
                const Strip & strip) const   [private]
```

### WfieldStripZ()

```
void Garfield::ComponentAnalyticField::WfieldStripZ (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const int ip,
                const Strip & strip) const   [private]
```

### WfieldWireA00()

```
void Garfield::ComponentAnalyticField::WfieldWireA00 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const int mx,
                const int my,
                const std::vector< double > & qw) const   [private]
```

### WfieldWireB2X()

```
void Garfield::ComponentAnalyticField::WfieldWireB2X (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const int my,
                const std::vector< double > & qw) const   [private]
```

### WfieldWireB2Y()

```
void Garfield::ComponentAnalyticField::WfieldWireB2Y (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const int mx,
                const std::vector< double > & qw) const   [private]
```

### WfieldWireC2X()

```
void Garfield::ComponentAnalyticField::WfieldWireC2X (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qw) const    [private]
```

### WfieldWireC2Y()

```
void Garfield::ComponentAnalyticField::WfieldWireC2Y (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qw) const    [private]
```

### WfieldWireC30()

```
void Garfield::ComponentAnalyticField::WfieldWireC30 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qw) const    [private]
```

### WfieldWireD10()

```
void Garfield::ComponentAnalyticField::WfieldWireD10 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qw) const    [private]
```

### WfieldWireD30()

```
void Garfield::ComponentAnalyticField::WfieldWireD30 (
                const double x,
                const double y,
                double & ex,
                double & ey,
                const std::vector< double > & qw) const    [private]
```

### WireCheck()

```
bool Garfield::ComponentAnalyticField::WireCheck () const    [private]
```

### WireDisplacement()

```
bool Garfield::ComponentAnalyticField::WireDisplacement (
                const unsigned int iw,
                const bool detailed,
                std::vector< double > & csag,
                std::vector< double > & xsag,
                std::vector< double > & ysag,
                double & stretch,
                const bool print = true)
```

Compute the sag profile of a wire.

Parameters

| | |
|---|---|
| iw | index of the wire |
| detailed | flag to request a detailed calculation of the profile or only a fast one. |
| csag | coordinate along the wire. |
| xsag | x components of the sag profile. |
| ysag | y components of the sag profile. |
| stretch | relative elongation. |
| print | flag to print the calculation results or not. |

### Wpot()

```
double Garfield::ComponentAnalyticField::Wpot (
                const double x,
                const double y,
                const double z,
                const std::string & label) const    [private]
```

### WpotPixel()

```
double Garfield::ComponentAnalyticField::WpotPixel (
                const double x,
                const double y,
                const double z,
                const int ip,
                const Pixel & pixel) const    [private]
```

### WpotPlaneA00()

```
double Garfield::ComponentAnalyticField::WpotPlaneA00 (
                const double x,
                const double y,
                const int mx,
                const int my,
                const std::vector< double > & qp) const    [private]
```

### WpotPlaneB2X()

```
double Garfield::ComponentAnalyticField::WpotPlaneB2X (
                const double x,
                const double y,
                const int my,
                const std::vector< double > & qp) const   [private]
```

### WpotPlaneB2Y()

```
double Garfield::ComponentAnalyticField::WpotPlaneB2Y (
                const double x,
                const double y,
                const int mx,
                const std::vector< double > & qp) const   [private]
```

### WpotPlaneC2X()

```
double Garfield::ComponentAnalyticField::WpotPlaneC2X (
                const double x,
                const double y,
                const std::vector< double > & qp) const   [private]
```

### WpotPlaneC2Y()

```
double Garfield::ComponentAnalyticField::WpotPlaneC2Y (
                const double x,
                const double y,
                const std::vector< double > & qp) const   [private]
```

### WpotPlaneC30()

```
double Garfield::ComponentAnalyticField::WpotPlaneC30 (
                const double x,
                const double y,
                const std::vector< double > & qp) const   [private]
```

### WpotPlaneD10()

```
double Garfield::ComponentAnalyticField::WpotPlaneD10 (
                const double x,
                const double y,
                const std::vector< double > & qp) const   [private]
```

### WpotPlaneD30()

```
double Garfield::ComponentAnalyticField::WpotPlaneD30 (
                const double x,
                const double y,
                const std::vector< double > & qp) const   [private]
```

### WpotStripXy()

```
double Garfield::ComponentAnalyticField::WpotStripXy (
                const double x,
                const double y,
                const double z,
                const int ip,
                const Strip & strip) const   [private]
```

### WpotStripZ()

```
double Garfield::ComponentAnalyticField::WpotStripZ (
                const double x,
                const double y,
                const int ip,
                const Strip & strip) const   [private]
```

### WpotWireA00()

```
double Garfield::ComponentAnalyticField::WpotWireA00 (
                const double x,
                const double y,
                const int mx,
                const int my,
                const std::vector< double > & qw) const   [private]
```

### WpotWireB2X()

```
double Garfield::ComponentAnalyticField::WpotWireB2X (
                const double x,
                const double y,
                const int my,
                const std::vector< double > & qw) const   [private]
```

### WpotWireB2Y()

```
double Garfield::ComponentAnalyticField::WpotWireB2Y (
                const double x,
                const double y,
                const int mx,
                const std::vector< double > & qw) const   [private]
```

### WpotWireC2X()

```
double Garfield::ComponentAnalyticField::WpotWireC2X (
                const double x,
                const double y,
                const std::vector< double > & qw) const   [private]
```

WpotWireC2Y()

double Garfield::ComponentAnalyticField::WpotWireC2Y (
const double x,
const double y,
const std::vector< double > & qw) const [private]

WpotWireC30()

double Garfield::ComponentAnalyticField::WpotWireC30 (
const double x,
const double y,
const std::vector< double > & qw) const [private]

WpotWireD10()

double Garfield::ComponentAnalyticField::WpotWireD10 (
const double x,
const double y,
const std::vector< double > & qw) const [private]

WpotWireD30()

double Garfield::ComponentAnalyticField::WpotWireD30 (
const double x,
const double y,
const std::vector< double > & qw) const [private]

### 7.16.5 Member Data Documentation

m_amp2

std::vector<double> Garfield::ComponentAnalyticField::m_amp2 [private]

Definition at line 447 of file ComponentAnalyticField.hh.

m_b2sin

std::vector<double> Garfield::ComponentAnalyticField::m_b2sin [private]

Definition at line 450 of file ComponentAnalyticField.hh.

m_c1

double Garfield::ComponentAnalyticField::m_c1 [private]

Definition at line 454 of file ComponentAnalyticField.hh.

### m_cellset

bool Garfield::ComponentAnalyticField::m_cellset = false [private]

Definition at line 391 of file ComponentAnalyticField.hh.

### m_cellType

Cell Garfield::ComponentAnalyticField::m_cellType [private]

Definition at line 397 of file ComponentAnalyticField.hh.

### m_cellTypeFourier

Cell Garfield::ComponentAnalyticField::m_cellTypeFourier = A00 [private]

Definition at line 414 of file ComponentAnalyticField.hh.

### m_ch3d

std::vector<Charge3d> Garfield::ComponentAnalyticField::m_ch3d [private]

Definition at line 522 of file ComponentAnalyticField.hh.

### m_chargeCheck

bool Garfield::ComponentAnalyticField::m_chargeCheck = false [private]

Definition at line 389 of file ComponentAnalyticField.hh.

### m_coplan

double Garfield::ComponentAnalyticField::m_coplan[4] [private]

Definition at line 469 of file ComponentAnalyticField.hh.

### m_coplax

double Garfield::ComponentAnalyticField::m_coplax [private]

Definition at line 470 of file ComponentAnalyticField.hh.

### m_coplay

double Garfield::ComponentAnalyticField::m_coplay [private]

Definition at line 470 of file ComponentAnalyticField.hh.

### m_corvta

double Garfield::ComponentAnalyticField::m_corvta [private]

Definition at line 462 of file ComponentAnalyticField.hh.

### m_corvtb

double Garfield::ComponentAnalyticField::m_corvtb [private]

Definition at line 462 of file ComponentAnalyticField.hh.

### m_corvtc

double Garfield::ComponentAnalyticField::m_corvtc [private]

Definition at line 462 of file ComponentAnalyticField.hh.

### m_cosph2

std::vector<double> Garfield::ComponentAnalyticField::m_cosph2 [private]

Definition at line 445 of file ComponentAnalyticField.hh.

### m_cotube

double Garfield::ComponentAnalyticField::m_cotube = 1. [private]

Definition at line 505 of file ComponentAnalyticField.hh.

### m_cotube2

double Garfield::ComponentAnalyticField::m_cotube2 = 1. [private]

Definition at line 506 of file ComponentAnalyticField.hh.

### m_dipole

bool Garfield::ComponentAnalyticField::m_dipole = false [private]

Definition at line 443 of file ComponentAnalyticField.hh.

### m_dmin

double Garfield::ComponentAnalyticField::m_dmin = -1. [private]

Definition at line 510 of file ComponentAnalyticField.hh.

### m_down

std::array<double, 3> Garfield::ComponentAnalyticField::m_down {{0, 0, 1}} [private]

Definition at line 529 of file ComponentAnalyticField.hh.
00529 {{0, 0, 1}};

### m_extrapolateForces

bool Garfield::ComponentAnalyticField::m_extrapolateForces = false [private]

Definition at line 549 of file ComponentAnalyticField.hh.

### m_fperx

bool Garfield::ComponentAnalyticField::m_fperx = false [private]

Definition at line 415 of file ComponentAnalyticField.hh.

### m_fpery

bool Garfield::ComponentAnalyticField::m_fpery = false [private]

Definition at line 416 of file ComponentAnalyticField.hh.

### m_kappa

double Garfield::ComponentAnalyticField::m_kappa [private]

Definition at line 458 of file ComponentAnalyticField.hh.

### m_medium

Medium* Garfield::ComponentAnalyticField::m_medium = nullptr [private]

Definition at line 387 of file ComponentAnalyticField.hh.

### m_mfexp

int Garfield::ComponentAnalyticField::m_mfexp = 0 [private]

Definition at line 421 of file ComponentAnalyticField.hh.

### m_mode

int Garfield::ComponentAnalyticField::m_mode [private]

Definition at line 452 of file ComponentAnalyticField.hh.

### m_mtube

int Garfield::ComponentAnalyticField::m_mtube = 0 [private]

Definition at line 503 of file ComponentAnalyticField.hh.

### m_mutex

std::mutex Garfield::ComponentAnalyticField::m_mutex [private]

Definition at line 385 of file ComponentAnalyticField.hh.

### m_mxmax

int Garfield::ComponentAnalyticField::m_mxmax = 0 [private]

Definition at line 418 of file ComponentAnalyticField.hh.

### m_mxmin

int Garfield::ComponentAnalyticField::m_mxmin = 0 [private]

Definition at line 417 of file ComponentAnalyticField.hh.

### m_mymax

int Garfield::ComponentAnalyticField::m_mymax = 0 [private]

Definition at line 420 of file ComponentAnalyticField.hh.

### m_mymin

int Garfield::ComponentAnalyticField::m_mymin = 0 [private]

Definition at line 419 of file ComponentAnalyticField.hh.

### m_nFourier

int Garfield::ComponentAnalyticField::m_nFourier = 1 [private]

Definition at line 413 of file ComponentAnalyticField.hh.

### m_nScanX

unsigned int Garfield::ComponentAnalyticField::m_nScanX = 11 [private]

Definition at line 546 of file ComponentAnalyticField.hh.

### m_nScanY

unsigned int Garfield::ComponentAnalyticField::m_nScanY = 11 [private]

Definition at line 547 of file ComponentAnalyticField.hh.

### m_nShots

unsigned int Garfield::ComponentAnalyticField::m_nShots = 2 [private]

Definition at line 531 of file ComponentAnalyticField.hh.

### m_nSteps

unsigned int Garfield::ComponentAnalyticField::m_nSteps = 20 [private]

Definition at line 533 of file ComponentAnalyticField.hh.

### m_nTermBessel

unsigned int Garfield::ComponentAnalyticField::m_nTermBessel = 10 [private]

Definition at line 523 of file ComponentAnalyticField.hh.

### m_nTermPoly

unsigned int Garfield::ComponentAnalyticField::m_nTermPoly = 100 [private]

Definition at line 524 of file ComponentAnalyticField.hh.

### m_ntube

int Garfield::ComponentAnalyticField::m_ntube = 1 [private]

Definition at line 504 of file ComponentAnalyticField.hh.

### m_nWires

unsigned int Garfield::ComponentAnalyticField::m_nWires [private]

Definition at line 426 of file ComponentAnalyticField.hh.

### m_optDist

double Garfield::ComponentAnalyticField::m_optDist = 1. [private]

Definition at line 553 of file ComponentAnalyticField.hh.

### m_optEps

double Garfield::ComponentAnalyticField::m_optEps = 1.e-4 [private]

Definition at line 556 of file ComponentAnalyticField.hh.

### m_optNitmax

unsigned int Garfield::ComponentAnalyticField::m_optNitmax = 10 [private]

Definition at line 558 of file ComponentAnalyticField.hh.

### m_p1

double Garfield::ComponentAnalyticField::m_p1 [private]

Definition at line 454 of file ComponentAnalyticField.hh.

### m_p2

double Garfield::ComponentAnalyticField::m_p2 [private]

Definition at line 454 of file ComponentAnalyticField.hh.

### m_perx

bool Garfield::ComponentAnalyticField::m_perx = false [private]

Definition at line 408 of file ComponentAnalyticField.hh.

### m_pery

bool Garfield::ComponentAnalyticField::m_pery = false [private]

Definition at line 409 of file ComponentAnalyticField.hh.

### m_planes

std::array<Plane, 5> Garfield::ComponentAnalyticField::m_planes [private]

Definition at line 499 of file ComponentAnalyticField.hh.

### m_polar

bool Garfield::ComponentAnalyticField::m_polar = false [private]

Definition at line 394 of file ComponentAnalyticField.hh.

### m_qplane

std::vector<std::vector<std::vector<double> > > Garfield::ComponentAnalyticField::m_qplane [private]

Definition at line 515 of file ComponentAnalyticField.hh.

### m_qwire

std::vector<std::vector<std::vector<double> > > Garfield::ComponentAnalyticField::m_qwire [private]

Definition at line 513 of file ComponentAnalyticField.hh.

### m_readout

std::vector<std::string> Garfield::ComponentAnalyticField::m_readout [private]

Definition at line 423 of file ComponentAnalyticField.hh.

### m_scaleRange

double Garfield::ComponentAnalyticField::m_scaleRange = 2. [private]

Definition at line 544 of file ComponentAnalyticField.hh.

### m_scanRange

ScanningRange Garfield::ComponentAnalyticField::m_scanRange = ScanningRange::FirstOrder [private]

Definition at line 537 of file ComponentAnalyticField.hh.

### m_sigset

bool Garfield::ComponentAnalyticField::m_sigset = false [private]

Definition at line 392 of file ComponentAnalyticField.hh.

### m_sinph2

std::vector<double> Garfield::ComponentAnalyticField::m_sinph2 [private]

Definition at line 446 of file ComponentAnalyticField.hh.

### m_sx

double Garfield::ComponentAnalyticField::m_sx [private]

Definition at line 410 of file ComponentAnalyticField.hh.

### m_sy

double Garfield::ComponentAnalyticField::m_sy [private]

Definition at line 410 of file ComponentAnalyticField.hh.

### m_tube

bool Garfield::ComponentAnalyticField::m_tube = false [private]

Definition at line 502 of file ComponentAnalyticField.hh.

### m_useElectrostaticForce

bool Garfield::ComponentAnalyticField::m_useElectrostaticForce = true [private]

Definition at line 526 of file ComponentAnalyticField.hh.

### m_useGravitationalForce

bool Garfield::ComponentAnalyticField::m_useGravitationalForce = true [private]

Definition at line 527 of file ComponentAnalyticField.hh.

### m_v0

double Garfield::ComponentAnalyticField::m_v0 [private]

Definition at line 461 of file ComponentAnalyticField.hh.

### m_vmax

double Garfield::ComponentAnalyticField::m_vmax [private]

Definition at line 405 of file ComponentAnalyticField.hh.

### m_vmin

double Garfield::ComponentAnalyticField::m_vmin [private]

Definition at line 405 of file ComponentAnalyticField.hh.

### m_vtplan

double Garfield::ComponentAnalyticField::m_vtplan[4] [private]

Definition at line 472 of file ComponentAnalyticField.hh.

m_vttube

double Garfield::ComponentAnalyticField::m_vttube = 0. [private]

Definition at line 507 of file ComponentAnalyticField.hh.

m_w

std::vector<Wire> Garfield::ComponentAnalyticField::m_w [private]

Definition at line 440 of file ComponentAnalyticField.hh.

m_xmax

double Garfield::ComponentAnalyticField::m_xmax [private]

Definition at line 400 of file ComponentAnalyticField.hh.

m_xmin

double Garfield::ComponentAnalyticField::m_xmin [private]

Definition at line 400 of file ComponentAnalyticField.hh.

m_xScanMax

double Garfield::ComponentAnalyticField::m_xScanMax = 0. [private]

Definition at line 540 of file ComponentAnalyticField.hh.

m_xScanMin

double Garfield::ComponentAnalyticField::m_xScanMin = 0. [private]

Definition at line 539 of file ComponentAnalyticField.hh.

m_ymax

double Garfield::ComponentAnalyticField::m_ymax [private]

Definition at line 401 of file ComponentAnalyticField.hh.

m_ymin

double Garfield::ComponentAnalyticField::m_ymin [private]

Definition at line 401 of file ComponentAnalyticField.hh.

### m_ynplan

bool Garfield::ComponentAnalyticField::m_ynplan[4] [private]

Definition at line 466 of file ComponentAnalyticField.hh.

### m_ynplax

bool Garfield::ComponentAnalyticField::m_ynplax [private]

Definition at line 467 of file ComponentAnalyticField.hh.

### m_ynplay

bool Garfield::ComponentAnalyticField::m_ynplay [private]

Definition at line 467 of file ComponentAnalyticField.hh.

### m_yScanMax

double Garfield::ComponentAnalyticField::m_yScanMax = 0. [private]

Definition at line 542 of file ComponentAnalyticField.hh.

### m_yScanMin

double Garfield::ComponentAnalyticField::m_yScanMin = 0. [private]

Definition at line 541 of file ComponentAnalyticField.hh.

### m_zmax

double Garfield::ComponentAnalyticField::m_zmax [private]

Definition at line 402 of file ComponentAnalyticField.hh.

### m_zmin

double Garfield::ComponentAnalyticField::m_zmin [private]

Definition at line 402 of file ComponentAnalyticField.hh.

### m_zmult

std::complex<double> Garfield::ComponentAnalyticField::m_zmult [private]

Definition at line 453 of file ComponentAnalyticField.hh.

m_zw

std::vector<std::complex<double> > Garfield::ComponentAnalyticField::m_zw [private]

Definition at line 457 of file ComponentAnalyticField.hh.

The documentation for this class was generated from the following file:

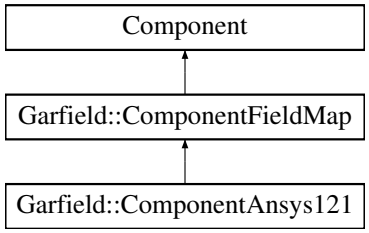
- /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh

## 7.17 Garfield::ComponentAnsys121 Class Reference

Component for importing and interpolating two-dimensional ANSYS field maps.

#include <ComponentAnsys121.hh>

Inheritance diagram for Garfield::ComponentAnsys121:




Public Member Functions

- ComponentAnsys121 ()
  Constructor.
- ~ComponentAnsys121 ()
  Destructor.
- bool Initialise (const std::string &elist="ELIST.lis", const std::string &nlist="NLIST.lis", const std::string &mplist="MPLIST.lis", const std::string &prnsol="PRNSOL.lis", const std::string &unit="cm")
  Import a field map.
- bool SetWeightingPotential (const std::string &prnsol, const std::string &label)
  Import weighting potentials.
- bool SetWeightingField (const std::string &prnsol, const std::string &label)
- void SetRangeZ (const double zmin, const double zmax)
  Set the limits of the active region along z.

Public Member Functions inherited from Garfield::ComponentFieldMap

- ComponentFieldMap ()=delete
  Default constructor.
- ComponentFieldMap (const std::string &name)
  Constructor.
- virtual ~ComponentFieldMap ()
  Destructor.
- bool Check ()
  Check element aspect ratio.
- void PrintRange ()
  Show x, y, z, V and angular ranges.
- void PrintMaterials ()
  List all currently defined materials.
- void DriftMedium (const size_t imat)
  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)
  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const
  Return the number of materials in the field map.
- double GetPermittivity (const size_t imat) const
  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const
  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)
  Associate a field map material with a Medium object.
- Medium * GetMedium (const size_t imat) const
  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)
  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const
  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const
  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)
  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)
  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)
  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override

- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Private Member Functions

- bool LoadPotentials (const std::string &prnsol, std::vector< double > &pot)

Additional Inherited Members

Protected Types inherited from Garfield::ComponentFieldMap

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Protected Member Functions inherited from Garfield::ComponentFieldMap

- void Reset () override
- void Prepare ()
- virtual void SetRange ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.

- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.
- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

  Move (ex, ey, ez) to global coordinates.
- virtual double GetElementVolume (const size_t i) const
- virtual void GetAspectRatio (const size_t i, double &dmin, double &dmax) const
- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.
- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Static Protected Member Functions inherited from Garfield::ComponentFieldMap

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)

  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a triangle.
- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)

  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a curved quadrilateral.
- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)

  Interpolate the potential in a curved quadratic tetrahedron.
- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

  Interpolate the field in a curved quadratic tetrahedron.
- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)
- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes inherited from Garfield::ComponentFieldMap

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

Flag to check if bounding boxes of elements are cached.

- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.
- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false
- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.17.1 Detailed Description

Component for importing and interpolating two-dimensional ANSYS field maps.

Definition at line 10 of file ComponentAnsys121.hh.

### 7.17.2 Constructor & Destructor Documentation

#### ComponentAnsys121()

Garfield::ComponentAnsys121::ComponentAnsys121 ()

Constructor.

#### ~ComponentAnsys121()

Garfield::ComponentAnsys121::~ComponentAnsys121 () [inline]

Destructor.

Definition at line 15 of file ComponentAnsys121.hh.

```
00015 {}
```

### 7.17.3 Member Function Documentation

#### Initialise()

```
bool Garfield::ComponentAnsys121::Initialise (
                const std::string & elist = "ELIST.lis",
                const std::string & nlist = "NLIST.lis",
                const std::string & mplist = "MPLIST.lis",
                const std::string & prnsol = "PRNSOL.lis",
                const std::string & unit = "cm")
```

Import a field map.

Parameters

| | |
|---|---|
| elist | name of the file containing the list of elements |
| nlist | name of the file containing the list of nodes |
| mplist | name of the file containing the list of materials |
| prnsol | name of the file containing the nodal solutions |
| unit | length unit |

### LoadPotentials()

```
bool Garfield::ComponentAnsys121::LoadPotentials (
                const std::string & prnsol,
                std::vector< double > & pot)    [private]
```

### SetRangeZ()

```
void Garfield::ComponentAnsys121::SetRangeZ (
                const double zmin,
                const double zmax)
```

Set the limits of the active region along z.

### SetWeightingField()

```
bool Garfield::ComponentAnsys121::SetWeightingField (
                const std::string & prnsol,
                const std::string & label)
```

### SetWeightingPotential()

```
bool Garfield::ComponentAnsys121::SetWeightingPotential (
                const std::string & prnsol,
                const std::string & label)    [inline]
```

Import weighting potentials.

Definition at line 30 of file ComponentAnsys121.hh.

```
00031                                                 {
00032     return SetWeightingField(prnsol, label);
00033   }
```

The documentation for this class was generated from the following file:

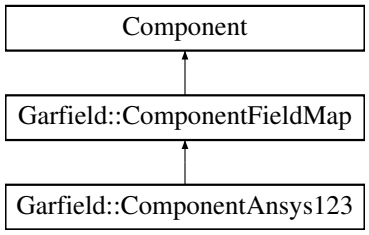
- /builds/garfield/docs/source/Include/Garfield/ComponentAnsys121.hh

## 7.18 Garfield::ComponentAnsys123 Class Reference

Component for importing and interpolating three-dimensional ANSYS field maps.

#include <ComponentAnsys123.hh>

Inheritance diagram for Garfield::ComponentAnsys123:




Public Member Functions

- ComponentAnsys123 ()
  Constructor.
- ~ComponentAnsys123 ()
  Destructor.
- bool Initialise (const std::string &elist="ELIST.lis", const std::string &nlist="NLIST.lis", const std::string &mplist="MPLIST.lis", const std::string &prnsol="PRNSOL.lis", const std::string &unit="cm")
  Import a field map.
- bool SetWeightingPotential (const std::string &prnsol, const std::string &label)
  Import weighting potentials.
- bool SetWeightingField (const std::string &prnsol, const std::string &label)
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override
  Create and initialise GPU Transfer class.

Public Member Functions inherited from Garfield::ComponentFieldMap

- ComponentFieldMap ()=delete
  Default constructor.
- ComponentFieldMap (const std::string &name)
  Constructor.
- virtual ~ComponentFieldMap ()
  Destructor.
- bool Check ()
  Check element aspect ratio.
- void PrintRange ()
  Show x, y, z, V and angular ranges.
- void PrintMaterials ()
  List all currently defined materials.
- void DriftMedium (const size_t imat)
  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)
  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const

Return the number of materials in the field map.

- double GetPermittivity (const size_t imat) const

  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const

  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)

  Associate a field map material with a Medium object.
- Medium * GetMedium (const size_t imat) const

  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)

  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const

  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const

  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)

  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)

  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)

  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Private Member Functions

- bool LoadPotentials (const std::string prnsol, std::vector< double > &pot)

Additional Inherited Members

Protected Types inherited from Garfield::ComponentFieldMap

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Protected Member Functions inherited from Garfield::ComponentFieldMap

- void Reset () override
- void Prepare ()
- virtual void SetRange ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.
- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.
- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

  Move (ex, ey, ez) to global coordinates.
- virtual double GetElementVolume (const size_t i) const
- virtual void GetAspectRatio (const size_t i, double &dmin, double &dmax) const
- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.

- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Static Protected Member Functions inherited from Garfield::ComponentFieldMap

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)

  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a triangle.
- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)

  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a curved quadrilateral.
- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)

  Interpolate the potential in a curved quadratic tetrahedron.
- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

  Interpolate the field in a curved quadratic tetrahedron.
- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)

- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes inherited from Garfield::ComponentFieldMap

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

  Flag to check if bounding boxes of elements are cached.
- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.
- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false
- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.18.1 Detailed Description

Component for importing and interpolating three-dimensional ANSYS field maps.

Definition at line 16 of file ComponentAnsys123.hh.

### 7.18.2 Constructor & Destructor Documentation

#### ComponentAnsys123()

Garfield::ComponentAnsys123::ComponentAnsys123 ()

Constructor.

#### ~ComponentAnsys123()

Garfield::ComponentAnsys123::~ComponentAnsys123 () [inline]

Destructor.

Definition at line 21 of file ComponentAnsys123.hh.

```
00021 {}
```

### 7.18.3 Member Function Documentation

#### CreateGPUTransferObject()

```
double Garfield::ComponentAnsys123::CreateGPUTransferObject (
            ComponentGPU *& comp_gpu) [override]
```

Create and initialise GPU Transfer class.

#### Initialise()

```
bool Garfield::ComponentAnsys123::Initialise (
            const std::string & elist = "ELIST.lis",
            const std::string & nlist = "NLIST.lis",
            const std::string & mplist = "MPLIST.lis",
            const std::string & prnsol = "PRNSOL.lis",
            const std::string & unit = "cm")
```

Import a field map.

Parameters

| | |
|---|---|
| elist | name of the file containing the list of elements |
| nlist | name of the file containing the list of nodes |
| mplist | name of the file containing the list of materials |
| prnsol | name of the file containing the nodal solutions |
| unit | length unit |

LoadPotentials()

bool Garfield::ComponentAnsys123::LoadPotentials (
const std::string prnsol,
std::vector< double > & pot) [private]

SetWeightingField()

bool Garfield::ComponentAnsys123::SetWeightingField (
const std::string & prnsol,
const std::string & label)

SetWeightingPotential()

bool Garfield::ComponentAnsys123::SetWeightingPotential (
const std::string & prnsol,
const std::string & label) [inline]

Import weighting potentials.

Definition at line 37 of file ComponentAnsys123.hh.

```
00038                                                 {
00039     return SetWeightingField(prnsol, label);
00040   }
```

The documentation for this class was generated from the following file:

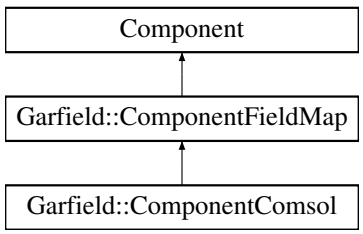
- /builds/garfield/docs/source/Include/Garfield/ComponentAnsys123.hh

## 7.19 Garfield::ComponentComsol Class Reference

Component for importing and interpolating Comsol field maps.

#include <ComponentComsol.hh>

Inheritance diagram for Garfield::ComponentComsol:




Classes

- struct Range

Public Member Functions

- ComponentComsol ()

  Default constructor.
- ComponentComsol (const std::string &mesh, const std::string &mplist, const std::string &field, const std::string &unit="m")

  Constructor from file names.
- ~ComponentComsol ()

  Destructor.
- void SetImportRange (const double xmin, const double xmax, const double ymin, const double ymax, const double zmin, const double zmax)
- bool Initialise (const std::string &header="mesh.mphtxt", const std::string &mplist="dielectrics.↩ dat", const std::string &field="field.txt", const std::string &unit="m")

  Import a field map.
- bool SetWeightingPotential (const std::string &file, const std::string &label)

  Import the weighting potential maps.
- bool SetWeightingField (const std::string &file, const std::string &label)
- bool SetDynamicWeightingPotential (const std::string &file, const std::string &label)

  Import the time-dependent weighting field maps.
- void SetTimeInterval (const double mint, const double maxt, const double stept)

  Set the time interval of the time-dependent weighting field.
- void GetTimeInterval (std::vector< double > &delayedTimes)

  Get times of time-sliced dynamic weighting field.
- void SetImportMaterial (const double epsr=1.)

  Select material that will exclusivaly be imported based on its relative permitivity.

Public Member Functions inherited from Garfield::ComponentFieldMap

- ComponentFieldMap ()=delete

  Default constructor.
- ComponentFieldMap (const std::string &name)

  Constructor.
- virtual ~ComponentFieldMap ()

  Destructor.
- bool Check ()

  Check element aspect ratio.
- void PrintRange ()

  Show x, y, z, V and angular ranges.
- void PrintMaterials ()

  List all currently defined materials.
- void DriftMedium (const size_t imat)

  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)

  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const

  Return the number of materials in the field map.
- double GetPermittivity (const size_t imat) const

  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const

  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)

Associate a field map material with a Medium object.

- Medium * GetMedium (const size_t imat) const

  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)

  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const

  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const

  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)

  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)

  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)

  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Private Member Functions

- bool GetTimeInterval (const std::string &file)
- bool CheckInRange (const double x, const double y, const double z) const
- bool ElementInRange (const Element &element, const std::vector< Node > &nodes) const
- bool LoadPotentials (const std::string &field, std::vector< double > &pot)

Private Attributes

- double m_unit = 100.
- bool m_timeset = false
- bool m_materialSelect = false
- double m_epsr = 1.
- Range m_range

Static Private Attributes

- static constexpr double MaxNodeDistance = 1.e-8

Additional Inherited Members

Protected Types inherited from Garfield::ComponentFieldMap

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Protected Member Functions inherited from Garfield::ComponentFieldMap

- void Reset () override
- void Prepare ()
- virtual void SetRange ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.
- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.
- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

  Move (ex, ey, ez) to global coordinates.
- virtual double GetElementVolume (const size_t i) const

- virtual void GetAspectRatio (const size_t i, double &dmin, double &dmax) const
- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.
- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Static Protected Member Functions inherited from Garfield::ComponentFieldMap

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)

  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a triangle.
- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)

  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)

Interpolate the field in a curved quadrilateral.

- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)

  Interpolate the potential in a curved quadratic tetrahedron.
- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

  Interpolate the field in a curved quadratic tetrahedron.
- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)
- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes inherited from Garfield::ComponentFieldMap

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

  Flag to check if bounding boxes of elements are cached.
- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.
- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false

- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.19.1 Detailed Description

Component for importing and interpolating Comsol field maps.

Definition at line 9 of file ComponentComsol.hh.

### 7.19.2 Constructor & Destructor Documentation

#### ComponentComsol() [1/2]

Garfield::ComponentComsol::ComponentComsol ()

Default constructor.

#### ComponentComsol() [2/2]

```
Garfield::ComponentComsol::ComponentComsol (
                const std::string & mesh,
                const std::string & mplist,
                const std::string & field,
                const std::string & unit = "m")
```

Constructor from file names.

#### ~ComponentComsol()

Garfield::ComponentComsol::~ComponentComsol ()  [inline]

Destructor.

Definition at line 17 of file ComponentComsol.hh.

```
00017 {}
```

### 7.19.3 Member Function Documentation

#### CheckInRange()

```
bool Garfield::ComponentComsol::CheckInRange (
                const double x,
                const double y,
                const double z) const   [inline], [private]
```

Definition at line 96 of file ComponentComsol.hh.

```
00096                                                   {
00097     if (!m_range.set) return true;
00098
00099     if (x < m_range.xmin || x > m_range.xmax || y < m_range.ymin ||
00100         y > m_range.ymax || z < m_range.zmin || z > m_range.zmax)
00101       return false;
00102
00103     return true;
00104   }
```

### ElementInRange()

```
bool Garfield::ComponentComsol::ElementInRange (
                const Element & element,
                const std::vector< Node > & nodes) const    [inline], [private]
```

Definition at line 106 of file ComponentComsol.hh.

```
00107                                                 {
00108     if (m_range.set) {
00109       for (size_t i = 0; i < 10; i++) {
00110         const Node &node = nodes[element.emap[i]];
00111         if (!CheckInRange(node.x, node.y, node.z)) return false;
00112       }
00113     }
00114     if (m_materialSelect && m_materials[element.matmap].eps != m_epsr)
00115       return false;
00116     return true;
00117   }
```

### GetTimeInterval() [1/2]

```
bool Garfield::ComponentComsol::GetTimeInterval (
                const std::string & file)    [private]
```

### GetTimeInterval() [2/2]

```
void Garfield::ComponentComsol::GetTimeInterval (
                std::vector< double > & delayedTimes)    [inline]
```

Get times of time-sliced dynamic weighting field.

Definition at line 55 of file ComponentComsol.hh.

```
00055                                                 {
00056     delayedTimes = m_wdtimes;
00057   }
```

### Initialise()

```
bool Garfield::ComponentComsol::Initialise (
                const std::string & header = "mesh.mphtxt",
                const std::string & mplist = "dielectrics.dat",
                const std::string & field = "field.txt",
                const std::string & unit = "m")
```

Import a field map.

Parameters

| | |
|---|---|
| header | name of the file containing the list of nodes |
| mplist | name of the file containing the material properties |
| field | name of the file containing the potentials at the nodes |
| unit | length unit |

### LoadPotentials()

```
bool Garfield::ComponentComsol::LoadPotentials (
                const std::string & field,
                std::vector< double > & pot)   [private]
```

### SetDynamicWeightingPotential()

```
bool Garfield::ComponentComsol::SetDynamicWeightingPotential (
                const std::string & file,
                const std::string & label)
```

Import the time-dependent weighting field maps.

### SetImportMaterial()

```
void Garfield::ComponentComsol::SetImportMaterial (
                const double epsr = 1.)   [inline]
```

Select material that will exclusivaly be imported based on its relative permitivity.

Definition at line 61 of file ComponentComsol.hh.

```
00061                                       {
00062     m_materialSelect = true;
00063     m_epsr = epsr;
00064   }
```

### SetImportRange()

```
void Garfield::ComponentComsol::SetImportRange (
                const double xmin,
                const double xmax,
                const double ymin,
                const double ymax,
                const double zmin,
                const double zmax)   [inline]
```

Definition at line 19 of file ComponentComsol.hh.

```
00020                                                                   {
00021     // TODO: Must happen before initialise function
00022     m_range.set = true;
00023     m_range.xmin = xmin;
00024     m_range.ymin = ymin;
00025     m_range.zmin = zmin;
00026 
00027     m_range.xmax = xmax;
00028     m_range.ymax = ymax;
00029     m_range.zmax = zmax;
00030   }
```

### SetTimeInterval()

```
void Garfield::ComponentComsol::SetTimeInterval (
                const double mint,
                const double maxt,
                const double stept)
```

Set the time interval of the time-dependent weighting field.

SetWeightingField()

```
bool Garfield::ComponentComsol::SetWeightingField (
                const std::string & file,
                const std::string & label)   [inline]
```

Definition at line 45 of file ComponentComsol.hh.

```
00045                                                 {
00046     return SetWeightingPotential(file, label);
00047   }
```

SetWeightingPotential()

```
bool Garfield::ComponentComsol::SetWeightingPotential (
                const std::string & file,
                const std::string & label)
```

Import the weighting potential maps.

### 7.19.4 Member Data Documentation

m_epsr

```
double Garfield::ComponentComsol::m_epsr = 1.   [private]
```

Definition at line 77 of file ComponentComsol.hh.

m_materialSelect

```
bool Garfield::ComponentComsol::m_materialSelect = false   [private]
```

Definition at line 76 of file ComponentComsol.hh.

m_range

```
Range Garfield::ComponentComsol::m_range   [private]
```

Definition at line 94 of file ComponentComsol.hh.

m_timeset

```
bool Garfield::ComponentComsol::m_timeset = false   [private]
```

Definition at line 73 of file ComponentComsol.hh.

m_unit

```
double Garfield::ComponentComsol::m_unit = 100.   [private]
```

Definition at line 72 of file ComponentComsol.hh.

MaxNodeDistance

double Garfield::ComponentComsol::MaxNodeDistance = 1.e-8 [static], [constexpr], [private]

Definition at line 74 of file ComponentComsol.hh.

The documentation for this class was generated from the following file:

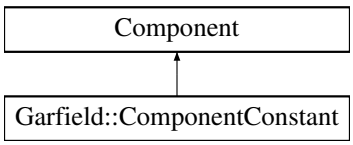
- /builds/garfield/docs/source/Include/Garfield/ComponentComsol.hh

## 7.20 Garfield::ComponentConstant Class Reference

Component with constant electric field.

#include <ComponentConstant.hh>

Inheritance diagram for Garfield::ComponentConstant:




Public Member Functions

- ComponentConstant ()
  Constructor.
- ~ComponentConstant ()
  Destructor.
- void SetElectricField (const double ex, const double ey, const double ez)
  Set the components of the electric field [V / cm].
- void SetPotential (const double x, const double y, const double z, const double v=0.)
  Specify the potential at a given point.
- void SetWeightingField (const double wx, const double wy, const double wz, const std::string label)
  Set the components of the weighting field [1 / cm].
- void SetWeightingPotential (const double x, const double y, const double z, const double v=0.)
  Specify the weighting potential at a given point.
- void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)
  Set the limits of the active area explicitly (instead of using a Geometry object).
- void UnsetArea ()
  Remove the explicit limits of the active area.
- void SetMedium (Medium *medium)
  Set the medium in the active area.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override

Private Member Functions

- void Reset () override
- void UpdatePeriodicity () override
- bool InArea (const double x, const double y, const double z)

Private Attributes

- std::array< double, 3 > m_efield = {{0., 0., 0.}}
- bool m_hasPotential = false
- double m_x0 = 0.
- double m_y0 = 0.
- double m_z0 = 0.
- double m_v0 = 0.
- bool m_hasWeightingField = false
- std::string m_label = ""
- std::array< double, 3 > m_wfield = {{0., 0., 0.}}
- bool m_hasWeightingPotential = false
- double m_wx0 = 0.
- double m_wy0 = 0.
- double m_wz0 = 0.
- double m_w0 = 0.
- std::array< double, 3 > m_xmin = {{0., 0., 0.}}
- std::array< double, 3 > m_xmax = {{0., 0., 0.}}
- bool m_hasArea = false
- Medium * m_medium = nullptr

### 7.20.1 Detailed Description

Component with constant electric field.

Definition at line 13 of file ComponentConstant.hh.

### 7.20.2 Constructor & Destructor Documentation

ComponentConstant()

Garfield::ComponentConstant::ComponentConstant ()

Constructor.

~ComponentConstant()

Garfield::ComponentConstant::~ComponentConstant () [inline]

Destructor.

Definition at line 18 of file ComponentConstant.hh.

```
00018 {}
```

### 7.20.3 Member Function Documentation

#### ElectricField() [1/2]

```
void Garfield::ComponentConstant::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)    [override]
```

#### ElectricField() [2/2]

```
void Garfield::ComponentConstant::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)    [override]
```

#### GetBoundingBox()

```
bool Garfield::ComponentConstant::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

#### GetMedium()

```
Medium * Garfield::ComponentConstant::GetMedium (
                const double x,
                const double y,
                const double z)    [override]
```

#### GetVoltageRange()

```
bool Garfield::ComponentConstant::GetVoltageRange (
                double & vmin,
                double & vmax)    [override]
```

### InArea()

```
bool Garfield::ComponentConstant::InArea (
                const double x,
                const double y,
                const double z)   [inline], [private]
```

Definition at line 94 of file ComponentConstant.hh.

```
00094                                                 {
00095     if (x < m_xmin[0] || x > m_xmax[0] || y < m_xmin[1] || y > m_xmax[1] ||
00096         z < m_xmin[2] || z > m_xmax[2]) {
00097       return false;
00098     }
00099     return true;
00100   }
```

### Reset()

```
void Garfield::ComponentConstant::Reset ()   [override], [private]
```

### SetArea()

```
void Garfield::ComponentConstant::SetArea (
                const double xmin,
                const double ymin,
                const double zmin,
                const double xmax,
                const double ymax,
                const double zmax)
```

Set the limits of the active area explicitly (instead of using a Geometry object).

### SetElectricField()

```
void Garfield::ComponentConstant::SetElectricField (
                const double ex,
                const double ey,
                const double ez)
```

Set the components of the electric field [V / cm].

### SetMedium()

```
void Garfield::ComponentConstant::SetMedium (
                Medium * medium)   [inline]
```

Set the medium in the active area.

Definition at line 40 of file ComponentConstant.hh.

```
00040 { m_medium = medium; }
```

SetPotential()

```
void Garfield::ComponentConstant::SetPotential (
                const double x,
                const double y,
                const double z,
                const double v = 0.)
```

Specify the potential at a given point.

SetWeightingField()

```
void Garfield::ComponentConstant::SetWeightingField (
                const double wx,
                const double wy,
                const double wz,
                const std::string label)
```

Set the components of the weighting field [1 / cm].

SetWeightingPotential()

```
void Garfield::ComponentConstant::SetWeightingPotential (
                const double x,
                const double y,
                const double z,
                const double v = 0.)
```

Specify the weighting potential at a given point.

UnsetArea()

```
void Garfield::ComponentConstant::UnsetArea ()
```

Remove the explicit limits of the active area.

UpdatePeriodicity()

```
void Garfield::ComponentConstant::UpdatePeriodicity ()   [override], [private]
```

WeightingField()

```
void Garfield::ComponentConstant::WeightingField (
                const double x,
                const double y,
                const double z,
                double & wx,
                double & wy,
                double & wz,
                const std::string & label)   [override]
```

WeightingPotential()

```
double Garfield::ComponentConstant::WeightingPotential (
                const double x,
                const double y,
                const double z,
                const std::string & label)   [override]
```

### 7.20.4 Member Data Documentation

m_efield

```
std::array<double, 3> Garfield::ComponentConstant::m_efield = {{0., 0., 0.}}   [private]
```

Definition at line 61 of file ComponentConstant.hh.

```
00061 {{0., 0., 0.}};
```

m_hasArea

```
bool Garfield::ComponentConstant::m_hasArea = false   [private]
```

Definition at line 87 of file ComponentConstant.hh.

m_hasPotential

```
bool Garfield::ComponentConstant::m_hasPotential = false   [private]
```

Definition at line 64 of file ComponentConstant.hh.

m_hasWeightingField

```
bool Garfield::ComponentConstant::m_hasWeightingField = false   [private]
```

Definition at line 71 of file ComponentConstant.hh.

m_hasWeightingPotential

```
bool Garfield::ComponentConstant::m_hasWeightingPotential = false   [private]
```

Definition at line 77 of file ComponentConstant.hh.

m_label

```
std::string Garfield::ComponentConstant::m_label = ""   [private]
```

Definition at line 73 of file ComponentConstant.hh.

### m_medium

Medium* Garfield::ComponentConstant::m_medium = nullptr [private]

Definition at line 89 of file ComponentConstant.hh.

### m_v0

double Garfield::ComponentConstant::m_v0 = 0. [private]

Definition at line 68 of file ComponentConstant.hh.

### m_w0

double Garfield::ComponentConstant::m_w0 = 0. [private]

Definition at line 81 of file ComponentConstant.hh.

### m_wfield

std::array<double, 3> Garfield::ComponentConstant::m_wfield = {{0., 0., 0.}} [private]

Definition at line 75 of file ComponentConstant.hh.
00075 {{0., 0., 0.}};

### m_wx0

double Garfield::ComponentConstant::m_wx0 = 0. [private]

Definition at line 79 of file ComponentConstant.hh.

### m_wy0

double Garfield::ComponentConstant::m_wy0 = 0. [private]

Definition at line 79 of file ComponentConstant.hh.

### m_wz0

double Garfield::ComponentConstant::m_wz0 = 0. [private]

Definition at line 79 of file ComponentConstant.hh.

### m_x0

double Garfield::ComponentConstant::m_x0 = 0. [private]

Definition at line 66 of file ComponentConstant.hh.

m_xmax

std::array<double, 3> Garfield::ComponentConstant::m_xmax = {{0., 0., 0.}} [private]

Definition at line 85 of file ComponentConstant.hh.
00085 {{0., 0., 0.}};

m_xmin

std::array<double, 3> Garfield::ComponentConstant::m_xmin = {{0., 0., 0.}} [private]

Definition at line 84 of file ComponentConstant.hh.
00084 {{0., 0., 0.}};

m_y0

double Garfield::ComponentConstant::m_y0 = 0. [private]

Definition at line 66 of file ComponentConstant.hh.

m_z0

double Garfield::ComponentConstant::m_z0 = 0. [private]

Definition at line 66 of file ComponentConstant.hh.

The documentation for this class was generated from the following file:

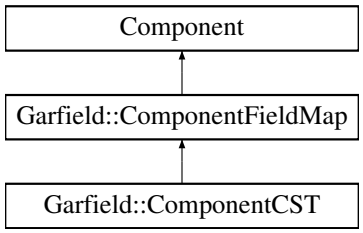
- /builds/garfield/docs/source/Include/Garfield/ComponentConstant.hh

## 7.21 Garfield::ComponentCST Class Reference

Component for importing and interpolating field maps from CST.

#include <ComponentCST.hh>

Inheritance diagram for Garfield::ComponentCST:



Public Member Functions

- ComponentCST ()

  Constructor.
- ~ComponentCST ()

  Destructor.
- void ShiftComponent (const double xShift, const double yShift, const double zShift)
- void GetNumberOfMeshLines (unsigned int &nx, unsigned int &ny, unsigned int &nz) const
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- void GetElementBoundaries (unsigned int element, double &xmin, double &xmax, double &ymin, double &ymax, double &zmin, double &zmax) const
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- bool Initialise (std::string elist, std::string nlist, std::string mplist, std::string prnsol, std::string unit="cm")
- bool Initialise (std::string dataFile, std::string unit="cm")

  Import of field data based on binary files.
- bool SetWeightingField (std::string prnsol, std::string label, bool isBinary=true)

  Initialise a weighting field.
- void SetRangeZ (const double zmin, const double zmax)
- void DisableXField ()

  Use these functions to disable a certain field component.
- void DisableYField ()
- void DisableZField ()
- void EnableShaping ()

  If you calculate the electric field component in $x$ direction along a line in x direction this field component will be constant inside mesh elements (by construction).
- void DisableShaping ()
- int Index2Element (const unsigned int i, const unsigned int j, const unsigned int k) const

  Calculate the element index from the position in the x/y/z position vectors (m_xlines, m_ylines, m_zlines).
- bool Coordinate2Index (const double x, const double y, const double z, unsigned int &i, unsigned int &j, unsigned int &k) const

  Find the positions in the x/y/z position vectors (m_xlines, m_ylines, m_zlines) for a given point.

Public Member Functions inherited from Garfield::ComponentFieldMap

- ComponentFieldMap ()=delete

  Default constructor.
- ComponentFieldMap (const std::string &name)

  Constructor.
- virtual ~ComponentFieldMap ()

Destructor.

- bool Check ()

  Check element aspect ratio.
- void PrintRange ()

  Show x, y, z, V and angular ranges.
- void PrintMaterials ()

  List all currently defined materials.
- void DriftMedium (const size_t imat)

  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)

  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const

  Return the number of materials in the field map.
- double GetPermittivity (const size_t imat) const

  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const

  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)

  Associate a field map material with a Medium object.
- Medium * GetMedium (const size_t imat) const

  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)

  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const

  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const

  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)

  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)

  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)

  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override

- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Protected Member Functions

- void SetRange () override
- double GetElementVolume (const size_t i) const override
- void GetAspectRatio (const size_t i, double &dmin, double &dmax) const override
- bool Coordinate2Index (const double x, const double y, const double z, unsigned int &i, unsigned int &j, unsigned int &k, double *position_mapped, bool *mirrored) const

  Calculate the index in the vectors m_xlines, m_ylines, m_zlines, which is before the given coordinate.

Protected Member Functions inherited from Garfield::ComponentFieldMap

- void Reset () override
- void Prepare ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.
- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.
- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

Move (ex, ey, ez) to global coordinates.

- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.
- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Private Member Functions

- void ElectricFieldBinary (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status, const bool calculatePotential=false) const
- float GetFieldComponent (const unsigned int i, const unsigned int j, const unsigned int k, const double rx, const double ry, const double rz, const char component, const std::vector< float > &potentials) const
- float GetPotential (const unsigned int i, const unsigned int j, const unsigned int k, const double rx, const double ry, const double rz, const std::vector< float > &potentials) const
- void ShapeField (float &ex, float &ey, float &ez, const double rx, const double ry, const double rz, const unsigned int i, const unsigned int j, const unsigned int k, const std::vector< float > &potentials) const
- void Element2Index (const size_t element, unsigned int &i, unsigned int &j, unsigned int &k) const
- int Index2Node (const unsigned int i, const unsigned int j, const unsigned int k) const
- void Node2Index (const size_t node, unsigned int &i, unsigned int &j, unsigned int &k) const

Private Attributes

- std::vector< double > m_xlines
  x positions used in the CST mesh
- std::vector< double > m_ylines
  y positions used in the CST mesh
- std::vector< double > m_zlines
  z positions used in the CST mesh
- std::vector< float > m_potential
  Potentials resulting from the CST simulation.
- std::map< std::string, std::vector< float > > m_weightingFields
  Map of weighting field potentials.
- std::vector< unsigned char > m_elementMaterial
  Material id for each element (unsigned char since it uses only 1 byte)
- unsigned int m_nx = 0
  Number of mesh lines in x direction.
- unsigned int m_ny = 0
  Number of mesh lines in y direction.
- unsigned int m_nz = 0
  Number of mesh lines in z direction.
- size_t m_nElements = 0
  Number of elements.
- size_t m_nNodes = 0
  Number of nodes.
- bool disableFieldComponent [3] = {false, false, false}
- bool doShaping = false

Additional Inherited Members

Protected Types inherited from Garfield::ComponentFieldMap

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Static Protected Member Functions inherited from Garfield::ComponentFieldMap

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)
  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)
  Interpolate the field in a triangle.
- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)
  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)
  Interpolate the field in a curved quadrilateral.
- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)
  Interpolate the potential in a curved quadratic tetrahedron.
- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

Interpolate the field in a curved quadratic tetrahedron.

- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)
- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes inherited from Garfield::ComponentFieldMap

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

  Flag to check if bounding boxes of elements are cached.
- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.
- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false
- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.21.1 Detailed Description

Component for importing and interpolating field maps from CST.

This interface assumes a certain format of the ascii files Please find the tools to extract the field data from CST in the correct way here: http://www.desy.de/~zenker/garfieldpp.html

Definition at line 16 of file ComponentCST.hh.

### 7.21.2 Constructor & Destructor Documentation

#### ComponentCST()

Garfield::ComponentCST::ComponentCST ()

Constructor.

#### ~ComponentCST()

Garfield::ComponentCST::~ComponentCST () [inline]

Destructor.

Definition at line 21 of file ComponentCST.hh.

```
00021 {}
```

### 7.21.3 Member Function Documentation

#### Coordinate2Index() [1/2]

```
bool Garfield::ComponentCST::Coordinate2Index (
                const double x,
                const double y,
                const double z,
                unsigned int & i,
                unsigned int & j,
                unsigned int & k) const
```

Find the positions in the x/y/z position vectors (m_xlines, m_ylines, m_zlines) for a given point.

The operator used for the comparison is <=. Therefore, the last entry in the vector will never be returned for a point inside the mesh.

#### Coordinate2Index() [2/2]

```
bool Garfield::ComponentCST::Coordinate2Index (
                const double x,
                const double y,
                const double z,
                unsigned int & i,
                unsigned int & j,
                unsigned int & k,
                double * position_mapped,
                bool * mirrored) const   [protected]
```

Calculate the index in the vectors m_xlines, m_ylines, m_zlines, which is before the given coordinate.

Remarks

x, y, z need to be mapped before using this function!

Parameters

| | |
|---|---|
| x | The x coordinate mapped to the basic cell. |
| y | The y coordinate mapped to the basic cell. |
| z | The z coordinate mapped to the basic cell. |
| i | The index of the m_xlines vector, where m_xlines.at(i) < x < m_xlines.at(i+1). |
| j | The index of the m_ylines vector, where m_ylines.at(j) < y < m_ylines.at(j+1). |
| k | The index of the m_zlines vector, where m_zlines.at(k) < z < m_zlines.at(k+1). |
| position_mapped | The calculated mapped position (x,y,z) -> (x_mapped, y_mapped, z_mapped) |
| mirrored | Information if x, y, or z direction is mirrored. |

### DisableShaping()

void Garfield::ComponentCST::DisableShaping () [inline]

Definition at line 186 of file ComponentCST.hh.

```
00186 { doShaping = false; }
```

### DisableXField()

void Garfield::ComponentCST::DisableXField () [inline]

Use these functions to disable a certain field component.

Is a field component is disabled ElectricField and WeightingField will return 0 for this component. This is useful if you want to have calculated global field distortions and you want to add the field of a GEM. If you would simply add both components the field component in drift direction would be added twice!

Definition at line 139 of file ComponentCST.hh.

```
00139 { disableFieldComponent[0] = true; }
```

### DisableYField()

void Garfield::ComponentCST::DisableYField () [inline]

Definition at line 140 of file ComponentCST.hh.

```
00140 { disableFieldComponent[1] = true; }
```

### DisableZField()

void Garfield::ComponentCST::DisableZField () [inline]

Definition at line 141 of file ComponentCST.hh.

```
00141 { disableFieldComponent[2] = true; }
```

ElectricField() [1/2]

```
void Garfield::ComponentCST::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)   [override]
```

ElectricField() [2/2]

```
void Garfield::ComponentCST::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)   [override]
```

ElectricFieldBinary()

```
void Garfield::ComponentCST::ElectricFieldBinary (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status,
                const bool calculatePotential = false) const   [private]
```

Element2Index()

```
void Garfield::ComponentCST::Element2Index (
                const size_t element,
                unsigned int & i,
                unsigned int & j,
                unsigned int & k) const   [private]
```

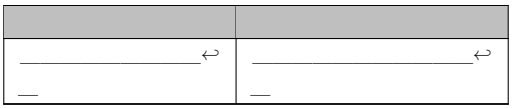
EnableShaping()

```
void Garfield::ComponentCST::EnableShaping ()   [inline]
```

If you calculate the electric field component in $x$ direction along a line in x direction this field component will be constant inside mesh elements (by construction).

This can be observed by plotting $E_x$ in $x$ direction. If you plot $E_x$ in y direction the field will be smooth (also by construction). Yuri Piadyk proposed also to shape the electric field. This is done as follows. The field component calculated as described above is assumed to appear in the center of the mesh element.

| M1 P | M2 | x direction |-—x—x-|--—x---—|-->



element 1 element 2
Lets consider only the $x$ direction and we want to calculate $E_x(P)$. The field in the center of the element containing $P$ is $E_x(M_1) = E_1$. Without shaping it is $E_1$ along the $x$ direction in everywhere in element 1. The idea of the shaping is to do a linear interpolation of the $E_x$ between the field $E_1$ and $E_x(M_2) = E_2$. This results in a smooth electric field $E_x$ also in $x$ direction. If P would be left from $M_1$ the field in the left neighboring element would be considered. In addition it is also checked if the material in both elements used for the interpolation is the same. Else no interpolation is done.

Remarks

This shaping gives you a nice and smooth field, but you introduce additional information. This information is not coming from the CST simulation, but from the assumption that the field between elements changes in a linear way, which might be wrong! So you might consider to increase the number of mesh cells used in the simulation rather than using this smoothing.

Definition at line 185 of file ComponentCST.hh.
00185 { doShaping = true; }

### GetAspectRatio()

void Garfield::ComponentCST::GetAspectRatio (
const size_t i,
double & dmin,
double & dmax) const [override], [protected], [virtual]
Reimplemented from Garfield::ComponentFieldMap.

### GetElementBoundaries()

void Garfield::ComponentCST::GetElementBoundaries (
unsigned int element,
double & xmin,
double & xmax,
double & ymin,
double & ymax,
double & zmin,
double & zmax) const

### GetElementNodes()

bool Garfield::ComponentCST::GetElementNodes (
const size_t i,
std::vector< size_t > & nodes) const [override]

### GetElementRegion()

bool Garfield::ComponentCST::GetElementRegion (
const size_t i,
size_t & mat,
bool & drift) const [override]

### GetElementVolume()

double Garfield::ComponentCST::GetElementVolume (
const size_t i) const [override], [protected], [virtual]
Reimplemented from Garfield::ComponentFieldMap.

### GetFieldComponent()

float Garfield::ComponentCST::GetFieldComponent (
const unsigned int i,
const unsigned int j,
const unsigned int k,
const double rx,
const double ry,
const double rz,
const char component,
const std::vector< float > & potentials) const [private]

### GetMedium()

Medium * Garfield::ComponentCST::GetMedium (
const double x,
const double y,
const double z) [override]

### GetNode()

bool Garfield::ComponentCST::GetNode (
const size_t i,
double & x,
double & y,
double & z) const [override]

### GetNumberOfElements()

size_t Garfield::ComponentCST::GetNumberOfElements () const [inline], [override]
Definition at line 28 of file ComponentCST.hh.
00028 { return m_nElements; }

### GetNumberOfMeshLines()

void Garfield::ComponentCST::GetNumberOfMeshLines (
unsigned int & nx,
unsigned int & ny,
unsigned int & nz) const

### GetNumberOfNodes()

size_t Garfield::ComponentCST::GetNumberOfNodes () const [inline], [override]
Definition at line 33 of file ComponentCST.hh.
00033 { return m_nNodes; }

### GetPotential()

float Garfield::ComponentCST::GetPotential (
const unsigned int i,
const unsigned int j,
const unsigned int k,
const double rx,
const double ry,
const double rz,
const std::vector< float > & potentials) const [private]

### Index2Element()

int Garfield::ComponentCST::Index2Element (
const unsigned int i,

const unsigned int j,
const unsigned int k) const

Calculate the element index from the position in the x/y/z position vectors (m_xlines, m_ylines, m_zlines). This is public since it is used in ViewFEMesh::DrawCST.

### Index2Node()

int Garfield::ComponentCST::Index2Node (
const unsigned int i,
const unsigned int j,
const unsigned int k) const [private]

### Initialise() [1/2]

bool Garfield::ComponentCST::Initialise (
std::string dataFile,
std::string unit = "cm")

Import of field data based on binary files.
See http://www.desy.de/~zenker/garfieldpp.html to get information about the binary files export from CST.

Parameters

| | |
|---|---|
| dataFile | The binary file containing the field data exported from CST. |
| unit | The units used in the binary file. They are not necessarily equal to CST units. |

### Initialise() [2/2]

bool Garfield::ComponentCST::Initialise (
std::string elist,
std::string nlist,
std::string mplist,
std::string prnsol,
std::string unit = "cm")

### Node2Index()

void Garfield::ComponentCST::Node2Index (
const size_t node,
unsigned int & i,
unsigned int & j,
unsigned int & k) const [private]

### SetRange()

void Garfield::ComponentCST::SetRange () [override], [protected], [virtual]

Reimplemented from Garfield::ComponentFieldMap.

### SetRangeZ()

void Garfield::ComponentCST::SetRangeZ (
const double zmin,
const double zmax)

### SetWeightingField()

bool Garfield::ComponentCST::SetWeightingField (
std::string prnsol,

std::string label,
bool isBinary = true)

Initialise a weighting field.
This function can handle the deprecated text based file format (see Initialise( std::string elist...) for the expected file format, which is similar to prnsol. It also also handles binary files including the weighting field.

Parameters

| | |
|---|---|
| prnsol | The input file (binary/text file) |
| label | The name of the weighting field to be added. If a weighting field with same name already exist it is replaced by the new one. |
| isBinary | Depending on the file type you use, adapt this switch. |

### ShapeField()

void Garfield::ComponentCST::ShapeField (
float & ex,
float & ey,
float & ez,
const double rx,
const double ry,
const double rz,
const unsigned int i,
const unsigned int j,
const unsigned int k,
const std::vector< float > & potentials) const [private]

### ShiftComponent()

void Garfield::ComponentCST::ShiftComponent (
const double xShift,
const double yShift,
const double zShift)

### WeightingField()

void Garfield::ComponentCST::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

### WeightingPotential()

double Garfield::ComponentCST::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

## 7.21.4 Member Data Documentation

### disableFieldComponent

bool Garfield::ComponentCST::disableFieldComponent[3] = {false, false, false} [private]

Definition at line 250 of file ComponentCST.hh.
00250 {false, false, false};

### doShaping

bool Garfield::ComponentCST::doShaping = false [private]
Definition at line 251 of file ComponentCST.hh.

### m_elementMaterial

std::vector<unsigned char> Garfield::ComponentCST::m_elementMaterial [private]
Material id for each element (unsigned char since it uses only 1 byte)
Definition at line 242 of file ComponentCST.hh.

### m_nElements

size_t Garfield::ComponentCST::m_nElements = 0 [private]
Number of elements.
Definition at line 247 of file ComponentCST.hh.

### m_nNodes

size_t Garfield::ComponentCST::m_nNodes = 0 [private]
Number of nodes.
Definition at line 248 of file ComponentCST.hh.

### m_nx

unsigned int Garfield::ComponentCST::m_nx = 0 [private]
Number of mesh lines in x direction.
Definition at line 244 of file ComponentCST.hh.

### m_ny

unsigned int Garfield::ComponentCST::m_ny = 0 [private]
Number of mesh lines in y direction.
Definition at line 245 of file ComponentCST.hh.

### m_nz

unsigned int Garfield::ComponentCST::m_nz = 0 [private]
Number of mesh lines in z direction.
Definition at line 246 of file ComponentCST.hh.

### m_potential

std::vector<float> Garfield::ComponentCST::m_potential [private]
Potentials resulting from the CST simulation.
Definition at line 238 of file ComponentCST.hh.

### m_weightingFields

std::map<std::string, std::vector<float> > Garfield::ComponentCST::m_weightingFields [private]
Map of weighting field potentials.
Definition at line 240 of file ComponentCST.hh.

### m_xlines

std::vector<double> Garfield::ComponentCST::m_xlines [private]
x positions used in the CST mesh
Definition at line 234 of file ComponentCST.hh.

m_ylines

std::vector<double> Garfield::ComponentCST::m_ylines [private]
y positions used in the CST mesh
Definition at line 235 of file ComponentCST.hh.

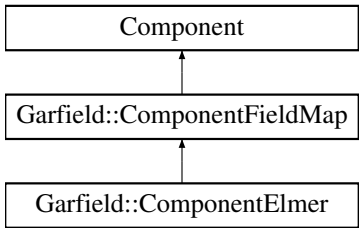
m_zlines

std::vector<double> Garfield::ComponentCST::m_zlines [private]
z positions used in the CST mesh
Definition at line 236 of file ComponentCST.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentCST.hh

## 7.22 Garfield::ComponentElmer Class Reference

Component for importing field maps computed by Elmer.
#include <ComponentElmer.hh>
Inheritance diagram for Garfield::ComponentElmer:




Public Member Functions

- ComponentElmer ()
  Default constructor.
- ComponentElmer (const std::string &header, const std::string &elist, const std::string &nlist, const std::string &mplist, const std::string &volt, const std::string &unit)
  Constructor with a set of field map files, see Initialise().
- ~ComponentElmer ()
  Destructor.
- bool Initialise (const std::string &header="mesh.header", const std::string &elist="mesh.elements", const std::string &nlist="mesh.nodes", const std::string &mplist="dielectrics.dat", const std::string &volt="out.result", const std::string &unit="cm")
  Import a field map from a set of files.
- bool SetWeightingPotential (const std::string &prnsol, const std::string &label)
  Import a list of voltages to be used as weighting potentials.
- bool SetWeightingField (const std::string &prnsol, const std::string &label)

Public Member Functions inherited from Garfield::ComponentFieldMap

- ComponentFieldMap ()=delete
  Default constructor.
- ComponentFieldMap (const std::string &name)
  Constructor.
- virtual ~ComponentFieldMap ()
  Destructor.
- bool Check ()
  Check element aspect ratio.

- void PrintRange ()

  Show x, y, z, V and angular ranges.
- void PrintMaterials ()

  List all currently defined materials.
- void DriftMedium (const size_t imat)

  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)

  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const

  Return the number of materials in the field map.
- double GetPermittivity (const size_t imat) const

  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const

  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)

  Associate a field map material with a Medium object.
- Medium * GetMedium (const size_t imat) const

  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)

  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const

  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const

  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)

  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)

  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)

  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override

- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Private Member Functions

- bool LoadPotentials (const std::string &prnsol, std::vector< double > &pot)

Additional Inherited Members

Protected Types inherited from Garfield::ComponentFieldMap

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Protected Member Functions inherited from Garfield::ComponentFieldMap

- void Reset () override
- void Prepare ()
- virtual void SetRange ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.
- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.
- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

  Move (ex, ey, ez) to global coordinates.
- virtual double GetElementVolume (const size_t i) const
- virtual void GetAspectRatio (const size_t i, double &dmin, double &dmax) const

- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.
- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Static Protected Member Functions inherited from Garfield::ComponentFieldMap

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)

  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a triangle.
- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)

  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a curved quadrilateral.
- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)

Interpolate the potential in a curved quadratic tetrahedron.

- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

  Interpolate the field in a curved quadratic tetrahedron.
- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)
- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes inherited from Garfield::ComponentFieldMap

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

  Flag to check if bounding boxes of elements are cached.
- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.
- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false
- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.22.1 Detailed Description

Component for importing field maps computed by Elmer.
Definition at line 12 of file ComponentElmer.hh.

### 7.22.2 Constructor & Destructor Documentation

ComponentElmer() [1/2]

Garfield::ComponentElmer::ComponentElmer ()
Default constructor.

ComponentElmer() [2/2]

```
Garfield::ComponentElmer::ComponentElmer (
                const std::string & header,
                const std::string & elist,
                const std::string & nlist,
                const std::string & mplist,
                const std::string & volt,
                const std::string & unit)
```

Constructor with a set of field map files, see Initialise().

~ComponentElmer()

Garfield::ComponentElmer::~ComponentElmer () [inline]
Destructor.
Definition at line 21 of file ComponentElmer.hh.

```
00021 {}
```

### 7.22.3 Member Function Documentation

Initialise()

```
bool Garfield::ComponentElmer::Initialise (
                const std::string & header = "mesh.header",
                const std::string & elist = "mesh.elements",
                const std::string & nlist = "mesh.nodes",
                const std::string & mplist = "dielectrics.dat",
                const std::string & volt = "out.result",
                const std::string & unit = "cm")
```

Import a field map from a set of files.

Parameters

| | |
|---|---|
| header | name of the header file (contains the number of elements and nodes). |
| elist | name of the file that contains the list of mesh elements |
| nlist | name of the file that contains the list of mesh nodes |
| mplist | name of the file that contains the material properties |
| volt | output of the field solver (list of voltages) |
| unit | length unit to be used |

LoadPotentials()

```
bool Garfield::ComponentElmer::LoadPotentials (
                const std::string & prnsol,
                std::vector< double > & pot)   [private]
```

SetWeightingField()

```
bool Garfield::ComponentElmer::SetWeightingField (
                const std::string & prnsol,
                const std::string & label)
```

SetWeightingPotential()

```
bool Garfield::ComponentElmer::SetWeightingPotential (
                const std::string & prnsol,
                const std::string & label)    [inline]
```

Import a list of voltages to be used as weighting potentials.
Definition at line 39 of file ComponentElmer.hh.

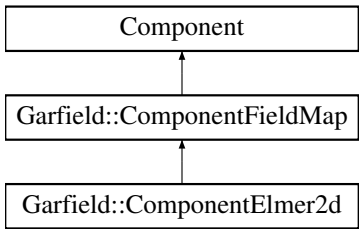
```
00040                                                 {
00041     return SetWeightingField(prnsol, label);
00042   }
```

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentElmer.hh

## 7.23 Garfield::ComponentElmer2d Class Reference

Component for importing two-dimensional field maps computed by Elmer.
#include <ComponentElmer2d.hh>
Inheritance diagram for Garfield::ComponentElmer2d:




Public Member Functions

- ComponentElmer2d ()
  Default constructor.
- ComponentElmer2d (const std::string &header, const std::string &elist, const std::string &nlist, const std::string &mplist, const std::string &volt, const std::string &unit)
  Constructor with a set of field map files, see Initialise().
- ~ComponentElmer2d ()
  Destructor.
- bool Initialise (const std::string &header="mesh.header", const std::string &elist="mesh.elements", const std::string &nlist="mesh.nodes", const std::string &mplist="dielectrics.dat", const std::string &volt="out.result", const std::string &unit="cm")
  Import a field map from a set of files.
- bool SetWeightingPotential (const std::string &prnsol, const std::string &label)
  Import a list of voltages to be used as weighting potentials.
- bool SetWeightingField (const std::string &prnsol, const std::string &label)
- void SetRangeZ (const double zmin, const double zmax)

Public Member Functions inherited from Garfield::ComponentFieldMap

- ComponentFieldMap ()=delete
  Default constructor.
- ComponentFieldMap (const std::string &name)
  Constructor.

- virtual ~ComponentFieldMap ()

  Destructor.
- bool Check ()

  Check element aspect ratio.
- void PrintRange ()

  Show x, y, z, V and angular ranges.
- void PrintMaterials ()

  List all currently defined materials.
- void DriftMedium (const size_t imat)

  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)

  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const

  Return the number of materials in the field map.
- double GetPermittivity (const size_t imat) const

  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const

  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)

  Associate a field map material with a Medium object.
- Medium * GetMedium (const size_t imat) const

  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)

  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const

  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const

  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)

  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)

  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)

  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override

- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Private Member Functions

- bool LoadPotentials (const std::string &prnsol, std::vector< double > &pot)

Additional Inherited Members

Protected Types inherited from Garfield::ComponentFieldMap

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Protected Member Functions inherited from Garfield::ComponentFieldMap

- void Reset () override
- void Prepare ()
- virtual void SetRange ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.
- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.

- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

  Move (ex, ey, ez) to global coordinates.
- virtual double GetElementVolume (const size_t i) const
- virtual void GetAspectRatio (const size_t i, double &dmin, double &dmax) const
- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.
- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Static Protected Member Functions inherited from Garfield::ComponentFieldMap

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)

  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a triangle.

- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)

  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a curved quadrilateral.
- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)

  Interpolate the potential in a curved quadratic tetrahedron.
- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

  Interpolate the field in a curved quadratic tetrahedron.
- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)
- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes inherited from Garfield::ComponentFieldMap

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

  Flag to check if bounding boxes of elements are cached.
- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.

- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false
- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.23.1 Detailed Description

Component for importing two-dimensional field maps computed by Elmer.
Definition at line 12 of file ComponentElmer2d.hh.

### 7.23.2 Constructor & Destructor Documentation

#### ComponentElmer2d() [1/2]

Garfield::ComponentElmer2d::ComponentElmer2d ()
Default constructor.

#### ComponentElmer2d() [2/2]

```
Garfield::ComponentElmer2d::ComponentElmer2d (
                const std::string & header,
                const std::string & elist,
                const std::string & nlist,
                const std::string & mplist,
                const std::string & volt,
                const std::string & unit)
```

Constructor with a set of field map files, see Initialise().

#### ~ComponentElmer2d()

Garfield::ComponentElmer2d::~ComponentElmer2d () [inline]
Destructor.
Definition at line 21 of file ComponentElmer2d.hh.
00021 {}

### 7.23.3 Member Function Documentation

#### Initialise()

```
bool Garfield::ComponentElmer2d::Initialise (
                const std::string & header = "mesh.header",
                const std::string & elist = "mesh.elements",
                const std::string & nlist = "mesh.nodes",
                const std::string & mplist = "dielectrics.dat",
                const std::string & volt = "out.result",
                const std::string & unit = "cm")
```

Import a field map from a set of files.

Parameters

| | |
|---|---|
| header | name of the header file (contains the number of elements and nodes). |
| elist | name of the file that contains the list of mesh elements |
| nlist | name of the file that contains the list of mesh nodes |
| mplist | name of the file that contains the material properties |

Parameters

| | |
|---|---|
| volt | output of the field solver (list of voltages) |
| unit | length unit to be used |

### LoadPotentials()

bool Garfield::ComponentElmer2d::LoadPotentials (
const std::string & prnsol,
std::vector< double > & pot) [private]

### SetRangeZ()

void Garfield::ComponentElmer2d::SetRangeZ (
const double zmin,
const double zmax)

### SetWeightingField()

bool Garfield::ComponentElmer2d::SetWeightingField (
const std::string & prnsol,
const std::string & label)

### SetWeightingPotential()

bool Garfield::ComponentElmer2d::SetWeightingPotential (
const std::string & prnsol,
const std::string & label) [inline]

Import a list of voltages to be used as weighting potentials.
Definition at line 39 of file ComponentElmer2d.hh.

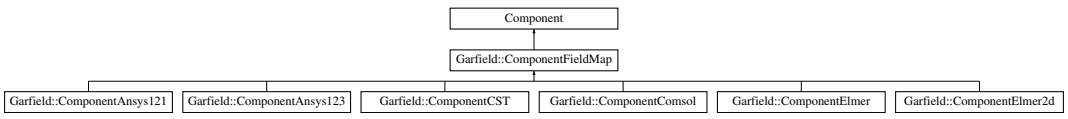
```
00040                                                 {
00041     return SetWeightingField(prnsol, label);
00042   }
```

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentElmer2d.hh

## 7.24 Garfield::ComponentFieldMap Class Reference

Base class for components based on finite-element field maps.
#include <ComponentFieldMap.hh>
Inheritance diagram for Garfield::ComponentFieldMap:




Classes

- struct Element
- struct Material
- struct Node
- struct WeightingFieldCopy

Public Member Functions

- ComponentFieldMap ()=delete

  Default constructor.
- ComponentFieldMap (const std::string &name)

  Constructor.
- virtual ~ComponentFieldMap ()

  Destructor.
- bool Check ()

  Check element aspect ratio.
- void PrintRange ()

  Show x, y, z, V and angular ranges.
- void PrintMaterials ()

  List all currently defined materials.
- void DriftMedium (const size_t imat)

  Flag a field map material as a drift medium.
- void NotDriftMedium (const size_t imat)

  Flag a field map materials as a non-drift medium.
- size_t GetNumberOfMaterials () const

  Return the number of materials in the field map.
- double GetPermittivity (const size_t imat) const

  Return the relative permittivity of a field map material.
- double GetConductivity (const size_t imat) const

  Return the conductivity of a field map material.
- void SetMedium (const size_t imat, Medium *medium)

  Associate a field map material with a Medium object.
- Medium * GetMedium (const size_t imat) const

  Return the Medium associated to a field map material.
- void SetGas (Medium *medium)

  Associate all field map materials with a relative permittivity of unity to a given Medium class.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &mat, bool &drift) const override
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax) const

  Return the volume and aspect ratio of a mesh element.
- size_t GetNumberOfNodes () const override
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
- double GetPotential (const size_t i) const

  Return the potential at a given node.
- void EnableCheckMapIndices (const bool on=true)
- void EnableDeleteBackgroundElements (const bool on=true)

  Option to eliminate mesh elements in conductors (default: on).
- void EnableTetrahedralTreeForElementSearch (const bool on=true)

  Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.
- void EnableConvergenceWarnings (const bool on=true)

  Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override

- ___DEVICE___ void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- double DelayedWeightingPotential (double x, double y, double z, const double t, const std::string &label) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool IsInBoundingBox (const double x, const double y, const double z) const
- bool Is3d () override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- std::map< std::string, std::vector< double > > GetWeightingPotentials ()
- bool GetVoltageRange (double &vmin, double &vmax) override
- void CopyWeightingPotential (const std::string &label, const std::string &labelSource, const double x, const double y, const double z, const double alpha, const double beta, const double gamma)

  Makes a weighting potential copy of a imported map which can be translated and rotated.
- double CreateGPUTransferObject (ComponentGPU *&comp_gpu) override

  Create and initialise GPU Transfer class.

Protected Types

- enum class ElementType { Unknown = 0 , Serendipity = 5 , CurvedTetrahedron = 13 }

Protected Member Functions

- void Reset () override
- void Prepare ()
- virtual void SetRange ()
- void UpdatePeriodicity () override
- void UpdatePeriodicity2d ()
- void UpdatePeriodicityCommon ()
- bool SetDefaultDriftMedium ()

  Find lowest epsilon, check for eps = 0, set default drift media flags.
- ___DEVICE___ int Field (const double x, const double y, const double z, double &fx, double &fy, double &fz, int &iel, const std::vector< double > &potentials) const

  Compute the electric/weighting field.
- double Potential (const double x, const double y, const double z, const std::vector< double > &potentials) const

  Compute the electrostatic/weighting potential.
- int FindElement5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic quadrilaterals.
- ___DEVICE___ int FindElement13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det) const

  Find the element for a point in curved quadratic tetrahedra.
- int FindElementCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Find the element for a point in a cube.
- ___DEVICE___ void MapCoordinates (double &xpos, double &ypos, double &zpos, bool &xmirrored, bool &ymirrored, bool &zmirrored, double &rcoordinate, double &rotation) const

  Move (xpos, ypos, zpos) to field map coordinates.

- ___DEVICE___ void UnmapFields (double &ex, double &ey, double &ez, const double xpos, const double ypos, const double zpos, const bool xmirrored, const bool ymirrored, const bool zmirrored, const double rcoordinate, const double rotation) const

  Move (ex, ey, ez) to global coordinates.
- virtual double GetElementVolume (const size_t i) const
- virtual void GetAspectRatio (const size_t i, double &dmin, double &dmax) const
- void PrintWarning (const std::string &header)
- void PrintNotReady (const std::string &header) const
- void PrintCouldNotOpen (const std::string &header, const std::string &filename) const
- void PrintElement (const std::string &header, const double x, const double y, const double z, const double t1, const double t2, const double t3, const double t4, const size_t i, const std::vector< double > &potential) const
- void TimeInterpolation (const double t, double &f0, double &f1, int &i0, int &i1)

  Interpolation of potential between two time slices.
- int Coordinates3 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic triangles.
- int Coordinates4 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for linear quadrilaterals.
- int Coordinates5 (const double x, const double y, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 8 > &xn, const std::array< double, 8 > &yn) const

  Calculate local coordinates for curved quadratic quadrilaterals.
- ___DEVICE___ void Coordinates12 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates in linear tetrahedra.
- ___DEVICE___ int Coordinates13 (const double x, const double y, const double z, double &t1, double &t2, double &t3, double &t4, double jac[4][4], double &det, const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const std::array< std::array< double, 3 >, 4 > &w) const

  Calculate local coordinates for curved quadratic tetrahedra.
- int CoordinatesCube (const double x, const double y, const double z, double &t1, double &t2, double &t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN, const Element &element) const

  Calculate local coordinates for a cube.
- void JacobianCube (const Element &element, const double t1, const double t2, const double t3, TMatrixD *&jac, std::vector< TMatrixD * > &dN) const

  Calculate Jacobian for a cube.
- void CalculateElementBoundingBoxes ()

  Calculate the bounding boxes of all elements after initialization.
- bool InitializeTetrahedralTree ()

  Initialize the tetrahedral tree.

Static Protected Member Functions

- static double ScalingFactor (std::string unit)
- static double Potential3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t)

  Interpolate the potential in a triangle.
- static void Field3 (const std::array< double, 6 > &v, const std::array< double, 3 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a triangle.

- static double Potential5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t)

  Interpolate the potential in a curved quadrilateral.
- static void Field5 (const std::array< double, 8 > &v, const std::array< double, 2 > &t, double jac[4][4], const double det, double &ex, double &ey)

  Interpolate the field in a curved quadrilateral.
- static double Potential13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t)

  Interpolate the potential in a curved quadratic tetrahedron.
- static ___DEVICE___ void Field13 (const std::array< double, 10 > &v, const std::array< double, 4 > &t, double jac[4][4], const double det, double &ex, double &ey, double &ez)

  Interpolate the field in a curved quadratic tetrahedron.
- static int ReadInteger (char *token, int def, bool &error)
- static double ReadDouble (char *token, double def, bool &error)
- static void Jacobian3 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, const double w, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic triangles.
- static void Jacobian5 (const std::array< double, 8 > &xn, const std::array< double, 8 > &yn, const double u, const double v, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic quadrilaterals.
- static ___DEVICE___ void Jacobian13 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn, const double fourt0, const double fourt1, const double fourt2, const double fourt3, double &det, double jac[4][4])

  Calculate Jacobian for curved quadratic tetrahedra.
- static std::array< std::array< double, 3 >, 4 > Weights12 (const std::array< double, 10 > &xn, const std::array< double, 10 > &yn, const std::array< double, 10 > &zn)

Protected Attributes

- bool m_is3d = true
- ElementType m_elementType = ElementType::CurvedTetrahedron
- std::vector< Element > m_elements
- std::vector< int > m_elementIndices
- std::vector< bool > m_degenerate
- std::vector< std::array< double, 3 > > m_bbMin
- std::vector< std::array< double, 3 > > m_bbMax
- std::vector< std::array< std::array< double, 3 >, 4 > > m_w12
- std::vector< Node > m_nodes
- std::vector< double > m_pot
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< std::vector< double > > > m_dwpot
- std::vector< Material > m_materials
- std::map< std::string, WeightingFieldCopy > m_wfieldCopies
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_minBoundingBox = {{0., 0., 0.}}
- std::array< double, 3 > m_maxBoundingBox = {{0., 0., 0.}}
- bool m_cacheElemBoundingBoxes = false

  Flag to check if bounding boxes of elements are cached.
- std::array< double, 3 > m_mapmin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapmax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamin = {{0., 0., 0.}}
- std::array< double, 3 > m_mapamax = {{0., 0., 0.}}
- std::array< double, 3 > m_mapna = {{0., 0., 0.}}
- std::array< double, 3 > m_cells = {{0., 0., 0.}}
- double m_mapvmin = 0.
- double m_mapvmax = 0.

- std::array< bool, 3 > m_setang
- bool m_deleteBackground = true
- bool m_warning = false
- unsigned int m_nWarnings = 0
- bool m_printConvergenceWarnings = true
- bool m_checkMultipleElement = false

  Scan for multiple elements that contain a point.
- bool m_useTetrahedralTree = true
- std::unique_ptr< GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree

### 7.24.1 Detailed Description

Base class for components based on finite-element field maps.
Definition at line 29 of file ComponentFieldMap.hh.

### 7.24.2 Member Enumeration Documentation

#### ElementType

enum class Garfield::ComponentFieldMap::ElementType [strong], [protected]

Enumerator

| | |
|---|---|
| Unknown | |
| Serendipity | |
| CurvedTetrahedron | |

Definition at line 163 of file ComponentFieldMap.hh.

```
00163                       {
00164     Unknown = 0,
00165     Serendipity = 5,
00166     CurvedTetrahedron = 13
00167   };
```

### 7.24.3 Constructor & Destructor Documentation

#### ComponentFieldMap() [1/2]

Garfield::ComponentFieldMap::ComponentFieldMap () [delete]
Default constructor.

#### ComponentFieldMap() [2/2]

Garfield::ComponentFieldMap::ComponentFieldMap (
const std::string & name)
Constructor.

#### ~ComponentFieldMap()

virtual Garfield::ComponentFieldMap::~ComponentFieldMap () [virtual]
Destructor.

### 7.24.4 Member Function Documentation

#### CalculateElementBoundingBoxes()

void Garfield::ComponentFieldMap::CalculateElementBoundingBoxes () [protected]
Calculate the bounding boxes of all elements after initialization.

#### Check()

bool Garfield::ComponentFieldMap::Check ()
Check element aspect ratio.

### Coordinates12()

```
___DEVICE___ void Garfield::ComponentFieldMap::Coordinates12 (
                const double x,
                const double y,
                const double z,
                double & t1,
                double & t2,
                double & t3,
                double & t4,
                const std::array< double, 10 > & xn,
                const std::array< double, 10 > & yn,
                const std::array< double, 10 > & zn,
                const std::array< std::array< double, 3 >, 4 > & w) const   [protected]
```

Calculate local coordinates in linear tetrahedra.

### Coordinates13()

```
___DEVICE___ int Garfield::ComponentFieldMap::Coordinates13 (
                const double x,
                const double y,
                const double z,
                double & t1,
                double & t2,
                double & t3,
                double & t4,
                double jac[4][4],
                double & det,
                const std::array< double, 10 > & xn,
                const std::array< double, 10 > & yn,
                const std::array< double, 10 > & zn,
                const std::array< std::array< double, 3 >, 4 > & w) const   [protected]
```

Calculate local coordinates for curved quadratic tetrahedra.

### Coordinates3()

```
int Garfield::ComponentFieldMap::Coordinates3 (
                const double x,
                const double y,
                double & t1,
                double & t2,
                double & t3,
                double & t4,
                double jac[4][4],
                double & det,
                const std::array< double, 8 > & xn,
                const std::array< double, 8 > & yn) const   [protected]
```

Calculate local coordinates for curved quadratic triangles.

### Coordinates4()

```
int Garfield::ComponentFieldMap::Coordinates4 (
                const double x,
                const double y,
                double & t1,
                double & t2,
                double & t3,
                double & t4,
                double & det,
```

const std::array< double, 8 > & xn,
const std::array< double, 8 > & yn) const [protected]

Calculate local coordinates for linear quadrilaterals.

### Coordinates5()

int Garfield::ComponentFieldMap::Coordinates5 (
const double x,
const double y,
double & t1,
double & t2,
double & t3,
double & t4,
double jac[4][4],
double & det,
const std::array< double, 8 > & xn,
const std::array< double, 8 > & yn) const [protected]

Calculate local coordinates for curved quadratic quadrilaterals.

### CoordinatesCube()

int Garfield::ComponentFieldMap::CoordinatesCube (
const double x,
const double y,
const double z,
double & t1,
double & t2,
double & t3,
TMatrixD *& jac,
std::vector< TMatrixD * > & dN,
const Element & element) const [protected]

Calculate local coordinates for a cube.

### CopyWeightingPotential()

void Garfield::ComponentFieldMap::CopyWeightingPotential (
const std::string & label,
const std::string & labelSource,
const double x,
const double y,
const double z,
const double alpha,
const double beta,
const double gamma)

Makes a weighting potential copy of a imported map which can be translated and rotated.

Parameters

| | |
|---|---|
| label | name of new electrode |
| labelSource | name of the source electrode that will be copied |
| x | translation in the x-direction. |
| y | translation in the y-direction. |
| z | translation in the z-direction. |
| alpha | rotation around the x-axis. |
| beta | rotation around the y-axis. |
| gamma | rotation around the z-axis. |

### CreateGPUTransferObject()

```
double Garfield::ComponentFieldMap::CreateGPUTransferObject (
            ComponentGPU *& comp_gpu)   [override]
```

Create and initialise GPU Transfer class.

### DelayedWeightingPotential()

```
double Garfield::ComponentFieldMap::DelayedWeightingPotential (
            double x,
            double y,
            double z,
            const double t,
            const std::string & label)   [override]
```

### DelayedWeightingPotentials()

```
void Garfield::ComponentFieldMap::DelayedWeightingPotentials (
            const double x,
            const double y,
            const double z,
            const std::string & label,
            std::vector< double > & dwp)   [override]
```

### DriftMedium()

```
void Garfield::ComponentFieldMap::DriftMedium (
            const size_t imat)
```

Flag a field map material as a drift medium.

### ElectricField() [1/2]

```
void Garfield::ComponentFieldMap::ElectricField (
            const double x,
            const double y,
            const double z,
            double & ex,
            double & ey,
            double & ez,
            double & v,
            Medium *& m,
            int & status)   [override]
```

### ElectricField() [2/2]

```
void Garfield::ComponentFieldMap::ElectricField (
            const double x,
            const double y,
            const double z,
            double & ex,
            double & ey,
            double & ez,
            Medium *& m,
            int & status)   [override]
```

### EnableCheckMapIndices()

```
void Garfield::ComponentFieldMap::EnableCheckMapIndices (
            const bool on = true)   [inline]
```

Definition at line 79 of file ComponentFieldMap.hh.

```
00079                                   {
```

```
00080     m_checkMultipleElement = on;
00081   }
```

EnableConvergenceWarnings()

void Garfield::ComponentFieldMap::EnableConvergenceWarnings (
const bool on = true) [inline]

Enable or disable warnings that the calculation of the local coordinates did not achieve the requested precision.

Definition at line 95 of file ComponentFieldMap.hh.

```
00095                                                 {
00096     m_printConvergenceWarnings = on;
00097   }
```

EnableDeleteBackgroundElements()

void Garfield::ComponentFieldMap::EnableDeleteBackgroundElements (
const bool on = true) [inline]

Option to eliminate mesh elements in conductors (default: on).

Definition at line 83 of file ComponentFieldMap.hh.

```
00083                                                 {
00084     m_deleteBackground = on;
00085   }
```

EnableTetrahedralTreeForElementSearch()

void Garfield::ComponentFieldMap::EnableTetrahedralTreeForElementSearch (
const bool on = true) [inline]

Enable or disable the usage of the tetrahedral tree for searching the element in the mesh.

Definition at line 89 of file ComponentFieldMap.hh.

```
00089                                                 {
00090     m_useTetrahedralTree = on;
00091   }
```

Field()

___DEVICE___ int Garfield::ComponentFieldMap::Field (
const double x,
const double y,
const double z,
double & fx,
double & fy,
double & fz,
int & iel,
const std::vector< double > & potentials) const [protected]

Compute the electric/weighting field.

Field13()

___DEVICE___ void Garfield::ComponentFieldMap::Field13 (
const std::array< double, 10 > & v,
const std::array< double, 4 > & t,
double jac[4][4],
const double det,
double & ex,
double & ey,
double & ez) [static], [protected]

Interpolate the field in a curved quadratic tetrahedron.

Field3()

void Garfield::ComponentFieldMap::Field3 (
const std::array< double, 6 > & v,

const std::array< double, 3 > & t,
double jac[4][4],
const double det,
double & ex,
double & ey) [static], [protected]

Interpolate the field in a triangle.

### Field5()

void Garfield::ComponentFieldMap::Field5 (
const std::array< double, 8 > & v,
const std::array< double, 2 > & t,
double jac[4][4],
const double det,
double & ex,
double & ey) [static], [protected]

Interpolate the field in a curved quadrilateral.

### FindElement13()

___DEVICE___ int Garfield::ComponentFieldMap::FindElement13 (
const double x,
const double y,
const double z,
double & t1,
double & t2,
double & t3,
double & t4,
double jac[4][4],
double & det) const [protected]

Find the element for a point in curved quadratic tetrahedra.

### FindElement5()

int Garfield::ComponentFieldMap::FindElement5 (
const double x,
const double y,
double & t1,
double & t2,
double & t3,
double & t4,
double jac[4][4],
double & det) const [protected]

Find the element for a point in curved quadratic quadrilaterals.

### FindElementCube()

int Garfield::ComponentFieldMap::FindElementCube (
const double x,
const double y,
const double z,
double & t1,
double & t2,
double & t3,
TMatrixD *& jac,
std::vector< TMatrixD * > & dN) const [protected]

Find the element for a point in a cube.

### GetAspectRatio()

virtual void Garfield::ComponentFieldMap::GetAspectRatio (
const size_t i,
double & dmin,
double & dmax) const [protected], [virtual]

Reimplemented in Garfield::ComponentCST.

### GetBoundingBox()

bool Garfield::ComponentFieldMap::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [override]

### GetConductivity()

double Garfield::ComponentFieldMap::GetConductivity (
const size_t imat) const

Return the conductivity of a field map material.

### GetElement()

bool Garfield::ComponentFieldMap::GetElement (
const size_t i,
double & vol,
double & dmin,
double & dmax) const

Return the volume and aspect ratio of a mesh element.

### GetElementaryCell()

bool Garfield::ComponentFieldMap::GetElementaryCell (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [override]

### GetElementNodes()

bool Garfield::ComponentFieldMap::GetElementNodes (
const size_t i,
std::vector< size_t > & nodes) const [override]

### GetElementRegion()

bool Garfield::ComponentFieldMap::GetElementRegion (
const size_t i,
size_t & mat,
bool & drift) const [override]

### GetElementVolume()

virtual double Garfield::ComponentFieldMap::GetElementVolume (
const size_t i) const [protected], [virtual]

Reimplemented in Garfield::ComponentCST.

### GetMedium() [1/2]

```
Medium * Garfield::ComponentFieldMap::GetMedium (
            const double x,
            const double y,
            const double z)    [override]
```

### GetMedium() [2/2]

```
Medium * Garfield::ComponentFieldMap::GetMedium (
            const size_t imat) const
```

Return the Medium associated to a field map material.

### GetNode()

```
bool Garfield::ComponentFieldMap::GetNode (
            const size_t i,
            double & x,
            double & y,
            double & z) const    [override]
```

### GetNumberOfElements()

```
size_t Garfield::ComponentFieldMap::GetNumberOfElements () const    [inline], [override]
```

Definition at line 65 of file ComponentFieldMap.hh.

```
00065 { return m_elements.size(); }
```

### GetNumberOfMaterials()

```
size_t Garfield::ComponentFieldMap::GetNumberOfMaterials () const    [inline]
```

Return the number of materials in the field map.

Definition at line 51 of file ComponentFieldMap.hh.

```
00051 { return m_materials.size(); }
```

### GetNumberOfNodes()

```
size_t Garfield::ComponentFieldMap::GetNumberOfNodes () const    [inline], [override]
```

Definition at line 73 of file ComponentFieldMap.hh.

```
00073 { return m_nodes.size(); }
```

### GetPermittivity()

```
double Garfield::ComponentFieldMap::GetPermittivity (
            const size_t imat) const
```

Return the relative permittivity of a field map material.

### GetPotential()

```
double Garfield::ComponentFieldMap::GetPotential (
            const size_t i) const
```

Return the potential at a given node.

### GetVoltageRange()

```
bool Garfield::ComponentFieldMap::GetVoltageRange (
            double & vmin,
            double & vmax)    [inline], [override]
```

Definition at line 139 of file ComponentFieldMap.hh.

```
00139                                                {
00140   vmin = m_mapvmin;
00141   vmax = m_mapvmax;
00142   return true;
00143 }
```

### GetWeightingPotentials()

std::map< std::string, std::vector< double > > Garfield::ComponentFieldMap::GetWeightingPotentials () [inline]

Definition at line 135 of file ComponentFieldMap.hh.

```
00135                                                 {
00136   return m_wpot;
00137 }
```

### InitializeTetrahedralTree()

bool Garfield::ComponentFieldMap::InitializeTetrahedralTree () [protected]

Initialize the tetrahedral tree.

### Is3d()

bool Garfield::ComponentFieldMap::Is3d () [inline], [override]

Definition at line 129 of file ComponentFieldMap.hh.

```
00129 { return m_is3d; }
```

### IsInBoundingBox()

bool Garfield::ComponentFieldMap::IsInBoundingBox (
const double x,
const double y,
const double z) const [inline]

Definition at line 123 of file ComponentFieldMap.hh.

```
00123                                                                 {
00124   return x >= m_minBoundingBox[0] && x <= m_maxBoundingBox[0] &&
00125          y >= m_minBoundingBox[1] && y <= m_maxBoundingBox[1] &&
00126          z >= m_minBoundingBox[2] && y <= m_maxBoundingBox[2];
00127 }
```

### Jacobian13()

___DEVICE___ void Garfield::ComponentFieldMap::Jacobian13 (
const std::array< double, 10 > & xn,
const std::array< double, 10 > & yn,
const std::array< double, 10 > & zn,
const double fourt0,
const double fourt1,
const double fourt2,
const double fourt3,
double & det,
double jac[4][4]) [static], [protected]

Calculate Jacobian for curved quadratic tetrahedra.

### Jacobian3()

void Garfield::ComponentFieldMap::Jacobian3 (
const std::array< double, 8 > & xn,
const std::array< double, 8 > & yn,
const double u,
const double v,
const double w,
double & det,
double jac[4][4]) [static], [protected]

Calculate Jacobian for curved quadratic triangles.

### Jacobian5()

void Garfield::ComponentFieldMap::Jacobian5 (
const std::array< double, 8 > & xn,
const std::array< double, 8 > & yn,

const double u,
const double v,
double & det,
double jac[4][4]) [static], [protected]

Calculate Jacobian for curved quadratic quadrilaterals.

### JacobianCube()

void Garfield::ComponentFieldMap::JacobianCube (
const Element & element,
const double t1,
const double t2,
const double t3,
TMatrixD *& jac,
std::vector< TMatrixD * > & dN) const [protected]

Calculate Jacobian for a cube.

### MapCoordinates()

___DEVICE___ void Garfield::ComponentFieldMap::MapCoordinates (
double & xpos,
double & ypos,
double & zpos,
bool & xmirrored,
bool & ymirrored,
bool & zmirrored,
double & rcoordinate,
double & rotation) const [protected]

Move (xpos, ypos, zpos) to field map coordinates.

### NotDriftMedium()

void Garfield::ComponentFieldMap::NotDriftMedium (
const size_t imat)

Flag a field map materials as a non-drift medium.

### Potential()

double Garfield::ComponentFieldMap::Potential (
const double x,
const double y,
const double z,
const std::vector< double > & potentials) const [protected]

Compute the electrostatic/weighting potential.

### Potential13()

double Garfield::ComponentFieldMap::Potential13 (
const std::array< double, 10 > & v,
const std::array< double, 4 > & t) [static], [protected]

Interpolate the potential in a curved quadratic tetrahedron.

### Potential3()

double Garfield::ComponentFieldMap::Potential3 (
const std::array< double, 6 > & v,
const std::array< double, 3 > & t) [static], [protected]

Interpolate the potential in a triangle.

### Potential5()

```
double Garfield::ComponentFieldMap::Potential5 (
                const std::array< double, 8 > & v,
                const std::array< double, 2 > & t)   [static], [protected]
```

Interpolate the potential in a curved quadrilateral.

### Prepare()

```
void Garfield::ComponentFieldMap::Prepare ()   [protected]
```

### PrintCouldNotOpen()

```
void Garfield::ComponentFieldMap::PrintCouldNotOpen (
                const std::string & header,
                const std::string & filename) const   [protected]
```

### PrintElement()

```
void Garfield::ComponentFieldMap::PrintElement (
                const std::string & header,
                const double x,
                const double y,
                const double z,
                const double t1,
                const double t2,
                const double t3,
                const double t4,
                const size_t i,
                const std::vector< double > & potential) const   [protected]
```

### PrintMaterials()

```
void Garfield::ComponentFieldMap::PrintMaterials ()
```

List all currently defined materials.

### PrintNotReady()

```
void Garfield::ComponentFieldMap::PrintNotReady (
                const std::string & header) const   [protected]
```

### PrintRange()

```
void Garfield::ComponentFieldMap::PrintRange ()
```

Show x, y, z, V and angular ranges.

### PrintWarning()

```
void Garfield::ComponentFieldMap::PrintWarning (
                const std::string & header)   [protected]
```

### ReadDouble()

```
double Garfield::ComponentFieldMap::ReadDouble (
                char * token,
                double def,
                bool & error)   [static], [protected]
```

### ReadInteger()

int Garfield::ComponentFieldMap::ReadInteger (
char * token,
int def,
bool & error) [static], [protected]

### Reset()

void Garfield::ComponentFieldMap::Reset () [override], [protected]

### ScalingFactor()

double Garfield::ComponentFieldMap::ScalingFactor (
std::string unit) [static], [protected]

### SetDefaultDriftMedium()

bool Garfield::ComponentFieldMap::SetDefaultDriftMedium () [protected]
Find lowest epsilon, check for eps = 0, set default drift media flags.

### SetGas()

void Garfield::ComponentFieldMap::SetGas (
Medium * medium)
Associate all field map materials with a relative permittivity of unity to a given Medium class.

### SetMedium()

void Garfield::ComponentFieldMap::SetMedium (
const size_t imat,
Medium * medium)
Associate a field map material with a Medium object.

### SetRange()

virtual void Garfield::ComponentFieldMap::SetRange () [protected], [virtual]
Reimplemented in Garfield::ComponentCST.

### TimeInterpolation()

void Garfield::ComponentFieldMap::TimeInterpolation (
const double t,
double & f0,
double & f1,
int & i0,
int & i1) [protected]
Interpolation of potential between two time slices.

### UnmapFields()

___DEVICE___ void Garfield::ComponentFieldMap::UnmapFields (
double & ex,
double & ey,
double & ez,
const double xpos,
const double ypos,
const double zpos,
const bool xmirrored,
const bool ymirrored,
const bool zmirrored,

const double rcoordinate,
const double rotation) const [protected]

Move (ex, ey, ez) to global coordinates.

### UpdatePeriodicity()

void Garfield::ComponentFieldMap::UpdatePeriodicity () [inline], [override], [protected]

Definition at line 327 of file ComponentFieldMap.hh.

```
00327                                {
00328     if (!m_is3d) UpdatePeriodicity2d();
00329     UpdatePeriodicityCommon();
00330   }
```

### UpdatePeriodicity2d()

void Garfield::ComponentFieldMap::UpdatePeriodicity2d () [protected]

### UpdatePeriodicityCommon()

void Garfield::ComponentFieldMap::UpdatePeriodicityCommon () [protected]

### WeightingField()

___DEVICE___ void Garfield::ComponentFieldMap::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

### WeightingPotential()

double Garfield::ComponentFieldMap::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

### Weights12()

std::array< std::array< double, 3 >, 4 > Garfield::ComponentFieldMap::Weights12 (
const std::array< double, 10 > & xn,
const std::array< double, 10 > & yn,
const std::array< double, 10 > & zn) [static], [protected]

## 7.24.5 Member Data Documentation

### m_bbMax

std::vector<std::array<double, 3> > Garfield::ComponentFieldMap::m_bbMax [protected]

Definition at line 199 of file ComponentFieldMap.hh.

### m_bbMin

std::vector<std::array<double, 3> > Garfield::ComponentFieldMap::m_bbMin [protected]

Definition at line 198 of file ComponentFieldMap.hh.

### m_cacheElemBoundingBoxes

bool Garfield::ComponentFieldMap::m_cacheElemBoundingBoxes = false [protected]
Flag to check if bounding boxes of elements are cached.
Definition at line 279 of file ComponentFieldMap.hh.

### m_cells

std::array<double, 3> Garfield::ComponentFieldMap::m_cells = {{0., 0., 0.}} [protected]
Definition at line 297 of file ComponentFieldMap.hh.
00297 {{0., 0., 0.}};

### m_checkMultipleElement

bool Garfield::ComponentFieldMap::m_checkMultipleElement = false [protected]
Scan for multiple elements that contain a point.
Definition at line 429 of file ComponentFieldMap.hh.

### m_degenerate

std::vector<bool> Garfield::ComponentFieldMap::m_degenerate [protected]
Definition at line 190 of file ComponentFieldMap.hh.

### m_deleteBackground

bool Garfield::ComponentFieldMap::m_deleteBackground = true [protected]
Definition at line 305 of file ComponentFieldMap.hh.

### m_dwpot

std::map<std::string, std::vector<std::vector<double> > > Garfield::ComponentFieldMap::m_dwpot [protected]
Definition at line 227 of file ComponentFieldMap.hh.

### m_elementIndices

std::vector<int> Garfield::ComponentFieldMap::m_elementIndices [protected]
Definition at line 183 of file ComponentFieldMap.hh.

### m_elements

std::vector<Element> Garfield::ComponentFieldMap::m_elements [protected]
Definition at line 182 of file ComponentFieldMap.hh.

### m_elementType

ElementType Garfield::ComponentFieldMap::m_elementType = ElementType::CurvedTetrahedron [protected]
Definition at line 168 of file ComponentFieldMap.hh.

### m_hasBoundingBox

bool Garfield::ComponentFieldMap::m_hasBoundingBox = false [protected]
Definition at line 274 of file ComponentFieldMap.hh.

### m_is3d

bool Garfield::ComponentFieldMap::m_is3d = true [protected]
Definition at line 161 of file ComponentFieldMap.hh.

### m_mapamax

std::array<double, 3> Garfield::ComponentFieldMap::m_mapamax = {{0., 0., 0.}} [protected]
Definition at line 292 of file ComponentFieldMap.hh.
00292 {{0., 0., 0.}};

### m_mapamin

std::array<double, 3> Garfield::ComponentFieldMap::m_mapamin = {{0., 0., 0.}} [protected]
Definition at line 291 of file ComponentFieldMap.hh.
00291 {{0., 0., 0.}};

### m_mapmax

std::array<double, 3> Garfield::ComponentFieldMap::m_mapmax = {{0., 0., 0.}} [protected]
Definition at line 290 of file ComponentFieldMap.hh.
00290 {{0., 0., 0.}};

### m_mapmin

std::array<double, 3> Garfield::ComponentFieldMap::m_mapmin = {{0., 0., 0.}} [protected]
Definition at line 289 of file ComponentFieldMap.hh.
00289 {{0., 0., 0.}};

### m_mapna

std::array<double, 3> Garfield::ComponentFieldMap::m_mapna = {{0., 0., 0.}} [protected]
Definition at line 296 of file ComponentFieldMap.hh.
00296 {{0., 0., 0.}};

### m_mapvmax

double Garfield::ComponentFieldMap::m_mapvmax = 0. [protected]
Definition at line 300 of file ComponentFieldMap.hh.

### m_mapvmin

double Garfield::ComponentFieldMap::m_mapvmin = 0. [protected]
Definition at line 299 of file ComponentFieldMap.hh.

### m_materials

std::vector<Material> Garfield::ComponentFieldMap::m_materials [protected]
Definition at line 258 of file ComponentFieldMap.hh.

### m_maxBoundingBox

std::array<double, 3> Garfield::ComponentFieldMap::m_maxBoundingBox = {{0., 0., 0.}} [protected]
Definition at line 276 of file ComponentFieldMap.hh.
00276 {{0., 0., 0.}};

### m_minBoundingBox

std::array<double, 3> Garfield::ComponentFieldMap::m_minBoundingBox = {{0., 0., 0.}} [protected]
Definition at line 275 of file ComponentFieldMap.hh.
00275 {{0., 0., 0.}};

### m_nodes

std::vector<Node> Garfield::ComponentFieldMap::m_nodes [protected]
Definition at line 217 of file ComponentFieldMap.hh.

### m_nWarnings

unsigned int Garfield::ComponentFieldMap::m_nWarnings = 0 [protected]
Definition at line 309 of file ComponentFieldMap.hh.

### m_octree

std::unique_ptr<GARFIELD_CLASS_NAME(TetrahedralTree)> Garfield::ComponentFieldMap::m_octree [protected]
Definition at line 436 of file ComponentFieldMap.hh.

### m_pot

std::vector<double> Garfield::ComponentFieldMap::m_pot [protected]
Definition at line 223 of file ComponentFieldMap.hh.

### m_printConvergenceWarnings

bool Garfield::ComponentFieldMap::m_printConvergenceWarnings = true [protected]
Definition at line 313 of file ComponentFieldMap.hh.

### m_setang

std::array<bool, 3> Garfield::ComponentFieldMap::m_setang [protected]
Definition at line 302 of file ComponentFieldMap.hh.

### m_useTetrahedralTree

bool Garfield::ComponentFieldMap::m_useTetrahedralTree = true [protected]
Definition at line 432 of file ComponentFieldMap.hh.

### m_w12

std::vector<std::array<std::array<double, 3>, 4> > Garfield::ComponentFieldMap::m_w12 [protected]
Definition at line 205 of file ComponentFieldMap.hh.

### m_warning

bool Garfield::ComponentFieldMap::m_warning = false [protected]
Definition at line 308 of file ComponentFieldMap.hh.

### m_wfieldCopies

std::map<std::string, WeightingFieldCopy> Garfield::ComponentFieldMap::m_wfieldCopies [protected]
Definition at line 271 of file ComponentFieldMap.hh.

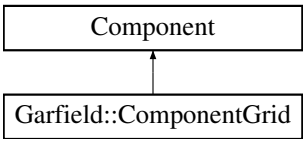
### m_wpot

std::map<std::string, std::vector<double> > Garfield::ComponentFieldMap::m_wpot [protected]
Definition at line 225 of file ComponentFieldMap.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh

## 7.25 Garfield::ComponentGrid Class Reference

Component for interpolating field maps on a regular mesh.
#include <ComponentGrid.hh>
Inheritance diagram for Garfield::ComponentGrid:



Classes

- struct Node

Public Member Functions

- ComponentGrid ()

  Constructor.
- ~ComponentGrid ()

  Destructor.
- bool SetMesh (const unsigned int nx, const unsigned int ny, const unsigned int nz, const double xmin, const double xmax, const double ymin, const double ymax, const double zmin, const double zmax)

  Define the grid.
- bool GetMesh (unsigned int &nx, unsigned int &ny, unsigned int &nz, double &xmin, double &xmax, double &ymin, double &ymax, double &zmin, double &zmax) const

  Retrieve the parameters of the grid.
- void SetCartesianCoordinates ()

  Use Cartesian coordinates (default).
- void SetCylindricalCoordinates ()

  Use cylindrical coordinates.
- bool LoadElectricField (const std::string &filename, const std::string &format, const bool withPotential, const bool withFlag, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import electric field and potential values from a file.
- bool LoadWeightingField (const std::string &filename, const std::string &format, const bool withPotential, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import (prompt) weighting field from file.
- bool LoadWeightingField (const std::string &filename, const std::string &format, const double time, const bool withPotential, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import delayed weighting field from file.
- void SetWeightingFieldOffset (const double x, const double y, const double z)

  Offset coordinates in the weighting field, such that the same numerical weighting field map can be used for electrodes at different positions.
- bool LoadMagneticField (const std::string &filename, const std::string &format, const double scaleX=1., const double scaleB=1.)

  Import magnetic field values from a file.
- bool SaveElectricField (Component *cmp, const std::string &filename, const std::string &fmt)

  Export the electric field and potential of a component to a text file.
- bool SaveWeightingField (Component *cmp, const std::string &id, const std::string &filename, const std::string &fmt)

  Export the weighting field and potential of a component to a text file.
- bool GetElectricField (const unsigned int i, const unsigned int j, const unsigned int k, double &v, double &ex, double &ey, double &ez) const

  Return the field at a given node.
- void SetMedium (Medium *m)

  Set the medium.
- Medium * GetMedium () const

  Get the medium.
- void Print ()

  Print information about the mesh and the available data.
- bool LoadElectronAttachment (const std::string &fname, const std::string &fmt, const unsigned int col, const double scaleX=1.)

  Import electron attachment coefficients from a file.

- bool LoadHoleAttachment (const std::string &fname, const std::string &fmt, const unsigned int col, const double scaleX=1.)

  Import hole attachment coefficients from a file.
- bool LoadElectronVelocity (const std::string &fname, const std::string &fmt, const double scaleX=1., const double scaleV=1.e-9)

  Import a map of electron drift velocities from a file.
- bool LoadHoleVelocity (const std::string &fname, const std::string &fmt, const double scaleX=1., const double scaleV=1.e-9)

  Import a map of hole drift velocities from a file.
- void Clear () override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- void DelayedWeightingField (const double x, const double y, const double z, const double t, double &wx, double &wy, double &wz, const std::string &label) override
- double DelayedWeightingPotential (const double x, const double y, const double z, const double t, const std::string &label) override
- void MagneticField (const double x, const double y, const double z, double &bx, double &by, double &bz, int &status) override
- Medium * GetMedium (const double x, const double y, const double z) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool GetElectricFieldRange (double &exmin, double &exmax, double &eymin, double &eymax, double &ezmin, double &ezmax)
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool HasMagneticField () const override
- bool HasAttachmentMap () const override
- bool ElectronAttachment (const double x, const double y, const double z, double &att) override
- bool HoleAttachment (const double x, const double y, const double z, double &att) override
- bool HasMobilityMap () const override
- bool ElectronMobility (const double x, const double y, const double z, double &mu) override
- bool HoleMobility (const double x, const double y, const double z, double &mu) override
- bool HasVelocityMap () const override
- bool ElectronVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override
- bool HoleVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override

Private Types

- enum class Format {
  Unknown , XY , XZ , XYZ ,
  IJ , IK , IJK , YXZ }
- enum class Coordinates { Cartesian , Cylindrical }

Private Member Functions

- bool LoadMesh (const std::string &filename, std::string format, const double scaleX)

  Read/determine mesh parameters from file.
- bool LoadData (const std::string &filename, std::string format, const bool withPotential, const bool withFlag, const double scaleX, const double scaleF, const double scaleP, std::vector< std::vector< std::vector< Node > > > &field)

  Read electric field and potential from file.
- bool LoadData (const std::string &filename, std::string format, const double scaleX, std::vector< std::vector< std::vector< double > > > &tab, const unsigned int col)

  Load scalar data (e. g. attachment coefficients) from file.
- void Reset () override
- void UpdatePeriodicity () override
- bool GetField (const double x, const double y, const double z, const std::vector< std::vector< std::vector< Node > > > &field, double &fx, double &fy, double &fz, double &p, bool &active)

  Interpolation of the field and potential at a given point.
- bool GetData (const double x, const double y, const double z, const std::vector< std::vector< std::vector< double > > > &table, double &value)

  Interpolation in a table of scalars.
- double Reduce (const double xin, const double xmin, const double xmax, const bool simplePeriodic, const bool mirrorPeriodic, bool &isMirrored) const

  Reduce a coordinate to the basic cell (in case of periodicity).
- void Initialise (std::vector< std::vector< std::vector< Node > > > &fields)

  Set the dimensions of a table according to the mesh.
- Format GetFormat (std::string fmt)

  Decode a format string.

Private Attributes

- Coordinates m_coordinates = Coordinates::Cartesian
- Medium * m_medium = nullptr
- std::vector< std::vector< std::vector< Node > > > m_efields

  Electric field values and potentials.
- std::vector< std::vector< std::vector< Node > > > m_bfields

  Magnetic field values.
- std::vector< std::vector< std::vector< Node > > > m_wfields

  Prompt weighting field values and potentials.
- std::vector< std::vector< std::vector< std::vector< Node > > > > m_wdfields

  Delayed weighting field values and potentials.
- std::vector< std::vector< std::vector< double > > > m_eAttachment

  Attachment maps for electrons and holes.
- std::vector< std::vector< std::vector< double > > > m_hAttachment
- std::vector< std::vector< std::vector< double > > > m_eMobility

  Mobility maps for electrons and holes.
- std::vector< std::vector< std::vector< double > > > m_hMobility
- std::vector< std::vector< std::vector< Node > > > m_eVelocity

  Velocity maps for electrons and holes.
- std::vector< std::vector< std::vector< Node > > > m_hVelocity
- std::vector< std::vector< std::vector< bool > > > m_active

  Active medium flag.
- std::array< unsigned int, 3 > m_nX = {{1, 1, 1}}
- std::array< double, 3 > m_xMin = {{0., 0., 0.}}
- std::array< double, 3 > m_xMax = {{0., 0., 0.}}
- std::array< double, 3 > m_sX = {{0., 0., 0.}}

- bool m_hasMesh = false
- bool m_hasPotential = false
- std::array< double, 3 > m_wFieldOffset = {{0., 0., 0.}}
- double m_pMin = 0.
- double m_pMax = 0.

### 7.25.1 Detailed Description

Component for interpolating field maps on a regular mesh.
Definition at line 14 of file ComponentGrid.hh.

### 7.25.2 Member Enumeration Documentation

#### Coordinates

enum class Garfield::ComponentGrid::Coordinates [strong], [private]

Enumerator

| | |
|---|---|
| Cartesian | |
| Cylindrical | |

Definition at line 197 of file ComponentGrid.hh.
00197 { Cartesian, Cylindrical };

#### Format

enum class Garfield::ComponentGrid::Format [strong], [private]

Enumerator

| | |
|---|---|
| Unknown | |
| XY | |
| XZ | |
| XYZ | |
| IJ | |
| IK | |
| IJK | |
| YXZ | |

Definition at line 196 of file ComponentGrid.hh.
00196 { Unknown, XY, XZ, XYZ, IJ, IK, IJK, YXZ };

### 7.25.3 Constructor & Destructor Documentation

#### ComponentGrid()

Garfield::ComponentGrid::ComponentGrid ()
Constructor.

#### ~ComponentGrid()

Garfield::ComponentGrid::~ComponentGrid () [inline]
Destructor.
Definition at line 19 of file ComponentGrid.hh.
00019 {}

### 7.25.4 Member Function Documentation

#### Clear()

void Garfield::ComponentGrid::Clear () [inline], [override]
Definition at line 139 of file ComponentGrid.hh.
00139 { Reset(); }

#### DelayedWeightingField()

void Garfield::ComponentGrid::DelayedWeightingField (
const double x,
const double y,
const double z,
const double t,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

#### DelayedWeightingPotential()

double Garfield::ComponentGrid::DelayedWeightingPotential (
const double x,
const double y,
const double z,
const double t,
const std::string & label) [override]

#### ElectricField() [1/2]

void Garfield::ComponentGrid::ElectricField (
const double x,
const double y,
const double z,
double & ex,
double & ey,
double & ez,
double & v,
Medium *& m,
int & status) [override]

#### ElectricField() [2/2]

void Garfield::ComponentGrid::ElectricField (
const double x,
const double y,
const double z,
double & ex,
double & ey,
double & ez,
Medium *& m,
int & status) [override]

#### ElectronAttachment()

bool Garfield::ComponentGrid::ElectronAttachment (
const double x,
const double y,
const double z,
double & att) [override]

### ElectronMobility()

```
bool Garfield::ComponentGrid::ElectronMobility (
                const double x,
                const double y,
                const double z,
                double & mu)    [override]
```

### ElectronVelocity()

```
bool Garfield::ComponentGrid::ElectronVelocity (
                const double x,
                const double y,
                const double z,
                double & vx,
                double & vy,
                double & vz)    [override]
```

### GetBoundingBox()

```
bool Garfield::ComponentGrid::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

### GetData()

```
bool Garfield::ComponentGrid::GetData (
                const double x,
                const double y,
                const double z,
                const std::vector< std::vector< std::vector< double > > > & table,
                double & value)    [private]
```

Interpolation in a table of scalars.

### GetElectricField()

```
bool Garfield::ComponentGrid::GetElectricField (
                const unsigned int i,
                const unsigned int j,
                const unsigned int k,
                double & v,
                double & ex,
                double & ey,
                double & ez) const
```

Return the field at a given node.

### GetElectricFieldRange()

```
bool Garfield::ComponentGrid::GetElectricFieldRange (
                double & exmin,
                double & exmax,
                double & eymin,
                double & eymax,
                double & ezmin,
                double & ezmax)
```

### GetElementaryCell()

```
bool Garfield::ComponentGrid::GetElementaryCell (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)   [override]
```

### GetField()

```
bool Garfield::ComponentGrid::GetField (
                const double x,
                const double y,
                const double z,
                const std::vector< std::vector< std::vector< Node > > > & field,
                double & fx,
                double & fy,
                double & fz,
                double & p,
                bool & active)   [private]
```

Interpolation of the field and potential at a given point.

### GetFormat()

```
Format Garfield::ComponentGrid::GetFormat (
                std::string fmt)   [private]
```

Decode a format string.

### GetMedium() [1/2]

```
Medium * Garfield::ComponentGrid::GetMedium () const   [inline]
```

Get the medium.

Definition at line 109 of file ComponentGrid.hh.

```
00109 { return m_medium; }
```

### GetMedium() [2/2]

```
Medium * Garfield::ComponentGrid::GetMedium (
                const double x,
                const double y,
                const double z)   [override]
```

### GetMesh()

```
bool Garfield::ComponentGrid::GetMesh (
                unsigned int & nx,
                unsigned int & ny,
                unsigned int & nz,
                double & xmin,
                double & xmax,
                double & ymin,
                double & ymax,
                double & zmin,
                double & zmax) const
```

Retrieve the parameters of the grid.

### GetVoltageRange()

```
bool Garfield::ComponentGrid::GetVoltageRange (
                double & vmin,
                double & vmax)    [override]
```

### HasAttachmentMap()

```
bool Garfield::ComponentGrid::HasAttachmentMap () const    [inline], [override]
```

Definition at line 172 of file ComponentGrid.hh.

```
00172                                    {
00173     return !(m_eAttachment.empty() && m_hAttachment.empty());
00174   }
```

### HasMagneticField()

```
bool Garfield::ComponentGrid::HasMagneticField () const    [override]
```

### HasMobilityMap()

```
bool Garfield::ComponentGrid::HasMobilityMap () const    [inline], [override]
```

Definition at line 179 of file ComponentGrid.hh.

```
00179                                    {
00180     return !(m_eMobility.empty() && m_hMobility.empty());
00181   }
```

### HasVelocityMap()

```
bool Garfield::ComponentGrid::HasVelocityMap () const    [inline], [override]
```

Definition at line 187 of file ComponentGrid.hh.

```
00187                                    {
00188     return !(m_eVelocity.empty() && m_hVelocity.empty());
00189   }
```

### HoleAttachment()

```
bool Garfield::ComponentGrid::HoleAttachment (
                const double x,
                const double y,
                const double z,
                double & att)    [override]
```

### HoleMobility()

```
bool Garfield::ComponentGrid::HoleMobility (
                const double x,
                const double y,
                const double z,
                double & mu)    [override]
```

### HoleVelocity()

```
bool Garfield::ComponentGrid::HoleVelocity (
                const double x,
                const double y,
                const double z,
                double & vx,
                double & vy,
                double & vz)    [override]
```

Initialise()

void Garfield::ComponentGrid::Initialise (
std::vector< std::vector< std::vector< Node > > > & fields) [private]
Set the dimensions of a table according to the mesh.

LoadData() [1/2]

bool Garfield::ComponentGrid::LoadData (
const std::string & filename,
std::string format,
const bool withPotential,
const bool withFlag,
const double scaleX,
const double scaleF,
const double scaleP,
std::vector< std::vector< std::vector< Node > > > & field) [private]
Read electric field and potential from file.

LoadData() [2/2]

bool Garfield::ComponentGrid::LoadData (
const std::string & filename,
std::string format,
const double scaleX,
std::vector< std::vector< std::vector< double > > > & tab,
const unsigned int col) [private]
Load scalar data (e. g. attachment coefficients) from file.

LoadElectricField()

bool Garfield::ComponentGrid::LoadElectricField (
const std::string & filename,
const std::string & format,
const bool withPotential,
const bool withFlag,
const double scaleX = 1.,
const double scaleE = 1.,
const double scaleP = 1.)
Import electric field and potential values from a file.
The file is supposed to contain one line for each grid point starting with

- either two or three floating point numbers, specifying the coordinates (in cm) of the grid node or
- two or three integers specifying the index of the node,

followed by

- two or three floating point numbers for the electric field (in V/cm), and (depending on the value of withPotential and withFlag),
- a floating point number specifying the potential (in V), and
- an integer flag indicating whether the point is in an active region (1) or not (0).

Format types are:

- ”xy”, ”xz”, ”xyz”: nodes are specified by their coordinates
- ”ij”, ”ik”, ”ijk”: nodes are specified by their indices

If cylindrical coordinates are used, the first coordinate (x) corresponds to the radial distance and the second coordinate (y) corresponds to the azimuth (in radian).

### LoadElectronAttachment()

```
bool Garfield::ComponentGrid::LoadElectronAttachment (
                const std::string & fname,
                const std::string & fmt,
                const unsigned int col,
                const double scaleX = 1.)
```

Import electron attachment coefficients from a file.

Parameters

| | |
|---|---|
| fname | name of the text file. |
| fmt | format string, see LoadElectricField. |
| col | column in the file which has the attachment coefficient. |
| scaleX | scaling factor to be applied to the coordinates. |

### LoadElectronVelocity()

```
bool Garfield::ComponentGrid::LoadElectronVelocity (
                const std::string & fname,
                const std::string & fmt,
                const double scaleX = 1.,
                const double scaleV = 1.e-9)
```

Import a map of electron drift velocities from a file.

Parameters

| | |
|---|---|
| fname | name of the text file. |
| fmt | format string, see LoadElectricField |
| scaleX | scaling factor to be applied to the coordinates. |
| scaleV | scaling factor to be applied to the velocity components. |

### LoadHoleAttachment()

```
bool Garfield::ComponentGrid::LoadHoleAttachment (
                const std::string & fname,
                const std::string & fmt,
                const unsigned int col,
                const double scaleX = 1.)
```

Import hole attachment coefficients from a file.

### LoadHoleVelocity()

```
bool Garfield::ComponentGrid::LoadHoleVelocity (
                const std::string & fname,
                const std::string & fmt,
                const double scaleX = 1.,
                const double scaleV = 1.e-9)
```

Import a map of hole drift velocities from a file.

### LoadMagneticField()

```
bool Garfield::ComponentGrid::LoadMagneticField (
                const std::string & filename,
                const std::string & format,
```

const double scaleX = 1.,
const double scaleB = 1.)

Import magnetic field values from a file.

LoadMesh()

bool Garfield::ComponentGrid::LoadMesh (
const std::string & filename,
std::string format,
const double scaleX) [private]

Read/determine mesh parameters from file.

LoadWeightingField() [1/2]

bool Garfield::ComponentGrid::LoadWeightingField (
const std::string & filename,
const std::string & format,
const bool withPotential,
const double scaleX = 1.,
const double scaleE = 1.,
const double scaleP = 1.)

Import (prompt) weighting field from file.

LoadWeightingField() [2/2]

bool Garfield::ComponentGrid::LoadWeightingField (
const std::string & filename,
const std::string & format,
const double time,
const bool withPotential,
const double scaleX = 1.,
const double scaleE = 1.,
const double scaleP = 1.)

Import delayed weighting field from file.

MagneticField()

void Garfield::ComponentGrid::MagneticField (
const double x,
const double y,
const double z,
double & bx,
double & by,
double & bz,
int & status) [override]

Print()

void Garfield::ComponentGrid::Print ()

Print information about the mesh and the available data.

Reduce()

double Garfield::ComponentGrid::Reduce (
const double xin,
const double xmin,
const double xmax,
const bool simplePeriodic,
const bool mirrorPeriodic,
bool & isMirrored) const [private]

Reduce a coordinate to the basic cell (in case of periodicity).

### Reset()

void Garfield::ComponentGrid::Reset () [override], [private]

### SaveElectricField()

bool Garfield::ComponentGrid::SaveElectricField (
Component * cmp,
const std::string & filename,
const std::string & fmt)

Export the electric field and potential of a component to a text file.

Parameters

| | |
|---|---|
| cmp | Component object for which to export the field/potential |
| filename | name of the text file |
| fmt | format string, see LoadElectricField |

### SaveWeightingField()

bool Garfield::ComponentGrid::SaveWeightingField (
Component * cmp,
const std::string & id,
const std::string & filename,
const std::string & fmt)

Export the weighting field and potential of a component to a text file.

Parameters

| | |
|---|---|
| cmp | Component object for which to export the field/potential |
| id | identifier of the weighting field |
| filename | name of the text file |
| fmt | format string, see LoadElectricField |

### SetCartesianCoordinates()

void Garfield::ComponentGrid::SetCartesianCoordinates () [inline]

Use Cartesian coordinates (default).

Definition at line 36 of file ComponentGrid.hh.

```
00036 { m_coordinates = Coordinates::Cartesian; }
```

### SetCylindricalCoordinates()

void Garfield::ComponentGrid::SetCylindricalCoordinates ()

Use cylindrical coordinates.

### SetMedium()

void Garfield::ComponentGrid::SetMedium (
Medium * m)

Set the medium.

SetMesh()

```
bool Garfield::ComponentGrid::SetMesh (
                const unsigned int nx,
                const unsigned int ny,
                const unsigned int nz,
                const double xmin,
                const double xmax,
                const double ymin,
                const double ymax,
                const double zmin,
                const double zmax)
```

Define the grid.

Parameters

| | |
|---|---|
| nx,ny,nz | number of nodes along $x, y, z$. |
| xmin,xmax | range along $x$. |
| ymin,ymax | range along $y$. |
| zmin,zmax | range along $z$. |

SetWeightingFieldOffset()

```
void Garfield::ComponentGrid::SetWeightingFieldOffset (
                const double x,
                const double y,
                const double z)
```

Offset coordinates in the weighting field, such that the same numerical weighting field map can be used for electrodes at different positions.

UpdatePeriodicity()

```
void Garfield::ComponentGrid::UpdatePeriodicity ()  [override], [private]
```

WeightingField()

```
void Garfield::ComponentGrid::WeightingField (
                const double x,
                const double y,
                const double z,
                double & wx,
                double & wy,
                double & wz,
                const std::string & label)  [override]
```

WeightingPotential()

```
double Garfield::ComponentGrid::WeightingPotential (
                const double x,
                const double y,
                const double z,
                const std::string & label)  [override]
```

### 7.25.5 Member Data Documentation

m_active

```
std::vector<std::vector<std::vector<bool> > > Garfield::ComponentGrid::m_active  [private]
```

Active medium flag.

Definition at line 224 of file ComponentGrid.hh.

### m_bfields

std::vector<std::vector<std::vector<Node> > > Garfield::ComponentGrid::m_bfields [private]
Magnetic field values.
Definition at line 209 of file ComponentGrid.hh.

### m_coordinates

Coordinates Garfield::ComponentGrid::m_coordinates = Coordinates::Cartesian [private]
Definition at line 198 of file ComponentGrid.hh.

### m_eAttachment

std::vector<std::vector<std::vector<double> > > Garfield::ComponentGrid::m_eAttachment [private]
Attachment maps for electrons and holes.
Definition at line 215 of file ComponentGrid.hh.

### m_efields

std::vector<std::vector<std::vector<Node> > > Garfield::ComponentGrid::m_efields [private]
Electric field values and potentials.
Definition at line 207 of file ComponentGrid.hh.

### m_eMobility

std::vector<std::vector<std::vector<double> > > Garfield::ComponentGrid::m_eMobility [private]
Mobility maps for electrons and holes.
Definition at line 218 of file ComponentGrid.hh.

### m_eVelocity

std::vector<std::vector<std::vector<Node> > > Garfield::ComponentGrid::m_eVelocity [private]
Velocity maps for electrons and holes.
Definition at line 221 of file ComponentGrid.hh.

### m_hasMesh

bool Garfield::ComponentGrid::m_hasMesh = false [private]
Definition at line 232 of file ComponentGrid.hh.

### m_hasPotential

bool Garfield::ComponentGrid::m_hasPotential = false [private]
Definition at line 233 of file ComponentGrid.hh.

### m_hAttachment

std::vector<std::vector<std::vector<double> > > Garfield::ComponentGrid::m_hAttachment [private]
Definition at line 216 of file ComponentGrid.hh.

### m_hMobility

std::vector<std::vector<std::vector<double> > > Garfield::ComponentGrid::m_hMobility [private]
Definition at line 219 of file ComponentGrid.hh.

### m_hVelocity

std::vector<std::vector<std::vector<Node> > > Garfield::ComponentGrid::m_hVelocity [private]
Definition at line 222 of file ComponentGrid.hh.

### m_medium

Medium* Garfield::ComponentGrid::m_medium = nullptr [private]
Definition at line 200 of file ComponentGrid.hh.

### m_nX

std::array<unsigned int, 3> Garfield::ComponentGrid::m_nX = {{1, 1, 1}} [private]
Definition at line 227 of file ComponentGrid.hh.
00227 {{1, 1, 1}};

### m_pMax

double Garfield::ComponentGrid::m_pMax = 0. [private]
Definition at line 239 of file ComponentGrid.hh.

### m_pMin

double Garfield::ComponentGrid::m_pMin = 0. [private]
Definition at line 239 of file ComponentGrid.hh.

### m_sX

std::array<double, 3> Garfield::ComponentGrid::m_sX = {{0., 0., 0.}} [private]
Definition at line 230 of file ComponentGrid.hh.
00230 {{0., 0., 0.}};

### m_wdfields

std::vector<std::vector<std::vector<std::vector<Node> > > > Garfield::ComponentGrid::m_wdfields [private]
Delayed weighting field values and potentials.
Definition at line 213 of file ComponentGrid.hh.

### m_wFieldOffset

std::array<double, 3> Garfield::ComponentGrid::m_wFieldOffset = {{0., 0., 0.}} [private]
Definition at line 236 of file ComponentGrid.hh.
00236 {{0., 0., 0.}};

### m_wfields

std::vector<std::vector<std::vector<Node> > > Garfield::ComponentGrid::m_wfields [private]
Prompt weighting field values and potentials.
Definition at line 211 of file ComponentGrid.hh.

### m_xMax

std::array<double, 3> Garfield::ComponentGrid::m_xMax = {{0., 0., 0.}} [private]
Definition at line 229 of file ComponentGrid.hh.
00229 {{0., 0., 0.}};

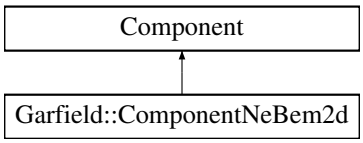
### m_xMin

std::array<double, 3> Garfield::ComponentGrid::m_xMin = {{0., 0., 0.}} [private]
Definition at line 228 of file ComponentGrid.hh.
00228 {{0., 0., 0.}};
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentGrid.hh

## 7.26 Garfield::ComponentNeBem2d Class Reference

Two-dimensional implementation of the nearly exact Boundary Element Method.
#include <ComponentNeBem2d.hh>
Inheritance diagram for Garfield::ComponentNeBem2d:




Classes

- struct Element
- struct Region
- struct Segment
- struct SpaceCharge
- struct Wire

Public Member Functions

- ComponentNeBem2d ()

  Constructor.
- ~ComponentNeBem2d ()

  Destructor.
- void SetMedium (Medium *medium)

  Set the "background" medium.
- bool AddSegment (const double x0, const double y0, const double x1, const double y1, const double v, const int ndiv=-1)

  Add a conducting straight-line segment.
- bool AddWire (const double x, const double y, const double d, const double v, const int ntrap=5)

  Add a wire.
- bool AddRegion (const std::vector< double > &xp, const std::vector< double > &yp, Medium *medium, const unsigned int bctype=4, const double v=0., const int ndiv=-1)

  Add a region bounded by a polygon.
- void AddChargeDistribution (const double x, const double y, const double a, const double b, const double rho)
- void SetRangeZ (const double zmin, const double zmax)

  Set the extent of the drift region along z.
- bool Initialise ()

  Discretise the geometry and compute the solution.
- void SetNumberOfDivisions (const unsigned int ndiv)

  Set the default number of elements per segment.
- void SetNumberOfCollocationPoints (const unsigned int ncoll)
- void EnableAutoResizing (const bool on=true)
- void EnableRandomCollocation (const bool on=true)
- void SetMaxNumberOfIterations (const unsigned int niter)
- unsigned int GetNumberOfRegions () const

  Return the number of regions.
- bool GetRegion (const unsigned int i, std::vector< double > &xv, std::vector< double > &yv, Medium *&medium, unsigned int &bctype, double &v)

  Return the properties of a given region.
- unsigned int GetNumberOfSegments () const

Return the number of conducting straight-line segments.

- bool GetSegment (const unsigned int i, double &x0, double &y0, double &x1, double &x2, double &v) const

  Return the coordinates and voltage of a given straight-line segment.
- unsigned int GetNumberOfWires () const

  Return the number of wires.
- bool GetWire (const unsigned int i, double &x, double &y, double &d, double &v, double &q) const

  Return the coordinates, diameter, potential and charge of a given wire.
- size_t GetNumberOfElements () const override

  Return the number of boundary elements.
- bool GetElement (const unsigned int i, double &x0, double &y0, double &x1, double &y1, double &q) const

  Return the coordinates and charge of a given boundary element.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool CrossedWire (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, double &xc, double &yc, double &zc, const bool centre, double &rc) override
- bool InTrapRadius (const double q0, const double x0, const double y0, const double z0, double &xw, double &yx, double &rw) override

Private Types

- enum BC { Voltage = 1 , Charge , Floating , Dielectric }

  Boundary condition type.

Private Member Functions

- void EliminateOverlaps (std::vector< Segment > &segments)

  Split/merge overlapping segments.
- bool Discretise (const Segment &segment, std::vector< Element > &elements, const double lambda, const unsigned int ndiv)

  Create elements from a straight-line segment.
- bool ComputeInfluenceMatrix (std::vector< std::vector< double > > &infmat) const
- bool InvertMatrix (std::vector< std::vector< double > > &influenceMatrix, std::vector< std::vector< double > > &inverseMatrix) const
- bool LUDecomposition (std::vector< std::vector< double > > &mat, std::vector< int > &index) const
- void LUSubstitution (const std::vector< std::vector< double > > &mat, const std::vector< int > &index, std::vector< double > &col) const
- bool Solve (const std::vector< std::vector< double > > &inverseMatrix, const std::vector< double > &bc)
- bool CheckConvergence (const double tol, std::vector< bool > &ok)
- void SplitElement (Element &oldElement, std::vector< Element > &elements)
- double LinePotential (const double a, const double x, const double y) const

  Potential of a line segment (half-width a) in local coordinates.
- double WirePotential (const double r0, const double x, const double y) const

  Potential of a thin wire with radius r0.

- void LineField (const double a, const double x, const double y, double &ex, double &ey) const

  Field of a line segment (half-width a) in local coordinates.
- void WireField (const double r0, const double x, const double y, double &ex, double &ey) const

  Field of a thin wire with radius r0.
- double BoxPotential (const double a, const double b, const double x, const double y, const double v0) const

  Potential of a uniformly charged rectangle.
- void BoxField (const double a, const double b, const double x, const double y, double &ex, double &ey) const

  Field of a uniformly charged rectangle.
- void Reset () override
- void UpdatePeriodicity () override
- void ToLocal (const double xIn, const double yIn, const double cphi, const double sphi, double &xOut, double &yOut) const

  Rotation from global to local coordinates.
- void ToGlobal (const double xIn, const double yIn, const double cphi, const double sphi, double &xOut, double &yOut) const

  Rotation from local to global coordinates.
- int Field (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, const bool opt)

  Evaluation of the electric field (and optionally potential).

Private Attributes

- unsigned int m_nDivisions = 5

  Default number elements per segment.
- unsigned int m_nCollocationPoints = 1
- bool m_autoSize = false
- bool m_randomCollocation = false
- unsigned int m_nMaxIterations = 3
- Medium * m_medium = nullptr

  Background medium.
- bool m_useRangeZ = false

  Flag whether a z-range has been defined by the user.
- double m_zmin = -1.

  Lower z limit.
- double m_zmax = 1.

  Upper z limit.
- std::vector< Region > m_regions

  Regions.
- std::vector< Segment > m_segments

  User-specified conducting straight-line segments.
- std::vector< Wire > m_wires

  Wires.
- std::vector< Element > m_elements

  Straight-line boundary elements.
- std::vector< SpaceCharge > m_spaceCharge

Static Private Attributes

- static const double InvEpsilon0
- static const double InvTwoPiEpsilon0

### 7.26.1 Detailed Description

Two-dimensional implementation of the nearly exact Boundary Element Method.
Definition at line 12 of file ComponentNeBem2d.hh.

### 7.26.2 Member Enumeration Documentation

BC

enum Garfield::ComponentNeBem2d::BC [private]
Boundary condition type.

Enumerator

| | |
|---|---|
| Voltage | Fixed potential. |
| Charge | Fixed charge density (not implemented). |
| Floating | Floating conductor (not implemented). |
| Dielectric | Dielectric-dielectric interface. |

Definition at line 131 of file ComponentNeBem2d.hh.

```
00131         {
00132     Voltage = 1,
00133     Charge,
00134     Floating,
00135     Dielectric
00136   };
```

### 7.26.3 Constructor & Destructor Documentation

ComponentNeBem2d()

Garfield::ComponentNeBem2d::ComponentNeBem2d ()
Constructor.

~ComponentNeBem2d()

Garfield::ComponentNeBem2d::~ComponentNeBem2d () [inline]
Destructor.
Definition at line 17 of file ComponentNeBem2d.hh.

```
00017 {}
```

### 7.26.4 Member Function Documentation

AddChargeDistribution()

```
void Garfield::ComponentNeBem2d::AddChargeDistribution (
                const double x,
                const double y,
                const double a,
                const double b,
                const double rho)
```

AddRegion()

```
bool Garfield::ComponentNeBem2d::AddRegion (
                const std::vector< double > & xp,
                const std::vector< double > & yp,
                Medium * medium,
                const unsigned int bctype = 4,
                const double v = 0.,
                const int ndiv = -1)
```

Add a region bounded by a polygon.

Parameters

| | |
|---|---|
| xp,yp | x/y-coordinates of the vertices of the polygon. |
| medium | pointer to the medium associated to the region. |
| bctype | 1: fixed voltage, 4: dielectric-dielectric interface. |
| v | applied potential. |
| ndiv | number of elements on each edge segment. |

AddSegment()

```
bool Garfield::ComponentNeBem2d::AddSegment (
                const double x0,
                const double y0,
                const double x1,
                const double y1,
                const double v,
                const int ndiv = -1)
```

Add a conducting straight-line segment.

Parameters

| | |
|---|---|
| x0,y0,x1,y1 | coordinates of start and end point. |
| v | applied potential. |
| ndiv | number of elements in which to split the segment. |

AddWire()

```
bool Garfield::ComponentNeBem2d::AddWire (
                const double x,
                const double y,
                const double d,
                const double v,
                const int ntrap = 5)
```

Add a wire.

Parameters

| | |
|---|---|
| x,y | centre of the wire. |
| d | wire diameter. |
| v | applied potential. |
| ntrap | multiple of the wire radius within which a particle is considered to be trapped by the wire. |

BoxField()

```
void Garfield::ComponentNeBem2d::BoxField (
                const double a,
                const double b,
                const double x,
                const double y,
                double & ex,
                double & ey) const   [private]
```

Field of a uniformly charged rectangle.

### BoxPotential()

```
double Garfield::ComponentNeBem2d::BoxPotential (
            const double a,
            const double b,
            const double x,
            const double y,
            const double v0) const    [private]
```

Potential of a uniformly charged rectangle.

### CheckConvergence()

```
bool Garfield::ComponentNeBem2d::CheckConvergence (
            const double tol,
            std::vector< bool > & ok)    [private]
```

### ComputeInfluenceMatrix()

```
bool Garfield::ComponentNeBem2d::ComputeInfluenceMatrix (
            std::vector< std::vector< double > > & infmat) const    [private]
```

### CrossedWire()

```
bool Garfield::ComponentNeBem2d::CrossedWire (
            const double x0,
            const double y0,
            const double z0,
            const double x1,
            const double y1,
            const double z1,
            double & xc,
            double & yc,
            double & zc,
            const bool centre,
            double & rc)    [override]
```

### Discretise()

```
bool Garfield::ComponentNeBem2d::Discretise (
            const Segment & segment,
            std::vector< Element > & elements,
            const double lambda,
            const unsigned int ndiv)    [private]
```

Create elements from a straight-line segment.

### ElectricField() [1/2]

```
void Garfield::ComponentNeBem2d::ElectricField (
            const double x,
            const double y,
            const double z,
            double & ex,
            double & ey,
            double & ez,
            double & v,
            Medium *& m,
            int & status)    [override]
```

### ElectricField() [2/2]

```
void Garfield::ComponentNeBem2d::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)   [override]
```

### EliminateOverlaps()

```
void Garfield::ComponentNeBem2d::EliminateOverlaps (
                std::vector< Segment > & segments)   [private]
```

Split/merge overlapping segments.

### EnableAutoResizing()

```
void Garfield::ComponentNeBem2d::EnableAutoResizing (
                const bool on = true)   [inline]
```

Definition at line 60 of file ComponentNeBem2d.hh.

```
00060 { m_autoSize = on; }
```

### EnableRandomCollocation()

```
void Garfield::ComponentNeBem2d::EnableRandomCollocation (
                const bool on = true)   [inline]
```

Definition at line 61 of file ComponentNeBem2d.hh.

```
00061                                       {
00062     m_randomCollocation = on;
00063   }
```

### Field()

```
int Garfield::ComponentNeBem2d::Field (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                const bool opt)   [private]
```

Evaluation of the electric field (and optionally potential).

### GetBoundingBox()

```
bool Garfield::ComponentNeBem2d::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)   [override]
```

### GetElement()

```
bool Garfield::ComponentNeBem2d::GetElement (
                const unsigned int i,
```

double & x0,
double & y0,
double & x1,
double & y1,
double & q) const

Return the coordinates and charge of a given boundary element.

### GetElementaryCell()

bool Garfield::ComponentNeBem2d::GetElementaryCell (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [override]

### GetMedium()

Medium * Garfield::ComponentNeBem2d::GetMedium (
const double x,
const double y,
const double z) [override]

### GetNumberOfElements()

size_t Garfield::ComponentNeBem2d::GetNumberOfElements () const [inline], [override]

Return the number of boundary elements.

Definition at line 83 of file ComponentNeBem2d.hh.

```
00083 { return m_elements.size(); }
```

### GetNumberOfRegions()

unsigned int Garfield::ComponentNeBem2d::GetNumberOfRegions () const [inline]

Return the number of regions.

Definition at line 67 of file ComponentNeBem2d.hh.

```
00067 { return m_regions.size(); }
```

### GetNumberOfSegments()

unsigned int Garfield::ComponentNeBem2d::GetNumberOfSegments () const [inline]

Return the number of conducting straight-line segments.

Definition at line 73 of file ComponentNeBem2d.hh.

```
00073 { return m_segments.size(); }
```

### GetNumberOfWires()

unsigned int Garfield::ComponentNeBem2d::GetNumberOfWires () const [inline]

Return the number of wires.

Definition at line 78 of file ComponentNeBem2d.hh.

```
00078 { return m_wires.size(); }
```

### GetRegion()

bool Garfield::ComponentNeBem2d::GetRegion (
const unsigned int i,
std::vector< double > & xv,
std::vector< double > & yv,
Medium *& medium,
unsigned int & bctype,
double & v)

Return the properties of a given region.

GetSegment()

```
bool Garfield::ComponentNeBem2d::GetSegment (
                const unsigned int i,
                double & x0,
                double & y0,
                double & x1,
                double & x2,
                double & v) const
```

Return the coordinates and voltage of a given straight-line segment.

GetVoltageRange()

```
bool Garfield::ComponentNeBem2d::GetVoltageRange (
                double & vmin,
                double & vmax)    [override]
```

GetWire()

```
bool Garfield::ComponentNeBem2d::GetWire (
                const unsigned int i,
                double & x,
                double & y,
                double & d,
                double & v,
                double & q) const
```

Return the coordinates, diameter, potential and charge of a given wire.

Initialise()

```
bool Garfield::ComponentNeBem2d::Initialise ()
```

Discretise the geometry and compute the solution.

InTrapRadius()

```
bool Garfield::ComponentNeBem2d::InTrapRadius (
                const double q0,
                const double x0,
                const double y0,
                const double z0,
                double & xw,
                double & yx,
                double & rw)    [override]
```

InvertMatrix()

```
bool Garfield::ComponentNeBem2d::InvertMatrix (
                std::vector< std::vector< double > > & influenceMatrix,
                std::vector< std::vector< double > > & inverseMatrix) const    [private]
```

LineField()

```
void Garfield::ComponentNeBem2d::LineField (
                const double a,
                const double x,
                const double y,
                double & ex,
                double & ey) const    [private]
```

Field of a line segment (half-width a) in local coordinates.

### LinePotential()

```
double Garfield::ComponentNeBem2d::LinePotential (
            const double a,
            const double x,
            const double y) const   [private]
```

Potential of a line segment (half-width a) in local coordinates.

### LUDecomposition()

```
bool Garfield::ComponentNeBem2d::LUDecomposition (
            std::vector< std::vector< double > > & mat,
            std::vector< int > & index) const   [private]
```

### LUSubstitution()

```
void Garfield::ComponentNeBem2d::LUSubstitution (
            const std::vector< std::vector< double > > & mat,
            const std::vector< int > & index,
            std::vector< double > & col) const   [private]
```

### Reset()

```
void Garfield::ComponentNeBem2d::Reset ()   [override], [private]
```

### SetMaxNumberOfIterations()

```
void Garfield::ComponentNeBem2d::SetMaxNumberOfIterations (
            const unsigned int niter)
```

### SetMedium()

```
void Garfield::ComponentNeBem2d::SetMedium (
            Medium * medium)   [inline]
```

Set the "background" medium.
Definition at line 20 of file ComponentNeBem2d.hh.

```
00020 { m_medium = medium; }
```

### SetNumberOfCollocationPoints()

```
void Garfield::ComponentNeBem2d::SetNumberOfCollocationPoints (
            const unsigned int ncoll)
```

### SetNumberOfDivisions()

```
void Garfield::ComponentNeBem2d::SetNumberOfDivisions (
            const unsigned int ndiv)
```

Set the default number of elements per segment.

### SetRangeZ()

```
void Garfield::ComponentNeBem2d::SetRangeZ (
            const double zmin,
            const double zmax)
```

Set the extent of the drift region along z.

### Solve()

```
bool Garfield::ComponentNeBem2d::Solve (
            const std::vector< std::vector< double > > & inverseMatrix,
            const std::vector< double > & bc)   [private]
```

### SplitElement()

void Garfield::ComponentNeBem2d::SplitElement (
Element & oldElement,
std::vector< Element > & elements) [private]

### ToGlobal()

void Garfield::ComponentNeBem2d::ToGlobal (
const double xIn,
const double yIn,
const double cphi,
const double sphi,
double & xOut,
double & yOut) const [private]

Rotation from local to global coordinates.

### ToLocal()

void Garfield::ComponentNeBem2d::ToLocal (
const double xIn,
const double yIn,
const double cphi,
const double sphi,
double & xOut,
double & yOut) const [private]

Rotation from global to local coordinates.

### UpdatePeriodicity()

void Garfield::ComponentNeBem2d::UpdatePeriodicity () [override], [private]

### WireField()

void Garfield::ComponentNeBem2d::WireField (
const double r0,
const double x,
const double y,
double & ex,
double & ey) const [private]

Field of a thin wire with radius r0.

### WirePotential()

double Garfield::ComponentNeBem2d::WirePotential (
const double r0,
const double x,
const double y) const [private]

Potential of a thin wire with radius r0.

## 7.26.5 Member Data Documentation

### InvEpsilon0

const double Garfield::ComponentNeBem2d::InvEpsilon0 [static], [private]

Definition at line 112 of file ComponentNeBem2d.hh.

### InvTwoPiEpsilon0

const double Garfield::ComponentNeBem2d::InvTwoPiEpsilon0 [static], [private]

Definition at line 113 of file ComponentNeBem2d.hh.

### m_autoSize

bool Garfield::ComponentNeBem2d::m_autoSize = false [private]
Definition at line 118 of file ComponentNeBem2d.hh.

### m_elements

std::vector<Element> Garfield::ComponentNeBem2d::m_elements [private]
Straight-line boundary elements.
Definition at line 179 of file ComponentNeBem2d.hh.

### m_medium

Medium* Garfield::ComponentNeBem2d::m_medium = nullptr [private]
Background medium.
Definition at line 123 of file ComponentNeBem2d.hh.

### m_nCollocationPoints

unsigned int Garfield::ComponentNeBem2d::m_nCollocationPoints = 1 [private]
Definition at line 117 of file ComponentNeBem2d.hh.

### m_nDivisions

unsigned int Garfield::ComponentNeBem2d::m_nDivisions = 5 [private]
Default number elements per segment.
Definition at line 116 of file ComponentNeBem2d.hh.

### m_nMaxIterations

unsigned int Garfield::ComponentNeBem2d::m_nMaxIterations = 3 [private]
Definition at line 120 of file ComponentNeBem2d.hh.

### m_randomCollocation

bool Garfield::ComponentNeBem2d::m_randomCollocation = false [private]
Definition at line 119 of file ComponentNeBem2d.hh.

### m_regions

std::vector<Region> Garfield::ComponentNeBem2d::m_regions [private]
Regions.
Definition at line 146 of file ComponentNeBem2d.hh.

### m_segments

std::vector Garfield::ComponentNeBem2d::m_segments [private]
User-specified conducting straight-line segments.
Definition at line 157 of file ComponentNeBem2d.hh.

### m_spaceCharge

std::vector<SpaceCharge> Garfield::ComponentNeBem2d::m_spaceCharge [private]
Definition at line 187 of file ComponentNeBem2d.hh.

### m_useRangeZ

bool Garfield::ComponentNeBem2d::m_useRangeZ = false [private]
Flag whether a z-range has been defined by the user.
Definition at line 125 of file ComponentNeBem2d.hh.

m_wires

std::vector<Wire> Garfield::ComponentNeBem2d::m_wires [private]
Wires.
Definition at line 167 of file ComponentNeBem2d.hh.

m_zmax

double Garfield::ComponentNeBem2d::m_zmax = 1. [private]
Upper z limit.
Definition at line 129 of file ComponentNeBem2d.hh.

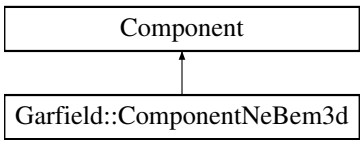
m_zmin

double Garfield::ComponentNeBem2d::m_zmin = -1. [private]
Lower z limit.
Definition at line 127 of file ComponentNeBem2d.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh

## 7.27 Garfield::ComponentNeBem3d Class Reference

Interface to neBEM.
#include <ComponentNeBem3d.hh>
Inheritance diagram for Garfield::ComponentNeBem3d:




Classes

- struct Element
- struct Primitive

Public Member Functions

- ComponentNeBem3d ()
  Constructor.
- ~ComponentNeBem3d ()
  Destructor.
- Medium * GetMedium (const double x, const double y, const double z) override
- void AddPlaneX (const double x, const double voltage)
  Add a plane at constant x.
- void AddPlaneY (const double y, const double voltage)
  Add a plane at constant y.
- void AddPlaneZ (const double z, const double voltage)
  Add a plane at constant z.
- unsigned int GetNumberOfPlanesX () const
  Get the number of equipotential planes at constant x.
- unsigned int GetNumberOfPlanesY () const
  Get the number of equipotential planes at constant y.
- unsigned int GetNumberOfPlanesZ () const
  Get the number of equipotential planes at constant z.

- bool GetPlaneX (const unsigned int i, double &x, double &v) const

  Retrieve the parameters of a plane at constant x.
- bool GetPlaneY (const unsigned int i, double &y, double &v) const

  Retrieve the parameters of a plane at constant y.
- bool GetPlaneZ (const unsigned int i, double &z, double &v) const

  Retrieve the parameters of a plane at constant z.
- unsigned int GetNumberOfPrimitives () const
- bool GetPrimitive (const unsigned int i, double &a, double &b, double &c, std::vector< double > &xv, std::vector< double > &yv, std::vector< double > &zv, int &interface, double &v, double &q, double &lambda) const
- bool GetPrimitive (const unsigned int i, double &a, double &b, double &c, std::vector< double > &xv, std::vector< double > &yv, std::vector< double > &zv, int &vol1, int &vol2) const
- bool GetVolume (const unsigned int vol, int &shape, int &material, double &eps, double &potential, double &charge, int &bc)
- int GetVolume (const double x, const double y, const double z)
- size_t GetNumberOfElements () const override
- bool GetElement (const unsigned int i, std::vector< double > &xv, std::vector< double > &yv, std::vector< double > &zv, int &interface, double &bc, double &lambda) const
- bool Initialise ()

  Retrieve surface panels, remove contacts and cut polygons to rectangles and right-angle triangles.
- void SetTargetElementSize (const double length)

  Set the default value of the target linear size of the elements produced by neBEM's discretisation process.
- void SetMinMaxNumberOfElements (const unsigned int nmin, const unsigned int nmax)

  Set the smallest and largest allowed number of elements along the lenght of a primitive.
- void SetNewModel (const unsigned int NewModel)
- void SetNewMesh (const unsigned int NewMesh)
- void SetNewBC (const unsigned int NewBC)
- void SetNewPP (const unsigned int NewPP)
- void SetModelOptions (const unsigned int NewModel, const unsigned int NewMesh, const unsigned int NewBC, const unsigned int NewPP)
- void SetStoreInflMatrix (const unsigned int OptStoreInflMatrix)

  Set storing options (OptStoreInflMatrix, OptStoreInvMatrix, OptStoreInvMatrix, OptStoreInvMatrix) OptStorePrimitives, OptStorePrimitives) OptStoreElements, OptStoreElements) OptFormattedFile, OptUnformattedFile)
- void SetReadInflMatrix (const unsigned int OptReadInflMatrix)
- void SetStoreInvMatrix (const unsigned int OptStoreInvMatrix)
- void SetReadInvMatrix (const unsigned int OptReadInvMatrix)
- void SetStorePrimitives (const unsigned int OptStorePrimitives)
- void SetReadPrimitives (const unsigned int OptReadPrimitives)
- void SetStoreElements (const unsigned int OptStoreElements)
- void SetReadElements (const unsigned int OptReadElements)
- void SetFormattedFile (const unsigned int OptFormattedFile)
- void SetUnformattedFile (const unsigned int OptUnformattedFile)
- void SetStoreReadOptions (const unsigned int OptStoreInflMatrix, const unsigned int OptRead↩ InflMatrix, const unsigned int OptStoreInvMatrix, const unsigned int OptReadInvMatrix, const unsigned int OptStorePrimitives, const unsigned int OptReadPrimitives, const unsigned int Opt↩ StoreElements, const unsigned int OptReadElements, const unsigned int OptFormattedFile, const unsigned int OptUnformattedFile)
- void SetReuseModel (void)
- void SetSystemChargeZero (const unsigned int OptSystemChargeZero)

  Other functions to be, are void SetPlotOptions(OptGnuplot=0, OptGnuplotPrimitives=0, OptGnuplot↩ Elements=0, OptPrimitiveFiles=0, OptElementFiles=0)
- void SetValidateSolution (const unsigned int OptValidateSolution)
- void SetForceValidation (const unsigned int OptForceValidation)

- void SetRepeatLHMatrix (const unsigned int OptRepeatLHMatrix)
- void SetComputeOptions (const unsigned int OptSystemChargeZero, const unsigned int OptValidateSolution, const unsigned int OptForceValidation, const unsigned int OptRepeatLHMatrix)
- void SetFastVolOptions (const unsigned int OptFastVol, const unsigned int OptCreateFastPF, const unsigned int OptReadFastPF)
- void SetFastVolVersion (const unsigned int VersionFV)
- void SetFastVolBlocks (const unsigned int NbBlocksFV)
- void SetWtFldFastVolOptions (const unsigned int IdWtField, const unsigned int OptWtFldFastVol, const unsigned int OptCreateWtFldFastPF, const unsigned int OptReadWtFldFastPF)
- void SetWtFldFastVolVersion (const unsigned int IdWtField, const unsigned int VersionWtFldFV)
- void SetWtFldFastVolBlocks (const unsigned int IdWtField, const unsigned int NbBlocksWtFldFV)
- void SetKnownChargeOptions (const unsigned int OptKnownCharge)
- void SetChargingUpOptions (const unsigned int OptChargingUp)
- void UseLUInversion ()

  Invert the influence matrix using lower-upper (LU) decomposition.
- void UseSVDInversion ()

  Invert the influence matrix using singular value decomposition.
- void SetPeriodicCopies (const unsigned int nx, const unsigned int ny, const unsigned int nz)

  Set the parameters $n_x, n_y, n_z$ defining the number of periodic copies that neBEM will use when dealing with periodic configurations.
- void GetPeriodicCopies (unsigned int &nx, unsigned int &ny, unsigned int &nz) const

  Retrieve the number of periodic copies used by neBEM.
- void SetPeriodicityX (const double s)

  Set the periodic length [cm] in the x-direction.
- void SetPeriodicityY (const double s)

  Set the periodic length [cm] in the y-direction.
- void SetPeriodicityZ (const double s)

  Set the periodic length [cm] in the z-direction.
- void SetMirrorPeriodicityX (const double s)

  Set the periodic length [cm] in the x-direction.
- void SetMirrorPeriodicityY (const double s)

  Set the periodic length [cm] in the y-direction.
- void SetMirrorPeriodicityZ (const double s)

  Set the periodic length [cm] in the z-direction.
- bool GetPeriodicityX (double &s) const

  Get the periodic length in the x-direction.
- bool GetPeriodicityY (double &s) const

  Get the periodic length in the y-direction.
- bool GetPeriodicityZ (double &s) const

  Get the periodic length in the z-direction.
- void SetNumberOfThreads (const unsigned int n)

  Set the number of threads to be used by neBEM.
- void SetPrimAfter (const int n)

  Set the number of repetitions after which primitive properties are used for the physical field.
- void SetWtFldPrimAfter (const int n)

  Set the number of repetitions after which primitive properties are used for the weighting field.
- void SetOptRmPrim (const unsigned int n)

  Set option related to removal of primitives.
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override

- bool GetVoltageRange (double &vmin, double &vmax) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override

Protected Member Functions

- void Reset () override
- void UpdatePeriodicity () override

Private Types

- enum class Inversion { LU = 0 , SVD }

Private Member Functions

- void InitValues ()
- void ShiftPanels (std::vector< Panel > &panels) const

  Reduce panels to the basic period.
- bool EliminateOverlaps (const Panel &panel1, const Panel &panel2, std::vector< Panel > &panelsOut, std::vector< int > &itypo)

  Isolate the parts of polygon 1 that are not hidden by 2 and vice versa.
- bool TraceEnclosed (const std::vector< double > &xl1, const std::vector< double > &yl1, const std::vector< double > &xl2, const std::vector< double > &yl2, const Panel &originalPanel, std::vector< Panel > &newPanels) const
- void TraceNonOverlap (const std::vector< double > &xp1, const std::vector< double > &yp1, const std::vector< double > &xl1, const std::vector< double > &yl1, const std::vector< double > &xl2, const std::vector< double > &yl2, const std::vector< int > &flags1, const std::vector< int > &flags2, const std::vector< int > &links1, const std::vector< int > &links2, std::vector< bool > &mark1, int ip1, const Panel &originalPanel, std::vector< Panel > &newPanels) const
- void TraceOverlap (const std::vector< double > &xp1, const std::vector< double > &yp1, const std::vector< double > &xp2, const std::vector< double > &yp2, const std::vector< double > &xl1, const std::vector< double > &yl1, const std::vector< double > &xl2, const std::vector< double > &yl2, const std::vector< int > &flags1, const std::vector< int > &links1, const std::vector< int > &links2, std::vector< bool > &mark1, int ip1, int ip2, const Panel &originalPanel, std::vector< Panel > &newPanels) const
- bool MakePrimitives (const Panel &panelIn, std::vector< Panel > &panelsOut) const

  Split a polygon into rectangles and right-angled triangles.
- bool SplitTrapezium (const Panel panelIn, std::vector< Panel > &stack, std::vector< Panel > &panelsOut, const double epsang) const

  Check whether a polygon contains parallel lines.
- unsigned int NbOfSegments (const double length, const double target) const
- bool DiscretizeWire (const Primitive &primitive, const double targetSize, std::vector< Element > &elements) const
- bool DiscretizeTriangle (const Primitive &primitive, const double targetSize, std::vector< Element > &elements) const
- bool DiscretizeRectangle (const Primitive &prim, const double targetSize, std::vector< Element > &elements) const
- int InterfaceType (const Solid::BoundaryCondition bc) const

Private Attributes

- std::vector< Primitive > m_primitives

  List of primitives.
- std::vector< Element > m_elements

  List of elements.

- std::array< bool, 6 > m_ynplan {{false, false, false, false, false, false}}

  Plane existence.
- std::array< double, 6 > m_coplan {{0., 0., 0., 0., 0., 0.}}

  Plane coordinates.
- std::array< double, 6 > m_vtplan {{0., 0., 0., 0., 0., 0.}}

  Plane potentials.
- unsigned int m_newModel = 1
- unsigned int m_newMesh = 1
- unsigned int m_newBC = 1
- unsigned int m_newPP = 1
- unsigned int m_optStoreInflMatrix = 0
- unsigned int m_optReadInflMatrix = 0
- unsigned int m_optStoreInvMatrix = 1
- unsigned int m_optReadInvMatrix = 0
- unsigned int m_optStorePrimitives = 0
- unsigned int m_optReadPrimitives = 0
- unsigned int m_optStoreElements = 0
- unsigned int m_optReadElements = 0
- unsigned int m_optStoreFormatted = 1
- unsigned int m_optStoreUnformatted = 0
- unsigned int m_optSystemChargeZero = 1
- unsigned int m_optValidateSolution = 1
- unsigned int m_optForceValidation = 0
- unsigned int m_optRepeatLHMatrix = 0
- unsigned int m_optFastVol = 0
- unsigned int m_optCreateFastPF = 0
- unsigned int m_optReadFastPF = 0
- unsigned int m_versionFV = 0
- unsigned int m_nbBlocksFV = 0
- unsigned int m_idWtField = 0
- unsigned int m_optWtFldFastVol [11]
- unsigned int m_optCreateWtFldFastPF [11]
- unsigned int m_optReadWtFldFastPF [11]
- unsigned int m_versionWtFldFV [11]
- unsigned int m_nbBlocksWtFldFV [11]
- unsigned int m_optKnownCharge = 0
- unsigned int m_optChargingUp = 0
- unsigned int m_nThreads = 1
- int m_primAfter = -1
- int m_wtFldPrimAfter = -1
- unsigned int m_optRmPrim = 0
- double m_targetElementSize = 50.0e-4

  Target size of elements [cm].
- unsigned int m_minNbElementsOnLength = 1

  Smallest number of elements produced along the axis of a primitive.
- unsigned int m_maxNbElementsOnLength = 100

  Largest number of elements produced along the axis of a primitive.
- std::array< double, 3 > m_periodicLength {{0., 0., 0.}}

  Periodic lengths.
- unsigned int m_nCopiesX = 5

  Number of periodic copies along x.
- unsigned int m_nCopiesY = 5

  Number of periodic copies along y.

- unsigned int m_nCopiesZ = 5

  Number of periodic copies along z.
- Inversion m_inversion = Inversion::LU
- std::map< std::string, int > m_wfields

  Electrode labels and corresponding neBEM weighting field indices.

Static Private Attributes

- static constexpr double MinDist = 1.e-6

### 7.27.1 Detailed Description

Interface to neBEM.
Definition at line 14 of file ComponentNeBem3d.hh.

### 7.27.2 Member Enumeration Documentation

Inversion

enum class Garfield::ComponentNeBem3d::Inversion [strong], [private]

Enumerator

| | |
|---|---|
| LU | |
| SVD | |

Definition at line 365 of file ComponentNeBem3d.hh.
00365 { LU = 0, SVD };

### 7.27.3 Constructor & Destructor Documentation

ComponentNeBem3d()

Garfield::ComponentNeBem3d::ComponentNeBem3d ()
Constructor.

~ComponentNeBem3d()

Garfield::ComponentNeBem3d::~ComponentNeBem3d () [inline]
Destructor.
Definition at line 19 of file ComponentNeBem3d.hh.
00019 {}

### 7.27.4 Member Function Documentation

AddPlaneX()

void Garfield::ComponentNeBem3d::AddPlaneX (
const double x,
const double voltage)
Add a plane at constant x.

AddPlaneY()

void Garfield::ComponentNeBem3d::AddPlaneY (
const double y,
const double voltage)
Add a plane at constant y.

### AddPlaneZ()

void Garfield::ComponentNeBem3d::AddPlaneZ (
const double z,
const double voltage)

Add a plane at constant z.

### DiscretizeRectangle()

bool Garfield::ComponentNeBem3d::DiscretizeRectangle (
const Primitive & prim,
const double targetSize,
std::vector< Element > & elements) const [private]

### DiscretizeTriangle()

bool Garfield::ComponentNeBem3d::DiscretizeTriangle (
const Primitive & primitive,
const double targetSize,
std::vector< Element > & elements) const [private]

### DiscretizeWire()

bool Garfield::ComponentNeBem3d::DiscretizeWire (
const Primitive & primitive,
const double targetSize,
std::vector< Element > & elements) const [private]

### ElectricField() [1/2]

void Garfield::ComponentNeBem3d::ElectricField (
const double x,
const double y,
const double z,
double & ex,
double & ey,
double & ez,
double & v,
Medium *& m,
int & status) [override]

### ElectricField() [2/2]

void Garfield::ComponentNeBem3d::ElectricField (
const double x,
const double y,
const double z,
double & ex,
double & ey,
double & ez,
Medium *& m,
int & status) [override]

### EliminateOverlaps()

bool Garfield::ComponentNeBem3d::EliminateOverlaps (
const Panel & panel1,
const Panel & panel2,
std::vector< Panel > & panelsOut,
std::vector< int > & itypo) [private]

Isolate the parts of polygon 1 that are not hidden by 2 and vice versa.

### GetElement()

```
bool Garfield::ComponentNeBem3d::GetElement (
                const unsigned int i,
                std::vector< double > & xv,
                std::vector< double > & yv,
                std::vector< double > & zv,
                int & interface,
                double & bc,
                double & lambda) const
```

### GetMedium()

```
Medium * Garfield::ComponentNeBem3d::GetMedium (
                const double x,
                const double y,
                const double z)   [override]
```

### GetNumberOfElements()

```
size_t Garfield::ComponentNeBem3d::GetNumberOfElements () const   [inline], [override]
```

Definition at line 54 of file ComponentNeBem3d.hh.

```
00054 { return m_elements.size(); }
```

### GetNumberOfPlanesX()

```
unsigned int Garfield::ComponentNeBem3d::GetNumberOfPlanesX () const
```

Get the number of equipotential planes at constant x.

### GetNumberOfPlanesY()

```
unsigned int Garfield::ComponentNeBem3d::GetNumberOfPlanesY () const
```

Get the number of equipotential planes at constant y.

### GetNumberOfPlanesZ()

```
unsigned int Garfield::ComponentNeBem3d::GetNumberOfPlanesZ () const
```

Get the number of equipotential planes at constant z.

### GetNumberOfPrimitives()

```
unsigned int Garfield::ComponentNeBem3d::GetNumberOfPrimitives () const   [inline]
```

Definition at line 42 of file ComponentNeBem3d.hh.

```
00042 { return m_primitives.size(); }
```

### GetPeriodicCopies()

```
void Garfield::ComponentNeBem3d::GetPeriodicCopies (
                unsigned int & nx,
                unsigned int & ny,
                unsigned int & nz) const   [inline]
```

Retrieve the number of periodic copies used by neBEM.

Definition at line 159 of file ComponentNeBem3d.hh.

```
00160                                       {
00161     nx = m_nCopiesX;
00162     ny = m_nCopiesY;
00163     nz = m_nCopiesZ;
00164   }
```

### GetPeriodicityX()

```
bool Garfield::ComponentNeBem3d::GetPeriodicityX (
                double & s) const
```

Get the periodic length in the x-direction.

GetPeriodicityY()

bool Garfield::ComponentNeBem3d::GetPeriodicityY (
double & s) const

Get the periodic length in the y-direction.

GetPeriodicityZ()

bool Garfield::ComponentNeBem3d::GetPeriodicityZ (
double & s) const

Get the periodic length in the z-direction.

GetPlaneX()

bool Garfield::ComponentNeBem3d::GetPlaneX (
const unsigned int i,
double & x,
double & v) const

Retrieve the parameters of a plane at constant x.

GetPlaneY()

bool Garfield::ComponentNeBem3d::GetPlaneY (
const unsigned int i,
double & y,
double & v) const

Retrieve the parameters of a plane at constant y.

GetPlaneZ()

bool Garfield::ComponentNeBem3d::GetPlaneZ (
const unsigned int i,
double & z,
double & v) const

Retrieve the parameters of a plane at constant z.

GetPrimitive() [1/2]

bool Garfield::ComponentNeBem3d::GetPrimitive (
const unsigned int i,
double & a,
double & b,
double & c,
std::vector< double > & xv,
std::vector< double > & yv,
std::vector< double > & zv,
int & interface,
double & v,
double & q,
double & lambda) const

GetPrimitive() [2/2]

bool Garfield::ComponentNeBem3d::GetPrimitive (
const unsigned int i,
double & a,
double & b,
double & c,

std::vector< double > & xv,
std::vector< double > & yv,
std::vector< double > & zv,
int & vol1,
int & vol2) const

### GetVoltageRange()

bool Garfield::ComponentNeBem3d::GetVoltageRange (
double & vmin,
double & vmax) [override]

### GetVolume() [1/2]

int Garfield::ComponentNeBem3d::GetVolume (
const double x,
const double y,
const double z)

### GetVolume() [2/2]

bool Garfield::ComponentNeBem3d::GetVolume (
const unsigned int vol,
int & shape,
int & material,
double & eps,
double & potential,
double & charge,
int & bc)

### Initialise()

bool Garfield::ComponentNeBem3d::Initialise ()
Retrieve surface panels, remove contacts and cut polygons to rectangles and right-angle triangles.

### InitValues()

void Garfield::ComponentNeBem3d::InitValues () [private]

### InterfaceType()

int Garfield::ComponentNeBem3d::InterfaceType (
const Solid::BoundaryCondition bc) const [private]

### MakePrimitives()

bool Garfield::ComponentNeBem3d::MakePrimitives (
const Panel & panelIn,
std::vector< Panel > & panelsOut) const [private]
Split a polygon into rectangles and right-angled triangles.

### NbOfSegments()

unsigned int Garfield::ComponentNeBem3d::NbOfSegments (
const double length,
const double target) const [private]

### Reset()

void Garfield::ComponentNeBem3d::Reset () [override], [protected]

### SetChargingUpOptions()

```
void Garfield::ComponentNeBem3d::SetChargingUpOptions (
                const unsigned int OptChargingUp)
```

### SetComputeOptions()

```
void Garfield::ComponentNeBem3d::SetComputeOptions (
                const unsigned int OptSystemChargeZero,
                const unsigned int OptValidateSolution,
                const unsigned int OptForceValidation,
                const unsigned int OptRepeatLHMatrix)
```

### SetFastVolBlocks()

```
void Garfield::ComponentNeBem3d::SetFastVolBlocks (
                const unsigned int NbBlocksFV)
```

### SetFastVolOptions()

```
void Garfield::ComponentNeBem3d::SetFastVolOptions (
                const unsigned int OptFastVol,
                const unsigned int OptCreateFastPF,
                const unsigned int OptReadFastPF)
```

### SetFastVolVersion()

```
void Garfield::ComponentNeBem3d::SetFastVolVersion (
                const unsigned int VersionFV)
```

### SetForceValidation()

```
void Garfield::ComponentNeBem3d::SetForceValidation (
                const unsigned int OptForceValidation)
```

### SetFormattedFile()

```
void Garfield::ComponentNeBem3d::SetFormattedFile (
                const unsigned int OptFormattedFile)
```

### SetKnownChargeOptions()

```
void Garfield::ComponentNeBem3d::SetKnownChargeOptions (
                const unsigned int OptKnownCharge)
```

### SetMinMaxNumberOfElements()

```
void Garfield::ComponentNeBem3d::SetMinMaxNumberOfElements (
                const unsigned int nmin,
                const unsigned int nmax)
```

Set the smallest and largest allowed number of elements along the lenght of a primitive.

### SetMirrorPeriodicityX()

```
void Garfield::ComponentNeBem3d::SetMirrorPeriodicityX (
                const double s)
```

Set the periodic length [cm] in the x-direction.

SetMirrorPeriodicityY()

void Garfield::ComponentNeBem3d::SetMirrorPeriodicityY (
const double s)
Set the periodic length [cm] in the y-direction.

SetMirrorPeriodicityZ()

void Garfield::ComponentNeBem3d::SetMirrorPeriodicityZ (
const double s)
Set the periodic length [cm] in the z-direction.

SetModelOptions()

void Garfield::ComponentNeBem3d::SetModelOptions (
const unsigned int NewModel,
const unsigned int NewMesh,
const unsigned int NewBC,
const unsigned int NewPP)

SetNewBC()

void Garfield::ComponentNeBem3d::SetNewBC (
const unsigned int NewBC)

SetNewMesh()

void Garfield::ComponentNeBem3d::SetNewMesh (
const unsigned int NewMesh)

SetNewModel()

void Garfield::ComponentNeBem3d::SetNewModel (
const unsigned int NewModel)

SetNewPP()

void Garfield::ComponentNeBem3d::SetNewPP (
const unsigned int NewPP)

SetNumberOfThreads()

void Garfield::ComponentNeBem3d::SetNumberOfThreads (
const unsigned int n) [inline]
Set the number of threads to be used by neBEM.
Definition at line 185 of file ComponentNeBem3d.hh.
00185 { m_nThreads = n > 0 ? n : 1; }

SetOptRmPrim()

void Garfield::ComponentNeBem3d::SetOptRmPrim (
const unsigned int n) [inline]
Set option related to removal of primitives.
Definition at line 198 of file ComponentNeBem3d.hh.
00198 { m_optRmPrim = n; }

SetPeriodicCopies()

void Garfield::ComponentNeBem3d::SetPeriodicCopies (
const unsigned int nx,
const unsigned int ny,
const unsigned int nz)

Set the parameters $n_x, n_y, n_z$ defining the number of periodic copies that neBEM will use when dealing with periodic configurations.
neBEM will use $2 \times n + 1$ copies (default: $n = 5$).

SetPeriodicityX()

void Garfield::ComponentNeBem3d::SetPeriodicityX (
const double s)
Set the periodic length [cm] in the x-direction.

SetPeriodicityY()

void Garfield::ComponentNeBem3d::SetPeriodicityY (
const double s)
Set the periodic length [cm] in the y-direction.

SetPeriodicityZ()

void Garfield::ComponentNeBem3d::SetPeriodicityZ (
const double s)
Set the periodic length [cm] in the z-direction.

SetPrimAfter()

void Garfield::ComponentNeBem3d::SetPrimAfter (
const int n) [inline]
Set the number of repetitions after which primitive properties are used for the physical field.
A negative value (default) implies all the elements are always evaluated.
Definition at line 190 of file ComponentNeBem3d.hh.
00190 { m_primAfter = n; }

SetReadElements()

void Garfield::ComponentNeBem3d::SetReadElements (
const unsigned int OptReadElements)

SetReadInflMatrix()

void Garfield::ComponentNeBem3d::SetReadInflMatrix (
const unsigned int OptReadInflMatrix)

SetReadInvMatrix()

void Garfield::ComponentNeBem3d::SetReadInvMatrix (
const unsigned int OptReadInvMatrix)

SetReadPrimitives()

void Garfield::ComponentNeBem3d::SetReadPrimitives (
const unsigned int OptReadPrimitives)

SetRepeatLHMatrix()

void Garfield::ComponentNeBem3d::SetRepeatLHMatrix (
const unsigned int OptRepeatLHMatrix)

SetReuseModel()

void Garfield::ComponentNeBem3d::SetReuseModel (
void )

SetStoreElements()

```
void Garfield::ComponentNeBem3d::SetStoreElements (
        const unsigned int OptStoreElements)
```

SetStoreInflMatrix()

```
void Garfield::ComponentNeBem3d::SetStoreInflMatrix (
        const unsigned int OptStoreInflMatrix)
```

Set storing options (OptStoreInflMatrix, OptStoreInvMatrix, OptStoreInvMatrix, OptStoreInvMatrix) OptStorePrimitives, OptStorePrimitives) OptStoreElements, OptStoreElements) OptFormattedFile, Opt↩ UnformattedFile)

SetStoreInvMatrix()

```
void Garfield::ComponentNeBem3d::SetStoreInvMatrix (
        const unsigned int OptStoreInvMatrix)
```

SetStorePrimitives()

```
void Garfield::ComponentNeBem3d::SetStorePrimitives (
        const unsigned int OptStorePrimitives)
```

SetStoreReadOptions()

```
void Garfield::ComponentNeBem3d::SetStoreReadOptions (
        const unsigned int OptStoreInflMatrix,
        const unsigned int OptReadInflMatrix,
        const unsigned int OptStoreInvMatrix,
        const unsigned int OptReadInvMatrix,
        const unsigned int OptStorePrimitives,
        const unsigned int OptReadPrimitives,
        const unsigned int OptStoreElements,
        const unsigned int OptReadElements,
        const unsigned int OptFormattedFile,
        const unsigned int OptUnformattedFile)
```

SetSystemChargeZero()

```
void Garfield::ComponentNeBem3d::SetSystemChargeZero (
        const unsigned int OptSystemChargeZero)
```

Other functions to be, are void SetPlotOptions(OptGnuplot=0, OptGnuplotPrimitives=0, OptGnuplot↩ Elements=0, OptPrimitiveFiles=0, OptElementFiles=0)

SetTargetElementSize()

```
void Garfield::ComponentNeBem3d::SetTargetElementSize (
        const double length)
```

Set the default value of the target linear size of the elements produced by neBEM's discretisation process.

SetUnformattedFile()

```
void Garfield::ComponentNeBem3d::SetUnformattedFile (
        const unsigned int OptUnformattedFile)
```

SetValidateSolution()

```
void Garfield::ComponentNeBem3d::SetValidateSolution (
        const unsigned int OptValidateSolution)
```

### SetWtFldFastVolBlocks()

```
void Garfield::ComponentNeBem3d::SetWtFldFastVolBlocks (
                const unsigned int IdWtField,
                const unsigned int NbBlocksWtFldFV)
```

### SetWtFldFastVolOptions()

```
void Garfield::ComponentNeBem3d::SetWtFldFastVolOptions (
                const unsigned int IdWtField,
                const unsigned int OptWtFldFastVol,
                const unsigned int OptCreateWtFldFastPF,
                const unsigned int OptReadWtFldFastPF)
```

### SetWtFldFastVolVersion()

```
void Garfield::ComponentNeBem3d::SetWtFldFastVolVersion (
                const unsigned int IdWtField,
                const unsigned int VersionWtFldFV)
```

### SetWtFldPrimAfter()

```
void Garfield::ComponentNeBem3d::SetWtFldPrimAfter (
                const int n)   [inline]
```

Set the number of repetitions after which primitive properties are used for the weighting field.
A negative value (default) implies all the elements are always evaluated.
Definition at line 195 of file ComponentNeBem3d.hh.

```
00195 { m_wtFldPrimAfter = n; }
```

### ShiftPanels()

```
void Garfield::ComponentNeBem3d::ShiftPanels (
                std::vector< Panel > & panels) const   [private]
```

Reduce panels to the basic period.

### SplitTrapezium()

```
bool Garfield::ComponentNeBem3d::SplitTrapezium (
                const Panel panelIn,
                std::vector< Panel > & stack,
                std::vector< Panel > & panelsOut,
                const double epsang) const   [private]
```

Check whether a polygon contains parallel lines.
If it does, split it in rectangular and non-rectangular parts.

### TraceEnclosed()

```
bool Garfield::ComponentNeBem3d::TraceEnclosed (
                const std::vector< double > & xl1,
                const std::vector< double > & yl1,
                const std::vector< double > & xl2,
                const std::vector< double > & yl2,
                const Panel & originalPanel,
                std::vector< Panel > & newPanels) const   [private]
```

### TraceNonOverlap()

```
void Garfield::ComponentNeBem3d::TraceNonOverlap (
                const std::vector< double > & xp1,
                const std::vector< double > & yp1,
                const std::vector< double > & xl1,
```

const std::vector< double > & yl1,
const std::vector< double > & xl2,
const std::vector< double > & yl2,
const std::vector< int > & flags1,
const std::vector< int > & flags2,
const std::vector< int > & links1,
const std::vector< int > & links2,
std::vector< bool > & mark1,
int ip1,
const Panel & originalPanel,
std::vector< Panel > & newPanels) const [private]

### TraceOverlap()

void Garfield::ComponentNeBem3d::TraceOverlap (
const std::vector< double > & xp1,
const std::vector< double > & yp1,
const std::vector< double > & xp2,
const std::vector< double > & yp2,
const std::vector< double > & xl1,
const std::vector< double > & yl1,
const std::vector< double > & xl2,
const std::vector< double > & yl2,
const std::vector< int > & flags1,
const std::vector< int > & links1,
const std::vector< int > & links2,
std::vector< bool > & mark1,
int ip1,
int ip2,
const Panel & originalPanel,
std::vector< Panel > & newPanels) const [private]

### UpdatePeriodicity()

void Garfield::ComponentNeBem3d::UpdatePeriodicity () [override], [protected]

### UseLUInversion()

void Garfield::ComponentNeBem3d::UseLUInversion () [inline]
Invert the influence matrix using lower-upper (LU) decomposition.
Definition at line 149 of file ComponentNeBem3d.hh.
00149 { m_inversion = Inversion::LU; }

### UseSVDInversion()

void Garfield::ComponentNeBem3d::UseSVDInversion () [inline]
Invert the influence matrix using singular value decomposition.
Definition at line 151 of file ComponentNeBem3d.hh.
00151 { m_inversion = Inversion::SVD; }

### WeightingField()

void Garfield::ComponentNeBem3d::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

WeightingPotential()

double Garfield::ComponentNeBem3d::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

### 7.27.5 Member Data Documentation

m_coplan

std::array<double, 6> Garfield::ComponentNeBem3d::m_coplan {{0., 0., 0., 0., 0., 0.}} [private]
Plane coordinates.
Definition at line 278 of file ComponentNeBem3d.hh.
00278 {{0., 0., 0., 0., 0., 0.}};

m_elements

std::vector<Element> Garfield::ComponentNeBem3d::m_elements [private]
List of elements.
Definition at line 273 of file ComponentNeBem3d.hh.

m_idWtField

unsigned int Garfield::ComponentNeBem3d::m_idWtField = 0 [private]
Definition at line 320 of file ComponentNeBem3d.hh.

m_inversion

Inversion Garfield::ComponentNeBem3d::m_inversion = Inversion::LU [private]
Definition at line 366 of file ComponentNeBem3d.hh.

m_maxNbElementsOnLength

unsigned int Garfield::ComponentNeBem3d::m_maxNbElementsOnLength = 100 [private]
Largest number of elements produced along the axis of a primitive.
Definition at line 355 of file ComponentNeBem3d.hh.

m_minNbElementsOnLength

unsigned int Garfield::ComponentNeBem3d::m_minNbElementsOnLength = 1 [private]
Smallest number of elements produced along the axis of a primitive.
Definition at line 353 of file ComponentNeBem3d.hh.

m_nbBlocksFV

unsigned int Garfield::ComponentNeBem3d::m_nbBlocksFV = 0 [private]
Definition at line 317 of file ComponentNeBem3d.hh.

m_nbBlocksWtFldFV

unsigned int Garfield::ComponentNeBem3d::m_nbBlocksWtFldFV[11] [private]
Definition at line 325 of file ComponentNeBem3d.hh.

m_nCopiesX

unsigned int Garfield::ComponentNeBem3d::m_nCopiesX = 5 [private]
Number of periodic copies along x.
Definition at line 359 of file ComponentNeBem3d.hh.

m_nCopiesY

unsigned int Garfield::ComponentNeBem3d::m_nCopiesY = 5 [private]
Number of periodic copies along y.
Definition at line 361 of file ComponentNeBem3d.hh.

m_nCopiesZ

unsigned int Garfield::ComponentNeBem3d::m_nCopiesZ = 5 [private]
Number of periodic copies along z.
Definition at line 363 of file ComponentNeBem3d.hh.

m_newBC

unsigned int Garfield::ComponentNeBem3d::m_newBC = 1 [private]
Definition at line 285 of file ComponentNeBem3d.hh.

m_newMesh

unsigned int Garfield::ComponentNeBem3d::m_newMesh = 1 [private]
Definition at line 284 of file ComponentNeBem3d.hh.

m_newModel

unsigned int Garfield::ComponentNeBem3d::m_newModel = 1 [private]
Definition at line 283 of file ComponentNeBem3d.hh.

m_newPP

unsigned int Garfield::ComponentNeBem3d::m_newPP = 1 [private]
Definition at line 286 of file ComponentNeBem3d.hh.

m_nThreads

unsigned int Garfield::ComponentNeBem3d::m_nThreads = 1 [private]
Definition at line 334 of file ComponentNeBem3d.hh.

m_optChargingUp

unsigned int Garfield::ComponentNeBem3d::m_optChargingUp = 0 [private]
Definition at line 331 of file ComponentNeBem3d.hh.

m_optCreateFastPF

unsigned int Garfield::ComponentNeBem3d::m_optCreateFastPF = 0 [private]
Definition at line 314 of file ComponentNeBem3d.hh.

m_optCreateWtFldFastPF

unsigned int Garfield::ComponentNeBem3d::m_optCreateWtFldFastPF[11] [private]
Definition at line 322 of file ComponentNeBem3d.hh.

m_optFastVol

unsigned int Garfield::ComponentNeBem3d::m_optFastVol = 0 [private]
Definition at line 313 of file ComponentNeBem3d.hh.

m_optForceValidation

unsigned int Garfield::ComponentNeBem3d::m_optForceValidation = 0 [private]
Definition at line 309 of file ComponentNeBem3d.hh.

### m_optKnownCharge

unsigned int Garfield::ComponentNeBem3d::m_optKnownCharge = 0 [private]
Definition at line 328 of file ComponentNeBem3d.hh.

### m_optReadElements

unsigned int Garfield::ComponentNeBem3d::m_optReadElements = 0 [private]
Definition at line 296 of file ComponentNeBem3d.hh.

### m_optReadFastPF

unsigned int Garfield::ComponentNeBem3d::m_optReadFastPF = 0 [private]
Definition at line 315 of file ComponentNeBem3d.hh.

### m_optReadInflMatrix

unsigned int Garfield::ComponentNeBem3d::m_optReadInflMatrix = 0 [private]
Definition at line 290 of file ComponentNeBem3d.hh.

### m_optReadInvMatrix

unsigned int Garfield::ComponentNeBem3d::m_optReadInvMatrix = 0 [private]
Definition at line 292 of file ComponentNeBem3d.hh.

### m_optReadPrimitives

unsigned int Garfield::ComponentNeBem3d::m_optReadPrimitives = 0 [private]
Definition at line 294 of file ComponentNeBem3d.hh.

### m_optReadWtFldFastPF

unsigned int Garfield::ComponentNeBem3d::m_optReadWtFldFastPF[11] [private]
Definition at line 323 of file ComponentNeBem3d.hh.

### m_optRepeatLHMatrix

unsigned int Garfield::ComponentNeBem3d::m_optRepeatLHMatrix = 0 [private]
Definition at line 310 of file ComponentNeBem3d.hh.

### m_optRmPrim

unsigned int Garfield::ComponentNeBem3d::m_optRmPrim = 0 [private]
Definition at line 347 of file ComponentNeBem3d.hh.

### m_optStoreElements

unsigned int Garfield::ComponentNeBem3d::m_optStoreElements = 0 [private]
Definition at line 295 of file ComponentNeBem3d.hh.

### m_optStoreFormatted

unsigned int Garfield::ComponentNeBem3d::m_optStoreFormatted = 1 [private]
Definition at line 297 of file ComponentNeBem3d.hh.

### m_optStoreInflMatrix

unsigned int Garfield::ComponentNeBem3d::m_optStoreInflMatrix = 0 [private]
Definition at line 289 of file ComponentNeBem3d.hh.

### m_optStoreInvMatrix

unsigned int Garfield::ComponentNeBem3d::m_optStoreInvMatrix = 1 [private]
Definition at line 291 of file ComponentNeBem3d.hh.

### m_optStorePrimitives

unsigned int Garfield::ComponentNeBem3d::m_optStorePrimitives = 0 [private]
Definition at line 293 of file ComponentNeBem3d.hh.

### m_optStoreUnformatted

unsigned int Garfield::ComponentNeBem3d::m_optStoreUnformatted = 0 [private]
Definition at line 298 of file ComponentNeBem3d.hh.

### m_optSystemChargeZero

unsigned int Garfield::ComponentNeBem3d::m_optSystemChargeZero = 1 [private]
Definition at line 307 of file ComponentNeBem3d.hh.

### m_optValidateSolution

unsigned int Garfield::ComponentNeBem3d::m_optValidateSolution = 1 [private]
Definition at line 308 of file ComponentNeBem3d.hh.

### m_optWtFldFastVol

unsigned int Garfield::ComponentNeBem3d::m_optWtFldFastVol[11] [private]
Definition at line 321 of file ComponentNeBem3d.hh.

### m_periodicLength

std::array<double, 3> Garfield::ComponentNeBem3d::m_periodicLength {{0., 0., 0.}} [private]
Periodic lengths.
Definition at line 357 of file ComponentNeBem3d.hh.
00357 {{0., 0., 0.}};

### m_primAfter

int Garfield::ComponentNeBem3d::m_primAfter = -1 [private]
Definition at line 338 of file ComponentNeBem3d.hh.

### m_primitives

std::vector<Primitive> Garfield::ComponentNeBem3d::m_primitives [private]
List of primitives.
Definition at line 242 of file ComponentNeBem3d.hh.

### m_targetElementSize

double Garfield::ComponentNeBem3d::m_targetElementSize = 50.0e-4 [private]
Target size of elements [cm].
Definition at line 351 of file ComponentNeBem3d.hh.

### m_versionFV

unsigned int Garfield::ComponentNeBem3d::m_versionFV = 0 [private]
Definition at line 316 of file ComponentNeBem3d.hh.

### m_versionWtFldFV

unsigned int Garfield::ComponentNeBem3d::m_versionWtFldFV[11] [private]
Definition at line 324 of file ComponentNeBem3d.hh.

m_vtplan

std::array<double, 6> Garfield::ComponentNeBem3d::m_vtplan {{0., 0., 0., 0., 0., 0.}} [private]
Plane potentials.
Definition at line 280 of file ComponentNeBem3d.hh.
00280 {{0., 0., 0., 0., 0., 0.}};

m_wfields

std::map<std::string, int> Garfield::ComponentNeBem3d::m_wfields [private]
Electrode labels and corresponding neBEM weighting field indices.
Definition at line 369 of file ComponentNeBem3d.hh.

m_wtFldPrimAfter

int Garfield::ComponentNeBem3d::m_wtFldPrimAfter = -1 [private]
Definition at line 343 of file ComponentNeBem3d.hh.

m_ynplan

std::array<bool, 6> Garfield::ComponentNeBem3d::m_ynplan {{false, false, false, false, false, false}} [private]
Plane existence.
Definition at line 276 of file ComponentNeBem3d.hh.
00276 {{false, false, false, false, false, false}};

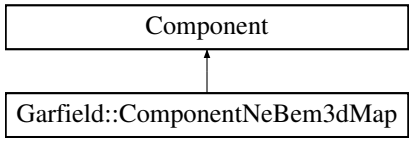
MinDist

double Garfield::ComponentNeBem3d::MinDist = 1.e-6 [static], [constexpr], [private]
Definition at line 349 of file ComponentNeBem3d.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3d.hh

## 7.28 Garfield::ComponentNeBem3dMap Class Reference

Component for interpolating field maps stored in a mesh generated by neBEM.
#include <ComponentNeBem3dMap.hh>
Inheritance diagram for Garfield::ComponentNeBem3dMap:




Classes

- struct Element

Public Member Functions

- ComponentNeBem3dMap ()
  Constructor.
- ~ComponentNeBem3dMap ()
  Destructor.
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override

- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- void MagneticField (const double x, const double y, const double z, double &bx, double &by, double &bz, int &status) override
- void SetWeightingFieldOffset (const double x, const double y, const double z)

  Offset coordinates in the weighting field, such that the same numerical weighting field map can be used for electrodes at different positions.
- Medium * GetMedium (const double x, const double y, const double z) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool GetElectricFieldRange (double &exmin, double &exmax, double &eymin, double &eymax, double &ezmin, double &ezmax)
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool LoadMapInfo (const std::string &MapInfoFile, std::string &MapVersion, int &OptMap, int &OptStaggerMap, unsigned int &NbOfXCells, unsigned int &NbOfYCells, unsigned int &NbOfZCells, double &Xmin, double &Xmax, double &Ymin, double &Ymax, double &Zmin, double &Zmax, double &XStagger, double &YStagger, double &ZStagger, std::string &MapDataFile)

  Map related information.
- void SetMesh (const unsigned int nx, const unsigned int ny, const unsigned int nz, const double xmin, const double xmax, const double ymin, const double ymax, const double zmin, const double zmax)

  Define the grid.
- bool LoadElectricField (const std::string &filename, const std::string &format, const bool withPotential, const bool withRegion, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import electric field and potential values from a file.
- bool LoadMagneticField (const std::string &filename, const std::string &format, const double scaleX=1., const double scaleB=1.)

  Import magnetic field values from a file.
- bool GetElement (const double xi, const double yi, const double zi, unsigned int &i, unsigned int &j, unsigned int &k, bool &xMirrored, bool &yMirrored, bool &zMirrored) const

  Return the indices of the element at a given point.
- bool GetElement (const unsigned int i, const unsigned int j, const unsigned int k, double &v, double &ex, double &ey, double &ez) const

  Return the field for an element with given index.
- void SetMedium (const unsigned int i, Medium *m)

  Set the medium in region i.
- Medium * GetMedium (const unsigned int i) const

  Get the medium in region i.
- void PrintRegions () const

  Print all regions.

Private Member Functions

- bool LoadData (const std::string &filename, std::string format, const bool withPotential, const bool withRegion, const double scaleX, const double scaleF, const double scaleP, const char field)

  Read data from file.
- void Reset () override
- void UpdatePeriodicity () override
- double Reduce (const double xin, const double xmin, const double xmax, const bool simplePeriodic, const bool mirrorPeriodic, bool &isMirrored) const

  Reduce a coordinate to the basic cell (in case of periodicity).

- double TriLinInt (const double xd, const double yd, const double zd, const double c000, const double c100, const double c010, const double c001, const double c110, const double c101, const double c011, const double c111)

  Interpolate tri-linear.

Private Attributes

- std::vector< Medium * > m_media
- std::vector< std::vector< std::vector< Element > > > m_efields

  Electric field values and potentials at each mesh element.
- std::vector< std::vector< std::vector< Element > > > m_bfields

  Magnetic field values at each mesh element.
- std::vector< std::vector< std::vector< int > > > m_regions

  Region indices.
- unsigned int m_nX = 0
- unsigned int m_nY = 0
- unsigned int m_nZ = 0
- double m_xMin = 0.
- double m_yMin = 0.
- double m_zMin = 0.
- double m_xMax = 0.
- double m_yMax = 0.
- double m_zMax = 0.
- bool m_hasMesh = false
- bool m_hasPotential = false
- bool m_hasEfield = false
- bool m_hasBfield = false
- double m_wField_xOffset = 0.
- double m_wField_yOffset = 0.
- double m_wField_zOffset = 0.
- double m_pMin = 0.
- double m_pMax = 0.

### 7.28.1 Detailed Description

Component for interpolating field maps stored in a mesh generated by neBEM.
Definition at line 10 of file ComponentNeBem3dMap.hh.

### 7.28.2 Constructor & Destructor Documentation

ComponentNeBem3dMap()

Garfield::ComponentNeBem3dMap::ComponentNeBem3dMap ()
Constructor.

~ComponentNeBem3dMap()

Garfield::ComponentNeBem3dMap::~ComponentNeBem3dMap () [inline]
Destructor.
Definition at line 15 of file ComponentNeBem3dMap.hh.
00015 {}

### 7.28.3 Member Function Documentation

#### ElectricField() [1/2]

```
void Garfield::ComponentNeBem3dMap::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)   [override]
```

#### ElectricField() [2/2]

```
void Garfield::ComponentNeBem3dMap::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)   [override]
```

#### GetBoundingBox()

```
bool Garfield::ComponentNeBem3dMap::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)   [override]
```

#### GetElectricFieldRange()

```
bool Garfield::ComponentNeBem3dMap::GetElectricFieldRange (
                double & exmin,
                double & exmax,
                double & eymin,
                double & eymax,
                double & ezmin,
                double & ezmax)
```

#### GetElement() [1/2]

```
bool Garfield::ComponentNeBem3dMap::GetElement (
                const double xi,
                const double yi,
                const double zi,
                unsigned int & i,
                unsigned int & j,
                unsigned int & k,
                bool & xMirrored,
                bool & yMirrored,
                bool & zMirrored) const
```

Return the indices of the element at a given point.

GetElement() [2/2]

```
bool Garfield::ComponentNeBem3dMap::GetElement (
                const unsigned int i,
                const unsigned int j,
                const unsigned int k,
                double & v,
                double & ex,
                double & ey,
                double & ez) const
```

Return the field for an element with given index.

GetMedium() [1/2]

```
Medium * Garfield::ComponentNeBem3dMap::GetMedium (
                const double x,
                const double y,
                const double z)   [override]
```

GetMedium() [2/2]

```
Medium * Garfield::ComponentNeBem3dMap::GetMedium (
                const unsigned int i) const
```

Get the medium in region i.

GetVoltageRange()

```
bool Garfield::ComponentNeBem3dMap::GetVoltageRange (
                double & vmin,
                double & vmax)   [override]
```

LoadData()

```
bool Garfield::ComponentNeBem3dMap::LoadData (
                const std::string & filename,
                std::string format,
                const bool withPotential,
                const bool withRegion,
                const double scaleX,
                const double scaleF,
                const double scaleP,
                const char field)   [private]
```

Read data from file.

LoadElectricField()

```
bool Garfield::ComponentNeBem3dMap::LoadElectricField (
                const std::string & filename,
                const std::string & format,
                const bool withPotential,
                const bool withRegion,
                const double scaleX = 1.,
                const double scaleE = 1.,
                const double scaleP = 1.)
```

Import electric field and potential values from a file.
The file is supposed to contain one line for each mesh point starting with

- either two or three floating point numbers, specifying the coordinates (in cm) of the element centre or
- two or three integers specifying the index of the element in the mesh,

followed by

- two or three floating point numbers for the electric field (in V/cm), and (depending on the values of withPotential and withRegion),
- a floating point number specifying the potential (in V), and
- an integer specifying the "region" of the element.

Format types are:

- "xy", "xyz": elements are specified by the coordinates of their centres
- "ij", "ijk": elements are specified by their indices

### LoadMagneticField()

```
bool Garfield::ComponentNeBem3dMap::LoadMagneticField (
                const std::string & filename,
                const std::string & format,
                const double scaleX = 1.,
                const double scaleB = 1.)
```

Import magnetic field values from a file.

### LoadMapInfo()

```
bool Garfield::ComponentNeBem3dMap::LoadMapInfo (
                const std::string & MapInfoFile,
                std::string & MapVersion,
                int & OptMap,
                int & OptStaggerMap,
                unsigned int & NbOfXCells,
                unsigned int & NbOfYCells,
                unsigned int & NbOfZCells,
                double & Xmin,
                double & Xmax,
                double & Ymin,
                double & Ymax,
                double & Zmin,
                double & Zmax,
                double & XStagger,
                double & YStagger,
                double & ZStagger,
                std::string & MapDataFile)
```

Map related information.

### MagneticField()

```
void Garfield::ComponentNeBem3dMap::MagneticField (
                const double x,
                const double y,
                const double z,
                double & bx,
                double & by,
                double & bz,
                int & status)   [override]
```

### PrintRegions()

```
void Garfield::ComponentNeBem3dMap::PrintRegions () const
```

Print all regions.

### Reduce()

```
double Garfield::ComponentNeBem3dMap::Reduce (
                const double xin,
                const double xmin,
                const double xmax,
                const bool simplePeriodic,
                const bool mirrorPeriodic,
                bool & isMirrored) const   [private]
```

Reduce a coordinate to the basic cell (in case of periodicity).

### Reset()

```
void Garfield::ComponentNeBem3dMap::Reset ()   [override], [private]
```

### SetMedium()

```
void Garfield::ComponentNeBem3dMap::SetMedium (
                const unsigned int i,
                Medium * m)
```

Set the medium in region i.

### SetMesh()

```
void Garfield::ComponentNeBem3dMap::SetMesh (
                const unsigned int nx,
                const unsigned int ny,
                const unsigned int nz,
                const double xmin,
                const double xmax,
                const double ymin,
                const double ymax,
                const double zmin,
                const double zmax)
```

Define the grid.

Parameters

| | |
|---|---|
| nx,ny,nz | number of bins along x, y, z. |
| xmin,xmax | range along $x$. |
| ymin,ymax | range along $y$. |
| zmin,zmax | range along $z$. |

### SetWeightingFieldOffset()

```
void Garfield::ComponentNeBem3dMap::SetWeightingFieldOffset (
                const double x,
                const double y,
                const double z)
```

Offset coordinates in the weighting field, such that the same numerical weighting field map can be used for electrodes at different positions.

### TriLinInt()

```
double Garfield::ComponentNeBem3dMap::TriLinInt (
                const double xd,
                const double yd,
                const double zd,
```

const double c000,
const double c100,
const double c010,
const double c001,
const double c110,
const double c101,
const double c011,
const double c111) [private]

Interpolate tri-linear.

### UpdatePeriodicity()

void Garfield::ComponentNeBem3dMap::UpdatePeriodicity () [override], [private]

### WeightingField()

void Garfield::ComponentNeBem3dMap::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

### WeightingPotential()

double Garfield::ComponentNeBem3dMap::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

## 7.28.4 Member Data Documentation

### m_bfields

std::vector<std::vector<std::vector<Element> > > Garfield::ComponentNeBem3dMap::m_bfields [private]

Magnetic field values at each mesh element.

Definition at line 112 of file ComponentNeBem3dMap.hh.

### m_efields

std::vector<std::vector<std::vector<Element> > > Garfield::ComponentNeBem3dMap::m_efields [private]

Electric field values and potentials at each mesh element.

Definition at line 110 of file ComponentNeBem3dMap.hh.

### m_hasBfield

bool Garfield::ComponentNeBem3dMap::m_hasBfield = false [private]

Definition at line 123 of file ComponentNeBem3dMap.hh.

### m_hasEfield

bool Garfield::ComponentNeBem3dMap::m_hasEfield = false [private]

Definition at line 122 of file ComponentNeBem3dMap.hh.

### m_hasMesh

bool Garfield::ComponentNeBem3dMap::m_hasMesh = false [private]

Definition at line 120 of file ComponentNeBem3dMap.hh.

### m_hasPotential

bool Garfield::ComponentNeBem3dMap::m_hasPotential = false [private]
Definition at line 121 of file ComponentNeBem3dMap.hh.

### m_media

std::vector<Medium*> Garfield::ComponentNeBem3dMap::m_media [private]
Definition at line 104 of file ComponentNeBem3dMap.hh.

### m_nX

unsigned int Garfield::ComponentNeBem3dMap::m_nX = 0 [private]
Definition at line 116 of file ComponentNeBem3dMap.hh.

### m_nY

unsigned int Garfield::ComponentNeBem3dMap::m_nY = 0 [private]
Definition at line 116 of file ComponentNeBem3dMap.hh.

### m_nZ

unsigned int Garfield::ComponentNeBem3dMap::m_nZ = 0 [private]
Definition at line 116 of file ComponentNeBem3dMap.hh.

### m_pMax

double Garfield::ComponentNeBem3dMap::m_pMax = 0. [private]
Definition at line 131 of file ComponentNeBem3dMap.hh.

### m_pMin

double Garfield::ComponentNeBem3dMap::m_pMin = 0. [private]
Definition at line 131 of file ComponentNeBem3dMap.hh.

### m_regions

std::vector<std::vector<std::vector<int> > > Garfield::ComponentNeBem3dMap::m_regions [private]
Region indices.
Definition at line 114 of file ComponentNeBem3dMap.hh.

### m_wField_xOffset

double Garfield::ComponentNeBem3dMap::m_wField_xOffset = 0. [private]
Definition at line 126 of file ComponentNeBem3dMap.hh.

### m_wField_yOffset

double Garfield::ComponentNeBem3dMap::m_wField_yOffset = 0. [private]
Definition at line 127 of file ComponentNeBem3dMap.hh.

### m_wField_zOffset

double Garfield::ComponentNeBem3dMap::m_wField_zOffset = 0. [private]
Definition at line 128 of file ComponentNeBem3dMap.hh.

### m_xMax

double Garfield::ComponentNeBem3dMap::m_xMax = 0. [private]
Definition at line 118 of file ComponentNeBem3dMap.hh.

m_xMin

double Garfield::ComponentNeBem3dMap::m_xMin = 0. [private]
Definition at line 117 of file ComponentNeBem3dMap.hh.

m_yMax

double Garfield::ComponentNeBem3dMap::m_yMax = 0. [private]
Definition at line 118 of file ComponentNeBem3dMap.hh.

m_yMin

double Garfield::ComponentNeBem3dMap::m_yMin = 0. [private]
Definition at line 117 of file ComponentNeBem3dMap.hh.

m_zMax

double Garfield::ComponentNeBem3dMap::m_zMax = 0. [private]
Definition at line 118 of file ComponentNeBem3dMap.hh.

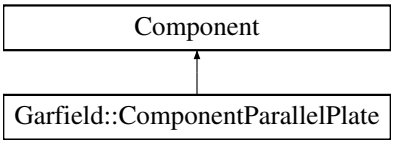
m_zMin

double Garfield::ComponentNeBem3dMap::m_zMin = 0. [private]
Definition at line 117 of file ComponentNeBem3dMap.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3dMap.hh

## 7.29 Garfield::ComponentParallelPlate Class Reference

Component for parallel-plate geometries.
#include <ComponentParallelPlate.hh>
Inheritance diagram for Garfield::ComponentParallelPlate:




Classes

- struct Electrode

  Structure that captures the information of the electrodes under study.

Public Member Functions

- ComponentParallelPlate ()

  Constructor.
- ~ComponentParallelPlate ()

  Destructor.
- void Setup (const unsigned int N, std::vector< double > eps, std::vector< double > d, const double V, std::vector< int > sigmaIndex={})

  Define the geometry.
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override

- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- void AddPixel (double x, double z, double lx, double lz, const std::string &label, bool fromAnode=true)

  Add a pixel electrode.
- void AddStrip (double z, double lz, const std::string &label, bool fromAnode=true)

  Add strip electrode.
- void AddPlane (const std::string &label, bool fromAnode=true)

  Add plane electrode, if you want to read the signal from the cathode set the second argument to false.
- void SetMedium (Medium *medium)

  Setting the medium.
- void SetWeightingPotentialGrid (const double xmin, const double xmax, const double xsteps, const double ymin, const double ymax, const double ysteps, const double zmin, const double zmax, const double zsteps, const std::string &label)

  Calculate time-dependent weighting potential on a grid.
- void SetWeightingPotentialGrids (const double xmin, const double xmax, const double xsteps, const double ymin, const double ymax, const double ysteps, const double zmin, const double zmax, const double zsteps)

  This will calculate all electrodes time-dependent weighting potential on the specified grid.
- void LoadWeightingPotentialGrid (const std::string &label)

  This will load a previously calculated grid of time-dependent weighting potential values.
- Medium * GetMedium (const double x, const double y, const double z) override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool getLayer (const double y, int &m, double &epsM)
- void getPermittivityFromLayer (int m, double &eps)
- void getZBoundFromLayer (int m, double &zbottom, double &ztop)
- int NumberOfLayers ()
- void IndexOfGasGaps (std::vector< int > &indexGasGap)
- void SetIntegrationPrecision (const double eps)
- void SetIntegrationUpperbound (const double p)
- void DisablePotentialCalculationOutsideGasGap ()

Private Types

- enum structureelectrode { NotSet = -1 , Plane , Strip , Pixel }

  Possible readout groups.

Private Member Functions

- double IntegratePromptPotential (const Electrode &el, const double x, const double y, const double z)
- void CalculateDynamicalWeightingPotential (const Electrode &el)
- double FindWeightingPotentialInGrid (Electrode &el, const double x, const double y, const double z)
- bool Nsigma (int N, std::vector< std::vector< int > > &sigmaMatrix)
- bool Ntheta (int N, std::vector< std::vector< int > > &thetaMatrix, std::vector< std::vector< int > > &sigmaMatrix)
- void constructGeometryMatrices (const int N)
- void constructGeometryFunction (const int N, const std::vector< double > &d)
- void setHIntegrand ()
- void setwpPixelIntegrand ()
- void setwpStripIntegrand ()
- double constWEFieldLayer (const int indexLayer)

- double wpPlane (const double z)
- double constEFieldLayer (const int indexLayer)
- bool decToBinary (int n, std::vector< int > &binaryNum)
- void LayerUpdate (const double z, const int im, const double epsM)
- void UpdatePeriodicity () override
- void Reset () override

Private Attributes

- double m_precision = 1.e-12
- double m_V = 0.

  Voltage difference between the parallel plates.
- bool m_getPotentialInPlate = true
- int m_N = 0

  Number of layers.
- double m_upperBoundIntegration = 30
- std::vector< double > m_eps

  relative permittivity of each layer
- std::vector< double > m_epsHolder
- std::vector< double > m_d

  thickness of each layer
- std::vector< double > m_z
- std::vector< bool > m_conductive

  Flag whether a layer is conductive.
- TF2 m_hIntegrand
- TF1 m_wpStripIntegral

  Weighting potential integrand for strips.
- TF2 m_wpPixelIntegral

  Weighting potential integrand for pixels.
- std::vector< std::vector< std::vector< int > > > m_sigmaMatrix
- std::vector< std::vector< std::vector< int > > > m_thetaMatrix
- std::vector< std::vector< double > > m_cMatrix

  c-matrixl.
- std::vector< std::vector< double > > m_vMatrix

  v-matrixl.
- std::vector< std::vector< double > > m_gMatrix

  g-matrixl.
- std::vector< std::vector< double > > m_wMatrix

  w-matrixl.
- int m_currentLayer = 0

  Index of the current layer.
- double m_currentPosition = -1
- Medium * m_medium = nullptr
- std::vector< std::string > m_readout
- std::vector< Electrode > m_readout_p

Static Private Attributes

- static constexpr double m_Vw = 1.

### 7.29.1 Detailed Description

Component for parallel-plate geometries.
Definition at line 19 of file ComponentParallelPlate.hh.

### 7.29.2 Member Enumeration Documentation

structureelectrode

enum Garfield::ComponentParallelPlate::structureelectrode [private]
Possible readout groups.

Enumerator

| | |
|---|---|
| NotSet | |
| Plane | |
| Strip | |
| Pixel | |

Definition at line 214 of file ComponentParallelPlate.hh.
00214 { NotSet = -1, Plane, Strip, Pixel };

### 7.29.3 Constructor & Destructor Documentation

ComponentParallelPlate()

Garfield::ComponentParallelPlate::ComponentParallelPlate ()
Constructor.

~ComponentParallelPlate()

Garfield::ComponentParallelPlate::~ComponentParallelPlate () [inline]
Destructor.
Definition at line 24 of file ComponentParallelPlate.hh.
00024 {}

### 7.29.4 Member Function Documentation

AddPixel()

void Garfield::ComponentParallelPlate::AddPixel (
double x,
double z,
double lx,
double lz,
const std::string & label,
bool fromAnode = true)
Add a pixel electrode.

Parameters

| | |
|---|---|
| x,z | position of the center of the electrode in the xz-plane. |
| lx | width in the along $x$. |
| lz | width in the along $z$. |
| label | give name using a string. |
| fromAnode | is *true* is the electrode is the andode and *false* if it is the cathode. |

AddPlane()

void Garfield::ComponentParallelPlate::AddPlane (
const std::string & label,
bool fromAnode = true)
Add plane electrode, if you want to read the signal from the cathode set the second argument to false.

### AddStrip()

```
void Garfield::ComponentParallelPlate::AddStrip (
                double z,
                double lz,
                const std::string & label,
                bool fromAnode = true)
```

Add strip electrode.

### CalculateDynamicalWeightingPotential()

```
void Garfield::ComponentParallelPlate::CalculateDynamicalWeightingPotential (
                const Electrode & el)  [private]
```

### constEFieldLayer()

```
double Garfield::ComponentParallelPlate::constEFieldLayer (
                const int indexLayer)  [inline], [private]
```

Definition at line 278 of file ComponentParallelPlate.hh.

```
00278                                          {
00279     if (m_conductive[indexLayer]) return 0.;
00280     double invEz = 0;
00281     for (int i = 1; i <= m_N - 1; i++) {
00282       // TODO!
00283       if (m_conductive[indexLayer]) continue;
00284       invEz -= m_d[i - 1] / m_epsHolder[i - 1];
00285     }
00286     return m_V / (m_epsHolder[indexLayer - 1] * invEz);
00287   }
```

### constructGeometryFunction()

```
void Garfield::ComponentParallelPlate::constructGeometryFunction (
                const int N,
                const std::vector< double > & d)  [private]
```

### constructGeometryMatrices()

```
void Garfield::ComponentParallelPlate::constructGeometryMatrices (
                const int N)  [private]
```

### constWEFieldLayer()

```
double Garfield::ComponentParallelPlate::constWEFieldLayer (
                const int indexLayer)  [inline], [private]
```

Definition at line 256 of file ComponentParallelPlate.hh.

```
00256                                          {
00257     double invEz = 0;
00258     for (int i = 1; i <= m_N - 1; i++) {
00259       invEz += m_d[i - 1] / m_epsHolder[i - 1];
00260     }
00261     return 1 / (m_epsHolder[indexLayer - 1] * invEz);
00262   }
```

### decToBinary()

```
bool Garfield::ComponentParallelPlate::decToBinary (
                int n,
                std::vector< int > & binaryNum)  [private]
```

### DisablePotentialCalculationOutsideGasGap()

```
void Garfield::ComponentParallelPlate::DisablePotentialCalculationOutsideGasGap ()  [inline]
```

Definition at line 155 of file ComponentParallelPlate.hh.

```
00155                                          {
00156     m_getPotentialInPlate = false;
00157   }
```

### ElectricField() [1/2]

```
void Garfield::ComponentParallelPlate::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)    [override]
```

### ElectricField() [2/2]

```
void Garfield::ComponentParallelPlate::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)    [override]
```

### FindWeightingPotentialInGrid()

```
double Garfield::ComponentParallelPlate::FindWeightingPotentialInGrid (
                Electrode & el,
                const double x,
                const double y,
                const double z)    [private]
```

### GetBoundingBox()

```
bool Garfield::ComponentParallelPlate::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

### getLayer()

```
bool Garfield::ComponentParallelPlate::getLayer (
                const double y,
                int & m,
                double & epsM)    [inline]
```

Definition at line 119 of file ComponentParallelPlate.hh.

```
00119                                                {
00120     m = -1;
00121     if (y < m_z[0]) return false;
00122     for (int i = 1; i < m_N; i++) {
00123       if (y <= m_z[i]) {
00124         m = i;
00125         break;
00126       }
00127     }
00128     if (m == -1) return false;
00129     epsM = m_epsHolder[m - 1];
00130     return true;
00131   }
```

### GetMedium()

```
Medium * Garfield::ComponentParallelPlate::GetMedium (
            const double x,
            const double y,
            const double z)   [override]
```

### getPermittivityFromLayer()

```
void Garfield::ComponentParallelPlate::getPermittivityFromLayer (
            int m,
            double & eps)   [inline]
```

Definition at line 133 of file ComponentParallelPlate.hh.

```
00133                                                 {
00134     eps = m_epsHolder.at(m - 1);
00135   }
```

### GetVoltageRange()

```
bool Garfield::ComponentParallelPlate::GetVoltageRange (
            double & vmin,
            double & vmax)   [override]
```

### getZBoundFromLayer()

```
void Garfield::ComponentParallelPlate::getZBoundFromLayer (
            int m,
            double & zbottom,
            double & ztop)   [inline]
```

Definition at line 137 of file ComponentParallelPlate.hh.

```
00137                                                       {
00138     ztop = m_z.at(m);
00139     zbottom = m_z.at(m - 1);
00140   }
```

### IndexOfGasGaps()

```
void Garfield::ComponentParallelPlate::IndexOfGasGaps (
            std::vector< int > & indexGasGap)   [inline]
```

Definition at line 144 of file ComponentParallelPlate.hh.

```
00144                                                 {
00145     indexGasGap = {};
00146     for (int i = 1; i < m_N; i++) {
00147       if (!m_conductive[i]) indexGasGap.push_back(i);
00148     }
00149   }
```

### IntegratePromptPotential()

```
double Garfield::ComponentParallelPlate::IntegratePromptPotential (
            const Electrode & el,
            const double x,
            const double y,
            const double z)   [private]
```

### LayerUpdate()

```
void Garfield::ComponentParallelPlate::LayerUpdate (
            const double z,
            const int im,
            const double epsM)   [inline], [private]
```

Definition at line 293 of file ComponentParallelPlate.hh.

```
00293                                                 {
00294     if (z == m_currentPosition) return;
00295
```

```
00296     m_currentPosition = z;
00297
00298     if (im != m_currentLayer) {
00299       m_currentLayer = im;
00300       for (int i = 0; i < im - 1; i++) m_eps[i] = m_epsHolder[i];
00301       m_eps[im - 1] = epsM;
00302       m_eps[im] = epsM;
00303       for (int i = im + 1; i < m_N; i++) m_eps[i] = m_epsHolder[i - 1];
00304     }
00305
00306     double diff1 = m_z[im] - z;
00307     double diff2 = z - m_z[im - 1];
00308
00309     std::vector<double> d(m_N, 0.);
00310     for (int i = 0; i < im - 1; i++) d[i] = m_d[i];
00311     d[im - 1] = diff2;
00312     d[im] = diff1;
00313     for (int i = im + 1; i < m_N; i++) d[i] = m_d[i - 1];
00314     // TODO::Construct c and g matrices only for im != m_currentLayer.
00315     constructGeometryFunction(m_N, d);
00316   };
```

## LoadWeightingPotentialGrid()

void Garfield::ComponentParallelPlate::LoadWeightingPotentialGrid (
const std::string & label) [inline]

This will load a previously calculated grid of time-dependent weighting potential values.
Definition at line 97 of file ComponentParallelPlate.hh.

```
00097                                                 {
00098     for (auto &electrode : m_readout_p) {
00099       if (electrode.label != label) continue;
00100       if (electrode.grid.LoadWeightingField(label + "map", "xyz", true)) {
00101         std::cout << m_className << "::LoadWeightingPotentialGrid: "
00102                   << "Weighting potential set for " << label << ".\n";
00103         electrode.m_usegrid = true;
00104         return;
00105       }
00106     }
00107     std::cerr << m_className
00108               << "::LoadWeightingPotentialGrid: Could not find file for "
00109               << label << ".\n";
00110   }
```

## Nsigma()

bool Garfield::ComponentParallelPlate::Nsigma (
int N,
std::vector< std::vector< int > > & sigmaMatrix) [private]

## Ntheta()

bool Garfield::ComponentParallelPlate::Ntheta (
int N,
std::vector< std::vector< int > > & thetaMatrix,
std::vector< std::vector< int > > & sigmaMatrix) [private]

## NumberOfLayers()

int Garfield::ComponentParallelPlate::NumberOfLayers () [inline]

Definition at line 142 of file ComponentParallelPlate.hh.

```
00142 { return m_N - 1; }
```

## Reset()

void Garfield::ComponentParallelPlate::Reset () [override], [private]

## setHIntegrand()

void Garfield::ComponentParallelPlate::setHIntegrand () [private]

SetIntegrationPrecision()

void Garfield::ComponentParallelPlate::SetIntegrationPrecision (
const double eps) [inline]
Definition at line 151 of file ComponentParallelPlate.hh.
00151 { m_precision = eps; }

SetIntegrationUpperbound()

void Garfield::ComponentParallelPlate::SetIntegrationUpperbound (
const double p) [inline]
Definition at line 153 of file ComponentParallelPlate.hh.
00153 { m_upperBoundIntegration = p; }

SetMedium()

void Garfield::ComponentParallelPlate::SetMedium (
Medium * medium) [inline]
Setting the medium.
Definition at line 72 of file ComponentParallelPlate.hh.
00072 { m_medium = medium; }

Setup()

void Garfield::ComponentParallelPlate::Setup (
const unsigned int N,
std::vector< double > eps,
std::vector< double > d,
const double V,
std::vector< int > sigmaIndex = {})
Define the geometry.

Parameters

| | |
|---|---|
| N | amount of layers in the geometry, this includes the gas gaps $y$. |
| d | thickness of the layers starting from the bottom to the top layer along $y$. |
| eps | relative permittivities of the layers starting from the bottom to the top layer along $y$ . Here, the gas gaps having a value of 1. |
| sigmaIndex | Indices of the resistive layers (optional). |
| V | applied potential difference between the parallel plates. |

SetWeightingPotentialGrid()

void Garfield::ComponentParallelPlate::SetWeightingPotentialGrid (
const double xmin,
const double xmax,
const double xsteps,
const double ymin,
const double ymax,
const double ysteps,
const double zmin,
const double zmax,
const double zsteps,
const std::string & label)
Calculate time-dependent weighting potential on a grid.

Parameters

| | |
|---|---|
| xmin,ymin,zmin | minimum value of the interval in the $x$-, $y$- and $z$-direction. |
| xmax,ymax,zmax | maximum value of the interval in the $x$-, $y$- and $z$-direction. |
| xsteps,ysteps,zsteps | mumber of grid nodes in the $x$-, $y$- and $z$-direction. |
| label | give name using a string. |

### SetWeightingPotentialGrids()

```
void Garfield::ComponentParallelPlate::SetWeightingPotentialGrids (
                const double xmin,
                const double xmax,
                const double xsteps,
                const double ymin,
                const double ymax,
                const double ysteps,
                const double zmin,
                const double zmax,
                const double zsteps)
```

This will calculate all electrodes time-dependent weighting potential on the specified grid.

### setwpPixelIntegrand()

```
void Garfield::ComponentParallelPlate::setwpPixelIntegrand ()  [private]
```

### setwpStripIntegrand()

```
void Garfield::ComponentParallelPlate::setwpStripIntegrand ()  [private]
```

### UpdatePeriodicity()

```
void Garfield::ComponentParallelPlate::UpdatePeriodicity ()  [override], [private]
```

### WeightingPotential()

```
double Garfield::ComponentParallelPlate::WeightingPotential (
                const double x,
                const double y,
                const double z,
                const std::string & label)  [override]
```

### wpPlane()

```
double Garfield::ComponentParallelPlate::wpPlane (
                const double z)  [inline], [private]
```

Definition at line 265 of file ComponentParallelPlate.hh.

```
00265                                {
00266     int im = -1;
00267     double epsM = -1;
00268     if (!getLayer(z, im, epsM)) return 0.;
00269     double v = 1 - (z - m_z[im - 1]) * constWEFieldLayer(im);
00270     for (int i = 1; i <= im - 1; i++) {
00271       v -= m_d[i - 1] * constWEFieldLayer(i);
00272     }
00273
00274     return v;
00275   }
```

## 7.29.5 Member Data Documentation

### m_cMatrix

```
std::vector<std::vector<double> > Garfield::ComponentParallelPlate::m_cMatrix  [private]
```

c-matrixl.
Definition at line 191 of file ComponentParallelPlate.hh.

### m_conductive

std::vector<bool> Garfield::ComponentParallelPlate::m_conductive [private]
Flag whether a layer is conductive.
Definition at line 177 of file ComponentParallelPlate.hh.

### m_currentLayer

int Garfield::ComponentParallelPlate::m_currentLayer = 0 [private]
Index of the current layer.
Definition at line 196 of file ComponentParallelPlate.hh.

### m_currentPosition

double Garfield::ComponentParallelPlate::m_currentPosition = -1 [private]
Definition at line 197 of file ComponentParallelPlate.hh.

### m_d

std::vector<double> Garfield::ComponentParallelPlate::m_d [private]
thickness of each layer
Definition at line 173 of file ComponentParallelPlate.hh.

### m_eps

std::vector<double> Garfield::ComponentParallelPlate::m_eps [private]
relative permittivity of each layer
Definition at line 171 of file ComponentParallelPlate.hh.

### m_epsHolder

std::vector<double> Garfield::ComponentParallelPlate::m_epsHolder [private]
Definition at line 172 of file ComponentParallelPlate.hh.

### m_getPotentialInPlate

bool Garfield::ComponentParallelPlate::m_getPotentialInPlate = true [private]
Definition at line 165 of file ComponentParallelPlate.hh.

### m_gMatrix

std::vector<std::vector<double> > Garfield::ComponentParallelPlate::m_gMatrix [private]
g-matrixl.
Definition at line 193 of file ComponentParallelPlate.hh.

### m_hIntegrand

TF2 Garfield::ComponentParallelPlate::m_hIntegrand [private]
Definition at line 179 of file ComponentParallelPlate.hh.

### m_medium

Medium* Garfield::ComponentParallelPlate::m_medium = nullptr [private]
Definition at line 199 of file ComponentParallelPlate.hh.

### m_N

int Garfield::ComponentParallelPlate::m_N = 0 [private]
Number of layers.
Definition at line 167 of file ComponentParallelPlate.hh.

### m_precision

double Garfield::ComponentParallelPlate::m_precision = 1.e-12 [private]
Definition at line 160 of file ComponentParallelPlate.hh.

### m_readout

std::vector<std::string> Garfield::ComponentParallelPlate::m_readout [private]
Definition at line 217 of file ComponentParallelPlate.hh.

### m_readout_p

std::vector<Electrode> Garfield::ComponentParallelPlate::m_readout_p [private]
Definition at line 218 of file ComponentParallelPlate.hh.

### m_sigmaMatrix

std::vector<std::vector<std::vector<int> > > Garfield::ComponentParallelPlate::m_sigmaMatrix [private]
Definition at line 184 of file ComponentParallelPlate.hh.

### m_thetaMatrix

std::vector<std::vector<std::vector<int> > > Garfield::ComponentParallelPlate::m_thetaMatrix [private]
Definition at line 187 of file ComponentParallelPlate.hh.

### m_upperBoundIntegration

double Garfield::ComponentParallelPlate::m_upperBoundIntegration = 30 [private]
Definition at line 169 of file ComponentParallelPlate.hh.

### m_V

double Garfield::ComponentParallelPlate::m_V = 0. [private]
Voltage difference between the parallel plates.
Definition at line 163 of file ComponentParallelPlate.hh.

### m_vMatrix

std::vector<std::vector<double> > Garfield::ComponentParallelPlate::m_vMatrix [private]
v-matrixl.
Definition at line 192 of file ComponentParallelPlate.hh.

### m_Vw

double Garfield::ComponentParallelPlate::m_Vw = 1. [static], [constexpr], [private]
Definition at line 161 of file ComponentParallelPlate.hh.

### m_wMatrix

std::vector<std::vector<double> > Garfield::ComponentParallelPlate::m_wMatrix [private]
w-matrixl.
Definition at line 194 of file ComponentParallelPlate.hh.

m_wpPixelIntegral

TF2 Garfield::ComponentParallelPlate::m_wpPixelIntegral [private]
Weighting potential integrand for pixels.
Definition at line 182 of file ComponentParallelPlate.hh.

m_wpStripIntegral

TF1 Garfield::ComponentParallelPlate::m_wpStripIntegral [private]
Weighting potential integrand for strips.
Definition at line 181 of file ComponentParallelPlate.hh.

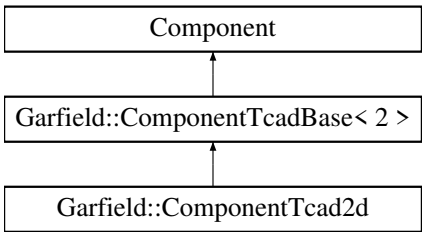
m_z

std::vector<double> Garfield::ComponentParallelPlate::m_z [private]
Definition at line 174 of file ComponentParallelPlate.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentParallelPlate.hh

## 7.30 Garfield::ComponentTcad2d Class Reference

Interpolation in a two-dimensional field map created by Sentaurus Device.
#include <ComponentTcad2d.hh>
Inheritance diagram for Garfield::ComponentTcad2d:




Public Member Functions

- ComponentTcad2d ()
  Constructor.
- ~ComponentTcad2d ()
  Destructor.
- void SetRangeZ (const double zmin, const double zmax)
  Set the z-extent of the bounding box (default: unlimited).
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax, int &type) const
  Retrieve the properties of an element.
- bool GetNode (const size_t i, double &x, double &y, double &z) const override
  Get the coordinates of a mesh node.
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- Medium * GetMedium (const double x, const double y, const double z) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override

Public Member Functions inherited from Garfield::ComponentTcadBase< 2 >

- ComponentTcadBase ()=delete

  Default constructor.
- virtual ~ComponentTcadBase ()

  Destructor.
- bool Initialise (const std::string &gridfilename, const std::string &datafilename)

  Import mesh and field map from files.
- bool SetWeightingField (const std::string &datfile1, const std::string &datfile2, const double dv, const std::string &label)

  Import field maps defining the prompt weighting field and potential.
- bool SetWeightingPotential (const std::string &datfile1, const std::string &datfile2, const double dv, const std::string &label)

  Import field maps defining the prompt weighting field and potential.
- bool SetWeightingFieldShift (const std::string &label, const double x, const double y, const double z)

  Shift the maps of weighting field/potential for a given electrode with respect to the original mesh.
- bool SetDynamicWeightingPotential (const std::string &datfile1, const std::string &datfile2, const double dv, const double t, const std::string &label)

  Import time-dependent weighting potentials at t >= 0.
- bool SetDynamicWeightingField (const std::string &datfile1, const std::string &datfile2, const double dv, const double t, const std::string &label)

  Import time-dependent weighting fields at t >= 0.
- void PrintRegions () const

  List all currently defined regions.
- size_t GetNumberOfRegions () const

  Get the number of regions in the device.
- void GetRegion (const size_t ireg, std::string &name, bool &active) const

  Get the name and "active volume" flag of a region.
- void SetDriftRegion (const size_t ireg)

  Make a region active ("driftable").
- void UnsetDriftRegion (const size_t ireg)

  Make a region inactive.
- void SetMedium (const size_t ireg, Medium *m)

  Set the medium to be associated to a given region.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &region, bool &active) const override
- size_t GetNumberOfNodes () const override
- void EnableVelocityMap (const bool on)

  Switch use of the imported velocity map on/off.
- size_t GetNumberOfDonors ()

  Get the number of donor states found in the map.
- size_t GetNumberOfAcceptors ()

  Get the number of acceptor states found in the map.
- bool SetDonor (const size_t donorNumber, const double exsec, const double hxsec, const double concentration)

  Set the properties of a donor-type defect state.
- bool SetAcceptor (const size_t acceptorNumber, const double exsec, const double hxsec, const double concentration)

  Set the properties of an acceptor-type defect state.
- void EnableAlphaMap (const bool on=true)

Use the imported impact ionisation map or not.

- void EnableTrapOccupationMap (const bool on=true)

  Use the imported trapping map or not.
- void EnableLifetimeMap (const bool on=true)

  Use the imported lifetime map or not.
- bool GetElectronMobility (const double x, const double y, const double z, double &mob)

  Get the electron mobility at a given point in the mesh.
- bool GetHoleMobility (const double x, const double y, const double z, double &mob)

  Get the hole mobility at a given point in the mesh.
- double ElectronLifetime (const double x, const double y, const double z)
- double HoleLifetime (const double x, const double y, const double z)
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- const std::vector< double > & DelayedSignalTimes (const std::string &label) override
- void DelayedWeightingField (const double x, const double y, const double z, const double t, double &wx, double &wy, double &wz, const std::string &label) override
- double DelayedWeightingPotential (const double x, const double y, const double z, const double t, const std::string &label) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool HasVelocityMap () const override
- bool ElectronVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override
- bool HoleVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override
- bool HasTownsendMap () const override
- bool HasAttachmentMap () const override
- bool HasMobilityMap () const override
- bool ElectronAttachment (const double x, const double y, const double z, double &eta) override
- bool HoleAttachment (const double x, const double y, const double z, double &eta) override
- bool ElectronMobility (const double x, const double y, const double z, double &mu) override
- bool HoleMobility (const double x, const double y, const double z, double &mu) override
- bool ElectronTownsend (const double x, const double y, const double z, double &alpha) override
- bool HoleTownsend (const double x, const double y, const double z, double &alpha) override

Private Member Functions

- void Reset () override
- size_t FindElement (const double x, const double y, std::array< double, nMaxVertices > &w) const
- bool InElement (const double x, const double y, const Element &element, std::array< double, nMaxVertices > &w) const
- bool InRectangle (const double x, const double y, const Element &element, std::array< double, nMaxVertices > &w) const
- bool InTriangle (const double x, const double y, const Element &element, std::array< double, nMaxVertices > &w) const
- bool OnLine (const double x, const double y, const Element &element, std::array< double, nMaxVertices > &w) const
- bool AtPoint (const double x, const double y, const Element &element, std::array< double, nMaxVertices > &w) const
- bool Interpolate (const double x, const double y, const double z, const std::vector< double > &field, double &f) override
- bool Interpolate (const double x, const double y, const double z, const std::vector< std::array< double, 2 > > &field, double &fx, double &fy, double &fz) override
- void FillTree () override

Private Attributes

- bool m_hasRangeZ = false
- std::unique_ptr< QuadTree > m_tree

Additional Inherited Members

Protected Member Functions inherited from Garfield::ComponentTcadBase< 2 >

- void UpdatePeriodicity () override
- void Cleanup ()
- size_t FindRegion (const std::string &name) const
- void MapCoordinates (std::array< double, N > &x, std::array< bool, N > &mirr) const
- bool InBoundingBox (const std::array< double, N > &x) const
- void UpdateAttachment ()
- void ComputeEtaFromLifetime ()
- void ComputeEtaFromTraps ()
- bool GetOffset (const std::string &label, double &dx, double &dy, double &dz) const
- bool LoadGrid (const std::string &gridfilename)
- bool LoadData (const std::string &datafilename)
- bool ReadDataset (std::ifstream &datafile, const std::string &dataset)
- bool LoadWeightingField (const std::string &datafilename, std::vector< std::array< double, N > > &wf, std::vector< double > &wp)

Static Protected Member Functions inherited from Garfield::ComponentTcadBase< 2 >

- static unsigned int ElementVertices (const Element &element)

Protected Attributes inherited from Garfield::ComponentTcadBase< 2 >

- std::vector< Region > m_regions
- std::vector< std::array< double, N > > m_vertices
- std::vector< Element > m_elements
- std::vector< double > m_epot
- std::vector< std::array< double, N > > m_efield
- std::map< std::string, std::vector< std::array< double, N > > > m_wfield
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< double > > m_wshift
- std::map< std::string, std::vector< std::vector< std::array< double, N > > > > m_dwf
- std::map< std::string, std::vector< std::vector< double > > > m_dwp
- std::map< std::string, std::vector< double > > m_dwtf
- std::map< std::string, std::vector< double > > m_dwtp
- std::vector< std::array< double, N > > m_eVelocity
- std::vector< std::array< double, N > > m_hVelocity
- std::vector< double > m_eMobility
- std::vector< double > m_hMobility
- std::vector< double > m_eAlpha
- std::vector< double > m_hAlpha
- std::vector< double > m_eLifetime
- std::vector< double > m_hLifetime
- std::vector< std::vector< float > > m_donorOcc
- std::vector< std::vector< float > > m_acceptorOcc
- std::vector< double > m_eEta
- std::vector< double > m_hEta
- std::vector< Defect > m_donors
- std::vector< Defect > m_acceptors
- bool m_useVelocityMap

- bool m_useTrapOccMap
- bool m_useLifetimeMap
- bool m_useAlphaMap
- std::array< double, 3 > m_bbMin
- std::array< double, 3 > m_bbMax
- double m_pMin
- double m_pMax

Static Protected Attributes inherited from Garfield::ComponentTcadBase< 2 >

- static constexpr size_t nMaxVertices

### 7.30.1 Detailed Description

Interpolation in a two-dimensional field map created by Sentaurus Device.
Definition at line 14 of file ComponentTcad2d.hh.

### 7.30.2 Constructor & Destructor Documentation

#### ComponentTcad2d()

Garfield::ComponentTcad2d::ComponentTcad2d ()
Constructor.

#### ~ComponentTcad2d()

Garfield::ComponentTcad2d::~ComponentTcad2d () [inline]
Destructor.
Definition at line 19 of file ComponentTcad2d.hh.
00019 {}

### 7.30.3 Member Function Documentation

#### AtPoint()

bool Garfield::ComponentTcad2d::AtPoint (
const double x,
const double y,
const Element & element,
std::array< double, nMaxVertices > & w) const [private]

#### DelayedWeightingPotentials()

void Garfield::ComponentTcad2d::DelayedWeightingPotentials (
const double x,
const double y,
const double z,
const std::string & label,
std::vector< double > & dwp) [override]

#### ElectricField() [1/2]

void Garfield::ComponentTcad2d::ElectricField (
const double x,
const double y,
const double z,
double & ex,
double & ey,
double & ez,
double & v,
Medium *& m,
int & status) [override]

### ElectricField() [2/2]

```
void Garfield::ComponentTcad2d::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)   [inline], [override]
```

Definition at line 40 of file ComponentTcad2d.hh.

```
00041                                                                 {
00042     double v = 0.;
00043     ElectricField(x, y, z, ex, ey, ez, v, m, status);
00044   }
```

### FillTree()

```
void Garfield::ComponentTcad2d::FillTree ()   [override], [private], [virtual]
```

Implements Garfield::ComponentTcadBase< 2 >.

### FindElement()

```
size_t Garfield::ComponentTcad2d::FindElement (
                const double x,
                const double y,
                std::array< double, nMaxVertices > & w) const   [private]
```

### GetBoundingBox()

```
bool Garfield::ComponentTcad2d::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)   [override]
```

### GetElement()

```
bool Garfield::ComponentTcad2d::GetElement (
                const size_t i,
                double & vol,
                double & dmin,
                double & dmax,
                int & type) const
```

Retrieve the properties of an element.

Parameters

| | |
|---|---|
| i | index of the element |
| vol | volume |
| dmin | smallest length in the element |
| dmax | largest length in the element |
| type | element type |

### GetElementaryCell()

```
bool Garfield::ComponentTcad2d::GetElementaryCell (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

### GetMedium()

```
Medium * Garfield::ComponentTcad2d::GetMedium (
                const double x,
                const double y,
                const double z)    [override]
```

### GetNode()

```
bool Garfield::ComponentTcad2d::GetNode (
                const size_t i,
                double & x,
                double & y,
                double & z) const    [override]
```

Get the coordinates of a mesh node.

### InElement()

```
bool Garfield::ComponentTcad2d::InElement (
                const double x,
                const double y,
                const Element & element,
                std::array< double, nMaxVertices > & w) const    [private]
```

### InRectangle()

```
bool Garfield::ComponentTcad2d::InRectangle (
                const double x,
                const double y,
                const Element & element,
                std::array< double, nMaxVertices > & w) const    [private]
```

### Interpolate() [1/2]

```
bool Garfield::ComponentTcad2d::Interpolate (
                const double x,
                const double y,
                const double z,
                const std::vector< double > & field,
                double & f)    [override], [private], [virtual]
```

Implements Garfield::ComponentTcadBase< 2 >.

### Interpolate() [2/2]

```
bool Garfield::ComponentTcad2d::Interpolate (
                const double x,
                const double y,
                const double z,
                const std::vector< std::array< double, 2 > > & field,
                double & fx,
```

double & fy,
double & fz) [override], [private]

### InTriangle()

bool Garfield::ComponentTcad2d::InTriangle (
const double x,
const double y,
const Element & element,
std::array< double, nMaxVertices > & w) const [private]

### OnLine()

bool Garfield::ComponentTcad2d::OnLine (
const double x,
const double y,
const Element & element,
std::array< double, nMaxVertices > & w) const [private]

### Reset()

void Garfield::ComponentTcad2d::Reset () [inline], [override], [private]
Definition at line 63 of file ComponentTcad2d.hh.

```
00063                       {
00064     Cleanup();
00065     m_hasRangeZ = false;
00066     m_ready = false;
00067   }
```

### SetRangeZ()

void Garfield::ComponentTcad2d::SetRangeZ (
const double zmin,
const double zmax)
Set the z-extent of the bounding box (default: unlimited).

#### 7.30.4 Member Data Documentation

### m_hasRangeZ

bool Garfield::ComponentTcad2d::m_hasRangeZ = false [private]
Definition at line 58 of file ComponentTcad2d.hh.

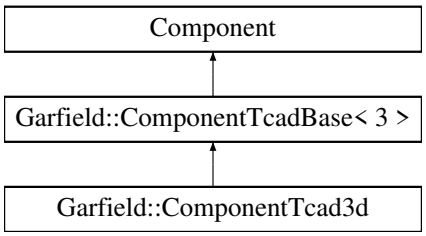
### m_tree

std::unique_ptr<QuadTree> Garfield::ComponentTcad2d::m_tree [private]
Definition at line 61 of file ComponentTcad2d.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentTcad2d.hh

## 7.31 Garfield::ComponentTcad3d Class Reference

Interpolation in a three-dimensional field map created by Sentaurus Device.
#include <ComponentTcad3d.hh>
Inheritance diagram for Garfield::ComponentTcad3d:




Public Member Functions

- ComponentTcad3d ()

  Constructor.
- ~ComponentTcad3d ()

  Destructor.
- bool GetElement (const size_t i, double &vol, double &dmin, double &dmax, int &type) const

  Retrieve the properties of an element.
- bool GetNode (const size_t i, double &x, double &y, double &z) const override

  Get the coordinates of a mesh node.
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- Medium * GetMedium (const double x, const double y, const double z) override
- void DelayedWeightingPotentials (const double x, const double y, const double z, const std::string &label, std::vector< double > &dwp) override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override

Public Member Functions inherited from Garfield::ComponentTcadBase< 3 >

- ComponentTcadBase ()=delete

  Default constructor.
- virtual ~ComponentTcadBase ()

  Destructor.
- bool Initialise (const std::string &gridfilename, const std::string &datafilename)

  Import mesh and field map from files.
- bool SetWeightingField (const std::string &datfile1, const std::string &datfile2, const double dv, const std::string &label)

  Import field maps defining the prompt weighting field and potential.
- bool SetWeightingPotential (const std::string &datfile1, const std::string &datfile2, const double dv, const std::string &label)

  Import field maps defining the prompt weighting field and potential.
- bool SetWeightingFieldShift (const std::string &label, const double x, const double y, const double z)

  Shift the maps of weighting field/potential for a given electrode with respect to the original mesh.
- bool SetDynamicWeightingPotential (const std::string &datfile1, const std::string &datfile2, const double dv, const double t, const std::string &label)

  Import time-dependent weighting potentials at t >= 0.
- bool SetDynamicWeightingField (const std::string &datfile1, const std::string &datfile2, const double dv, const double t, const std::string &label)

  Import time-dependent weighting fields at t >= 0.
- void PrintRegions () const

List all currently defined regions.

- size_t GetNumberOfRegions () const

  Get the number of regions in the device.

- void GetRegion (const size_t ireg, std::string &name, bool &active) const

  Get the name and "active volume" flag of a region.

- void SetDriftRegion (const size_t ireg)

  Make a region active ("driftable").

- void UnsetDriftRegion (const size_t ireg)

  Make a region inactive.

- void SetMedium (const size_t ireg, Medium *m)

  Set the medium to be associated to a given region.

- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &region, bool &active) const override
- size_t GetNumberOfNodes () const override
- void EnableVelocityMap (const bool on)

  Switch use of the imported velocity map on/off.

- size_t GetNumberOfDonors ()

  Get the number of donor states found in the map.

- size_t GetNumberOfAcceptors ()

  Get the number of acceptor states found in the map.

- bool SetDonor (const size_t donorNumber, const double exsec, const double hxsec, const double concentration)

  Set the properties of a donor-type defect state.

- bool SetAcceptor (const size_t acceptorNumber, const double exsec, const double hxsec, const double concentration)

  Set the properties of an acceptor-type defect state.

- void EnableAlphaMap (const bool on=true)

  Use the imported impact ionisation map or not.

- void EnableTrapOccupationMap (const bool on=true)

  Use the imported trapping map or not.

- void EnableLifetimeMap (const bool on=true)

  Use the imported lifetime map or not.

- bool GetElectronMobility (const double x, const double y, const double z, double &mob)

  Get the electron mobility at a given point in the mesh.

- bool GetHoleMobility (const double x, const double y, const double z, double &mob)

  Get the hole mobility at a given point in the mesh.

- double ElectronLifetime (const double x, const double y, const double z)
- double HoleLifetime (const double x, const double y, const double z)
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- const std::vector< double > & DelayedSignalTimes (const std::string &label) override
- void DelayedWeightingField (const double x, const double y, const double z, const double t, double &wx, double &wy, double &wz, const std::string &label) override
- double DelayedWeightingPotential (const double x, const double y, const double z, const double t, const std::string &label) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool HasVelocityMap () const override
- bool ElectronVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override

- bool HoleVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override
- bool HasTownsendMap () const override
- bool HasAttachmentMap () const override
- bool HasMobilityMap () const override
- bool ElectronAttachment (const double x, const double y, const double z, double &eta) override
- bool HoleAttachment (const double x, const double y, const double z, double &eta) override
- bool ElectronMobility (const double x, const double y, const double z, double &mu) override
- bool HoleMobility (const double x, const double y, const double z, double &mu) override
- bool ElectronTownsend (const double x, const double y, const double z, double &alpha) override
- bool HoleTownsend (const double x, const double y, const double z, double &alpha) override

Private Member Functions

- void Reset () override
- size_t FindElement (const double x, const double y, const double z, std::array< double, nMaxVertices > &w) const
- bool InElement (const double x, const double y, const double z, const Element &element, std::array< double, nMaxVertices > &w) const
- bool InTetrahedron (const double x, const double y, const double z, const Element &element, std::array< double, nMaxVertices > &w) const
- bool InTriangle (const double x, const double y, const double z, const Element &element, std::array< double, nMaxVertices > &w) const
- bool Interpolate (const double x, const double y, const double z, const std::vector< double > &field, double &f) override
- bool Interpolate (const double x, const double y, const double z, const std::vector< std::array< double, 3 > > &field, double &fx, double &fy, double &fz) override
- void FillTree () override

Private Attributes

- std::unique_ptr< TetrahedralTree > m_tree

Additional Inherited Members

Protected Member Functions inherited from Garfield::ComponentTcadBase< 3 >

- void UpdatePeriodicity () override
- void Cleanup ()
- size_t FindRegion (const std::string &name) const
- void MapCoordinates (std::array< double, N > &x, std::array< bool, N > &mirr) const
- bool InBoundingBox (const std::array< double, N > &x) const
- void UpdateAttachment ()
- void ComputeEtaFromLifetime ()
- void ComputeEtaFromTraps ()
- bool GetOffset (const std::string &label, double &dx, double &dy, double &dz) const
- bool LoadGrid (const std::string &gridfilename)
- bool LoadData (const std::string &datafilename)
- bool ReadDataset (std::ifstream &datafile, const std::string &dataset)
- bool LoadWeightingField (const std::string &datafilename, std::vector< std::array< double, N > > &wf, std::vector< double > &wp)

Static Protected Member Functions inherited from Garfield::ComponentTcadBase< 3 >

- static unsigned int ElementVertices (const Element &element)

Protected Attributes inherited from Garfield::ComponentTcadBase< 3 >

- std::vector< Region > m_regions
- std::vector< std::array< double, N > > m_vertices
- std::vector< Element > m_elements
- std::vector< double > m_epot
- std::vector< std::array< double, N > > m_efield
- std::map< std::string, std::vector< std::array< double, N > > > m_wfield
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< double > > m_wshift
- std::map< std::string, std::vector< std::vector< std::array< double, N > > > > m_dwf
- std::map< std::string, std::vector< std::vector< double > > > m_dwp
- std::map< std::string, std::vector< double > > m_dwtf
- std::map< std::string, std::vector< double > > m_dwtp
- std::vector< std::array< double, N > > m_eVelocity
- std::vector< std::array< double, N > > m_hVelocity
- std::vector< double > m_eMobility
- std::vector< double > m_hMobility
- std::vector< double > m_eAlpha
- std::vector< double > m_hAlpha
- std::vector< double > m_eLifetime
- std::vector< double > m_hLifetime
- std::vector< std::vector< float > > m_donorOcc
- std::vector< std::vector< float > > m_acceptorOcc
- std::vector< double > m_eEta
- std::vector< double > m_hEta
- std::vector< Defect > m_donors
- std::vector< Defect > m_acceptors
- bool m_useVelocityMap
- bool m_useTrapOccMap
- bool m_useLifetimeMap
- bool m_useAlphaMap
- std::array< double, 3 > m_bbMin
- std::array< double, 3 > m_bbMax
- double m_pMin
- double m_pMax

Static Protected Attributes inherited from Garfield::ComponentTcadBase< 3 >

- static constexpr size_t nMaxVertices

### 7.31.1 Detailed Description

Interpolation in a three-dimensional field map created by Sentaurus Device.
Definition at line 15 of file ComponentTcad3d.hh.

### 7.31.2 Constructor & Destructor Documentation

ComponentTcad3d()

Garfield::ComponentTcad3d::ComponentTcad3d ()
Constructor.

~ComponentTcad3d()

Garfield::ComponentTcad3d::~ComponentTcad3d () [inline]
Destructor.
Definition at line 20 of file ComponentTcad3d.hh.
00020 {}

### 7.31.3 Member Function Documentation

#### DelayedWeightingPotentials()

```
void Garfield::ComponentTcad3d::DelayedWeightingPotentials (
                const double x,
                const double y,
                const double z,
                const std::string & label,
                std::vector< double > & dwp)    [override]
```

#### ElectricField() [1/2]

```
void Garfield::ComponentTcad3d::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)    [override]
```

#### ElectricField() [2/2]

```
void Garfield::ComponentTcad3d::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)    [override]
```

#### FillTree()

```
void Garfield::ComponentTcad3d::FillTree ()    [override], [private], [virtual]
```

Implements Garfield::ComponentTcadBase< 3 >.

#### FindElement()

```
size_t Garfield::ComponentTcad3d::FindElement (
                const double x,
                const double y,
                const double z,
                std::array< double, nMaxVertices > & w) const    [private]
```

#### GetBoundingBox()

```
bool Garfield::ComponentTcad3d::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

### GetElement()

bool Garfield::ComponentTcad3d::GetElement (
const size_t i,
double & vol,
double & dmin,
double & dmax,
int & type) const

Retrieve the properties of an element.

Parameters

| | |
|---|---|
| i | index of the element |
| vol | volume |
| dmin | smallest length in the element |
| dmax | largest length in the element |
| type | element type |

### GetElementaryCell()

bool Garfield::ComponentTcad3d::GetElementaryCell (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [override]

### GetMedium()

Medium * Garfield::ComponentTcad3d::GetMedium (
const double x,
const double y,
const double z) [override]

### GetNode()

bool Garfield::ComponentTcad3d::GetNode (
const size_t i,
double & x,
double & y,
double & z) const [override]

Get the coordinates of a mesh node.

### InElement()

bool Garfield::ComponentTcad3d::InElement (
const double x,
const double y,
const double z,
const Element & element,
std::array< double, nMaxVertices > & w) const [inline], [private]

Definition at line 61 of file ComponentTcad3d.hh.

```
00063                                                {
00064     if (x < element.bbMin[0] || x > element.bbMax[0] || y < element.bbMin[1] ||
00065         y > element.bbMax[1] || z < element.bbMin[2] || z > element.bbMax[2]) {
00066       return false;
00067     }
00068     bool inside = false;
00069     switch (element.type) {
```

```
00070       case 2:
00071         if (InTriangle(x, y, z, element, w)) inside = true;
00072         break;
00073       case 5:
00074         if (InTetrahedron(x, y, z, element, w)) inside = true;
00075         break;
00076       default:
00077         std::cerr « m_className « "::InElement:\n"
00078                   « "    Invalid element type (" « element.type « ").\n";
00079         break;
00080     }
00081     return inside;
00082   }
```

### Interpolate() [1/2]

```
bool Garfield::ComponentTcad3d::Interpolate (
                const double x,
                const double y,
                const double z,
                const std::vector< double > & field,
                double & f)    [override], [private], [virtual]
```

Implements Garfield::ComponentTcadBase< 3 >.

### Interpolate() [2/2]

```
bool Garfield::ComponentTcad3d::Interpolate (
                const double x,
                const double y,
                const double z,
                const std::vector< std::array< double, 3 > > & field,
                double & fx,
                double & fy,
                double & fz)    [override], [private]
```

### InTetrahedron()

```
bool Garfield::ComponentTcad3d::InTetrahedron (
                const double x,
                const double y,
                const double z,
                const Element & element,
                std::array< double, nMaxVertices > & w) const    [private]
```

### InTriangle()

```
bool Garfield::ComponentTcad3d::InTriangle (
                const double x,
                const double y,
                const double z,
                const Element & element,
                std::array< double, nMaxVertices > & w) const    [private]
```

### Reset()

```
void Garfield::ComponentTcad3d::Reset ()    [inline], [override], [private]
```

Definition at line 54 of file ComponentTcad3d.hh.

```
00054                       {
00055     Cleanup();
00056     m_ready = false;
00057   }
```

### 7.31.4 Member Data Documentation

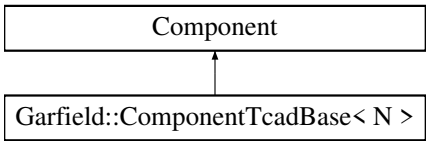
m_tree

std::unique_ptr<TetrahedralTree> Garfield::ComponentTcad3d::m_tree [private]
Definition at line 52 of file ComponentTcad3d.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentTcad3d.hh

## 7.32 Garfield::ComponentTcadBase< N > Class Template Reference

Interpolation in a field map created by Sentaurus Device.
#include <ComponentTcadBase.hh>
Inheritance diagram for Garfield::ComponentTcadBase< N >:




Classes

- struct Defect
- struct Element
- struct Region

Public Member Functions

- ComponentTcadBase ()=delete

  Default constructor.
- ComponentTcadBase (const std::string &name)

  Constructor.
- virtual ~ComponentTcadBase ()

  Destructor.
- bool Initialise (const std::string &gridfilename, const std::string &datafilename)

  Import mesh and field map from files.
- bool SetWeightingField (const std::string &datfile1, const std::string &datfile2, const double dv, const std::string &label)

  Import field maps defining the prompt weighting field and potential.
- bool SetWeightingPotential (const std::string &datfile1, const std::string &datfile2, const double dv, const std::string &label)

  Import field maps defining the prompt weighting field and potential.
- bool SetWeightingFieldShift (const std::string &label, const double x, const double y, const double z)

  Shift the maps of weighting field/potential for a given electrode with respect to the original mesh.
- bool SetDynamicWeightingPotential (const std::string &datfile1, const std::string &datfile2, const double dv, const double t, const std::string &label)

  Import time-dependent weighting potentials at t >= 0.
- bool SetDynamicWeightingField (const std::string &datfile1, const std::string &datfile2, const double dv, const double t, const std::string &label)

  Import time-dependent weighting fields at t >= 0.
- void PrintRegions () const

  List all currently defined regions.
- size_t GetNumberOfRegions () const

Get the number of regions in the device.

- void GetRegion (const size_t ireg, std::string &name, bool &active) const

  Get the name and "active volume" flag of a region.
- void SetDriftRegion (const size_t ireg)

  Make a region active ("driftable").
- void UnsetDriftRegion (const size_t ireg)

  Make a region inactive.
- void SetMedium (const size_t ireg, Medium *m)

  Set the medium to be associated to a given region.
- void SetMedium (const std::string &material, Medium *m)

  Set the medium to be associated to all regions with a given material.
- size_t GetNumberOfElements () const override
- bool GetElementNodes (const size_t i, std::vector< size_t > &nodes) const override
- bool GetElementRegion (const size_t i, size_t &region, bool &active) const override
- size_t GetNumberOfNodes () const override
- void EnableVelocityMap (const bool on)

  Switch use of the imported velocity map on/off.
- size_t GetNumberOfDonors ()

  Get the number of donor states found in the map.
- size_t GetNumberOfAcceptors ()

  Get the number of acceptor states found in the map.
- bool SetDonor (const size_t donorNumber, const double exsec, const double hxsec, const double concentration)

  Set the properties of a donor-type defect state.
- bool SetAcceptor (const size_t acceptorNumber, const double exsec, const double hxsec, const double concentration)

  Set the properties of an acceptor-type defect state.
- void EnableAlphaMap (const bool on=true)

  Use the imported impact ionisation map or not.
- void EnableTrapOccupationMap (const bool on=true)

  Use the imported trapping map or not.
- void EnableLifetimeMap (const bool on=true)

  Use the imported lifetime map or not.
- bool GetElectronMobility (const double x, const double y, const double z, double &mob)

  Get the electron mobility at a given point in the mesh.
- bool GetHoleMobility (const double x, const double y, const double z, double &mob)

  Get the hole mobility at a given point in the mesh.
- double ElectronLifetime (const double x, const double y, const double z)
- double HoleLifetime (const double x, const double y, const double z)
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- const std::vector< double > & DelayedSignalTimes (const std::string &label) override
- void DelayedWeightingField (const double x, const double y, const double z, const double t, double &wx, double &wy, double &wz, const std::string &label) override
- double DelayedWeightingPotential (const double x, const double y, const double z, const double t, const std::string &label) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool HasVelocityMap () const override
- bool ElectronVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override

- bool HoleVelocity (const double x, const double y, const double z, double &vx, double &vy, double &vz) override
- bool HasTownsendMap () const override
- bool HasAttachmentMap () const override
- bool HasMobilityMap () const override
- bool ElectronAttachment (const double x, const double y, const double z, double &eta) override
- bool HoleAttachment (const double x, const double y, const double z, double &eta) override
- bool ElectronMobility (const double x, const double y, const double z, double &mu) override
- bool HoleMobility (const double x, const double y, const double z, double &mu) override
- bool ElectronTownsend (const double x, const double y, const double z, double &alpha) override
- bool HoleTownsend (const double x, const double y, const double z, double &alpha) override

Protected Member Functions

- void UpdatePeriodicity () override
- void Cleanup ()
- virtual bool Interpolate (const double x, const double y, const double z, const std::vector< double > &field, double &f)=0
- virtual bool Interpolate (const double x, const double y, const double z, const std::vector< std::array< double, N > > &field, double &fx, double &fy, double &fz)=0
- virtual void FillTree ()=0
- size_t FindRegion (const std::string &name) const
- void MapCoordinates (std::array< double, N > &x, std::array< bool, N > &mirr) const
- bool InBoundingBox (const std::array< double, N > &x) const
- void UpdateAttachment ()
- void ComputeEtaFromLifetime ()
- void ComputeEtaFromTraps ()
- bool GetOffset (const std::string &label, double &dx, double &dy, double &dz) const
- bool LoadGrid (const std::string &gridfilename)
- bool LoadData (const std::string &datafilename)
- bool ReadDataset (std::ifstream &datafile, const std::string &dataset)
- bool LoadWeightingField (const std::string &datafilename, std::vector< std::array< double, N > > &wf, std::vector< double > &wp)

Static Protected Member Functions

- static unsigned int ElementVertices (const Element &element)

Protected Attributes

- std::vector< Region > m_regions
- std::vector< std::array< double, N > > m_vertices
- std::vector< Element > m_elements
- std::vector< double > m_epot
- std::vector< std::array< double, N > > m_efield
- std::map< std::string, std::vector< std::array< double, N > > > m_wfield
- std::map< std::string, std::vector< double > > m_wpot
- std::map< std::string, std::vector< double > > m_wshift
- std::map< std::string, std::vector< std::vector< std::array< double, N > > > > m_dwf
- std::map< std::string, std::vector< std::vector< double > > > m_dwp
- std::map< std::string, std::vector< double > > m_dwtf
- std::map< std::string, std::vector< double > > m_dwtp
- std::vector< std::array< double, N > > m_eVelocity
- std::vector< std::array< double, N > > m_hVelocity
- std::vector< double > m_eMobility
- std::vector< double > m_hMobility

- std::vector< double > m_eAlpha
- std::vector< double > m_hAlpha
- std::vector< double > m_eLifetime
- std::vector< double > m_hLifetime
- std::vector< std::vector< float > > m_donorOcc
- std::vector< std::vector< float > > m_acceptorOcc
- std::vector< double > m_eEta
- std::vector< double > m_hEta
- std::vector< Defect > m_donors
- std::vector< Defect > m_acceptors
- bool m_useVelocityMap = false
- bool m_useTrapOccMap = false
- bool m_useLifetimeMap = false
- bool m_useAlphaMap = false
- std::array< double, 3 > m_bbMin = {{0., 0., 0.}}
- std::array< double, 3 > m_bbMax = {{0., 0., 0.}}
- double m_pMin = 0.
- double m_pMax = 0.

Static Protected Attributes

- static constexpr size_t nMaxVertices = 4

### 7.32.1 Detailed Description

template<size_t N>
class Garfield::ComponentTcadBase< N >

Interpolation in a field map created by Sentaurus Device.
Definition at line 15 of file ComponentTcadBase.hh.

### 7.32.2 Constructor & Destructor Documentation

ComponentTcadBase() [1/2]

template<size_t N>
Garfield::ComponentTcadBase< N >::ComponentTcadBase () [delete]
Default constructor.

ComponentTcadBase() [2/2]

template<size_t N>
Garfield::ComponentTcadBase< N >::ComponentTcadBase (
const std::string & name) [inline]
Constructor.
Definition at line 20 of file ComponentTcadBase.hh.

```
00020                                       : Component(name) {
00021     m_regions.reserve(10);
00022     m_vertices.reserve(10000);
00023     m_elements.reserve(10000);
00024   }
```

~ComponentTcadBase()

template<size_t N>
virtual Garfield::ComponentTcadBase< N >::~ComponentTcadBase () [inline], [virtual]
Destructor.
Definition at line 26 of file ComponentTcadBase.hh.

```
00026 {}
```

### 7.32.3 Member Function Documentation

#### Cleanup()

template<size_t N>
void Garfield::ComponentTcadBase< N >::Cleanup () [protected]

#### ComputeEtaFromLifetime()

template<size_t N>
void Garfield::ComponentTcadBase< N >::ComputeEtaFromLifetime () [protected]

#### ComputeEtaFromTraps()

template<size_t N>
void Garfield::ComponentTcadBase< N >::ComputeEtaFromTraps () [protected]

#### DelayedSignalTimes()

template<size_t N>
const std::vector< double > & Garfield::ComponentTcadBase< N >::DelayedSignalTimes (
const std::string & label) [inline], [override]

Definition at line 135 of file ComponentTcadBase.hh.

```
00136                                    {
00137     if (m_dwtp.count(label) > 0) return m_dwtp[label];
00138     static const std::vector<double> emptyVector;
00139     return emptyVector;
00140   }
```

#### DelayedWeightingField()

template<size_t N>
void Garfield::ComponentTcadBase< N >::DelayedWeightingField (
const double x,
const double y,
const double z,
const double t,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

#### DelayedWeightingPotential()

template<size_t N>
double Garfield::ComponentTcadBase< N >::DelayedWeightingPotential (
const double x,
const double y,
const double z,
const double t,
const std::string & label) [override]

#### ElectronAttachment()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::ElectronAttachment (
const double x,
const double y,
const double z,
double & eta) [override]

### ElectronLifetime()

```
template<size_t N>
double Garfield::ComponentTcadBase< N >::ElectronLifetime (
                const double x,
                const double y,
                const double z)
```

### ElectronMobility()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::ElectronMobility (
                const double x,
                const double y,
                const double z,
                double & mu)   [override]
```

### ElectronTownsend()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::ElectronTownsend (
                const double x,
                const double y,
                const double z,
                double & alpha)   [override]
```

### ElectronVelocity()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::ElectronVelocity (
                const double x,
                const double y,
                const double z,
                double & vx,
                double & vy,
                double & vz)   [override]
```

### ElementVertices()

```
template<size_t N>
unsigned int Garfield::ComponentTcadBase< N >::ElementVertices (
                const Element & element)   [inline], [static], [protected]
```

Definition at line 299 of file ComponentTcadBase.hh.

```
00299                                                 {
00300     return std::min(element.type + 1, 4U);
00301   }
```

### EnableAlphaMap()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::EnableAlphaMap (
                const bool on = true)   [inline]
```

Use the imported impact ionisation map or not.

Definition at line 113 of file ComponentTcadBase.hh.

```
00113 { m_useAlphaMap = on; }
```

### EnableLifetimeMap()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::EnableLifetimeMap (
                const bool on = true)
```

Use the imported lifetime map or not.

### EnableTrapOccupationMap()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::EnableTrapOccupationMap (
            const bool on = true)
```

Use the imported trapping map or not.

### EnableVelocityMap()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::EnableVelocityMap (
            const bool on)
```

Switch use of the imported velocity map on/off.

### FillTree()

```
template<size_t N>
virtual void Garfield::ComponentTcadBase< N >::FillTree ()  [protected], [pure virtual]
```

Implemented in Garfield::ComponentTcad2d, and Garfield::ComponentTcad3d.

### FindRegion()

```
template<size_t N>
size_t Garfield::ComponentTcadBase< N >::FindRegion (
            const std::string & name) const  [protected]
```

### GetElectronMobility()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::GetElectronMobility (
            const double x,
            const double y,
            const double z,
            double & mob)
```

Get the electron mobility at a given point in the mesh.

### GetElementNodes()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::GetElementNodes (
            const size_t i,
            std::vector< size_t > & nodes) const  [override]
```

### GetElementRegion()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::GetElementRegion (
            const size_t i,
            size_t & region,
            bool & active) const  [override]
```

### GetHoleMobility()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::GetHoleMobility (
            const double x,
            const double y,
            const double z,
            double & mob)
```

Get the hole mobility at a given point in the mesh.

### GetNumberOfAcceptors()

template<size_t N>
size_t Garfield::ComponentTcadBase< N >::GetNumberOfAcceptors () [inline]
Get the number of acceptor states found in the map.
Definition at line 98 of file ComponentTcadBase.hh.

```
00098 { return m_acceptors.size(); }
```

### GetNumberOfDonors()

template<size_t N>
size_t Garfield::ComponentTcadBase< N >::GetNumberOfDonors () [inline]
Get the number of donor states found in the map.
Definition at line 96 of file ComponentTcadBase.hh.

```
00096 { return m_donors.size(); }
```

### GetNumberOfElements()

template<size_t N>
size_t Garfield::ComponentTcadBase< N >::GetNumberOfElements () const [inline], [override]
Definition at line 85 of file ComponentTcadBase.hh.

```
00085 { return m_elements.size(); }
```

### GetNumberOfNodes()

template<size_t N>
size_t Garfield::ComponentTcadBase< N >::GetNumberOfNodes () const [inline], [override]
Definition at line 90 of file ComponentTcadBase.hh.

```
00090 { return m_vertices.size(); }
```

### GetNumberOfRegions()

template<size_t N>
size_t Garfield::ComponentTcadBase< N >::GetNumberOfRegions () const [inline]
Get the number of regions in the device.
Definition at line 73 of file ComponentTcadBase.hh.

```
00073 { return m_regions.size(); }
```

### GetOffset()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::GetOffset (
const std::string & label,
double & dx,
double & dy,
double & dz) const [protected]

### GetRegion()

template<size_t N>
void Garfield::ComponentTcadBase< N >::GetRegion (
const size_t ireg,
std::string & name,
bool & active) const
Get the name and "active volume" flag of a region.

### GetVoltageRange()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::GetVoltageRange (

double & vmin,
double & vmax) [override]

### HasAttachmentMap()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HasAttachmentMap () const [inline], [override]

Definition at line 161 of file ComponentTcadBase.hh.

```
00161                                    {
00162     return ((m_useTrapOccMap || m_useLifetimeMap) &&
00163             !(m_eEta.empty() && m_hEta.empty()));
00164   }
```

### HasMobilityMap()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HasMobilityMap () const [inline], [override]

Definition at line 165 of file ComponentTcadBase.hh.

```
00165                                    {
00166     return !(m_eMobility.empty() && m_hMobility.empty());
00167   }
```

### HasTownsendMap()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HasTownsendMap () const [inline], [override]

Definition at line 158 of file ComponentTcadBase.hh.

```
00158                                    {
00159     return m_useAlphaMap && !(m_eAlpha.empty() && m_hAlpha.empty());
00160   }
```

### HasVelocityMap()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HasVelocityMap () const [inline], [override]

Definition at line 150 of file ComponentTcadBase.hh.

```
00150                                    {
00151     return m_useVelocityMap && !(m_eVelocity.empty() && m_hVelocity.empty());
00152   }
```

### HoleAttachment()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HoleAttachment (
const double x,
const double y,
const double z,
double & eta) [override]

### HoleLifetime()

template<size_t N>
double Garfield::ComponentTcadBase< N >::HoleLifetime (
const double x,
const double y,
const double z)

### HoleMobility()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HoleMobility (
const double x,
const double y,

const double z,
double & mu) [override]

### HoleTownsend()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HoleTownsend (
const double x,
const double y,
const double z,
double & alpha) [override]

### HoleVelocity()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::HoleVelocity (
const double x,
const double y,
const double z,
double & vx,
double & vy,
double & vz) [override]

### InBoundingBox()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::InBoundingBox (
const std::array< double, N > & x) const [inline], [protected]

Definition at line 312 of file ComponentTcadBase.hh.

```
00312                                                 {
00313     for (size_t i = 0; i < N; ++i) {
00314       if (x[i] < m_bbMin[i] || x[i] > m_bbMax[i]) return false;
00315     }
00316     return true;
00317   }
```

### Initialise()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::Initialise (
const std::string & gridfilename,
const std::string & datafilename)

Import mesh and field map from files.

Parameters

| | |
|---|---|
| gridfilename | name of the .grd file containing the mesh |
| datafilename | name of the .dat file containing the nodal solution |

### Interpolate() [1/2]

template<size_t N>
virtual bool Garfield::ComponentTcadBase< N >::Interpolate (
const double x,
const double y,
const double z,
const std::vector< double > & field,
double & f) [protected], [pure virtual]

Implemented in Garfield::ComponentTcad2d, and Garfield::ComponentTcad3d.

### Interpolate() [2/2]

```
template<size_t N>
virtual bool Garfield::ComponentTcadBase< N >::Interpolate (
                const double x,
                const double y,
                const double z,
                const std::vector< std::array< double, N > > & field,
                double & fx,
                double & fy,
                double & fz)    [protected], [pure virtual]
```

### LoadData()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::LoadData (
                const std::string & datafilename)    [protected]
```

### LoadGrid()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::LoadGrid (
                const std::string & gridfilename)    [protected]
```

### LoadWeightingField()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::LoadWeightingField (
                const std::string & datafilename,
                std::vector< std::array< double, N > > & wf,
                std::vector< double > & wp)    [protected]
```

### MapCoordinates()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::MapCoordinates (
                std::array< double, N > & x,
                std::array< bool, N > & mirr) const    [protected]
```

### PrintRegions()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::PrintRegions () const
```

List all currently defined regions.

### ReadDataset()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::ReadDataset (
                std::ifstream & datafile,
                const std::string & dataset)    [protected]
```

### SetAcceptor()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetAcceptor (
                const size_t acceptorNumber,
                const double exsec,
                const double hxsec,
                const double concentration)
```

Set the properties of an acceptor-type defect state.

SetDonor()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetDonor (
                const size_t donorNumber,
                const double exsec,
                const double hxsec,
                const double concentration)
```

Set the properties of a donor-type defect state.

Parameters

| | |
|---|---|
| donorNumber | index of the donor |
| exsec | cross-section [cm2] for electrons |
| hxsec | cross-section [cm2] for holes |
| concentration | defect density [cm-3] |

SetDriftRegion()

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::SetDriftRegion (
                const size_t ireg)
```

Make a region active ("driftable").

SetDynamicWeightingField()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetDynamicWeightingField (
                const std::string & datfile1,
                const std::string & datfile2,
                const double dv,
                const double t,
                const std::string & label)
```

Import time-dependent weighting fields at t >= 0.

SetDynamicWeightingPotential()

```
template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetDynamicWeightingPotential (
                const std::string & datfile1,
                const std::string & datfile2,
                const double dv,
                const double t,
                const std::string & label)
```

Import time-dependent weighting potentials at t >= 0.

SetMedium() [1/2]

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::SetMedium (
                const size_t ireg,
                Medium * m)
```

Set the medium to be associated to a given region.

SetMedium() [2/2]

```
template<size_t N>
void Garfield::ComponentTcadBase< N >::SetMedium (
```

const std::string & material,
Medium * m)

Set the medium to be associated to all regions with a given material.

### SetWeightingField()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetWeightingField (
const std::string & datfile1,
const std::string & datfile2,
const double dv,
const std::string & label)

Import field maps defining the prompt weighting field and potential.

Parameters

| | |
|---|---|
| datfile1 | .dat file containing the field map at nominal bias. |
| datfile2 | .dat file containing the field map for a configuration with the potential at the electrode to be read out increased by a small voltage dv. |
| dv | increase in electrode potential between the two field maps. |
| label | name of the electrode |

The field maps must use the same mesh as the drift field.

### SetWeightingFieldShift()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetWeightingFieldShift (
const std::string & label,
const double x,
const double y,
const double z)

Shift the maps of weighting field/potential for a given electrode with respect to the original mesh.
If the electrode does not exist yet, a new one will be added to the list.

### SetWeightingPotential()

template<size_t N>
bool Garfield::ComponentTcadBase< N >::SetWeightingPotential (
const std::string & datfile1,
const std::string & datfile2,
const double dv,
const std::string & label) [inline]

Import field maps defining the prompt weighting field and potential.
Definition at line 50 of file ComponentTcadBase.hh.

```
00052                                                 {
00053     return SetWeightingField(datfile1, datfile2, dv, label);
00054   }
```

### UnsetDriftRegion()

template<size_t N>
void Garfield::ComponentTcadBase< N >::UnsetDriftRegion (
const size_t ireg)

Make a region inactive.

### UpdateAttachment()

template<size_t N>
void Garfield::ComponentTcadBase< N >::UpdateAttachment () [protected]

### UpdatePeriodicity()

template<size_t N>
void Garfield::ComponentTcadBase< N >::UpdatePeriodicity () [override], [protected]

### WeightingField()

template<size_t N>
void Garfield::ComponentTcadBase< N >::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

### WeightingPotential()

template<size_t N>
double Garfield::ComponentTcadBase< N >::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

## 7.32.4 Member Data Documentation

### m_acceptorOcc

template<size_t N>
std::vector<std::vector<float> > Garfield::ComponentTcadBase< N >::m_acceptorOcc [protected]
Definition at line 262 of file ComponentTcadBase.hh.

### m_acceptors

template<size_t N>
std::vector<Defect> Garfield::ComponentTcadBase< N >::m_acceptors [protected]
Definition at line 276 of file ComponentTcadBase.hh.

### m_bbMax

template<size_t N>
std::array<double, 3> Garfield::ComponentTcadBase< N >::m_bbMax = {{0., 0., 0.}} [protected]
Definition at line 289 of file ComponentTcadBase.hh.
00289 {{0., 0., 0.}};

### m_bbMin

template<size_t N>
std::array<double, 3> Garfield::ComponentTcadBase< N >::m_bbMin = {{0., 0., 0.}} [protected]
Definition at line 288 of file ComponentTcadBase.hh.
00288 {{0., 0., 0.}};

### m_donorOcc

template<size_t N>
std::vector<std::vector<float> > Garfield::ComponentTcadBase< N >::m_donorOcc [protected]
Definition at line 261 of file ComponentTcadBase.hh.

### m_donors

template<size_t N>
std::vector<Defect> Garfield::ComponentTcadBase< N >::m_donors [protected]
Definition at line 275 of file ComponentTcadBase.hh.

### m_dwf

template<size_t N>
std::map<std::string, std::vector<std::vector<std::array<double, N> > > > Garfield::ComponentTcadBase< N >::m_dwf [protected]
Definition at line 242 of file ComponentTcadBase.hh.

### m_dwp

template<size_t N>
std::map<std::string, std::vector<std::vector<double> > > Garfield::ComponentTcadBase< N >::m_dwp [protected]
Definition at line 243 of file ComponentTcadBase.hh.

### m_dwtf

template<size_t N>
std::map<std::string, std::vector<double> > Garfield::ComponentTcadBase< N >::m_dwtf [protected]
Definition at line 245 of file ComponentTcadBase.hh.

### m_dwtp

template<size_t N>
std::map<std::string, std::vector<double> > Garfield::ComponentTcadBase< N >::m_dwtp [protected]
Definition at line 246 of file ComponentTcadBase.hh.

### m_eAlpha

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_eAlpha [protected]
Definition at line 255 of file ComponentTcadBase.hh.

### m_eEta

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_eEta [protected]
Definition at line 264 of file ComponentTcadBase.hh.

### m_efield

template<size_t N>
std::vector<std::array<double, N> > Garfield::ComponentTcadBase< N >::m_efield [protected]
Definition at line 232 of file ComponentTcadBase.hh.

### m_elements

template<size_t N>
std::vector<Element> Garfield::ComponentTcadBase< N >::m_elements [protected]
Definition at line 227 of file ComponentTcadBase.hh.

### m_eLifetime

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_eLifetime [protected]
Definition at line 258 of file ComponentTcadBase.hh.

### m_eMobility

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_eMobility [protected]
Definition at line 252 of file ComponentTcadBase.hh.

### m_epot

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_epot [protected]
Definition at line 230 of file ComponentTcadBase.hh.

### m_eVelocity

template<size_t N>
std::vector<std::array<double, N> > Garfield::ComponentTcadBase< N >::m_eVelocity [protected]
Definition at line 249 of file ComponentTcadBase.hh.

### m_hAlpha

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_hAlpha [protected]
Definition at line 256 of file ComponentTcadBase.hh.

### m_hEta

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_hEta [protected]
Definition at line 265 of file ComponentTcadBase.hh.

### m_hLifetime

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_hLifetime [protected]
Definition at line 259 of file ComponentTcadBase.hh.

### m_hMobility

template<size_t N>
std::vector<double> Garfield::ComponentTcadBase< N >::m_hMobility [protected]
Definition at line 253 of file ComponentTcadBase.hh.

### m_hVelocity

template<size_t N>
std::vector<std::array<double, N> > Garfield::ComponentTcadBase< N >::m_hVelocity [protected]
Definition at line 250 of file ComponentTcadBase.hh.

### m_pMax

template<size_t N>
double Garfield::ComponentTcadBase< N >::m_pMax = 0. [protected]
Definition at line 293 of file ComponentTcadBase.hh.

### m_pMin

template<size_t N>
double Garfield::ComponentTcadBase< N >::m_pMin = 0. [protected]
Definition at line 292 of file ComponentTcadBase.hh.

### m_regions

template<size_t N>
std::vector<Region> Garfield::ComponentTcadBase< N >::m_regions [protected]
Definition at line 197 of file ComponentTcadBase.hh.

### m_useAlphaMap

template<size_t N>
bool Garfield::ComponentTcadBase< N >::m_useAlphaMap = false [protected]
Definition at line 285 of file ComponentTcadBase.hh.

### m_useLifetimeMap

template<size_t N>
bool Garfield::ComponentTcadBase< N >::m_useLifetimeMap = false [protected]
Definition at line 283 of file ComponentTcadBase.hh.

### m_useTrapOccMap

template<size_t N>
bool Garfield::ComponentTcadBase< N >::m_useTrapOccMap = false [protected]
Definition at line 281 of file ComponentTcadBase.hh.

### m_useVelocityMap

template<size_t N>
bool Garfield::ComponentTcadBase< N >::m_useVelocityMap = false [protected]
Definition at line 279 of file ComponentTcadBase.hh.

### m_vertices

template<size_t N>
std::vector<std::array<double, N> > Garfield::ComponentTcadBase< N >::m_vertices [protected]
Definition at line 200 of file ComponentTcadBase.hh.

### m_wfield

template<size_t N>
std::map<std::string, std::vector<std::array<double, N> > > Garfield::ComponentTcadBase< N >::m_wfield [protected]
Definition at line 235 of file ComponentTcadBase.hh.

### m_wpot

template<size_t N>
std::map<std::string, std::vector<double> > Garfield::ComponentTcadBase< N >::m_wpot [protected]
Definition at line 236 of file ComponentTcadBase.hh.

### m_wshift

template<size_t N>
std::map<std::string, std::vector<double> > Garfield::ComponentTcadBase< N >::m_wshift [protected]
Definition at line 238 of file ComponentTcadBase.hh.

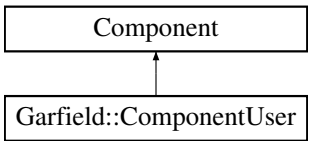
### nMaxVertices

template<size_t N>
size_t Garfield::ComponentTcadBase< N >::nMaxVertices = 4 [static], [constexpr], [protected]
Definition at line 184 of file ComponentTcadBase.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh

## 7.33 Garfield::ComponentUser Class Reference

Simple component with electric field given by a user function.
#include <ComponentUser.hh>
Inheritance diagram for Garfield::ComponentUser:




Public Member Functions

- ComponentUser ()

  Constructor.
- ~ComponentUser ()

  Destructor.
- void SetElectricField (std::function< void(const double, const double, const double, double &, double &, double &)>)

  Set the function to be called for calculating the electric field.
- void SetElectricField (const std::string &expression)

  Set the expression to be used for calculating the electric field.
- void SetPotential (std::function< double(const double, const double, const double)>)

  Set the function to be called for calculating the electric potential.
- void SetPotential (const std::string &expression)

  Set the expression to be used for calculating the electric potential.
- void SetWeightingField (std::function< void(const double, const double, const double, double &, double &, double &)>, const std::string &label)

  Set the function to be called for calculating the weighting field.
- void SetWeightingField (const std::string &expression, const std::string &label)

  Set the expression to be used for calculating the weighting field.
- void SetWeightingPotential (std::function< double(const double, const double, const double)>, const std::string &label)

  Set the function to be called for calculating the weighting potential.
- void SetWeightingPotential (const std::string &expression, const std::string &label)

  Set the expression to be used for calculating the weighting potential.
- void SetDelayedWeightingField (std::function< void(const double, const double, const double, const double, double &, double &, double &)>, const std::string &label)

  Set the function to be called for calculating the delayed weighting field.
- void SetDelayedWeightingField (const std::string &expression, const std::string &label)

  Set the expression to be used for calculating the delayed weighting field.
- void SetDelayedWeightingPotential (std::function< double(const double, const double, const double, const double)>, const std::string &label)

  Set the function to be called for calculating the delayed weighting potential.
- void SetDelayedWeightingPotential (const std::string &expression, const std::string &label)

  Set the expression to be used for calculating the delayed weighting potential.
- void SetDelayedSignalTimes (const std::vector< double > &ts)

  Set the time steps at which to evaluate the delayed weighting potential/field.
- void SetMagneticField (std::function< void(const double, const double, const double, double &, double &, double &)>)

  Set the function to be called for calculating the magnetic field.
- void SetMagneticField (const std::string &expression)

Set the expression to be used for calculating the magnetic field.

- void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set the limits of the active area explicitly (instead of using a Geometry object).
- void UnsetArea ()

  Remove the explicit limits of the active area.
- void SetMedium (Medium *medium)

  Set the medium in the active area.
- Medium * GetMedium (const double x, const double y, const double z) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- void MagneticField (const double x, const double y, const double z, double &bx, double &by, double &bz, int &status) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- void DelayedWeightingField (const double x, const double y, const double z, const double t, double &wx, double &wy, double &wz, const std::string &label) override
- double DelayedWeightingPotential (const double x, const double y, const double z, const double t, const std::string &label) override
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool HasMagneticField () const override

Private Member Functions

- void Reset () override

  Reset the component.
- void UpdatePeriodicity () override
- bool InArea (const double x, const double y, const double z)

Private Attributes

- std::function< void(const double, const double, const double, double &, double &, double &)> m_efield

  Electric field function.
- std::function< double(const double, const double, const double)> m_epot

  Electric potential function.
- std::map< std::string, std::function< void(const double, const double, const double, double &, double &, double &)> > m_wfield

  Weighting field functions.
- std::map< std::string, std::function< double(const double, const double, const double)> > m_wpot

  Weighting potential functions.
- std::map< std::string, std::function< void(const double, const double, const double, const double, double &, double &, double &)> > m_dwfield

  Delayed weighting field functions.
- std::map< std::string, std::function< double(const double, const double, const double, const double)> > m_dwpot

  Delayed weighting potential functions.
- std::function< void(const double, const double, const double, double &, double &, double &)> m_bfield

Magnetic field function.

- std::array< double, 3 > m_xmin = {{0., 0., 0.}}
- std::array< double, 3 > m_xmax = {{0., 0., 0.}}
- bool m_hasArea = false
- Medium * m_medium = nullptr

### 7.33.1 Detailed Description

Simple component with electric field given by a user function.
Definition at line 14 of file ComponentUser.hh.

### 7.33.2 Constructor & Destructor Documentation

#### ComponentUser()

Garfield::ComponentUser::ComponentUser ()
Constructor.

#### ~ComponentUser()

Garfield::ComponentUser::~ComponentUser () [inline]
Destructor.
Definition at line 19 of file ComponentUser.hh.
00019 {}

### 7.33.3 Member Function Documentation

#### DelayedWeightingField()

void Garfield::ComponentUser::DelayedWeightingField (
const double x,
const double y,
const double z,
const double t,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

#### DelayedWeightingPotential()

double Garfield::ComponentUser::DelayedWeightingPotential (
const double x,
const double y,
const double z,
const double t,
const std::string & label) [override]

#### ElectricField() [1/2]

void Garfield::ComponentUser::ElectricField (
const double x,
const double y,
const double z,
double & ex,
double & ey,
double & ez,
double & v,
Medium *& m,
int & status) [override]

### ElectricField() [2/2]

```
void Garfield::ComponentUser::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)    [override]
```

### GetBoundingBox()

```
bool Garfield::ComponentUser::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

### GetMedium()

```
Medium * Garfield::ComponentUser::GetMedium (
                const double x,
                const double y,
                const double z)    [inline], [override]
```

Definition at line 89 of file ComponentUser.hh.

```
00089                                                                 {
00090     return !m_hasArea       ? Component::GetMedium(x, y, z)
00091          : InArea(x, y, z) ? m_medium
00092                            : nullptr;
00093   }
```

### GetVoltageRange()

```
bool Garfield::ComponentUser::GetVoltageRange (
                double & vmin,
                double & vmax)    [override]
```

### HasMagneticField()

```
bool Garfield::ComponentUser::HasMagneticField () const    [override]
```

### InArea()

```
bool Garfield::ComponentUser::InArea (
                const double x,
                const double y,
                const double z)    [inline], [private]
```

Definition at line 167 of file ComponentUser.hh.

```
00167                                                                 {
00168     return (x >= m_xmin[0] && x <= m_xmax[0] && y >= m_xmin[1] &&
00169             y <= m_xmax[1] && z >= m_xmin[2] && z <= m_xmax[2]);
00170   }
```

### MagneticField()

```
void Garfield::ComponentUser::MagneticField (
                const double x,
                const double y,
                const double z,
```

double & bx,
double & by,
double & bz,
int & status) [override]

### Reset()

void Garfield::ComponentUser::Reset () [override], [private]
Reset the component.

### SetArea()

void Garfield::ComponentUser::SetArea (
const double xmin,
const double ymin,
const double zmin,
const double xmax,
const double ymax,
const double zmax)
Set the limits of the active area explicitly (instead of using a Geometry object).

### SetDelayedSignalTimes()

void Garfield::ComponentUser::SetDelayedSignalTimes (
const std::vector< double > & ts)
Set the time steps at which to evaluate the delayed weighting potential/field.

### SetDelayedWeightingField() [1/2]

void Garfield::ComponentUser::SetDelayedWeightingField (
const std::string & expression,
const std::string & label)
Set the expression to be used for calculating the delayed weighting field.

### SetDelayedWeightingField() [2/2]

void Garfield::ComponentUser::SetDelayedWeightingField (
std::function< void(const double, const double, const double, const double, double &, double &, double &)>
,
const std::string & label)
Set the function to be called for calculating the delayed weighting field.

### SetDelayedWeightingPotential() [1/2]

void Garfield::ComponentUser::SetDelayedWeightingPotential (
const std::string & expression,
const std::string & label)
Set the expression to be used for calculating the delayed weighting potential.

### SetDelayedWeightingPotential() [2/2]

void Garfield::ComponentUser::SetDelayedWeightingPotential (
std::function< double(const double, const double, const double, const double)> ,
const std::string & label)
Set the function to be called for calculating the delayed weighting potential.

### SetElectricField() [1/2]

void Garfield::ComponentUser::SetElectricField (
const std::string & expression)
Set the expression to be used for calculating the electric field.

SetElectricField() [2/2]

void Garfield::ComponentUser::SetElectricField (
std::function< void(const double, const double, const double, double &, double &, double &)> )
Set the function to be called for calculating the electric field.

SetMagneticField() [1/2]

void Garfield::ComponentUser::SetMagneticField (
const std::string & expression)
Set the expression to be used for calculating the magnetic field.

SetMagneticField() [2/2]

void Garfield::ComponentUser::SetMagneticField (
std::function< void(const double, const double, const double, double &, double &, double &)> )
Set the function to be called for calculating the magnetic field.

SetMedium()

void Garfield::ComponentUser::SetMedium (
Medium * medium) [inline]
Set the medium in the active area.
Definition at line 87 of file ComponentUser.hh.
00087 { m_medium = medium; }

SetPotential() [1/2]

void Garfield::ComponentUser::SetPotential (
const std::string & expression)
Set the expression to be used for calculating the electric potential.

SetPotential() [2/2]

void Garfield::ComponentUser::SetPotential (
std::function< double(const double, const double, const double)> )
Set the function to be called for calculating the electric potential.

SetWeightingField() [1/2]

void Garfield::ComponentUser::SetWeightingField (
const std::string & expression,
const std::string & label)
Set the expression to be used for calculating the weighting field.

SetWeightingField() [2/2]

void Garfield::ComponentUser::SetWeightingField (
std::function< void(const double, const double, const double, double &, double &, double &)> ,
const std::string & label)
Set the function to be called for calculating the weighting field.

SetWeightingPotential() [1/2]

void Garfield::ComponentUser::SetWeightingPotential (
const std::string & expression,
const std::string & label)
Set the expression to be used for calculating the weighting potential.

SetWeightingPotential() [2/2]

void Garfield::ComponentUser::SetWeightingPotential (
std::function< double(const double, const double, const double)> ,
const std::string & label)
Set the function to be called for calculating the weighting potential.

UnsetArea()

void Garfield::ComponentUser::UnsetArea ()
Remove the explicit limits of the active area.

UpdatePeriodicity()

void Garfield::ComponentUser::UpdatePeriodicity () [override], [private]

WeightingField()

void Garfield::ComponentUser::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

WeightingPotential()

double Garfield::ComponentUser::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

### 7.33.4 Member Data Documentation

m_bfield

std::function<void(const double, const double, const double, double&, double&, double&)> Garfield::ComponentUser::m_↩
bfield [private]
Magnetic field function.
Definition at line 152 of file ComponentUser.hh.

m_dwfield

std::map<std::string, std::function<void(const double, const double, const double, const double, double&, double&, double&)>
> Garfield::ComponentUser::m_dwfield [private]
Delayed weighting field functions.
Definition at line 142 of file ComponentUser.hh.

m_dwpot

std::map<std::string, std::function<double(const double, const double, const double, const double)> > Garfield::Component↩
User::m_dwpot [private]
Delayed weighting potential functions.
Definition at line 147 of file ComponentUser.hh.

m_efield

std::function<void(const double, const double, const double, double&, double&, double&)> Garfield::ComponentUser::m_↩
efield [private]

Electric field function.
Definition at line 123 of file ComponentUser.hh.

### m_epot

std::function<double(const double, const double, const double)> Garfield::ComponentUser::m_epot [private]
Electric potential function.
Definition at line 125 of file ComponentUser.hh.

### m_hasArea

bool Garfield::ComponentUser::m_hasArea = false [private]
Definition at line 158 of file ComponentUser.hh.

### m_medium

Medium* Garfield::ComponentUser::m_medium = nullptr [private]
Definition at line 160 of file ComponentUser.hh.

### m_wfield

std::map<std::string, std::function<void(const double, const double, const double, double&, double&, double&)> > Garfield::ComponentUser::m_wfield [private]
Weighting field functions.
Definition at line 131 of file ComponentUser.hh.

### m_wpot

std::map<std::string, std::function<double(const double, const double, const double)> > Garfield::ComponentUser::m_wpot [private]
Weighting potential functions.
Definition at line 136 of file ComponentUser.hh.

### m_xmax

std::array<double, 3> Garfield::ComponentUser::m_xmax = {{0., 0., 0.}} [private]
Definition at line 156 of file ComponentUser.hh.
00156 {{0., 0., 0.}};

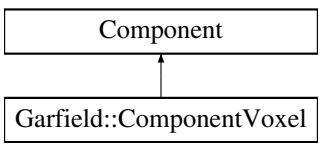
### m_xmin

std::array<double, 3> Garfield::ComponentUser::m_xmin = {{0., 0., 0.}} [private]
Definition at line 155 of file ComponentUser.hh.
00155 {{0., 0., 0.}};
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentUser.hh

## 7.34 Garfield::ComponentVoxel Class Reference

Component for interpolating field maps stored in a regular mesh.
#include <ComponentVoxel.hh>
Inheritance diagram for Garfield::ComponentVoxel:



Classes

- struct Element

Public Member Functions

- ComponentVoxel ()

  Constructor.
- ~ComponentVoxel ()

  Destructor.
- void EnableInterpolation (const bool on=true)

  Interpolate between field values at the element centres.
- void SetMesh (const unsigned int nx, const unsigned int ny, const unsigned int nz, const double xmin, const double xmax, const double ymin, const double ymax, const double zmin, const double zmax)

  Define the grid.
- bool LoadElectricField (const std::string &filename, const std::string &format, const bool withPotential, const bool withRegion, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import electric field and potential values from a file.
- bool LoadWeightingField (const std::string &filename, const std::string &format, const bool withPotential, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import (prompt) weighting field from file.
- void SetWeightingFieldOffset (const double x, const double y, const double z)

  Offset coordinates in the weighting field, such that the same numerical weighting field map can be used for electrodes at different positions.
- bool LoadWeightingField (const std::string &filename, const std::string &format, const double time, const bool withPotential, const double scaleX=1., const double scaleE=1., const double scaleP=1.)

  Import delayed weighting field from file.
- bool LoadMagneticField (const std::string &filename, const std::string &format, const double scaleX=1., const double scaleB=1.)

  Import magnetic field values from a file.
- void SetMedium (const unsigned int i, Medium *m)

  Set the medium in region i.
- Medium * GetMedium (const unsigned int i) const

  Get the medium in region i.
- void PrintRegions () const

  Print all regions.
- bool GetElement (const double xi, const double yi, const double zi, unsigned int &i, unsigned int &j, unsigned int &k, bool &xMirrored, bool &yMirrored, bool &zMirrored) const

  Return the indices of the element at a given point.
- bool GetElement (const unsigned int i, const unsigned int j, const unsigned int k, double &v, double &ex, double &ey, double &ez) const

  Return the field for an element with given index.
- void Clear () override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, double &v, Medium *&m, int &status) override
- void ElectricField (const double x, const double y, const double z, double &ex, double &ey, double &ez, Medium *&m, int &status) override
- void WeightingField (const double x, const double y, const double z, double &wx, double &wy, double &wz, const std::string &label) override
- double WeightingPotential (const double x, const double y, const double z, const std::string &label) override
- void DelayedWeightingField (const double x, const double y, const double z, const double t, double &wx, double &wy, double &wz, const std::string &label) override

- double DelayedWeightingPotential (const double x, const double y, const double z, const double t, const std::string &label) override
- void MagneticField (const double x, const double y, const double z, double &bx, double &by, double &bz, int &status) override
- bool HasMagneticField () const override
- Medium * GetMedium (const double x, const double y, const double z) override
- bool GetVoltageRange (double &vmin, double &vmax) override
- bool GetElectricFieldRange (double &exmin, double &exmax, double &eymin, double &eymax, double &ezmin, double &ezmax)
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
- bool GetElementaryCell (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override

Private Member Functions

- bool LoadData (const std::string &filename, std::string format, const bool withPotential, const bool withRegion, const double scaleX, const double scaleF, const double scaleP, std::vector< std::vector< std::vector< Element > > > &field)

  Read data from file.
- void Reset () override
- void UpdatePeriodicity () override
- bool GetField (const double x, const double y, const double z, const std::vector< std::vector< std::vector< Element > > > &field, double &fx, double &fy, double &fz, double &p, int &region)

  Look up/interpolate the field at a given point.
- double Reduce (const double xin, const double xmin, const double xmax, const bool simplePeriodic, const bool mirrorPeriodic, bool &isMirrored) const

  Reduce a coordinate to the basic cell (in case of periodicity).
- void Initialise (std::vector< std::vector< std::vector< Element > > > &fields)

  Set the dimensions of a table according to the mesh.
- void InitialiseRegions ()

Private Attributes

- std::vector< Medium * > m_media
- std::vector< std::vector< std::vector< int > > > m_regions

  Region indices.
- std::vector< std::vector< std::vector< Element > > > m_efields

  Electric field values and potentials at each mesh element.
- std::vector< std::vector< std::vector< Element > > > m_bfields

  Magnetic field values at each mesh element.
- std::vector< std::vector< std::vector< Element > > > m_wfields

  Prompt weighting field values and potentials at each mesh element.
- std::vector< std::vector< std::vector< std::vector< Element > > > > m_wdfields

  Delayed weighting field values and potentials at each mesh element.
- unsigned int m_nX = 0
- unsigned int m_nY = 0
- unsigned int m_nZ = 0
- double m_xMin = 0.
- double m_yMin = 0.
- double m_zMin = 0.
- double m_xMax = 0.
- double m_yMax = 0.
- double m_zMax = 0.
- double m_dx = 0.

- double m_dy = 0.
- double m_dz = 0.
- bool m_interpolate = false
- bool m_hasMesh = false
- bool m_hasPotential = false
- bool m_hasEfield = false
- bool m_hasBfield = false
- bool m_hasWfield = false
- double m_wField_xOffset = 0.
- double m_wField_yOffset = 0.
- double m_wField_zOffset = 0.
- double m_pMin = 0.
- double m_pMax = 0.

### 7.34.1 Detailed Description

Component for interpolating field maps stored in a regular mesh.
Definition at line 10 of file ComponentVoxel.hh.

### 7.34.2 Constructor & Destructor Documentation

#### ComponentVoxel()

Garfield::ComponentVoxel::ComponentVoxel ()
Constructor.

#### ~ComponentVoxel()

Garfield::ComponentVoxel::~ComponentVoxel () [inline]
Destructor.
Definition at line 15 of file ComponentVoxel.hh.

```
00015 {}
```

### 7.34.3 Member Function Documentation

#### Clear()

void Garfield::ComponentVoxel::Clear () [inline], [override]
Definition at line 85 of file ComponentVoxel.hh.

```
00085 { Reset(); }
```

#### DelayedWeightingField()

```
void Garfield::ComponentVoxel::DelayedWeightingField (
                const double x,
                const double y,
                const double z,
                const double t,
                double & wx,
                double & wy,
                double & wz,
                const std::string & label)   [override]
```

#### DelayedWeightingPotential()

```
double Garfield::ComponentVoxel::DelayedWeightingPotential (
                const double x,
                const double y,
                const double z,
                const double t,
                const std::string & label)   [override]
```

### ElectricField() [1/2]

```
void Garfield::ComponentVoxel::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                double & v,
                Medium *& m,
                int & status)    [override]
```

### ElectricField() [2/2]

```
void Garfield::ComponentVoxel::ElectricField (
                const double x,
                const double y,
                const double z,
                double & ex,
                double & ey,
                double & ez,
                Medium *& m,
                int & status)    [override]
```

### EnableInterpolation()

```
void Garfield::ComponentVoxel::EnableInterpolation (
                const bool on = true)    [inline]
```

Interpolate between field values at the element centres.
Definition at line 18 of file ComponentVoxel.hh.

```
00018 { m_interpolate = on; }
```

### GetBoundingBox()

```
bool Garfield::ComponentVoxel::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax)    [override]
```

### GetElectricFieldRange()

```
bool Garfield::ComponentVoxel::GetElectricFieldRange (
                double & exmin,
                double & exmax,
                double & eymin,
                double & eymax,
                double & ezmin,
                double & ezmax)
```

### GetElement() [1/2]

```
bool Garfield::ComponentVoxel::GetElement (
                const double xi,
                const double yi,
                const double zi,
                unsigned int & i,
```

unsigned int & j,
unsigned int & k,
bool & xMirrored,
bool & yMirrored,
bool & zMirrored) const

Return the indices of the element at a given point.

### GetElement() [2/2]

bool Garfield::ComponentVoxel::GetElement (
const unsigned int i,
const unsigned int j,
const unsigned int k,
double & v,
double & ex,
double & ey,
double & ez) const

Return the field for an element with given index.

### GetElementaryCell()

bool Garfield::ComponentVoxel::GetElementaryCell (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [override]

### GetField()

bool Garfield::ComponentVoxel::GetField (
const double x,
const double y,
const double z,
const std::vector< std::vector< std::vector< Element > > > & field,
double & fx,
double & fy,
double & fz,
double & p,
int & region) [private]

Look up/interpolate the field at a given point.

### GetMedium() [1/2]

Medium * Garfield::ComponentVoxel::GetMedium (
const double x,
const double y,
const double z) [override]

### GetMedium() [2/2]

Medium * Garfield::ComponentVoxel::GetMedium (
const unsigned int i) const

Get the medium in region i.

### GetVoltageRange()

bool Garfield::ComponentVoxel::GetVoltageRange (
double & vmin,

double & vmax) [override]

### HasMagneticField()

bool Garfield::ComponentVoxel::HasMagneticField () const [override]

### Initialise()

void Garfield::ComponentVoxel::Initialise (
std::vector< std::vector< std::vector< Element > > > & fields) [private]

Set the dimensions of a table according to the mesh.

### InitialiseRegions()

void Garfield::ComponentVoxel::InitialiseRegions () [private]

### LoadData()

bool Garfield::ComponentVoxel::LoadData (
const std::string & filename,
std::string format,
const bool withPotential,
const bool withRegion,
const double scaleX,
const double scaleF,
const double scaleP,
std::vector< std::vector< std::vector< Element > > > & field) [private]

Read data from file.

### LoadElectricField()

bool Garfield::ComponentVoxel::LoadElectricField (
const std::string & filename,
const std::string & format,
const bool withPotential,
const bool withRegion,
const double scaleX = 1.,
const double scaleE = 1.,
const double scaleP = 1.)

Import electric field and potential values from a file.
The file is supposed to contain one line for each mesh point starting with

- either two or three floating point numbers, specifying the coordinates (in cm) of the element centre or
- two or three integers specifying the index of the element in the mesh,

followed by

- two or three floating point numbers for the electric field (in V/cm), and (depending on the values of withPotential and withRegion),
- a floating point number specifying the potential (in V), and
- an integer specifying the ”region” of the element.

Format types are:

- ”xy”, ”xyz”: elements are specified by the coordinates of their centres
- ”ij”, ”ijk”: elements are specified by their indices

LoadMagneticField()

bool Garfield::ComponentVoxel::LoadMagneticField (
const std::string & filename,
const std::string & format,
const double scaleX = 1.,
const double scaleB = 1.)

Import magnetic field values from a file.

LoadWeightingField() [1/2]

bool Garfield::ComponentVoxel::LoadWeightingField (
const std::string & filename,
const std::string & format,
const bool withPotential,
const double scaleX = 1.,
const double scaleE = 1.,
const double scaleP = 1.)

Import (prompt) weighting field from file.

LoadWeightingField() [2/2]

bool Garfield::ComponentVoxel::LoadWeightingField (
const std::string & filename,
const std::string & format,
const double time,
const bool withPotential,
const double scaleX = 1.,
const double scaleE = 1.,
const double scaleP = 1.)

Import delayed weighting field from file.

MagneticField()

void Garfield::ComponentVoxel::MagneticField (
const double x,
const double y,
const double z,
double & bx,
double & by,
double & bz,
int & status) [override]

PrintRegions()

void Garfield::ComponentVoxel::PrintRegions () const

Print all regions.

Reduce()

double Garfield::ComponentVoxel::Reduce (
const double xin,
const double xmin,
const double xmax,
const bool simplePeriodic,
const bool mirrorPeriodic,
bool & isMirrored) const [private]

Reduce a coordinate to the basic cell (in case of periodicity).

### Reset()

void Garfield::ComponentVoxel::Reset () [override], [private]

### SetMedium()

void Garfield::ComponentVoxel::SetMedium (
const unsigned int i,
Medium * m)

Set the medium in region i.

### SetMesh()

void Garfield::ComponentVoxel::SetMesh (
const unsigned int nx,
const unsigned int ny,
const unsigned int nz,
const double xmin,
const double xmax,
const double ymin,
const double ymax,
const double zmin,
const double zmax)

Define the grid.

Parameters

| | |
|---|---|
| nx,ny,nz | number of bins along x, y, z. |
| xmin,xmax | range along $x$. |
| ymin,ymax | range along $y$. |
| zmin,zmax | range along $z$. |

### SetWeightingFieldOffset()

void Garfield::ComponentVoxel::SetWeightingFieldOffset (
const double x,
const double y,
const double z)

Offset coordinates in the weighting field, such that the same numerical weighting field map can be used for electrodes at different positions.

### UpdatePeriodicity()

void Garfield::ComponentVoxel::UpdatePeriodicity () [override], [private]

### WeightingField()

void Garfield::ComponentVoxel::WeightingField (
const double x,
const double y,
const double z,
double & wx,
double & wy,
double & wz,
const std::string & label) [override]

WeightingPotential()

double Garfield::ComponentVoxel::WeightingPotential (
const double x,
const double y,
const double z,
const std::string & label) [override]

### 7.34.4 Member Data Documentation

m_bfields

std::vector<std::vector<std::vector<Element> > > Garfield::ComponentVoxel::m_bfields [private]
Magnetic field values at each mesh element.
Definition at line 129 of file ComponentVoxel.hh.

m_dx

double Garfield::ComponentVoxel::m_dx = 0. [private]
Definition at line 139 of file ComponentVoxel.hh.

m_dy

double Garfield::ComponentVoxel::m_dy = 0. [private]
Definition at line 139 of file ComponentVoxel.hh.

m_dz

double Garfield::ComponentVoxel::m_dz = 0. [private]
Definition at line 139 of file ComponentVoxel.hh.

m_efields

std::vector<std::vector<std::vector<Element> > > Garfield::ComponentVoxel::m_efields [private]
Electric field values and potentials at each mesh element.
Definition at line 127 of file ComponentVoxel.hh.

m_hasBfield

bool Garfield::ComponentVoxel::m_hasBfield = false [private]
Definition at line 146 of file ComponentVoxel.hh.

m_hasEfield

bool Garfield::ComponentVoxel::m_hasEfield = false [private]
Definition at line 145 of file ComponentVoxel.hh.

m_hasMesh

bool Garfield::ComponentVoxel::m_hasMesh = false [private]
Definition at line 143 of file ComponentVoxel.hh.

m_hasPotential

bool Garfield::ComponentVoxel::m_hasPotential = false [private]
Definition at line 144 of file ComponentVoxel.hh.

m_hasWfield

bool Garfield::ComponentVoxel::m_hasWfield = false [private]
Definition at line 147 of file ComponentVoxel.hh.

### m_interpolate

bool Garfield::ComponentVoxel::m_interpolate = false [private]
Definition at line 141 of file ComponentVoxel.hh.

### m_media

std::vector<Medium*> Garfield::ComponentVoxel::m_media [private]
Definition at line 118 of file ComponentVoxel.hh.

### m_nX

unsigned int Garfield::ComponentVoxel::m_nX = 0 [private]
Definition at line 136 of file ComponentVoxel.hh.

### m_nY

unsigned int Garfield::ComponentVoxel::m_nY = 0 [private]
Definition at line 136 of file ComponentVoxel.hh.

### m_nZ

unsigned int Garfield::ComponentVoxel::m_nZ = 0 [private]
Definition at line 136 of file ComponentVoxel.hh.

### m_pMax

double Garfield::ComponentVoxel::m_pMax = 0. [private]
Definition at line 155 of file ComponentVoxel.hh.

### m_pMin

double Garfield::ComponentVoxel::m_pMin = 0. [private]
Definition at line 155 of file ComponentVoxel.hh.

### m_regions

std::vector<std::vector<std::vector<int> > > Garfield::ComponentVoxel::m_regions [private]
Region indices.
Definition at line 125 of file ComponentVoxel.hh.

### m_wdfields

std::vector<std::vector<std::vector<std::vector<Element> > > > Garfield::ComponentVoxel::m_wdfields [private]
Delayed weighting field values and potentials at each mesh element.
Definition at line 133 of file ComponentVoxel.hh.

### m_wField_xOffset

double Garfield::ComponentVoxel::m_wField_xOffset = 0. [private]
Definition at line 150 of file ComponentVoxel.hh.

### m_wField_yOffset

double Garfield::ComponentVoxel::m_wField_yOffset = 0. [private]
Definition at line 151 of file ComponentVoxel.hh.

### m_wField_zOffset

double Garfield::ComponentVoxel::m_wField_zOffset = 0. [private]
Definition at line 152 of file ComponentVoxel.hh.

m_wfields

std::vector<std::vector<std::vector<Element> > > Garfield::ComponentVoxel::m_wfields [private]
Prompt weighting field values and potentials at each mesh element.
Definition at line 131 of file ComponentVoxel.hh.

m_xMax

double Garfield::ComponentVoxel::m_xMax = 0. [private]
Definition at line 138 of file ComponentVoxel.hh.

m_xMin

double Garfield::ComponentVoxel::m_xMin = 0. [private]
Definition at line 137 of file ComponentVoxel.hh.

m_yMax

double Garfield::ComponentVoxel::m_yMax = 0. [private]
Definition at line 138 of file ComponentVoxel.hh.

m_yMin

double Garfield::ComponentVoxel::m_yMin = 0. [private]
Definition at line 137 of file ComponentVoxel.hh.

m_zMax

double Garfield::ComponentVoxel::m_zMax = 0. [private]
Definition at line 138 of file ComponentVoxel.hh.

m_zMin

double Garfield::ComponentVoxel::m_zMin = 0. [private]
Definition at line 137 of file ComponentVoxel.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentVoxel.hh

## 7.35 Garfield::MediumMagboltz::Deexcitation Struct Reference

Public Attributes

- int gas
- int level
- std::string label
- double energy
- std::vector< double > p
- std::vector< int > final
- std::vector< int > type
- double osc
- double rate
- double sDoppler
- double gPressure
- double width
- double cf

### 7.35.1 Detailed Description

Definition at line 382 of file MediumMagboltz.hh.

### 7.35.2 Member Data Documentation

#### cf

double Garfield::MediumMagboltz::Deexcitation::cf
Definition at line 408 of file MediumMagboltz.hh.

#### energy

double Garfield::MediumMagboltz::Deexcitation::energy
Definition at line 390 of file MediumMagboltz.hh.

#### final

std::vector<int> Garfield::MediumMagboltz::Deexcitation::final
Definition at line 394 of file MediumMagboltz.hh.

#### gas

int Garfield::MediumMagboltz::Deexcitation::gas
Definition at line 384 of file MediumMagboltz.hh.

#### gPressure

double Garfield::MediumMagboltz::Deexcitation::gPressure
Definition at line 404 of file MediumMagboltz.hh.

#### label

std::string Garfield::MediumMagboltz::Deexcitation::label
Definition at line 388 of file MediumMagboltz.hh.

#### level

int Garfield::MediumMagboltz::Deexcitation::level
Definition at line 386 of file MediumMagboltz.hh.

#### osc

double Garfield::MediumMagboltz::Deexcitation::osc
Definition at line 398 of file MediumMagboltz.hh.

#### p

std::vector<double> Garfield::MediumMagboltz::Deexcitation::p
Definition at line 392 of file MediumMagboltz.hh.

#### rate

double Garfield::MediumMagboltz::Deexcitation::rate
Definition at line 400 of file MediumMagboltz.hh.

#### sDoppler

double Garfield::MediumMagboltz::Deexcitation::sDoppler
Definition at line 402 of file MediumMagboltz.hh.

#### type

std::vector<int> Garfield::MediumMagboltz::Deexcitation::type
Definition at line 396 of file MediumMagboltz.hh.

width

double Garfield::MediumMagboltz::Deexcitation::width
Definition at line 406 of file MediumMagboltz.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumMagboltz.hh

## 7.36 Garfield::ComponentTcadBase< N >::Defect Struct Reference

#include <ComponentTcadBase.hh>

Public Attributes

- double xsece
- double xsech
- double conc

### 7.36.1 Detailed Description

template<size_t N>
struct Garfield::ComponentTcadBase< N >::Defect

Definition at line 267 of file ComponentTcadBase.hh.

### 7.36.2 Member Data Documentation

conc

template<size_t N>
double Garfield::ComponentTcadBase< N >::Defect::conc
Definition at line 273 of file ComponentTcadBase.hh.

xsece

template<size_t N>
double Garfield::ComponentTcadBase< N >::Defect::xsece
Definition at line 269 of file ComponentTcadBase.hh.

xsech

template<size_t N>
double Garfield::ComponentTcadBase< N >::Defect::xsech
Definition at line 271 of file ComponentTcadBase.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh

## 7.37 Garfield::DriftLineRKF Class Reference

Calculation of drift lines based on macroscopic transport coefficients using Runge-Kutta-Fehlberg integration.
#include <DriftLineRKF.hh>

Public Member Functions

- DriftLineRKF ()
  Default constructor.
- DriftLineRKF (Sensor *sensor)
  Constructor.
- ~DriftLineRKF ()
  Destructor.

- void SetSensor (Sensor *s)

  Set the sensor.
- void EnablePlotting (ViewDrift *view)

  Switch on drift line plotting.
- void DisablePlotting ()

  Switch off drift line plotting.
- void EnableSignalCalculation (const bool on=true)

  Switch calculation of induced currents on or off (default: enabled).
- void SetSignalAveragingOrder (const unsigned int navg)

  Set the number of points to be used when averaging the delayed signal vector over a time bin in the Sensor class.
- void UseWeightingPotential (const bool on=true)

  Use the weighting potential (as opposed to the weighting field) for calculating the induced signal.
- void SetIntegrationAccuracy (const double eps)

  Set the accuracy of the Runge Kutta Fehlberg drift line integration.
- void SetMaximumStepSize (const double ms)

  Set (explicitly) the maximum step size that is allowed.
- void SetMaximumStepSize ()

  Try to set an upper limit to the allowable step size based on the feature size of the sensor.
- void UnsetMaximumStepSize ()

  Do not apply an upper limit to the allowable step size.
- void RejectKinks (const bool on=true)

  Request (or not) the drift line calculation to be aborted if the drift line makes a bend sharper than 90 degrees.
- void SetElectronSignalScalingFactor (const double scale)

  Set multiplication factor for the signal induced by electrons.
- void SetHoleSignalScalingFactor (const double scale)

  Set multiplication factor for the signal induced by holes.
- void SetIonSignalScalingFactor (const double scale)

  Set multiplication factor for the signal induced by ions.
- void EnableAvalanche (const bool on=true)

  Enable/disable simulation electron multiplication (default: on).
- void EnableIonTail (const bool on=true)

  Enable/disable simulation of the ion tail (default: on).
- void EnableNegativeIonTail (const bool on=true)

  Enable/disable simulation of the negative ion tail (default: off).
- void SetGainFluctuationsFixed (const double gain=-1.)

  Do not randomize the avalanche size.
- void SetGainFluctuationsPolya (const double theta, const double mean=-1., const bool quiet=false)

  Sample the avalanche size from a Polya distribution with shape parameter theta.
- void EnableTownsendMap (const bool on=true)

  Retrieve the Townsend coefficient from the component.
- void EnableVelocityMap (const bool on=true)

  Retrieve the drift velocity from the component.
- bool DriftElectron (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of an electron with a given starting point.
- bool DriftHole (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of a hole with a given starting point.
- bool DriftIon (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of an ion with a given starting point.

- bool DriftPositron (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of an electron with a given starting point, assuming that it has positive charge.
- bool DriftNegativeIon (const double x, const double y, const double z, const double t, const size_t w=1)

  Simulate the drift line of an ion with a given starting point, assuming that it has negative charge.
- void PrintDriftLine () const

  Print the trajectory of the most recent drift line.
- void GetEndPoint (double &x, double &y, double &z, double &t, int &st) const

  Get the end point and status flag of the most recent drift line.
- size_t GetNumberOfDriftLinePoints () const

  Get the number of points of the most recent drift line.
- void GetDriftLinePoint (const size_t i, double &x, double &y, double &z, double &t) const

  Get the coordinates and time of a point along the most recent drift line.
- double GetArrivalTimeSpread (const double eps=1.e-4) const

  Compute the sigma of the arrival time distribution for the current drift line by integrating the longitudinal diffusion coefficient.
- double GetGain (const double eps=1.e-4) const

  Compute the multiplication factor for the current drift line.
- double GetLoss (const double eps=1.e-4) const

  Compute the attachment loss factor for the current drift line.
- double GetDriftTime () const

  Get the cumulative drift time.
- double GetPathLength () const

  Get the cumulative path length.
- void GetAvalancheSize (double &ne, double &ni) const

  Return the number of electrons and ions in the avalanche.
- std::pair< double, double > GetAvalancheSize () const

  Return the number of electrons and ions in the avalanche.
- bool FieldLine (const double xi, const double yi, const double zi, std::vector< std::array< float, 3 > > &xl, const bool electron=true) const

  Compute an electric field line.
- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off (default: off).

Private Types

- enum class GainFluctuations { None = 0 , Polya }

Private Member Functions

- bool DriftLine (const std::array< double, 3 > &x0, const double t0, const Particle particle, std::vector< double > &ts, std::vector< std::array< double, 3 > > &xs, int &status) const
- bool Avalanche (const Particle particle, const std::vector< std::array< double, 3 > > &xs, std::vector< double > &ne, std::vector< double > &ni, std::vector< double > &nn, double &scale) const
- bool AddIonTail (const std::vector< double > &te, const std::vector< std::array< double, 3 > > &xe, const std::vector< double > &ni, const double scale) const
- bool AddNegativeIonTail (const std::vector< double > &te, const std::vector< std::array< double, 3 > > &xe, const std::vector< double > &nn, const double scale) const
- int GetField (const std::array< double, 3 > &x, double &ex, double &ey, double &ez, double &bx, double &by, double &bz, Medium *&medium) const
- std::array< double, 3 > GetVelocity (const std::array< double, 3 > &x, const Particle particle, int &status) const

- bool GetDiffusion (const std::array< double, 3 > &x, const Particle particle, double &dl, double &dt) const
- double GetVar (const std::array< double, 3 > &x, const Particle particle) const
- double GetAlpha (const std::array< double, 3 > &x, const Particle particle) const
- double GetEta (const std::array< double, 3 > &x, const Particle particle) const
- bool Terminate (const std::array< double, 3 > &xx0, const std::array< double, 3 > &xx1, const Particle particle, std::vector< double > &ts, std::vector< std::array< double, 3 > > &xs) const
- bool DriftToWire (const double xw, const double yw, const double rw, const Particle particle, std::vector< double > &ts, std::vector< std::array< double, 3 > > &xs, int &stat) const
- double ComputeSigma (const std::vector< std::array< double, 3 > > &x, const Particle particle, const double eps) const
- double ComputeGain (const std::vector< std::array< double, 3 > > &x, const Particle particle, const double eps) const
- double ComputeLoss (const std::vector< std::array< double, 3 > > &x, const Particle particle, const double eps) const
- double IntegrateDiffusion (const std::array< double, 3 > &xi, const std::array< double, 3 > &xe, const Particle particle, const double tol) const
- double IntegrateAlpha (const std::array< double, 3 > &xi, const std::array< double, 3 > &xe, const Particle particle, const double tol) const
- double IntegrateEta (const std::array< double, 3 > &xi, const std::array< double, 3 > &xe, const Particle particle, const double tol) const
- void ComputeSignal (const Particle particle, const double scale, const std::vector< double > &ts, const std::vector< std::array< double, 3 > > &xs, const std::vector< double > &ne) const
- void Terminate (const std::array< double, 3 > &xx0, const std::array< double, 3 > &xx1, std::vector< std::array< float, 3 > > &xs) const

Static Private Member Functions

- static double Charge (const Particle particle)

Private Attributes

- std::string m_className = "DriftLineRKF"
- Sensor * m_sensor = nullptr
- Particle m_particle = Particle::Electron
- double m_maxStepSize = 0.
- double m_accuracy = 1.e-8
- bool m_rejectKinks = true
- bool m_useStepSizeLimit = false
- ViewDrift * m_view = nullptr
- std::vector< std::array< double, 3 > > m_x
- std::vector< double > m_t
- int m_status = 0
- bool m_doSignal = true
- unsigned int m_navg = 2
- bool m_useWeightingPotential = true
- double m_scaleE = 1.
- double m_scaleH = 1.
- double m_scaleI = 1.
- bool m_useVelocityMap = false
- bool m_useTownsendMap = false
- bool m_doAvalanche = true
- GainFluctuations m_gainFluctuations = GainFluctuations::None
- double m_theta = 0.
- double m_gain = -1.
- bool m_doIonTail = true

- bool m_doIonTailAuto = true
- bool m_doNegativeIonTail = false
- double m_nE = 0.
- double m_nI = 0.
- bool m_debug = false

### 7.37.1 Detailed Description

Calculation of drift lines based on macroscopic transport coefficients using Runge-Kutta-Fehlberg integration.
Definition at line 18 of file DriftLineRKF.hh.

### 7.37.2 Member Enumeration Documentation

#### GainFluctuations

enum class Garfield::DriftLineRKF::GainFluctuations [strong], [private]

Enumerator

| | |
|---|---|
| None | |
| Polya | |

Definition at line 203 of file DriftLineRKF.hh.
00203 { None = 0, Polya };

### 7.37.3 Constructor & Destructor Documentation

#### DriftLineRKF() [1/2]

Garfield::DriftLineRKF::DriftLineRKF () [inline]
Default constructor.
Definition at line 21 of file DriftLineRKF.hh.
00021 : DriftLineRKF(nullptr) {}

#### DriftLineRKF() [2/2]

Garfield::DriftLineRKF::DriftLineRKF (
Sensor * sensor)
Constructor.

#### ~DriftLineRKF()

Garfield::DriftLineRKF::~DriftLineRKF () [inline]
Destructor.
Definition at line 25 of file DriftLineRKF.hh.
00025 {}

### 7.37.4 Member Function Documentation

#### AddIonTail()

bool Garfield::DriftLineRKF::AddIonTail (
const std::vector< double > & te,
const std::vector< std::array< double, 3 > > & xe,
const std::vector< double > & ni,
const double scale) const [private]

### AddNegativeIonTail()

bool Garfield::DriftLineRKF::AddNegativeIonTail (
const std::vector< double > & te,
const std::vector< std::array< double, 3 > > & xe,
const std::vector< double > & nn,
const double scale) const [private]

### Avalanche()

bool Garfield::DriftLineRKF::Avalanche (
const Particle particle,
const std::vector< std::array< double, 3 > > & xs,
std::vector< double > & ne,
std::vector< double > & ni,
std::vector< double > & nn,
double & scale) const [private]

### Charge()

double Garfield::DriftLineRKF::Charge (
const Particle particle) [inline], [static], [private]

Definition at line 301 of file DriftLineRKF.hh.

```
00301                                    {
00302     if (particle == Particle::Electron || particle == Particle::NegativeIon) {
00303       return -1.;
00304     }
00305     return 1.;
00306   }
```

### ComputeGain()

double Garfield::DriftLineRKF::ComputeGain (
const std::vector< std::array< double, 3 > > & x,
const Particle particle,
const double eps) const [private]

### ComputeLoss()

double Garfield::DriftLineRKF::ComputeLoss (
const std::vector< std::array< double, 3 > > & x,
const Particle particle,
const double eps) const [private]

### ComputeSigma()

double Garfield::DriftLineRKF::ComputeSigma (
const std::vector< std::array< double, 3 > > & x,
const Particle particle,
const double eps) const [private]

### ComputeSignal()

void Garfield::DriftLineRKF::ComputeSignal (
const Particle particle,
const double scale,
const std::vector< double > & ts,
const std::vector< std::array< double, 3 > > & xs,
const std::vector< double > & ne) const [private]

### DisablePlotting()

void Garfield::DriftLineRKF::DisablePlotting ()
Switch off drift line plotting.

### DriftElectron()

bool Garfield::DriftLineRKF::DriftElectron (
const double x,
const double y,
const double z,
const double t,
const size_t w = 1)
Simulate the drift line of an electron with a given starting point.

### DriftHole()

bool Garfield::DriftLineRKF::DriftHole (
const double x,
const double y,
const double z,
const double t,
const size_t w = 1)
Simulate the drift line of a hole with a given starting point.

### DriftIon()

bool Garfield::DriftLineRKF::DriftIon (
const double x,
const double y,
const double z,
const double t,
const size_t w = 1)
Simulate the drift line of an ion with a given starting point.

### DriftLine()

bool Garfield::DriftLineRKF::DriftLine (
const std::array< double, 3 > & x0,
const double t0,
const Particle particle,
std::vector< double > & ts,
std::vector< std::array< double, 3 > > & xs,
int & status) const [private]

### DriftNegativeIon()

bool Garfield::DriftLineRKF::DriftNegativeIon (
const double x,
const double y,
const double z,
const double t,
const size_t w = 1)
Simulate the drift line of an ion with a given starting point, assuming that it has negative charge.

### DriftPositron()

bool Garfield::DriftLineRKF::DriftPositron (
const double x,
const double y,
const double z,

const double t,
const size_t w = 1)

Simulate the drift line of an electron with a given starting point, assuming that it has positive charge.

### DriftToWire()

bool Garfield::DriftLineRKF::DriftToWire (
const double xw,
const double yw,
const double rw,
const Particle particle,
std::vector< double > & ts,
std::vector< std::array< double, 3 > > & xs,
int & stat) const [private]

### EnableAvalanche()

void Garfield::DriftLineRKF::EnableAvalanche (
const bool on = true) [inline]

Enable/disable simulation electron multiplication (default: on).
Definition at line 70 of file DriftLineRKF.hh.

```
00070 { m_doAvalanche = on; }
```

### EnableDebugging()

void Garfield::DriftLineRKF::EnableDebugging (
const bool on = true) [inline]

Switch debugging messages on/off (default: off).
Definition at line 152 of file DriftLineRKF.hh.

```
00152 { m_debug = on; }
```

### EnableIonTail()

void Garfield::DriftLineRKF::EnableIonTail (
const bool on = true) [inline]

Enable/disable simulation of the ion tail (default: on).
Definition at line 72 of file DriftLineRKF.hh.

```
00072                                   {
00073     m_doIonTail = on;
00074     m_doIonTailAuto = false;
00075   }
```

### EnableNegativeIonTail()

void Garfield::DriftLineRKF::EnableNegativeIonTail (
const bool on = true) [inline]

Enable/disable simulation of the negative ion tail (default: off).
Definition at line 77 of file DriftLineRKF.hh.

```
00077 { m_doNegativeIonTail = on; }
```

### EnablePlotting()

void Garfield::DriftLineRKF::EnablePlotting (
ViewDrift * view)

Switch on drift line plotting.

### EnableSignalCalculation()

void Garfield::DriftLineRKF::EnableSignalCalculation (
const bool on = true) [inline]

Switch calculation of induced currents on or off (default: enabled).

Definition at line 36 of file DriftLineRKF.hh.

```
00036 { m_doSignal = on; }
```

### EnableTownsendMap()

```
void Garfield::DriftLineRKF::EnableTownsendMap (
                const bool on = true)   [inline]
```

Retrieve the Townsend coefficient from the component.
Definition at line 86 of file DriftLineRKF.hh.

```
00086 { m_useTownsendMap = on; }
```

### EnableVelocityMap()

```
void Garfield::DriftLineRKF::EnableVelocityMap (
                const bool on = true)   [inline]
```

Retrieve the drift velocity from the component.
Definition at line 88 of file DriftLineRKF.hh.

```
00088 { m_useVelocityMap = on; }
```

### FieldLine()

```
bool Garfield::DriftLineRKF::FieldLine (
                const double xi,
                const double yi,
                const double zi,
                std::vector< std::array< float, 3 > > & xl,
                const bool electron = true) const
```

Compute an electric field line.

Parameters

| | |
|---|---|
| xi,yi,zi | starting point |
| xl | points along the field line |
| electron | flag to set the direction in which to follow the field |

### GetAlpha()

```
double Garfield::DriftLineRKF::GetAlpha (
                const std::array< double, 3 > & x,
                const Particle particle) const   [private]
```

### GetArrivalTimeSpread()

```
double Garfield::DriftLineRKF::GetArrivalTimeSpread (
                const double eps = 1.e-4) const
```

Compute the sigma of the arrival time distribution for the current drift line by integrating the longitudinal diffusion coefficient.

### GetAvalancheSize() [1/2]

```
std::pair< double, double > Garfield::DriftLineRKF::GetAvalancheSize () const   [inline]
```

Return the number of electrons and ions in the avalanche.
Definition at line 138 of file DriftLineRKF.hh.

```
00138                                       {
00139     return std::make_pair(m_nE, m_nI);
00140   }
```

### GetAvalancheSize() [2/2]

```
void Garfield::DriftLineRKF::GetAvalancheSize (
                double & ne,
                double & ni) const    [inline]
```

Return the number of electrons and ions in the avalanche.
Definition at line 133 of file DriftLineRKF.hh.

```
00133                                                 {
00134     ne = m_nE;
00135     ni = m_nI;
00136   }
```

### GetDiffusion()

```
bool Garfield::DriftLineRKF::GetDiffusion (
                const std::array< double, 3 > & x,
                const Particle particle,
                double & dl,
                double & dt) const    [private]
```

### GetDriftLinePoint()

```
void Garfield::DriftLineRKF::GetDriftLinePoint (
                const size_t i,
                double & x,
                double & y,
                double & z,
                double & t) const
```

Get the coordinates and time of a point along the most recent drift line.

### GetDriftTime()

```
double Garfield::DriftLineRKF::GetDriftTime () const    [inline]
```

Get the cumulative drift time.
Definition at line 126 of file DriftLineRKF.hh.

```
00126                                 {
00127     return m_t.empty() ? 0. : m_t.back() - m_t.front();
00128   }
```

### GetEndPoint()

```
void Garfield::DriftLineRKF::GetEndPoint (
                double & x,
                double & y,
                double & z,
                double & t,
                int & st) const
```

Get the end point and status flag of the most recent drift line.

### GetEta()

```
double Garfield::DriftLineRKF::GetEta (
                const std::array< double, 3 > & x,
                const Particle particle) const    [private]
```

### GetField()

```
int Garfield::DriftLineRKF::GetField (
                const std::array< double, 3 > & x,
                double & ex,
                double & ey,
                double & ez,
```

double & bx,
double & by,
double & bz,
Medium *& medium) const [private]

### GetGain()

double Garfield::DriftLineRKF::GetGain (
const double eps = 1.e-4) const

Compute the multiplication factor for the current drift line.

### GetLoss()

double Garfield::DriftLineRKF::GetLoss (
const double eps = 1.e-4) const

Compute the attachment loss factor for the current drift line.

### GetNumberOfDriftLinePoints()

size_t Garfield::DriftLineRKF::GetNumberOfDriftLinePoints () const [inline]

Get the number of points of the most recent drift line.

Definition at line 113 of file DriftLineRKF.hh.

```
00113 { return m_x.size(); }
```

### GetPathLength()

double Garfield::DriftLineRKF::GetPathLength () const

Get the cumulative path length.

### GetVar()

double Garfield::DriftLineRKF::GetVar (
const std::array< double, 3 > & x,
const Particle particle) const [private]

### GetVelocity()

std::array< double, 3 > Garfield::DriftLineRKF::GetVelocity (
const std::array< double, 3 > & x,
const Particle particle,
int & status) const [private]

### IntegrateAlpha()

double Garfield::DriftLineRKF::IntegrateAlpha (
const std::array< double, 3 > & xi,
const std::array< double, 3 > & xe,
const Particle particle,
const double tol) const [private]

### IntegrateDiffusion()

double Garfield::DriftLineRKF::IntegrateDiffusion (
const std::array< double, 3 > & xi,
const std::array< double, 3 > & xe,
const Particle particle,
const double tol) const [private]

### IntegrateEta()

```
double Garfield::DriftLineRKF::IntegrateEta (
            const std::array< double, 3 > & xi,
            const std::array< double, 3 > & xe,
            const Particle particle,
            const double tol) const    [private]
```

### PrintDriftLine()

```
void Garfield::DriftLineRKF::PrintDriftLine () const
```

Print the trajectory of the most recent drift line.

### RejectKinks()

```
void Garfield::DriftLineRKF::RejectKinks (
            const bool on = true)    [inline]
```

Request (or not) the drift line calculation to be aborted if the drift line makes a bend sharper than 90 degrees.

Definition at line 60 of file DriftLineRKF.hh.

```
00060 { m_rejectKinks = on; }
```

### SetElectronSignalScalingFactor()

```
void Garfield::DriftLineRKF::SetElectronSignalScalingFactor (
            const double scale)    [inline]
```

Set multiplication factor for the signal induced by electrons.

Definition at line 63 of file DriftLineRKF.hh.

```
00063 { m_scaleE = scale; }
```

### SetGainFluctuationsFixed()

```
void Garfield::DriftLineRKF::SetGainFluctuationsFixed (
            const double gain = -1.)
```

Do not randomize the avalanche size.

### SetGainFluctuationsPolya()

```
void Garfield::DriftLineRKF::SetGainFluctuationsPolya (
            const double theta,
            const double mean = -1.,
            const bool quiet = false)
```

Sample the avalanche size from a Polya distribution with shape parameter theta.

### SetHoleSignalScalingFactor()

```
void Garfield::DriftLineRKF::SetHoleSignalScalingFactor (
            const double scale)    [inline]
```

Set multiplication factor for the signal induced by holes.

Definition at line 65 of file DriftLineRKF.hh.

```
00065 { m_scaleH = scale; }
```

### SetIntegrationAccuracy()

```
void Garfield::DriftLineRKF::SetIntegrationAccuracy (
            const double eps)
```

Set the accuracy of the Runge Kutta Fehlberg drift line integration.

### SetIonSignalScalingFactor()

void Garfield::DriftLineRKF::SetIonSignalScalingFactor (
const double scale) [inline]

Set multiplication factor for the signal induced by ions.
Definition at line 67 of file DriftLineRKF.hh.

```
00067 { m_scaleI = scale; }
```

### SetMaximumStepSize() [1/2]

void Garfield::DriftLineRKF::SetMaximumStepSize ()

Try to set an upper limit to the allowable step size based on the feature size of the sensor.

### SetMaximumStepSize() [2/2]

void Garfield::DriftLineRKF::SetMaximumStepSize (
const double ms)

Set (explicitly) the maximum step size that is allowed.

### SetSensor()

void Garfield::DriftLineRKF::SetSensor (
Sensor * s)

Set the sensor.

### SetSignalAveragingOrder()

void Garfield::DriftLineRKF::SetSignalAveragingOrder (
const unsigned int navg) [inline]

Set the number of points to be used when averaging the delayed signal vector over a time bin in the Sensor class.
The averaging is done with a $2 \times navg + 1$ point Newton-Raphson integration. Default: 2.
Definition at line 41 of file DriftLineRKF.hh.

```
00041 { m_navg = navg; }
```

### Terminate() [1/2]

bool Garfield::DriftLineRKF::Terminate (
const std::array< double, 3 > & xx0,
const std::array< double, 3 > & xx1,
const Particle particle,
std::vector< double > & ts,
std::vector< std::array< double, 3 > > & xs) const [private]

### Terminate() [2/2]

void Garfield::DriftLineRKF::Terminate (
const std::array< double, 3 > & xx0,
const std::array< double, 3 > & xx1,
std::vector< std::array< float, 3 > > & xs) const [private]

### UnsetMaximumStepSize()

void Garfield::DriftLineRKF::UnsetMaximumStepSize () [inline]

Do not apply an upper limit to the allowable step size.
Definition at line 57 of file DriftLineRKF.hh.

```
00057 { m_useStepSizeLimit = false; }
```

UseWeightingPotential()

void Garfield::DriftLineRKF::UseWeightingPotential (
const bool on = true) [inline]

Use the weighting potential (as opposed to the weighting field) for calculating the induced signal.
By default, the weighting potential is used.
Definition at line 45 of file DriftLineRKF.hh.

```
00045                                          {
00046     m_useWeightingPotential = on;
00047   }
```

### 7.37.5 Member Data Documentation

m_accuracy

double Garfield::DriftLineRKF::m_accuracy = 1.e-8 [private]
Definition at line 166 of file DriftLineRKF.hh.

m_className

std::string Garfield::DriftLineRKF::m_className = "DriftLineRKF" [private]
Definition at line 155 of file DriftLineRKF.hh.

m_debug

bool Garfield::DriftLineRKF::m_debug = false [private]
Definition at line 221 of file DriftLineRKF.hh.

m_doAvalanche

bool Garfield::DriftLineRKF::m_doAvalanche = true [private]
Definition at line 202 of file DriftLineRKF.hh.

m_doIonTail

bool Garfield::DriftLineRKF::m_doIonTail = true [private]
Definition at line 212 of file DriftLineRKF.hh.

m_doIonTailAuto

bool Garfield::DriftLineRKF::m_doIonTailAuto = true [private]
Definition at line 214 of file DriftLineRKF.hh.

m_doNegativeIonTail

bool Garfield::DriftLineRKF::m_doNegativeIonTail = false [private]
Definition at line 216 of file DriftLineRKF.hh.

m_doSignal

bool Garfield::DriftLineRKF::m_doSignal = true [private]
Definition at line 183 of file DriftLineRKF.hh.

m_gain

double Garfield::DriftLineRKF::m_gain = -1. [private]
Definition at line 209 of file DriftLineRKF.hh.

m_gainFluctuations

GainFluctuations Garfield::DriftLineRKF::m_gainFluctuations = GainFluctuations::None [private]
Definition at line 205 of file DriftLineRKF.hh.

### m_maxStepSize

double Garfield::DriftLineRKF::m_maxStepSize = 0. [private]
Definition at line 164 of file DriftLineRKF.hh.

### m_navg

unsigned int Garfield::DriftLineRKF::m_navg = 2 [private]
Definition at line 185 of file DriftLineRKF.hh.

### m_nE

double Garfield::DriftLineRKF::m_nE = 0. [private]
Definition at line 218 of file DriftLineRKF.hh.

### m_nI

double Garfield::DriftLineRKF::m_nI = 0. [private]
Definition at line 218 of file DriftLineRKF.hh.

### m_particle

Particle Garfield::DriftLineRKF::m_particle = Particle::Electron [private]
Definition at line 161 of file DriftLineRKF.hh.

### m_rejectKinks

bool Garfield::DriftLineRKF::m_rejectKinks = true [private]
Definition at line 168 of file DriftLineRKF.hh.

### m_scaleE

double Garfield::DriftLineRKF::m_scaleE = 1. [private]
Definition at line 190 of file DriftLineRKF.hh.

### m_scaleH

double Garfield::DriftLineRKF::m_scaleH = 1. [private]
Definition at line 192 of file DriftLineRKF.hh.

### m_scaleI

double Garfield::DriftLineRKF::m_scaleI = 1. [private]
Definition at line 194 of file DriftLineRKF.hh.

### m_sensor

Sensor* Garfield::DriftLineRKF::m_sensor = nullptr [private]
Definition at line 158 of file DriftLineRKF.hh.

### m_status

int Garfield::DriftLineRKF::m_status = 0 [private]
Definition at line 180 of file DriftLineRKF.hh.

### m_t

std::vector<double> Garfield::DriftLineRKF::m_t [private]
Definition at line 178 of file DriftLineRKF.hh.

m_theta

double Garfield::DriftLineRKF::m_theta = 0. [private]
Definition at line 207 of file DriftLineRKF.hh.

m_useStepSizeLimit

bool Garfield::DriftLineRKF::m_useStepSizeLimit = false [private]
Definition at line 170 of file DriftLineRKF.hh.

m_useTownsendMap

bool Garfield::DriftLineRKF::m_useTownsendMap = false [private]
Definition at line 199 of file DriftLineRKF.hh.

m_useVelocityMap

bool Garfield::DriftLineRKF::m_useVelocityMap = false [private]
Definition at line 197 of file DriftLineRKF.hh.

m_useWeightingPotential

bool Garfield::DriftLineRKF::m_useWeightingPotential = true [private]
Definition at line 187 of file DriftLineRKF.hh.

m_view

ViewDrift* Garfield::DriftLineRKF::m_view = nullptr [private]
Definition at line 173 of file DriftLineRKF.hh.

m_x

std::vector<std::array<double, 3> > Garfield::DriftLineRKF::m_x [private]
Definition at line 176 of file DriftLineRKF.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/DriftLineRKF.hh

## 7.38 Garfield::ComponentParallelPlate::Electrode Struct Reference

Structure that captures the information of the electrodes under study.

Public Attributes

- std::string label
  Label.
- int ind = structureelectrode::NotSet
  Readout group.
- double xpos
- double ypos
  Coordinates in x/y.
- double lx
- double ly
  Dimensions in the x-y plane.
- bool formAnode = true
- bool m_usegrid = false
  Enabling grid based calculations.
- ComponentGrid grid
  grid object.

### 7.38.1 Detailed Description

Structure that captures the information of the electrodes under study.
Definition at line 202 of file ComponentParallelPlate.hh.

### 7.38.2 Member Data Documentation

formAnode

bool Garfield::ComponentParallelPlate::Electrode::formAnode = true
Definition at line 207 of file ComponentParallelPlate.hh.

grid

ComponentGrid Garfield::ComponentParallelPlate::Electrode::grid
grid object.
Definition at line 210 of file ComponentParallelPlate.hh.

ind

int Garfield::ComponentParallelPlate::Electrode::ind = structureelectrode::NotSet
Readout group.
Definition at line 204 of file ComponentParallelPlate.hh.

label

std::string Garfield::ComponentParallelPlate::Electrode::label
Label.
Definition at line 203 of file ComponentParallelPlate.hh.

lx

double Garfield::ComponentParallelPlate::Electrode::lx
Definition at line 206 of file ComponentParallelPlate.hh.

ly

double Garfield::ComponentParallelPlate::Electrode::ly
Dimensions in the x-y plane.
Definition at line 206 of file ComponentParallelPlate.hh.

m_usegrid

bool Garfield::ComponentParallelPlate::Electrode::m_usegrid = false
Enabling grid based calculations.
Definition at line 209 of file ComponentParallelPlate.hh.

xpos

double Garfield::ComponentParallelPlate::Electrode::xpos
Definition at line 205 of file ComponentParallelPlate.hh.

ypos

double Garfield::ComponentParallelPlate::Electrode::ypos
Coordinates in x/y.
Definition at line 205 of file ComponentParallelPlate.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentParallelPlate.hh

## 7.39 Garfield::AvalancheMicroscopic::Electron Struct Reference

#include <AvalancheMicroscopic.hh>

Public Attributes

- int status = 0
  Status.
- std::vector< Point > path
  Drift line.
- size_t weight = 1
  Multiplicity.
- double pathLength = 0.
  Path length.

### 7.39.1 Detailed Description

Definition at line 181 of file AvalancheMicroscopic.hh.

### 7.39.2 Member Data Documentation

path

std::vector<Point> Garfield::AvalancheMicroscopic::Electron::path
Drift line.
Definition at line 183 of file AvalancheMicroscopic.hh.

pathLength

double Garfield::AvalancheMicroscopic::Electron::pathLength = 0.
Path length.
Definition at line 185 of file AvalancheMicroscopic.hh.

status

int Garfield::AvalancheMicroscopic::Electron::status = 0
Status.
Definition at line 182 of file AvalancheMicroscopic.hh.

weight

size_t Garfield::AvalancheMicroscopic::Electron::weight = 1
Multiplicity.
Definition at line 184 of file AvalancheMicroscopic.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh

## 7.40 Garfield::TrackDegrade::Electron Struct Reference

#include <TrackDegrade.hh>

Public Attributes

- double x = 0.
- double y = 0.
- double z = 0.
- double t = 0.
- double energy = 0.
- double dx = 0.
- double dy = 0.
- double dz = 0.

### 7.40.1 Detailed Description

Definition at line 16 of file TrackDegrade.hh.

### 7.40.2 Member Data Documentation

dx

double Garfield::TrackDegrade::Electron::dx = 0.
Definition at line 22 of file TrackDegrade.hh.

dy

double Garfield::TrackDegrade::Electron::dy = 0.
Definition at line 23 of file TrackDegrade.hh.

dz

double Garfield::TrackDegrade::Electron::dz = 0.
Definition at line 24 of file TrackDegrade.hh.

energy

double Garfield::TrackDegrade::Electron::energy = 0.
Definition at line 21 of file TrackDegrade.hh.

t

double Garfield::TrackDegrade::Electron::t = 0.
Definition at line 20 of file TrackDegrade.hh.

x

double Garfield::TrackDegrade::Electron::x = 0.
Definition at line 17 of file TrackDegrade.hh.

y

double Garfield::TrackDegrade::Electron::y = 0.
Definition at line 18 of file TrackDegrade.hh.

z

double Garfield::TrackDegrade::Electron::z = 0.
Definition at line 19 of file TrackDegrade.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh

## 7.41 Garfield::TrackHeed::Electron Struct Reference

#include <TrackHeed.hh>

Public Attributes

- double x = 0.
- double y = 0.
- double z = 0.
- double t = 0.
- double e = 0.
- double dx = 0.
- double dy = 0.
- double dz = 0.

### 7.41.1 Detailed Description

Definition at line 36 of file TrackHeed.hh.

### 7.41.2 Member Data Documentation

dx

double Garfield::TrackHeed::Electron::dx = 0.
Definition at line 42 of file TrackHeed.hh.

dy

double Garfield::TrackHeed::Electron::dy = 0.
Definition at line 43 of file TrackHeed.hh.

dz

double Garfield::TrackHeed::Electron::dz = 0.
Definition at line 44 of file TrackHeed.hh.

e

double Garfield::TrackHeed::Electron::e = 0.
Definition at line 41 of file TrackHeed.hh.

t

double Garfield::TrackHeed::Electron::t = 0.
Definition at line 40 of file TrackHeed.hh.

x

double Garfield::TrackHeed::Electron::x = 0.
Definition at line 37 of file TrackHeed.hh.

y

double Garfield::TrackHeed::Electron::y = 0.
Definition at line 38 of file TrackHeed.hh.

z

double Garfield::TrackHeed::Electron::z = 0.
Definition at line 39 of file TrackHeed.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackHeed.hh

## 7.42 Garfield::ComponentFieldMap::Element Struct Reference

#include <ComponentFieldMap.hh>

Public Attributes

- int emap [10]
- unsigned int matmap

### 7.42.1 Detailed Description

Definition at line 171 of file ComponentFieldMap.hh.

### 7.42.2 Member Data Documentation

emap

int Garfield::ComponentFieldMap::Element::emap[10]
Definition at line 173 of file ComponentFieldMap.hh.

matmap

unsigned int Garfield::ComponentFieldMap::Element::matmap
Definition at line 175 of file ComponentFieldMap.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh

## 7.43 Garfield::ComponentNeBem2d::Element Struct Reference

Public Attributes

- double x
- double y

  Coordinates of the element centre (collocation point).
- double a

  Half-length.
- double cphi

  Rotation.
- double sphi

  Rotation.
- double q

  Charge density (solution).
- std::pair< BC, double > bc

  Boundary condition.
- double lambda

  Ratio of dielectric permittivities.

### 7.43.1 Detailed Description

Definition at line 169 of file ComponentNeBem2d.hh.

### 7.43.2 Member Data Documentation

a

double Garfield::ComponentNeBem2d::Element::a
Half-length.
Definition at line 171 of file ComponentNeBem2d.hh.

bc

std::pair<BC, double> Garfield::ComponentNeBem2d::Element::bc
Boundary condition.
Definition at line 175 of file ComponentNeBem2d.hh.

cphi

double Garfield::ComponentNeBem2d::Element::cphi
Rotation.
Definition at line 172 of file ComponentNeBem2d.hh.

lambda

double Garfield::ComponentNeBem2d::Element::lambda
Ratio of dielectric permittivities.
Definition at line 176 of file ComponentNeBem2d.hh.

q

double Garfield::ComponentNeBem2d::Element::q
Charge density (solution).
Definition at line 174 of file ComponentNeBem2d.hh.

sphi

double Garfield::ComponentNeBem2d::Element::sphi
Rotation.
Definition at line 173 of file ComponentNeBem2d.hh.

x

double Garfield::ComponentNeBem2d::Element::x
Definition at line 170 of file ComponentNeBem2d.hh.

y

double Garfield::ComponentNeBem2d::Element::y
Coordinates of the element centre (collocation point).
Definition at line 170 of file ComponentNeBem2d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh

## 7.44 Garfield::ComponentNeBem3d::Element Struct Reference

Public Attributes

- std::array< double, 3 > origin
  Local origin.
- double lx
- double lz
- double dA
  Area.
- std::array< std::array< double, 3 >, 3 > dcos
  Direction cosines.
- std::vector< double > xv
  X-coordinates of vertices.
- std::vector< double > yv
  Y-coordinates of vertices.
- std::vector< double > zv
  Z-coordinates of vertices.
- int interface = 0
  Interface type.
- double lambda = 0.
  Ratio of dielectric permittivities.
- std::array< double, 3 > collocationPoint
  Collocation point.
- double bc = 0.
  Boundary condition.

- double assigned = 0.
  Fixed charge density.
- double solution = 0.
  Solution (accumulated charge).

### 7.44.1 Detailed Description

Definition at line 244 of file ComponentNeBem3d.hh.

### 7.44.2 Member Data Documentation

#### assigned

double Garfield::ComponentNeBem3d::Element::assigned = 0.
Fixed charge density.
Definition at line 268 of file ComponentNeBem3d.hh.

#### bc

double Garfield::ComponentNeBem3d::Element::bc = 0.
Boundary condition.
Definition at line 266 of file ComponentNeBem3d.hh.

#### collocationPoint

std::array<double, 3> Garfield::ComponentNeBem3d::Element::collocationPoint
Collocation point.
Definition at line 264 of file ComponentNeBem3d.hh.

#### dA

double Garfield::ComponentNeBem3d::Element::dA
Area.
Definition at line 250 of file ComponentNeBem3d.hh.

#### dcos

std::array<std::array<double, 3>, 3> Garfield::ComponentNeBem3d::Element::dcos
Direction cosines.
Definition at line 252 of file ComponentNeBem3d.hh.

#### interface

int Garfield::ComponentNeBem3d::Element::interface = 0
Interface type.
Definition at line 260 of file ComponentNeBem3d.hh.

#### lambda

double Garfield::ComponentNeBem3d::Element::lambda = 0.
Ratio of dielectric permittivities.
Definition at line 262 of file ComponentNeBem3d.hh.

#### lx

double Garfield::ComponentNeBem3d::Element::lx
Definition at line 247 of file ComponentNeBem3d.hh.

#### lz

double Garfield::ComponentNeBem3d::Element::lz
Definition at line 248 of file ComponentNeBem3d.hh.

origin

std::array<double, 3> Garfield::ComponentNeBem3d::Element::origin
Local origin.
Definition at line 246 of file ComponentNeBem3d.hh.

solution

double Garfield::ComponentNeBem3d::Element::solution = 0.
Solution (accumulated charge).
Definition at line 270 of file ComponentNeBem3d.hh.

xv

std::vector<double> Garfield::ComponentNeBem3d::Element::xv
X-coordinates of vertices.
Definition at line 254 of file ComponentNeBem3d.hh.

yv

std::vector<double> Garfield::ComponentNeBem3d::Element::yv
Y-coordinates of vertices.
Definition at line 256 of file ComponentNeBem3d.hh.

zv

std::vector<double> Garfield::ComponentNeBem3d::Element::zv
Z-coordinates of vertices.
Definition at line 258 of file ComponentNeBem3d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3d.hh

## 7.45 Garfield::ComponentNeBem3dMap::Element Struct Reference

Public Attributes

- double fx
- double fy
- double fz
  Field.
- double v
  Potential.

### 7.45.1 Detailed Description

Definition at line 105 of file ComponentNeBem3dMap.hh.

### 7.45.2 Member Data Documentation

fx

double Garfield::ComponentNeBem3dMap::Element::fx
Definition at line 106 of file ComponentNeBem3dMap.hh.

fy

double Garfield::ComponentNeBem3dMap::Element::fy
Definition at line 106 of file ComponentNeBem3dMap.hh.

fz

double Garfield::ComponentNeBem3dMap::Element::fz
Field.
Definition at line 106 of file ComponentNeBem3dMap.hh.

v

double Garfield::ComponentNeBem3dMap::Element::v
Potential.
Definition at line 107 of file ComponentNeBem3dMap.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3dMap.hh

## 7.46 Garfield::ComponentTcadBase< N >::Element Struct Reference

#include <ComponentTcadBase.hh>

Public Attributes

- unsigned int vertex [nMaxVertices]
- unsigned int type
- unsigned int region
- std::array< float, N > bbMin
- std::array< float, N > bbMax

### 7.46.1 Detailed Description

template<size_t N>
struct Garfield::ComponentTcadBase< N >::Element

Definition at line 203 of file ComponentTcadBase.hh.

### 7.46.2 Member Data Documentation

bbMax

template<size_t N>
std::array<float, N> Garfield::ComponentTcadBase< N >::Element::bbMax
Definition at line 225 of file ComponentTcadBase.hh.

bbMin

template<size_t N>
std::array<float, N> Garfield::ComponentTcadBase< N >::Element::bbMin
Definition at line 224 of file ComponentTcadBase.hh.

region

template<size_t N>
unsigned int Garfield::ComponentTcadBase< N >::Element::region
Definition at line 222 of file ComponentTcadBase.hh.

type

template<size_t N>
unsigned int Garfield::ComponentTcadBase< N >::Element::type
Definition at line 220 of file ComponentTcadBase.hh.

vertex

template<size_t N>
unsigned int Garfield::ComponentTcadBase< N >::Element::vertex[nMaxVertices]
Definition at line 205 of file ComponentTcadBase.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh

## 7.47 Garfield::ComponentVoxel::Element Struct Reference

Public Attributes

- double fx
- double fy
- double fz
  Field.
- double v
  Potential.

### 7.47.1 Detailed Description

Definition at line 119 of file ComponentVoxel.hh.

### 7.47.2 Member Data Documentation

fx

double Garfield::ComponentVoxel::Element::fx
Definition at line 120 of file ComponentVoxel.hh.

fy

double Garfield::ComponentVoxel::Element::fy
Definition at line 120 of file ComponentVoxel.hh.

fz

double Garfield::ComponentVoxel::Element::fz
Field.
Definition at line 120 of file ComponentVoxel.hh.

v

double Garfield::ComponentVoxel::Element::v
Potential.
Definition at line 121 of file ComponentVoxel.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentVoxel.hh

## 7.48 Garfield::AvalancheMC::EndPoint Struct Reference

#include <AvalancheMC.hh>

Public Attributes

- int status
  Status flag.
- std::vector< Point > path
  Drift line.
- size_t weight
  Multiplicity.

### 7.48.1 Detailed Description

Definition at line 78 of file AvalancheMC.hh.

### 7.48.2 Member Data Documentation

path

std::vector<Point> Garfield::AvalancheMC::EndPoint::path
Drift line.
Definition at line 80 of file AvalancheMC.hh.

status

int Garfield::AvalancheMC::EndPoint::status
Status flag.
Definition at line 79 of file AvalancheMC.hh.

weight

size_t Garfield::AvalancheMC::EndPoint::weight
Multiplicity.
Definition at line 81 of file AvalancheMC.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh

## 7.49 Garfield::TrackDegrade::Excitation Struct Reference

#include <TrackDegrade.hh>

Public Attributes

- double x = 0.
- double y = 0.
- double z = 0.
- double t = 0.
- double energy = 0.

### 7.49.1 Detailed Description

Definition at line 27 of file TrackDegrade.hh.

### 7.49.2 Member Data Documentation

energy

double Garfield::TrackDegrade::Excitation::energy = 0.
Definition at line 32 of file TrackDegrade.hh.

t

double Garfield::TrackDegrade::Excitation::t = 0.
Definition at line 31 of file TrackDegrade.hh.

x

double Garfield::TrackDegrade::Excitation::x = 0.
Definition at line 28 of file TrackDegrade.hh.

y

double Garfield::TrackDegrade::Excitation::y = 0.
Definition at line 29 of file TrackDegrade.hh.

z

double Garfield::TrackDegrade::Excitation::z = 0.
Definition at line 30 of file TrackDegrade.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh

## 7.50 Garfield::MediumGas::ExcLevel Struct Reference

#include <MediumGas.hh>

Public Attributes

- std::string label
- double energy
- double prob
- double rms
- double dt

### 7.50.1 Detailed Description

Definition at line 208 of file MediumGas.hh.

### 7.50.2 Member Data Documentation

dt

double Garfield::MediumGas::ExcLevel::dt
Definition at line 213 of file MediumGas.hh.

energy

double Garfield::MediumGas::ExcLevel::energy
Definition at line 210 of file MediumGas.hh.

label

std::string Garfield::MediumGas::ExcLevel::label
Definition at line 209 of file MediumGas.hh.

prob

double Garfield::MediumGas::ExcLevel::prob
Definition at line 211 of file MediumGas.hh.

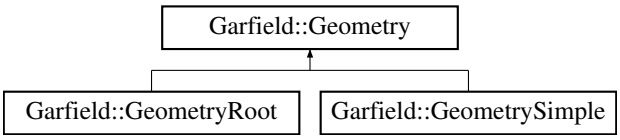
rms

double Garfield::MediumGas::ExcLevel::rms
Definition at line 212 of file MediumGas.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumGas.hh

## 7.51 Garfield::Geometry Class Reference

Abstract base class for geometry classes.
#include <Geometry.hh>
Inheritance diagram for Garfield::Geometry:




Public Member Functions

- Geometry ()=delete
  Default constructor.
- Geometry (const std::string &name)
  Constructor.
- virtual ~Geometry ()
  Destructor.
- virtual Medium * GetMedium (const double x, const double y, const double z, const bool tesselated=false) const =0
  Retrieve the medium at a given point.
- virtual std::size_t GetNumberOfSolids () const
  Return the number of solids in the geometry.
- virtual Solid * GetSolid (const std::size_t) const
  Get a solid from the list.
- virtual Solid * GetSolid (const std::size_t, Medium *&) const
  Get a solid from the list, together with the associated medium.
- virtual bool IsInside (const double x, const double y, const double z, const bool tesselated=false) const =0
  Check if a point is inside the geometry.
- virtual bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax)=0
  Get the bounding box (envelope of the geometry).

Protected Attributes

- std::string m_className = "Geometry"

### 7.51.1 Detailed Description

Abstract base class for geometry classes.
Definition at line 13 of file Geometry.hh.

### 7.51.2 Constructor & Destructor Documentation

Geometry() [1/2]

Garfield::Geometry::Geometry () [delete]
Default constructor.

Geometry() [2/2]

Garfield::Geometry::Geometry (
const std::string & name) [inline]
Constructor.
Definition at line 18 of file Geometry.hh.
00018 : m_className(name) {}

### ~Geometry()

virtual Garfield::Geometry::~Geometry () [inline], [virtual]
Destructor.
Definition at line 20 of file Geometry.hh.
00020 {}

## 7.51.3 Member Function Documentation

### GetBoundingBox()

virtual bool Garfield::Geometry::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [pure virtual]
Get the bounding box (envelope of the geometry).
Implemented in Garfield::GeometryRoot, and Garfield::GeometrySimple.

### GetMedium()

virtual Medium * Garfield::Geometry::GetMedium (
const double x,
const double y,
const double z,
const bool tesselated = false) const [pure virtual]
Retrieve the medium at a given point.
Implemented in Garfield::GeometryRoot, and Garfield::GeometrySimple.

### GetNumberOfSolids()

virtual std::size_t Garfield::Geometry::GetNumberOfSolids () const [inline], [virtual]
Return the number of solids in the geometry.
Reimplemented in Garfield::GeometrySimple.
Definition at line 27 of file Geometry.hh.
00027 { return 0; }

### GetSolid() [1/2]

virtual Solid * Garfield::Geometry::GetSolid (
const std::size_t ) const [inline], [virtual]
Get a solid from the list.
Definition at line 29 of file Geometry.hh.
00029 { return nullptr; }

### GetSolid() [2/2]

virtual Solid * Garfield::Geometry::GetSolid (
const std::size_t ,
Medium *& ) const [inline], [virtual]
Get a solid from the list, together with the associated medium.
Definition at line 31 of file Geometry.hh.
00031 {
00032 return nullptr;
00033 }

### IsInside()

virtual bool Garfield::Geometry::IsInside (
const double x,

const double y,
const double z,
const bool tesselated = false) const [pure virtual]

Check if a point is inside the geometry.
Implemented in Garfield::GeometryRoot, and Garfield::GeometrySimple.

### 7.51.4 Member Data Documentation

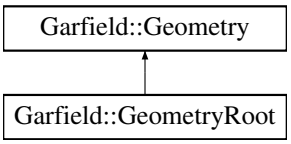
m_className

std::string Garfield::Geometry::m_className = "Geometry" [protected]
Definition at line 43 of file Geometry.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Geometry.hh

## 7.52 Garfield::GeometryRoot Class Reference

Use a geometry defined using the ROOT TGeo package.
#include <GeometryRoot.hh>
Inheritance diagram for Garfield::GeometryRoot:




Public Member Functions

- GeometryRoot ()
  Constructor.
- ~GeometryRoot ()
  Destructor.
- void SetGeometry (TGeoManager *geoman)
  Set the geometry (pointer to ROOT TGeoManager).
- Medium * GetMedium (const double x, const double y, const double z, const bool tesselated=false) const override
  Retrieve the medium at a given point.
- unsigned int GetNumberOfMaterials ()
  Get the number of materials defined in the ROOT geometry.
- TGeoMaterial * GetMaterial (const unsigned int i)
  Get a pointer to the ROOT material with a given index.
- TGeoMaterial * GetMaterial (const char *name)
  Get a pointer to the ROOT material with a given name.
- void SetMedium (const unsigned int imat, Medium *med)
  Associate a ROOT material with a Garfield medium.
- void SetMedium (const char *mat, Medium *med)
  Associate a ROOT material with a Garfield medium.
- bool IsInside (const double x, const double y, const double z, const bool tesselated=false) const override
  Check if a point is inside the geometry.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override
  Get the bounding box (envelope of the geometry).
- void EnableDebugging (const bool on=true)
  Switch debugging and warning messages on/off.

Public Member Functions inherited from Garfield::Geometry

- Geometry ()=delete
  Default constructor.
- Geometry (const std::string &name)
  Constructor.
- virtual ~Geometry ()
  Destructor.
- virtual std::size_t GetNumberOfSolids () const
  Return the number of solids in the geometry.
- virtual Solid * GetSolid (const std::size_t) const
  Get a solid from the list.
- virtual Solid * GetSolid (const std::size_t, Medium *&) const
  Get a solid from the list, together with the associated medium.

Protected Member Functions

- void PrintGeoNotDefined (const std::string &fcn) const

Protected Attributes

- TGeoManager * m_geoManager = nullptr
- std::map< std::string, Medium * > m_materials
- bool m_debug = false

Protected Attributes inherited from Garfield::Geometry

- std::string m_className = "Geometry"

### 7.52.1 Detailed Description

Use a geometry defined using the ROOT TGeo package.
Definition at line 15 of file GeometryRoot.hh.

### 7.52.2 Constructor & Destructor Documentation

GeometryRoot()

Garfield::GeometryRoot::GeometryRoot ()
Constructor.

~GeometryRoot()

Garfield::GeometryRoot::~GeometryRoot () [inline]
Destructor.
Definition at line 20 of file GeometryRoot.hh.

```
00020 {}
```

### 7.52.3 Member Function Documentation

EnableDebugging()

void Garfield::GeometryRoot::EnableDebugging (
const bool on = true) [inline]
Switch debugging and warning messages on/off.
Definition at line 45 of file GeometryRoot.hh.

```
00045 { m_debug = on; }
```

GetBoundingBox()

bool Garfield::GeometryRoot::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [override], [virtual]

Get the bounding box (envelope of the geometry).
Implements Garfield::Geometry.

GetMaterial() [1/2]

TGeoMaterial * Garfield::GeometryRoot::GetMaterial (
const char * name)

Get a pointer to the ROOT material with a given name.

GetMaterial() [2/2]

TGeoMaterial * Garfield::GeometryRoot::GetMaterial (
const unsigned int i)

Get a pointer to the ROOT material with a given index.

GetMedium()

Medium * Garfield::GeometryRoot::GetMedium (
const double x,
const double y,
const double z,
const bool tesselated = false) const [override], [virtual]

Retrieve the medium at a given point.
Implements Garfield::Geometry.

GetNumberOfMaterials()

unsigned int Garfield::GeometryRoot::GetNumberOfMaterials ()

Get the number of materials defined in the ROOT geometry.

IsInside()

bool Garfield::GeometryRoot::IsInside (
const double x,
const double y,
const double z,
const bool tesselated = false) const [override], [virtual]

Check if a point is inside the geometry.
Implements Garfield::Geometry.

PrintGeoNotDefined()

void Garfield::GeometryRoot::PrintGeoNotDefined (
const std::string & fcn) const [protected]

SetGeometry()

void Garfield::GeometryRoot::SetGeometry (
TGeoManager * geoman)

Set the geometry (pointer to ROOT TGeoManager).

SetMedium() [1/2]

void Garfield::GeometryRoot::SetMedium (
const char * mat,
Medium * med)
Associate a ROOT material with a Garfield medium.

SetMedium() [2/2]

void Garfield::GeometryRoot::SetMedium (
const unsigned int imat,
Medium * med)
Associate a ROOT material with a Garfield medium.

### 7.52.4 Member Data Documentation

m_debug

bool Garfield::GeometryRoot::m_debug = false [protected]
Definition at line 55 of file GeometryRoot.hh.

m_geoManager

TGeoManager* Garfield::GeometryRoot::m_geoManager = nullptr [protected]
Definition at line 49 of file GeometryRoot.hh.

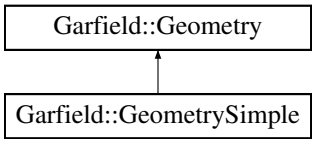
m_materials

std::map<std::string, Medium*> Garfield::GeometryRoot::m_materials [protected]
Definition at line 52 of file GeometryRoot.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/GeometryRoot.hh

## 7.53 Garfield::GeometrySimple Class Reference

"Native" geometry, using simple shapes.
#include <GeometrySimple.hh>
Inheritance diagram for Garfield::GeometrySimple:




Public Member Functions

- GeometrySimple ()
  Constructor.
- virtual ~GeometrySimple ()
  Destructor.
- Medium * GetMedium (const double x, const double y, const double z, const bool tesselated=false) const override
  Retrieve the medium at a given point.
- size_t GetNumberOfSolids () const override
  Return the number of solids in the geometry.
- Solid * GetSolid (const size_t i) const override
- Solid * GetSolid (const size_t i, Medium *&medium) const override

- void AddSolid (Solid *s, Medium *m)

  Add a solid to the geometry, together with the medium inside.
- Solid * GetSolid (const double x, const double y, const double z, const bool tesselated=false) const

  Get the solid at a given location (x, y, z).
- void SetMedium (Medium *medium)

  Set a background medium.
- void Clear ()

  Reset the geometry.
- void PrintSolids ()

  Print a summary of the solids present in the geometry.
- bool IsInside (const double x, const double y, const double z, const bool tesselated=false) const override

  Check if a point is inside the geometry.
- bool IsInBoundingBox (const double x, const double y, const double z) const

  Determine whether a point is inside the envelope of the geometry.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) override

  Get the bounding box (envelope of the geometry).
- void EnableDebugging (const bool on=true)

  Switch on/off debugging and warning messages.

Public Member Functions inherited from Garfield::Geometry

- Geometry ()=delete

  Default constructor.
- Geometry (const std::string &name)

  Constructor.
- virtual ~Geometry ()

  Destructor.
- virtual Solid * GetSolid (const std::size_t) const

  Get a solid from the list.
- virtual Solid * GetSolid (const std::size_t, Medium *&) const

  Get a solid from the list, together with the associated medium.

Protected Attributes

- std::vector< std::pair< Solid *, Medium * > > m_solids

  List of solids and associated media.
- Medium * m_medium = nullptr

  Background medium.
- bool m_hasBoundingBox = false
- std::array< double, 3 > m_bbMin = {{0., 0., 0.}}
- std::array< double, 3 > m_bbMax = {{0., 0., 0.}}
- bool m_debug = false

  Switch on/off debugging messages.

Protected Attributes inherited from Garfield::Geometry

- std::string m_className = "Geometry"

### 7.53.1 Detailed Description

"Native" geometry, using simple shapes.
Definition at line 13 of file GeometrySimple.hh.

### 7.53.2 Constructor & Destructor Documentation

#### GeometrySimple()

Garfield::GeometrySimple::GeometrySimple ()
Constructor.

#### ~GeometrySimple()

virtual Garfield::GeometrySimple::~GeometrySimple () [inline], [virtual]
Destructor.
Definition at line 18 of file GeometrySimple.hh.
00018 {}

### 7.53.3 Member Function Documentation

#### AddSolid()

void Garfield::GeometrySimple::AddSolid (
Solid * s,
Medium * m)
Add a solid to the geometry, together with the medium inside.

#### Clear()

void Garfield::GeometrySimple::Clear ()
Reset the geometry.

#### EnableDebugging()

void Garfield::GeometrySimple::EnableDebugging (
const bool on = true) [inline]
Switch on/off debugging and warning messages.
Definition at line 57 of file GeometrySimple.hh.
00057 { m_debug = on; }

#### GetBoundingBox()

bool Garfield::GeometrySimple::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) [inline], [override], [virtual]
Get the bounding box (envelope of the geometry).
Implements Garfield::Geometry.
Definition at line 45 of file GeometrySimple.hh.

```
00046                                                 {
00047     xmin = m_bbMin[0];
00048     ymin = m_bbMin[1];
00049     zmin = m_bbMin[2];
00050     xmax = m_bbMax[0];
00051     ymax = m_bbMax[1];
00052     zmax = m_bbMax[2];
00053     return true;
00054   }
```

#### GetMedium()

Medium * Garfield::GeometrySimple::GetMedium (
const double x,
const double y,

const double z,
const bool tesselated = false) const [override], [virtual]

Retrieve the medium at a given point.
Implements Garfield::Geometry.

### GetNumberOfSolids()

size_t Garfield::GeometrySimple::GetNumberOfSolids () const [inline], [override], [virtual]

Return the number of solids in the geometry.
Reimplemented from Garfield::Geometry.
Definition at line 23 of file GeometrySimple.hh.

```
00023 { return m_solids.size(); }
```

### GetSolid() [1/3]

Solid * Garfield::GeometrySimple::GetSolid (
const double x,
const double y,
const double z,
const bool tesselated = false) const

Get the solid at a given location (x, y, z).

### GetSolid() [2/3]

Solid * Garfield::GeometrySimple::GetSolid (
const size_t i) const [override]

### GetSolid() [3/3]

Solid * Garfield::GeometrySimple::GetSolid (
const size_t i,
Medium *& medium) const [override]

### IsInBoundingBox()

bool Garfield::GeometrySimple::IsInBoundingBox (
const double x,
const double y,
const double z) const

Determine whether a point is inside the envelope of the geometry.

### IsInside()

bool Garfield::GeometrySimple::IsInside (
const double x,
const double y,
const double z,
const bool tesselated = false) const [override], [virtual]

Check if a point is inside the geometry.
Implements Garfield::Geometry.

### PrintSolids()

void Garfield::GeometrySimple::PrintSolids ()

Print a summary of the solids present in the geometry.

### SetMedium()

void Garfield::GeometrySimple::SetMedium (
Medium * medium) [inline]

Set a background medium.

Definition at line 34 of file GeometrySimple.hh.
00034 { m_medium = medium; }

### 7.53.4 Member Data Documentation

#### m_bbMax

std::array<double, 3> Garfield::GeometrySimple::m_bbMax = {{0., 0., 0.}} [protected]
Definition at line 68 of file GeometrySimple.hh.
00068 {{0., 0., 0.}};

#### m_bbMin

std::array<double, 3> Garfield::GeometrySimple::m_bbMin = {{0., 0., 0.}} [protected]
Definition at line 67 of file GeometrySimple.hh.
00067 {{0., 0., 0.}};

#### m_debug

bool Garfield::GeometrySimple::m_debug = false [protected]
Switch on/off debugging messages.
Definition at line 71 of file GeometrySimple.hh.

#### m_hasBoundingBox

bool Garfield::GeometrySimple::m_hasBoundingBox = false [protected]
Definition at line 66 of file GeometrySimple.hh.

#### m_medium

Medium* Garfield::GeometrySimple::m_medium = nullptr [protected]
Background medium.
Definition at line 63 of file GeometrySimple.hh.

#### m_solids

std::vector<std::pair<Solid*, Medium*> > Garfield::GeometrySimple::m_solids [protected]
List of solids and associated media.
Definition at line 61 of file GeometrySimple.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/GeometrySimple.hh

## 7.54 Garfield::AvalancheGridSpaceCharge::GridNode Struct Reference

Public Attributes

- long nElectron = 0
  electrons on node
- double nPosIon = 0
  pos ion on node (smeared values allowed)
- double nNegIon = 0
  neg ion on node (smeared values allowed)
- long nElectronHolder = 0
  at t+dt
- double nPosIonHolder = 0
  at t+dt
- double nNegIonHolder = 0
  at t+dt
- double townsend = 0

townsend at this node 1/cm

- double attachment = 0

  attachment at this node 1/cm
- double velocity = 0.

  Magnitude of velocity of the node (not negative) cm/ns.
- double dSigmaL = 0

  Diffusion along E.
- double dSigmaT = 0

  Diffusion transverse to E (radial, phi dir is net 0).
- double Wv = 0

  flux drift cm/ns
- double Wr = 0

  bulk drift cm/ns
- double townsendPT = 0

  Ionization rate from TOF experiment 1/ns -> 1/cm.
- double attachmentPT = 0

  Attachment rate from TOF experiment 1/ns -> 1/cm.
- double eFieldR = 0

  Space-charge electric field in R direction (can be negative)
- double eFieldZ = 0

  Space-charge electric field in Z direction (can be negative)
- double time = 0.

  Node clock.
- bool anode = false

  init the anode
- int layerIndex = 0

  LayerIndex in ParallelPlate convention != gas gap index.
- int gasGapIndex = 0

  Gas gap index: -1 if not gas gap; starts with 0, 1, ...
- bool isGasGap = true

### 7.54.1 Detailed Description

Definition at line 134 of file AvalancheGridSpaceCharge.hh.

### 7.54.2 Member Data Documentation

anode

bool Garfield::AvalancheGridSpaceCharge::GridNode::anode = false
init the anode
Definition at line 165 of file AvalancheGridSpaceCharge.hh.

attachment

double Garfield::AvalancheGridSpaceCharge::GridNode::attachment = 0
attachment at this node 1/cm
Definition at line 144 of file AvalancheGridSpaceCharge.hh.

attachmentPT

double Garfield::AvalancheGridSpaceCharge::GridNode::attachmentPT = 0
Attachment rate from TOF experiment 1/ns -> 1/cm.
Definition at line 157 of file AvalancheGridSpaceCharge.hh.

dSigmaL

double Garfield::AvalancheGridSpaceCharge::GridNode::dSigmaL = 0
Diffusion along E.
Definition at line 148 of file AvalancheGridSpaceCharge.hh.

dSigmaT

double Garfield::AvalancheGridSpaceCharge::GridNode::dSigmaT = 0
Diffusion transverse to E (radial, phi dir is net 0).
Definition at line 150 of file AvalancheGridSpaceCharge.hh.

eFieldR

double Garfield::AvalancheGridSpaceCharge::GridNode::eFieldR = 0
Space-charge electric field in R direction (can be negative)
Definition at line 159 of file AvalancheGridSpaceCharge.hh.

eFieldZ

double Garfield::AvalancheGridSpaceCharge::GridNode::eFieldZ = 0
Space-charge electric field in Z direction (can be negative)
Definition at line 161 of file AvalancheGridSpaceCharge.hh.

gasGapIndex

int Garfield::AvalancheGridSpaceCharge::GridNode::gasGapIndex = 0
Gas gap index: -1 if not gas gap; starts with 0, 1, ...
Definition at line 169 of file AvalancheGridSpaceCharge.hh.

isGasGap

bool Garfield::AvalancheGridSpaceCharge::GridNode::isGasGap = true
Definition at line 170 of file AvalancheGridSpaceCharge.hh.

layerIndex

int Garfield::AvalancheGridSpaceCharge::GridNode::layerIndex = 0
LayerIndex in ParallelPlate convention != gas gap index.
Definition at line 167 of file AvalancheGridSpaceCharge.hh.

nElectron

long Garfield::AvalancheGridSpaceCharge::GridNode::nElectron = 0
electrons on node
Definition at line 135 of file AvalancheGridSpaceCharge.hh.

nElectronHolder

long Garfield::AvalancheGridSpaceCharge::GridNode::nElectronHolder = 0
at t+dt
Definition at line 139 of file AvalancheGridSpaceCharge.hh.

nNegIon

double Garfield::AvalancheGridSpaceCharge::GridNode::nNegIon = 0
neg ion on node (smeared values allowed)
Definition at line 137 of file AvalancheGridSpaceCharge.hh.

nNegIonHolder

double Garfield::AvalancheGridSpaceCharge::GridNode::nNegIonHolder = 0
at t+dt
Definition at line 141 of file AvalancheGridSpaceCharge.hh.

nPosIon

double Garfield::AvalancheGridSpaceCharge::GridNode::nPosIon = 0
pos ion on node (smeared values allowed)
Definition at line 136 of file AvalancheGridSpaceCharge.hh.

nPosIonHolder

double Garfield::AvalancheGridSpaceCharge::GridNode::nPosIonHolder = 0
at t+dt
Definition at line 140 of file AvalancheGridSpaceCharge.hh.

time

double Garfield::AvalancheGridSpaceCharge::GridNode::time = 0.
Node clock.
Definition at line 163 of file AvalancheGridSpaceCharge.hh.

townsend

double Garfield::AvalancheGridSpaceCharge::GridNode::townsend = 0
townsend at this node 1/cm
Definition at line 143 of file AvalancheGridSpaceCharge.hh.

townsendPT

double Garfield::AvalancheGridSpaceCharge::GridNode::townsendPT = 0
Ionization rate from TOF experiment 1/ns -> 1/cm.
Definition at line 155 of file AvalancheGridSpaceCharge.hh.

velocity

double Garfield::AvalancheGridSpaceCharge::GridNode::velocity = 0.
Magnitude of velocity of the node (not negative) cm/ns.
Definition at line 146 of file AvalancheGridSpaceCharge.hh.

Wr

double Garfield::AvalancheGridSpaceCharge::GridNode::Wr = 0
bulk drift cm/ns
Definition at line 153 of file AvalancheGridSpaceCharge.hh.

Wv

double Garfield::AvalancheGridSpaceCharge::GridNode::Wv = 0
flux drift cm/ns
Definition at line 152 of file AvalancheGridSpaceCharge.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheGridSpaceCharge.hh

## 7.55 Garfield::MediumGas::IonLevel Struct Reference

#include <MediumGas.hh>

Public Attributes

- std::string label
- double energy

### 7.55.1 Detailed Description

Definition at line 217 of file MediumGas.hh.

### 7.55.2 Member Data Documentation

energy

double Garfield::MediumGas::IonLevel::energy
Definition at line 219 of file MediumGas.hh.

label

std::string Garfield::MediumGas::IonLevel::label
Definition at line 218 of file MediumGas.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumGas.hh

## 7.56 Garfield::KDTree Class Reference

Main k-d tree class.
#include <KDTree.hh>

Public Member Functions

- KDTree ()=delete
- KDTree (KDTreeArray &data_in)

  Constructor.
- ~KDTree ()

  Destructor.
- void n_nearest (const std::vector< double > &qv, const unsigned int nn, std::vector< KDTreeResult > &result) const

  Search for nn nearest neighbours around a point.
- void n_nearest_around_point (const unsigned int idx, const unsigned int ndecorrel, const unsigned int nn, std::vector< KDTreeResult > &result) const

  Search for nn nearest neighbours around a node of the input data, excluding neighbors within a decorrelation interval.
- void r_nearest (const std::vector< double > &qv, const double r2, std::vector< KDTreeResult > &result) const

  Search for all neighbors in a ball of size r2.
- void r_nearest_around_point (const unsigned int idx, const unsigned int ndecorrel, const double r2, std::vector< KDTreeResult > &result) const

  Like r_nearest, but around an existing point, with decorrelation interval.

Public Attributes

- const KDTreeArray & m_data
- std::size_t m_dim = 3
- bool sort_results = false

Private Member Functions

- KDTreeNode * build_tree_for_range (int l, int u, KDTreeNode *parent)
- int select_on_coordinate_value (int c, double alpha, int l, int u)
- std::array< double, 2 > spread_in_coordinate (const int c, const int l, const int u) const

Private Attributes

- KDTreeNode * m_root = nullptr
- std::vector< std::size_t > m_ind

Static Private Attributes

- static constexpr int bucketsize = 12

Friends

- class KDTreeNode

### 7.56.1 Detailed Description

Main k-d tree class.
Fast search of points in k-dimensional Euclidean space.
Definition at line 33 of file KDTree.hh.

### 7.56.2 Constructor & Destructor Documentation

KDTree() [1/2]

Garfield::KDTree::KDTree () [delete]

KDTree() [2/2]

Garfield::KDTree::KDTree (
KDTreeArray & data_in)
Constructor.

~KDTree()

Garfield::KDTree::~KDTree ()
Destructor.

### 7.56.3 Member Function Documentation

build_tree_for_range()

KDTreeNode * Garfield::KDTree::build_tree_for_range (
int l,
int u,
KDTreeNode * parent) [private]

n_nearest()

void Garfield::KDTree::n_nearest (
const std::vector< double > & qv,
const unsigned int nn,
std::vector< KDTreeResult > & result) const
Search for nn nearest neighbours around a point.

Parameters

| | |
|---|---|
| qv | input point |
| nn | number of nearest neighbours |
| result | indices and distances of the nearest neighbours |

### n_nearest_around_point()

```
void Garfield::KDTree::n_nearest_around_point (
        const unsigned int idx,
        const unsigned int ndecorrel,
        const unsigned int nn,
        std::vector< KDTreeResult > & result) const
```

Search for nn nearest neighbours around a node of the input data, excluding neighbors within a decorrelation interval.

Parameters

| | |
|---|---|
| idx | index of the input point |
| ndecorrel | decorrelation interval |
| nn | number of nearest neighbours |
| result | indices and distances of the nearest neighbours |

### r_nearest()

```
void Garfield::KDTree::r_nearest (
        const std::vector< double > & qv,
        const double r2,
        std::vector< KDTreeResult > & result) const
```

Search for all neighbors in a ball of size r2.

Parameters

| | |
|---|---|
| qv | input point |
| r2 | ball size (square Euclidean distance) |
| result | indices and distances of the nearest neighbours |

### r_nearest_around_point()

```
void Garfield::KDTree::r_nearest_around_point (
        const unsigned int idx,
        const unsigned int ndecorrel,
        const double r2,
        std::vector< KDTreeResult > & result) const
```

Like r_nearest, but around an existing point, with decorrelation interval.

### select_on_coordinate_value()

```
int Garfield::KDTree::select_on_coordinate_value (
        int c,
        double alpha,
        int l,
        int u)    [private]
```

### spread_in_coordinate()

```
std::array< double, 2 > Garfield::KDTree::spread_in_coordinate (
        const int c,
        const int l,
        const int u) const    [private]
```

### 7.56.4 Friends And Related Symbol Documentation

KDTreeNode

friend class KDTreeNode [friend]
Definition at line 82 of file KDTree.hh.

### 7.56.5 Member Data Documentation

bucketsize

int Garfield::KDTree::bucketsize = 12 [static], [constexpr], [private]
Definition at line 91 of file KDTree.hh.

m_data

const KDTreeArray& Garfield::KDTree::m_data
Definition at line 36 of file KDTree.hh.

m_dim

std::size_t Garfield::KDTree::m_dim = 3
Definition at line 38 of file KDTree.hh.

m_ind

std::vector<std::size_t> Garfield::KDTree::m_ind [private]
Definition at line 89 of file KDTree.hh.

m_root

KDTreeNode* Garfield::KDTree::m_root = nullptr [private]
Definition at line 85 of file KDTree.hh.

sort_results

bool Garfield::KDTree::sort_results = false
Definition at line 39 of file KDTree.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/KDTree.hh

## 7.57 Garfield::KDTreeNode Class Reference

A node in the k-d tree.
#include <KDTree.hh>

Public Member Functions

- KDTreeNode (int dim)
  Constructor.
- ~KDTreeNode ()
  Destructor.

Private Member Functions

- void search_n (const int idx0, const int nd, const unsigned int nn, double &r2, const std::vector< double > &qv, const KDTree &tree, std::priority_queue< KDTreeResult > &res) const
- void search_r (const int idx0, const int nd, const double r2, const std::vector< double > &qv, const KDTree &tree, std::vector< KDTreeResult > &res) const
- bool box_in_search_range (const double r2, const std::vector< double > &qv) const

- void process_terminal_node_n (const int idx0, const int nd, const unsigned int nn, double &r2, const std::vector< double > &qv, const KDTree &tree, std::priority_queue< KDTreeResult > &res) const
- void process_terminal_node_r (const int idx0, const int nd, const double r2, const std::vector< double > &qv, const KDTree &tree, std::vector< KDTreeResult > &res) const

Private Attributes

- std::size_t cut_dim = 0
- double cut_val = 0.
- double cut_val_left = 0.
- double cut_val_right = 0.
- int m_l = 0
- int m_u = 0
- std::vector< std::array< double, 2 > > box
- KDTreeNode * left = nullptr
- KDTreeNode * right = nullptr

Friends

- class KDTree

### 7.57.1 Detailed Description

A node in the k-d tree.
Definition at line 102 of file KDTree.hh.

### 7.57.2 Constructor & Destructor Documentation

KDTreeNode()

Garfield::KDTreeNode::KDTreeNode (
int dim)
Constructor.

~KDTreeNode()

Garfield::KDTreeNode::~KDTreeNode ()
Destructor.

### 7.57.3 Member Function Documentation

box_in_search_range()

bool Garfield::KDTreeNode::box_in_search_range (
const double r2,
const std::vector< double > & qv) const [private]

process_terminal_node_n()

void Garfield::KDTreeNode::process_terminal_node_n (
const int idx0,
const int nd,
const unsigned int nn,
double & r2,
const std::vector< double > & qv,
const KDTree & tree,
std::priority_queue< KDTreeResult > & res) const [private]

process_terminal_node_r()

void Garfield::KDTreeNode::process_terminal_node_r (
const int idx0,
const int nd,
const double r2,
const std::vector< double > & qv,
const KDTree & tree,
std::vector< KDTreeResult > & res) const [private]

search_n()

void Garfield::KDTreeNode::search_n (
const int idx0,
const int nd,
const unsigned int nn,
double & r2,
const std::vector< double > & qv,
const KDTree & tree,
std::priority_queue< KDTreeResult > & res) const [private]

search_r()

void Garfield::KDTreeNode::search_r (
const int idx0,
const int nd,
const double r2,
const std::vector< double > & qv,
const KDTree & tree,
std::vector< KDTreeResult > & res) const [private]

### 7.57.4 Friends And Related Symbol Documentation

KDTree

friend class KDTree [friend]
Definition at line 110 of file KDTree.hh.

### 7.57.5 Member Data Documentation

box

std::vector<std::array<double, 2> > Garfield::KDTreeNode::box [private]
Definition at line 122 of file KDTree.hh.

cut_dim

std::size_t Garfield::KDTreeNode::cut_dim = 0 [private]
Definition at line 113 of file KDTree.hh.

cut_val

double Garfield::KDTreeNode::cut_val = 0. [private]
Definition at line 115 of file KDTree.hh.

cut_val_left

double Garfield::KDTreeNode::cut_val_left = 0. [private]
Definition at line 116 of file KDTree.hh.

cut_val_right

double Garfield::KDTreeNode::cut_val_right = 0. [private]
Definition at line 117 of file KDTree.hh.

left

KDTreeNode* Garfield::KDTreeNode::left = nullptr [private]
Definition at line 125 of file KDTree.hh.

m_l

int Garfield::KDTreeNode::m_l = 0 [private]
Definition at line 119 of file KDTree.hh.

m_u

int Garfield::KDTreeNode::m_u = 0 [private]
Definition at line 120 of file KDTree.hh.

right

KDTreeNode* Garfield::KDTreeNode::right = nullptr [private]
Definition at line 126 of file KDTree.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/KDTree.hh

## 7.58 Garfield::KDTreeResult Struct Reference

Search result.
#include <KDTree.hh>

Public Attributes

- double dis
- std::size_t idx

### 7.58.1 Detailed Description

Search result.
Definition at line 25 of file KDTree.hh.

### 7.58.2 Member Data Documentation

dis

double Garfield::KDTreeResult::dis
Definition at line 26 of file KDTree.hh.

idx

std::size_t Garfield::KDTreeResult::idx
Definition at line 27 of file KDTree.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/KDTree.hh

## 7.59 Garfield::ComponentFieldMap::Material Struct Reference

#include <ComponentFieldMap.hh>

Public Attributes

- double eps
- double ohm
- bool driftmedium
- Medium * medium

### 7.59.1 Detailed Description

Definition at line 240 of file ComponentFieldMap.hh.

### 7.59.2 Member Data Documentation

driftmedium

bool Garfield::ComponentFieldMap::Material::driftmedium
Definition at line 245 of file ComponentFieldMap.hh.

eps

double Garfield::ComponentFieldMap::Material::eps
Definition at line 242 of file ComponentFieldMap.hh.

medium

Medium* Garfield::ComponentFieldMap::Material::medium
Definition at line 250 of file ComponentFieldMap.hh.

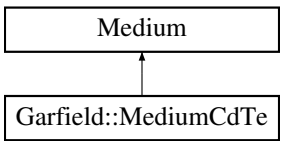
ohm

double Garfield::ComponentFieldMap::Material::ohm
Definition at line 244 of file ComponentFieldMap.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh

## 7.60 Garfield::MediumCdTe Class Reference

Cadmium-Telluride.
#include <MediumCdTe.hh>
Inheritance diagram for Garfield::MediumCdTe:




Public Member Functions

- MediumCdTe ()
  Constructor.
- virtual ~MediumCdTe ()
  Destructor.
- bool IsSemiconductor () const override
- void GetComponent (const unsigned int i, std::string &label, double &f) override
- bool ElectronVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool ElectronTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override

- bool ElectronAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double ElectronMobility () override
- bool HoleVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool HoleTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool HoleAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double HoleMobility () override
- void SetLowFieldMobility (const double mue, const double muh)
- void UnsetLowFieldMobility ()

Private Member Functions

- void UpdateTransportParameters ()

Private Attributes

- double m_eMobility = 1.05e-6
- double m_hMobility = 0.1e-6
- double m_eHallFactor = 1.15
- double m_hHallFactor = 0.7
- bool m_userMobility = false

### 7.60.1 Detailed Description

Cadmium-Telluride.
Definition at line 10 of file MediumCdTe.hh.

### 7.60.2 Constructor & Destructor Documentation

MediumCdTe()

Garfield::MediumCdTe::MediumCdTe ()
Constructor.

~MediumCdTe()

virtual Garfield::MediumCdTe::~MediumCdTe () [inline], [virtual]
Destructor.
Definition at line 15 of file MediumCdTe.hh.
00015 {}

### 7.60.3 Member Function Documentation

ElectronAttachment()

bool Garfield::MediumCdTe::ElectronAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

ElectronMobility()

double Garfield::MediumCdTe::ElectronMobility () [inline], [override]
Definition at line 32 of file MediumCdTe.hh.
00032 { return m_eMobility; }

### ElectronTownsend()

```
bool Garfield::MediumCdTe::ElectronTownsend (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & alpha)   [override]
```

### ElectronVelocity()

```
bool Garfield::MediumCdTe::ElectronVelocity (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & vx,
                double & vy,
                double & vz)   [override]
```

### GetComponent()

```
void Garfield::MediumCdTe::GetComponent (
                const unsigned int i,
                std::string & label,
                double & f)   [override]
```

### HoleAttachment()

```
bool Garfield::MediumCdTe::HoleAttachment (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & eta)   [override]
```

### HoleMobility()

double Garfield::MediumCdTe::HoleMobility ()  [inline], [override]

Definition at line 43 of file MediumCdTe.hh.

```
00043 { return m_hMobility; }
```

### HoleTownsend()

```
bool Garfield::MediumCdTe::HoleTownsend (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & alpha)   [override]
```

### HoleVelocity()

```
bool Garfield::MediumCdTe::HoleVelocity (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & vx,
                double & vy,
                double & vz)    [override]
```

### IsSemiconductor()

```
bool Garfield::MediumCdTe::IsSemiconductor () const    [inline], [override]
```

Definition at line 17 of file MediumCdTe.hh.

```
00017 { return true; }
```

### SetLowFieldMobility()

```
void Garfield::MediumCdTe::SetLowFieldMobility (
                const double mue,
                const double muh)
```

### UnsetLowFieldMobility()

```
void Garfield::MediumCdTe::UnsetLowFieldMobility ()
```

### UpdateTransportParameters()

```
void Garfield::MediumCdTe::UpdateTransportParameters ()    [private]
```

## 7.60.4 Member Data Documentation

### m_eHallFactor

```
double Garfield::MediumCdTe::m_eHallFactor = 1.15    [private]
```

Definition at line 55 of file MediumCdTe.hh.

### m_eMobility

```
double Garfield::MediumCdTe::m_eMobility = 1.05e-6    [private]
```

Definition at line 52 of file MediumCdTe.hh.

### m_hHallFactor

```
double Garfield::MediumCdTe::m_hHallFactor = 0.7    [private]
```

Definition at line 56 of file MediumCdTe.hh.

### m_hMobility

```
double Garfield::MediumCdTe::m_hMobility = 0.1e-6    [private]
```

Definition at line 53 of file MediumCdTe.hh.

### m_userMobility

```
bool Garfield::MediumCdTe::m_userMobility = false    [private]
```

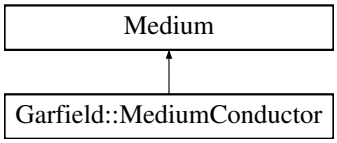
Definition at line 58 of file MediumCdTe.hh.

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumCdTe.hh

## 7.61 Garfield::MediumConductor Class Reference

Conducting medium.
#include <MediumConductor.hh>
Inheritance diagram for Garfield::MediumConductor:




Public Member Functions

- MediumConductor ()
  Constructor.
- virtual ~MediumConductor ()
  Destructor.
- bool IsConductor () const override
- void EnableDrift (const bool) override
- void EnablePrimaryIonisation (const bool) override

### 7.61.1 Detailed Description

Conducting medium.
Definition at line 10 of file MediumConductor.hh.

### 7.61.2 Constructor & Destructor Documentation

MediumConductor()

Garfield::MediumConductor::MediumConductor () [inline]
Constructor.
Definition at line 13 of file MediumConductor.hh.

```
00013                   : Medium() {
00014     m_className = "MediumConductor";
00015     m_name = "Conductor";
00016   }
```

~MediumConductor()

virtual Garfield::MediumConductor::~MediumConductor () [inline], [virtual]
Destructor.
Definition at line 18 of file MediumConductor.hh.

```
00018 {}
```

### 7.61.3 Member Function Documentation

EnableDrift()

void Garfield::MediumConductor::EnableDrift (
const bool ) [inline], [override]
Definition at line 21 of file MediumConductor.hh.

```
00021 {}
```

EnablePrimaryIonisation()

void Garfield::MediumConductor::EnablePrimaryIonisation (
const bool ) [inline], [override]
Definition at line 22 of file MediumConductor.hh.

```
00022 {}
```

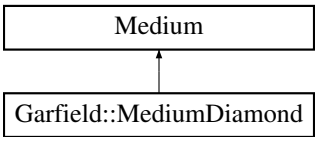
IsConductor()

bool Garfield::MediumConductor::IsConductor () const [inline], [override]
Definition at line 20 of file MediumConductor.hh.
00020 { return true; }
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumConductor.hh

## 7.62 Garfield::MediumDiamond Class Reference

Diamond.
#include <MediumDiamond.hh>
Inheritance diagram for Garfield::MediumDiamond:




Public Member Functions

- MediumDiamond ()

  Constructor.
- virtual ~MediumDiamond ()

  Destructor.
- bool IsSemiconductor () const override
- void GetComponent (const unsigned int i, std::string &label, double &f) override
- bool ElectronVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool ElectronTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool ElectronAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double ElectronMobility () override
- bool HoleVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool HoleTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool HoleAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double HoleMobility () override
- void SetLowFieldMobility (const double mue, const double muh)
- void UnsetLowFieldMobility ()
- void SetSaturationVelocity (const double vsate, const double vsath)
- void UnsetSaturationVelocity ()

Private Member Functions

- void UpdateTransportParameters ()

Private Attributes

- std::mutex m_mutex
- double m_eMobility = 4.551e-6
- double m_hMobility = 2.750e-6
- double m_eHallFactor = 1.

- double m_hHallFactor = 1.
- double m_eSatVel = 2.6e-2
- double m_hSatVel = 1.6e-2
- bool m_userMobility = false

### 7.62.1 Detailed Description

Diamond.
Definition at line 12 of file MediumDiamond.hh.

### 7.62.2 Constructor & Destructor Documentation

#### MediumDiamond()

Garfield::MediumDiamond::MediumDiamond ()
Constructor.

#### ~MediumDiamond()

virtual Garfield::MediumDiamond::~MediumDiamond () [inline], [virtual]
Destructor.
Definition at line 17 of file MediumDiamond.hh.
00017 {}

### 7.62.3 Member Function Documentation

#### ElectronAttachment()

bool Garfield::MediumDiamond::ElectronAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

#### ElectronMobility()

double Garfield::MediumDiamond::ElectronMobility () [inline], [override]
Definition at line 34 of file MediumDiamond.hh.
00034 { return m_eMobility; }

#### ElectronTownsend()

bool Garfield::MediumDiamond::ElectronTownsend (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & alpha) [override]

#### ElectronVelocity()

bool Garfield::MediumDiamond::ElectronVelocity (
const double ex,
const double ey,
const double ez,
const double bx,

```
                const double by,
                const double bz,
                double & vx,
                double & vy,
                double & vz)    [override]
```

### GetComponent()

```
void Garfield::MediumDiamond::GetComponent (
                const unsigned int i,
                std::string & label,
                double & f)    [override]
```

### HoleAttachment()

```
bool Garfield::MediumDiamond::HoleAttachment (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & eta)    [override]
```

### HoleMobility()

```
double Garfield::MediumDiamond::HoleMobility ()    [inline], [override]
```

Definition at line 45 of file MediumDiamond.hh.

```
00045 { return m_hMobility; }
```

### HoleTownsend()

```
bool Garfield::MediumDiamond::HoleTownsend (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & alpha)    [override]
```

### HoleVelocity()

```
bool Garfield::MediumDiamond::HoleVelocity (
                const double ex,
                const double ey,
                const double ez,
                const double bx,
                const double by,
                const double bz,
                double & vx,
                double & vy,
                double & vz)    [override]
```

### IsSemiconductor()

```
bool Garfield::MediumDiamond::IsSemiconductor () const    [inline], [override]
```

Definition at line 19 of file MediumDiamond.hh.

```
00019 { return true; }
```

SetLowFieldMobility()

void Garfield::MediumDiamond::SetLowFieldMobility (
const double mue,
const double muh)

SetSaturationVelocity()

void Garfield::MediumDiamond::SetSaturationVelocity (
const double vsate,
const double vsath)

UnsetLowFieldMobility()

void Garfield::MediumDiamond::UnsetLowFieldMobility ()

UnsetSaturationVelocity()

void Garfield::MediumDiamond::UnsetSaturationVelocity ()

UpdateTransportParameters()

void Garfield::MediumDiamond::UpdateTransportParameters () [private]

### 7.62.4 Member Data Documentation

m_eHallFactor

double Garfield::MediumDiamond::m_eHallFactor = 1. [private]
Definition at line 60 of file MediumDiamond.hh.

m_eMobility

double Garfield::MediumDiamond::m_eMobility = 4.551e-6 [private]
Definition at line 57 of file MediumDiamond.hh.

m_eSatVel

double Garfield::MediumDiamond::m_eSatVel = 2.6e-2 [private]
Definition at line 63 of file MediumDiamond.hh.

m_hHallFactor

double Garfield::MediumDiamond::m_hHallFactor = 1. [private]
Definition at line 61 of file MediumDiamond.hh.

m_hMobility

double Garfield::MediumDiamond::m_hMobility = 2.750e-6 [private]
Definition at line 58 of file MediumDiamond.hh.

m_hSatVel

double Garfield::MediumDiamond::m_hSatVel = 1.6e-2 [private]
Definition at line 64 of file MediumDiamond.hh.

m_mutex

std::mutex Garfield::MediumDiamond::m_mutex [private]
Definition at line 54 of file MediumDiamond.hh.

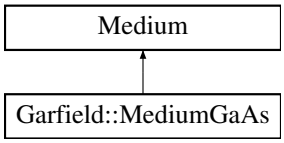
m_userMobility

bool Garfield::MediumDiamond::m_userMobility = false [private]
Definition at line 66 of file MediumDiamond.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumDiamond.hh

## 7.63 Garfield::MediumGaAs Class Reference

Gallium-Arsenide.
#include <MediumGaAs.hh>
Inheritance diagram for Garfield::MediumGaAs:




Public Member Functions

- MediumGaAs ()
  Constructor.
- virtual ~MediumGaAs ()
  Destructor.
- bool IsSemiconductor () const override
- void GetComponent (const unsigned int i, std::string &label, double &f) override
- bool ElectronVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool ElectronTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool ElectronAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double ElectronMobility () override
- bool HoleVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool HoleTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool HoleAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double HoleMobility () override
- void SetLowFieldMobility (const double mue, const double muh)
- void UnsetLowFieldMobility ()

Private Member Functions

- void UpdateTransportParameters ()

Private Attributes

- double m_eMobility = 8.0e-6
- double m_hMobility = 0.4e-6
- double m_eSatVel = 7.7e-3
- double m_hSatVel = 7.7e-3
- double m_eHallFactor = 1.05
- double m_hHallFactor = 1.25
- double m_eImpactA = 1.889e5

- double m_hImpactA = 2.215e5
- double m_eImpactB = 5.75e5
- double m_hImpactB = 6.57e5
- bool m_userMobility = false

### 7.63.1 Detailed Description

Gallium-Arsenide.
Definition at line 10 of file MediumGaAs.hh.

### 7.63.2 Constructor & Destructor Documentation

#### MediumGaAs()

Garfield::MediumGaAs::MediumGaAs ()
Constructor.

#### ~MediumGaAs()

virtual Garfield::MediumGaAs::~MediumGaAs () [inline], [virtual]
Destructor.
Definition at line 15 of file MediumGaAs.hh.
00015 {}

### 7.63.3 Member Function Documentation

#### ElectronAttachment()

bool Garfield::MediumGaAs::ElectronAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

#### ElectronMobility()

double Garfield::MediumGaAs::ElectronMobility () [inline], [override]
Definition at line 32 of file MediumGaAs.hh.
00032 { return m_eMobility; }

#### ElectronTownsend()

bool Garfield::MediumGaAs::ElectronTownsend (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & alpha) [override]

#### ElectronVelocity()

bool Garfield::MediumGaAs::ElectronVelocity (
const double ex,
const double ey,
const double ez,
const double bx,

const double by,
const double bz,
double & vx,
double & vy,
double & vz) [override]

### GetComponent()

void Garfield::MediumGaAs::GetComponent (
const unsigned int i,
std::string & label,
double & f) [override]

### HoleAttachment()

bool Garfield::MediumGaAs::HoleAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

### HoleMobility()

double Garfield::MediumGaAs::HoleMobility () [inline], [override]
Definition at line 43 of file MediumGaAs.hh.
00043 { return m_hMobility; }

### HoleTownsend()

bool Garfield::MediumGaAs::HoleTownsend (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & alpha) [override]

### HoleVelocity()

bool Garfield::MediumGaAs::HoleVelocity (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & vx,
double & vy,
double & vz) [override]

### IsSemiconductor()

bool Garfield::MediumGaAs::IsSemiconductor () const [inline], [override]
Definition at line 17 of file MediumGaAs.hh.
00017 { return true; }

SetLowFieldMobility()

void Garfield::MediumGaAs::SetLowFieldMobility (
const double mue,
const double muh)

UnsetLowFieldMobility()

void Garfield::MediumGaAs::UnsetLowFieldMobility ()

UpdateTransportParameters()

void Garfield::MediumGaAs::UpdateTransportParameters () [private]

### 7.63.4 Member Data Documentation

m_eHallFactor

double Garfield::MediumGaAs::m_eHallFactor = 1.05 [private]
Definition at line 58 of file MediumGaAs.hh.

m_eImpactA

double Garfield::MediumGaAs::m_eImpactA = 1.889e5 [private]
Definition at line 61 of file MediumGaAs.hh.

m_eImpactB

double Garfield::MediumGaAs::m_eImpactB = 5.75e5 [private]
Definition at line 63 of file MediumGaAs.hh.

m_eMobility

double Garfield::MediumGaAs::m_eMobility = 8.0e-6 [private]
Definition at line 52 of file MediumGaAs.hh.

m_eSatVel

double Garfield::MediumGaAs::m_eSatVel = 7.7e-3 [private]
Definition at line 55 of file MediumGaAs.hh.

m_hHallFactor

double Garfield::MediumGaAs::m_hHallFactor = 1.25 [private]
Definition at line 59 of file MediumGaAs.hh.

m_hImpactA

double Garfield::MediumGaAs::m_hImpactA = 2.215e5 [private]
Definition at line 62 of file MediumGaAs.hh.

m_hImpactB

double Garfield::MediumGaAs::m_hImpactB = 6.57e5 [private]
Definition at line 64 of file MediumGaAs.hh.

m_hMobility

double Garfield::MediumGaAs::m_hMobility = 0.4e-6 [private]
Definition at line 53 of file MediumGaAs.hh.

m_hSatVel

double Garfield::MediumGaAs::m_hSatVel = 7.7e-3 [private]
Definition at line 56 of file MediumGaAs.hh.

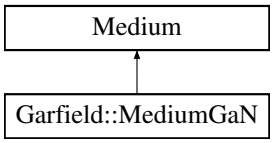
m_userMobility

bool Garfield::MediumGaAs::m_userMobility = false [private]
Definition at line 66 of file MediumGaAs.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumGaAs.hh

## 7.64 Garfield::MediumGaN Class Reference

Gallium-Nitride.
#include <MediumGaN.hh>
Inheritance diagram for Garfield::MediumGaN:




Public Member Functions

- MediumGaN ()

  Constructor.
- virtual ~MediumGaN ()

  Destructor.
- bool IsSemiconductor () const override
- void GetComponent (const unsigned int i, std::string &label, double &f) override
- bool ElectronVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool ElectronTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool ElectronAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double ElectronMobility () override
- bool HoleVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool HoleTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool HoleAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double HoleMobility () override
- void SetElectronConcentration (const double c)

  Set the electron concentration [cm-3].
- void SetLowFieldMobility (const double mue, const double muh)

  Set the low-field mobility values [cm2 / (V ns)] explicitly.
- void UnsetLowFieldMobility ()

  Use the default mobility models.

Private Member Functions

- void UpdateTransportParameters ()

Private Attributes

- double m_eMobility = 1.405e-6
- double m_hMobility = 0.170e-6
- double m_eDensity = 7.78e16
- double m_eHallFactor = 1.
- double m_hHallFactor = 1.
- double m_eImpactA = 1.5e5
- double m_hImpactA = 6.4e5
- double m_eImpactB = 1.41e7
- double m_hImpactB = 1.46e7
- bool m_userMobility = false

### 7.64.1 Detailed Description

Gallium-Nitride.
Definition at line 10 of file MediumGaN.hh.

### 7.64.2 Constructor & Destructor Documentation

MediumGaN()

Garfield::MediumGaN::MediumGaN ()
Constructor.

~MediumGaN()

virtual Garfield::MediumGaN::~MediumGaN () [inline], [virtual]
Destructor.
Definition at line 15 of file MediumGaN.hh.
00015 {}

### 7.64.3 Member Function Documentation

ElectronAttachment()

bool Garfield::MediumGaN::ElectronAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

ElectronMobility()

double Garfield::MediumGaN::ElectronMobility () [inline], [override]
Definition at line 32 of file MediumGaN.hh.
00032 { return m_eMobility; }

ElectronTownsend()

bool Garfield::MediumGaN::ElectronTownsend (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & alpha) [override]

### ElectronVelocity()

bool Garfield::MediumGaN::ElectronVelocity (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & vx,
double & vy,
double & vz) [override]

### GetComponent()

void Garfield::MediumGaN::GetComponent (
const unsigned int i,
std::string & label,
double & f) [override]

### HoleAttachment()

bool Garfield::MediumGaN::HoleAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

### HoleMobility()

double Garfield::MediumGaN::HoleMobility () [inline], [override]
Definition at line 43 of file MediumGaN.hh.
00043 { return m_hMobility; }

### HoleTownsend()

bool Garfield::MediumGaN::HoleTownsend (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & alpha) [override]

### HoleVelocity()

bool Garfield::MediumGaN::HoleVelocity (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & vx,
double & vy,
double & vz) [override]

### IsSemiconductor()

bool Garfield::MediumGaN::IsSemiconductor () const [inline], [override]
Definition at line 17 of file MediumGaN.hh.
00017 { return true; }

### SetElectronConcentration()

void Garfield::MediumGaN::SetElectronConcentration (
const double c)
Set the electron concentration [cm-3].

### SetLowFieldMobility()

void Garfield::MediumGaN::SetLowFieldMobility (
const double mue,
const double muh)
Set the low-field mobility values [cm2 / (V ns)] explicitly.

### UnsetLowFieldMobility()

void Garfield::MediumGaN::UnsetLowFieldMobility ()
Use the default mobility models.

### UpdateTransportParameters()

void Garfield::MediumGaN::UpdateTransportParameters () [private]

## 7.64.4 Member Data Documentation

### m_eDensity

double Garfield::MediumGaN::m_eDensity = 7.78e16 [private]
Definition at line 59 of file MediumGaN.hh.

### m_eHallFactor

double Garfield::MediumGaN::m_eHallFactor = 1. [private]
Definition at line 62 of file MediumGaN.hh.

### m_eImpactA

double Garfield::MediumGaN::m_eImpactA = 1.5e5 [private]
Definition at line 68 of file MediumGaN.hh.

### m_eImpactB

double Garfield::MediumGaN::m_eImpactB = 1.41e7 [private]
Definition at line 70 of file MediumGaN.hh.

### m_eMobility

double Garfield::MediumGaN::m_eMobility = 1.405e-6 [private]
Definition at line 55 of file MediumGaN.hh.

### m_hHallFactor

double Garfield::MediumGaN::m_hHallFactor = 1. [private]
Definition at line 63 of file MediumGaN.hh.

### m_hImpactA

double Garfield::MediumGaN::m_hImpactA = 6.4e5 [private]
Definition at line 69 of file MediumGaN.hh.

### m_hImpactB

double Garfield::MediumGaN::m_hImpactB = 1.46e7 [private]
Definition at line 71 of file MediumGaN.hh.

### m_hMobility

double Garfield::MediumGaN::m_hMobility = 0.170e-6 [private]
Definition at line 56 of file MediumGaN.hh.

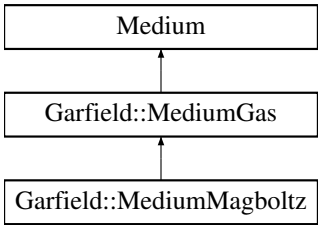
### m_userMobility

bool Garfield::MediumGaN::m_userMobility = false [private]
Definition at line 73 of file MediumGaN.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumGaN.hh

## 7.65 Garfield::MediumGas Class Reference

Base class for gas media.
#include <MediumGas.hh>
Inheritance diagram for Garfield::MediumGas:




Classes

- struct ExcLevel
- struct IonLevel

Public Member Functions

- MediumGas ()

  Constructor.
- virtual ~MediumGas ()

  Destructor.
- bool SetComposition (const std::string &gas1, const double f1=1., const std::string &gas2="", const double f2=0., const std::string &gas3="", const double f3=0., const std::string &gas4="", const double f4=0., const std::string &gas5="", const double f5=0., const std::string &gas6="", const double f6=0.)

  Set the gas mixture.
- void GetComposition (std::string &gas1, double &f1, std::string &gas2, double &f2, std::string &gas3, double &f3, std::string &gas4, double &f4, std::string &gas5, double &f5, std::string &gas6, double &f6) const

  Retrieve the gas mixture.
- bool LoadGasFile (const std::string &filename, const bool quiet=false)

Read table of gas properties (transport parameters) from file.

- bool WriteGasFile (const std::string &filename)

  Save the present table of gas properties (transport parameters) to a file.
- bool MergeGasFile (const std::string &filename, const bool replaceOld)

  Read table of gas properties from and merge with the existing dataset.
- virtual bool EnablePenningTransfer ()

  Switch on simulation of Penning transfers, using pre-implemented parameterisations of the transfer probability (if available).
- virtual bool EnablePenningTransfer (const double r, const double lambda)

  Switch on simulation of Penning transfers by means of transfer probabilities, for all excitation levels in the mixture.
- virtual bool EnablePenningTransfer (const double r, const double lambda, std::string gasname)

  Switch on simulation of Penning transfers by means of transfer probabilities, for all excitations of a given component.
- virtual void DisablePenningTransfer ()

  Switch the simulation of Penning transfers off globally.
- virtual bool DisablePenningTransfer (std::string gasname)

  Switch the simulation of Penning transfers off for a given component.
- bool GetPenningTransfer (const std::string &gasname, double &r, double &lambda)

  Retrieve the Penning transfer probability and distance for a specific component.
- virtual void PrintGas ()

  Print information about the present gas mixture and available data.
- bool LoadIonMobility (const std::string &filename, const bool quiet=false)

  Read a table of (positive) ion mobilities vs. electric field from file.
- bool LoadNegativeIonMobility (const std::string &filename, const bool quiet=false)

  Read a table of negative ion mobilities vs. electric field from file.
- bool AdjustTownsendCoefficient ()

  Adjust the Townsend coefficient using the excitation and ionisation rates stored in the gas table and the Penning transfer probabilities.
- size_t GetNumberOfIonisationLevels () const

  Return the number of ionisation levels in the table.
- size_t GetNumberOfExcitationLevels () const

  Return the number of excitation levels in the table.
- void GetIonisationLevel (const size_t level, std::string &label, double &energy) const

  Return the identifier and threshold of an ionisation level.
- void GetExcitationLevel (const size_t level, std::string &label, double &energy) const

  Return the identifier and energy of an excitation level.
- bool GetElectronIonisationRate (const size_t level, const size_t ie, const size_t ib, const size_t ia, double &f) const

  Get an entry in the table of ionisation rates.
- bool GetElectronExcitationRate (const size_t level, const size_t ie, const size_t ib, const size_t ia, double &f) const

  Get an entry in the table of excitation rates.
- bool IsGas () const override
- void GetComponent (const unsigned int i, std::string &label, double &f) override
- void SetAtomicNumber (const double z) override
- double GetAtomicNumber () const override
- void SetAtomicWeight (const double a) override
- double GetAtomicWeight () const override
- void SetNumberDensity (const double n) override
- double GetNumberDensity () const override
- void SetMassDensity (const double rho) override

- double GetMassDensity () const override
- void ResetTables () override
- void SetExtrapolationMethodExcitationRates (const std::string &low, const std::string &high)
- void SetExtrapolationMethodIonisationRates (const std::string &low, const std::string &high)
- void SetInterpolationMethodExcitationRates (const unsigned int intrp)
- void SetInterpolationMethodIonisationRates (const unsigned int intrp)
- double ScaleElectricField (const double e) const override
- double UnScaleElectricField (const double e) const override
- double ScaleDiffusion (const double d) const override
- double ScaleDiffusionTensor (const double d) const override
- double ScaleTownsend (const double alpha) const override
- double ScaleAttachment (const double eta) const override
- double ScaleLorentzAngle (const double lor) const override
- bool GetPhotoAbsorptionCrossSection (const double e, double &sigma, const unsigned int i) override
- virtual double CreateGPUTransferObject (MediumGPU *&med_gpu) override

  Create and initialise GPU Transfer class.

Static Public Member Functions

- static void PrintGases ()

  Print a list of all available gases.

Protected Member Functions

- bool LoadMobility (const std::string &filename, const bool quiet, const bool negative)
- bool ReadHeader (std::ifstream &gasfile, int &version, std::bitset< 20 > &gasok, bool &is3d, std::vector< double > &mixture, std::vector< double > &efields, std::vector< double > &bfields, std::vector< double > &angles, std::vector< ExcLevel > &excLevels, std::vector< IonLevel > &ionLevels)
- void ReadFooter (std::ifstream &gasfile, std::array< unsigned int, 13 > &extrapH, std::array< unsigned int, 13 > &extrapL, std::array< unsigned int, 13 > &interp, unsigned int &thrAlp, unsigned int &thrAtt, unsigned int &thrDis, double &ionDiffL, double &ionDiffT, double &pgas, double &tgas)
- void ReadRecord3D (std::ifstream &gasfile, double &ve, double &vb, double &vx, double &dl, double &dt, double &alpha, double &alpha0, double &eta, double &mu, double &lor, double &dis, std::array< double, 6 > &dif, std::vector< double > &rexc, std::vector< double > &rion, std::bitset< 20 > gasok)
- void ReadRecord1D (std::ifstream &gasfile, double &ve, double &vb, double &vx, double &wv, double &wr, double &dl, double &dt, double &alpha, double &alpha0, double &eta, double &riontof, double &ratttof, double &mu, double &lor, double &dis, std::array< double, 6 > &dif, std::vector< double > &rexc, std::vector< double > &rion, std::bitset< 20 > gasok)
- void InsertE (const int ie, const int ne, const int nb, const int na)
- void InsertB (const int ib, const int ne, const int nb, const int na)
- void InsertA (const int ia, const int ne, const int nb, const int na)
- void ZeroRowE (const int ie, const int nb, const int na)
- void ZeroRowB (const int ib, const int ne, const int na)
- void ZeroRowA (const int ia, const int ne, const int nb)
- bool GetMixture (const std::vector< double > &mixture, const int version, std::vector< std::string > &gasnames, std::vector< double > &percentages) const
- void GetGasBits (std::bitset< 20 > &gasok) const

Static Protected Member Functions

- static bool GetGasInfo (const std::string &gasname, double &a, double &z, double &w, double &f)
- static std::string GetGasName (const int gasnumber, const int version)
- static std::string GetGasName (std::string input)
- static int GetGasNumberGasFile (const std::string &input)
- static const std::vector< std::string > GetAliases (const std::string &gas)

Protected Attributes

- std::array< std::string, m_nMaxGases > m_gas
- std::array< double, m_nMaxGases > m_fraction
- std::array< double, m_nMaxGases > m_atWeight
- std::array< double, m_nMaxGases > m_atNum
- bool m_usePenning = false
- double m_rPenningGlobal = 0.
- double m_lambdaPenningGlobal = 0.
- std::array< double, m_nMaxGases > m_rPenningGas
- std::array< double, m_nMaxGases > m_lambdaPenningGas
- double m_pressureTable
- double m_temperatureTable
- std::vector< std::vector< std::vector< double > > > m_eAlp0
- std::vector< std::vector< std::vector< std::vector< double > > > > m_excRates
- std::vector< std::vector< std::vector< std::vector< double > > > > m_ionRates
- std::vector< ExcLevel > m_excLevels
- std::vector< IonLevel > m_ionLevels
- std::pair< unsigned int, unsigned int > m_extrExc = {0, 1}
- std::pair< unsigned int, unsigned int > m_extrIon = {0, 1}
- unsigned int m_intpExc = 2
- unsigned int m_intpIon = 2

Static Protected Attributes

- static constexpr unsigned int m_nMaxGases = 6

### 7.65.1 Detailed Description

Base class for gas media.
Definition at line 25 of file MediumGas.hh.

### 7.65.2 Constructor & Destructor Documentation

MediumGas()

Garfield::MediumGas::MediumGas ()
Constructor.

~MediumGas()

virtual Garfield::MediumGas::~MediumGas () [inline], [virtual]
Destructor.
Definition at line 30 of file MediumGas.hh.
00030 {}

### 7.65.3 Member Function Documentation

AdjustTownsendCoefficient()

bool Garfield::MediumGas::AdjustTownsendCoefficient ()
Adjust the Townsend coefficient using the excitation and ionisation rates stored in the gas table and the Penning transfer probabilities.

CreateGPUTransferObject()

virtual double Garfield::MediumGas::CreateGPUTransferObject (
MediumGPU *& med_gpu) [override], [virtual]
Create and initialise GPU Transfer class.
Reimplemented in Garfield::MediumMagboltz.

### DisablePenningTransfer() [1/2]

virtual void Garfield::MediumGas::DisablePenningTransfer () [virtual]
Switch the simulation of Penning transfers off globally.
Reimplemented in Garfield::MediumMagboltz.

### DisablePenningTransfer() [2/2]

virtual bool Garfield::MediumGas::DisablePenningTransfer (
std::string gasname) [virtual]
Switch the simulation of Penning transfers off for a given component.
Reimplemented in Garfield::MediumMagboltz.

### EnablePenningTransfer() [1/3]

virtual bool Garfield::MediumGas::EnablePenningTransfer () [virtual]
Switch on simulation of Penning transfers, using pre-implemented parameterisations of the transfer probability (if available).
Reimplemented in Garfield::MediumMagboltz.

### EnablePenningTransfer() [2/3]

virtual bool Garfield::MediumGas::EnablePenningTransfer (
const double r,
const double lambda) [virtual]
Switch on simulation of Penning transfers by means of transfer probabilities, for all excitation levels in the mixture.

Parameters

| | |
|---|---|
| r | transfer probability [0, 1] |
| lambda | parameter for sampling the distance of the Penning electron with respect to the excitation. |

Reimplemented in Garfield::MediumMagboltz.

### EnablePenningTransfer() [3/3]

virtual bool Garfield::MediumGas::EnablePenningTransfer (
const double r,
const double lambda,
std::string gasname) [virtual]
Switch on simulation of Penning transfers by means of transfer probabilities, for all excitations of a given component.
Reimplemented in Garfield::MediumMagboltz.

### GetAliases()

const std::vector< std::string > Garfield::MediumGas::GetAliases (
const std::string & gas) [static], [protected]

### GetAtomicNumber()

double Garfield::MediumGas::GetAtomicNumber () const [override]

### GetAtomicWeight()

double Garfield::MediumGas::GetAtomicWeight () const [override]

GetComponent()

void Garfield::MediumGas::GetComponent (
const unsigned int i,
std::string & label,
double & f) [override]

GetComposition()

void Garfield::MediumGas::GetComposition (
std::string & gas1,
double & f1,
std::string & gas2,
double & f2,
std::string & gas3,
double & f3,
std::string & gas4,
double & f4,
std::string & gas5,
double & f5,
std::string & gas6,
double & f6) const

Retrieve the gas mixture.

GetElectronExcitationRate()

bool Garfield::MediumGas::GetElectronExcitationRate (
const size_t level,
const size_t ie,
const size_t ib,
const size_t ia,
double & f) const

Get an entry in the table of excitation rates.

GetElectronIonisationRate()

bool Garfield::MediumGas::GetElectronIonisationRate (
const size_t level,
const size_t ie,
const size_t ib,
const size_t ia,
double & f) const

Get an entry in the table of ionisation rates.

GetExcitationLevel()

void Garfield::MediumGas::GetExcitationLevel (
const size_t level,
std::string & label,
double & energy) const

Return the identifier and energy of an excitation level.

GetGasBits()

void Garfield::MediumGas::GetGasBits (
std::bitset< 20 > & gasok) const [protected]

GetGasInfo()

bool Garfield::MediumGas::GetGasInfo (
const std::string & gasname,

double & a,
double & z,
double & w,
double & f) [static], [protected]

### GetGasName() [1/2]

std::string Garfield::MediumGas::GetGasName (
const int gasnumber,
const int version) [static], [protected]

### GetGasName() [2/2]

std::string Garfield::MediumGas::GetGasName (
std::string input) [static], [protected]

### GetGasNumberGasFile()

int Garfield::MediumGas::GetGasNumberGasFile (
const std::string & input) [static], [protected]

### GetIonisationLevel()

void Garfield::MediumGas::GetIonisationLevel (
const size_t level,
std::string & label,
double & energy) const

Return the identifier and threshold of an ionisation level.

### GetMassDensity()

double Garfield::MediumGas::GetMassDensity () const [override]

### GetMixture()

bool Garfield::MediumGas::GetMixture (
const std::vector< double > & mixture,
const int version,
std::vector< std::string > & gasnames,
std::vector< double > & percentages) const [protected]

### GetNumberDensity()

double Garfield::MediumGas::GetNumberDensity () const [override]

### GetNumberOfExcitationLevels()

size_t Garfield::MediumGas::GetNumberOfExcitationLevels () const [inline]

Return the number of excitation levels in the table.
Definition at line 94 of file MediumGas.hh.

```
00094 { return m_excLevels.size(); }
```

### GetNumberOfIonisationLevels()

size_t Garfield::MediumGas::GetNumberOfIonisationLevels () const [inline]

Return the number of ionisation levels in the table.
Definition at line 92 of file MediumGas.hh.

```
00092 { return m_ionLevels.size(); }
```

### GetPenningTransfer()

```
bool Garfield::MediumGas::GetPenningTransfer (
                const std::string & gasname,
                double & r,
                double & lambda)
```

Retrieve the Penning transfer probability and distance for a specific component.

### GetPhotoAbsorptionCrossSection()

```
bool Garfield::MediumGas::GetPhotoAbsorptionCrossSection (
                const double e,
                double & sigma,
                const unsigned int i)   [override]
```

### InsertA()

```
void Garfield::MediumGas::InsertA (
                const int ia,
                const int ne,
                const int nb,
                const int na)   [protected]
```

### InsertB()

```
void Garfield::MediumGas::InsertB (
                const int ib,
                const int ne,
                const int nb,
                const int na)   [protected]
```

### InsertE()

```
void Garfield::MediumGas::InsertE (
                const int ie,
                const int ne,
                const int nb,
                const int na)   [protected]
```

### IsGas()

```
bool Garfield::MediumGas::IsGas () const   [inline], [override]
```

Definition at line 110 of file MediumGas.hh.

```
00110 { return true; }
```

### LoadGasFile()

```
bool Garfield::MediumGas::LoadGasFile (
                const std::string & filename,
                const bool quiet = false)
```

Read table of gas properties (transport parameters) from file.

### LoadIonMobility()

```
bool Garfield::MediumGas::LoadIonMobility (
                const std::string & filename,
                const bool quiet = false)
```

Read a table of (positive) ion mobilities vs. electric field from file.

### LoadMobility()

bool Garfield::MediumGas::LoadMobility (
const std::string & filename,
const bool quiet,
const bool negative) [protected]

### LoadNegativeIonMobility()

bool Garfield::MediumGas::LoadNegativeIonMobility (
const std::string & filename,
const bool quiet = false)
Read a table of negative ion mobilities vs. electric field from file.

### MergeGasFile()

bool Garfield::MediumGas::MergeGasFile (
const std::string & filename,
const bool replaceOld)
Read table of gas properties from and merge with the existing dataset.

### PrintGas()

virtual void Garfield::MediumGas::PrintGas () [virtual]
Print information about the present gas mixture and available data.
Reimplemented in Garfield::MediumMagboltz.

### PrintGases()

void Garfield::MediumGas::PrintGases () [static]
Print a list of all available gases.

### ReadFooter()

void Garfield::MediumGas::ReadFooter (
std::ifstream & gasfile,
std::array< unsigned int, 13 > & extrapH,
std::array< unsigned int, 13 > & extrapL,
std::array< unsigned int, 13 > & interp,
unsigned int & thrAlp,
unsigned int & thrAtt,
unsigned int & thrDis,
double & ionDiffL,
double & ionDiffT,
double & pgas,
double & tgas) [protected]

### ReadHeader()

bool Garfield::MediumGas::ReadHeader (
std::ifstream & gasfile,
int & version,
std::bitset< 20 > & gasok,
bool & is3d,
std::vector< double > & mixture,
std::vector< double > & efields,
std::vector< double > & bfields,
std::vector< double > & angles,
std::vector< ExcLevel > & excLevels,
std::vector< IonLevel > & ionLevels) [protected]

### ReadRecord1D()

```
void Garfield::MediumGas::ReadRecord1D (
                std::ifstream & gasfile,
                double & ve,
                double & vb,
                double & vx,
                double & wv,
                double & wr,
                double & dl,
                double & dt,
                double & alpha,
                double & alpha0,
                double & eta,
                double & riontof,
                double & ratttof,
                double & mu,
                double & lor,
                double & dis,
                std::array< double, 6 > & dif,
                std::vector< double > & rexc,
                std::vector< double > & rion,
                std::bitset< 20 > gasok)   [protected]
```

### ReadRecord3D()

```
void Garfield::MediumGas::ReadRecord3D (
                std::ifstream & gasfile,
                double & ve,
                double & vb,
                double & vx,
                double & dl,
                double & dt,
                double & alpha,
                double & alpha0,
                double & eta,
                double & mu,
                double & lor,
                double & dis,
                std::array< double, 6 > & dif,
                std::vector< double > & rexc,
                std::vector< double > & rion,
                std::bitset< 20 > gasok)   [protected]
```

### ResetTables()

```
void Garfield::MediumGas::ResetTables ()   [override]
```

### ScaleAttachment()

```
double Garfield::MediumGas::ScaleAttachment (
                const double eta) const   [inline], [override]
```

Definition at line 158 of file MediumGas.hh.

```
00158                                                 {
00159     return eta * m_pressure / m_pressureTable;
00160   }
```

### ScaleDiffusion()

```
double Garfield::MediumGas::ScaleDiffusion (
                const double d) const   [inline], [override]
```

Definition at line 149 of file MediumGas.hh.

```
00149                                                 {
00150     return d * sqrt(m_pressureTable / m_pressure);
00151   }
```

### ScaleDiffusionTensor()

double Garfield::MediumGas::ScaleDiffusionTensor (
const double d) const [inline], [override]

Definition at line 152 of file MediumGas.hh.

```
00152                                                      {
00153     return d * m_pressureTable / m_pressure;
00154   }
```

### ScaleElectricField()

double Garfield::MediumGas::ScaleElectricField (
const double e) const [inline], [override]

Definition at line 143 of file MediumGas.hh.

```
00143                                                    {
00144     return e * m_pressureTable / m_pressure;
00145   }
```

### ScaleLorentzAngle()

double Garfield::MediumGas::ScaleLorentzAngle (
const double lor) const [inline], [override]

Definition at line 161 of file MediumGas.hh.

```
00161                                                    {
00162     return lor * m_pressure / m_pressureTable;
00163   }
```

### ScaleTownsend()

double Garfield::MediumGas::ScaleTownsend (
const double alpha) const [inline], [override]

Definition at line 155 of file MediumGas.hh.

```
00155                                                    {
00156     return alpha * m_pressure / m_pressureTable;
00157   }
```

### SetAtomicNumber()

void Garfield::MediumGas::SetAtomicNumber (
const double z) [override]

### SetAtomicWeight()

void Garfield::MediumGas::SetAtomicWeight (
const double a) [override]

### SetComposition()

bool Garfield::MediumGas::SetComposition (
const std::string & gas1,
const double f1 = 1.,
const std::string & gas2 = "",
const double f2 = 0.,
const std::string & gas3 = "",
const double f3 = 0.,
const std::string & gas4 = "",
const double f4 = 0.,
const std::string & gas5 = "",
const double f5 = 0.,

```
                const std::string & gas6 = "",
                const double f6 = 0.)
```

Set the gas mixture.

### SetExtrapolationMethodExcitationRates()

```
void Garfield::MediumGas::SetExtrapolationMethodExcitationRates (
                const std::string & low,
                const std::string & high)   [inline]
```

Definition at line 126 of file MediumGas.hh.

```
00127                                                   {
00128     SetExtrapolationMethod(low, high, m_extrExc, "ExcitationRates");
00129   }
```

### SetExtrapolationMethodIonisationRates()

```
void Garfield::MediumGas::SetExtrapolationMethodIonisationRates (
                const std::string & low,
                const std::string & high)   [inline]
```

Definition at line 130 of file MediumGas.hh.

```
00131                                                   {
00132     SetExtrapolationMethod(low, high, m_extrIon, "IonisationRates");
00133   }
```

### SetInterpolationMethodExcitationRates()

```
void Garfield::MediumGas::SetInterpolationMethodExcitationRates (
                const unsigned int intrp)   [inline]
```

Definition at line 134 of file MediumGas.hh.

```
00134                                                   {
00135     if (intrp > 0) m_intpExc = intrp;
00136   }
```

### SetInterpolationMethodIonisationRates()

```
void Garfield::MediumGas::SetInterpolationMethodIonisationRates (
                const unsigned int intrp)   [inline]
```

Definition at line 137 of file MediumGas.hh.

```
00137                                                   {
00138     if (intrp > 0) m_intpIon = intrp;
00139   }
```

### SetMassDensity()

```
void Garfield::MediumGas::SetMassDensity (
                const double rho)   [override]
```

### SetNumberDensity()

```
void Garfield::MediumGas::SetNumberDensity (
                const double n)   [override]
```

### UnScaleElectricField()

```
double Garfield::MediumGas::UnScaleElectricField (
                const double e) const   [inline], [override]
```

Definition at line 146 of file MediumGas.hh.

```
00146                                                   {
00147     return e * m_pressure / m_pressureTable;
00148   }
```

### WriteGasFile()

bool Garfield::MediumGas::WriteGasFile (
const std::string & filename)

Save the present table of gas properties (transport parameters) to a file.

### ZeroRowA()

void Garfield::MediumGas::ZeroRowA (
const int ia,
const int ne,
const int nb) [protected]

### ZeroRowB()

void Garfield::MediumGas::ZeroRowB (
const int ib,
const int ne,
const int na) [protected]

### ZeroRowE()

void Garfield::MediumGas::ZeroRowE (
const int ie,
const int nb,
const int na) [protected]

## 7.65.4 Member Data Documentation

### m_atNum

std::array<double, m_nMaxGases> Garfield::MediumGas::m_atNum [protected]

Definition at line 181 of file MediumGas.hh.

### m_atWeight

std::array<double, m_nMaxGases> Garfield::MediumGas::m_atWeight [protected]

Definition at line 180 of file MediumGas.hh.

### m_eAlp0

std::vector<std::vector<std::vector<double> > > Garfield::MediumGas::m_eAlp0 [protected]

Definition at line 201 of file MediumGas.hh.

### m_excLevels

std::vector<ExcLevel> Garfield::MediumGas::m_excLevels [protected]

Definition at line 215 of file MediumGas.hh.

### m_excRates

std::vector<std::vector<std::vector<std::vector<double> > > > Garfield::MediumGas::m_excRates [protected]

Definition at line 204 of file MediumGas.hh.

### m_extrExc

std::pair<unsigned int, unsigned int> Garfield::MediumGas::m_extrExc = {0, 1} [protected]

Definition at line 224 of file MediumGas.hh.

00224 {0, 1};

### m_extrIon

std::pair<unsigned int, unsigned int> Garfield::MediumGas::m_extrIon = {0, 1} [protected]
Definition at line 225 of file MediumGas.hh.
00225 {0, 1};

### m_fraction

std::array<double, m_nMaxGases> Garfield::MediumGas::m_fraction [protected]
Definition at line 179 of file MediumGas.hh.

### m_gas

std::array<std::string, m_nMaxGases> Garfield::MediumGas::m_gas [protected]
Definition at line 178 of file MediumGas.hh.

### m_intpExc

unsigned int Garfield::MediumGas::m_intpExc = 2 [protected]
Definition at line 226 of file MediumGas.hh.

### m_intpIon

unsigned int Garfield::MediumGas::m_intpIon = 2 [protected]
Definition at line 227 of file MediumGas.hh.

### m_ionLevels

std::vector<IonLevel> Garfield::MediumGas::m_ionLevels [protected]
Definition at line 221 of file MediumGas.hh.

### m_ionRates

std::vector<std::vector<std::vector<std::vector<double> > > > Garfield::MediumGas::m_ionRates [protected]
Definition at line 205 of file MediumGas.hh.

### m_lambdaPenningGas

std::array<double, m_nMaxGases> Garfield::MediumGas::m_lambdaPenningGas [protected]
Definition at line 193 of file MediumGas.hh.

### m_lambdaPenningGlobal

double Garfield::MediumGas::m_lambdaPenningGlobal = 0. [protected]
Definition at line 189 of file MediumGas.hh.

### m_nMaxGases

unsigned int Garfield::MediumGas::m_nMaxGases = 6 [static], [constexpr], [protected]
Definition at line 174 of file MediumGas.hh.

### m_pressureTable

double Garfield::MediumGas::m_pressureTable [protected]
Definition at line 196 of file MediumGas.hh.

### m_rPenningGas

std::array<double, m_nMaxGases> Garfield::MediumGas::m_rPenningGas [protected]
Definition at line 191 of file MediumGas.hh.

m_rPenningGlobal

double Garfield::MediumGas::m_rPenningGlobal = 0. [protected]
Definition at line 187 of file MediumGas.hh.

m_temperatureTable

double Garfield::MediumGas::m_temperatureTable [protected]
Definition at line 198 of file MediumGas.hh.

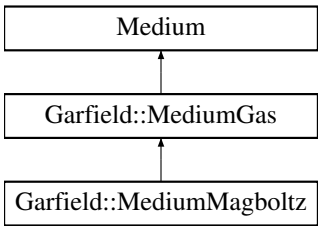
m_usePenning

bool Garfield::MediumGas::m_usePenning = false [protected]
Definition at line 185 of file MediumGas.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumGas.hh

## 7.66 Garfield::MediumMagboltz Class Reference

Interface to Magboltz (version 11).
#include <MediumMagboltz.hh>
Inheritance diagram for Garfield::MediumMagboltz:




Classes

- struct Deexcitation

Public Member Functions

- MediumMagboltz ()

  Default constructor.
- MediumMagboltz (const std::string &gas1, const double f1=1., const std::string &gas2="", const double f2=0., const std::string &gas3="", const double f3=0., const std::string &gas4="", const double f4=0., const std::string &gas5="", const double f5=0., const std::string &gas6="", const double f6=0.)

  Constructor.
- virtual ~MediumMagboltz ()

  Destructor.
- bool SetMaxElectronEnergy (const double e)

  Set the highest electron energy to be included in the table of scattering rates.
- double GetMaxElectronEnergy () const

  Get the highest electron energy in the table of scattering rates.
- bool SetMaxPhotonEnergy (const double e)

  Set the highest photon energy to be included in the table of scattering rates.
- double GetMaxPhotonEnergy () const

  Get the highest photon energy in the table of scattering rates.
- void EnableAnisotropicScattering (const bool on=true)

  Switch on/off anisotropic scattering (enabled by default)

- void SetSplittingFunctionOpalBeaty ()

  Sample the secondary electron energy according to the Opal-Beaty model.
- void SetSplittingFunctionGreenSawada ()

  Sample the secondary electron energy according to the Green-Sawada model.
- void SetSplittingFunctionFlat ()

  Sample the secondary electron energy from a flat distribution.
- void EnableDeexcitation ()

  Switch on (microscopic) de-excitation handling.
- void DisableDeexcitation ()

  Switch off (microscopic) de-excitation handling.
- void EnableRadiationTrapping ()

  Switch on discrete photoabsorption levels.
- void DisableRadiationTrapping ()

  Switch off discrete photoabsorption levels.
- void SetLineWidth (const double n)

  Set the number of emission line widths within which to apply absorption by discrete lines.
- bool EnablePenningTransfer () override

  Switch on simulation of Penning transfers, using pre-implemented parameterisations of the transfer probability (if available).
- bool EnablePenningTransfer (const double r, const double lambda) override

  Switch on simulation of Penning transfers by means of transfer probabilities, for all excitation levels in the mixture.
- bool EnablePenningTransfer (const double r, const double lambda, std::string gasname) override

  Switch on simulation of Penning transfers by means of transfer probabilities, for all excitations of a given component.
- void DisablePenningTransfer () override

  Switch the simulation of Penning transfers off globally.
- bool DisablePenningTransfer (std::string gasname) override

  Switch the simulation of Penning transfers off for a given component.
- void EnableCrossSectionOutput (const bool on=true)

  Write the gas cross-section table to a file during the initialisation.
- void SetExcitationScaling (const double r, std::string gasname)

  Multiply all excitation cross-sections by a uniform scaling factor.
- bool Initialise (const bool verbose=false)

  Initialise the table of scattering rates (called internally when a collision rate is requested and the gas mixture or other parameters have changed).
- void PrintGas () override

  Print information about the present gas mixture and available data.
- double GetElectronNullCollisionRate (const int band) override
- double GetElectronCollisionRate (const double e, const int band) override

  Get the (real) collision rate [ns-1] at a given electron energy e [eV].
- double GetElectronCollisionRate (const double e, const unsigned int level, const int band)

  Get the collision rate [ns-1] for a specific level.
- bool ElectronCollision (const double e, int &type, int &level, double &e1, double &dx, double &dy, double &dz, std::vector< Secondary > &secondaries, int &band) override

  Sample the collision type.
- void ComputeDeexcitation (int iLevel, int &fLevel, std::vector< Secondary > &secondaries)
- double GetPhotonCollisionRate (const double e) override
- bool PhotonCollision (const double e, int &type, int &level, double &e1, double &ctheta, std::vector< Secondary > &secondaries) override
- void ResetCollisionCounters ()

  Reset the collision counters.

- unsigned int GetNumberOfElectronCollisions () const

  Get the total number of electron collisions.
- unsigned int GetNumberOfElectronCollisions (unsigned int &nElastic, unsigned int &nIonising, unsigned int &nAttachment, unsigned int &nInelastic, unsigned int &nExcitation, unsigned int &nSuperelastic) const

  Get the number of collisions broken down by cross-section type.
- unsigned int GetNumberOfLevels ()

  Get the number of cross-section terms.
- bool GetLevel (const unsigned int i, int &ngas, int &type, std::string &descr, double &e)

  Get detailed information about a given cross-section term i.
- bool GetPenningTransfer (const unsigned int i, double &r, double &lambda)

  Get the Penning transfer probability and distance of a specific level.
- unsigned int GetNumberOfElectronCollisions (const unsigned int level) const

  Get the number of collisions for a specific cross-section term.
- unsigned int GetNumberOfPenningTransfers () const

  Get the number of Penning transfers that occured since the last reset.
- unsigned int GetNumberOfPhotonCollisions () const

  Get the total number of photon collisions.
- unsigned int GetNumberOfPhotonCollisions (unsigned int &nElastic, unsigned int &nIonising, unsigned int &nInelastic) const

  Get number of photon collisions by collision type.
- void EnableThermalMotion (const bool on=true)

  Take the thermal motion of the gas at the selected temperature into account in the calculations done by Magboltz.
- void EnableAutoEnergyLimit (const bool on=true)

  Let Magboltz determine the upper energy limit (this is the default) or use the energy limit specified using SetMaxElectronEnergy).
- void RunMagboltz (const double e, const double b, const double btheta, const int ncoll, bool verbose, double &vx, double &vy, double &vz, double &wv, double &wr, double &dl, double &dt, double &alpha, double &eta, double &riontof, double &ratttof, double &lor, double &vxerr, double &vyerr, double &vzerr, double &wverr, double &wrerr, double &dlerr, double &dterr, double &alphaerr, double &etaerr, double &riontoferr, double &ratttoferr, double &lorerr, double &alphatof, std::array< double, 6 > &difftens)

  Run Magboltz for a given electric field, magnetic field and angle.
- void GenerateGasTable (const int numCollisions=10, const bool verbose=true)

  Generate a new gas table (can later be saved to file) by running Magboltz for all electric fields, magnetic fields, and angles in the currently set grid.
- void PlotElectronCrossSections (const unsigned int i, TPad *pad)
- void PlotElectronCollisionRates (TPad *pad)
- void PlotElectronInverseMeanFreePath (TPad *pad)
- double CreateGPUTransferObject (MediumGPU *&med_gpu) override

  Create and initialise GPU Transfer class.

Public Member Functions inherited from Garfield::MediumGas

- MediumGas ()

  Constructor.
- virtual ~MediumGas ()

  Destructor.
- bool SetComposition (const std::string &gas1, const double f1=1., const std::string &gas2="", const double f2=0., const std::string &gas3="", const double f3=0., const std::string &gas4="", const double f4=0., const std::string &gas5="", const double f5=0., const std::string &gas6="", const double f6=0.)

  Set the gas mixture.

- void GetComposition (std::string &gas1, double &f1, std::string &gas2, double &f2, std::string &gas3, double &f3, std::string &gas4, double &f4, std::string &gas5, double &f5, std::string &gas6, double &f6) const

  Retrieve the gas mixture.
- bool LoadGasFile (const std::string &filename, const bool quiet=false)

  Read table of gas properties (transport parameters) from file.
- bool WriteGasFile (const std::string &filename)

  Save the present table of gas properties (transport parameters) to a file.
- bool MergeGasFile (const std::string &filename, const bool replaceOld)

  Read table of gas properties from and merge with the existing dataset.
- bool GetPenningTransfer (const std::string &gasname, double &r, double &lambda)

  Retrieve the Penning transfer probability and distance for a specific component.
- bool LoadIonMobility (const std::string &filename, const bool quiet=false)

  Read a table of (positive) ion mobilities vs. electric field from file.
- bool LoadNegativeIonMobility (const std::string &filename, const bool quiet=false)

  Read a table of negative ion mobilities vs. electric field from file.
- bool AdjustTownsendCoefficient ()

  Adjust the Townsend coefficient using the excitation and ionisation rates stored in the gas table and the Penning transfer probabilities.
- size_t GetNumberOfIonisationLevels () const

  Return the number of ionisation levels in the table.
- size_t GetNumberOfExcitationLevels () const

  Return the number of excitation levels in the table.
- void GetIonisationLevel (const size_t level, std::string &label, double &energy) const

  Return the identifier and threshold of an ionisation level.
- void GetExcitationLevel (const size_t level, std::string &label, double &energy) const

  Return the identifier and energy of an excitation level.
- bool GetElectronIonisationRate (const size_t level, const size_t ie, const size_t ib, const size_t ia, double &f) const

  Get an entry in the table of ionisation rates.
- bool GetElectronExcitationRate (const size_t level, const size_t ie, const size_t ib, const size_t ia, double &f) const

  Get an entry in the table of excitation rates.
- bool IsGas () const override
- void GetComponent (const unsigned int i, std::string &label, double &f) override
- void SetAtomicNumber (const double z) override
- double GetAtomicNumber () const override
- void SetAtomicWeight (const double a) override
- double GetAtomicWeight () const override
- void SetNumberDensity (const double n) override
- double GetNumberDensity () const override
- void SetMassDensity (const double rho) override
- double GetMassDensity () const override
- void ResetTables () override
- void SetExtrapolationMethodExcitationRates (const std::string &low, const std::string &high)
- void SetExtrapolationMethodIonisationRates (const std::string &low, const std::string &high)
- void SetInterpolationMethodExcitationRates (const unsigned int intrp)
- void SetInterpolationMethodIonisationRates (const unsigned int intrp)
- double ScaleElectricField (const double e) const override
- double UnScaleElectricField (const double e) const override
- double ScaleDiffusion (const double d) const override
- double ScaleDiffusionTensor (const double d) const override
- double ScaleTownsend (const double alpha) const override
- double ScaleAttachment (const double eta) const override
- double ScaleLorentzAngle (const double lor) const override
- bool GetPhotoAbsorptionCrossSection (const double e, double &sigma, const unsigned int i) override

Static Public Member Functions

- static int GetGasNumberMagboltz (const std::string &input)

Static Public Member Functions inherited from Garfield::MediumGas

- static void PrintGases ()

  Print a list of all available gases.

Private Member Functions

- bool Update (const bool verbose=false)
- bool Mixer (const bool verbose=false)
- void SetupGreenSawada ()
- void GetExcitationIonisationLevels ()
- void ComputeDeexcitationTable (const bool verbose)
- void AddPenningDeexcitation (Deexcitation &dxc, const double rate, const double pPenning)
- double RateConstantWK (const double energy, const double osc, const double pacs, const int igas1, const int igas2) const
- double RateConstantHardSphere (const double r1, const double r2, const int igas1, const int igas2) const
- double CalcDiscreteLineCf (const Deexcitation &dxc, double e, double cfOth) const
- void ComputeDeexcitationInternal (int iLevel, int &fLevel, std::vector< Secondary > &secondaries)
- bool ComputePhotonCollisionTable (const bool verbose)

Private Attributes

- std::mutex m_mutex

  Mutex.
- bool m_useGasMotion = false

  Simulate thermal motion of the gas or not (when running Magboltz).
- bool m_autoEnergyLimit = true

  Automatic calculation of the energy limit by Magboltz or not.
- double m_eMax

  Max. electron energy in the collision rate tables.
- double m_eStep

  Energy spacing in the linear part of the collision rate tables.
- double m_eStepInv

  Inverse energy spacing.
- double m_eHigh
- double m_eHighLog
- double m_lnStep
- bool m_useCsOutput = false

  Flag enabling/disabling output of cross-section table to file.
- unsigned int m_nTerms = 0

  Number of different cross-section types in the current gas mixture.
- std::array< double, m_nMaxGases > m_mgas

  Mass.
- std::array< double, m_nMaxGases > m_rgas

  Recoil energy parameter.
- std::array< double, m_nMaxGases > m_s2
- std::array< double, Magboltz::nMaxLevels > m_wOpalBeaty

  Opal-Beaty-Peterson splitting parameter [eV].
- std::array< std::array< double, 5 >, m_nMaxGases > m_parGreenSawada

  Green-Sawada splitting parameters [eV] (Γs, Γb, Ts, Ta, Tb).

- std::array< bool, m_nMaxGases > m_hasGreenSawada
- bool m_useOpalBeaty = true

  Sample secondary electron energies using Opal-Beaty parameterisation.
- bool m_useGreenSawada = false

  Sample secondary electron energies using Green-Sawada parameterisation.
- std::array< double, Magboltz::nMaxLevels > m_energyLoss

  Energy loss.
- std::array< int, Magboltz::nMaxLevels > m_csType

  Cross-section type.
- std::array< double, Magboltz::nMaxLevels > m_yFluorescence

  Fluorescence yield.
- std::array< unsigned int, Magboltz::nMaxLevels > m_nAuger1

  Number of Auger electrons produced in a collision.
- std::array< unsigned int, Magboltz::nMaxLevels > m_nAuger2
- std::array< double, Magboltz::nMaxLevels > m_eAuger1

  Energy imparted to Auger electrons.
- std::array< double, Magboltz::nMaxLevels > m_eAuger2
- std::array< unsigned int, Magboltz::nMaxLevels > m_nFluorescence
- std::array< double, Magboltz::nMaxLevels > m_eFluorescence
- std::vector< std::vector< double > > m_scatPar
- std::vector< std::vector< double > > m_scatCut
- std::vector< std::vector< double > > m_scatParLog
- std::vector< std::vector< double > > m_scatCutLog
- std::array< int, Magboltz::nMaxLevels > m_scatModel
- std::vector< std::string > m_description

  Level description.
- std::vector< double > m_cfTot
- std::vector< double > m_cfTotLog
- std::vector< std::vector< double > > m_cf
- std::vector< std::vector< double > > m_cfLog
- bool m_useAnisotropic = true
- double m_cfNull = 0.

  Null-collision frequency.
- std::array< unsigned int, nCsTypes > m_nCollisions

  Collision counters 0: elastic 1: ionisation 2: attachment 3: inelastic 4: excitation 5: super-elastic.
- std::vector< unsigned int > m_nCollisionsDetailed

  Number of collisions for each cross-section term.
- std::array< double, Magboltz::nMaxLevels > m_rPenning

  Penning transfer probability (by level)
- std::array< double, Magboltz::nMaxLevels > m_lambdaPenning

  Mean distance of Penning ionisation (by level)
- unsigned int m_nPenning = 0

  Number of Penning ionisations.
- bool m_useDeexcitation = false

  Flag enabling/disabling detailed simulation of de-excitation process.
- bool m_useRadTrap = true

  Flag enabling/disable radiation trapping (absorption of photons discrete excitation lines)
- std::vector< Deexcitation > m_deexcitations
- std::array< int, Magboltz::nMaxLevels > m_iDeexcitation
- double m_nAbsWidths = 1000.
- std::array< double, m_nMaxGases > m_ionPot

  Ionisation potentials of each component.

- double m_minIonPot = -1.

  Minimum ionisation potential.
- std::array< double, m_nMaxGases > m_scaleExc
- double m_eFinalGamma
- double m_eStepGamma
- unsigned int m_nPhotonTerms = 0
- std::vector< double > m_cfTotGamma
- std::vector< std::vector< double > > m_cfGamma
- std::vector< int > csTypeGamma
- std::array< unsigned int, nCsTypesGamma > m_nPhotonCollisions

Static Private Attributes

- static constexpr int nEnergyStepsLog = 1000
- static constexpr int nEnergyStepsGamma = 5000
- static constexpr int nCsTypes = 7
- static constexpr int nCsTypesGamma = 4
- static const int DxcTypeRad
- static const int DxcTypeCollIon
- static const int DxcTypeCollNonIon

Additional Inherited Members

Protected Member Functions inherited from Garfield::MediumGas

- bool LoadMobility (const std::string &filename, const bool quiet, const bool negative)
- bool ReadHeader (std::ifstream &gasfile, int &version, std::bitset< 20 > &gasok, bool &is3d, std::vector< double > &mixture, std::vector< double > &efields, std::vector< double > &bfields, std::vector< double > &angles, std::vector< ExcLevel > &excLevels, std::vector< IonLevel > &ionLevels)
- void ReadFooter (std::ifstream &gasfile, std::array< unsigned int, 13 > &extrapH, std::array< unsigned int, 13 > &extrapL, std::array< unsigned int, 13 > &interp, unsigned int &thrAlp, unsigned int &thrAtt, unsigned int &thrDis, double &ionDiffL, double &ionDiffT, double &pgas, double &tgas)
- void ReadRecord3D (std::ifstream &gasfile, double &ve, double &vb, double &vx, double &dl, double &dt, double &alpha, double &alpha0, double &eta, double &mu, double &lor, double &dis, std::array< double, 6 > &dif, std::vector< double > &rexc, std::vector< double > &rion, std::bitset< 20 > gasok)
- void ReadRecord1D (std::ifstream &gasfile, double &ve, double &vb, double &vx, double &wv, double &wr, double &dl, double &dt, double &alpha, double &alpha0, double &eta, double &riontof, double &ratttof, double &mu, double &lor, double &dis, std::array< double, 6 > &dif, std::vector< double > &rexc, std::vector< double > &rion, std::bitset< 20 > gasok)
- void InsertE (const int ie, const int ne, const int nb, const int na)
- void InsertB (const int ib, const int ne, const int nb, const int na)
- void InsertA (const int ia, const int ne, const int nb, const int na)
- void ZeroRowE (const int ie, const int nb, const int na)
- void ZeroRowB (const int ib, const int ne, const int na)
- void ZeroRowA (const int ia, const int ne, const int nb)
- bool GetMixture (const std::vector< double > &mixture, const int version, std::vector< std::string > &gasnames, std::vector< double > &percentages) const
- void GetGasBits (std::bitset< 20 > &gasok) const

Static Protected Member Functions inherited from Garfield::MediumGas

- static bool GetGasInfo (const std::string &gasname, double &a, double &z, double &w, double &f)
- static std::string GetGasName (const int gasnumber, const int version)
- static std::string GetGasName (std::string input)
- static int GetGasNumberGasFile (const std::string &input)
- static const std::vector< std::string > GetAliases (const std::string &gas)

Protected Attributes inherited from Garfield::MediumGas

- std::array< std::string, m_nMaxGases > m_gas
- std::array< double, m_nMaxGases > m_fraction
- std::array< double, m_nMaxGases > m_atWeight
- std::array< double, m_nMaxGases > m_atNum
- bool m_usePenning = false
- double m_rPenningGlobal = 0.
- double m_lambdaPenningGlobal = 0.
- std::array< double, m_nMaxGases > m_rPenningGas
- std::array< double, m_nMaxGases > m_lambdaPenningGas
- double m_pressureTable
- double m_temperatureTable
- std::vector< std::vector< std::vector< double > > > m_eAlp0
- std::vector< std::vector< std::vector< std::vector< double > > > > m_excRates
- std::vector< std::vector< std::vector< std::vector< double > > > > m_ionRates
- std::vector< ExcLevel > m_excLevels
- std::vector< IonLevel > m_ionLevels
- std::pair< unsigned int, unsigned int > m_extrExc = {0, 1}
- std::pair< unsigned int, unsigned int > m_extrIon = {0, 1}
- unsigned int m_intpExc = 2
- unsigned int m_intpIon = 2

Static Protected Attributes inherited from Garfield::MediumGas

- static constexpr unsigned int m_nMaxGases = 6

### 7.66.1 Detailed Description

Interface to Magboltz (version 11).

- http://magboltz.web.cern.ch/magboltz/

Definition at line 22 of file MediumMagboltz.hh.

### 7.66.2 Constructor & Destructor Documentation

MediumMagboltz() [1/2]

Garfield::MediumMagboltz::MediumMagboltz ()
Default constructor.

MediumMagboltz() [2/2]

```
Garfield::MediumMagboltz::MediumMagboltz (
                const std::string & gas1,
                const double f1 = 1.,
                const std::string & gas2 = "",
                const double f2 = 0.,
                const std::string & gas3 = "",
                const double f3 = 0.,
                const std::string & gas4 = "",
                const double f4 = 0.,
                const std::string & gas5 = "",
                const double f5 = 0.,
                const std::string & gas6 = "",
                const double f6 = 0.)
```

Constructor.

### ~MediumMagboltz()

virtual Garfield::MediumMagboltz::~MediumMagboltz () [inline], [virtual]
Destructor.
Definition at line 34 of file MediumMagboltz.hh.

```
00034 {}
```

## 7.66.3 Member Function Documentation

### AddPenningDeexcitation()

void Garfield::MediumMagboltz::AddPenningDeexcitation (
Deexcitation & dxc,
const double rate,
const double pPenning) [inline], [private]
Definition at line 449 of file MediumMagboltz.hh.

```
00450                                                 {
00451     dxc.p.push_back(rate * (1. - pPenning));
00452     dxc.type.push_back(DxcTypeCollNonIon);
00453     dxc.final.push_back(-1);
00454     if (pPenning > 0.) {
00455       dxc.p.push_back(rate * pPenning);
00456       dxc.type.push_back(DxcTypeCollIon);
00457       dxc.final.push_back(-1);
00458     }
00459   }
```

### CalcDiscreteLineCf()

double Garfield::MediumMagboltz::CalcDiscreteLineCf (
const Deexcitation & dxc,
double e,
double cfOth) const [private]

### ComputeDeexcitation()

void Garfield::MediumMagboltz::ComputeDeexcitation (
int iLevel,
int & fLevel,
std::vector< Secondary > & secondaries)

### ComputeDeexcitationInternal()

void Garfield::MediumMagboltz::ComputeDeexcitationInternal (
int iLevel,
int & fLevel,
std::vector< Secondary > & secondaries) [private]

### ComputeDeexcitationTable()

void Garfield::MediumMagboltz::ComputeDeexcitationTable (
const bool verbose) [private]

### ComputePhotonCollisionTable()

bool Garfield::MediumMagboltz::ComputePhotonCollisionTable (
const bool verbose) [private]

### CreateGPUTransferObject()

double Garfield::MediumMagboltz::CreateGPUTransferObject (
MediumGPU *& med_gpu) [override], [virtual]
Create and initialise GPU Transfer class.
Reimplemented from Garfield::MediumGas.

### DisableDeexcitation()

void Garfield::MediumMagboltz::DisableDeexcitation () [inline]
Switch off (microscopic) de-excitation handling.
Definition at line 64 of file MediumMagboltz.hh.

```
00064 { m_useDeexcitation = false; }
```

### DisablePenningTransfer() [1/2]

void Garfield::MediumMagboltz::DisablePenningTransfer () [override], [virtual]
Switch the simulation of Penning transfers off globally.
Reimplemented from Garfield::MediumGas.

### DisablePenningTransfer() [2/2]

bool Garfield::MediumMagboltz::DisablePenningTransfer (
std::string gasname) [override], [virtual]
Switch the simulation of Penning transfers off for a given component.
Reimplemented from Garfield::MediumGas.

### DisableRadiationTrapping()

void Garfield::MediumMagboltz::DisableRadiationTrapping () [inline]
Switch off discrete photoabsorption levels.
Definition at line 68 of file MediumMagboltz.hh.

```
00068 { m_useRadTrap = false; }
```

### ElectronCollision()

bool Garfield::MediumMagboltz::ElectronCollision (
const double e,
int & type,
int & level,
double & e1,
double & dx,
double & dy,
double & dz,
std::vector< Secondary > & secondaries,
int & band) [override]
Sample the collision type.

### EnableAnisotropicScattering()

void Garfield::MediumMagboltz::EnableAnisotropicScattering (
const bool on = true) [inline]
Switch on/off anisotropic scattering (enabled by default)
Definition at line 49 of file MediumMagboltz.hh.

```
00049                                                {
00050     m_useAnisotropic = on;
00051     m_isChanged = true;
00052   }
```

### EnableAutoEnergyLimit()

void Garfield::MediumMagboltz::EnableAutoEnergyLimit (
const bool on = true) [inline]
Let Magboltz determine the upper energy limit (this is the default) or use the energy limit specified using SetMaxElectronEnergy).
Definition at line 164 of file MediumMagboltz.hh.

```
00164 { m_autoEnergyLimit = on; }
```

### EnableCrossSectionOutput()

void Garfield::MediumMagboltz::EnableCrossSectionOutput (
const bool on = true) [inline]

Write the gas cross-section table to a file during the initialisation.
Definition at line 81 of file MediumMagboltz.hh.

```
00081 { m_useCsOutput = on; }
```

### EnableDeexcitation()

void Garfield::MediumMagboltz::EnableDeexcitation ()

Switch on (microscopic) de-excitation handling.

### EnablePenningTransfer() [1/3]

bool Garfield::MediumMagboltz::EnablePenningTransfer () [override], [virtual]

Switch on simulation of Penning transfers, using pre-implemented parameterisations of the transfer probability (if available).
Reimplemented from Garfield::MediumGas.

### EnablePenningTransfer() [2/3]

bool Garfield::MediumMagboltz::EnablePenningTransfer (
const double r,
const double lambda) [override], [virtual]

Switch on simulation of Penning transfers by means of transfer probabilities, for all excitation levels in the mixture.

Parameters

| | |
|---|---|
| r | transfer probability [0, 1] |
| lambda | parameter for sampling the distance of the Penning electron with respect to the excitation. |

Reimplemented from Garfield::MediumGas.

### EnablePenningTransfer() [3/3]

bool Garfield::MediumMagboltz::EnablePenningTransfer (
const double r,
const double lambda,
std::string gasname) [override], [virtual]

Switch on simulation of Penning transfers by means of transfer probabilities, for all excitations of a given component.
Reimplemented from Garfield::MediumGas.

### EnableRadiationTrapping()

void Garfield::MediumMagboltz::EnableRadiationTrapping ()

Switch on discrete photoabsorption levels.

### EnableThermalMotion()

void Garfield::MediumMagboltz::EnableThermalMotion (
const bool on = true) [inline]

Take the thermal motion of the gas at the selected temperature into account in the calculations done by Magboltz.
By the default, this feature is off (static gas at 0 K).
Definition at line 161 of file MediumMagboltz.hh.

```
00161 { m_useGasMotion = on; }
```

### GenerateGasTable()

```
void Garfield::MediumMagboltz::GenerateGasTable (
        const int numCollisions = 10,
        const bool verbose = true)
```

Generate a new gas table (can later be saved to file) by running Magboltz for all electric fields, magnetic fields, and angles in the currently set grid.

### GetElectronCollisionRate() [1/2]

```
double Garfield::MediumMagboltz::GetElectronCollisionRate (
        const double e,
        const int band)  [override]
```

Get the (real) collision rate [ns-1] at a given electron energy e [eV].

### GetElectronCollisionRate() [2/2]

```
double Garfield::MediumMagboltz::GetElectronCollisionRate (
        const double e,
        const unsigned int level,
        const int band)
```

Get the collision rate [ns-1] for a specific level.

### GetElectronNullCollisionRate()

```
double Garfield::MediumMagboltz::GetElectronNullCollisionRate (
        const int band)  [override]
```

### GetExcitationIonisationLevels()

```
void Garfield::MediumMagboltz::GetExcitationIonisationLevels ()  [private]
```

### GetGasNumberMagboltz()

```
int Garfield::MediumMagboltz::GetGasNumberMagboltz (
        const std::string & input)  [static]
```

### GetLevel()

```
bool Garfield::MediumMagboltz::GetLevel (
        const unsigned int i,
        int & ngas,
        int & type,
        std::string & descr,
        double & e)
```

Get detailed information about a given cross-section term i.

### GetMaxElectronEnergy()

```
double Garfield::MediumMagboltz::GetMaxElectronEnergy () const  [inline]
```

Get the highest electron energy in the table of scattering rates.
Definition at line 40 of file MediumMagboltz.hh.

```
00040 { return m_eMax; }
```

### GetMaxPhotonEnergy()

```
double Garfield::MediumMagboltz::GetMaxPhotonEnergy () const  [inline]
```

Get the highest photon energy in the table of scattering rates.
Definition at line 46 of file MediumMagboltz.hh.

```
00046 { return m_eFinalGamma; }
```

### GetNumberOfElectronCollisions() [1/3]

unsigned int Garfield::MediumMagboltz::GetNumberOfElectronCollisions () const
Get the total number of electron collisions.

### GetNumberOfElectronCollisions() [2/3]

unsigned int Garfield::MediumMagboltz::GetNumberOfElectronCollisions (
const unsigned int level) const
Get the number of collisions for a specific cross-section term.

### GetNumberOfElectronCollisions() [3/3]

unsigned int Garfield::MediumMagboltz::GetNumberOfElectronCollisions (
unsigned int & nElastic,
unsigned int & nIonising,
unsigned int & nAttachment,
unsigned int & nInelastic,
unsigned int & nExcitation,
unsigned int & nSuperelastic) const
Get the number of collisions broken down by cross-section type.

### GetNumberOfLevels()

unsigned int Garfield::MediumMagboltz::GetNumberOfLevels ()
Get the number of cross-section terms.

### GetNumberOfPenningTransfers()

unsigned int Garfield::MediumMagboltz::GetNumberOfPenningTransfers () const [inline]
Get the number of Penning transfers that occured since the last reset.
Definition at line 149 of file MediumMagboltz.hh.

```
00149 { return m_nPenning; }
```

### GetNumberOfPhotonCollisions() [1/2]

unsigned int Garfield::MediumMagboltz::GetNumberOfPhotonCollisions () const
Get the total number of photon collisions.

### GetNumberOfPhotonCollisions() [2/2]

unsigned int Garfield::MediumMagboltz::GetNumberOfPhotonCollisions (
unsigned int & nElastic,
unsigned int & nIonising,
unsigned int & nInelastic) const
Get number of photon collisions by collision type.

### GetPenningTransfer()

bool Garfield::MediumMagboltz::GetPenningTransfer (
const unsigned int i,
double & r,
double & lambda)
Get the Penning transfer probability and distance of a specific level.

### GetPhotonCollisionRate()

double Garfield::MediumMagboltz::GetPhotonCollisionRate (
const double e) [override]

### Initialise()

bool Garfield::MediumMagboltz::Initialise (
const bool verbose = false)

Initialise the table of scattering rates (called internally when a collision rate is requested and the gas mixture or other parameters have changed).

### Mixer()

bool Garfield::MediumMagboltz::Mixer (
const bool verbose = false) [private]

### PhotonCollision()

bool Garfield::MediumMagboltz::PhotonCollision (
const double e,
int & type,
int & level,
double & e1,
double & ctheta,
std::vector< Secondary > & secondaries) [override]

### PlotElectronCollisionRates()

void Garfield::MediumMagboltz::PlotElectronCollisionRates (
TPad * pad)

### PlotElectronCrossSections()

void Garfield::MediumMagboltz::PlotElectronCrossSections (
const unsigned int i,
TPad * pad)

### PlotElectronInverseMeanFreePath()

void Garfield::MediumMagboltz::PlotElectronInverseMeanFreePath (
TPad * pad)

### PrintGas()

void Garfield::MediumMagboltz::PrintGas () [override], [virtual]

Print information about the present gas mixture and available data.
Reimplemented from Garfield::MediumGas.

### RateConstantHardSphere()

double Garfield::MediumMagboltz::RateConstantHardSphere (
const double r1,
const double r2,
const int igas1,
const int igas2) const [private]

### RateConstantWK()

double Garfield::MediumMagboltz::RateConstantWK (
const double energy,
const double osc,
const double pacs,
const int igas1,
const int igas2) const [private]

### ResetCollisionCounters()

void Garfield::MediumMagboltz::ResetCollisionCounters ()

Reset the collision counters.

### RunMagboltz()

void Garfield::MediumMagboltz::RunMagboltz (
const double e,
const double b,
const double btheta,
const int ncoll,
bool verbose,
double & vx,
double & vy,
double & vz,
double & wv,
double & wr,
double & dl,
double & dt,
double & alpha,
double & eta,
double & riontof,
double & ratttof,
double & lor,
double & vxerr,
double & vyerr,
double & vzerr,
double & wverr,
double & wrerr,
double & dlerr,
double & dterr,
double & alphaerr,
double & etaerr,
double & riontoferr,
double & ratttoferr,
double & lorerr,
double & alphatof,
std::array< double, 6 > & difftens)

Run Magboltz for a given electric field, magnetic field and angle.

Parameters

| | | |
|---|---|---|
| in | e | electric field |
| in | b | magnetic field |
| in | btheta | angle between electric and magnetic field |
| in | ncoll | number of collisions (in multiples of $10^7$) to be simulated |
| in | verbose | verbosity flag |
| out | vx,vy,vz | drift velocity vector |
| out | wv,wr | flux and bulk drift velocity |
| out | dl,dt | diffusion cofficients |
| out | alpha | Townsend cofficient |
| out | eta | attachment cofficient |
| out | riontof | TOF ionisation rate |
| out | ratttof | TOF attachment rate |
| out | lor | Lorentz angle |

Parameters

| | | |
|---|---|---|
| out | vxerr,vyerr,vzerr | errors on drift velocity |
| out | wverr,wrerr | errors on TOF drifts |
| out | dlerr,dterr | errors on diffusion coefficients |
| out | alphaerr,etaerr | errors on Townsend/attachment coefficients |
| out | riontoferr,ratttoferr | errors on TOF rates |
| out | lorerr | error on Lorentz angle |
| out | alphatof | effective Townsend coefficient ($\alpha - \eta$) calculated using time-of-flight method |
| out | difftens | components of the diffusion tensor (zz, xx, yy, xz, yz, xy) |

### SetExcitationScaling()

void Garfield::MediumMagboltz::SetExcitationScaling (
const double r,
std::string gasname)

Multiply all excitation cross-sections by a uniform scaling factor.

### SetLineWidth()

void Garfield::MediumMagboltz::SetLineWidth (
const double n) [inline]

Set the number of emission line widths within which to apply absorption by discrete lines.
Definition at line 71 of file MediumMagboltz.hh.

00071 { m_nAbsWidths = n; }

### SetMaxElectronEnergy()

bool Garfield::MediumMagboltz::SetMaxElectronEnergy (
const double e)

Set the highest electron energy to be included in the table of scattering rates.

### SetMaxPhotonEnergy()

bool Garfield::MediumMagboltz::SetMaxPhotonEnergy (
const double e)

Set the highest photon energy to be included in the table of scattering rates.

### SetSplittingFunctionFlat()

void Garfield::MediumMagboltz::SetSplittingFunctionFlat ()

Sample the secondary electron energy from a flat distribution.

### SetSplittingFunctionGreenSawada()

void Garfield::MediumMagboltz::SetSplittingFunctionGreenSawada ()

Sample the secondary electron energy according to the Green-Sawada model.

### SetSplittingFunctionOpalBeaty()

void Garfield::MediumMagboltz::SetSplittingFunctionOpalBeaty ()

Sample the secondary electron energy according to the Opal-Beaty model.

### SetupGreenSawada()

void Garfield::MediumMagboltz::SetupGreenSawada () [private]

### Update()

bool Garfield::MediumMagboltz::Update (
const bool verbose = false) [private]

## 7.66.4 Member Data Documentation

### csTypeGamma

std::vector<int> Garfield::MediumMagboltz::csTypeGamma [private]
Definition at line 434 of file MediumMagboltz.hh.

### DxcTypeCollIon

const int Garfield::MediumMagboltz::DxcTypeCollIon [static], [private]
Definition at line 225 of file MediumMagboltz.hh.

### DxcTypeCollNonIon

const int Garfield::MediumMagboltz::DxcTypeCollNonIon [static], [private]
Definition at line 226 of file MediumMagboltz.hh.

### DxcTypeRad

const int Garfield::MediumMagboltz::DxcTypeRad [static], [private]
Definition at line 224 of file MediumMagboltz.hh.

### m_autoEnergyLimit

bool Garfield::MediumMagboltz::m_autoEnergyLimit = true [private]
Automatic calculation of the energy limit by Magboltz or not.
Definition at line 234 of file MediumMagboltz.hh.

### m_cf

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_cf [private]
Definition at line 332 of file MediumMagboltz.hh.

### m_cfGamma

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_cfGamma [private]
Definition at line 433 of file MediumMagboltz.hh.

### m_cfLog

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_cfLog [private]
Definition at line 333 of file MediumMagboltz.hh.

### m_cfNull

double Garfield::MediumMagboltz::m_cfNull = 0. [private]
Null-collision frequency.
Definition at line 338 of file MediumMagboltz.hh.

### m_cfTot

std::vector<double> Garfield::MediumMagboltz::m_cfTot [private]
Definition at line 329 of file MediumMagboltz.hh.

### m_cfTotGamma

std::vector<double> Garfield::MediumMagboltz::m_cfTotGamma [private]
Definition at line 431 of file MediumMagboltz.hh.

### m_cfTotLog

std::vector<double> Garfield::MediumMagboltz::m_cfTotLog [private]
Definition at line 330 of file MediumMagboltz.hh.

### m_csType

std::array<int, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_csType [private]
Cross-section type.
Definition at line 305 of file MediumMagboltz.hh.

### m_deexcitations

std::vector<Deexcitation> Garfield::MediumMagboltz::m_deexcitations [private]
Definition at line 410 of file MediumMagboltz.hh.

### m_description

std::vector<std::string> Garfield::MediumMagboltz::m_description [private]
Level description.
Definition at line 326 of file MediumMagboltz.hh.

### m_eAuger1

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_eAuger1 [private]
Energy imparted to Auger electrons.
Definition at line 313 of file MediumMagboltz.hh.

### m_eAuger2

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_eAuger2 [private]
Definition at line 314 of file MediumMagboltz.hh.

### m_eFinalGamma

double Garfield::MediumMagboltz::m_eFinalGamma [private]
Definition at line 427 of file MediumMagboltz.hh.

### m_eFluorescence

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_eFluorescence [private]
Definition at line 316 of file MediumMagboltz.hh.

### m_eHigh

double Garfield::MediumMagboltz::m_eHigh [private]
Definition at line 243 of file MediumMagboltz.hh.

### m_eHighLog

double Garfield::MediumMagboltz::m_eHighLog [private]
Definition at line 243 of file MediumMagboltz.hh.

### m_eMax

double Garfield::MediumMagboltz::m_eMax [private]
Max. electron energy in the collision rate tables.
Definition at line 238 of file MediumMagboltz.hh.

m_energyLoss

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_energyLoss [private]
Energy loss.
Definition at line 303 of file MediumMagboltz.hh.

m_eStep

double Garfield::MediumMagboltz::m_eStep [private]
Energy spacing in the linear part of the collision rate tables.
Definition at line 240 of file MediumMagboltz.hh.

m_eStepGamma

double Garfield::MediumMagboltz::m_eStepGamma [private]
Definition at line 427 of file MediumMagboltz.hh.

m_eStepInv

double Garfield::MediumMagboltz::m_eStepInv [private]
Inverse energy spacing.
Definition at line 242 of file MediumMagboltz.hh.

m_hasGreenSawada

std::array<bool, m_nMaxGases> Garfield::MediumMagboltz::m_hasGreenSawada [private]
Definition at line 261 of file MediumMagboltz.hh.

m_iDeexcitation

std::array<int, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_iDeexcitation [private]
Definition at line 412 of file MediumMagboltz.hh.

m_ionPot

std::array<double, m_nMaxGases> Garfield::MediumMagboltz::m_ionPot [private]
Ionisation potentials of each component.
Definition at line 419 of file MediumMagboltz.hh.

m_lambdaPenning

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_lambdaPenning [private]
Mean distance of Penning ionisation (by level)
Definition at line 371 of file MediumMagboltz.hh.

m_lnStep

double Garfield::MediumMagboltz::m_lnStep [private]
Definition at line 244 of file MediumMagboltz.hh.

m_mgas

std::array<double, m_nMaxGases> Garfield::MediumMagboltz::m_mgas [private]
Mass.
Definition at line 252 of file MediumMagboltz.hh.

m_minIonPot

double Garfield::MediumMagboltz::m_minIonPot = -1. [private]
Minimum ionisation potential.
Definition at line 421 of file MediumMagboltz.hh.

m_mutex

std::mutex Garfield::MediumMagboltz::m_mutex [private]
Mutex.
Definition at line 229 of file MediumMagboltz.hh.

m_nAbsWidths

double Garfield::MediumMagboltz::m_nAbsWidths = 1000. [private]
Definition at line 416 of file MediumMagboltz.hh.

m_nAuger1

std::array<unsigned int, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_nAuger1 [private]
Number of Auger electrons produced in a collision.
Definition at line 310 of file MediumMagboltz.hh.

m_nAuger2

std::array<unsigned int, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_nAuger2 [private]
Definition at line 311 of file MediumMagboltz.hh.

m_nCollisions

std::array<unsigned int, nCsTypes> Garfield::MediumMagboltz::m_nCollisions [private]
Collision counters 0: elastic 1: ionisation 2: attachment 3: inelastic 4: excitation 5: super-elastic.
Definition at line 363 of file MediumMagboltz.hh.

m_nCollisionsDetailed

std::vector<unsigned int> Garfield::MediumMagboltz::m_nCollisionsDetailed [private]
Number of collisions for each cross-section term.
Definition at line 365 of file MediumMagboltz.hh.

m_nFluorescence

std::array<unsigned int, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_nFluorescence [private]
Definition at line 315 of file MediumMagboltz.hh.

m_nPenning

unsigned int Garfield::MediumMagboltz::m_nPenning = 0 [private]
Number of Penning ionisations.
Definition at line 373 of file MediumMagboltz.hh.

m_nPhotonCollisions

std::array<unsigned int, nCsTypesGamma> Garfield::MediumMagboltz::m_nPhotonCollisions [private]
Definition at line 440 of file MediumMagboltz.hh.

m_nPhotonTerms

unsigned int Garfield::MediumMagboltz::m_nPhotonTerms = 0 [private]
Definition at line 429 of file MediumMagboltz.hh.

m_nTerms

unsigned int Garfield::MediumMagboltz::m_nTerms = 0 [private]
Number of different cross-section types in the current gas mixture.
Definition at line 249 of file MediumMagboltz.hh.

### m_parGreenSawada

std::array<std::array<double, 5>, m_nMaxGases> Garfield::MediumMagboltz::m_parGreenSawada [private]
Green-Sawada splitting parameters [eV] (Γs, Γb, Ts, Ta, Tb).
Definition at line 260 of file MediumMagboltz.hh.

### m_rgas

std::array<double, m_nMaxGases> Garfield::MediumMagboltz::m_rgas [private]
Recoil energy parameter.
Definition at line 254 of file MediumMagboltz.hh.

### m_rPenning

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_rPenning [private]
Penning transfer probability (by level)
Definition at line 369 of file MediumMagboltz.hh.

### m_s2

std::array<double, m_nMaxGases> Garfield::MediumMagboltz::m_s2 [private]
Definition at line 255 of file MediumMagboltz.hh.

### m_scaleExc

std::array<double, m_nMaxGases> Garfield::MediumMagboltz::m_scaleExc [private]
Definition at line 424 of file MediumMagboltz.hh.

### m_scatCut

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_scatCut [private]
Definition at line 320 of file MediumMagboltz.hh.

### m_scatCutLog

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_scatCutLog [private]
Definition at line 322 of file MediumMagboltz.hh.

### m_scatModel

std::array<int, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_scatModel [private]
Definition at line 323 of file MediumMagboltz.hh.

### m_scatPar

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_scatPar [private]
Definition at line 319 of file MediumMagboltz.hh.

### m_scatParLog

std::vector<std::vector<double> > Garfield::MediumMagboltz::m_scatParLog [private]
Definition at line 321 of file MediumMagboltz.hh.

### m_useAnisotropic

bool Garfield::MediumMagboltz::m_useAnisotropic = true [private]
Definition at line 336 of file MediumMagboltz.hh.

### m_useCsOutput

bool Garfield::MediumMagboltz::m_useCsOutput = false [private]
Flag enabling/disabling output of cross-section table to file.
Definition at line 247 of file MediumMagboltz.hh.

m_useDeexcitation

bool Garfield::MediumMagboltz::m_useDeexcitation = false [private]
Flag enabling/disabling detailed simulation of de-excitation process.
Definition at line 377 of file MediumMagboltz.hh.

m_useGasMotion

bool Garfield::MediumMagboltz::m_useGasMotion = false [private]
Simulate thermal motion of the gas or not (when running Magboltz).
Definition at line 232 of file MediumMagboltz.hh.

m_useGreenSawada

bool Garfield::MediumMagboltz::m_useGreenSawada = false [private]
Sample secondary electron energies using Green-Sawada parameterisation.
Definition at line 266 of file MediumMagboltz.hh.

m_useOpalBeaty

bool Garfield::MediumMagboltz::m_useOpalBeaty = true [private]
Sample secondary electron energies using Opal-Beaty parameterisation.
Definition at line 264 of file MediumMagboltz.hh.

m_useRadTrap

bool Garfield::MediumMagboltz::m_useRadTrap = true [private]
Flag enabling/disable radiation trapping (absorption of photons discrete excitation lines)
Definition at line 380 of file MediumMagboltz.hh.

m_wOpalBeaty

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_wOpalBeaty [private]
Opal-Beaty-Peterson splitting parameter [eV].
Definition at line 257 of file MediumMagboltz.hh.

m_yFluorescence

std::array<double, Magboltz::nMaxLevels> Garfield::MediumMagboltz::m_yFluorescence [private]
Fluorescence yield.
Definition at line 308 of file MediumMagboltz.hh.

nCsTypes

int Garfield::MediumMagboltz::nCsTypes = 7 [static], [constexpr], [private]
Definition at line 219 of file MediumMagboltz.hh.

nCsTypesGamma

int Garfield::MediumMagboltz::nCsTypesGamma = 4 [static], [constexpr], [private]
Definition at line 222 of file MediumMagboltz.hh.

nEnergyStepsGamma

int Garfield::MediumMagboltz::nEnergyStepsGamma = 5000 [static], [constexpr], [private]
Definition at line 218 of file MediumMagboltz.hh.

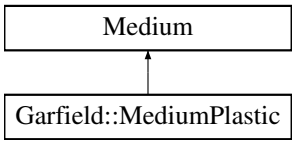
nEnergyStepsLog

int Garfield::MediumMagboltz::nEnergyStepsLog = 1000 [static], [constexpr], [private]
Definition at line 217 of file MediumMagboltz.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumMagboltz.hh

## 7.67 Garfield::MediumPlastic Class Reference

Plastic medium.
#include <MediumPlastic.hh>
Inheritance diagram for Garfield::MediumPlastic:




Public Member Functions

- MediumPlastic ()
  Default constructor.
- MediumPlastic (const double eps)
  Constructor from dielectric constant.
- virtual ~MediumPlastic ()
  Destructor.
- void EnableDrift (const bool) override
- void EnablePrimaryIonisation (const bool) override

### 7.67.1 Detailed Description

Plastic medium.
Definition at line 10 of file MediumPlastic.hh.

### 7.67.2 Constructor & Destructor Documentation

#### MediumPlastic() [1/2]

Garfield::MediumPlastic::MediumPlastic () [inline]
Default constructor.
Definition at line 13 of file MediumPlastic.hh.

```
00013 : MediumPlastic(1.) {}
```

#### MediumPlastic() [2/2]

Garfield::MediumPlastic::MediumPlastic (
const double eps) [inline]
Constructor from dielectric constant.
Definition at line 15 of file MediumPlastic.hh.

```
00015                                 : Medium() {
00016     m_className = "MediumPlastic";
00017     m_name = "Plastic";
00018     if (eps > 1.) m_epsilon = eps;
00019   }
```

#### ~MediumPlastic()

virtual Garfield::MediumPlastic::~MediumPlastic () [inline], [virtual]
Destructor.
Definition at line 21 of file MediumPlastic.hh.

```
00021 {}
```

### 7.67.3 Member Function Documentation

EnableDrift()

void Garfield::MediumPlastic::EnableDrift (
const bool ) [inline], [override]
Definition at line 23 of file MediumPlastic.hh.
00023 {}

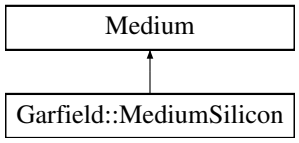
EnablePrimaryIonisation()

void Garfield::MediumPlastic::EnablePrimaryIonisation (
const bool ) [inline], [override]
Definition at line 24 of file MediumPlastic.hh.
00024 {}
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumPlastic.hh

## 7.68 Garfield::MediumSilicon Class Reference

Solid crystalline silicon
#include <MediumSilicon.hh>
Inheritance diagram for Garfield::MediumSilicon:




Classes

- struct Band

Public Member Functions

- MediumSilicon ()
  Constructor.
- virtual ~MediumSilicon ()
  Destructor.
- bool IsSemiconductor () const override
- void SetDoping (const char type, const double c)
  Set doping concentration [cm-3] and type ('i', 'n', 'p').
- void GetDoping (char &type, double &c) const
  Retrieve doping concentration.
- void SetTrapCrossSection (const double ecs, const double hcs)
  Trapping cross-sections for electrons and holes.
- void SetTrapDensity (const double n)
  Trap density [cm-3], by default set to zero.
- void SetTrappingTime (const double etau, const double htau)
  Set time constant for trapping of electrons and holes [ns].
- void SetLowFieldMobility (const double mue, const double muh)
  Specify the low field values of the electron and hole mobilities.
- void SetLatticeMobilityModel (const std::string &model)
  Set the parameterisation to be used for calculating the lattice mobility model.
- void SetLatticeMobilityModelMinimos ()

Calculate the lattice mobility using the Minimos model.

- void SetLatticeMobilityModelSentaurus ()

  Calculate the lattice mobility using the Sentaurus model (default).
- void SetLatticeMobilityModelReggiani ()

  Calculate the lattice mobility using the Reggiani model.
- void SetDopingMobilityModelMinimos ()

  Use the Minimos model for the doping-dependence of the mobility.
- void SetDopingMobilityModelMasetti ()

  Use the Masetti model for the doping-dependence of the mobility (default).
- void SetSaturationVelocity (const double vsate, const double vsath)

  Specify the saturation velocities of electrons and holes.
- void SetSaturationVelocityModelMinimos ()

  Calculate the saturation velocities using the Minimos model.
- void SetSaturationVelocityModelCanali ()

  Calculate the saturation velocities using the Canali model (default).
- void SetSaturationVelocityModelReggiani ()

  Calculate the saturation velocities using the Reggiani model.
- void SetHighFieldMobilityModel (const std::string &model)

  Set the parameterisation to be used for the drift velocity as function of the electric field.
- void SetHighFieldMobilityModelMinimos ()

  Parameterize the high-field mobility using the Minimos model.
- void SetHighFieldMobilityModelCanali ()

  Parameterize the high-field mobility using the Canali model (default).
- void SetHighFieldMobilityModelReggiani ()

  Parameterize the high-field mobility using the Reggiani model.
- void SetHighFieldMobilityModelConstant ()

  Make the velocity proportional to the electric field (no saturation).
- void SetImpactIonisationModel (const std::string &model)

  Set the parameterisation to be used for calculating the impact ionisation coefficient.
- void SetImpactIonisationModelVanOverstraetenDeMan ()

  Calculate $\alpha$ using the van Overstraeten-de Man model (default).
- void SetImpactIonisationModelGrant ()

  Calculate $\alpha$ using the Grant model.
- void SetImpactIonisationModelMassey ()

  Calculate $\alpha$ using the Massey model.
- void SetImpactIonisationModelOkutoCrowell ()

  Calculate $\alpha$ using the Okuto-Crowell model.
- void SetDiffusionScaling (const double d)

  Apply a scaling factor to the diffusion coefficients.
- bool SetMaxElectronEnergy (const double e)
- double GetMaxElectronEnergy () const
- bool Initialise ()
- void EnableScatteringRateOutput (const bool on=true)
- void EnableNonParabolicity (const bool on=true)
- void EnableFullBandDensityOfStates (const bool on=true)
- void EnableAnisotropy (const bool on=true)
- bool ElectronVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool ElectronTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool ElectronAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override

- double ElectronMobility () override
- bool HoleVelocity (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &vx, double &vy, double &vz) override
- bool HoleTownsend (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &alpha) override
- bool HoleAttachment (const double ex, const double ey, const double ez, const double bx, const double by, const double bz, double &eta) override
- double HoleMobility () override
- double GetElectronEnergy (const double px, const double py, const double pz, double &vx, double &vy, double &vz, const int band=0) override
- void GetElectronMomentum (const double e, double &px, double &py, double &pz, int &band) override
- double GetElectronNullCollisionRate (const int band) override
- double GetElectronCollisionRate (const double e, const int band) override
- bool ElectronCollision (const double e, int &type, int &level, double &e1, double &dx, double &dy, double &dz, std::vector< Secondary > &secondaries, int &band) override
- void ResetCollisionCounters ()
- unsigned int GetNumberOfElectronCollisions () const
- unsigned int GetNumberOfLevels () const
- unsigned int GetNumberOfElectronCollisions (const unsigned int level) const
- unsigned int GetNumberOfElectronBands () const
- int GetElectronBandPopulation (const int band)
- bool GetOpticalDataRange (double &emin, double &emax, const unsigned int i=0) override
- bool GetDielectricFunction (const double e, double &eps1, double &eps2, const unsigned int i=0) override
- void ComputeSecondaries (const double e0, double &ee, double &eh)

Private Types

- enum class LatticeMobility { Sentaurus = 0 , Minimos , Reggiani }
- enum class DopingMobility { Minimos = 0 , Masetti }
- enum class SaturationVelocity { Minimos = 0 , Canali , Reggiani }
- enum class HighFieldMobility { Minimos = 0 , Canali , Reggiani , Constant }
- enum class ImpactIonisation { VanOverstraeten = 0 , Grant , Massey , Okuto }

Private Member Functions

- bool Update ()
- void UpdateLatticeMobility ()
- void UpdateDopingMobilityMinimos ()
- void UpdateDopingMobilityMasetti ()
- void UpdateSaturationVelocity ()
- void UpdateHighFieldMobilityCanali ()
- void UpdateImpactIonisation ()
- double ElectronMobility (const double e) const
- double ElectronAlpha (const double e) const
- double HoleMobility (const double e) const
- double HoleAlpha (const double e) const
- bool LoadOpticalData (const std::string &filename)
- bool ElectronScatteringRates ()
- bool HoleScatteringRates ()
- bool AcousticScatteringRates (const double rho, const double kbt, const double dp, Band &band)
- bool OpticalScatteringRates (const double rho, const double kbt, const double dtk, const double eph, Band &band)
- bool IntervalleyScatteringRates (const double rho, const double kbt, const double dtk, const double eph, Band &bndI, Band &bndF, const double zF, const int collType)

- bool IonisationRates (const std::vector< double > &p, const std::vector< double > &eth, const std::vector< double > &b, Band &band)
- bool ImpurityScatteringRates (const double kbt, Band &band)
- void InitialiseDOS ()
- void ComputeDOS ()

Private Attributes

- std::mutex m_mutex
- double m_diffScale = 1.
- double m_bandGap = 1.12
- char m_dopingType = 'i'
- double m_cDop = 0.
- double m_eMuLat = 1.35e-6
- double m_hMuLat = 0.45e-6
- double m_eMu = 1.35e-6
- double m_hMu = 0.45e-6
- double m_eBetaCanali = 1.109
- double m_hBetaCanali = 1.213
- double m_eBetaCanaliInv = 1. / 1.109
- double m_hBetaCanaliInv = 1. / 1.213
- double m_eVs = 1.02e-2
- double m_hVs = 0.72e-2
- double m_eRs = 1.35e-6 / 1.02e-2
- double m_hRs = 0.45e-6 / 0.72e-2
- double m_eHallFactor = 1.15
- double m_hHallFactor = 0.7
- double m_eTrapCs = 1.e-15
- double m_hTrapCs = 1.e-15
- double m_eTrapDensity = 0.
- double m_hTrapDensity = 0.
- double m_eTrapTime = 0.
- double m_hTrapTime = 0.
- double m_eTrapRate = 0.
- double m_hTrapRate = 0.
- int m_trappingModel = 0
- double m_eImpactA0 = 3.318e5
- double m_eImpactA1 = 0.703e6
- double m_eImpactA2 = 0.
- double m_eImpactB0 = 1.135e6
- double m_eImpactB1 = 1.231e6
- double m_eImpactB2 = 0.
- double m_hImpactA0 = 1.582e6
- double m_hImpactA1 = 0.671e6
- double m_hImpactB0 = 2.036e6
- double m_hImpactB1 = 1.693e6
- bool m_hasUserMobility = false
- bool m_hasUserSaturationVelocity = false
- LatticeMobility m_latticeMobilityModel = LatticeMobility::Sentaurus
- DopingMobility m_dopingMobilityModel = DopingMobility::Masetti
- SaturationVelocity m_saturationVelocityModel = SaturationVelocity::Canali
- HighFieldMobility m_highFieldMobilityModel = HighFieldMobility::Canali
- ImpactIonisation m_impactIonisationModel = ImpactIonisation::VanOverstraeten
- bool m_cfOutput = false
- bool m_nonParabolic = true

- bool m_fullBandDos = false
- bool m_anisotropic = true
- std::array< Band, 3 > m_cb
- std::vector< size_t > m_cbIndex
- Band m_vb
- unsigned int m_nCollElectronAcoustic = 0
- unsigned int m_nCollElectronOptical = 0
- unsigned int m_nCollElectronIntervalley = 0
- unsigned int m_nCollElectronImpurity = 0
- unsigned int m_nCollElectronIonisation = 0
- std::vector< unsigned int > m_nCollElectronDetailed
- std::vector< unsigned int > m_nCollElectronBand
- double m_eStepDos = 0.
- double m_invStepDos = 0.
- std::vector< double > m_fbDosV
- std::vector< double > m_fbDosC
- double m_fbDosMaxV
- double m_fbDosMaxC
- std::string m_opticalDataFile = "OpticalData_Si.txt"
- std::vector< double > m_egamma
- std::vector< double > m_eps1
- std::vector< double > m_eps2

### 7.68.1 Detailed Description

Solid crystalline silicon
Definition at line 14 of file MediumSilicon.hh.

### 7.68.2 Member Enumeration Documentation

#### DopingMobility

enum class Garfield::MediumSilicon::DopingMobility [strong], [private]

Enumerator

| | |
|---|---|
| Minimos | |
| Masetti | |

Definition at line 181 of file MediumSilicon.hh.
00181 { Minimos = 0, Masetti };

#### HighFieldMobility

enum class Garfield::MediumSilicon::HighFieldMobility [strong], [private]

Enumerator

| | |
|---|---|
| Minimos | |
| Canali | |
| Reggiani | |
| Constant | |

Definition at line 183 of file MediumSilicon.hh.
00183 { Minimos = 0, Canali, Reggiani, Constant };

#### ImpactIonisation

enum class Garfield::MediumSilicon::ImpactIonisation [strong], [private]

Enumerator

| | |
|---|---|
| VanOverstraeten | |
| Grant | |
| Massey | |
| Okuto | |

Definition at line 184 of file MediumSilicon.hh.
00184 { VanOverstraeten = 0, Grant, Massey, Okuto };

LatticeMobility

enum class Garfield::MediumSilicon::LatticeMobility [strong], [private]

Enumerator

| | |
|---|---|
| Sentaurus | |
| Minimos | |
| Reggiani | |

Definition at line 180 of file MediumSilicon.hh.
00180 { Sentaurus = 0, Minimos, Reggiani };

SaturationVelocity

enum class Garfield::MediumSilicon::SaturationVelocity [strong], [private]

Enumerator

| | |
|---|---|
| Minimos | |
| Canali | |
| Reggiani | |

Definition at line 182 of file MediumSilicon.hh.
00182 { Minimos = 0, Canali, Reggiani };

### 7.68.3 Constructor & Destructor Documentation

MediumSilicon()

Garfield::MediumSilicon::MediumSilicon ()
Constructor.

~MediumSilicon()

virtual Garfield::MediumSilicon::~MediumSilicon () [inline], [virtual]
Destructor.
Definition at line 19 of file MediumSilicon.hh.
00019 {}

### 7.68.4 Member Function Documentation

AcousticScatteringRates()

bool Garfield::MediumSilicon::AcousticScatteringRates (
const double rho,
const double kbt,
const double dp,
Band & band) [private]

### ComputeDOS()

void Garfield::MediumSilicon::ComputeDOS () [private]

### ComputeSecondaries()

void Garfield::MediumSilicon::ComputeSecondaries (
const double e0,
double & ee,
double & eh)

### ElectronAlpha()

double Garfield::MediumSilicon::ElectronAlpha (
const double e) const [private]

### ElectronAttachment()

bool Garfield::MediumSilicon::ElectronAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

### ElectronCollision()

bool Garfield::MediumSilicon::ElectronCollision (
const double e,
int & type,
int & level,
double & e1,
double & dx,
double & dy,
double & dz,
std::vector< Secondary > & secondaries,
int & band) [override]

### ElectronMobility() [1/2]

double Garfield::MediumSilicon::ElectronMobility () [inline], [override]
Definition at line 128 of file MediumSilicon.hh.
00128 { return m_eMu; }

### ElectronMobility() [2/2]

double Garfield::MediumSilicon::ElectronMobility (
const double e) const [private]

### ElectronScatteringRates()

bool Garfield::MediumSilicon::ElectronScatteringRates () [private]

### ElectronTownsend()

bool Garfield::MediumSilicon::ElectronTownsend (
const double ex,
const double ey,
const double ez,

const double bx,
const double by,
const double bz,
double & alpha) [override]

### ElectronVelocity()

bool Garfield::MediumSilicon::ElectronVelocity (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & vx,
double & vy,
double & vz) [override]

### EnableAnisotropy()

void Garfield::MediumSilicon::EnableAnisotropy (
const bool on = true) [inline]

Definition at line 116 of file MediumSilicon.hh.

```
00116 { m_anisotropic = on; }
```

### EnableFullBandDensityOfStates()

void Garfield::MediumSilicon::EnableFullBandDensityOfStates (
const bool on = true) [inline]

Definition at line 113 of file MediumSilicon.hh.

```
00113                                                 {
00114     m_fullBandDos = on;
00115   }
```

### EnableNonParabolicity()

void Garfield::MediumSilicon::EnableNonParabolicity (
const bool on = true) [inline]

Definition at line 112 of file MediumSilicon.hh.

```
00112 { m_nonParabolic = on; }
```

### EnableScatteringRateOutput()

void Garfield::MediumSilicon::EnableScatteringRateOutput (
const bool on = true) [inline]

Definition at line 111 of file MediumSilicon.hh.

```
00111 { m_cfOutput = on; }
```

### GetDielectricFunction()

bool Garfield::MediumSilicon::GetDielectricFunction (
const double e,
double & eps1,
double & eps2,
const unsigned int i = 0) [override]

### GetDoping()

void Garfield::MediumSilicon::GetDoping (
char & type,
double & c) const

Retrieve doping concentration.

### GetElectronBandPopulation()

```
int Garfield::MediumSilicon::GetElectronBandPopulation (
        const int band)
```

### GetElectronCollisionRate()

```
double Garfield::MediumSilicon::GetElectronCollisionRate (
        const double e,
        const int band)    [override]
```

### GetElectronEnergy()

```
double Garfield::MediumSilicon::GetElectronEnergy (
        const double px,
        const double py,
        const double pz,
        double & vx,
        double & vy,
        double & vz,
        const int band = 0)    [override]
```

### GetElectronMomentum()

```
void Garfield::MediumSilicon::GetElectronMomentum (
        const double e,
        double & px,
        double & py,
        double & pz,
        int & band)    [override]
```

### GetElectronNullCollisionRate()

```
double Garfield::MediumSilicon::GetElectronNullCollisionRate (
        const int band)    [override]
```

### GetMaxElectronEnergy()

```
double Garfield::MediumSilicon::GetMaxElectronEnergy () const    [inline]
```

Definition at line 105 of file MediumSilicon.hh.

```
00105 { return m_cb[2].eFinal; }
```

### GetNumberOfElectronBands()

```
unsigned int Garfield::MediumSilicon::GetNumberOfElectronBands () const
```

### GetNumberOfElectronCollisions() [1/2]

```
unsigned int Garfield::MediumSilicon::GetNumberOfElectronCollisions () const
```

### GetNumberOfElectronCollisions() [2/2]

```
unsigned int Garfield::MediumSilicon::GetNumberOfElectronCollisions (
        const unsigned int level) const
```

### GetNumberOfLevels()

```
unsigned int Garfield::MediumSilicon::GetNumberOfLevels () const
```

### GetOpticalDataRange()

bool Garfield::MediumSilicon::GetOpticalDataRange (
double & emin,
double & emax,
const unsigned int i = 0) [override]

### HoleAlpha()

double Garfield::MediumSilicon::HoleAlpha (
const double e) const [private]

### HoleAttachment()

bool Garfield::MediumSilicon::HoleAttachment (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & eta) [override]

### HoleMobility() [1/2]

double Garfield::MediumSilicon::HoleMobility () [inline], [override]
Definition at line 140 of file MediumSilicon.hh.
00140 { return m_hMu; }

### HoleMobility() [2/2]

double Garfield::MediumSilicon::HoleMobility (
const double e) const [private]

### HoleScatteringRates()

bool Garfield::MediumSilicon::HoleScatteringRates () [private]

### HoleTownsend()

bool Garfield::MediumSilicon::HoleTownsend (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & alpha) [override]

### HoleVelocity()

bool Garfield::MediumSilicon::HoleVelocity (
const double ex,
const double ey,
const double ez,
const double bx,
const double by,
const double bz,
double & vx,
double & vy,
double & vz) [override]

### ImpurityScatteringRates()

bool Garfield::MediumSilicon::ImpurityScatteringRates (
const double kbt,
Band & band) [private]

### Initialise()

bool Garfield::MediumSilicon::Initialise ()

### InitialiseDOS()

void Garfield::MediumSilicon::InitialiseDOS () [private]

### IntervalleyScatteringRates()

bool Garfield::MediumSilicon::IntervalleyScatteringRates (
const double rho,
const double kbt,
const double dtk,
const double eph,
Band & bndI,
Band & bndF,
const double zF,
const int collType) [private]

### IonisationRates()

bool Garfield::MediumSilicon::IonisationRates (
const std::vector< double > & p,
const std::vector< double > & eth,
const std::vector< double > & b,
Band & band) [private]

### IsSemiconductor()

bool Garfield::MediumSilicon::IsSemiconductor () const [inline], [override]
Definition at line 21 of file MediumSilicon.hh.
00021 { return true; }

### LoadOpticalData()

bool Garfield::MediumSilicon::LoadOpticalData (
const std::string & filename) [private]

### OpticalScatteringRates()

bool Garfield::MediumSilicon::OpticalScatteringRates (
const double rho,
const double kbt,
const double dtk,
const double eph,
Band & band) [private]

### ResetCollisionCounters()

void Garfield::MediumSilicon::ResetCollisionCounters ()

SetDiffusionScaling()

void Garfield::MediumSilicon::SetDiffusionScaling (
const double d) [inline]
Apply a scaling factor to the diffusion coefficients.
Definition at line 101 of file MediumSilicon.hh.
00101 { m_diffScale = d; }

SetDoping()

void Garfield::MediumSilicon::SetDoping (
const char type,
const double c)
Set doping concentration [cm-3] and type ('i', 'n', 'p').

SetDopingMobilityModelMasetti()

void Garfield::MediumSilicon::SetDopingMobilityModelMasetti ()
Use the Masetti model for the doping-dependence of the mobility (default).

SetDopingMobilityModelMinimos()

void Garfield::MediumSilicon::SetDopingMobilityModelMinimos ()
Use the Minimos model for the doping-dependence of the mobility.

SetHighFieldMobilityModel()

void Garfield::MediumSilicon::SetHighFieldMobilityModel (
const std::string & model)
Set the parameterisation to be used for the drift velocity as function of the electric field.
The currently implemented models are

- Canali
- Reggiani
- Minimos
- Constant (velocity increases linearly with the electric field) The default is Canali.

SetHighFieldMobilityModelCanali()

void Garfield::MediumSilicon::SetHighFieldMobilityModelCanali ()
Parameterize the high-field mobility using the Canali model (default).

SetHighFieldMobilityModelConstant()

void Garfield::MediumSilicon::SetHighFieldMobilityModelConstant ()
Make the velocity proportional to the electric field (no saturation).

SetHighFieldMobilityModelMinimos()

void Garfield::MediumSilicon::SetHighFieldMobilityModelMinimos ()
Parameterize the high-field mobility using the Minimos model.

SetHighFieldMobilityModelReggiani()

void Garfield::MediumSilicon::SetHighFieldMobilityModelReggiani ()
Parameterize the high-field mobility using the Reggiani model.

SetImpactIonisationModel()

void Garfield::MediumSilicon::SetImpactIonisationModel (
const std::string & model)
Set the parameterisation to be used for calculating the impact ionisation coefficient.
The currently implemented models are

- van Overstraeten - de Man
- Okuto - Crowell
- Massey
- Grant The default is van Overstraeten - de Man.

SetImpactIonisationModelGrant()

void Garfield::MediumSilicon::SetImpactIonisationModelGrant ()
Calculate $\alpha$ using the Grant model.

SetImpactIonisationModelMassey()

void Garfield::MediumSilicon::SetImpactIonisationModelMassey ()
Calculate $\alpha$ using the Massey model.

SetImpactIonisationModelOkutoCrowell()

void Garfield::MediumSilicon::SetImpactIonisationModelOkutoCrowell ()
Calculate $\alpha$ using the Okuto-Crowell model.

SetImpactIonisationModelVanOverstraetenDeMan()

void Garfield::MediumSilicon::SetImpactIonisationModelVanOverstraetenDeMan ()
Calculate $\alpha$ using the van Overstraeten-de Man model (default).

SetLatticeMobilityModel()

void Garfield::MediumSilicon::SetLatticeMobilityModel (
const std::string & model)
Set the parameterisation to be used for calculating the lattice mobility model.
The currently implemented models are

- Sentaurus
- Reggiani
- Minimos The default is Sentaurus.

SetLatticeMobilityModelMinimos()

void Garfield::MediumSilicon::SetLatticeMobilityModelMinimos ()
Calculate the lattice mobility using the Minimos model.

SetLatticeMobilityModelReggiani()

void Garfield::MediumSilicon::SetLatticeMobilityModelReggiani ()
Calculate the lattice mobility using the Reggiani model.

SetLatticeMobilityModelSentaurus()

void Garfield::MediumSilicon::SetLatticeMobilityModelSentaurus ()
Calculate the lattice mobility using the Sentaurus model (default).

SetLowFieldMobility()

void Garfield::MediumSilicon::SetLowFieldMobility (
const double mue,
const double muh)
Specify the low field values of the electron and hole mobilities.

SetMaxElectronEnergy()

bool Garfield::MediumSilicon::SetMaxElectronEnergy (
const double e)

SetSaturationVelocity()

void Garfield::MediumSilicon::SetSaturationVelocity (
const double vsate,
const double vsath)
Specify the saturation velocities of electrons and holes.

SetSaturationVelocityModelCanali()

void Garfield::MediumSilicon::SetSaturationVelocityModelCanali ()
Calculate the saturation velocities using the Canali model (default).

SetSaturationVelocityModelMinimos()

void Garfield::MediumSilicon::SetSaturationVelocityModelMinimos ()
Calculate the saturation velocities using the Minimos model.

SetSaturationVelocityModelReggiani()

void Garfield::MediumSilicon::SetSaturationVelocityModelReggiani ()
Calculate the saturation velocities using the Reggiani model.

SetTrapCrossSection()

void Garfield::MediumSilicon::SetTrapCrossSection (
const double ecs,
const double hcs)
Trapping cross-sections for electrons and holes.

SetTrapDensity()

void Garfield::MediumSilicon::SetTrapDensity (
const double n)
Trap density [cm-3], by default set to zero.

SetTrappingTime()

void Garfield::MediumSilicon::SetTrappingTime (
const double etau,
const double htau)
Set time constant for trapping of electrons and holes [ns].

Update()

bool Garfield::MediumSilicon::Update () [private]

UpdateDopingMobilityMasetti()

void Garfield::MediumSilicon::UpdateDopingMobilityMasetti () [private]

### UpdateDopingMobilityMinimos()

void Garfield::MediumSilicon::UpdateDopingMobilityMinimos () [private]

### UpdateHighFieldMobilityCanali()

void Garfield::MediumSilicon::UpdateHighFieldMobilityCanali () [private]

### UpdateImpactIonisation()

void Garfield::MediumSilicon::UpdateImpactIonisation () [private]

### UpdateLatticeMobility()

void Garfield::MediumSilicon::UpdateLatticeMobility () [private]

### UpdateSaturationVelocity()

void Garfield::MediumSilicon::UpdateSaturationVelocity () [private]

## 7.68.5 Member Data Documentation

### m_anisotropic

bool Garfield::MediumSilicon::m_anisotropic = true [private]
Definition at line 254 of file MediumSilicon.hh.

### m_bandGap

double Garfield::MediumSilicon::m_bandGap = 1.12 [private]
Definition at line 191 of file MediumSilicon.hh.

### m_cb

std::array<Band, 3> Garfield::MediumSilicon::m_cb [private]
Definition at line 292 of file MediumSilicon.hh.

### m_cbIndex

std::vector<size_t> Garfield::MediumSilicon::m_cbIndex [private]
Definition at line 293 of file MediumSilicon.hh.

### m_cDop

double Garfield::MediumSilicon::m_cDop = 0. [private]
Definition at line 195 of file MediumSilicon.hh.

### m_cfOutput

bool Garfield::MediumSilicon::m_cfOutput = false [private]
Definition at line 251 of file MediumSilicon.hh.

### m_diffScale

double Garfield::MediumSilicon::m_diffScale = 1. [private]
Definition at line 189 of file MediumSilicon.hh.

### m_dopingMobilityModel

DopingMobility Garfield::MediumSilicon::m_dopingMobilityModel = DopingMobility::Masetti [private]
Definition at line 245 of file MediumSilicon.hh.

### m_dopingType

char Garfield::MediumSilicon::m_dopingType = 'i' [private]
Definition at line 193 of file MediumSilicon.hh.

### m_eBetaCanali

double Garfield::MediumSilicon::m_eBetaCanali = 1.109 [private]
Definition at line 204 of file MediumSilicon.hh.

### m_eBetaCanaliInv

double Garfield::MediumSilicon::m_eBetaCanaliInv = 1. / 1.109 [private]
Definition at line 206 of file MediumSilicon.hh.

### m_egamma

std::vector<double> Garfield::MediumSilicon::m_egamma [private]
Definition at line 315 of file MediumSilicon.hh.

### m_eHallFactor

double Garfield::MediumSilicon::m_eHallFactor = 1.15 [private]
Definition at line 215 of file MediumSilicon.hh.

### m_eImpactA0

double Garfield::MediumSilicon::m_eImpactA0 = 3.318e5 [private]
Definition at line 230 of file MediumSilicon.hh.

### m_eImpactA1

double Garfield::MediumSilicon::m_eImpactA1 = 0.703e6 [private]
Definition at line 231 of file MediumSilicon.hh.

### m_eImpactA2

double Garfield::MediumSilicon::m_eImpactA2 = 0. [private]
Definition at line 232 of file MediumSilicon.hh.

### m_eImpactB0

double Garfield::MediumSilicon::m_eImpactB0 = 1.135e6 [private]
Definition at line 233 of file MediumSilicon.hh.

### m_eImpactB1

double Garfield::MediumSilicon::m_eImpactB1 = 1.231e6 [private]
Definition at line 234 of file MediumSilicon.hh.

### m_eImpactB2

double Garfield::MediumSilicon::m_eImpactB2 = 0. [private]
Definition at line 235 of file MediumSilicon.hh.

### m_eMu

double Garfield::MediumSilicon::m_eMu = 1.35e-6 [private]
Definition at line 201 of file MediumSilicon.hh.

### m_eMuLat

double Garfield::MediumSilicon::m_eMuLat = 1.35e-6 [private]
Definition at line 198 of file MediumSilicon.hh.

### m_eps1

std::vector<double> Garfield::MediumSilicon::m_eps1 [private]
Definition at line 316 of file MediumSilicon.hh.

### m_eps2

std::vector<double> Garfield::MediumSilicon::m_eps2 [private]
Definition at line 317 of file MediumSilicon.hh.

### m_eRs

double Garfield::MediumSilicon::m_eRs = 1.35e-6 / 1.02e-2 [private]
Definition at line 212 of file MediumSilicon.hh.

### m_eStepDos

double Garfield::MediumSilicon::m_eStepDos = 0. [private]
Definition at line 307 of file MediumSilicon.hh.

### m_eTrapCs

double Garfield::MediumSilicon::m_eTrapCs = 1.e-15 [private]
Definition at line 219 of file MediumSilicon.hh.

### m_eTrapDensity

double Garfield::MediumSilicon::m_eTrapDensity = 0. [private]
Definition at line 221 of file MediumSilicon.hh.

### m_eTrapRate

double Garfield::MediumSilicon::m_eTrapRate = 0. [private]
Definition at line 225 of file MediumSilicon.hh.

### m_eTrapTime

double Garfield::MediumSilicon::m_eTrapTime = 0. [private]
Definition at line 223 of file MediumSilicon.hh.

### m_eVs

double Garfield::MediumSilicon::m_eVs = 1.02e-2 [private]
Definition at line 209 of file MediumSilicon.hh.

### m_fbDosC

std::vector<double> Garfield::MediumSilicon::m_fbDosC [private]
Definition at line 310 of file MediumSilicon.hh.

### m_fbDosMaxC

double Garfield::MediumSilicon::m_fbDosMaxC [private]
Definition at line 311 of file MediumSilicon.hh.

### m_fbDosMaxV

double Garfield::MediumSilicon::m_fbDosMaxV [private]
Definition at line 311 of file MediumSilicon.hh.

### m_fbDosV

std::vector<double> Garfield::MediumSilicon::m_fbDosV [private]
Definition at line 309 of file MediumSilicon.hh.

### m_fullBandDos

bool Garfield::MediumSilicon::m_fullBandDos = false [private]
Definition at line 253 of file MediumSilicon.hh.

### m_hasUserMobility

bool Garfield::MediumSilicon::m_hasUserMobility = false [private]
Definition at line 242 of file MediumSilicon.hh.

### m_hasUserSaturationVelocity

bool Garfield::MediumSilicon::m_hasUserSaturationVelocity = false [private]
Definition at line 243 of file MediumSilicon.hh.

### m_hBetaCanali

double Garfield::MediumSilicon::m_hBetaCanali = 1.213 [private]
Definition at line 205 of file MediumSilicon.hh.

### m_hBetaCanaliInv

double Garfield::MediumSilicon::m_hBetaCanaliInv = 1. / 1.213 [private]
Definition at line 207 of file MediumSilicon.hh.

### m_hHallFactor

double Garfield::MediumSilicon::m_hHallFactor = 0.7 [private]
Definition at line 216 of file MediumSilicon.hh.

### m_highFieldMobilityModel

HighFieldMobility Garfield::MediumSilicon::m_highFieldMobilityModel = HighFieldMobility::Canali [private]
Definition at line 247 of file MediumSilicon.hh.

### m_hImpactA0

double Garfield::MediumSilicon::m_hImpactA0 = 1.582e6 [private]
Definition at line 236 of file MediumSilicon.hh.

### m_hImpactA1

double Garfield::MediumSilicon::m_hImpactA1 = 0.671e6 [private]
Definition at line 237 of file MediumSilicon.hh.

### m_hImpactB0

double Garfield::MediumSilicon::m_hImpactB0 = 2.036e6 [private]
Definition at line 238 of file MediumSilicon.hh.

### m_hImpactB1

double Garfield::MediumSilicon::m_hImpactB1 = 1.693e6 [private]
Definition at line 239 of file MediumSilicon.hh.

### m_hMu

double Garfield::MediumSilicon::m_hMu = 0.45e-6 [private]
Definition at line 202 of file MediumSilicon.hh.

### m_hMuLat

double Garfield::MediumSilicon::m_hMuLat = 0.45e-6 [private]
Definition at line 199 of file MediumSilicon.hh.

### m_hRs

double Garfield::MediumSilicon::m_hRs = 0.45e-6 / 0.72e-2 [private]
Definition at line 213 of file MediumSilicon.hh.

### m_hTrapCs

double Garfield::MediumSilicon::m_hTrapCs = 1.e-15 [private]
Definition at line 220 of file MediumSilicon.hh.

### m_hTrapDensity

double Garfield::MediumSilicon::m_hTrapDensity = 0. [private]
Definition at line 222 of file MediumSilicon.hh.

### m_hTrapRate

double Garfield::MediumSilicon::m_hTrapRate = 0. [private]
Definition at line 226 of file MediumSilicon.hh.

### m_hTrapTime

double Garfield::MediumSilicon::m_hTrapTime = 0. [private]
Definition at line 224 of file MediumSilicon.hh.

### m_hVs

double Garfield::MediumSilicon::m_hVs = 0.72e-2 [private]
Definition at line 210 of file MediumSilicon.hh.

### m_impactIonisationModel

ImpactIonisation Garfield::MediumSilicon::m_impactIonisationModel = ImpactIonisation::VanOverstraeten [private]
Definition at line 248 of file MediumSilicon.hh.

### m_invStepDos

double Garfield::MediumSilicon::m_invStepDos = 0. [private]
Definition at line 308 of file MediumSilicon.hh.

### m_latticeMobilityModel

LatticeMobility Garfield::MediumSilicon::m_latticeMobilityModel = LatticeMobility::Sentaurus [private]
Definition at line 244 of file MediumSilicon.hh.

### m_mutex

std::mutex Garfield::MediumSilicon::m_mutex [private]
Definition at line 186 of file MediumSilicon.hh.

### m_nCollElectronAcoustic

unsigned int Garfield::MediumSilicon::m_nCollElectronAcoustic = 0 [private]
Definition at line 298 of file MediumSilicon.hh.

### m_nCollElectronBand

std::vector<unsigned int> Garfield::MediumSilicon::m_nCollElectronBand [private]
Definition at line 304 of file MediumSilicon.hh.

### m_nCollElectronDetailed

std::vector<unsigned int> Garfield::MediumSilicon::m_nCollElectronDetailed [private]
Definition at line 303 of file MediumSilicon.hh.

### m_nCollElectronImpurity

unsigned int Garfield::MediumSilicon::m_nCollElectronImpurity = 0 [private]
Definition at line 301 of file MediumSilicon.hh.

### m_nCollElectronIntervalley

unsigned int Garfield::MediumSilicon::m_nCollElectronIntervalley = 0 [private]
Definition at line 300 of file MediumSilicon.hh.

### m_nCollElectronIonisation

unsigned int Garfield::MediumSilicon::m_nCollElectronIonisation = 0 [private]
Definition at line 302 of file MediumSilicon.hh.

### m_nCollElectronOptical

unsigned int Garfield::MediumSilicon::m_nCollElectronOptical = 0 [private]
Definition at line 299 of file MediumSilicon.hh.

### m_nonParabolic

bool Garfield::MediumSilicon::m_nonParabolic = true [private]
Definition at line 252 of file MediumSilicon.hh.

### m_opticalDataFile

std::string Garfield::MediumSilicon::m_opticalDataFile = "OpticalData_Si.txt" [private]
Definition at line 314 of file MediumSilicon.hh.

### m_saturationVelocityModel

SaturationVelocity Garfield::MediumSilicon::m_saturationVelocityModel = SaturationVelocity::Canali [private]
Definition at line 246 of file MediumSilicon.hh.

### m_trappingModel

int Garfield::MediumSilicon::m_trappingModel = 0 [private]
Definition at line 227 of file MediumSilicon.hh.

m_vb

Band Garfield::MediumSilicon::m_vb [private]
Definition at line 295 of file MediumSilicon.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/MediumSilicon.hh

## 7.69 Garfield::ComponentFieldMap::Node Struct Reference

#include <ComponentFieldMap.hh>

Public Attributes

- double x
- double y
- double z

### 7.69.1 Detailed Description

Definition at line 209 of file ComponentFieldMap.hh.

### 7.69.2 Member Data Documentation

x

double Garfield::ComponentFieldMap::Node::x
Definition at line 211 of file ComponentFieldMap.hh.

y

double Garfield::ComponentFieldMap::Node::y
Definition at line 211 of file ComponentFieldMap.hh.

z

double Garfield::ComponentFieldMap::Node::z
Definition at line 211 of file ComponentFieldMap.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh

## 7.70 Garfield::ComponentGrid::Node Struct Reference

Public Attributes

- double fx
- double fy
- double fz

  Field.
- double v

  Potential.

### 7.70.1 Detailed Description

Definition at line 201 of file ComponentGrid.hh.

### 7.70.2 Member Data Documentation

fx

double Garfield::ComponentGrid::Node::fx
Definition at line 202 of file ComponentGrid.hh.

fy

double Garfield::ComponentGrid::Node::fy
Definition at line 202 of file ComponentGrid.hh.

fz

double Garfield::ComponentGrid::Node::fz
Field.
Definition at line 202 of file ComponentGrid.hh.

v

double Garfield::ComponentGrid::Node::v
Potential.
Definition at line 203 of file ComponentGrid.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentGrid.hh

## 7.71 Garfield::OpticalData Class Reference

Photoabsorption cross-sections for some gases.
#include <OpticalData.hh>

Public Member Functions

- OpticalData ()=default
  Constructor.
- ~OpticalData ()=default
  Destructor.

Static Public Member Functions

- static bool IsAvailable (const std::string &material)
  Check whether optical data have been implemented for a given gas.
- static bool PhotoabsorptionCrossSection (const std::string &material, const double energy, double &cs, double &eta)
  Photo-absorption cross-section and ionisation yield at a given energy.
- static double PhotoabsorptionCrossSection (const std::string &material, const double energy)
  Photo-absorption cross-section at a given energy.
- static double PhotoionisationYield (const std::string &material, const double energy)
  Photo-ionisation yield at a given energy.

Static Private Member Functions

- static bool PhotoAbsorptionCsNeon (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsArgon (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsCO2 (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsMethane (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsEthane (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsButane (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsAcetylene (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsCF4 (const double e, double &cs, double &eta)
- static bool PhotoAbsorptionCsNitrogen (const double e, double &cs, double &eta)

Static Private Attributes

- static constexpr double OscToPacs = 8.067283e-18
- static constexpr double Mbarn = 1.e-18

### 7.71.1 Detailed Description

Photoabsorption cross-sections for some gases.
Definition at line 10 of file OpticalData.hh.

### 7.71.2 Constructor & Destructor Documentation

#### OpticalData()

Garfield::OpticalData::OpticalData () [default]
Constructor.

#### ~OpticalData()

Garfield::OpticalData::~OpticalData () [default]
Destructor.

### 7.71.3 Member Function Documentation

#### IsAvailable()

bool Garfield::OpticalData::IsAvailable (
const std::string & material) [static]
Check whether optical data have been implemented for a given gas.

#### PhotoabsorptionCrossSection() [1/2]

double Garfield::OpticalData::PhotoabsorptionCrossSection (
const std::string & material,
const double energy) [static]
Photo-absorption cross-section at a given energy.

#### PhotoabsorptionCrossSection() [2/2]

bool Garfield::OpticalData::PhotoabsorptionCrossSection (
const std::string & material,
const double energy,
double & cs,
double & eta) [static]
Photo-absorption cross-section and ionisation yield at a given energy.

#### PhotoAbsorptionCsAcetylene()

bool Garfield::OpticalData::PhotoAbsorptionCsAcetylene (
const double e,
double & cs,
double & eta) [static], [private]

#### PhotoAbsorptionCsArgon()

bool Garfield::OpticalData::PhotoAbsorptionCsArgon (
const double e,
double & cs,
double & eta) [static], [private]

#### PhotoAbsorptionCsButane()

bool Garfield::OpticalData::PhotoAbsorptionCsButane (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoAbsorptionCsCF4()

bool Garfield::OpticalData::PhotoAbsorptionCsCF4 (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoAbsorptionCsCO2()

bool Garfield::OpticalData::PhotoAbsorptionCsCO2 (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoAbsorptionCsEthane()

bool Garfield::OpticalData::PhotoAbsorptionCsEthane (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoAbsorptionCsMethane()

bool Garfield::OpticalData::PhotoAbsorptionCsMethane (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoAbsorptionCsNeon()

bool Garfield::OpticalData::PhotoAbsorptionCsNeon (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoAbsorptionCsNitrogen()

bool Garfield::OpticalData::PhotoAbsorptionCsNitrogen (
const double e,
double & cs,
double & eta) [static], [private]

### PhotoionisationYield()

double Garfield::OpticalData::PhotoionisationYield (
const std::string & material,
const double energy) [static]

Photo-ionisation yield at a given energy.

## 7.71.4 Member Data Documentation

### Mbarn

double Garfield::OpticalData::Mbarn = 1.e-18 [static], [constexpr], [private]

Definition at line 32 of file OpticalData.hh.

### OscToPacs

double Garfield::OpticalData::OscToPacs = 8.067283e-18 [static], [constexpr], [private]

Definition at line 31 of file OpticalData.hh.

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/OpticalData.hh

## 7.72 Garfield::Panel Struct Reference

Surface panel.
#include <Solid.hh>

Public Attributes

- double a

  Perpendicular vector.
- double b
- double c
- std::vector< double > xv

  X-coordinates of vertices.
- std::vector< double > yv

  Y-coordinates of vertices.
- std::vector< double > zv

  Z-coordinates of vertices.
- int colour

  Colour index.
- int volume

  Reference to solid to which the panel belongs.

### 7.72.1 Detailed Description

Surface panel.
Definition at line 11 of file Solid.hh.

### 7.72.2 Member Data Documentation

a

double Garfield::Panel::a
Perpendicular vector.
Definition at line 13 of file Solid.hh.

b

double Garfield::Panel::b
Definition at line 13 of file Solid.hh.

c

double Garfield::Panel::c
Definition at line 13 of file Solid.hh.

colour

int Garfield::Panel::colour
Colour index.
Definition at line 21 of file Solid.hh.

volume

int Garfield::Panel::volume
Reference to solid to which the panel belongs.
Definition at line 23 of file Solid.hh.

xv

std::vector<double> Garfield::Panel::xv
X-coordinates of vertices.
Definition at line 15 of file Solid.hh.

yv

std::vector<double> Garfield::Panel::yv
Y-coordinates of vertices.
Definition at line 17 of file Solid.hh.

zv

std::vector<double> Garfield::Panel::zv
Z-coordinates of vertices.
Definition at line 19 of file Solid.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Solid.hh

## 7.73 Garfield::TrackElectron::Parameters Struct Reference

Public Attributes

- double m2
- double cIon
- double x0
- double x1
- double cDens
- double aDens
- double mDens
- double wSplit
- double ethr

### 7.73.1 Detailed Description

Definition at line 38 of file TrackElectron.hh.

### 7.73.2 Member Data Documentation

aDens

double Garfield::TrackElectron::Parameters::aDens
Definition at line 47 of file TrackElectron.hh.

cDens

double Garfield::TrackElectron::Parameters::cDens
Definition at line 46 of file TrackElectron.hh.

cIon

double Garfield::TrackElectron::Parameters::cIon
Definition at line 42 of file TrackElectron.hh.

ethr

double Garfield::TrackElectron::Parameters::ethr
Definition at line 52 of file TrackElectron.hh.

m2

double Garfield::TrackElectron::Parameters::m2
Definition at line 40 of file TrackElectron.hh.

mDens

double Garfield::TrackElectron::Parameters::mDens
Definition at line 48 of file TrackElectron.hh.

wSplit

double Garfield::TrackElectron::Parameters::wSplit
Definition at line 50 of file TrackElectron.hh.

x0

double Garfield::TrackElectron::Parameters::x0
Definition at line 44 of file TrackElectron.hh.

x1

double Garfield::TrackElectron::Parameters::x1
Definition at line 45 of file TrackElectron.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackElectron.hh

## 7.74 Garfield::AvalancheGrid::Path Struct Reference

Public Attributes

- std::vector< double > ts = {}
- std::vector< std::array< double, 3 > > xs = {}
- std::vector< double > qs = {}

### 7.74.1 Detailed Description

Definition at line 124 of file AvalancheGrid.hh.

### 7.74.2 Member Data Documentation

qs

std::vector<double> Garfield::AvalancheGrid::Path::qs = {}
Definition at line 127 of file AvalancheGrid.hh.
00127 {};

ts

std::vector<double> Garfield::AvalancheGrid::Path::ts = {}
Definition at line 125 of file AvalancheGrid.hh.
00125 {};

xs

std::vector<std::array<double, 3> > Garfield::AvalancheGrid::Path::xs = {}
Definition at line 126 of file AvalancheGrid.hh.
00126 {};
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheGrid.hh

## 7.75 Garfield::AvalancheMicroscopic::Photon Struct Reference

Public Attributes

- int status

  Status.
- double energy

  Energy.
- double x0
- double y0
- double z0
- double t0

  Starting point and time.
- double x1
- double y1
- double z1
- double t1

  End point and time.

### 7.75.1 Detailed Description

Definition at line 332 of file AvalancheMicroscopic.hh.

### 7.75.2 Member Data Documentation

energy

double Garfield::AvalancheMicroscopic::Photon::energy
Energy.
Definition at line 334 of file AvalancheMicroscopic.hh.

status

int Garfield::AvalancheMicroscopic::Photon::status
Status.
Definition at line 333 of file AvalancheMicroscopic.hh.

t0

double Garfield::AvalancheMicroscopic::Photon::t0
Starting point and time.
Definition at line 335 of file AvalancheMicroscopic.hh.

t1

double Garfield::AvalancheMicroscopic::Photon::t1
End point and time.
Definition at line 336 of file AvalancheMicroscopic.hh.

x0

double Garfield::AvalancheMicroscopic::Photon::x0
Definition at line 335 of file AvalancheMicroscopic.hh.

x1

double Garfield::AvalancheMicroscopic::Photon::x1
Definition at line 336 of file AvalancheMicroscopic.hh.

y0

double Garfield::AvalancheMicroscopic::Photon::y0
Definition at line 335 of file AvalancheMicroscopic.hh.

y1

double Garfield::AvalancheMicroscopic::Photon::y1
Definition at line 336 of file AvalancheMicroscopic.hh.

z0

double Garfield::AvalancheMicroscopic::Photon::z0
Definition at line 335 of file AvalancheMicroscopic.hh.

z1

double Garfield::AvalancheMicroscopic::Photon::z1
Definition at line 336 of file AvalancheMicroscopic.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh

## 7.76 Garfield::TrackHeed::Photon Struct Reference

#include <TrackHeed.hh>

Public Attributes

- double x = 0.
- double y = 0.
- double z = 0.
- double t = 0.
- double e = 0.
- double dx = 0.
- double dy = 0.
- double dz = 0.

### 7.76.1 Detailed Description

Definition at line 47 of file TrackHeed.hh.

### 7.76.2 Member Data Documentation

dx

double Garfield::TrackHeed::Photon::dx = 0.
Definition at line 53 of file TrackHeed.hh.

dy

double Garfield::TrackHeed::Photon::dy = 0.
Definition at line 54 of file TrackHeed.hh.

dz

double Garfield::TrackHeed::Photon::dz = 0.
Definition at line 55 of file TrackHeed.hh.

e

double Garfield::TrackHeed::Photon::e = 0.
Definition at line 52 of file TrackHeed.hh.

t

double Garfield::TrackHeed::Photon::t = 0.
Definition at line 51 of file TrackHeed.hh.

x

double Garfield::TrackHeed::Photon::x = 0.
Definition at line 48 of file TrackHeed.hh.

y

double Garfield::TrackHeed::Photon::y = 0.
Definition at line 49 of file TrackHeed.hh.

z

double Garfield::TrackHeed::Photon::z = 0.
Definition at line 50 of file TrackHeed.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackHeed.hh

## 7.77 Garfield::ComponentAnalyticField::Pixel Struct Reference

Public Attributes

- std::string type
  Label.
- int ind = 0
  Readout group.
- double smin = 0.
- double smax = 0.
  Coordinates in x/y.
- double zmin = 0.
- double zmax = 0.
  Coordinates in z.
- double gap = -1.
  Distance to the opposite electrode.
- double cphi = 1.
  Rotation.
- double sphi = 0.
  Rotation.

### 7.77.1 Detailed Description

Definition at line 481 of file ComponentAnalyticField.hh.

### 7.77.2 Member Data Documentation

cphi

double Garfield::ComponentAnalyticField::Pixel::cphi = 1.
Rotation.
Definition at line 487 of file ComponentAnalyticField.hh.

gap

double Garfield::ComponentAnalyticField::Pixel::gap = -1.
Distance to the opposite electrode.
Definition at line 486 of file ComponentAnalyticField.hh.

ind

int Garfield::ComponentAnalyticField::Pixel::ind = 0
Readout group.
Definition at line 483 of file ComponentAnalyticField.hh.

smax

double Garfield::ComponentAnalyticField::Pixel::smax = 0.
Coordinates in x/y.
Definition at line 484 of file ComponentAnalyticField.hh.

smin

double Garfield::ComponentAnalyticField::Pixel::smin = 0.
Definition at line 484 of file ComponentAnalyticField.hh.

sphi

double Garfield::ComponentAnalyticField::Pixel::sphi = 0.
Rotation.
Definition at line 488 of file ComponentAnalyticField.hh.

type

std::string Garfield::ComponentAnalyticField::Pixel::type
Label.
Definition at line 482 of file ComponentAnalyticField.hh.

zmax

double Garfield::ComponentAnalyticField::Pixel::zmax = 0.
Coordinates in z.
Definition at line 485 of file ComponentAnalyticField.hh.

zmin

double Garfield::ComponentAnalyticField::Pixel::zmin = 0.
Definition at line 485 of file ComponentAnalyticField.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh

## 7.78 Garfield::ComponentAnalyticField::Plane Struct Reference

Public Attributes

- std::string type
  Label.
- int ind
  Readout group.
- double ewxcor
- double ewycor
  Background weighting fields.
- std::vector< Strip > strips1
  x/y strips.
- std::vector< Strip > strips2
  z strips.
- std::vector< Pixel > pixels
  Pixels.

### 7.78.1 Detailed Description

Definition at line 491 of file ComponentAnalyticField.hh.

### 7.78.2 Member Data Documentation

ewxcor

double Garfield::ComponentAnalyticField::Plane::ewxcor
Definition at line 494 of file ComponentAnalyticField.hh.

ewycor

double Garfield::ComponentAnalyticField::Plane::ewycor
Background weighting fields.
Definition at line 494 of file ComponentAnalyticField.hh.

ind

int Garfield::ComponentAnalyticField::Plane::ind
Readout group.
Definition at line 493 of file ComponentAnalyticField.hh.

pixels

std::vector<Pixel> Garfield::ComponentAnalyticField::Plane::pixels
Pixels.
Definition at line 497 of file ComponentAnalyticField.hh.

strips1

std::vector<Strip> Garfield::ComponentAnalyticField::Plane::strips1
x/y strips.
Definition at line 495 of file ComponentAnalyticField.hh.

strips2

std::vector<Strip> Garfield::ComponentAnalyticField::Plane::strips2
z strips.
Definition at line 496 of file ComponentAnalyticField.hh.

type

std::string Garfield::ComponentAnalyticField::Plane::type
Label.
Definition at line 492 of file ComponentAnalyticField.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh

## 7.79 Garfield::PlottingEngine Class Reference

Plotting style.
#include <PlottingEngine.hh>

Public Member Functions

- PlottingEngine ()
  Default constructor.
- ~PlottingEngine ()=default
  Destructor.
- void SetSerif ()

Use serif font.

- void SetSansSerif ()

  Use sans-serif font.

- void SetPalette (int ncol)

  Set the colour palette.

- void SetDefaultStyle ()

  Apply the default Garfield ROOT style.

Private Attributes

- std::string m_className = "PlottingEngine"
- bool m_serif = false
- int m_palette = 0
- TStyle m_garfieldStyle

### 7.79.1 Detailed Description

Plotting style.
Definition at line 10 of file PlottingEngine.hh.

### 7.79.2 Constructor & Destructor Documentation

PlottingEngine()

Garfield::PlottingEngine::PlottingEngine ()
Default constructor.

~PlottingEngine()

Garfield::PlottingEngine::~PlottingEngine () [default]
Destructor.

### 7.79.3 Member Function Documentation

SetDefaultStyle()

void Garfield::PlottingEngine::SetDefaultStyle ()
Apply the default Garfield ROOT style.

SetPalette()

void Garfield::PlottingEngine::SetPalette (
int ncol) [inline]
Set the colour palette.
Definition at line 23 of file PlottingEngine.hh.
00023 { m_palette = ncol; }

SetSansSerif()

void Garfield::PlottingEngine::SetSansSerif () [inline]
Use sans-serif font.
Definition at line 20 of file PlottingEngine.hh.
00020 { m_serif = false; }

SetSerif()

void Garfield::PlottingEngine::SetSerif () [inline]
Use serif font.
Definition at line 18 of file PlottingEngine.hh.
00018 { m_serif = true; }

### 7.79.4 Member Data Documentation

m_className

std::string Garfield::PlottingEngine::m_className = "PlottingEngine" [private]
Definition at line 29 of file PlottingEngine.hh.

m_garfieldStyle

TStyle Garfield::PlottingEngine::m_garfieldStyle [private]
Definition at line 34 of file PlottingEngine.hh.

m_palette

int Garfield::PlottingEngine::m_palette = 0 [private]
Definition at line 32 of file PlottingEngine.hh.

m_serif

bool Garfield::PlottingEngine::m_serif = false [private]
Definition at line 31 of file PlottingEngine.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/PlottingEngine.hh

## 7.80 Garfield::AvalancheGridSpaceCharge::Point Struct Reference

Public Attributes

- double x
- double y
- double z
  coordinates
- double t
  time
- int gasLayerIndex

### 7.80.1 Detailed Description

Definition at line 173 of file AvalancheGridSpaceCharge.hh.

### 7.80.2 Member Data Documentation

gasLayerIndex

int Garfield::AvalancheGridSpaceCharge::Point::gasLayerIndex
Definition at line 177 of file AvalancheGridSpaceCharge.hh.

t

double Garfield::AvalancheGridSpaceCharge::Point::t
time
Definition at line 175 of file AvalancheGridSpaceCharge.hh.

x

double Garfield::AvalancheGridSpaceCharge::Point::x
Definition at line 174 of file AvalancheGridSpaceCharge.hh.

y

double Garfield::AvalancheGridSpaceCharge::Point::y
Definition at line 174 of file AvalancheGridSpaceCharge.hh.

z

double Garfield::AvalancheGridSpaceCharge::Point::z
coordinates
Definition at line 174 of file AvalancheGridSpaceCharge.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheGridSpaceCharge.hh

## 7.81 Garfield::AvalancheMC::Point Struct Reference

#include <AvalancheMC.hh>

Public Attributes

- double x
- double y
- double z
- double t

### 7.81.1 Detailed Description

Definition at line 73 of file AvalancheMC.hh.

### 7.81.2 Member Data Documentation

t

double Garfield::AvalancheMC::Point::t
Definition at line 75 of file AvalancheMC.hh.

x

double Garfield::AvalancheMC::Point::x
Definition at line 74 of file AvalancheMC.hh.

y

double Garfield::AvalancheMC::Point::y
Definition at line 74 of file AvalancheMC.hh.

z

double Garfield::AvalancheMC::Point::z
Definition at line 74 of file AvalancheMC.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh

## 7.82 Garfield::AvalancheMicroscopic::Point Struct Reference

#include <AvalancheMicroscopic.hh>

Public Attributes

- double x
- double y
- double z

  Coordinates.
- double t

  Time.

- double energy

  Kinetic energy.
- double kx
- double ky
- double kz

  Direction/wave vector.
- int band

  Band.

### 7.82.1 Detailed Description

Definition at line 173 of file AvalancheMicroscopic.hh.

### 7.82.2 Member Data Documentation

band

int Garfield::AvalancheMicroscopic::Point::band
Band.
Definition at line 178 of file AvalancheMicroscopic.hh.

energy

double Garfield::AvalancheMicroscopic::Point::energy
Kinetic energy.
Definition at line 176 of file AvalancheMicroscopic.hh.

kx

double Garfield::AvalancheMicroscopic::Point::kx
Definition at line 177 of file AvalancheMicroscopic.hh.

ky

double Garfield::AvalancheMicroscopic::Point::ky
Definition at line 177 of file AvalancheMicroscopic.hh.

kz

double Garfield::AvalancheMicroscopic::Point::kz
Direction/wave vector.
Definition at line 177 of file AvalancheMicroscopic.hh.

t

double Garfield::AvalancheMicroscopic::Point::t
Time.
Definition at line 175 of file AvalancheMicroscopic.hh.

x

double Garfield::AvalancheMicroscopic::Point::x
Definition at line 174 of file AvalancheMicroscopic.hh.

y

double Garfield::AvalancheMicroscopic::Point::y
Definition at line 174 of file AvalancheMicroscopic.hh.

z

double Garfield::AvalancheMicroscopic::Point::z
Coordinates.
Definition at line 174 of file AvalancheMicroscopic.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh

## 7.83 Garfield::ComponentNeBem3d::Primitive Struct Reference

Public Attributes

- double a
  Perpendicular vector.
- double b
- double c
- std::vector< double > xv
  X-coordinates of vertices.
- std::vector< double > yv
  Y-coordinates of vertices.
- std::vector< double > zv
  Z-coordinates of vertices.
- int interface = 0
  Interface type.
- double v = 0.
  Potential.
- double q = 0.
  Charge.
- double lambda = 0.
  Ratio of dielectric constants.
- double elementSize
  Target element size.
- int vol1
  Volumes.
- int vol2

### 7.83.1 Detailed Description

Definition at line 219 of file ComponentNeBem3d.hh.

### 7.83.2 Member Data Documentation

a

double Garfield::ComponentNeBem3d::Primitive::a
Perpendicular vector.
Definition at line 221 of file ComponentNeBem3d.hh.

b

double Garfield::ComponentNeBem3d::Primitive::b
Definition at line 221 of file ComponentNeBem3d.hh.

c

double Garfield::ComponentNeBem3d::Primitive::c
Definition at line 221 of file ComponentNeBem3d.hh.

elementSize

double Garfield::ComponentNeBem3d::Primitive::elementSize
Target element size.
Definition at line 237 of file ComponentNeBem3d.hh.

interface

int Garfield::ComponentNeBem3d::Primitive::interface = 0
Interface type.
Definition at line 229 of file ComponentNeBem3d.hh.

lambda

double Garfield::ComponentNeBem3d::Primitive::lambda = 0.
Ratio of dielectric constants.
Definition at line 235 of file ComponentNeBem3d.hh.

q

double Garfield::ComponentNeBem3d::Primitive::q = 0.
Charge.
Definition at line 233 of file ComponentNeBem3d.hh.

v

double Garfield::ComponentNeBem3d::Primitive::v = 0.
Potential.
Definition at line 231 of file ComponentNeBem3d.hh.

vol1

int Garfield::ComponentNeBem3d::Primitive::vol1
Volumes.
Definition at line 239 of file ComponentNeBem3d.hh.

vol2

int Garfield::ComponentNeBem3d::Primitive::vol2
Definition at line 239 of file ComponentNeBem3d.hh.

xv

std::vector<double> Garfield::ComponentNeBem3d::Primitive::xv
X-coordinates of vertices.
Definition at line 223 of file ComponentNeBem3d.hh.

yv

std::vector<double> Garfield::ComponentNeBem3d::Primitive::yv
Y-coordinates of vertices.
Definition at line 225 of file ComponentNeBem3d.hh.

zv

std::vector<double> Garfield::ComponentNeBem3d::Primitive::zv
Z-coordinates of vertices.
Definition at line 227 of file ComponentNeBem3d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3d.hh

## 7.84 Garfield::QuadTree Class Reference

Quadtree search.
#include <QuadTree.hh>

Public Member Functions

- QuadTree (const double x0, const double y0, const double hx, const double hy)
  Constructor.
- ~QuadTree ()
  Destructor.
- void InsertMeshNode (const double x, const double y, const int index)
  Insert a mesh node (a vertex/point) to the tree.
- void InsertMeshElement (const double bb[4], const int index)
  Insert a mesh element with given bounding box and index to the tree.
- const std::vector< int > & GetElementsInBlock (const double x, const double y) const
  Get all elements linked to a block corresponding to the given point.

Private Member Functions

- bool DoesBoxOverlap (const double bb[4]) const
- int GetQuadrant (const double x, const double y) const
- bool IsLeafNode () const
- const QuadTree * GetBlockFromPoint (const double x, const double y) const
- const QuadTree * GetBlockFromPointHelper (const double x, const double y) const

Private Attributes

- double m_x0
- double m_y0
- double m_hx
- double m_hy
- double m_xmin
- double m_ymin
- double m_xmax
- double m_ymax
- QuadTree * children [4]
- std::vector< std::tuple< float, float, int > > nodes
- std::vector< int > elements

Static Private Attributes

- static std::vector< int > emptyBlock
- static const size_t BlockCapacity = 10

### 7.84.1 Detailed Description

Quadtree search.
Definition at line 12 of file QuadTree.hh.

### 7.84.2 Constructor & Destructor Documentation

QuadTree()

```
Garfield::QuadTree::QuadTree (
            const double x0,
            const double y0,
            const double hx,
            const double hy)
```

Constructor.

### ~QuadTree()

Garfield::QuadTree::~QuadTree ()
Destructor.

## 7.84.3 Member Function Documentation

### DoesBoxOverlap()

bool Garfield::QuadTree::DoesBoxOverlap (
const double bb[4]) const [private]

### GetBlockFromPoint()

const QuadTree * Garfield::QuadTree::GetBlockFromPoint (
const double x,
const double y) const [private]

### GetBlockFromPointHelper()

const QuadTree * Garfield::QuadTree::GetBlockFromPointHelper (
const double x,
const double y) const [private]

### GetElementsInBlock()

const std::vector< int > & Garfield::QuadTree::GetElementsInBlock (
const double x,
const double y) const
Get all elements linked to a block corresponding to the given point.

### GetQuadrant()

int Garfield::QuadTree::GetQuadrant (
const double x,
const double y) const [private]

### InsertMeshElement()

void Garfield::QuadTree::InsertMeshElement (
const double bb[4],
const int index)
Insert a mesh element with given bounding box and index to the tree.

### InsertMeshNode()

void Garfield::QuadTree::InsertMeshNode (
const double x,
const double y,
const int index)
Insert a mesh node (a vertex/point) to the tree.

### IsLeafNode()

bool Garfield::QuadTree::IsLeafNode () const [private]

## 7.84.4 Member Data Documentation

### BlockCapacity

const size_t Garfield::QuadTree::BlockCapacity = 10 [static], [private]
Definition at line 52 of file QuadTree.hh.

children

QuadTree* Garfield::QuadTree::children[4] [private]
Definition at line 41 of file QuadTree.hh.

elements

std::vector<int> Garfield::QuadTree::elements [private]
Definition at line 50 of file QuadTree.hh.

emptyBlock

std::vector<int> Garfield::QuadTree::emptyBlock [static], [private]
Definition at line 31 of file QuadTree.hh.

m_hx

double Garfield::QuadTree::m_hx [private]
Definition at line 36 of file QuadTree.hh.

m_hy

double Garfield::QuadTree::m_hy [private]
Definition at line 36 of file QuadTree.hh.

m_x0

double Garfield::QuadTree::m_x0 [private]
Definition at line 34 of file QuadTree.hh.

m_xmax

double Garfield::QuadTree::m_xmax [private]
Definition at line 38 of file QuadTree.hh.

m_xmin

double Garfield::QuadTree::m_xmin [private]
Definition at line 38 of file QuadTree.hh.

m_y0

double Garfield::QuadTree::m_y0 [private]
Definition at line 34 of file QuadTree.hh.

m_ymax

double Garfield::QuadTree::m_ymax [private]
Definition at line 38 of file QuadTree.hh.

m_ymin

double Garfield::QuadTree::m_ymin [private]
Definition at line 38 of file QuadTree.hh.

nodes

std::vector<std::tuple<float, float, int> > Garfield::QuadTree::nodes [private]
Definition at line 49 of file QuadTree.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/QuadTree.hh

## 7.85 Garfield::Random Class Reference

#include <Random.hh>

Public Member Functions

- Random ()=default

Static Public Member Functions

- template<typename T>
  static void SetEngine (T engine)
- static double Draw () noexcept

Static Private Attributes

- static std::function< double(void)> draw

### 7.85.1 Detailed Description

Definition at line 11 of file Random.hh.

### 7.85.2 Constructor & Destructor Documentation

Random()

Garfield::Random::Random () [default]

### 7.85.3 Member Function Documentation

Draw()

double Garfield::Random::Draw () [inline], [static], [noexcept]
Definition at line 18 of file Random.hh.

```
00018 { return draw(); };
```

SetEngine()

template<typename T>
void Garfield::Random::SetEngine (
T engine) [inline], [static]
Definition at line 15 of file Random.hh.

```
00015                                 {
00016     draw = std::bind(&T::Draw, engine);
00017   }
```

### 7.85.4 Member Data Documentation

draw

std::function<double(void)> Garfield::Random::draw [static], [private]
Definition at line 21 of file Random.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Random.hh

## 7.86 Garfield::RandomEngine< Engine, SeedType > Class Template Reference

Abstract base class for random number generators.
#include <RandomEngine.hh>

Public Types

- using seed_t = SeedType
- using engine_t = Engine

Public Member Functions

- RandomEngine ()=default
- RandomEngine (const SeedType &seed)
- void SetSeed (const seed_t &seed)

  Initialise the random number generator.
- double operator() ()

  Draw a random number.
- double Draw ()

  Draw a random number.
- seed_t GetSeed ()

  Retrieve the seed that was used.
- void Print ()

  Print some information about the random number generator.

Protected Attributes

- seed_t m_seed {0}

### 7.86.1 Detailed Description

template<typename Engine, typename SeedType>
class Garfield::RandomEngine< Engine, SeedType >

Abstract base class for random number generators.
Definition at line 8 of file RandomEngine.hh.

### 7.86.2 Member Typedef Documentation

engine_t

template<typename Engine, typename SeedType>
using Garfield::RandomEngine< Engine, SeedType >::engine_t = Engine
Definition at line 11 of file RandomEngine.hh.

seed_t

template<typename Engine, typename SeedType>
using Garfield::RandomEngine< Engine, SeedType >::seed_t = SeedType
Definition at line 10 of file RandomEngine.hh.

### 7.86.3 Constructor & Destructor Documentation

RandomEngine() [1/2]

template<typename Engine, typename SeedType>
Garfield::RandomEngine< Engine, SeedType >::RandomEngine () [default]

RandomEngine() [2/2]

template<typename Engine, typename SeedType>
Garfield::RandomEngine< Engine, SeedType >::RandomEngine (
const SeedType & seed) [inline]
Definition at line 13 of file RandomEngine.hh.
00013 { this->SetSeed(seed); }

### 7.86.4 Member Function Documentation

#### Draw()

template<typename Engine, typename SeedType>
double Garfield::RandomEngine< Engine, SeedType >::Draw () [inline]
Draw a random number.
Definition at line 22 of file RandomEngine.hh.

```
00022 { return static_cast<Engine*>(this)->Draw(); }
```

#### GetSeed()

template<typename Engine, typename SeedType>
seed_t Garfield::RandomEngine< Engine, SeedType >::GetSeed () [inline]
Retrieve the seed that was used.
Definition at line 24 of file RandomEngine.hh.

```
00024 { return static_cast<Engine*>(this)->GetSeed(); }
```

#### operator()()

template<typename Engine, typename SeedType>
double Garfield::RandomEngine< Engine, SeedType >::operator() () [inline]
Draw a random number.
Definition at line 20 of file RandomEngine.hh.

```
00020 { return Draw(); }
```

#### Print()

template<typename Engine, typename SeedType>
void Garfield::RandomEngine< Engine, SeedType >::Print () [inline]
Print some information about the random number generator.
Definition at line 26 of file RandomEngine.hh.

```
00026 { return static_cast<Engine*>(this)->Print(); }
```

#### SetSeed()

template<typename Engine, typename SeedType>
void Garfield::RandomEngine< Engine, SeedType >::SetSeed (
const seed_t & seed) [inline]
Initialise the random number generator.
Definition at line 15 of file RandomEngine.hh.

```
00015                                       {
00016     m_seed = seed;
00017     static_cast<Engine*>(this)->SetSeed(m_seed);
00018   }
```

### 7.86.5 Member Data Documentation

#### m_seed

template<typename Engine, typename SeedType>
seed_t Garfield::RandomEngine< Engine, SeedType >::m_seed {0} [protected]
Definition at line 29 of file RandomEngine.hh.

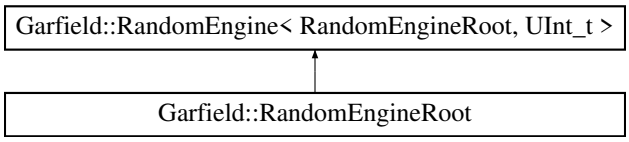
```
00029 {0};
```

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/RandomEngine.hh

## 7.87 Garfield::RandomEngineRoot Class Reference

#include <RandomEngineRoot.hh>
Inheritance diagram for Garfield::RandomEngineRoot:




Public Member Functions

- RandomEngineRoot ()
- RandomEngineRoot (const UInt_t &seed)
- double Draw ()
- void SetSeed (const seed_t &seed)
- seed_t GetSeed ()
- void Print ()

Public Member Functions inherited from Garfield::RandomEngine< RandomEngineRoot, UInt_t >

- RandomEngine ()=default
- void SetSeed (const seed_t &seed)

  Initialise the random number generator.
- double operator() ()

  Draw a random number.
- double Draw ()

  Draw a random number.
- seed_t GetSeed ()

  Retrieve the seed that was used.
- void Print ()

  Print some information about the random number generator.

Private Attributes

- TRandom3 m_rng

Additional Inherited Members

Public Types inherited from Garfield::RandomEngine< RandomEngineRoot, UInt_t >

- using seed_t
- using engine_t

Protected Attributes inherited from Garfield::RandomEngine< RandomEngineRoot, UInt_t >

- seed_t m_seed

### 7.87.1 Detailed Description

Definition at line 10 of file RandomEngineRoot.hh.

### 7.87.2 Constructor & Destructor Documentation

RandomEngineRoot() [1/2]

Garfield::RandomEngineRoot::RandomEngineRoot () [inline]
Definition at line 12 of file RandomEngineRoot.hh.

```
00012 { m_rng.SetSeed(); };
```

RandomEngineRoot() [2/2]

Garfield::RandomEngineRoot::RandomEngineRoot (
const UInt_t & seed) [inline]
Definition at line 13 of file RandomEngineRoot.hh.
00013 : RandomEngine(seed) {}

### 7.87.3 Member Function Documentation

Draw()

double Garfield::RandomEngineRoot::Draw () [inline]
Definition at line 14 of file RandomEngineRoot.hh.
00014 { return m_rng.Rndm(); }

GetSeed()

seed_t Garfield::RandomEngineRoot::GetSeed () [inline]
Definition at line 16 of file RandomEngineRoot.hh.
00016 { return m_seed; }

Print()

void Garfield::RandomEngineRoot::Print ()

SetSeed()

void Garfield::RandomEngineRoot::SetSeed (
const seed_t & seed) [inline]
Definition at line 15 of file RandomEngineRoot.hh.
00015 { m_rng.SetSeed(seed); }

### 7.87.4 Member Data Documentation

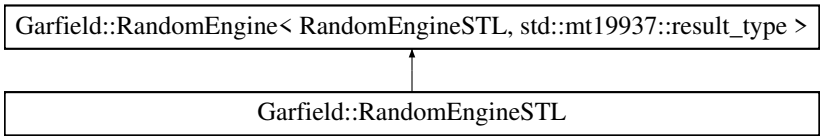
m_rng

TRandom3 Garfield::RandomEngineRoot::m_rng [private]
Definition at line 20 of file RandomEngineRoot.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/RandomEngineRoot.hh

## 7.88 Garfield::RandomEngineSTL Class Reference

#include <RandomEngineSTL.hh>
Inheritance diagram for Garfield::RandomEngineSTL:




Public Member Functions

- RandomEngineSTL ()=default
- RandomEngineSTL (const std::mt19937::result_type &seed)
- double Draw ()
- void SetSeed (const seed_t &seed)
- seed_t GetSeed ()
- void Print ()

Public Member Functions inherited from
Garfield::RandomEngine< RandomEngineSTL, std::mt19937::result_type >

- RandomEngine ()=default
- void SetSeed (const seed_t &seed)

  Initialise the random number generator.
- double operator() ()

  Draw a random number.
- double Draw ()

  Draw a random number.
- seed_t GetSeed ()

  Retrieve the seed that was used.
- void Print ()

  Print some information about the random number generator.

Private Attributes

- std::mt19937 m_rng {std::random_device{}()}

Additional Inherited Members

Public Types inherited from
Garfield::RandomEngine< RandomEngineSTL, std::mt19937::result_type >

- using seed_t
- using engine_t

Protected Attributes inherited from
Garfield::RandomEngine< RandomEngineSTL, std::mt19937::result_type >

- seed_t m_seed

### 7.88.1 Detailed Description

Definition at line 10 of file RandomEngineSTL.hh.

### 7.88.2 Constructor & Destructor Documentation

#### RandomEngineSTL() [1/2]

Garfield::RandomEngineSTL::RandomEngineSTL () [default]

#### RandomEngineSTL() [2/2]

Garfield::RandomEngineSTL::RandomEngineSTL (
const std::mt19937::result_type & seed) [inline]

Definition at line 14 of file RandomEngineSTL.hh.

```
00014 : RandomEngine(seed) {}
```

### 7.88.3 Member Function Documentation

#### Draw()

double Garfield::RandomEngineSTL::Draw () [inline]

Definition at line 15 of file RandomEngineSTL.hh.

```
00015                     {
00016     return std::generate_canonical<double, std::numeric_limits<double>::digits>(
00017         m_rng);
00018   }
```

GetSeed()

seed_t Garfield::RandomEngineSTL::GetSeed () [inline]
Definition at line 20 of file RandomEngineSTL.hh.
00020 { return m_seed; }

Print()

void Garfield::RandomEngineSTL::Print ()

SetSeed()

void Garfield::RandomEngineSTL::SetSeed (
const seed_t & seed) [inline]
Definition at line 19 of file RandomEngineSTL.hh.
00019 { m_rng.seed(seed); }

### 7.88.4 Member Data Documentation

m_rng

std::mt19937 Garfield::RandomEngineSTL::m_rng {std::random_device{}()} [private]
Definition at line 24 of file RandomEngineSTL.hh.
00024 {std::random_device{}()};
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/RandomEngineSTL.hh

## 7.89 Garfield::ComponentComsol::Range Struct Reference

Public Attributes

- bool set = false
- double xmin = 0
- double xmax = 0
- double ymin = 0
- double ymax = 0
- double zmin = 0
- double zmax = 0

### 7.89.1 Detailed Description

Definition at line 81 of file ComponentComsol.hh.

### 7.89.2 Member Data Documentation

set

bool Garfield::ComponentComsol::Range::set = false
Definition at line 82 of file ComponentComsol.hh.

xmax

double Garfield::ComponentComsol::Range::xmax = 0
Definition at line 85 of file ComponentComsol.hh.

xmin

double Garfield::ComponentComsol::Range::xmin = 0
Definition at line 84 of file ComponentComsol.hh.

ymax

double Garfield::ComponentComsol::Range::ymax = 0
Definition at line 88 of file ComponentComsol.hh.

ymin

double Garfield::ComponentComsol::Range::ymin = 0
Definition at line 87 of file ComponentComsol.hh.

zmax

double Garfield::ComponentComsol::Range::zmax = 0
Definition at line 91 of file ComponentComsol.hh.

zmin

double Garfield::ComponentComsol::Range::zmin = 0
Definition at line 90 of file ComponentComsol.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentComsol.hh

## 7.90 Garfield::ComponentNeBem2d::Region Struct Reference

Public Attributes

- std::vector< double > xv
  x-coordinates of the vertices.
- std::vector< double > yv
  y-coordinates of the vertices.
- Medium * medium
  Medium associated to the region.
- std::pair< BC, double > bc
  Applied boundary condition.
- unsigned int depth
  Level in the hierarchy.
- int ndiv
  Number of elements per edge segment.

### 7.90.1 Detailed Description

Definition at line 137 of file ComponentNeBem2d.hh.

### 7.90.2 Member Data Documentation

bc

std::pair<BC, double> Garfield::ComponentNeBem2d::Region::bc
Applied boundary condition.
Definition at line 141 of file ComponentNeBem2d.hh.

depth

unsigned int Garfield::ComponentNeBem2d::Region::depth
Level in the hierarchy.
Definition at line 142 of file ComponentNeBem2d.hh.

medium

Medium* Garfield::ComponentNeBem2d::Region::medium
Medium associated to the region.
Definition at line 140 of file ComponentNeBem2d.hh.

ndiv

int Garfield::ComponentNeBem2d::Region::ndiv
Number of elements per edge segment.
Definition at line 143 of file ComponentNeBem2d.hh.

xv

std::vector<double> Garfield::ComponentNeBem2d::Region::xv
x-coordinates of the vertices.
Definition at line 138 of file ComponentNeBem2d.hh.

yv

std::vector<double> Garfield::ComponentNeBem2d::Region::yv
y-coordinates of the vertices.
Definition at line 139 of file ComponentNeBem2d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh

## 7.91 Garfield::ComponentTcadBase< N >::Region Struct Reference

#include <ComponentTcadBase.hh>

Public Attributes

- std::string name
- std::string material
- bool drift
- Medium * medium

### 7.91.1 Detailed Description

template<size_t N>
struct Garfield::ComponentTcadBase< N >::Region

Definition at line 187 of file ComponentTcadBase.hh.

### 7.91.2 Member Data Documentation

drift

template<size_t N>
bool Garfield::ComponentTcadBase< N >::Region::drift
Definition at line 193 of file ComponentTcadBase.hh.

material

template<size_t N>
std::string Garfield::ComponentTcadBase< N >::Region::material
Definition at line 191 of file ComponentTcadBase.hh.

medium

template<size_t N>
Medium* Garfield::ComponentTcadBase< N >::Region::medium
Definition at line 195 of file ComponentTcadBase.hh.

name

template<size_t N>
std::string Garfield::ComponentTcadBase< N >::Region::name
Definition at line 189 of file ComponentTcadBase.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh

## 7.92 Garfield::AvalancheMC::Seed Struct Reference

#include <AvalancheMC.hh>

Public Attributes

- Point pt
  Starting point.
- Particle type
  Particle type.
- size_t w = 1
  Multiplicity.

### 7.92.1 Detailed Description

Definition at line 84 of file AvalancheMC.hh.

### 7.92.2 Member Data Documentation

pt

Point Garfield::AvalancheMC::Seed::pt
Starting point.
Definition at line 85 of file AvalancheMC.hh.

type

Particle Garfield::AvalancheMC::Seed::type
Particle type.
Definition at line 86 of file AvalancheMC.hh.

w

size_t Garfield::AvalancheMC::Seed::w = 1
Multiplicity.
Definition at line 87 of file AvalancheMC.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh

## 7.93 Garfield::AvalancheMicroscopic::Seed Struct Reference

#include <AvalancheMicroscopic.hh>

Public Attributes

- Point pt
  Starting point.
- Particle type
  Particle type.
- size_t w = 1
  Multiplicity.

### 7.93.1 Detailed Description

Definition at line 188 of file AvalancheMicroscopic.hh.

### 7.93.2 Member Data Documentation

pt

Point Garfield::AvalancheMicroscopic::Seed::pt
Starting point.
Definition at line 189 of file AvalancheMicroscopic.hh.

type

Particle Garfield::AvalancheMicroscopic::Seed::type
Particle type.
Definition at line 190 of file AvalancheMicroscopic.hh.

w

size_t Garfield::AvalancheMicroscopic::Seed::w = 1
Multiplicity.
Definition at line 191 of file AvalancheMicroscopic.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh

## 7.94 Garfield::ComponentNeBem2d::Segment Struct Reference

Public Attributes

- std::array< double, 2 > x0
  Coordinates of the start point.
- std::array< double, 2 > x1
  Coordinates of the end point.
- int region1
  Inner region.
- int region2
  Outer region.
- std::pair< BC, double > bc
  Applied boundary condition.
- int ndiv
  Number of elements.

### 7.94.1 Detailed Description

Definition at line 148 of file ComponentNeBem2d.hh.

### 7.94.2 Member Data Documentation

bc

std::pair<BC, double> Garfield::ComponentNeBem2d::Segment::bc
Applied boundary condition.
Definition at line 153 of file ComponentNeBem2d.hh.

ndiv

int Garfield::ComponentNeBem2d::Segment::ndiv
Number of elements.
Definition at line 154 of file ComponentNeBem2d.hh.

region1

int Garfield::ComponentNeBem2d::Segment::region1
Inner region.
Definition at line 151 of file ComponentNeBem2d.hh.

region2

int Garfield::ComponentNeBem2d::Segment::region2
Outer region.
Definition at line 152 of file ComponentNeBem2d.hh.

x0

std::array<double, 2> Garfield::ComponentNeBem2d::Segment::x0
Coordinates of the start point.
Definition at line 149 of file ComponentNeBem2d.hh.

x1

std::array<double, 2> Garfield::ComponentNeBem2d::Segment::x1
Coordinates of the end point.
Definition at line 150 of file ComponentNeBem2d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh

## 7.95 Garfield::Shaper Class Reference

Class for signal processing.
#include <Shaper.hh>

Public Member Functions

- Shaper ()=delete
  Default constructor.
- Shaper (const unsigned int n, const double tau, const double g, std::string shaperType)
  Constructor.
- ~Shaper ()
  Destructor.
- double Shape (const double t) const
  Evaluate the transfer function.
- double UnipolarShaper (const double t) const
  Transfer function for a unipolar shaper.
- double BipolarShaper (const double t) const
  Transfer function for a bipolar shaper.

- double PeakingTime () const
  Time for the transfer function to rise from zero to peak height.
- double TransferFuncSq () const
  Return the integral of the transfer function squared.
- bool IsUnipolar () const
  Is it a unipolar shaper?
- bool IsBipolar () const
  Is it a bipolar shaper?
- void GetParameters (unsigned int &n, double &tp)
  Retrieve the parameters.

Private Types

- enum class ShaperType { Unipolar = 0 , Bipolar }

Private Attributes

- std::string m_className = "Shaper"
- ShaperType m_type = ShaperType::Unipolar
- unsigned int m_n = 1
- double m_tau = 1.
- double m_tp = 1.
- double m_prefactor = 1.
- double m_g = 1.
- double m_transfer_func_sq = -1.

### 7.95.1 Detailed Description

Class for signal processing.
Definition at line 10 of file Shaper.hh.

### 7.95.2 Member Enumeration Documentation

ShaperType

enum class Garfield::Shaper::ShaperType [strong], [private]

Enumerator

| | |
|---|---|
| Unipolar | |
| Bipolar | |

Definition at line 46 of file Shaper.hh.
00046 { Unipolar = 0, Bipolar };

### 7.95.3 Constructor & Destructor Documentation

Shaper() [1/2]

Garfield::Shaper::Shaper () [delete]
Default constructor.

Shaper() [2/2]

```
Garfield::Shaper::Shaper (
                const unsigned int n,
                const double tau,
                const double g,
                std::string shaperType)
```

Constructor.

### ~Shaper()

Garfield::Shaper::~Shaper () [inline]
Destructor.
Definition at line 18 of file Shaper.hh.
00018 {}

## 7.95.4 Member Function Documentation

### BipolarShaper()

double Garfield::Shaper::BipolarShaper (
const double t) const
Transfer function for a bipolar shaper.

### GetParameters()

void Garfield::Shaper::GetParameters (
unsigned int & n,
double & tp) [inline]
Retrieve the parameters.
Definition at line 37 of file Shaper.hh.
```
00037                                          {
00038     n = m_n;
00039     tp = m_tp;
00040   }
```

### IsBipolar()

bool Garfield::Shaper::IsBipolar () const [inline]
Is it a bipolar shaper?
Definition at line 35 of file Shaper.hh.
00035 { return (m_type == ShaperType::Bipolar); }

### IsUnipolar()

bool Garfield::Shaper::IsUnipolar () const [inline]
Is it a unipolar shaper?
Definition at line 33 of file Shaper.hh.
00033 { return (m_type == ShaperType::Unipolar); }

### PeakingTime()

double Garfield::Shaper::PeakingTime () const [inline]
Time for the transfer function to rise from zero to peak height.
Definition at line 27 of file Shaper.hh.
00027 { return m_tp; }

### Shape()

double Garfield::Shaper::Shape (
const double t) const
Evaluate the transfer function.

### TransferFuncSq()

double Garfield::Shaper::TransferFuncSq () const [inline]
Return the integral of the transfer function squared.
Definition at line 30 of file Shaper.hh.
00030 { return m_transfer_func_sq; }

UnipolarShaper()

double Garfield::Shaper::UnipolarShaper (
const double t) const
Transfer function for a unipolar shaper.

### 7.95.5 Member Data Documentation

m_className

std::string Garfield::Shaper::m_className = "Shaper" [private]
Definition at line 43 of file Shaper.hh.

m_g

double Garfield::Shaper::m_g = 1. [private]
Definition at line 57 of file Shaper.hh.

m_n

unsigned int Garfield::Shaper::m_n = 1 [private]
Definition at line 49 of file Shaper.hh.

m_prefactor

double Garfield::Shaper::m_prefactor = 1. [private]
Definition at line 55 of file Shaper.hh.

m_tau

double Garfield::Shaper::m_tau = 1. [private]
Definition at line 51 of file Shaper.hh.

m_tp

double Garfield::Shaper::m_tp = 1. [private]
Definition at line 53 of file Shaper.hh.

m_transfer_func_sq

double Garfield::Shaper::m_transfer_func_sq = -1. [private]
Definition at line 59 of file Shaper.hh.

m_type

ShaperType Garfield::Shaper::m_type = ShaperType::Unipolar [private]
Definition at line 47 of file Shaper.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Shaper.hh

## 7.96 Garfield::Solid Class Reference

Abstract base class for solids.
#include <Solid.hh>
Inheritance diagram for Garfield::Solid:

Public Types

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Public Member Functions

- Solid ()=delete

  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)

  Constructor.
- virtual ~Solid ()

  Destructor.
- virtual bool IsInside (const double x, const double y, const double z, const bool tesselated=false) const =0

  Check whether a given point is inside the solid.
- virtual bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const =0

  Return the bounding box of the solid.
- virtual bool IsBox () const

  Return true if the solid is a box.
- virtual bool IsTube () const

  Return true if the solid is a tube.
- virtual bool IsSphere () const

  Return true if the solid is a sphere.
- virtual bool IsHole () const

  Return true if the solid is a hole.
- virtual bool IsRidge () const

  Return true if the solid is a ridge.
- virtual bool IsExtrusion () const

  Return true if the solid is an extrusion.
- virtual bool IsWire () const

  Return true if the solid is a wire.
- void SetLabel (const std::string &label)

  Set a label.
- std::string GetLabel () const

  Return the label.
- bool GetCentre (double &x, double &y, double &z) const

  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const

  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const

  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetHalfLengthX () const

  Return the half-length along x.
- virtual double GetHalfLengthY () const

  Return the half-length along y.
- virtual double GetHalfLengthZ () const

  Return the half-length along z.
- virtual double GetInnerRadius () const

  Return the inner radius.
- virtual double GetOuterRadius () const

Return the outer radius.

- virtual double GetRadius () const

  Return the radius.
- virtual double GetLowerRadius () const

  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const

  Return the upper radius (of a hole).
- virtual double GetRidgeOffset () const

  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const

  Return the height of a ridge.
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const

  Get the vertices defining an extrusion.
- unsigned int GetId () const

  Get the ID of the solid.
- virtual bool SolidPanels (std::vector< Panel > &panels)=0

  Retrieve the surface panels of the solid.
- virtual void SetDiscretisationLevel (const double dis)=0

  Set the discretisation level (for all panels).
- virtual double GetDiscretisationLevel (const Panel &panel)=0

  Retrieve the discretisation level of a panel.
- virtual void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels)=0
- void SetBoundaryPotential (const double v)

  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)

  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()

  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()

  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const

  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const

  Retrieve the potential.
- double GetBoundaryChargeDensity () const

  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off.
- void SetColour (const int col)

  Set the colour of the solid.
- int GetColour () const

  Get the colour of the solid.

Static Public Member Functions

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes

- double m_cX = 0.

  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.

  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.

  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.

  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"

  Class name.
- std::string m_label = ""

  Label.
- bool m_debug = false

  Debug flag.
- BoundaryCondition m_bctype = Unknown

  Type of boundary condition.
- double m_volt = 0.

  Potential at the surface.
- double m_charge = 0.

  Surface charge density.
- double m_eps = 0.

  Dielectric constant.
- int m_colour = -1

  Colour.

Private Member Functions

- double NotImplemented (const std::string &fcn) const

Private Attributes

- unsigned int m_id

  ID of the solid.

Static Private Attributes

- static unsigned int s_id

  ID counter.

### 7.96.1 Detailed Description

Abstract base class for solids.
Definition at line 28 of file Solid.hh.

### 7.96.2 Member Enumeration Documentation

BoundaryCondition

enum Garfield::Solid::BoundaryCondition

Enumerator

| | |
|---|---|
| Unknown | |
| Voltage | |
| Charge | |
| Float | |
| Dielectric | |
| DielectricCharge | |
| ParallelField | |
| PerpendicularField | |

Definition at line 151 of file Solid.hh.

```
00151                       {
00152     Unknown = 0,
00153     Voltage,
00154     Charge,
00155     Float,
00156     Dielectric,
00157     DielectricCharge,
00158     ParallelField,
00159     PerpendicularField
00160   };
```

### 7.96.3 Constructor & Destructor Documentation

Solid() [1/2]

Garfield::Solid::Solid () [delete]
Default constructor.

Solid() [2/2]

```
Garfield::Solid::Solid (
                const double cx,
                const double cy,
                const double cz,
                const std::string & name)    [inline]
```

Constructor.
Definition at line 33 of file Solid.hh.

```
00035       : m_cX(cx), m_cY(cy), m_cZ(cz), m_className(name) {
00036     m_id = s_id++;
00037   }
```

~Solid()

virtual Garfield::Solid::~Solid () [inline], [virtual]
Destructor.

Definition at line 40 of file Solid.hh.
00040 {}

### 7.96.4 Member Function Documentation

#### Cut()

virtual void Garfield::Solid::Cut (
const double x0,
const double y0,
const double z0,
const double xn,
const double yn,
const double zn,
std::vector< Panel > & panels) [pure virtual]

Implemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidRidge, Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.

#### EnableDebugging()

void Garfield::Solid::EnableDebugging (
const bool on = true) [inline]

Switch debugging messages on/off.
Definition at line 187 of file Solid.hh.
00187 { m_debug = on; }

#### GetBoundaryChargeDensity()

double Garfield::Solid::GetBoundaryChargeDensity () const [inline]

Retrieve the surface charge density.
Definition at line 184 of file Solid.hh.
00184 { return m_charge; }

#### GetBoundaryConditionType()

BoundaryCondition Garfield::Solid::GetBoundaryConditionType () const [inline]

Retrieve the type of boundary condition.
Definition at line 180 of file Solid.hh.
00180 { return m_bctype; }

#### GetBoundaryPotential()

double Garfield::Solid::GetBoundaryPotential () const [inline]

Retrieve the potential.
Definition at line 182 of file Solid.hh.
00182 { return m_volt; }

#### GetBoundingBox()

virtual bool Garfield::Solid::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) const [pure virtual]

Return the bounding box of the solid.
Implemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidRidge, Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.

### GetCentre()

```
bool Garfield::Solid::GetCentre (
                double & x,
                double & y,
                double & z) const   [inline]
```

Retrieve the centre point of the solid.
Definition at line 71 of file Solid.hh.

```
00071                                                {
00072     x = m_cX;
00073     y = m_cY;
00074     z = m_cZ;
00075     return true;
00076   }
```

### GetColour()

```
int Garfield::Solid::GetColour () const   [inline]
```

Get the colour of the solid.
Definition at line 192 of file Solid.hh.

```
00192 { return m_colour; }
```

### GetDirection()

```
bool Garfield::Solid::GetDirection (
                double & dx,
                double & dy,
                double & dz) const   [inline]
```

Retrieve the direction vector.
Definition at line 78 of file Solid.hh.

```
00078                                                    {
00079     dx = m_dX;
00080     dy = m_dY;
00081     dz = m_dZ;
00082     return true;
00083   }
```

### GetDiscretisationLevel()

```
virtual double Garfield::Solid::GetDiscretisationLevel (
                const Panel & panel)   [pure virtual]
```

Retrieve the discretisation level of a panel.
Implemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidRidge, Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.

### GetHalfLengthX()

```
virtual double Garfield::Solid::GetHalfLengthX () const   [inline], [virtual]
```

Return the half-length along x.
Reimplemented in Garfield::SolidBox, Garfield::SolidHole, and Garfield::SolidRidge.
Definition at line 95 of file Solid.hh.

```
00095                                   {
00096     return NotImplemented("GetHalfLengthX");
00097   }
```

### GetHalfLengthY()

```
virtual double Garfield::Solid::GetHalfLengthY () const   [inline], [virtual]
```

Return the half-length along y.
Reimplemented in Garfield::SolidBox, Garfield::SolidHole, and Garfield::SolidRidge.
Definition at line 99 of file Solid.hh.

```
00099                                   {
00100     return NotImplemented("GetHalfLengthY");
00101   }
```

### GetHalfLengthZ()

virtual double Garfield::Solid::GetHalfLengthZ () const [inline], [virtual]

Return the half-length along z.

Reimplemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidTube, and Garfield::SolidWire.

Definition at line 103 of file Solid.hh.

```
00103                                  {
00104     return NotImplemented("GetHalfLengthZ");
00105   }
```

### GetId()

unsigned int Garfield::Solid::GetId () const [inline]

Get the ID of the solid.

Definition at line 137 of file Solid.hh.

```
00137 { return m_id; }
```

### GetInnerRadius()

virtual double Garfield::Solid::GetInnerRadius () const [inline], [virtual]

Return the inner radius.

Reimplemented in Garfield::SolidSphere, and Garfield::SolidTube.

Definition at line 107 of file Solid.hh.

```
00107                                  {
00108     return NotImplemented("GetInnerRadius");
00109   }
```

### GetLabel()

std::string Garfield::Solid::GetLabel () const [inline]

Return the label.

Definition at line 68 of file Solid.hh.

```
00068 { return m_label; }
```

### GetLowerRadius()

virtual double Garfield::Solid::GetLowerRadius () const [inline], [virtual]

Return the lower radius (of a hole).

Reimplemented in Garfield::SolidHole.

Definition at line 117 of file Solid.hh.

```
00117                                  {
00118     return NotImplemented("GetLowerRadius");
00119   }
```

### GetOrientation()

bool Garfield::Solid::GetOrientation (
double & ctheta,
double & stheta,
double & cphi,
double & sphi) const [inline]

Retrieve the orientation (azimuthal and polar angles) of the solid.

Definition at line 85 of file Solid.hh.

```
00086                                  {
00087     ctheta = m_cTheta;
00088     stheta = m_sTheta;
00089     cphi = m_cPhi;
00090     sphi = m_sPhi;
00091     return true;
00092   }
```

### GetOuterRadius()

virtual double Garfield::Solid::GetOuterRadius () const [inline], [virtual]

Return the outer radius.
Reimplemented in Garfield::SolidSphere, and Garfield::SolidTube.
Definition at line 111 of file Solid.hh.

```
00111                                 {
00112     return NotImplemented("GetOuterRadius");
00113   }
```

### GetProfile()

```
virtual bool Garfield::Solid::GetProfile (
                std::vector< double > & xv,
                std::vector< double > & yv) const   [virtual]
```

Get the vertices defining an extrusion.
Reimplemented in Garfield::SolidExtrusion.

### GetRadius()

```
virtual double Garfield::Solid::GetRadius () const   [inline], [virtual]
```

Return the radius.
Reimplemented in Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.
Definition at line 115 of file Solid.hh.

```
00115 { return NotImplemented("GetRadius"); }
```

### GetRidgeHeight()

```
virtual double Garfield::Solid::GetRidgeHeight () const   [inline], [virtual]
```

Return the height of a ridge.
Reimplemented in Garfield::SolidRidge.
Definition at line 129 of file Solid.hh.

```
00129                                 {
00130     return NotImplemented("GetRidgeHeight");
00131   }
```

### GetRidgeOffset()

```
virtual double Garfield::Solid::GetRidgeOffset () const   [inline], [virtual]
```

Return the x-offset of a ridge.
Reimplemented in Garfield::SolidRidge.
Definition at line 125 of file Solid.hh.

```
00125                                 {
00126     return NotImplemented("GetRidgeOffset");
00127   }
```

### GetUpperRadius()

```
virtual double Garfield::Solid::GetUpperRadius () const   [inline], [virtual]
```

Return the upper radius (of a hole).
Reimplemented in Garfield::SolidHole.
Definition at line 121 of file Solid.hh.

```
00121                                 {
00122     return NotImplemented("GetUpperRadius");
00123   }
```

### Intersect()

```
bool Garfield::Solid::Intersect (
                const double x1,
                const double y1,
                const double z1,
                const double x2,
                const double y2,
                const double z2,
                const double x0,
```

const double y0,
const double z0,
const double a,
const double b,
const double c,
double & xc,
double & yc,
double & zc) [static]

### IsBox()

virtual bool Garfield::Solid::IsBox () const [inline], [virtual]
Return true if the solid is a box.
Reimplemented in Garfield::SolidBox.
Definition at line 51 of file Solid.hh.

```
00051 { return false; }
```

### IsExtrusion()

virtual bool Garfield::Solid::IsExtrusion () const [inline], [virtual]
Return true if the solid is an extrusion.
Reimplemented in Garfield::SolidExtrusion.
Definition at line 61 of file Solid.hh.

```
00061 { return false; }
```

### IsHole()

virtual bool Garfield::Solid::IsHole () const [inline], [virtual]
Return true if the solid is a hole.
Reimplemented in Garfield::SolidHole.
Definition at line 57 of file Solid.hh.

```
00057 { return false; }
```

### IsInside()

virtual bool Garfield::Solid::IsInside (
const double x,
const double y,
const double z,
const bool tesselated = false) const [pure virtual]
Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidRidge, Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.

### IsRidge()

virtual bool Garfield::Solid::IsRidge () const [inline], [virtual]
Return true if the solid is a ridge.
Reimplemented in Garfield::SolidRidge.
Definition at line 59 of file Solid.hh.

```
00059 { return false; }
```

### IsSphere()

virtual bool Garfield::Solid::IsSphere () const [inline], [virtual]
Return true if the solid is a sphere.
Reimplemented in Garfield::SolidSphere.
Definition at line 55 of file Solid.hh.

```
00055 { return false; }
```

### IsTube()

virtual bool Garfield::Solid::IsTube () const [inline], [virtual]
Return true if the solid is a tube.
Reimplemented in Garfield::SolidTube.
Definition at line 53 of file Solid.hh.

```
00053 { return false; }
```

### IsWire()

virtual bool Garfield::Solid::IsWire () const [inline], [virtual]
Return true if the solid is a wire.
Reimplemented in Garfield::SolidWire.
Definition at line 63 of file Solid.hh.

```
00063 { return false; }
```

### NotImplemented()

double Garfield::Solid::NotImplemented (
const std::string & fcn) const [private]

### SetBoundaryChargeDensity()

void Garfield::Solid::SetBoundaryChargeDensity (
const double q) [inline]
Apply fixed-charge boundary conditions.
Definition at line 168 of file Solid.hh.

```
00168                                          {
00169     m_charge = q;
00170     m_bctype = Charge;
00171   }
```

### SetBoundaryDielectric()

void Garfield::Solid::SetBoundaryDielectric () [inline]
Make the surfaces of the solid dielectric-dielectric interfaces.
Definition at line 175 of file Solid.hh.

```
00175 { m_bctype = Dielectric; }
```

### SetBoundaryFloat()

void Garfield::Solid::SetBoundaryFloat () [inline]
Make the potential at the surface of the solid floating.
Definition at line 173 of file Solid.hh.

```
00173 { m_bctype = Float; }
```

### SetBoundaryParallelField()

void Garfield::Solid::SetBoundaryParallelField () [inline]
Definition at line 176 of file Solid.hh.

```
00176 { m_bctype = ParallelField; }
```

### SetBoundaryPerpendicularField()

void Garfield::Solid::SetBoundaryPerpendicularField () [inline]
Definition at line 177 of file Solid.hh.

```
00177 { m_bctype = PerpendicularField; }
```

### SetBoundaryPotential()

void Garfield::Solid::SetBoundaryPotential (
const double v) [inline]

Apply Dirichlet boundary conditions (fixed voltage).
Definition at line 163 of file Solid.hh.

```
00163                                   {
00164     m_volt = v;
00165     m_bctype = Voltage;
00166   }
```

SetColour()

```
void Garfield::Solid::SetColour (
                const int col)   [inline]
```

Set the colour of the solid.
Definition at line 190 of file Solid.hh.

```
00190 { m_colour = col; }
```

SetDirection()

```
void Garfield::Solid::SetDirection (
                const double dx,
                const double dy,
                const double dz)   [protected]
```

SetDiscretisationLevel()

```
virtual void Garfield::Solid::SetDiscretisationLevel (
                const double dis)   [pure virtual]
```

Set the discretisation level (for all panels).
Implemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidRidge, Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.

SetLabel()

```
void Garfield::Solid::SetLabel (
                const std::string & label)   [inline]
```

Set a label.
Definition at line 66 of file Solid.hh.

```
00066 { m_label = label; }
```

SolidPanels()

```
virtual bool Garfield::Solid::SolidPanels (
                std::vector< Panel > & panels)   [pure virtual]
```

Retrieve the surface panels of the solid.
Implemented in Garfield::SolidBox, Garfield::SolidExtrusion, Garfield::SolidHole, Garfield::SolidRidge, Garfield::SolidSphere, Garfield::SolidTube, and Garfield::SolidWire.

ToGlobal()

```
void Garfield::Solid::ToGlobal (
                const double u,
                const double v,
                const double w,
                double & x,
                double & y,
                double & z) const   [inline], [protected]
```

Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
Definition at line 246 of file Solid.hh.

```
00247                                   {
00248     x = m_cX + m_cPhi * m_cTheta * u - m_sPhi * v + m_cPhi * m_sTheta * w;
00249     y = m_cY + m_sPhi * m_cTheta * u + m_cPhi * v + m_sPhi * m_sTheta * w;
00250     z = m_cZ - m_sTheta * u + m_cTheta * w;
00251   }
```

### ToLocal()

```
void Garfield::Solid::ToLocal (
                const double x,
                const double y,
                const double z,
                double & u,
                double & v,
                double & w) const    [inline], [protected]
```

Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
Definition at line 234 of file Solid.hh.

```
00235                                       {
00236     const double dx = x - m_cX;
00237     const double dy = y - m_cY;
00238     const double dz = z - m_cZ;
00239
00240     u = m_cPhi * m_cTheta * dx + m_sPhi * m_cTheta * dy - m_sTheta * dz;
00241     v = -m_sPhi * dx + m_cPhi * dy;
00242     w = m_cPhi * m_sTheta * dx + m_sPhi * m_sTheta * dy + m_cTheta * dz;
00243   }
```

### VectorToLocal()

```
void Garfield::Solid::VectorToLocal (
                const double x,
                const double y,
                const double z,
                double & u,
                double & v,
                double & w)    [inline], [protected]
```

Transform a vector from global to local coordinates.
Definition at line 253 of file Solid.hh.

```
00254                                       {
00255     u = m_cPhi * m_cTheta * x + m_sPhi * m_cTheta * y - m_sTheta * z;
00256     v = -m_sPhi * x + m_cPhi * y;
00257     w = m_cPhi * m_sTheta * x + m_sPhi * m_sTheta * y + m_cTheta * z;
00258   }
```

## 7.96.5 Member Data Documentation

### m_bctype

```
BoundaryCondition Garfield::Solid::m_bctype = Unknown    [protected]
```

Type of boundary condition.
Definition at line 221 of file Solid.hh.

### m_charge

```
double Garfield::Solid::m_charge = 0.    [protected]
```

Surface charge density.
Definition at line 225 of file Solid.hh.

### m_className

```
std::string Garfield::Solid::m_className = "Solid"    [protected]
```

Class name.
Definition at line 212 of file Solid.hh.

### m_colour

```
int Garfield::Solid::m_colour = -1    [protected]
```

Colour.
Definition at line 230 of file Solid.hh.

m_cPhi

double Garfield::Solid::m_cPhi = 1. [protected]
Azimuthal angle.
Definition at line 207 of file Solid.hh.

m_cTheta

double Garfield::Solid::m_cTheta = 1. [protected]
Polar angle.
Definition at line 209 of file Solid.hh.

m_cX

double Garfield::Solid::m_cX = 0. [protected]
Centre of the solid.
Definition at line 202 of file Solid.hh.

m_cY

double Garfield::Solid::m_cY = 0. [protected]
Definition at line 202 of file Solid.hh.

m_cZ

double Garfield::Solid::m_cZ = 0. [protected]
Definition at line 202 of file Solid.hh.

m_debug

bool Garfield::Solid::m_debug = false [protected]
Debug flag.
Definition at line 218 of file Solid.hh.

m_dX

double Garfield::Solid::m_dX = 0. [protected]
Direction vector.
Definition at line 205 of file Solid.hh.

m_dY

double Garfield::Solid::m_dY = 0. [protected]
Definition at line 205 of file Solid.hh.

m_dZ

double Garfield::Solid::m_dZ = 1. [protected]
Definition at line 205 of file Solid.hh.

m_eps

double Garfield::Solid::m_eps = 0. [protected]
Dielectric constant.
Definition at line 227 of file Solid.hh.

m_id

unsigned int Garfield::Solid::m_id [private]
ID of the solid.
Definition at line 267 of file Solid.hh.

m_label

std::string Garfield::Solid::m_label = "" [protected]
Label.
Definition at line 215 of file Solid.hh.

m_sPhi

double Garfield::Solid::m_sPhi = 0. [protected]
Definition at line 207 of file Solid.hh.

m_sTheta

double Garfield::Solid::m_sTheta = 0. [protected]
Definition at line 209 of file Solid.hh.

m_volt

double Garfield::Solid::m_volt = 0. [protected]
Potential at the surface.
Definition at line 223 of file Solid.hh.

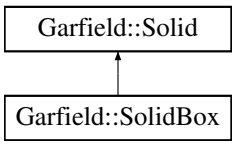
s_id

unsigned int Garfield::Solid::s_id [static], [private]
ID counter.
Definition at line 265 of file Solid.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Solid.hh

## 7.97 Garfield::SolidBox Class Reference

Box.
#include <SolidBox.hh>
Inheritance diagram for Garfield::SolidBox:




Public Member Functions

- SolidBox (const double cx, const double cy, const double cz, const double lx, const double ly, const double lz)

  Constructor from centre and half-widths.
- SolidBox (const double cx, const double cy, const double cz, const double lx, const double ly, const double lz, const double dx, const double dy, const double dz)

  Constructor from centre, half-widths, and orientation.
- ~SolidBox ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.

- bool IsBox () const override

  Return true if the solid is a box.
- double GetHalfLengthX () const override

  Return the half-length along x.
- double GetHalfLengthY () const override

  Return the half-length along y.
- double GetHalfLengthZ () const override

  Return the half-length along z.
- void SetHalfLengthX (const double lx)
- void SetHalfLengthY (const double ly)
- void SetHalfLengthZ (const double lz)
- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete

  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)

  Constructor.
- virtual ~Solid ()

  Destructor.
- virtual bool IsTube () const

  Return true if the solid is a tube.
- virtual bool IsSphere () const

  Return true if the solid is a sphere.
- virtual bool IsHole () const

  Return true if the solid is a hole.
- virtual bool IsRidge () const

  Return true if the solid is a ridge.
- virtual bool IsExtrusion () const

  Return true if the solid is an extrusion.
- virtual bool IsWire () const

  Return true if the solid is a wire.
- void SetLabel (const std::string &label)

  Set a label.
- std::string GetLabel () const

  Return the label.
- bool GetCentre (double &x, double &y, double &z) const

  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const

  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const

  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetInnerRadius () const

  Return the inner radius.

- virtual double GetOuterRadius () const
  Return the outer radius.
- virtual double GetRadius () const
  Return the radius.
- virtual double GetLowerRadius () const
  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const
  Return the upper radius (of a hole).
- virtual double GetRidgeOffset () const
  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const
  Return the height of a ridge.
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const
  Get the vertices defining an extrusion.
- unsigned int GetId () const
  Get the ID of the solid.
- void SetBoundaryPotential (const double v)
  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)
  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()
  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()
  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const
  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const
  Retrieve the potential.
- double GetBoundaryChargeDensity () const
  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)
  Switch debugging messages on/off.
- void SetColour (const int col)
  Set the colour of the solid.
- int GetColour () const
  Get the colour of the solid.

Private Attributes

- double m_lX = 0.
  Half lengths.
- double m_lY = 0.
- double m_lZ = 0.
- std::array< double, 6 > m_dis {{-1, -1, -1, -1, -1, -1}}
  Discretisation levels.

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.

  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.

  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.

  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.

  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"

  Class name.
- std::string m_label = ""

  Label.
- bool m_debug = false

  Debug flag.
- BoundaryCondition m_bctype = Unknown

  Type of boundary condition.
- double m_volt = 0.

  Potential at the surface.
- double m_charge = 0.

  Surface charge density.
- double m_eps = 0.

  Dielectric constant.
- int m_colour = -1

  Colour.

### 7.97.1 Detailed Description

Box.
Definition at line 12 of file SolidBox.hh.

### 7.97.2 Constructor & Destructor Documentation

#### SolidBox() [1/2]

```
Garfield::SolidBox::SolidBox (
            const double cx,
            const double cy,
            const double cz,
            const double lx,
            const double ly,
            const double lz)
```

Constructor from centre and half-widths.

#### SolidBox() [2/2]

```
Garfield::SolidBox::SolidBox (
            const double cx,
            const double cy,
            const double cz,
            const double lx,
            const double ly,
            const double lz,
            const double dx,
            const double dy,
            const double dz)
```

Constructor from centre, half-widths, and orientation.

#### ~SolidBox()

```
Garfield::SolidBox::~SolidBox ()  [inline]
```

Destructor.
Definition at line 22 of file SolidBox.hh.

```
00022 {}
```

### 7.97.3 Member Function Documentation

#### Cut()

```
void Garfield::SolidBox::Cut (
            const double x0,
            const double y0,
            const double z0,
            const double xn,
            const double yn,
            const double zn,
            std::vector< Panel > & panels)  [override], [virtual]
```

Implements Garfield::Solid.

#### GetBoundingBox()

```
bool Garfield::SolidBox::GetBoundingBox (
            double & xmin,
            double & ymin,
            double & zmin,
            double & xmax,
            double & ymax,
            double & zmax) const  [override], [virtual]
```

Return the bounding box of the solid.
Implements Garfield::Solid.

GetDiscretisationLevel()

double Garfield::SolidBox::GetDiscretisationLevel (
const Panel & panel) [override], [virtual]
Retrieve the discretisation level of a panel.
Implements Garfield::Solid.

GetHalfLengthX()

double Garfield::SolidBox::GetHalfLengthX () const [inline], [override], [virtual]
Return the half-length along x.
Reimplemented from Garfield::Solid.
Definition at line 30 of file SolidBox.hh.
00030 { return m_lX; }

GetHalfLengthY()

double Garfield::SolidBox::GetHalfLengthY () const [inline], [override], [virtual]
Return the half-length along y.
Reimplemented from Garfield::Solid.
Definition at line 31 of file SolidBox.hh.
00031 { return m_lY; }

GetHalfLengthZ()

double Garfield::SolidBox::GetHalfLengthZ () const [inline], [override], [virtual]
Return the half-length along z.
Reimplemented from Garfield::Solid.
Definition at line 32 of file SolidBox.hh.
00032 { return m_lZ; }

IsBox()

bool Garfield::SolidBox::IsBox () const [inline], [override], [virtual]
Return true if the solid is a box.
Reimplemented from Garfield::Solid.
Definition at line 28 of file SolidBox.hh.
00028 { return true; }

IsInside()

bool Garfield::SolidBox::IsInside (
const double x,
const double y,
const double z,
const bool tesselated) const [override], [virtual]
Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implements Garfield::Solid.

SetDiscretisationLevel()

void Garfield::SolidBox::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]
Set the discretisation level (for all panels).
Implements Garfield::Solid.
Definition at line 39 of file SolidBox.hh.
00039 { m_dis.fill(dis); }

SetHalfLengthX()

void Garfield::SolidBox::SetHalfLengthX (
const double lx)

SetHalfLengthY()

void Garfield::SolidBox::SetHalfLengthY (
const double ly)

SetHalfLengthZ()

void Garfield::SolidBox::SetHalfLengthZ (
const double lz)

SolidPanels()

bool Garfield::SolidBox::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

### 7.97.4 Member Data Documentation

m_dis

std::array<double, 6> Garfield::SolidBox::m_dis {{-1, -1, -1, -1, -1, -1}} [private]
Discretisation levels.
Definition at line 50 of file SolidBox.hh.
00050 {{-1, -1, -1, -1, -1, -1}};

m_lX

double Garfield::SolidBox::m_lX = 0. [private]
Half lengths.
Definition at line 48 of file SolidBox.hh.

m_lY

double Garfield::SolidBox::m_lY = 0. [private]
Definition at line 48 of file SolidBox.hh.

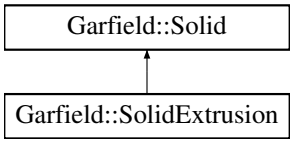
m_lZ

double Garfield::SolidBox::m_lZ = 0. [private]
Definition at line 48 of file SolidBox.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidBox.hh

## 7.98 Garfield::SolidExtrusion Class Reference

Extrusion.
#include <SolidExtrusion.hh>
Inheritance diagram for Garfield::SolidExtrusion:



Public Member Functions

- SolidExtrusion (const double lz, const std::vector< double > &xp, const std::vector< double > &yp)

  Constructor from half-length and profile.
- SolidExtrusion (const double lz, const std::vector< double > &xp, const std::vector< double > &yp, const double cx, const double cy, const double cz, const double dx, const double dy, const double dz)

  Constructor from half-length, profile, offset and orientation.
- ~SolidExtrusion ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.
- bool IsExtrusion () const override

  Return true if the solid is an extrusion.
- double GetHalfLengthZ () const override

  Return the half-length along z.
- bool GetProfile (std::vector< double > &xp, std::vector< double > &yp) const override

  Get the vertices defining an extrusion.
- void SetHalfLengthZ (const double lz)

  Set the half-length of the extrusion.
- void SetProfile (const std::vector< double > &xp, const std::vector< double > &yp)

  Set the coordinates of the extrusion profile.
- void SetTopLid (const bool closed)

  Request the extrusion to be closed with a lid at +z.
- void SetBottomLid (const bool closed)

  Request the extrusion to be closed with a lid at -z.
- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete

  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)

  Constructor.
- virtual ~Solid ()

  Destructor.
- virtual bool IsBox () const

  Return true if the solid is a box.
- virtual bool IsTube () const

  Return true if the solid is a tube.
- virtual bool IsSphere () const

  Return true if the solid is a sphere.
- virtual bool IsHole () const

Return true if the solid is a hole.

- virtual bool IsRidge () const

  Return true if the solid is a ridge.
- virtual bool IsWire () const

  Return true if the solid is a wire.
- void SetLabel (const std::string &label)

  Set a label.
- std::string GetLabel () const

  Return the label.
- bool GetCentre (double &x, double &y, double &z) const

  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const

  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const

  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetHalfLengthX () const

  Return the half-length along x.
- virtual double GetHalfLengthY () const

  Return the half-length along y.
- virtual double GetInnerRadius () const

  Return the inner radius.
- virtual double GetOuterRadius () const

  Return the outer radius.
- virtual double GetRadius () const

  Return the radius.
- virtual double GetLowerRadius () const

  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const

  Return the upper radius (of a hole).
- virtual double GetRidgeOffset () const

  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const

  Return the height of a ridge.
- unsigned int GetId () const

  Get the ID of the solid.
- void SetBoundaryPotential (const double v)

  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)

  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()

  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()

  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const

  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const

  Retrieve the potential.
- double GetBoundaryChargeDensity () const

  Retrieve the surface charge density.

- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off.
- void SetColour (const int col)

  Set the colour of the solid.
- int GetColour () const

  Get the colour of the solid.

Private Attributes

- double m_lZ = 0.

  Half length.
- std::vector< double > m_xp

  X coordinates of the profile.
- std::vector< double > m_yp

  Y coordinates of the profile.
- bool m_toplid = true

  Have a top lid?
- bool m_botlid = true

  Have a bottom lid?
- bool m_clockwise = true

  Orientation of the polygon.
- std::array< double, 3 > m_dis {{-1, -1, -1}}

  Discretisation levels.

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.
  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.
  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.
  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.
  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"
  Class name.
- std::string m_label = ""
  Label.
- bool m_debug = false
  Debug flag.
- BoundaryCondition m_bctype = Unknown
  Type of boundary condition.
- double m_volt = 0.
  Potential at the surface.
- double m_charge = 0.
  Surface charge density.
- double m_eps = 0.
  Dielectric constant.
- int m_colour = -1
  Colour.

### 7.98.1 Detailed Description

Extrusion.
Definition at line 12 of file SolidExtrusion.hh.

### 7.98.2 Constructor & Destructor Documentation

SolidExtrusion() [1/2]

```
Garfield::SolidExtrusion::SolidExtrusion (
            const double lz,
            const std::vector< double > & xp,
            const std::vector< double > & yp)
```

Constructor from half-length and profile.

SolidExtrusion() [2/2]

```
Garfield::SolidExtrusion::SolidExtrusion (
            const double lz,
            const std::vector< double > & xp,
            const std::vector< double > & yp,
            const double cx,
            const double cy,
```

const double cz,
const double dx,
const double dy,
const double dz)

Constructor from half-length, profile, offset and orientation.

### ~SolidExtrusion()

Garfield::SolidExtrusion::~SolidExtrusion () [inline]

Destructor.

Definition at line 23 of file SolidExtrusion.hh.

```
00023 {}
```

## 7.98.3 Member Function Documentation

### Cut()

void Garfield::SolidExtrusion::Cut (
const double x0,
const double y0,
const double z0,
const double xn,
const double yn,
const double zn,
std::vector< Panel > & panels) [override], [virtual]

Implements Garfield::Solid.

### GetBoundingBox()

bool Garfield::SolidExtrusion::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) const [override], [virtual]

Return the bounding box of the solid.

Implements Garfield::Solid.

### GetDiscretisationLevel()

double Garfield::SolidExtrusion::GetDiscretisationLevel (
const Panel & panel) [override], [virtual]

Retrieve the discretisation level of a panel.

Implements Garfield::Solid.

### GetHalfLengthZ()

double Garfield::SolidExtrusion::GetHalfLengthZ () const [inline], [override], [virtual]

Return the half-length along z.

Reimplemented from Garfield::Solid.

Definition at line 31 of file SolidExtrusion.hh.

```
00031 { return m_lZ; }
```

### GetProfile()

bool Garfield::SolidExtrusion::GetProfile (
std::vector< double > & xv,
std::vector< double > & yv) const [inline], [override], [virtual]

Get the vertices defining an extrusion.

Reimplemented from Garfield::Solid.

Definition at line 32 of file SolidExtrusion.hh.

```
00033                                                {
00034     xp = m_xp;
00035     yp = m_yp;
00036     return !m_xp.empty();
00037   }
```

### IsExtrusion()

bool Garfield::SolidExtrusion::IsExtrusion () const [inline], [override], [virtual]

Return true if the solid is an extrusion.

Reimplemented from Garfield::Solid.

Definition at line 29 of file SolidExtrusion.hh.

```
00029 { return true; }
```

### IsInside()

bool Garfield::SolidExtrusion::IsInside (
const double x,
const double y,
const double z,
const bool tesselated) const [override], [virtual]

Check whether a given point is inside the solid.

If requested, use the tesselated approximation of the solid (if applicable).

Implements Garfield::Solid.

### SetBottomLid()

void Garfield::SolidExtrusion::SetBottomLid (
const bool closed) [inline]

Request the extrusion to be closed with a lid at -z.

Definition at line 45 of file SolidExtrusion.hh.

```
00045 { m_botlid = closed; }
```

### SetDiscretisationLevel()

void Garfield::SolidExtrusion::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]

Set the discretisation level (for all panels).

Implements Garfield::Solid.

Definition at line 48 of file SolidExtrusion.hh.

```
00048 { m_dis.fill(dis); }
```

### SetHalfLengthZ()

void Garfield::SolidExtrusion::SetHalfLengthZ (
const double lz)

Set the half-length of the extrusion.

### SetProfile()

void Garfield::SolidExtrusion::SetProfile (
const std::vector< double > & xp,
const std::vector< double > & yp)

Set the coordinates of the extrusion profile.

### SetTopLid()

void Garfield::SolidExtrusion::SetTopLid (
const bool closed) [inline]

Request the extrusion to be closed with a lid at +z.

Definition at line 43 of file SolidExtrusion.hh.

00043 { m_toplid = closed; }

SolidPanels()

bool Garfield::SolidExtrusion::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

### 7.98.4 Member Data Documentation

m_botlid

bool Garfield::SolidExtrusion::m_botlid = true [private]
Have a bottom lid?
Definition at line 66 of file SolidExtrusion.hh.

m_clockwise

bool Garfield::SolidExtrusion::m_clockwise = true [private]
Orientation of the polygon.
Definition at line 69 of file SolidExtrusion.hh.

m_dis

std::array<double, 3> Garfield::SolidExtrusion::m_dis {{-1, -1, -1}} [private]
Discretisation levels.
Definition at line 72 of file SolidExtrusion.hh.
00072 {{-1, -1, -1}};

m_lZ

double Garfield::SolidExtrusion::m_lZ = 0. [private]
Half length.
Definition at line 57 of file SolidExtrusion.hh.

m_toplid

bool Garfield::SolidExtrusion::m_toplid = true [private]
Have a top lid?
Definition at line 64 of file SolidExtrusion.hh.

m_xp

std::vector<double> Garfield::SolidExtrusion::m_xp [private]
X coordinates of the profile.
Definition at line 59 of file SolidExtrusion.hh.

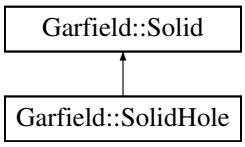
m_yp

std::vector<double> Garfield::SolidExtrusion::m_yp [private]
Y coordinates of the profile.
Definition at line 61 of file SolidExtrusion.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidExtrusion.hh

## 7.99 Garfield::SolidHole Class Reference

Box with a cylindrical hole.
#include <SolidHole.hh>
Inheritance diagram for Garfield::SolidHole:




Public Member Functions

- SolidHole (const double cx, const double cy, const double cz, const double rup, const double rlow, const double lx, const double ly, const double lz)

  Constructor from centre, upper/lower radii, half-lengths of the box.
- SolidHole (const double cx, const double cy, const double cz, const double rup, const double rlow, const double lx, const double ly, const double lz, const double dx, const double dy, const double dz)

  Constructor from centre, upper/lower radii, half-lengths of the box and orientation.
- ~SolidHole ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.
- bool IsHole () const override

  Return true if the solid is a hole.
- void SetHalfLengthX (const double lx)

  Set the half-length of the box along x.
- void SetHalfLengthY (const double ly)

  Set the half-length of the box along y.
- void SetHalfLengthZ (const double lz)

  Set the half-length of the box along z.
- void SetUpperRadius (const double r)

  Set the radius at z = +lz.
- void SetLowerRadius (const double r)

  Set the radius at z = -lz.
- double GetHalfLengthX () const override

  Return the half-length along x.
- double GetHalfLengthY () const override

  Return the half-length along y.
- double GetHalfLengthZ () const override

  Return the half-length along z.
- double GetUpperRadius () const override

  Return the upper radius (of a hole).
- double GetLowerRadius () const override

  Return the lower radius (of a hole).
- void SetSectors (const unsigned int n)

  When calculating the surface panels, the hole is approximated as a polygon with a finite number of panels.
- void SetAverageRadius (const bool average)

  By default, the polygon used for approximating the hole when calculating surface panels is inscribed in a circle of the specified radius.
- unsigned int GetSectors () const

  Return the order of the approximating polygon.
- bool GetAverage () const

Return the state of the "average-radius" flag.

- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete

  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)

  Constructor.
- virtual ~Solid ()

  Destructor.
- virtual bool IsBox () const

  Return true if the solid is a box.
- virtual bool IsTube () const

  Return true if the solid is a tube.
- virtual bool IsSphere () const

  Return true if the solid is a sphere.
- virtual bool IsRidge () const

  Return true if the solid is a ridge.
- virtual bool IsExtrusion () const

  Return true if the solid is an extrusion.
- virtual bool IsWire () const

  Return true if the solid is a wire.
- void SetLabel (const std::string &label)

  Set a label.
- std::string GetLabel () const

  Return the label.
- bool GetCentre (double &x, double &y, double &z) const

  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const

  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const

  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetInnerRadius () const

  Return the inner radius.
- virtual double GetOuterRadius () const

  Return the outer radius.
- virtual double GetRadius () const

  Return the radius.
- virtual double GetRidgeOffset () const

  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const

  Return the height of a ridge.
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const

  Get the vertices defining an extrusion.

- unsigned int GetId () const

  Get the ID of the solid.
- void SetBoundaryPotential (const double v)

  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)

  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()

  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()

  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const

  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const

  Retrieve the potential.
- double GetBoundaryChargeDensity () const

  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off.
- void SetColour (const int col)

  Set the colour of the solid.
- int GetColour () const

  Get the colour of the solid.

Private Member Functions

- void Update ()

Private Attributes

- std::mutex m_mutex

  Mutex.
- double m_rUp

  Upper radius.
- double m_rLow

  Lower radius.
- double m_lX

  Half-length in x.
- double m_lY

  Half-length in y.
- double m_lZ

  Half-length in z.
- unsigned int m_n = 2

  Number of sectors.
- bool m_average = false

  Average chord over the sectors.
- double m_fp = 1.

  Ratio between the approximating polygon's radius and the hole radius.
- double m_fi = 1.

  Ratio between inradius and exradius of the approximating polygon.
- std::array< double, 7 > m_dis {{-1., -1., -1., -1., -1., -1., -1.}}

  Discretisation levels.

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.

  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.

  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.

  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.

  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"

  Class name.
- std::string m_label = ""

  Label.
- bool m_debug = false

  Debug flag.
- BoundaryCondition m_bctype = Unknown

  Type of boundary condition.
- double m_volt = 0.

  Potential at the surface.
- double m_charge = 0.

  Surface charge density.
- double m_eps = 0.

Dielectric constant.

- int m_colour = -1

  Colour.

### 7.99.1 Detailed Description

Box with a cylindrical hole.
Definition at line 13 of file SolidHole.hh.

### 7.99.2 Constructor & Destructor Documentation

SolidHole() [1/2]

```
Garfield::SolidHole::SolidHole (
            const double cx,
            const double cy,
            const double cz,
            const double rup,
            const double rlow,
            const double lx,
            const double ly,
            const double lz)
```

Constructor from centre, upper/lower radii, half-lengths of the box.

SolidHole() [2/2]

```
Garfield::SolidHole::SolidHole (
            const double cx,
            const double cy,
            const double cz,
            const double rup,
            const double rlow,
            const double lx,
            const double ly,
            const double lz,
            const double dx,
            const double dy,
            const double dz)
```

Constructor from centre, upper/lower radii, half-lengths of the box and orientation.

~SolidHole()

```
Garfield::SolidHole::~SolidHole ()  [inline]
```

Destructor.
Definition at line 25 of file SolidHole.hh.

```
00025 {}
```

### 7.99.3 Member Function Documentation

Cut()

```
void Garfield::SolidHole::Cut (
            const double x0,
            const double y0,
            const double z0,
            const double xn,
            const double yn,
            const double zn,
            std::vector< Panel > & panels)  [override], [virtual]
```

Implements Garfield::Solid.

GetAverage()

bool Garfield::SolidHole::GetAverage () const [inline]
Return the state of the "average-radius" flag.
Definition at line 68 of file SolidHole.hh.
00068 { return m_average; }

GetBoundingBox()

bool Garfield::SolidHole::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) const [override], [virtual]
Return the bounding box of the solid.
Implements Garfield::Solid.

GetDiscretisationLevel()

double Garfield::SolidHole::GetDiscretisationLevel (
const Panel & panel) [override], [virtual]
Retrieve the discretisation level of a panel.
Implements Garfield::Solid.

GetHalfLengthX()

double Garfield::SolidHole::GetHalfLengthX () const [inline], [override], [virtual]
Return the half-length along x.
Reimplemented from Garfield::Solid.
Definition at line 44 of file SolidHole.hh.
00044 { return m_lX; }

GetHalfLengthY()

double Garfield::SolidHole::GetHalfLengthY () const [inline], [override], [virtual]
Return the half-length along y.
Reimplemented from Garfield::Solid.
Definition at line 45 of file SolidHole.hh.
00045 { return m_lY; }

GetHalfLengthZ()

double Garfield::SolidHole::GetHalfLengthZ () const [inline], [override], [virtual]
Return the half-length along z.
Reimplemented from Garfield::Solid.
Definition at line 46 of file SolidHole.hh.
00046 { return m_lZ; }

GetLowerRadius()

double Garfield::SolidHole::GetLowerRadius () const [inline], [override], [virtual]
Return the lower radius (of a hole).
Reimplemented from Garfield::Solid.
Definition at line 48 of file SolidHole.hh.
00048 { return m_rLow; }

GetSectors()

unsigned int Garfield::SolidHole::GetSectors () const [inline]

Return the order of the approximating polygon.
Definition at line 66 of file SolidHole.hh.

```
00066 { return m_n; }
```

### GetUpperRadius()

double Garfield::SolidHole::GetUpperRadius () const [inline], [override], [virtual]

Return the upper radius (of a hole).
Reimplemented from Garfield::Solid.
Definition at line 47 of file SolidHole.hh.

```
00047 { return m_rUp; }
```

### IsHole()

bool Garfield::SolidHole::IsHole () const [inline], [override], [virtual]

Return true if the solid is a hole.
Reimplemented from Garfield::Solid.
Definition at line 31 of file SolidHole.hh.

```
00031 { return true; }
```

### IsInside()

bool Garfield::SolidHole::IsInside (
const double x,
const double y,
const double z,
const bool tesselated) const [override], [virtual]

Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implements Garfield::Solid.

### SetAverageRadius()

void Garfield::SolidHole::SetAverageRadius (
const bool average) [inline]

By default, the polygon used for approximating the hole when calculating surface panels is inscribed in a circle of the specified radius.
If the "average-radius" flag is activated, then the radius will be interpreted as the mean radius of the polygon that approximates the cylinder.
Definition at line 60 of file SolidHole.hh.

```
00060                                       {
00061     m_average = average;
00062     Update();
00063   }
```

### SetDiscretisationLevel()

void Garfield::SolidHole::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]

Set the discretisation level (for all panels).
Implements Garfield::Solid.
Definition at line 71 of file SolidHole.hh.

```
00071 { m_dis.fill(dis); }
```

### SetHalfLengthX()

void Garfield::SolidHole::SetHalfLengthX (
const double lx)

Set the half-length of the box along x.

SetHalfLengthY()

void Garfield::SolidHole::SetHalfLengthY (
const double ly)
Set the half-length of the box along y.

SetHalfLengthZ()

void Garfield::SolidHole::SetHalfLengthZ (
const double lz)
Set the half-length of the box along z.

SetLowerRadius()

void Garfield::SolidHole::SetLowerRadius (
const double r)
Set the radius at z = -lz.

SetSectors()

void Garfield::SolidHole::SetSectors (
const unsigned int n)
When calculating the surface panels, the hole is approximated as a polygon with a finite number of panels. The number of corners of the polygon equals $4(n-1)$. Thus, $n = 2$ will produce a square, $n = 3$ an octagon etc.

SetUpperRadius()

void Garfield::SolidHole::SetUpperRadius (
const double r)
Set the radius at z = +lz.

SolidPanels()

bool Garfield::SolidHole::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

Update()

void Garfield::SolidHole::Update () [private]

### 7.99.4 Member Data Documentation

m_average

bool Garfield::SolidHole::m_average = false [private]
Average chord over the sectors.
Definition at line 96 of file SolidHole.hh.

m_dis

std::array<double, 7> Garfield::SolidHole::m_dis {{-1., -1., -1., -1., -1., -1., -1.}} [private]
Discretisation levels.
Definition at line 104 of file SolidHole.hh.
00104 {{-1., -1., -1., -1., -1., -1., -1.}};

m_fi

double Garfield::SolidHole::m_fi = 1. [private]
Ratio between inradius and exradius of the approximating polygon.
Definition at line 101 of file SolidHole.hh.

m_fp

double Garfield::SolidHole::m_fp = 1. [private]
Ratio between the approximating polygon's radius and the hole radius.
Definition at line 99 of file SolidHole.hh.

m_lX

double Garfield::SolidHole::m_lX [private]
Half-length in x.
Definition at line 87 of file SolidHole.hh.

m_lY

double Garfield::SolidHole::m_lY [private]
Half-length in y.
Definition at line 89 of file SolidHole.hh.

m_lZ

double Garfield::SolidHole::m_lZ [private]
Half-length in z.
Definition at line 91 of file SolidHole.hh.

m_mutex

std::mutex Garfield::SolidHole::m_mutex [private]
Mutex.
Definition at line 80 of file SolidHole.hh.

m_n

unsigned int Garfield::SolidHole::m_n = 2 [private]
Number of sectors.
Definition at line 94 of file SolidHole.hh.

m_rLow

double Garfield::SolidHole::m_rLow [private]
Lower radius.
Definition at line 85 of file SolidHole.hh.

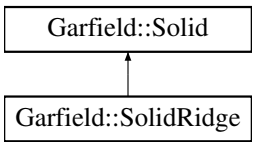
m_rUp

double Garfield::SolidHole::m_rUp [private]
Upper radius.
Definition at line 83 of file SolidHole.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidHole.hh

## 7.100 Garfield::SolidRidge Class Reference

Triangular prism (Toblerone bar).
#include <SolidRidge.hh>
Inheritance diagram for Garfield::SolidRidge:




Public Member Functions

- SolidRidge (const double cx, const double cy, const double cz, const double lx, const double ly, const double hz, const double offsetx)

  Constructor from centre, half-lengths, height and x-offset.
- SolidRidge (const double cx, const double cy, const double cz, const double lx, const double ly, const double hz, const double offsetx, const double dx, const double dy, const double dz)

  Constructor from centre, half-lengths, height, x-offset and orientation.
- ~SolidRidge ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.
- bool IsRidge () const override

  Return true if the solid is a ridge.
- void SetHalfLengthX (const double lx)

  Set the half-length along x.
- void SetHalfLengthY (const double ly)

  Set the half-length along y.
- void SetRidgeHeight (const double hz)

  Set the height of the ridge.
- void SetRidgeOffset (const double dx)

  Set the x-offset of the ridge.
- double GetHalfLengthX () const override

  Return the half-length along x.
- double GetHalfLengthY () const override

  Return the half-length along y.
- double GetRidgeHeight () const override

  Return the height of a ridge.
- double GetRidgeOffset () const override

  Return the x-offset of a ridge.
- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete
  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)
  Constructor.
- virtual ~Solid ()
  Destructor.
- virtual bool IsBox () const
  Return true if the solid is a box.
- virtual bool IsTube () const
  Return true if the solid is a tube.
- virtual bool IsSphere () const
  Return true if the solid is a sphere.
- virtual bool IsHole () const
  Return true if the solid is a hole.
- virtual bool IsExtrusion () const
  Return true if the solid is an extrusion.
- virtual bool IsWire () const
  Return true if the solid is a wire.
- void SetLabel (const std::string &label)
  Set a label.
- std::string GetLabel () const
  Return the label.
- bool GetCentre (double &x, double &y, double &z) const
  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const
  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const
  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetHalfLengthZ () const
  Return the half-length along z.
- virtual double GetInnerRadius () const
  Return the inner radius.
- virtual double GetOuterRadius () const
  Return the outer radius.
- virtual double GetRadius () const
  Return the radius.
- virtual double GetLowerRadius () const
  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const
  Return the upper radius (of a hole).
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const
  Get the vertices defining an extrusion.
- unsigned int GetId () const
  Get the ID of the solid.
- void SetBoundaryPotential (const double v)
  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)
  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()

Make the potential at the surface of the solid floating.

- void SetBoundaryDielectric ()

  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const

  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const

  Retrieve the potential.
- double GetBoundaryChargeDensity () const

  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off.
- void SetColour (const int col)

  Set the colour of the solid.
- int GetColour () const

  Get the colour of the solid.

Private Attributes

- double m_lX

  Half-length in x.
- double m_lY

  Half-length in y.
- double m_hz

  Height of the ridge.
- double m_hx

  Offset of the ridge in x.
- std::array< double, 5 > m_dis {{-1., -1., -1., -1., -1.}}

  Discretisation levels.

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.
  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.
  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.
  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.
  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"
  Class name.
- std::string m_label = ""
  Label.
- bool m_debug = false
  Debug flag.
- BoundaryCondition m_bctype = Unknown
  Type of boundary condition.
- double m_volt = 0.
  Potential at the surface.
- double m_charge = 0.
  Surface charge density.
- double m_eps = 0.
  Dielectric constant.
- int m_colour = -1
  Colour.

### 7.100.1 Detailed Description

Triangular prism (Toblerone bar).
Definition at line 12 of file SolidRidge.hh.

### 7.100.2 Constructor & Destructor Documentation

SolidRidge() [1/2]

```
Garfield::SolidRidge::SolidRidge (
            const double cx,
            const double cy,
            const double cz,
            const double lx,
            const double ly,
            const double hz,
            const double offsetx)
```

Constructor from centre, half-lengths, height and x-offset.

SolidRidge() [2/2]

Garfield::SolidRidge::SolidRidge (
const double cx,
const double cy,
const double cz,
const double lx,
const double ly,
const double hz,
const double offsetx,
const double dx,
const double dy,
const double dz)

Constructor from centre, half-lengths, height, x-offset and orientation.

~SolidRidge()

Garfield::SolidRidge::~SolidRidge () [inline]

Destructor.

Definition at line 22 of file SolidRidge.hh.

00022 {}

### 7.100.3 Member Function Documentation

Cut()

void Garfield::SolidRidge::Cut (
const double x0,
const double y0,
const double z0,
const double xn,
const double yn,
const double zn,
std::vector< Panel > & panels) [override], [virtual]

Implements Garfield::Solid.

GetBoundingBox()

bool Garfield::SolidRidge::GetBoundingBox (
double & xmin,
double & ymin,
double & zmin,
double & xmax,
double & ymax,
double & zmax) const [override], [virtual]

Return the bounding box of the solid.

Implements Garfield::Solid.

GetDiscretisationLevel()

double Garfield::SolidRidge::GetDiscretisationLevel (
const Panel & panel) [override], [virtual]

Retrieve the discretisation level of a panel.

Implements Garfield::Solid.

GetHalfLengthX()

double Garfield::SolidRidge::GetHalfLengthX () const [inline], [override], [virtual]

Return the half-length along x.

Reimplemented from Garfield::Solid.

Definition at line 39 of file SolidRidge.hh.

00039 { return m_lX; }

GetHalfLengthY()

double Garfield::SolidRidge::GetHalfLengthY () const [inline], [override], [virtual]
Return the half-length along y.
Reimplemented from Garfield::Solid.
Definition at line 40 of file SolidRidge.hh.
00040 { return m_lY; }

GetRidgeHeight()

double Garfield::SolidRidge::GetRidgeHeight () const [inline], [override], [virtual]
Return the height of a ridge.
Reimplemented from Garfield::Solid.
Definition at line 41 of file SolidRidge.hh.
00041 { return m_hz; }

GetRidgeOffset()

double Garfield::SolidRidge::GetRidgeOffset () const [inline], [override], [virtual]
Return the x-offset of a ridge.
Reimplemented from Garfield::Solid.
Definition at line 42 of file SolidRidge.hh.
00042 { return m_hx; }

IsInside()

bool Garfield::SolidRidge::IsInside (
const double x,
const double y,
const double z,
const bool tesselated) const [override], [virtual]
Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implements Garfield::Solid.

IsRidge()

bool Garfield::SolidRidge::IsRidge () const [inline], [override], [virtual]
Return true if the solid is a ridge.
Reimplemented from Garfield::Solid.
Definition at line 28 of file SolidRidge.hh.
00028 { return true; }

SetDiscretisationLevel()

void Garfield::SolidRidge::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]
Set the discretisation level (for all panels).
Implements Garfield::Solid.
Definition at line 45 of file SolidRidge.hh.
00045 { m_dis.fill(dis); }

SetHalfLengthX()

void Garfield::SolidRidge::SetHalfLengthX (
const double lx)
Set the half-length along x.

SetHalfLengthY()

void Garfield::SolidRidge::SetHalfLengthY (
const double ly)
Set the half-length along y.

SetRidgeHeight()

void Garfield::SolidRidge::SetRidgeHeight (
const double hz)
Set the height of the ridge.

SetRidgeOffset()

void Garfield::SolidRidge::SetRidgeOffset (
const double dx) [inline]
Set the x-offset of the ridge.
Definition at line 37 of file SolidRidge.hh.
00037 { m_hx = dx; }

SolidPanels()

bool Garfield::SolidRidge::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

### 7.100.4 Member Data Documentation

m_dis

std::array<double, 5> Garfield::SolidRidge::m_dis {{-1., -1., -1., -1., -1.}} [private]
Discretisation levels.
Definition at line 63 of file SolidRidge.hh.
00063 {{-1., -1., -1., -1., -1.}};

m_hx

double Garfield::SolidRidge::m_hx [private]
Offset of the ridge in x.
Definition at line 60 of file SolidRidge.hh.

m_hz

double Garfield::SolidRidge::m_hz [private]
Height of the ridge.
Definition at line 58 of file SolidRidge.hh.

m_lX

double Garfield::SolidRidge::m_lX [private]
Half-length in x.
Definition at line 54 of file SolidRidge.hh.

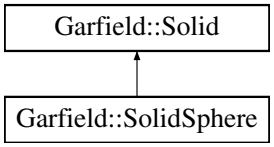
m_lY

double Garfield::SolidRidge::m_lY [private]
Half-length in y.
Definition at line 56 of file SolidRidge.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidRidge.hh

## 7.101 Garfield::SolidSphere Class Reference

Sphere.
#include <SolidSphere.hh>
Inheritance diagram for Garfield::SolidSphere:




Public Member Functions

- SolidSphere (const double cx, const double cy, const double cz, const double r)

  Constructor from centre and outer radius.
- SolidSphere (const double cx, const double cy, const double cz, const double rmin, const double rmax)

  Constructor from centre and inner/outer radii.
- ~SolidSphere ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.
- bool IsSphere () const override

  Return true if the solid is a sphere.
- void SetRadius (const double r)

  Set the radius of the sphere.
- void SetRadii (const double rmin, const double rmax)

  Set the inner and outer radius of the sphere.
- double GetRadius () const override

  Return the radius.
- double GetInnerRadius () const override

  Return the inner radius.
- double GetOuterRadius () const override

  Return the outer radius.
- void SetMeridians (const unsigned int n)

  When calculating surface panels, the sphere is approximated by a set of parallelograms, much the same way maps are drawn ("UV sphere").
- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete
  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)
  Constructor.
- virtual ~Solid ()
  Destructor.
- virtual bool IsBox () const
  Return true if the solid is a box.
- virtual bool IsTube () const
  Return true if the solid is a tube.
- virtual bool IsHole () const
  Return true if the solid is a hole.
- virtual bool IsRidge () const
  Return true if the solid is a ridge.
- virtual bool IsExtrusion () const
  Return true if the solid is an extrusion.
- virtual bool IsWire () const
  Return true if the solid is a wire.
- void SetLabel (const std::string &label)
  Set a label.
- std::string GetLabel () const
  Return the label.
- bool GetCentre (double &x, double &y, double &z) const
  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const
  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const
  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetHalfLengthX () const
  Return the half-length along x.
- virtual double GetHalfLengthY () const
  Return the half-length along y.
- virtual double GetHalfLengthZ () const
  Return the half-length along z.
- virtual double GetLowerRadius () const
  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const
  Return the upper radius (of a hole).
- virtual double GetRidgeOffset () const
  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const
  Return the height of a ridge.
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const
  Get the vertices defining an extrusion.
- unsigned int GetId () const
  Get the ID of the solid.
- void SetBoundaryPotential (const double v)
  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)

Apply fixed-charge boundary conditions.

- void SetBoundaryFloat ()

  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()

  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const

  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const

  Retrieve the potential.
- double GetBoundaryChargeDensity () const

  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)

  Switch debugging messages on/off.
- void SetColour (const int col)

  Set the colour of the solid.
- int GetColour () const

  Get the colour of the solid.

Private Member Functions

- void UpdatePanels ()
- void MakePanels (const int vol, const double r, const bool out, std::vector< Panel > &panels) const

Private Attributes

- std::mutex m_mutex

  Mutex.
- double m_rMin = 0.

  Inner and outer radii.
- double m_rMax = 1.
- unsigned int m_n = 10

  Number of meridians.
- double m_dis = -1.

  Discretisation level.
- std::vector< Panel > m_panelsO

  Surface panels.
- std::vector< Panel > m_panelsI

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.

  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.

  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.

  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.

  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"

  Class name.
- std::string m_label = ""

  Label.
- bool m_debug = false

  Debug flag.
- BoundaryCondition m_bctype = Unknown

  Type of boundary condition.
- double m_volt = 0.

  Potential at the surface.
- double m_charge = 0.

  Surface charge density.
- double m_eps = 0.

  Dielectric constant.
- int m_colour = -1

  Colour.

### 7.101.1 Detailed Description

Sphere.
Definition at line 12 of file SolidSphere.hh.

### 7.101.2 Constructor & Destructor Documentation

SolidSphere() [1/2]

```
Garfield::SolidSphere::SolidSphere (
            const double cx,
            const double cy,
            const double cz,
            const double r)
```

Constructor from centre and outer radius.

SolidSphere() [2/2]

```
Garfield::SolidSphere::SolidSphere (
            const double cx,
            const double cy,
            const double cz,
            const double rmin,
            const double rmax)
```

Constructor from centre and inner/outer radii.

~SolidSphere()

```
Garfield::SolidSphere::~SolidSphere ()  [inline]
```

Destructor.
Definition at line 21 of file SolidSphere.hh.

```
00021 {}
```

### 7.101.3 Member Function Documentation

Cut()

```
void Garfield::SolidSphere::Cut (
            const double x0,
            const double y0,
            const double z0,
            const double xn,
            const double yn,
            const double zn,
            std::vector< Panel > & panels)  [override], [virtual]
```

Implements Garfield::Solid.

GetBoundingBox()

```
bool Garfield::SolidSphere::GetBoundingBox (
            double & xmin,
            double & ymin,
            double & zmin,
            double & xmax,
            double & ymax,
            double & zmax) const  [override], [virtual]
```

Return the bounding box of the solid.
Implements Garfield::Solid.

GetDiscretisationLevel()

```
double Garfield::SolidSphere::GetDiscretisationLevel (
            const Panel & panel)  [override], [virtual]
```

Retrieve the discretisation level of a panel.
Implements Garfield::Solid.

### GetInnerRadius()

double Garfield::SolidSphere::GetInnerRadius () const [inline], [override], [virtual]
Return the inner radius.
Reimplemented from Garfield::Solid.
Definition at line 35 of file SolidSphere.hh.

```
00035 { return m_rMin; }
```

### GetOuterRadius()

double Garfield::SolidSphere::GetOuterRadius () const [inline], [override], [virtual]
Return the outer radius.
Reimplemented from Garfield::Solid.
Definition at line 36 of file SolidSphere.hh.

```
00036 { return m_rMax; }
```

### GetRadius()

double Garfield::SolidSphere::GetRadius () const [inline], [override], [virtual]
Return the radius.
Reimplemented from Garfield::Solid.
Definition at line 34 of file SolidSphere.hh.

```
00034 { return m_rMax; }
```

### IsInside()

bool Garfield::SolidSphere::IsInside (
const double x,
const double y,
const double z,
const bool tesselated) const [override], [virtual]
Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implements Garfield::Solid.

### IsSphere()

bool Garfield::SolidSphere::IsSphere () const [inline], [override], [virtual]
Return true if the solid is a sphere.
Reimplemented from Garfield::Solid.
Definition at line 27 of file SolidSphere.hh.

```
00027 { return true; }
```

### MakePanels()

void Garfield::SolidSphere::MakePanels (
const int vol,
const double r,
const bool out,
std::vector< Panel > & panels) const [private]

### SetDiscretisationLevel()

void Garfield::SolidSphere::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]
Set the discretisation level (for all panels).
Implements Garfield::Solid.
Definition at line 44 of file SolidSphere.hh.

```
00044 { m_dis = dis; }
```

SetMeridians()

void Garfield::SolidSphere::SetMeridians (
const unsigned int n)
When calculating surface panels, the sphere is approximated by a set of parallelograms, much the same way maps are drawn ("UV sphere").
N specifies the number of meridians and also the number of parallels.

SetRadii()

void Garfield::SolidSphere::SetRadii (
const double rmin,
const double rmax)
Set the inner and outer radius of the sphere.

SetRadius()

void Garfield::SolidSphere::SetRadius (
const double r)
Set the radius of the sphere.

SolidPanels()

bool Garfield::SolidSphere::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

UpdatePanels()

void Garfield::SolidSphere::UpdatePanels () [private]

### 7.101.4 Member Data Documentation

m_dis

double Garfield::SolidSphere::m_dis = -1. [private]
Discretisation level.
Definition at line 62 of file SolidSphere.hh.

m_mutex

std::mutex Garfield::SolidSphere::m_mutex [private]
Mutex.
Definition at line 53 of file SolidSphere.hh.

m_n

unsigned int Garfield::SolidSphere::m_n = 10 [private]
Number of meridians.
Definition at line 59 of file SolidSphere.hh.

m_panelsI

std::vector<Panel> Garfield::SolidSphere::m_panelsI [private]
Definition at line 66 of file SolidSphere.hh.

m_panelsO

std::vector<Panel> Garfield::SolidSphere::m_panelsO [private]
Surface panels.
Definition at line 65 of file SolidSphere.hh.

m_rMax

double Garfield::SolidSphere::m_rMax = 1. [private]
Definition at line 56 of file SolidSphere.hh.

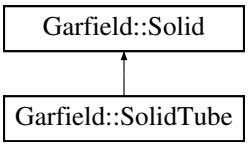
m_rMin

double Garfield::SolidSphere::m_rMin = 0. [private]
Inner and outer radii.
Definition at line 56 of file SolidSphere.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidSphere.hh

## 7.102 Garfield::SolidTube Class Reference

Cylindrical tube.
#include <SolidTube.hh>
Inheritance diagram for Garfield::SolidTube:




Public Member Functions

- SolidTube (const double cx, const double cy, const double cz, const double r, const double lz)

  Constructor from centre, outer radius, and half-length.
- SolidTube (const double cx, const double cy, const double cz, const double r, const double lz, const double dx, const double dy, const double dz)

  Constructor from centre, outer radius, half-length and orientation.
- SolidTube (const double cx, const double cy, const double cz, const double ri, const double ro, const double lz)

  Constructor from centre, inner radius, outer radius, and half-length.
- SolidTube (const double cx, const double cy, const double cz, const double ri, const double ro, const double lz, const double dx, const double dy, const double dz)

  Constructor from centre, inner radius, outer radius, half-length and orientation.
- ~SolidTube ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.
- bool IsTube () const override

  Return true if the solid is a tube.
- void SetHalfLength (const double lz)

  Set the half-length of the tube.
- void SetRadius (const double r)

  Set the outer radius.
- double GetHalfLengthZ () const override

  Retrieve the half-length of the tube.
- double GetRadius () const override

Retrieve the outer radius.

- double GetOuterRadius () const override

  Retrieve the outer radius.
- double GetInnerRadius () const override

  Retrieve the inner radius.
- void SetSectors (const unsigned int n)

  When calculating the surface panels, the cylinder is approximated as a polygon with a finite number of panels.
- void SetRotation (const double angle)

  Specify a rotation angle (radian) of the cylinder.
- void SetAverageRadius (const bool average)

  By default, the polygon used for approximating the cylinder when calculating surface panels is inscribed in a circle of the specified radius.
- void SetTopLid (const bool closed)

  Request the cylinder to be closed with a (polygonal) lid at +z.
- void SetBottomLid (const bool closed)

  Request the cylinder to be closed with a (polygonal) lid at -z.
- unsigned int GetSectors () const

  Return the number of sectors.
- double GetRotation () const

  Return the current rotation angle.
- bool GetAverage () const

  Return the status of the "average-radius" flag.
- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete

  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)

  Constructor.
- virtual ~Solid ()

  Destructor.
- virtual bool IsBox () const

  Return true if the solid is a box.
- virtual bool IsSphere () const

  Return true if the solid is a sphere.
- virtual bool IsHole () const

  Return true if the solid is a hole.
- virtual bool IsRidge () const

  Return true if the solid is a ridge.
- virtual bool IsExtrusion () const

  Return true if the solid is an extrusion.
- virtual bool IsWire () const

  Return true if the solid is a wire.

- void SetLabel (const std::string &label)
  Set a label.
- std::string GetLabel () const
  Return the label.
- bool GetCentre (double &x, double &y, double &z) const
  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const
  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const
  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetHalfLengthX () const
  Return the half-length along x.
- virtual double GetHalfLengthY () const
  Return the half-length along y.
- virtual double GetLowerRadius () const
  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const
  Return the upper radius (of a hole).
- virtual double GetRidgeOffset () const
  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const
  Return the height of a ridge.
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const
  Get the vertices defining an extrusion.
- unsigned int GetId () const
  Get the ID of the solid.
- void SetBoundaryPotential (const double v)
  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)
  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()
  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()
  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const
  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const
  Retrieve the potential.
- double GetBoundaryChargeDensity () const
  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)
  Switch debugging messages on/off.
- void SetColour (const int col)
  Set the colour of the solid.
- int GetColour () const
  Get the colour of the solid.

Private Member Functions

- void UpdatePolygon ()

Private Attributes

- std::mutex m_mutex
  Mutex.
- double m_rO
  Outer radius.
- double m_rI = 0.
  Inner radius.
- double m_lZ
  Half-length.
- double m_rot = 0.
  Rotation angle.
- unsigned int m_n = 2
  Number of sectors.
- bool m_average = false
  Average chord over the sectors.
- double m_rpO
  Radius of the outer approximating polygon.
- double m_rpI = 0.
  Radius of the inner approximating polygon.
- double m_riO
  Inradius of the outer approximating polygon.
- double m_riI = 0.
  Inradius of the inner approximating polygon.
- std::vector< double > m_xpO
  X-coordinates of the outer approximating polygon.
- std::vector< double > m_ypO
  Y-coordinates of the outer approximating polygon.
- std::vector< double > m_xpI
  X-coordinates of the inner approximating polygon.
- std::vector< double > m_ypI
  Y-coordinates of the inner approximating polygon.
- bool m_toplid = true
  Have a top lid?
- bool m_botlid = true
  Have a bottom lid?
- std::array< double, 3 > m_dis {{-1., -1., -1.}}
  Discretisation levels.

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.

  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.

  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.

  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.

  Polar angle.
- double m_sTheta = 0.
- std::string m_className = "Solid"

  Class name.
- std::string m_label = ""

  Label.
- bool m_debug = false

  Debug flag.
- BoundaryCondition m_bctype = Unknown

  Type of boundary condition.
- double m_volt = 0.

  Potential at the surface.
- double m_charge = 0.

  Surface charge density.
- double m_eps = 0.

  Dielectric constant.
- int m_colour = -1

  Colour.

### 7.102.1 Detailed Description

Cylindrical tube.
Definition at line 13 of file SolidTube.hh.

### 7.102.2 Constructor & Destructor Documentation

SolidTube() [1/4]

```
Garfield::SolidTube::SolidTube (
            const double cx,
            const double cy,
            const double cz,
            const double r,
            const double lz)
```

Constructor from centre, outer radius, and half-length.

SolidTube() [2/4]

```
Garfield::SolidTube::SolidTube (
            const double cx,
            const double cy,
            const double cz,
            const double r,
            const double lz,
            const double dx,
            const double dy,
            const double dz)
```

Constructor from centre, outer radius, half-length and orientation.

SolidTube() [3/4]

```
Garfield::SolidTube::SolidTube (
            const double cx,
            const double cy,
            const double cz,
            const double ri,
            const double ro,
            const double lz)
```

Constructor from centre, inner radius, outer radius, and half-length.

SolidTube() [4/4]

```
Garfield::SolidTube::SolidTube (
            const double cx,
            const double cy,
            const double cz,
            const double ri,
            const double ro,
            const double lz,
            const double dx,
            const double dy,
            const double dz)
```

Constructor from centre, inner radius, outer radius, half-length and orientation.

~SolidTube()

```
Garfield::SolidTube::~SolidTube ()  [inline]
```

Destructor.

Definition at line 30 of file SolidTube.hh.

```
00030 {}
```

### 7.102.3 Member Function Documentation

#### Cut()

```
void Garfield::SolidTube::Cut (
                const double x0,
                const double y0,
                const double z0,
                const double xn,
                const double yn,
                const double zn,
                std::vector< Panel > & panels)   [override], [virtual]
```

Implements Garfield::Solid.

#### GetAverage()

```
bool Garfield::SolidTube::GetAverage () const   [inline]
```

Return the status of the "average-radius" flag.
Definition at line 76 of file SolidTube.hh.

```
00076 { return m_average; }
```

#### GetBoundingBox()

```
bool Garfield::SolidTube::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax) const   [override], [virtual]
```

Return the bounding box of the solid.
Implements Garfield::Solid.

#### GetDiscretisationLevel()

```
double Garfield::SolidTube::GetDiscretisationLevel (
                const Panel & panel)   [override], [virtual]
```

Retrieve the discretisation level of a panel.
Implements Garfield::Solid.

#### GetHalfLengthZ()

```
double Garfield::SolidTube::GetHalfLengthZ () const   [inline], [override], [virtual]
```

Retrieve the half-length of the tube.
Reimplemented from Garfield::Solid.
Definition at line 43 of file SolidTube.hh.

```
00043 { return m_lZ; }
```

#### GetInnerRadius()

```
double Garfield::SolidTube::GetInnerRadius () const   [inline], [override], [virtual]
```

Retrieve the inner radius.
Reimplemented from Garfield::Solid.
Definition at line 49 of file SolidTube.hh.

```
00049 { return m_rI; }
```

#### GetOuterRadius()

```
double Garfield::SolidTube::GetOuterRadius () const   [inline], [override], [virtual]
```

Retrieve the outer radius.
Reimplemented from Garfield::Solid.

Definition at line 47 of file SolidTube.hh.
00047 { return m_rO; }

GetRadius()

double Garfield::SolidTube::GetRadius () const [inline], [override], [virtual]
Retrieve the outer radius.
Reimplemented from Garfield::Solid.
Definition at line 45 of file SolidTube.hh.
00045 { return m_rO; }

GetRotation()

double Garfield::SolidTube::GetRotation () const [inline]
Return the current rotation angle.
Definition at line 74 of file SolidTube.hh.
00074 { return m_rot; }

GetSectors()

unsigned int Garfield::SolidTube::GetSectors () const [inline]
Return the number of sectors.
Definition at line 72 of file SolidTube.hh.
00072 { return m_n; }

IsInside()

bool Garfield::SolidTube::IsInside (
const double x,
const double y,
const double z,
const bool tesselated) const [override], [virtual]
Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implements Garfield::Solid.

IsTube()

bool Garfield::SolidTube::IsTube () const [inline], [override], [virtual]
Return true if the solid is a tube.
Reimplemented from Garfield::Solid.
Definition at line 36 of file SolidTube.hh.
00036 { return true; }

SetAverageRadius()

void Garfield::SolidTube::SetAverageRadius (
const bool average) [inline]
By default, the polygon used for approximating the cylinder when calculating surface panels is inscribed in a circle of the specified radius.
If the "average-radius" flag is activated, then the radius will be interpreted as the mean radius of the polygon that approximates the cylinder.
Definition at line 65 of file SolidTube.hh.
00065 { m_average = average; }

SetBottomLid()

void Garfield::SolidTube::SetBottomLid (
const bool closed) [inline]
Request the cylinder to be closed with a (polygonal) lid at -z.

Definition at line 69 of file SolidTube.hh.
00069 { m_botlid = closed; }

SetDiscretisationLevel()

void Garfield::SolidTube::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]
Set the discretisation level (for all panels).
Implements Garfield::Solid.
Definition at line 79 of file SolidTube.hh.
00079 { m_dis.fill(dis); }

SetHalfLength()

void Garfield::SolidTube::SetHalfLength (
const double lz)
Set the half-length of the tube.

SetRadius()

void Garfield::SolidTube::SetRadius (
const double r)
Set the outer radius.

SetRotation()

void Garfield::SolidTube::SetRotation (
const double angle) [inline]
Specify a rotation angle (radian) of the cylinder.
Such a rotation is meaningful only if the number of sectors (when approximating the circle with a polygon) has been chosen small.
Definition at line 59 of file SolidTube.hh.
00059 { m_rot = angle; }

SetSectors()

void Garfield::SolidTube::SetSectors (
const unsigned int n)
When calculating the surface panels, the cylinder is approximated as a polygon with a finite number of panels.
The number of corners of the polygon equals $4(n-1)$. Thus, $n = 2$ will produce a square, $n = 3$ an octagon etc.

SetTopLid()

void Garfield::SolidTube::SetTopLid (
const bool closed) [inline]
Request the cylinder to be closed with a (polygonal) lid at +z.
Definition at line 67 of file SolidTube.hh.
00067 { m_toplid = closed; }

SolidPanels()

bool Garfield::SolidTube::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

UpdatePolygon()

void Garfield::SolidTube::UpdatePolygon () [private]

### 7.102.4 Member Data Documentation

#### m_average

bool Garfield::SolidTube::m_average = false [private]
Average chord over the sectors.
Definition at line 102 of file SolidTube.hh.

#### m_botlid

bool Garfield::SolidTube::m_botlid = true [private]
Have a bottom lid?
Definition at line 123 of file SolidTube.hh.

#### m_dis

std::array<double, 3> Garfield::SolidTube::m_dis {{-1., -1., -1.}} [private]
Discretisation levels.
Definition at line 126 of file SolidTube.hh.
00126 {{-1., -1., -1.}};

#### m_lZ

double Garfield::SolidTube::m_lZ [private]
Half-length.
Definition at line 95 of file SolidTube.hh.

#### m_mutex

std::mutex Garfield::SolidTube::m_mutex [private]
Mutex.
Definition at line 88 of file SolidTube.hh.

#### m_n

unsigned int Garfield::SolidTube::m_n = 2 [private]
Number of sectors.
Definition at line 100 of file SolidTube.hh.

#### m_rI

double Garfield::SolidTube::m_rI = 0. [private]
Inner radius.
Definition at line 93 of file SolidTube.hh.

#### m_riI

double Garfield::SolidTube::m_riI = 0. [private]
Inradius of the inner approximating polygon.
Definition at line 110 of file SolidTube.hh.

#### m_riO

double Garfield::SolidTube::m_riO [private]
Inradius of the outer approximating polygon.
Definition at line 108 of file SolidTube.hh.

#### m_rO

double Garfield::SolidTube::m_rO [private]
Outer radius.
Definition at line 91 of file SolidTube.hh.

m_rot

double Garfield::SolidTube::m_rot = 0. [private]
Rotation angle.
Definition at line 98 of file SolidTube.hh.

m_rpI

double Garfield::SolidTube::m_rpI = 0. [private]
Radius of the inner approximating polygon.
Definition at line 106 of file SolidTube.hh.

m_rpO

double Garfield::SolidTube::m_rpO [private]
Radius of the outer approximating polygon.
Definition at line 104 of file SolidTube.hh.

m_toplid

bool Garfield::SolidTube::m_toplid = true [private]
Have a top lid?
Definition at line 121 of file SolidTube.hh.

m_xpI

std::vector<double> Garfield::SolidTube::m_xpI [private]
X-coordinates of the inner approximating polygon.
Definition at line 116 of file SolidTube.hh.

m_xpO

std::vector<double> Garfield::SolidTube::m_xpO [private]
X-coordinates of the outer approximating polygon.
Definition at line 112 of file SolidTube.hh.

m_ypI

std::vector<double> Garfield::SolidTube::m_ypI [private]
Y-coordinates of the inner approximating polygon.
Definition at line 118 of file SolidTube.hh.

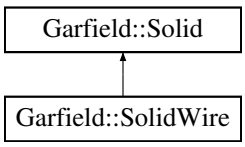
m_ypO

std::vector<double> Garfield::SolidTube::m_ypO [private]
Y-coordinates of the outer approximating polygon.
Definition at line 114 of file SolidTube.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidTube.hh

## 7.103 Garfield::SolidWire Class Reference

Wire.
#include <SolidWire.hh>
Inheritance diagram for Garfield::SolidWire:




Public Member Functions

- SolidWire (const double cx, const double cy, const double cz, const double r, const double lz)

  Constructor from centre, radius, and half-length.
- SolidWire (const double cx, const double cy, const double cz, const double r, const double lz, const double dx, const double dy, const double dz)

  Constructor from centre, radius, half-length and orientation.
- ~SolidWire ()

  Destructor.
- bool IsInside (const double x, const double y, const double z, const bool tesselated) const override

  Check whether a given point is inside the solid.
- bool GetBoundingBox (double &xmin, double &ymin, double &zmin, double &xmax, double &ymax, double &zmax) const override

  Return the bounding box of the solid.
- bool IsWire () const override

  Return true if the solid is a wire.
- void SetHalfLength (const double lz)
- void SetRadius (const double r)
- double GetHalfLengthZ () const override

  Return the half-length along z.
- double GetRadius () const override

  Return the radius.
- bool SolidPanels (std::vector< Panel > &panels) override

  Retrieve the surface panels of the solid.
- void SetDiscretisationLevel (const double dis) override

  Set the discretisation level (for all panels).
- double GetDiscretisationLevel (const Panel &panel) override

  Retrieve the discretisation level of a panel.
- void Cut (const double x0, const double y0, const double z0, const double xn, const double yn, const double zn, std::vector< Panel > &panels) override

Public Member Functions inherited from Garfield::Solid

- Solid ()=delete

  Default constructor.
- Solid (const double cx, const double cy, const double cz, const std::string &name)

  Constructor.
- virtual ~Solid ()

  Destructor.
- virtual bool IsBox () const

  Return true if the solid is a box.
- virtual bool IsTube () const

  Return true if the solid is a tube.
- virtual bool IsSphere () const

  Return true if the solid is a sphere.
- virtual bool IsHole () const

  Return true if the solid is a hole.

- virtual bool IsRidge () const

  Return true if the solid is a ridge.
- virtual bool IsExtrusion () const

  Return true if the solid is an extrusion.
- void SetLabel (const std::string &label)

  Set a label.
- std::string GetLabel () const

  Return the label.
- bool GetCentre (double &x, double &y, double &z) const

  Retrieve the centre point of the solid.
- bool GetDirection (double &dx, double &dy, double &dz) const

  Retrieve the direction vector.
- bool GetOrientation (double &ctheta, double &stheta, double &cphi, double &sphi) const

  Retrieve the orientation (azimuthal and polar angles) of the solid.
- virtual double GetHalfLengthX () const

  Return the half-length along x.
- virtual double GetHalfLengthY () const

  Return the half-length along y.
- virtual double GetInnerRadius () const

  Return the inner radius.
- virtual double GetOuterRadius () const

  Return the outer radius.
- virtual double GetLowerRadius () const

  Return the lower radius (of a hole).
- virtual double GetUpperRadius () const

  Return the upper radius (of a hole).
- virtual double GetRidgeOffset () const

  Return the x-offset of a ridge.
- virtual double GetRidgeHeight () const

  Return the height of a ridge.
- virtual bool GetProfile (std::vector< double > &xv, std::vector< double > &yv) const

  Get the vertices defining an extrusion.
- unsigned int GetId () const

  Get the ID of the solid.
- void SetBoundaryPotential (const double v)

  Apply Dirichlet boundary conditions (fixed voltage).
- void SetBoundaryChargeDensity (const double q)

  Apply fixed-charge boundary conditions.
- void SetBoundaryFloat ()

  Make the potential at the surface of the solid floating.
- void SetBoundaryDielectric ()

  Make the surfaces of the solid dielectric-dielectric interfaces.
- void SetBoundaryParallelField ()
- void SetBoundaryPerpendicularField ()
- BoundaryCondition GetBoundaryConditionType () const

  Retrieve the type of boundary condition.
- double GetBoundaryPotential () const

  Retrieve the potential.
- double GetBoundaryChargeDensity () const

  Retrieve the surface charge density.
- void EnableDebugging (const bool on=true)

Switch debugging messages on/off.

- void SetColour (const int col)

  Set the colour of the solid.
- int GetColour () const

  Get the colour of the solid.

Private Attributes

- double m_r

  Radius.
- double m_lZ

  Half-length.
- double m_dis = -1.

  Discretisation level.

Additional Inherited Members

Public Types inherited from Garfield::Solid

- enum BoundaryCondition {
  Unknown = 0 , Voltage , Charge , Float ,
  Dielectric , DielectricCharge , ParallelField , PerpendicularField }

Static Public Member Functions inherited from Garfield::Solid

- static bool Intersect (const double x1, const double y1, const double z1, const double x2, const double y2, const double z2, const double x0, const double y0, const double z0, const double a, const double b, const double c, double &xc, double &yc, double &zc)

Protected Member Functions inherited from Garfield::Solid

- void ToLocal (const double x, const double y, const double z, double &u, double &v, double &w) const

  Transform a point from global coordinates (x, y, z) to local coordinates (u, v, w).
- void ToGlobal (const double u, const double v, const double w, double &x, double &y, double &z) const

  Transform a point from local coordinates (u, v, w) to global coordinates (x, y, z).
- void VectorToLocal (const double x, const double y, const double z, double &u, double &v, double &w)

  Transform a vector from global to local coordinates.
- void SetDirection (const double dx, const double dy, const double dz)

Protected Attributes inherited from Garfield::Solid

- double m_cX = 0.

  Centre of the solid.
- double m_cY = 0.
- double m_cZ = 0.
- double m_dX = 0.

  Direction vector.
- double m_dY = 0.
- double m_dZ = 1.
- double m_cPhi = 1.

  Azimuthal angle.
- double m_sPhi = 0.
- double m_cTheta = 1.

Polar angle.

- double m_sTheta = 0.
- std::string m_className = "Solid"

  Class name.
- std::string m_label = ""

  Label.
- bool m_debug = false

  Debug flag.
- BoundaryCondition m_bctype = Unknown

  Type of boundary condition.
- double m_volt = 0.

  Potential at the surface.
- double m_charge = 0.

  Surface charge density.
- double m_eps = 0.

  Dielectric constant.
- int m_colour = -1

  Colour.

### 7.103.1 Detailed Description

Wire.
Definition at line 10 of file SolidWire.hh.

### 7.103.2 Constructor & Destructor Documentation

SolidWire() [1/2]

```
Garfield::SolidWire::SolidWire (
            const double cx,
            const double cy,
            const double cz,
            const double r,
            const double lz)
```

Constructor from centre, radius, and half-length.

SolidWire() [2/2]

```
Garfield::SolidWire::SolidWire (
            const double cx,
            const double cy,
            const double cz,
            const double r,
            const double lz,
            const double dx,
            const double dy,
            const double dz)
```

Constructor from centre, radius, half-length and orientation.

~SolidWire()

```
Garfield::SolidWire::~SolidWire ()  [inline]
```

Destructor.
Definition at line 19 of file SolidWire.hh.

```
00019 {}
```

### 7.103.3 Member Function Documentation

#### Cut()

```
void Garfield::SolidWire::Cut (
                const double x0,
                const double y0,
                const double z0,
                const double xn,
                const double yn,
                const double zn,
                std::vector< Panel > & panels)   [override], [virtual]
```

Implements Garfield::Solid.

#### GetBoundingBox()

```
bool Garfield::SolidWire::GetBoundingBox (
                double & xmin,
                double & ymin,
                double & zmin,
                double & xmax,
                double & ymax,
                double & zmax) const   [override], [virtual]
```

Return the bounding box of the solid.
Implements Garfield::Solid.

#### GetDiscretisationLevel()

```
double Garfield::SolidWire::GetDiscretisationLevel (
                const Panel & panel)   [override], [virtual]
```

Retrieve the discretisation level of a panel.
Implements Garfield::Solid.

#### GetHalfLengthZ()

```
double Garfield::SolidWire::GetHalfLengthZ () const   [inline], [override], [virtual]
```

Return the half-length along z.
Reimplemented from Garfield::Solid.
Definition at line 30 of file SolidWire.hh.

```
00030 { return m_lZ; }
```

#### GetRadius()

```
double Garfield::SolidWire::GetRadius () const   [inline], [override], [virtual]
```

Return the radius.
Reimplemented from Garfield::Solid.
Definition at line 31 of file SolidWire.hh.

```
00031 { return m_r; }
```

#### IsInside()

```
bool Garfield::SolidWire::IsInside (
                const double x,
                const double y,
                const double z,
                const bool tesselated) const   [override], [virtual]
```

Check whether a given point is inside the solid.
If requested, use the tesselated approximation of the solid (if applicable).
Implements Garfield::Solid.

IsWire()

bool Garfield::SolidWire::IsWire () const [inline], [override], [virtual]
Return true if the solid is a wire.
Reimplemented from Garfield::Solid.
Definition at line 25 of file SolidWire.hh.
00025 { return true; }

SetDiscretisationLevel()

void Garfield::SolidWire::SetDiscretisationLevel (
const double dis) [inline], [override], [virtual]
Set the discretisation level (for all panels).
Implements Garfield::Solid.
Definition at line 34 of file SolidWire.hh.
00034 { m_dis = dis; }

SetHalfLength()

void Garfield::SolidWire::SetHalfLength (
const double lz)

SetRadius()

void Garfield::SolidWire::SetRadius (
const double r)

SolidPanels()

bool Garfield::SolidWire::SolidPanels (
std::vector< Panel > & panels) [override], [virtual]
Retrieve the surface panels of the solid.
Implements Garfield::Solid.

### 7.103.4 Member Data Documentation

m_dis

double Garfield::SolidWire::m_dis = -1. [private]
Discretisation level.
Definition at line 48 of file SolidWire.hh.

m_lZ

double Garfield::SolidWire::m_lZ [private]
Half-length.
Definition at line 45 of file SolidWire.hh.

m_r

double Garfield::SolidWire::m_r [private]
Radius.
Definition at line 43 of file SolidWire.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/SolidWire.hh

## 7.104 Garfield::ComponentNeBem2d::SpaceCharge Struct Reference

Public Attributes

- double x

- double y

  Coordinates of the centre.
- double a
- double b

  Half-lengths.
- double q

  Charge density.
- double v0

  Offset.

### 7.104.1 Detailed Description

Definition at line 181 of file ComponentNeBem2d.hh.

### 7.104.2 Member Data Documentation

a

double Garfield::ComponentNeBem2d::SpaceCharge::a
Definition at line 183 of file ComponentNeBem2d.hh.

b

double Garfield::ComponentNeBem2d::SpaceCharge::b
Half-lengths.
Definition at line 183 of file ComponentNeBem2d.hh.

q

double Garfield::ComponentNeBem2d::SpaceCharge::q
Charge density.
Definition at line 184 of file ComponentNeBem2d.hh.

v0

double Garfield::ComponentNeBem2d::SpaceCharge::v0
Offset.
Definition at line 185 of file ComponentNeBem2d.hh.

x

double Garfield::ComponentNeBem2d::SpaceCharge::x
Definition at line 182 of file ComponentNeBem2d.hh.

y

double Garfield::ComponentNeBem2d::SpaceCharge::y
Coordinates of the centre.
Definition at line 182 of file ComponentNeBem2d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh

## 7.105 Garfield::AvalancheMicroscopic::Statistics Struct Reference

#include <AvalancheMicroscopic.hh>

Public Attributes

- std::vector< double > gpu_stack_process_time
- std::vector< double > cpu_stack_process_time
- std::vector< double > gpu_stack_transport_time
- std::vector< double > cpu_stack_transport_time
- std::vector< size_t > stack_old_size
- std::vector< size_t > stack_new_size

### 7.105.1 Detailed Description

Definition at line 290 of file AvalancheMicroscopic.hh.

### 7.105.2 Member Data Documentation

cpu_stack_process_time

std::vector<double> Garfield::AvalancheMicroscopic::Statistics::cpu_stack_process_time
Definition at line 292 of file AvalancheMicroscopic.hh.

cpu_stack_transport_time

std::vector<double> Garfield::AvalancheMicroscopic::Statistics::cpu_stack_transport_time
Definition at line 294 of file AvalancheMicroscopic.hh.

gpu_stack_process_time

std::vector<double> Garfield::AvalancheMicroscopic::Statistics::gpu_stack_process_time
Definition at line 291 of file AvalancheMicroscopic.hh.

gpu_stack_transport_time

std::vector<double> Garfield::AvalancheMicroscopic::Statistics::gpu_stack_transport_time
Definition at line 293 of file AvalancheMicroscopic.hh.

stack_new_size

std::vector<size_t> Garfield::AvalancheMicroscopic::Statistics::stack_new_size
Definition at line 297 of file AvalancheMicroscopic.hh.

stack_old_size

std::vector<size_t> Garfield::AvalancheMicroscopic::Statistics::stack_old_size
Definition at line 296 of file AvalancheMicroscopic.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh

## 7.106 Garfield::ComponentAnalyticField::Strip Struct Reference

Public Attributes

- std::string type
  Label.
- int ind
  Readout group.
- double smin
- double smax
  Coordinates.
- double gap
  Distance to the opposite electrode.

### 7.106.1 Detailed Description

Definition at line 474 of file ComponentAnalyticField.hh.

### 7.106.2 Member Data Documentation

gap

double Garfield::ComponentAnalyticField::Strip::gap
Distance to the opposite electrode.
Definition at line 478 of file ComponentAnalyticField.hh.

ind

int Garfield::ComponentAnalyticField::Strip::ind
Readout group.
Definition at line 476 of file ComponentAnalyticField.hh.

smax

double Garfield::ComponentAnalyticField::Strip::smax
Coordinates.
Definition at line 477 of file ComponentAnalyticField.hh.

smin

double Garfield::ComponentAnalyticField::Strip::smin
Definition at line 477 of file ComponentAnalyticField.hh.

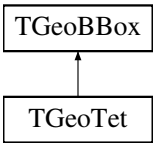
type

std::string Garfield::ComponentAnalyticField::Strip::type
Label.
Definition at line 475 of file ComponentAnalyticField.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh

## 7.107 TGeoTet Class Reference

#include <TGeoTet.hh>
Inheritance diagram for TGeoTet:




Public Member Functions

- TGeoTet ()
- TGeoTet (const char *name, const std::array< std::array< double, 3 >, 4 > &vertices)

  Constructor.
- virtual ~TGeoTet ()

  Destructor.
- void ComputeBBox () override
- int DistancetoPrimitive (int, int) override
- const TBuffer3D & GetBuffer3D (int reqSections, bool localFrame) const override
- void GetMeshNumbers (int &nvert, int &nsegs, int &npols) const override

- int GetNmeshVertices () const override
- void InspectShape () const override
- TBuffer3D * MakeBuffer3D () const override
- void Print (Option_t *option="") const override
- void SavePrimitive (std::ostream &, Option_t *) override
- void SetPoints (double *points) const override
- void SetPoints (float *points) const override
- void SetSegsAndPols (TBuffer3D &buff) const override
- void Sizeof3D () const override

Private Member Functions

- TGeoTet (const TGeoTet &)=delete
- TGeoTet & operator= (const TGeoTet &)=delete

Private Attributes

- std::array< std::array< double, 3 >, 4 > fVertices

### 7.107.1 Detailed Description

Definition at line 8 of file TGeoTet.hh.

### 7.107.2 Constructor & Destructor Documentation

TGeoTet() [1/3]

TGeoTet::TGeoTet () [inline]
Definition at line 10 of file TGeoTet.hh.
00010 {}

TGeoTet() [2/3]

TGeoTet::TGeoTet (
const char * name,
const std::array< std::array< double, 3 >, 4 > & vertices)
Constructor.

~TGeoTet()

virtual TGeoTet::~TGeoTet () [inline], [virtual]
Destructor.
Definition at line 15 of file TGeoTet.hh.
00015 {}

TGeoTet() [3/3]

TGeoTet::TGeoTet (
const TGeoTet & ) [private], [delete]

### 7.107.3 Member Function Documentation

ComputeBBox()

void TGeoTet::ComputeBBox () [override]

### DistancetoPrimitive()

```
int TGeoTet::DistancetoPrimitive (
                int ,
                int )   [inline], [override]
```

Definition at line 18 of file TGeoTet.hh.

```
00018 { return 99999; }
```

### GetBuffer3D()

```
const TBuffer3D & TGeoTet::GetBuffer3D (
                int reqSections,
                bool localFrame) const   [override]
```

### GetMeshNumbers()

```
void TGeoTet::GetMeshNumbers (
                int & nvert,
                int & nsegs,
                int & npols) const   [inline], [override]
```

Definition at line 20 of file TGeoTet.hh.

```
00020                                                   {
00021    nvert = 4;
00022    nsegs = 12;
00023    npols = 4;
00024  }
```

### GetNmeshVertices()

```
int TGeoTet::GetNmeshVertices () const   [inline], [override]
```

Definition at line 25 of file TGeoTet.hh.

```
00025 { return 4; }
```

### InspectShape()

```
void TGeoTet::InspectShape () const   [inline], [override]
```

Definition at line 26 of file TGeoTet.hh.

```
00026 {}
```

### MakeBuffer3D()

```
TBuffer3D * TGeoTet::MakeBuffer3D () const   [override]
```

### operator=()

```
TGeoTet & TGeoTet::operator= (
                const TGeoTet & )   [private], [delete]
```

### Print()

```
void TGeoTet::Print (
                Option_t * option = "") const   [override]
```

### SavePrimitive()

```
void TGeoTet::SavePrimitive (
                std::ostream & ,
                Option_t * )   [inline], [override]
```

Definition at line 29 of file TGeoTet.hh.

```
00029 {}
```

SetPoints() [1/2]

void TGeoTet::SetPoints (
double * points) const [override]

SetPoints() [2/2]

void TGeoTet::SetPoints (
float * points) const [override]

SetSegsAndPols()

void TGeoTet::SetSegsAndPols (
TBuffer3D & buff) const [override]

Sizeof3D()

void TGeoTet::Sizeof3D () const [inline], [override]
Definition at line 33 of file TGeoTet.hh.
00033 {}

### 7.107.4 Member Data Documentation

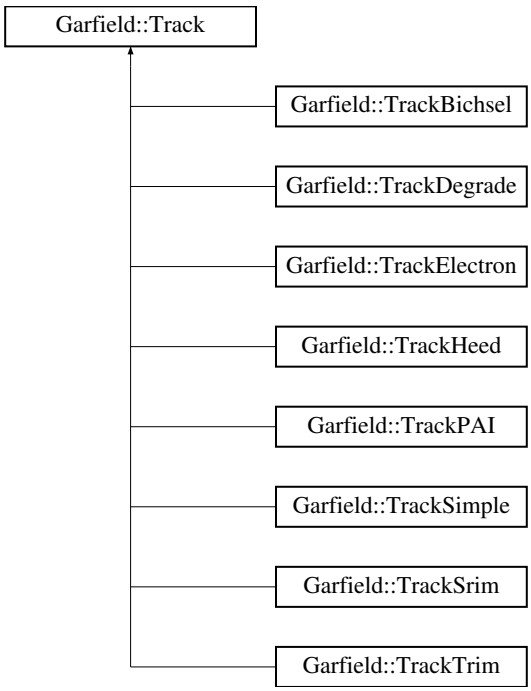
fVertices

std::array<std::array<double, 3>, 4> TGeoTet::fVertices [private]
Definition at line 38 of file TGeoTet.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TGeoTet.hh

## 7.108 Garfield::Track Class Reference

Abstract base class for track generation.
#include <Track.hh>
Inheritance diagram for Garfield::Track:



Public Member Functions

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- virtual void SetParticle (const std::string &part)

  Set the type of charged particle.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- virtual bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0)=0

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- virtual double GetClusterDensity ()

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- virtual double GetStoppingPower ()

  Get the stopping power (mean energy loss [eV] per cm).
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.

- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Protected Member Functions

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Static Protected Member Functions

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

Protected Attributes

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

### 7.108.1 Detailed Description

Abstract base class for track generation.
Definition at line 15 of file Track.hh.

### 7.108.2 Constructor & Destructor Documentation

Track() [1/2]

Garfield::Track::Track () [delete]
Default constructor.

Track() [2/2]

Garfield::Track::Track (
const std::string & name)
Constructor.

~Track()

virtual Garfield::Track::~Track () [inline], [virtual]
Destructor.
Definition at line 22 of file Track.hh.
00022 {}

### 7.108.3 Member Function Documentation

#### DisableDebugging()

void Garfield::Track::DisableDebugging () [inline]
Switch off debugging messages.
Definition at line 93 of file Track.hh.
00093 { m_debug = false; }

#### DisablePlotting()

void Garfield::Track::DisablePlotting ()
Switch off plotting.

#### EnableDebugging()

void Garfield::Track::EnableDebugging () [inline]
Switch on debugging messages.
Definition at line 91 of file Track.hh.
00091 { m_debug = true; }

#### EnablePlotting()

void Garfield::Track::EnablePlotting (
ViewDrift * viewer)
Switch on plotting.

#### GetBeta()

double Garfield::Track::GetBeta () const [inline]
Return the speed ( $\beta = v/c$) of the projectile.
Definition at line 57 of file Track.hh.
00057 { return sqrt(m_beta2); }

#### GetBetaGamma()

double Garfield::Track::GetBetaGamma () const [inline]
Return the $\beta\gamma$ of the projectile.
Definition at line 55 of file Track.hh.
00055 { return sqrt(m_beta2 / (1. - m_beta2)); }

#### GetCharge()

double Garfield::Track::GetCharge () const [inline]
Get the charge of the projectile.
Definition at line 66 of file Track.hh.
00066 { return m_q; }

#### GetClusterDensity()

virtual double Garfield::Track::GetClusterDensity () [inline], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented in Garfield::TrackBichsel, Garfield::TrackDegrade, Garfield::TrackElectron, Garfield::TrackHeed, Garfield::TrackPAI, and Garfield::TrackSimple.
Definition at line 81 of file Track.hh.
00081 { return 0.; }

#### GetEnergy()

double Garfield::Track::GetEnergy () const [inline]
Return the particle energy.
Definition at line 53 of file Track.hh.

00053 { return m_energy; }

### GetGamma()

double Garfield::Track::GetGamma () const [inline]
Return the Lorentz factor of the projectile.
Definition at line 59 of file Track.hh.
00059 { return sqrt(1. / (1. - m_beta2)); }

### GetKineticEnergy()

double Garfield::Track::GetKineticEnergy () const [inline]
Return the kinetic energy of the projectile.
Definition at line 63 of file Track.hh.
00063 { return m_energy - m_mass; }

### GetMass()

double Garfield::Track::GetMass () const [inline]
Get the mass [eV / c2] of the projectile.
Definition at line 68 of file Track.hh.
00068 { return m_mass; }

### GetMomentum()

double Garfield::Track::GetMomentum () const [inline]
Return the particle momentum.
Definition at line 61 of file Track.hh.
00061 { return m_mass * sqrt(m_beta2 / (1. - m_beta2)); }

### GetStoppingPower()

virtual double Garfield::Track::GetStoppingPower () [inline], [virtual]
Get the stopping power (mean energy loss [eV] per cm).
Reimplemented in Garfield::TrackBichsel, Garfield::TrackDegrade, Garfield::TrackElectron, Garfield::TrackHeed, Garfield::TrackPAI, and Garfield::TrackSimple.
Definition at line 83 of file Track.hh.
00083 { return 0.; }

### NewTrack()

virtual bool Garfield::Track::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [pure virtual]
Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implemented in Garfield::TrackBichsel, Garfield::TrackDegrade, Garfield::TrackElectron, Garfield::TrackHeed, Garfield::TrackPAI, Garfield::TrackSimple, Garfield::TrackSrim, and Garfield::TrackTrim.

### PlotCluster()

void Garfield::Track::PlotCluster (
const double x0,
const double y0,
const double z0) [protected]

PlotNewTrack()

void Garfield::Track::PlotNewTrack (
const double x0,
const double y0,
const double z0) [protected]

SetBeta()

void Garfield::Track::SetBeta (
const double beta)
Set the speed ( $\beta = v/c$) of the particle.

SetBetaGamma()

void Garfield::Track::SetBetaGamma (
const double bg)
Set the relative momentum of the particle.

SetEnergy()

void Garfield::Track::SetEnergy (
const double e)
Set the particle energy.

SetGamma()

void Garfield::Track::SetGamma (
const double gamma)
Set the Lorentz factor of the particle.

SetKineticEnergy()

void Garfield::Track::SetKineticEnergy (
const double ekin)
Set the kinetic energy of the particle.

SetMomentum()

void Garfield::Track::SetMomentum (
const double p)
Set the particle momentum.

SetParticle()

virtual void Garfield::Track::SetParticle (
const std::string & part) [virtual]
Set the type of charged particle.

- electron,e-
- positron,e+
- muon,mu-
- mu+
- pion,pi-
- pi+
- kaon,K-
- K+

- proton,p
- anti-proton,p-bar
- deuteron,d
- alpha

Reimplemented in Garfield::TrackDegrade, Garfield::TrackElectron, and Garfield::TrackTrim.

SetSensor()

void Garfield::Track::SetSensor (
Sensor * s)
Set the sensor through which to transport the particle.

StepBfield()

std::array< double, 3 > Garfield::Track::StepBfield (
const double dt,
const double qoverm,
const double vmag,
double bx,
double by,
double bz,
std::array< double, 3 > & dir) [static], [protected]

### 7.108.4 Member Data Documentation

m_beta2

double Garfield::Track::m_beta2 = 1. [protected]
Definition at line 102 of file Track.hh.

m_className

std::string Garfield::Track::m_className = "Track" [protected]
Definition at line 96 of file Track.hh.

m_debug

bool Garfield::Track::m_debug = false [protected]
Definition at line 112 of file Track.hh.

m_energy

double Garfield::Track::m_energy = 0. [protected]
Definition at line 101 of file Track.hh.

m_isChanged

bool Garfield::Track::m_isChanged = true [protected]
Definition at line 108 of file Track.hh.

m_isElectron

bool Garfield::Track::m_isElectron = false [protected]
Definition at line 103 of file Track.hh.

m_mass

double Garfield::Track::m_mass [protected]
Definition at line 100 of file Track.hh.

m_particleName

std::string Garfield::Track::m_particleName = "mu-" [protected]
Definition at line 104 of file Track.hh.

m_plotId

size_t Garfield::Track::m_plotId = 0 [protected]
Definition at line 114 of file Track.hh.

m_q

double Garfield::Track::m_q = -1. [protected]
Definition at line 98 of file Track.hh.

m_sensor

Sensor* Garfield::Track::m_sensor = nullptr [protected]
Definition at line 106 of file Track.hh.

m_spin

int Garfield::Track::m_spin = 1 [protected]
Definition at line 99 of file Track.hh.

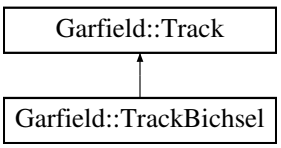
m_viewer

ViewDrift* Garfield::Track::m_viewer = nullptr [protected]
Definition at line 110 of file Track.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Track.hh

## 7.109 Garfield::TrackBichsel Class Reference

Generate tracks using differential cross-sections for silicon computed by Hans Bichsel.
#include <TrackBichsel.hh>
Inheritance diagram for Garfield::TrackBichsel:




Classes

- struct Cluster

Public Member Functions

- TrackBichsel ()
  Default constructor.
- TrackBichsel (Sensor *sensor)
  Constructor.
- virtual ~TrackBichsel ()
  Destructor.
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override
  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).

- const std::vector< Cluster > & GetClusters () const
- double GetClusterDensity () override

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- double GetStoppingPower () override

  Get the stopping power (mean energy loss [eV] per cm).
- bool Initialise ()
- bool ComputeCrossSection ()

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- virtual void SetParticle (const std::string &part)

  Set the type of charged particle.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

Switch off plotting.

- void EnableDebugging ()

  Switch on debugging messages.

- void DisableDebugging ()

  Switch off debugging messages.

Private Attributes

- std::array< double, NEnergyBins+1 > m_E
- std::array< double, NEnergyBins > m_dfdE

  Optical oscillator strength density.

- std::array< double, NEnergyBins > m_eps1

  Real part of the dielectric function.

- std::array< double, NEnergyBins > m_eps2

  Imaginary part of the dielectric function.

- std::array< double, NEnergyBins > m_int

  Integral over the generalised oscillator strength density.

- std::array< double, NEnergyBins > m_k1

  Lower limit of the integral over the GOS.

- std::array< double, NCdfBins > m_tab
- double m_density

  Density of silicon.

- double m_conv = 0.0092456

  Conversion from optical loss function to oscillator strength density.

- bool m_initialised = false
- double m_imfp = 0.

  Inverse mean free path [cm-1].

- double m_dEdx = 0.

  Stopping power [eV/cm].

- double m_speed = SpeedOfLight

  Particle speed.

- std::vector< Cluster > m_clusters

Static Private Attributes

- static constexpr size_t NEnergyBins = 1250
- static constexpr size_t NCdfBins = 10000

Additional Inherited Members

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

### 7.109.1 Detailed Description

Generate tracks using differential cross-sections for silicon computed by Hans Bichsel.
References:

- H. Bichsel, Rev. Mod. Phys. 60 (1988), 663-699
- https://faculty.washington.edu/hbichsel/

Definition at line 18 of file TrackBichsel.hh.

### 7.109.2 Constructor & Destructor Documentation

#### TrackBichsel() [1/2]

Garfield::TrackBichsel::TrackBichsel () [inline]
Default constructor.
Definition at line 26 of file TrackBichsel.hh.

```
00026 : TrackBichsel(nullptr) {}
```

#### TrackBichsel() [2/2]

Garfield::TrackBichsel::TrackBichsel (
Sensor * sensor)
Constructor.

#### ~TrackBichsel()

virtual Garfield::TrackBichsel::~TrackBichsel () [inline], [virtual]
Destructor.
Definition at line 30 of file TrackBichsel.hh.

```
00030 {}
```

### 7.109.3 Member Function Documentation

#### ComputeCrossSection()

bool Garfield::TrackBichsel::ComputeCrossSection ()

#### GetClusterDensity()

double Garfield::TrackBichsel::GetClusterDensity () [override], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented from Garfield::Track.

### GetClusters()

const std::vector< Cluster > & Garfield::TrackBichsel::GetClusters () const [inline]
Definition at line 35 of file TrackBichsel.hh.
00035 { return m_clusters; }

### GetStoppingPower()

double Garfield::TrackBichsel::GetStoppingPower () [override], [virtual]
Get the stopping power (mean energy loss [eV] per cm).
Reimplemented from Garfield::Track.

### Initialise()

bool Garfield::TrackBichsel::Initialise ()

### NewTrack()

bool Garfield::TrackBichsel::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]
Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

## 7.109.4 Member Data Documentation

### m_clusters

std::vector<Cluster> Garfield::TrackBichsel::m_clusters [private]
Definition at line 76 of file TrackBichsel.hh.

### m_conv

double Garfield::TrackBichsel::m_conv = 0.0092456 [private]
Conversion from optical loss function to oscillator strength density.
Definition at line 64 of file TrackBichsel.hh.

### m_dEdx

double Garfield::TrackBichsel::m_dEdx = 0. [private]
Stopping power [eV/cm].
Definition at line 71 of file TrackBichsel.hh.

### m_density

double Garfield::TrackBichsel::m_density [private]
Density of silicon.
Definition at line 62 of file TrackBichsel.hh.

### m_dfdE

std::array<double, NEnergyBins> Garfield::TrackBichsel::m_dfdE [private]
Optical oscillator strength density.
Definition at line 48 of file TrackBichsel.hh.

### m_E

std::array<double, NEnergyBins + 1> Garfield::TrackBichsel::m_E [private]
Definition at line 45 of file TrackBichsel.hh.

### m_eps1

std::array<double, NEnergyBins> Garfield::TrackBichsel::m_eps1 [private]
Real part of the dielectric function.
Definition at line 50 of file TrackBichsel.hh.

### m_eps2

std::array<double, NEnergyBins> Garfield::TrackBichsel::m_eps2 [private]
Imaginary part of the dielectric function.
Definition at line 52 of file TrackBichsel.hh.

### m_imfp

double Garfield::TrackBichsel::m_imfp = 0. [private]
Inverse mean free path [cm-1].
Definition at line 69 of file TrackBichsel.hh.

### m_initialised

bool Garfield::TrackBichsel::m_initialised = false [private]
Definition at line 66 of file TrackBichsel.hh.

### m_int

std::array<double, NEnergyBins> Garfield::TrackBichsel::m_int [private]
Integral over the generalised oscillator strength density.
Definition at line 54 of file TrackBichsel.hh.

### m_k1

std::array<double, NEnergyBins> Garfield::TrackBichsel::m_k1 [private]
Lower limit of the integral over the GOS.
Definition at line 56 of file TrackBichsel.hh.

### m_speed

double Garfield::TrackBichsel::m_speed = SpeedOfLight [private]
Particle speed.
Definition at line 74 of file TrackBichsel.hh.

### m_tab

std::array<double, NCdfBins> Garfield::TrackBichsel::m_tab [private]
Definition at line 59 of file TrackBichsel.hh.

### NCdfBins

size_t Garfield::TrackBichsel::NCdfBins = 10000 [static], [constexpr], [private]
Definition at line 58 of file TrackBichsel.hh.

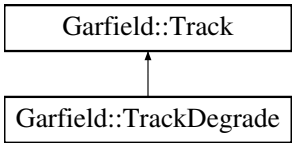
### NEnergyBins

size_t Garfield::TrackBichsel::NEnergyBins = 1250 [static], [constexpr], [private]
Definition at line 44 of file TrackBichsel.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackBichsel.hh

## 7.110 Garfield::TrackDegrade Class Reference

Interface to Degrade.
#include <TrackDegrade.hh>
Inheritance diagram for Garfield::TrackDegrade:




Classes

- struct Cluster
- struct Electron
- struct Excitation

Public Member Functions

- TrackDegrade ()

  Default constructor.
- TrackDegrade (Sensor *sensor)

  Constructor.
- virtual ~TrackDegrade ()

  Destructor.
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const
- double GetClusterDensity () override

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- double GetStoppingPower () override

  Get the stopping power (mean energy loss [eV] per cm).
- bool Initialise (Medium *medium, const bool verbose=false)
- void SetThresholdEnergy (const double eth)

  Set the energy down to which electrons are tracked (default: 2 eV).
- void StoreExcitations (const bool on=true, const double thr=4.)

  Store excitations in the cluster or not (off by default).
- void EnableBremsstrahlung (const bool on=true)

  Enable or disable bremsstrahlung.
- void EnableFullCascade (const bool on=true)

  Enable or disable detailed simulation of the deexcitation cascade.
- void SetParticle (const std::string &particle) override

  Set the type of charged particle.

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.

- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Protected Member Functions

- std::pair< std::vector< Electron >, std::vector< Excitation > > TransportDeltaElectron (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx, const double dy, const double dz)
- void SetupPenning (Medium *medium, std::array< double, 6 > &rP, std::array< double, 6 > &dP)
- bool IsInside (const double x, const double y, const double z)

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Protected Attributes

- std::vector< Cluster > m_clusters
- bool m_penning = true
- bool m_bremsStrahlung = true
- bool m_fullCascade = true
- bool m_storeExcitations = false
- double m_ethrExc = 4.
- double m_ethr = 2.
- double m_pressure = -1.
- double m_temperature = -1.
- std::string m_mediumName = ""
- unsigned int m_nGas = 0
- double m_dedx = -1.
- double m_clusterDensity = -1.
- std::array< double, 6 > m_rPenning
- std::array< double, 6 > m_dPenning

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

Additional Inherited Members

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

### 7.110.1 Detailed Description

Interface to Degrade.
Definition at line 14 of file TrackDegrade.hh.

### 7.110.2 Constructor & Destructor Documentation

TrackDegrade() [1/2]

Garfield::TrackDegrade::TrackDegrade () [inline]
Default constructor.
Definition at line 43 of file TrackDegrade.hh.
00043 : TrackDegrade(nullptr) {}

TrackDegrade() [2/2]

Garfield::TrackDegrade::TrackDegrade (
Sensor * sensor)
Constructor.

### ~TrackDegrade()

virtual Garfield::TrackDegrade::~TrackDegrade () [inline], [virtual]
Destructor.
Definition at line 47 of file TrackDegrade.hh.
00047 {}

## 7.110.3 Member Function Documentation

### EnableBremsstrahlung()

void Garfield::TrackDegrade::EnableBremsstrahlung (
const bool on = true) [inline]
Enable or disable bremsstrahlung.
Definition at line 66 of file TrackDegrade.hh.
00066 { m_bremsStrahlung = on; }

### EnableFullCascade()

void Garfield::TrackDegrade::EnableFullCascade (
const bool on = true) [inline]
Enable or disable detailed simulation of the deexcitation cascade.
Definition at line 68 of file TrackDegrade.hh.
00068 { m_fullCascade = on; }

### GetClusterDensity()

double Garfield::TrackDegrade::GetClusterDensity () [override], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented from Garfield::Track.

### GetClusters()

const std::vector< Cluster > & Garfield::TrackDegrade::GetClusters () const [inline]
Definition at line 52 of file TrackDegrade.hh.
00052 { return m_clusters; }

### GetStoppingPower()

double Garfield::TrackDegrade::GetStoppingPower () [override], [virtual]
Get the stopping power (mean energy loss [eV] per cm).
Reimplemented from Garfield::Track.

### Initialise()

bool Garfield::TrackDegrade::Initialise (
Medium * medium,
const bool verbose = false)

### IsInside()

bool Garfield::TrackDegrade::IsInside (
const double x,
const double y,
const double z) [protected]

### NewTrack()

bool Garfield::TrackDegrade::NewTrack (
const double x0,
const double y0,
const double z0,

const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]

Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

SetParticle()

void Garfield::TrackDegrade::SetParticle (
const std::string & part) [override], [virtual]

Set the type of charged particle.

- electron,e-
- positron,e+
- muon,mu-
- mu+
- pion,pi-
- pi+
- kaon,K-
- K+
- proton,p
- anti-proton,p-bar
- deuteron,d
- alpha

Reimplemented from Garfield::Track.

SetThresholdEnergy()

void Garfield::TrackDegrade::SetThresholdEnergy (
const double eth)

Set the energy down to which electrons are tracked (default: 2 eV).

SetupPenning()

void Garfield::TrackDegrade::SetupPenning (
Medium * medium,
std::array< double, 6 > & rP,
std::array< double, 6 > & dP) [protected]

StoreExcitations()

void Garfield::TrackDegrade::StoreExcitations (
const bool on = true,
const double thr = 4.) [inline]

Store excitations in the cluster or not (off by default).
Definition at line 61 of file TrackDegrade.hh.

```
00061                                                 {
00062     m_storeExcitations = on;
00063     m_ethrExc = std::max(thr, 1.e-3);
00064   }
```

TransportDeltaElectron()

std::pair< std::vector< Electron >, std::vector< Excitation > > Garfield::TrackDegrade::TransportDeltaElectron (
const double x0,
const double y0,
const double z0,
const double t0,
const double e0,
const double dx,
const double dy,
const double dz) [protected]

### 7.110.4 Member Data Documentation

m_bremsStrahlung

bool Garfield::TrackDegrade::m_bremsStrahlung = true [protected]
Definition at line 76 of file TrackDegrade.hh.

m_clusterDensity

double Garfield::TrackDegrade::m_clusterDensity = -1. [protected]
Definition at line 90 of file TrackDegrade.hh.

m_clusters

std::vector<Cluster> Garfield::TrackDegrade::m_clusters [protected]
Definition at line 73 of file TrackDegrade.hh.

m_dedx

double Garfield::TrackDegrade::m_dedx = -1. [protected]
Definition at line 89 of file TrackDegrade.hh.

m_dPenning

std::array<double, 6> Garfield::TrackDegrade::m_dPenning [protected]
Definition at line 93 of file TrackDegrade.hh.

m_ethr

double Garfield::TrackDegrade::m_ethr = 2. [protected]
Definition at line 82 of file TrackDegrade.hh.

m_ethrExc

double Garfield::TrackDegrade::m_ethrExc = 4. [protected]
Definition at line 80 of file TrackDegrade.hh.

m_fullCascade

bool Garfield::TrackDegrade::m_fullCascade = true [protected]
Definition at line 77 of file TrackDegrade.hh.

m_mediumName

std::string Garfield::TrackDegrade::m_mediumName = "" [protected]
Definition at line 86 of file TrackDegrade.hh.

m_nGas

unsigned int Garfield::TrackDegrade::m_nGas = 0 [protected]
Definition at line 87 of file TrackDegrade.hh.

m_penning

bool Garfield::TrackDegrade::m_penning = true [protected]
Definition at line 75 of file TrackDegrade.hh.

m_pressure

double Garfield::TrackDegrade::m_pressure = -1. [protected]
Definition at line 84 of file TrackDegrade.hh.

m_rPenning

std::array<double, 6> Garfield::TrackDegrade::m_rPenning [protected]
Definition at line 92 of file TrackDegrade.hh.

m_storeExcitations

bool Garfield::TrackDegrade::m_storeExcitations = false [protected]
Definition at line 78 of file TrackDegrade.hh.

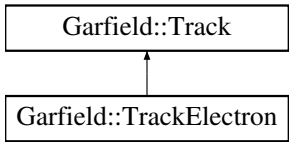
m_temperature

double Garfield::TrackDegrade::m_temperature = -1. [protected]
Definition at line 85 of file TrackDegrade.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh

## 7.111 Garfield::TrackElectron Class Reference

[WIP] Ionization calculation based on MIP program (S. Biagi).
#include <TrackElectron.hh>
Inheritance diagram for Garfield::TrackElectron:




Classes

- struct Cluster
- struct Parameters

Public Member Functions

- TrackElectron ()

  Constructor.
- virtual ~TrackElectron ()

  Destructor.
- void SetParticle (const std::string &particle) override

  Set the type of charged particle.
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const
- double GetClusterDensity () override

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).

- double GetStoppingPower () override

  Get the stopping power (mean energy loss [eV] per cm).

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Static Private Member Functions

- static bool Setup (Medium *gas, std::vector< Parameters > &par, std::vector< double > &frac)
- static bool Update (const double density, const double beta2, const std::vector< Parameters > &par, const std::vector< double > &frac, std::vector< double > &prob, double &mfp, double &dedx)
- static double Delta (const double x, const Parameters &par)
- static double Esec (const double e0, const Parameters &par)

Private Attributes

- std::vector< Cluster > m_clusters
- double m_mfp = 0.
- double m_dedx = 0.

Additional Inherited Members

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

### 7.111.1 Detailed Description

[WIP] Ionization calculation based on MIP program (S. Biagi).
Definition at line 14 of file TrackElectron.hh.

### 7.111.2 Constructor & Destructor Documentation

TrackElectron()

Garfield::TrackElectron::TrackElectron ()
Constructor.

~TrackElectron()

virtual Garfield::TrackElectron::~TrackElectron () [inline], [virtual]
Destructor.
Definition at line 24 of file TrackElectron.hh.
00024 {}

### 7.111.3 Member Function Documentation

#### Delta()

double Garfield::TrackElectron::Delta (
const double x,
const Parameters & par) [static], [private]

#### Esec()

double Garfield::TrackElectron::Esec (
const double e0,
const Parameters & par) [static], [private]

#### GetClusterDensity()

double Garfield::TrackElectron::GetClusterDensity () [override], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented from Garfield::Track.

#### GetClusters()

const std::vector< Cluster > & Garfield::TrackElectron::GetClusters () const [inline]
Definition at line 32 of file TrackElectron.hh.
00032 { return m_clusters; }

#### GetStoppingPower()

double Garfield::TrackElectron::GetStoppingPower () [override], [virtual]
Get the stopping power (mean energy loss [eV] per cm).
Reimplemented from Garfield::Track.

#### NewTrack()

bool Garfield::TrackElectron::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]
Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

#### SetParticle()

void Garfield::TrackElectron::SetParticle (
const std::string & part) [override], [virtual]
Set the type of charged particle.

- electron,e-
- positron,e+
- muon,mu-
- mu+
- pion,pi-
- pi+
- kaon,K-

- K+
- proton,p
- anti-proton,p-bar
- deuteron,d
- alpha

Reimplemented from Garfield::Track.

Setup()

bool Garfield::TrackElectron::Setup (
Medium * gas,
std::vector< Parameters > & par,
std::vector< double > & frac) [static], [private]

Update()

bool Garfield::TrackElectron::Update (
const double density,
const double beta2,
const std::vector< Parameters > & par,
const std::vector< double > & frac,
std::vector< double > & prob,
double & mfp,
double & dedx) [static], [private]

### 7.111.4 Member Data Documentation

m_clusters

std::vector<Cluster> Garfield::TrackElectron::m_clusters [private]
Definition at line 55 of file TrackElectron.hh.

m_dedx

double Garfield::TrackElectron::m_dedx = 0. [private]
Definition at line 60 of file TrackElectron.hh.

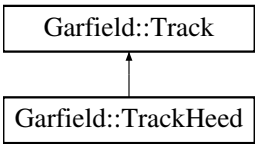
m_mfp

double Garfield::TrackElectron::m_mfp = 0. [private]
Definition at line 58 of file TrackElectron.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackElectron.hh

## 7.112 Garfield::TrackHeed Class Reference

Generate tracks using Heed++.
#include <TrackHeed.hh>
Inheritance diagram for Garfield::TrackHeed:



Classes

- struct Cluster
- struct Electron
- struct Photon

Public Member Functions

- TrackHeed ()

  Default constructor.
- TrackHeed (Sensor *sensor)

  Constructor.
- virtual ~TrackHeed ()

  Destructor.
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const
- bool GetCluster (double &xc, double &yc, double &zc, double &tc, int &nc, double &ec, double &extra)
- bool GetCluster (double &xc, double &yc, double &zc, double &tc, int &ne, int &ni, double &ec, double &extra)
- bool GetCluster (double &xc, double &yc, double &zc, double &tc, int &ne, int &ni, int &np, double &ec, double &extra)

  Get the next "cluster" (ionising collision of the charged particle).
- bool GetElectron (const unsigned int i, double &x, double &y, double &z, double &t, double &e, double &dx, double &dy, double &dz)

  Retrieve the properties of a conduction or delta electron in the current cluster.
- bool GetIon (const unsigned int i, double &x, double &y, double &z, double &t) const

  Retrieve the properties of an ion in the current cluster.
- bool GetPhoton (const unsigned int i, double &x, double &y, double &z, double &t, double &e, double &dx, double &dy, double &dz) const

  Retrieve the properties of an unabsorbed photon.
- double GetClusterDensity () override

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- double GetStoppingPower () override

  Get the stopping power (mean energy loss [eV] per cm).
- double GetW () const

  Return the W value of the medium (of the last simulated track).
- double GetFanoFactor () const

  Return the Fano factor of the medium (of the last simulated track).
- double GetPhotoAbsorptionCrossSection (const double e) const

  Return the photoabsorption cross-section at a given energy.
- bool Initialise (Medium *medium, const bool verbose=false)

  Compute the differential cross-section for a given medium.
- Cluster TransportDeltaElectron (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0)

  Simulate a delta electron.
- void TransportDeltaElectron (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0, int &ne, int &ni)

  Simulate a delta electron.
- void TransportDeltaElectron (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0, int &ne)

  Simulate a delta electron.

- Cluster TransportPhoton (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0)

  Simulate a photon.
- void TransportPhoton (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0, int &ne, int &ni, int &np)

  Simulate a photon.
- void TransportPhoton (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0, int &ne, int &ni)

  Simulate a photon.
- void TransportPhoton (const double x0, const double y0, const double z0, const double t0, const double e0, const double dx0, const double dy0, const double dz0, int &ne)

  Simulate a photon.
- void EnableElectricField ()

  Take the electric field into account in the stepping algorithm.
- void DisableElectricField ()

  Do not take the electric field into account in the stepping algorithm.
- void EnableMagneticField ()

  Take the magnetic field into account in the stepping algorithm.
- void DisableMagneticField ()

  Do not take the magnetic field into account in the stepping algorithm.
- void SetSteppingLimits (const double maxStep, const double radStraight, const double stepAngle↩ Straight, const double stepAngleCurved)

  Set parameters for calculating the particle trajectory.
- void GetSteppingLimits (double &maxStep, double &radStraight, double &stepAngleStraight, double &stepAngleCurved)
- void CrossInactiveMedia (const bool on=true)
- void EnableCoulombScattering (const bool on=true)
- void EnableDeltaElectronTransport ()

  Switch simulation of delta electrons on.
- void DisableDeltaElectronTransport ()

  Switch simulation of delta electrons off.
- void EnablePhotonReabsorption (const bool on=true)

  Simulate (or not) the photons produced in the atomic relaxation cascade.
- void EnablePhotoAbsorptionCrossSectionOutput (const bool on)

  Write the photoabsorption cross-sections used to a text file.
- void SetEnergyMesh (const double e0, const double e1, const int nsteps)

  Specify the energy mesh to be used.
- void SetParticleUser (const double m, const double z)

  Define particle mass and charge (for exotic particles).
- void EnableOneStepFly (const bool on)

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- virtual void SetParticle (const std::string &part)

  Set the type of charged particle.
- void SetEnergy (const double e)

Set the particle energy.

- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Private Member Functions

- TrackHeed (const TrackHeed &heed)
- TrackHeed & operator= (const TrackHeed &heed)
- bool SetupGas (Medium *medium)
- bool SetupMaterial (Medium *medium)
- bool SetupDelta (const std::string &databasePath)
- bool AddCluster (Heed::HeedPhoton *virtualPhoton, std::vector< Cluster > &clusters)
- void AddElectrons (const std::vector< Heed::HeedCondElectron > &conductionElectrons, std::vector< Electron > &electrons)
- bool IsInside (const double x, const double y, const double z)
- bool UpdateBoundingBox (bool &update)

Private Attributes

- bool m_oneStepFly = false
- bool m_ready = false
- bool m_hasActiveTrack = false
- double m_mediumDensity = -1.
- std::string m_mediumName = ""
- bool m_usePacsOutput = false
- bool m_doPhotonReabsorption = true
- bool m_crossInactiveMedia = false
- bool m_coulombScattering = false
- bool m_useBfieldAuto = true
- bool m_doDeltaTransport = true
- std::vector< Cluster > m_clusters
- size_t m_cluster = 0
- std::unique_ptr< Heed::particle_def > m_particle_def
- std::unique_ptr< Heed::HeedMatterDef > m_matter
- std::unique_ptr< Heed::GasDef > m_gas
- std::unique_ptr< Heed::MatterDef > m_material
- double m_emin = 2.e-6
- double m_emax = 2.e-1
- unsigned int m_nEnergyIntervals = 200
- std::unique_ptr< Heed::EnergyMesh > m_energyMesh
- std::unique_ptr< Heed::EnTransfCS > m_transferCs
- std::unique_ptr< Heed::ElElasticScat > m_elScat
- std::unique_ptr< Heed::ElElasticScatLowSigma > m_lowSigma
- std::unique_ptr< Heed::PairProd > m_pairProd
- std::unique_ptr< Heed::HeedDeltaElectronCS > m_deltaCs
- std::unique_ptr< HeedChamber > m_chamber
- std::unique_ptr< HeedFieldMap > m_fieldMap
- double m_lX = 0.
- double m_lY = 0.
- double m_lZ = 0.
- double m_cX = 0.
- double m_cY = 0.
- double m_cZ = 0.
- double m_maxStep = 100.

  Max. step length.
- double m_radStraight = 1000.

  Bending radius beyond which to use straight-line approximation.
- double m_stepAngleStraight = 0.1

  Angular step for curved trajectories approximated by straight-line steps.
- double m_stepAngleCurved = 0.2

  Angular step for curved lines.

Additional Inherited Members

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

### 7.112.1 Detailed Description

Generate tracks using Heed++.
Definition at line 34 of file TrackHeed.hh.

### 7.112.2 Constructor & Destructor Documentation

#### TrackHeed() [1/3]

Garfield::TrackHeed::TrackHeed () [inline]
Default constructor.
Definition at line 68 of file TrackHeed.hh.

```
00068 : TrackHeed(nullptr) {}
```

#### TrackHeed() [2/3]

Garfield::TrackHeed::TrackHeed (
Sensor * sensor)
Constructor.

#### ~TrackHeed()

virtual Garfield::TrackHeed::~TrackHeed () [virtual]
Destructor.

#### TrackHeed() [3/3]

Garfield::TrackHeed::TrackHeed (
const TrackHeed & heed) [private]

### 7.112.3 Member Function Documentation

#### AddCluster()

bool Garfield::TrackHeed::AddCluster (
Heed::HeedPhoton * virtualPhoton,
std::vector< Cluster > & clusters) [private]

#### AddElectrons()

void Garfield::TrackHeed::AddElectrons (
const std::vector< Heed::HeedCondElectron > & conductionElectrons,
std::vector< Electron > & electrons) [private]

### CrossInactiveMedia()

void Garfield::TrackHeed::CrossInactiveMedia (
const bool on = true) [inline]
Definition at line 250 of file TrackHeed.hh.
00250 { m_crossInactiveMedia = on; }

### DisableDeltaElectronTransport()

void Garfield::TrackHeed::DisableDeltaElectronTransport () [inline]
Switch simulation of delta electrons off.
Definition at line 257 of file TrackHeed.hh.
00257 { m_doDeltaTransport = false; }

### DisableElectricField()

void Garfield::TrackHeed::DisableElectricField ()
Do not take the electric field into account in the stepping algorithm.

### DisableMagneticField()

void Garfield::TrackHeed::DisableMagneticField ()
Do not take the magnetic field into account in the stepping algorithm.

### EnableCoulombScattering()

void Garfield::TrackHeed::EnableCoulombScattering (
const bool on = true) [inline]
Definition at line 251 of file TrackHeed.hh.
00251 {
00252 m_coulombScattering = on;
00253 }

### EnableDeltaElectronTransport()

void Garfield::TrackHeed::EnableDeltaElectronTransport () [inline]
Switch simulation of delta electrons on.
Definition at line 255 of file TrackHeed.hh.
00255 { m_doDeltaTransport = true; }

### EnableElectricField()

void Garfield::TrackHeed::EnableElectricField ()
Take the electric field into account in the stepping algorithm.

### EnableMagneticField()

void Garfield::TrackHeed::EnableMagneticField ()
Take the magnetic field into account in the stepping algorithm.

### EnableOneStepFly()

void Garfield::TrackHeed::EnableOneStepFly (
const bool on) [inline]
Definition at line 278 of file TrackHeed.hh.
00278 { m_oneStepFly = on; }

### EnablePhotoAbsorptionCrossSectionOutput()

void Garfield::TrackHeed::EnablePhotoAbsorptionCrossSectionOutput (
const bool on) [inline]
Write the photoabsorption cross-sections used to a text file.

Definition at line 265 of file TrackHeed.hh.

```
00265                                                {
00266     m_usePacsOutput = on;
00267   }
```

EnablePhotonReabsorption()

```
void Garfield::TrackHeed::EnablePhotonReabsorption (
                const bool on = true)   [inline]
```

Simulate (or not) the photons produced in the atomic relaxation cascade.
Definition at line 260 of file TrackHeed.hh.

```
00260                                                {
00261     m_doPhotonReabsorption = on;
00262   }
```

GetCluster() [1/3]

```
bool Garfield::TrackHeed::GetCluster (
                double & xc,
                double & yc,
                double & zc,
                double & tc,
                int & nc,
                double & ec,
                double & extra)
```

GetCluster() [2/3]

```
bool Garfield::TrackHeed::GetCluster (
                double & xc,
                double & yc,
                double & zc,
                double & tc,
                int & ne,
                int & ni,
                double & ec,
                double & extra)
```

GetCluster() [3/3]

```
bool Garfield::TrackHeed::GetCluster (
                double & xc,
                double & yc,
                double & zc,
                double & tc,
                int & ne,
                int & ni,
                int & np,
                double & ec,
                double & extra)
```

Get the next "cluster" (ionising collision of the charged particle).

Parameters

| | |
|---|---|
| xc,yc,zc | coordinates of the collision |
| tc | time of the collision |
| ne | number of electrons |
| ni | number of ions |
| np | number of fluorescence photons |
| ec | deposited energy |

Parameters

| | |
|---|---|
| extra | additional information (not always implemented) |

### GetClusterDensity()

double Garfield::TrackHeed::GetClusterDensity () [override], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented from Garfield::Track.

### GetClusters()

const std::vector< Cluster > & Garfield::TrackHeed::GetClusters () const [inline]
Definition at line 77 of file TrackHeed.hh.

```
00077 { return m_clusters; }
```

### GetElectron()

```
bool Garfield::TrackHeed::GetElectron (
                const unsigned int i,
                double & x,
                double & y,
                double & z,
                double & t,
                double & e,
                double & dx,
                double & dy,
                double & dz)
```

Retrieve the properties of a conduction or delta electron in the current cluster.

Parameters

| | |
|---|---|
| i | index of the electron |
| x,y,z | coordinates of the electron |
| t | time |
| e | kinetic energy (only meaningful for delta-electrons) |
| dx,dy,dz | direction vector (only meaningful for delta-electrons) |

### GetFanoFactor()

double Garfield::TrackHeed::GetFanoFactor () const
Return the Fano factor of the medium (of the last simulated track).

### GetIon()

```
bool Garfield::TrackHeed::GetIon (
                const unsigned int i,
                double & x,
                double & y,
                double & z,
                double & t) const
```

Retrieve the properties of an ion in the current cluster.

Parameters

| | |
|---|---|
| i | index of the ion |

Parameters

| | |
|---|---|
| x,y,z | coordinates of the ion |
| t | time |

### GetPhotoAbsorptionCrossSection()

double Garfield::TrackHeed::GetPhotoAbsorptionCrossSection (
const double e) const

Return the photoabsorption cross-section at a given energy.

### GetPhoton()

bool Garfield::TrackHeed::GetPhoton (
const unsigned int i,
double & x,
double & y,
double & z,
double & t,
double & e,
double & dx,
double & dy,
double & dz) const

Retrieve the properties of an unabsorbed photon.

Parameters

| | |
|---|---|
| i | index of the photon |
| x,y,z | coordinates of the photon |
| t | time |
| e | photon energy |
| dx,dy,dz | direction vector |

### GetSteppingLimits()

void Garfield::TrackHeed::GetSteppingLimits (
double & maxStep,
double & radStraight,
double & stepAngleStraight,
double & stepAngleCurved) [inline]

Definition at line 242 of file TrackHeed.hh.

```
00243                                                                 {
00244     maxStep = m_maxStep;
00245     radStraight = m_radStraight;
00246     stepAngleStraight = m_stepAngleStraight;
00247     stepAngleCurved = m_stepAngleCurved;
00248   }
```

### GetStoppingPower()

double Garfield::TrackHeed::GetStoppingPower () [override], [virtual]

Get the stopping power (mean energy loss [eV] per cm).

Reimplemented from Garfield::Track.

### GetW()

double Garfield::TrackHeed::GetW () const

Return the W value of the medium (of the last simulated track).

### Initialise()

```
bool Garfield::TrackHeed::Initialise (
        Medium * medium,
        const bool verbose = false)
```

Compute the differential cross-section for a given medium.

### IsInside()

```
bool Garfield::TrackHeed::IsInside (
        const double x,
        const double y,
        const double z)   [private]
```

### NewTrack()

```
bool Garfield::TrackHeed::NewTrack (
        const double x0,
        const double y0,
        const double z0,
        const double t0,
        const double dx0,
        const double dy0,
        const double dz0)   [override], [virtual]
```

Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

### operator=()

```
TrackHeed & Garfield::TrackHeed::operator= (
        const TrackHeed & heed)   [private]
```

### SetEnergyMesh()

```
void Garfield::TrackHeed::SetEnergyMesh (
        const double e0,
        const double e1,
        const int nsteps)
```

Specify the energy mesh to be used.

Parameters

| | |
|---|---|
| e0,e1 | lower/higher limit of the energy range [eV] |
| nsteps | number of intervals |

### SetParticleUser()

```
void Garfield::TrackHeed::SetParticleUser (
        const double m,
        const double z)
```

Define particle mass and charge (for exotic particles).
For standard particles Track::SetParticle should be used.

### SetSteppingLimits()

```
void Garfield::TrackHeed::SetSteppingLimits (
        const double maxStep,
        const double radStraight,
```

const double stepAngleStraight,
const double stepAngleCurved) [inline]

Set parameters for calculating the particle trajectory.

Parameters

| | |
|---|---|
| maxStep | maximum step length |
| radStraight | radius beyond which to approximate circles by polylines. |
| stepAngleStraight | max. angular step (in radian) when using polyline steps. |
| stepAngleCurved | max. angular step (in radian) when using circular steps. |

Definition at line 234 of file TrackHeed.hh.

```
00236                                                {
00237     m_maxStep = maxStep;
00238     m_radStraight = radStraight;
00239     m_stepAngleStraight = stepAngleStraight;
00240     m_stepAngleCurved = stepAngleCurved;
00241   }
```

### SetupDelta()

bool Garfield::TrackHeed::SetupDelta (
const std::string & databasePath) [private]

### SetupGas()

bool Garfield::TrackHeed::SetupGas (
Medium * medium) [private]

### SetupMaterial()

bool Garfield::TrackHeed::SetupMaterial (
Medium * medium) [private]

### TransportDeltaElectron() [1/3]

Cluster Garfield::TrackHeed::TransportDeltaElectron (
const double x0,
const double y0,
const double z0,
const double t0,
const double e0,
const double dx0,
const double dy0,
const double dz0)

Simulate a delta electron.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the delta electron |
| t0 | initial time |
| e0 | initial kinetic energy of the delta electron |
| dx0,dy0,dz0 | initial direction of the delta electron |

### TransportDeltaElectron() [2/3]

void Garfield::TrackHeed::TransportDeltaElectron (
const double x0,

const double y0,
const double z0,
const double t0,
const double e0,
const double dx0,
const double dy0,
const double dz0,
int & ne)

Simulate a delta electron.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the delta electron |
| t0 | initial time |
| e0 | initial kinetic energy of the delta electron |
| dx0,dy0,dz0 | initial direction of the delta electron |
| ne | number of electrons produced by the delta electron |

### TransportDeltaElectron() [3/3]

void Garfield::TrackHeed::TransportDeltaElectron (
const double x0,
const double y0,
const double z0,
const double t0,
const double e0,
const double dx0,
const double dy0,
const double dz0,
int & ne,
int & ni)

Simulate a delta electron.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the delta electron |
| t0 | initial time |
| e0 | initial kinetic energy of the delta electron |
| dx0,dy0,dz0 | initial direction of the delta electron |
| ne,ni | number of electrons/ions produced by the delta electron |

### TransportPhoton() [1/4]

Cluster Garfield::TrackHeed::TransportPhoton (
const double x0,
const double y0,
const double z0,
const double t0,
const double e0,
const double dx0,
const double dy0,
const double dz0)

Simulate a photon.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the photon |
| t0 | initial time |
| e0 | initial energy of the photon |
| dx0,dy0,dz0 | initial direction of the photon |

TransportPhoton() [2/4]

```
void Garfield::TrackHeed::TransportPhoton (
                const double x0,
                const double y0,
                const double z0,
                const double t0,
                const double e0,
                const double dx0,
                const double dy0,
                const double dz0,
                int & ne)
```

Simulate a photon.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the photon |
| t0 | initial time |
| e0 | initial energy of the photon |
| dx0,dy0,dz0 | initial direction of the photon |
| ne | number of electrons produced by the photon |

TransportPhoton() [3/4]

```
void Garfield::TrackHeed::TransportPhoton (
                const double x0,
                const double y0,
                const double z0,
                const double t0,
                const double e0,
                const double dx0,
                const double dy0,
                const double dz0,
                int & ne,
                int & ni)
```

Simulate a photon.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the photon |
| t0 | initial time |
| e0 | initial energy of the photon |
| dx0,dy0,dz0 | initial direction of the photon |
| ne | number of electrons produced by the photon |
| ni | number of ions produced by the photon |

### TransportPhoton() [4/4]

```
void Garfield::TrackHeed::TransportPhoton (
            const double x0,
            const double y0,
            const double z0,
            const double t0,
            const double e0,
            const double dx0,
            const double dy0,
            const double dz0,
            int & ne,
            int & ni,
            int & np)
```

Simulate a photon.

Parameters

| | |
|---|---|
| x0,y0,z0 | initial position of the photon |
| t0 | initial time |
| e0 | initial energy of the photon |
| dx0,dy0,dz0 | initial direction of the photon |
| ne | number of electrons produced by the photon |
| ni | number of ions produced by the photon |
| np | number of fluorescence photons |

### UpdateBoundingBox()

```
bool Garfield::TrackHeed::UpdateBoundingBox (
            bool & update)  [private]
```

## 7.112.4 Member Data Documentation

### m_chamber

```
std::unique_ptr<HeedChamber> Garfield::TrackHeed::m_chamber  [private]
```

Definition at line 325 of file TrackHeed.hh.

### m_cluster

```
size_t Garfield::TrackHeed::m_cluster = 0  [private]
```

Definition at line 302 of file TrackHeed.hh.

### m_clusters

```
std::vector<Cluster> Garfield::TrackHeed::m_clusters  [private]
```

Definition at line 301 of file TrackHeed.hh.

### m_coulombScattering

```
bool Garfield::TrackHeed::m_coulombScattering = false  [private]
```

Definition at line 297 of file TrackHeed.hh.

### m_crossInactiveMedia

```
bool Garfield::TrackHeed::m_crossInactiveMedia = false  [private]
```

Definition at line 296 of file TrackHeed.hh.

### m_cX

double Garfield::TrackHeed::m_cX = 0. [private]
Definition at line 330 of file TrackHeed.hh.

### m_cY

double Garfield::TrackHeed::m_cY = 0. [private]
Definition at line 330 of file TrackHeed.hh.

### m_cZ

double Garfield::TrackHeed::m_cZ = 0. [private]
Definition at line 330 of file TrackHeed.hh.

### m_deltaCs

std::unique_ptr<Heed::HeedDeltaElectronCS> Garfield::TrackHeed::m_deltaCs [private]
Definition at line 322 of file TrackHeed.hh.

### m_doDeltaTransport

bool Garfield::TrackHeed::m_doDeltaTransport = true [private]
Definition at line 299 of file TrackHeed.hh.

### m_doPhotonReabsorption

bool Garfield::TrackHeed::m_doPhotonReabsorption = true [private]
Definition at line 295 of file TrackHeed.hh.

### m_elScat

std::unique_ptr<Heed::ElElasticScat> Garfield::TrackHeed::m_elScat [private]
Definition at line 319 of file TrackHeed.hh.

### m_emax

double Garfield::TrackHeed::m_emax = 2.e-1 [private]
Definition at line 313 of file TrackHeed.hh.

### m_emin

double Garfield::TrackHeed::m_emin = 2.e-6 [private]
Definition at line 312 of file TrackHeed.hh.

### m_energyMesh

std::unique_ptr<Heed::EnergyMesh> Garfield::TrackHeed::m_energyMesh [private]
Definition at line 315 of file TrackHeed.hh.

### m_fieldMap

std::unique_ptr<HeedFieldMap> Garfield::TrackHeed::m_fieldMap [private]
Definition at line 326 of file TrackHeed.hh.

### m_gas

std::unique_ptr<Heed::GasDef> Garfield::TrackHeed::m_gas [private]
Definition at line 308 of file TrackHeed.hh.

### m_hasActiveTrack

bool Garfield::TrackHeed::m_hasActiveTrack = false [private]
Definition at line 288 of file TrackHeed.hh.

### m_lowSigma

std::unique_ptr<Heed::ElElasticScatLowSigma> Garfield::TrackHeed::m_lowSigma [private]
Definition at line 320 of file TrackHeed.hh.

### m_lX

double Garfield::TrackHeed::m_lX = 0. [private]
Definition at line 329 of file TrackHeed.hh.

### m_lY

double Garfield::TrackHeed::m_lY = 0. [private]
Definition at line 329 of file TrackHeed.hh.

### m_lZ

double Garfield::TrackHeed::m_lZ = 0. [private]
Definition at line 329 of file TrackHeed.hh.

### m_material

std::unique_ptr<Heed::MatterDef> Garfield::TrackHeed::m_material [private]
Definition at line 309 of file TrackHeed.hh.

### m_matter

std::unique_ptr<Heed::HeedMatterDef> Garfield::TrackHeed::m_matter [private]
Definition at line 307 of file TrackHeed.hh.

### m_maxStep

double Garfield::TrackHeed::m_maxStep = 100. [private]
Max. step length.
Definition at line 334 of file TrackHeed.hh.

### m_mediumDensity

double Garfield::TrackHeed::m_mediumDensity = -1. [private]
Definition at line 290 of file TrackHeed.hh.

### m_mediumName

std::string Garfield::TrackHeed::m_mediumName = "" [private]
Definition at line 291 of file TrackHeed.hh.

### m_nEnergyIntervals

unsigned int Garfield::TrackHeed::m_nEnergyIntervals = 200 [private]
Definition at line 314 of file TrackHeed.hh.

### m_oneStepFly

bool Garfield::TrackHeed::m_oneStepFly = false [private]
Definition at line 285 of file TrackHeed.hh.

### m_pairProd

std::unique_ptr<Heed::PairProd> Garfield::TrackHeed::m_pairProd [private]
Definition at line 321 of file TrackHeed.hh.

### m_particle_def

std::unique_ptr<Heed::particle_def> Garfield::TrackHeed::m_particle_def [private]
Definition at line 305 of file TrackHeed.hh.

### m_radStraight

double Garfield::TrackHeed::m_radStraight = 1000. [private]
Bending radius beyond which to use straight-line approximation.
Definition at line 336 of file TrackHeed.hh.

### m_ready

bool Garfield::TrackHeed::m_ready = false [private]
Definition at line 287 of file TrackHeed.hh.

### m_stepAngleCurved

double Garfield::TrackHeed::m_stepAngleCurved = 0.2 [private]
Angular step for curved lines.
Definition at line 340 of file TrackHeed.hh.

### m_stepAngleStraight

double Garfield::TrackHeed::m_stepAngleStraight = 0.1 [private]
Angular step for curved trajectories approximated by straight-line steps.
Definition at line 338 of file TrackHeed.hh.

### m_transferCs

std::unique_ptr<Heed::EnTransfCS> Garfield::TrackHeed::m_transferCs [private]
Definition at line 318 of file TrackHeed.hh.

### m_useBfieldAuto

bool Garfield::TrackHeed::m_useBfieldAuto = true [private]
Definition at line 298 of file TrackHeed.hh.

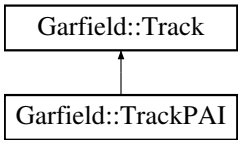
### m_usePacsOutput

bool Garfield::TrackHeed::m_usePacsOutput = false [private]
Definition at line 293 of file TrackHeed.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackHeed.hh

## 7.113 Garfield::TrackPAI Class Reference

#include <TrackPAI.hh>
Inheritance diagram for Garfield::TrackPAI:



Classes

- struct Cluster

Public Member Functions

- TrackPAI ()
- virtual ~TrackPAI ()
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const
- double GetClusterDensity () override

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- double GetStoppingPower () override

  Get the stopping power (mean energy loss [eV] per cm).

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- virtual void SetParticle (const std::string &part)

  Set the type of charged particle.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.

- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Private Member Functions

- bool SetupMedium (Medium *medium)
- bool SetupCrossSectionTable ()
- double ComputeMaxTransfer () const
- double ComputeCsTail (const double emin, const double emax)
- double ComputeDeDxTail (const double emin, const double emax)
- std::pair< double, double > SampleEnergyDeposit (const double u) const
- double SampleAsymptoticCs (double u) const
- double SampleAsymptoticCsSpinZero (const double emin, double u) const
- double SampleAsymptoticCsSpinHalf (const double emin, double u) const
- double SampleAsymptoticCsSpinOne (const double emin, double u) const
- double SampleAsymptoticCsElectron (const double emin, double u) const
- double SampleAsymptoticCsPositron (const double emin, double u) const
- double LossFunction (const double eps1, const double eps2) const

Private Attributes

- double m_speed = 0.
- double m_emax = 0.
- double m_imfp = 0.
- double m_dedx = 0.
- std::array< double, m_nSteps > m_eps1
- std::array< double, m_nSteps > m_eps2
- std::array< double, m_nSteps > m_epsInt
- std::array< double, m_nSteps > m_energies
- std::array< double, m_nSteps > m_cdf
- std::array< double, m_nSteps > m_rutherford
- std::vector< Cluster > m_clusters
- std::string m_mediumName = ""
- double m_mediumDensity = 0.
- double m_electronDensity = 0.

Static Private Attributes

- static constexpr size_t m_nSteps = 1000

Additional Inherited Members

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

### 7.113.1 Detailed Description

Definition at line 14 of file TrackPAI.hh.

### 7.113.2 Constructor & Destructor Documentation

#### TrackPAI()

Garfield::TrackPAI::TrackPAI ()

#### ~TrackPAI()

virtual Garfield::TrackPAI::~TrackPAI () [inline], [virtual]
Definition at line 24 of file TrackPAI.hh.
00024 {}

### 7.113.3 Member Function Documentation

#### ComputeCsTail()

double Garfield::TrackPAI::ComputeCsTail (
const double emin,
const double emax) [private]

#### ComputeDeDxTail()

double Garfield::TrackPAI::ComputeDeDxTail (
const double emin,
const double emax) [private]

#### ComputeMaxTransfer()

double Garfield::TrackPAI::ComputeMaxTransfer () const [private]

#### GetClusterDensity()

double Garfield::TrackPAI::GetClusterDensity () [override], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented from Garfield::Track.

### GetClusters()

const std::vector< Cluster > & Garfield::TrackPAI::GetClusters () const [inline]
Definition at line 30 of file TrackPAI.hh.

```
00030 { return m_clusters; }
```

### GetStoppingPower()

double Garfield::TrackPAI::GetStoppingPower () [override], [virtual]
Get the stopping power (mean energy loss [eV] per cm).
Reimplemented from Garfield::Track.

### LossFunction()

double Garfield::TrackPAI::LossFunction (
const double eps1,
const double eps2) const [inline], [private]
Definition at line 80 of file TrackPAI.hh.

```
00080                                                 {
00081     const double eps = eps1 * eps1 + eps2 * eps2;
00082     return eps > 0. ? eps2 / eps : 0.;
00083   }
```

### NewTrack()

bool Garfield::TrackPAI::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]
Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

### SampleAsymptoticCs()

double Garfield::TrackPAI::SampleAsymptoticCs (
double u) const [private]

### SampleAsymptoticCsElectron()

double Garfield::TrackPAI::SampleAsymptoticCsElectron (
const double emin,
double u) const [private]

### SampleAsymptoticCsPositron()

double Garfield::TrackPAI::SampleAsymptoticCsPositron (
const double emin,
double u) const [private]

### SampleAsymptoticCsSpinHalf()

double Garfield::TrackPAI::SampleAsymptoticCsSpinHalf (
const double emin,
double u) const [private]

### SampleAsymptoticCsSpinOne()

double Garfield::TrackPAI::SampleAsymptoticCsSpinOne (
const double emin,
double u) const [private]

### SampleAsymptoticCsSpinZero()

double Garfield::TrackPAI::SampleAsymptoticCsSpinZero (
const double emin,
double u) const [private]

### SampleEnergyDeposit()

std::pair< double, double > Garfield::TrackPAI::SampleEnergyDeposit (
const double u) const [private]

### SetupCrossSectionTable()

bool Garfield::TrackPAI::SetupCrossSectionTable () [private]

### SetupMedium()

bool Garfield::TrackPAI::SetupMedium (
Medium * medium) [private]

## 7.113.4 Member Data Documentation

### m_cdf

std::array<double, m_nSteps> Garfield::TrackPAI::m_cdf [private]
Definition at line 54 of file TrackPAI.hh.

### m_clusters

std::vector<Cluster> Garfield::TrackPAI::m_clusters [private]
Definition at line 57 of file TrackPAI.hh.

### m_dedx

double Garfield::TrackPAI::m_dedx = 0. [private]
Definition at line 44 of file TrackPAI.hh.

### m_electronDensity

double Garfield::TrackPAI::m_electronDensity = 0. [private]
Definition at line 62 of file TrackPAI.hh.

### m_emax

double Garfield::TrackPAI::m_emax = 0. [private]
Definition at line 39 of file TrackPAI.hh.

### m_energies

std::array<double, m_nSteps> Garfield::TrackPAI::m_energies [private]
Definition at line 53 of file TrackPAI.hh.

### m_eps1

std::array<double, m_nSteps> Garfield::TrackPAI::m_eps1 [private]
Definition at line 48 of file TrackPAI.hh.

### m_eps2

std::array<double, m_nSteps> Garfield::TrackPAI::m_eps2 [private]
Definition at line 49 of file TrackPAI.hh.

### m_epsInt

std::array<double, m_nSteps> Garfield::TrackPAI::m_epsInt [private]
Definition at line 50 of file TrackPAI.hh.

### m_imfp

double Garfield::TrackPAI::m_imfp = 0. [private]
Definition at line 42 of file TrackPAI.hh.

### m_mediumDensity

double Garfield::TrackPAI::m_mediumDensity = 0. [private]
Definition at line 61 of file TrackPAI.hh.

### m_mediumName

std::string Garfield::TrackPAI::m_mediumName = "" [private]
Definition at line 60 of file TrackPAI.hh.

### m_nSteps

size_t Garfield::TrackPAI::m_nSteps = 1000 [static], [constexpr], [private]
Definition at line 47 of file TrackPAI.hh.

### m_rutherford

std::array<double, m_nSteps> Garfield::TrackPAI::m_rutherford [private]
Definition at line 55 of file TrackPAI.hh.

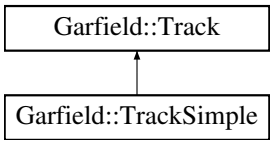
### m_speed

double Garfield::TrackPAI::m_speed = 0. [private]
Definition at line 37 of file TrackPAI.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackPAI.hh

## 7.114 Garfield::TrackSimple Class Reference

Generate tracks based on a cluster density given by the user.
#include <TrackSimple.hh>
Inheritance diagram for Garfield::TrackSimple:




Classes

- struct Cluster

Public Member Functions

- TrackSimple ()

  Default constructor.
- TrackSimple (Sensor *sensor)

  Constructor.
- virtual ~TrackSimple ()

  Destructor.
- void SetEqualSpacing ()

  Constant distance between clusters.
- void SetExponentialSpacing ()

  Exponentially distributed distance between clusters.
- void SetClusterDensity (const double d)

  Set the cluster density (inverse mean free path).
- double GetClusterDensity () override

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- void SetStoppingPower (const double dedx)

  Set the stopping power (dE/dx).
- double GetStoppingPower () override

  Get the stopping power (mean energy loss [eV] per cm).
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- virtual void SetParticle (const std::string &part)

  Set the type of charged particle.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

Return the speed ( $\beta = v/c$) of the projectile.

- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Protected Attributes

- double m_mfp = 0.04
- double m_eloss = 2530.
- bool m_useEqualSpacing = false
- std::vector< Cluster > m_clusters

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

Additional Inherited Members

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

### 7.114.1 Detailed Description

Generate tracks based on a cluster density given by the user.
Definition at line 12 of file TrackSimple.hh.

### 7.114.2 Constructor & Destructor Documentation

#### TrackSimple() [1/2]

Garfield::TrackSimple::TrackSimple () [inline]
Default constructor.
Definition at line 20 of file TrackSimple.hh.
00020 : TrackSimple(nullptr) {}

#### TrackSimple() [2/2]

Garfield::TrackSimple::TrackSimple (
Sensor * sensor)
Constructor.

#### ~TrackSimple()

virtual Garfield::TrackSimple::~TrackSimple () [inline], [virtual]
Destructor.
Definition at line 24 of file TrackSimple.hh.
00024 {}

### 7.114.3 Member Function Documentation

#### GetClusterDensity()

double Garfield::TrackSimple::GetClusterDensity () [override], [virtual]
Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
Reimplemented from Garfield::Track.

#### GetClusters()

const std::vector< Cluster > & Garfield::TrackSimple::GetClusters () const [inline]
Definition at line 41 of file TrackSimple.hh.
00041 { return m_clusters; }

#### GetStoppingPower()

double Garfield::TrackSimple::GetStoppingPower () [override], [virtual]
Get the stopping power (mean energy loss [eV] per cm).
Reimplemented from Garfield::Track.

#### NewTrack()

bool Garfield::TrackSimple::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]
Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

SetClusterDensity()

void Garfield::TrackSimple::SetClusterDensity (
const double d)
Set the cluster density (inverse mean free path).

SetEqualSpacing()

void Garfield::TrackSimple::SetEqualSpacing () [inline]
Constant distance between clusters.
Definition at line 27 of file TrackSimple.hh.
00027 { m_useEqualSpacing = true; }

SetExponentialSpacing()

void Garfield::TrackSimple::SetExponentialSpacing () [inline]
Exponentially distributed distance between clusters.
Definition at line 29 of file TrackSimple.hh.
00029 { m_useEqualSpacing = false; }

SetStoppingPower()

void Garfield::TrackSimple::SetStoppingPower (
const double dedx)
Set the stopping power (dE/dx).

### 7.114.4 Member Data Documentation

m_clusters

std::vector<Cluster> Garfield::TrackSimple::m_clusters [protected]
Definition at line 51 of file TrackSimple.hh.

m_eloss

double Garfield::TrackSimple::m_eloss = 2530. [protected]
Definition at line 47 of file TrackSimple.hh.

m_mfp

double Garfield::TrackSimple::m_mfp = 0.04 [protected]
Definition at line 45 of file TrackSimple.hh.

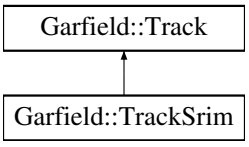
m_useEqualSpacing

bool Garfield::TrackSimple::m_useEqualSpacing = false [protected]
Definition at line 49 of file TrackSimple.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackSimple.hh

## 7.115 Garfield::TrackSrim Class Reference

Generate tracks based on SRIM energy loss, range and straggling tables.
#include <TrackSrim.hh>
Inheritance diagram for Garfield::TrackSrim:



Classes

- struct Cluster

Public Member Functions

- TrackSrim ()
  Default constructor.
- TrackSrim (Sensor *sensor)
  Constructor.
- virtual ~TrackSrim ()
  Destructor.
- bool ReadFile (const std::string &file)
  Load data from a SRIM file.
- void Print ()
  Print the energy loss table.
- void PlotEnergyLoss ()
  Plot the electromagnetic, hadronic, and total energy loss as function of the projectile energy.
- void PlotRange ()
  Plot the projected range as function of the projectile energy.
- void PlotStraggling ()
  Plot the transverse and longitudinal straggling as function of the projectile energy.
- void SetModel (const int m)
  Set the fluctuation model (0 = none, 1 = Landau, 2 = Vavilov, 3 = Gaussian, 4 = Combined).
- int GetModel () const
  Get the fluctuation model.
- void SetTargetClusterSize (const int n)
  Specify how many electrons should be grouped to a cluster.
- int GetTargetClusterSize () const
  Retrieve the target cluster size.
- void SetClustersMaximum (const int n)
  Set the max. number of clusters on a track.
- int GetClustersMaximum () const
  Retrieve the max. number of clusters on a track.
- void SetWorkFunction (const double w)
  Set the W value [eV].
- double GetWorkFunction () const
  Get the W value [eV].
- void SetFanoFactor (const double f)
  Set the Fano factor.
- void UnsetFanoFactor ()
  Use the default Fano factor.
- double GetFanoFactor () const
  Get the Fano factor.
- void SetDensity (const double density)
  Set the density [g/cm3] of the target medium.
- double GetDensity () const
  Get the density [g/cm3] of the target medium.
- void SetAtomicMassNumbers (const double a, const double z)
  Set A and Z of the target medium.
- void GetAtomicMassMumbers (double &a, double &z) const
  Get A and Z of the target medium.

- void EnableTransverseStraggling (const bool on=true)

  Simulate transverse straggling (default: on).
- void EnableLongitudinalStraggling (const bool on=true)

  Simulate longitudinal straggling (default: off).
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override

  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const
- bool GetCluster (double &xc, double &yc, double &zc, double &tc, int &nc, double &ec, double &extra)

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- virtual void SetParticle (const std::string &part)

  Set the type of charged particle.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- virtual double GetClusterDensity ()

Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).

- virtual double GetStoppingPower ()

  Get the stopping power (mean energy loss [eV] per cm).
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Protected Member Functions

- double Xi (const double x, const double beta2, const double edens) const
- double DedxEM (const double e) const
- double DedxHD (const double e) const
- bool PreciseLoss (const double step, const double estart, double &deem, double &dehd) const
- bool EstimateRange (const double ekin, const double step, double &stpmax) const
- bool SmallestStep (const double ekin, const double edens, double de, double step, double &stpmin)
- Medium * GetMedium (const std::array< double, 3 > &x) const
- double Terminate (const std::array< double, 3 > &x0, const std::array< double, 3 > &v0, const double step0) const
- double TerminateBfield (const std::array< double, 3 > &x0, const std::array< double, 3 > &v0, const double dt0, const double vmag) const
- double RndmEnergyLoss (const double ekin, const double de, const double step, const double edens) const

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Protected Attributes

- bool m_useTransStraggle = true

  Include transverse straggling.
- bool m_useLongStraggle = false

  Include longitudinal straggling.
- bool m_chargeset = false

  Has the charge been defined?
- double m_qion = 1.

  Charge of the projectile.
- double m_mion = -1.

  Mass [MeV] of the projectile.
- double m_rho = -1.

  Mass density [g/cm3] of the target.
- double m_work = -1.

  Work function [eV] of the target.
- bool m_fset = false

  Has the Fano factor been set?
- double m_fano = -1.

  Fano factor [-] of the target.
- double m_a = -1.

Effective A of the target.

- double m_z = -1.

  Effective Z of the target.
- int m_maxclusters = -1

  Maximum number of clusters allowed (infinite if 0)
- std::vector< double > m_ekin

  Energy in energy loss table [MeV].
- std::vector< double > m_emloss

  EM energy loss [MeV cm2/g].
- std::vector< double > m_hdloss

  Hadronic energy loss [MeV cm2/g].
- std::vector< double > m_range

  Projected range [cm].
- std::vector< double > m_transstraggle

  Transverse straggling [cm].
- std::vector< double > m_longstraggle

  Longitudinal straggling [cm].
- size_t m_currcluster = 0

  Index of the next cluster to be returned.
- unsigned int m_model = 4

  Fluctuation model (0 = none, 1 = Landau, 2 = Vavilov, 3 = Gaussian, 4 = Combined)
- int m_nsize = -1

  Targeted cluster size.
- std::vector< Cluster > m_clusters

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

Additional Inherited Members

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

### 7.115.1 Detailed Description

Generate tracks based on SRIM energy loss, range and straggling tables.

- http://www.srim.org

Definition at line 14 of file TrackSrim.hh.

### 7.115.2 Constructor & Destructor Documentation

#### TrackSrim() [1/2]

Garfield::TrackSrim::TrackSrim () [inline]
Default constructor.
Definition at line 17 of file TrackSrim.hh.

```
00017 : TrackSrim(nullptr) {}
```

#### TrackSrim() [2/2]

Garfield::TrackSrim::TrackSrim (
Sensor * sensor)
Constructor.

#### ~TrackSrim()

virtual Garfield::TrackSrim::~TrackSrim () [inline], [virtual]
Destructor.
Definition at line 21 of file TrackSrim.hh.

```
00021 {}
```

### 7.115.3 Member Function Documentation

#### DedxEM()

double Garfield::TrackSrim::DedxEM (
const double e) const [protected]

#### DedxHD()

double Garfield::TrackSrim::DedxHD (
const double e) const [protected]

#### EnableLongitudinalStraggling()

void Garfield::TrackSrim::EnableLongitudinalStraggling (
const bool on = true) [inline]
Simulate longitudinal straggling (default: off).
Definition at line 86 of file TrackSrim.hh.

```
00086                                                {
00087     m_useLongStraggle = on;
00088   }
```

#### EnableTransverseStraggling()

void Garfield::TrackSrim::EnableTransverseStraggling (
const bool on = true) [inline]
Simulate transverse straggling (default: on).
Definition at line 82 of file TrackSrim.hh.

```
00082                                                {
00083     m_useTransStraggle = on;
00084   }
```

#### EstimateRange()

bool Garfield::TrackSrim::EstimateRange (
const double ekin,
const double step,
double & stpmax) const [protected]

### GetAtomicMassMumbers()

void Garfield::TrackSrim::GetAtomicMassMumbers (
double & a,
double & z) const [inline]

Get A and Z of the target medium.

Definition at line 76 of file TrackSrim.hh.

```
00076                                                {
00077     a = m_a;
00078     z = m_z;
00079   }
```

### GetCluster()

bool Garfield::TrackSrim::GetCluster (
double & xc,
double & yc,
double & zc,
double & tc,
int & nc,
double & ec,
double & extra)

### GetClusters()

const std::vector< Cluster > & Garfield::TrackSrim::GetClusters () const [inline]

Definition at line 100 of file TrackSrim.hh.

```
00100 { return m_clusters; }
```

### GetClustersMaximum()

int Garfield::TrackSrim::GetClustersMaximum () const [inline]

Retrieve the max. number of clusters on a track.

Definition at line 51 of file TrackSrim.hh.

```
00051 { return m_maxclusters; }
```

### GetDensity()

double Garfield::TrackSrim::GetDensity () const [inline]

Get the density [g/cm3] of the target medium.

Definition at line 69 of file TrackSrim.hh.

```
00069 { return m_rho; }
```

### GetFanoFactor()

double Garfield::TrackSrim::GetFanoFactor () const [inline]

Get the Fano factor.

Definition at line 65 of file TrackSrim.hh.

```
00065 { return m_fano; }
```

### GetMedium()

Medium * Garfield::TrackSrim::GetMedium (
const std::array< double, 3 > & x) const [protected]

### GetModel()

int Garfield::TrackSrim::GetModel () const [inline]

Get the fluctuation model.

Definition at line 41 of file TrackSrim.hh.

```
00041 { return m_model; }
```

### GetTargetClusterSize()

int Garfield::TrackSrim::GetTargetClusterSize () const [inline]
Retrieve the target cluster size.
Definition at line 46 of file TrackSrim.hh.
00046 { return m_nsize; }

### GetWorkFunction()

double Garfield::TrackSrim::GetWorkFunction () const [inline]
Get the W value [eV].
Definition at line 56 of file TrackSrim.hh.
00056 { return m_work; }

### NewTrack()

bool Garfield::TrackSrim::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]
Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
Implements Garfield::Track.

### PlotEnergyLoss()

void Garfield::TrackSrim::PlotEnergyLoss ()
Plot the electromagnetic, hadronic, and total energy loss as function of the projectile energy.

### PlotRange()

void Garfield::TrackSrim::PlotRange ()
Plot the projected range as function of the projectile energy.

### PlotStraggling()

void Garfield::TrackSrim::PlotStraggling ()
Plot the transverse and longitudinal straggling as function of the projectile energy.

### PreciseLoss()

bool Garfield::TrackSrim::PreciseLoss (
const double step,
const double estart,
double & deem,
double & dehd) const [protected]

### Print()

void Garfield::TrackSrim::Print ()
Print the energy loss table.

### ReadFile()

bool Garfield::TrackSrim::ReadFile (
const std::string & file)
Load data from a SRIM file.

RndmEnergyLoss()

```
double Garfield::TrackSrim::RndmEnergyLoss (
                const double ekin,
                const double de,
                const double step,
                const double edens) const   [protected]
```

SetAtomicMassNumbers()

```
void Garfield::TrackSrim::SetAtomicMassNumbers (
                const double a,
                const double z)   [inline]
```

Set A and Z of the target medium.
Definition at line 71 of file TrackSrim.hh.

```
00071                                                 {
00072     m_a = a;
00073     m_z = z;
00074   }
```

SetClustersMaximum()

```
void Garfield::TrackSrim::SetClustersMaximum (
                const int n)   [inline]
```

Set the max. number of clusters on a track.
Definition at line 49 of file TrackSrim.hh.

```
00049 { m_maxclusters = n; }
```

SetDensity()

```
void Garfield::TrackSrim::SetDensity (
                const double density)   [inline]
```

Set the density [g/cm3] of the target medium.
Definition at line 67 of file TrackSrim.hh.

```
00067 { m_rho = density; }
```

SetFanoFactor()

```
void Garfield::TrackSrim::SetFanoFactor (
                const double f)   [inline]
```

Set the Fano factor.
Definition at line 58 of file TrackSrim.hh.

```
00058                                 {
00059     m_fano = f;
00060     m_fset = true;
00061   }
```

SetModel()

```
void Garfield::TrackSrim::SetModel (
                const int m)   [inline]
```

Set the fluctuation model (0 = none, 1 = Landau, 2 = Vavilov, 3 = Gaussian, 4 = Combined).
By default, the combined model (4) is used.
Definition at line 39 of file TrackSrim.hh.

```
00039 { m_model = m; }
```

SetTargetClusterSize()

```
void Garfield::TrackSrim::SetTargetClusterSize (
                const int n)   [inline]
```

Specify how many electrons should be grouped to a cluster.
Definition at line 44 of file TrackSrim.hh.

```
00044 { m_nsize = n; }
```

### SetWorkFunction()

void Garfield::TrackSrim::SetWorkFunction (
const double w) [inline]
Set the W value [eV].
Definition at line 54 of file TrackSrim.hh.
00054 { m_work = w; }

### SmallestStep()

bool Garfield::TrackSrim::SmallestStep (
const double ekin,
const double edens,
double de,
double step,
double & stpmin) [protected]

### Terminate()

double Garfield::TrackSrim::Terminate (
const std::array< double, 3 > & x0,
const std::array< double, 3 > & v0,
const double step0) const [protected]

### TerminateBfield()

double Garfield::TrackSrim::TerminateBfield (
const std::array< double, 3 > & x0,
const std::array< double, 3 > & v0,
const double dt0,
const double vmag) const [protected]

### UnsetFanoFactor()

void Garfield::TrackSrim::UnsetFanoFactor () [inline]
Use the default Fano factor.
Definition at line 63 of file TrackSrim.hh.
00063 { m_fset = false; }

### Xi()

double Garfield::TrackSrim::Xi (
const double x,
const double beta2,
const double edens) const [protected]

## 7.115.4 Member Data Documentation

### m_a

double Garfield::TrackSrim::m_a = -1. [protected]
Effective A of the target.
Definition at line 125 of file TrackSrim.hh.

### m_chargeset

bool Garfield::TrackSrim::m_chargeset = false [protected]
Has the charge been defined?
Definition at line 111 of file TrackSrim.hh.

m_clusters

std::vector<Cluster> Garfield::TrackSrim::m_clusters [protected]
Definition at line 151 of file TrackSrim.hh.

m_currcluster

size_t Garfield::TrackSrim::m_currcluster = 0 [protected]
Index of the next cluster to be returned.
Definition at line 145 of file TrackSrim.hh.

m_ekin

std::vector<double> Garfield::TrackSrim::m_ekin [protected]
Energy in energy loss table [MeV].
Definition at line 132 of file TrackSrim.hh.

m_emloss

std::vector<double> Garfield::TrackSrim::m_emloss [protected]
EM energy loss [MeV cm2/g].
Definition at line 134 of file TrackSrim.hh.

m_fano

double Garfield::TrackSrim::m_fano = -1. [protected]
Fano factor [-] of the target.
Definition at line 123 of file TrackSrim.hh.

m_fset

bool Garfield::TrackSrim::m_fset = false [protected]
Has the Fano factor been set?
Definition at line 121 of file TrackSrim.hh.

m_hdloss

std::vector<double> Garfield::TrackSrim::m_hdloss [protected]
Hadronic energy loss [MeV cm2/g].
Definition at line 136 of file TrackSrim.hh.

m_longstraggle

std::vector<double> Garfield::TrackSrim::m_longstraggle [protected]
Longitudinal straggling [cm].
Definition at line 142 of file TrackSrim.hh.

m_maxclusters

int Garfield::TrackSrim::m_maxclusters = -1 [protected]
Maximum number of clusters allowed (infinite if 0)
Definition at line 130 of file TrackSrim.hh.

m_mion

double Garfield::TrackSrim::m_mion = -1. [protected]
Mass [MeV] of the projectile.
Definition at line 115 of file TrackSrim.hh.

m_model

unsigned int Garfield::TrackSrim::m_model = 4 [protected]
Fluctuation model (0 = none, 1 = Landau, 2 = Vavilov, 3 = Gaussian, 4 = Combined)
Definition at line 148 of file TrackSrim.hh.

m_nsize

int Garfield::TrackSrim::m_nsize = -1 [protected]
Targeted cluster size.
Definition at line 150 of file TrackSrim.hh.

m_qion

double Garfield::TrackSrim::m_qion = 1. [protected]
Charge of the projectile.
Definition at line 113 of file TrackSrim.hh.

m_range

std::vector<double> Garfield::TrackSrim::m_range [protected]
Projected range [cm].
Definition at line 138 of file TrackSrim.hh.

m_rho

double Garfield::TrackSrim::m_rho = -1. [protected]
Mass density [g/cm3] of the target.
Definition at line 117 of file TrackSrim.hh.

m_transstraggle

std::vector<double> Garfield::TrackSrim::m_transstraggle [protected]
Transverse straggling [cm].
Definition at line 140 of file TrackSrim.hh.

m_useLongStraggle

bool Garfield::TrackSrim::m_useLongStraggle = false [protected]
Include longitudinal straggling.
Definition at line 108 of file TrackSrim.hh.

m_useTransStraggle

bool Garfield::TrackSrim::m_useTransStraggle = true [protected]
Include transverse straggling.
Definition at line 106 of file TrackSrim.hh.

m_work

double Garfield::TrackSrim::m_work = -1. [protected]
Work function [eV] of the target.
Definition at line 119 of file TrackSrim.hh.

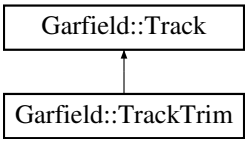
m_z

double Garfield::TrackSrim::m_z = -1. [protected]
Effective Z of the target.
Definition at line 127 of file TrackSrim.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackSrim.hh

## 7.116 Garfield::TrackTrim Class Reference

Generate tracks based on TRIM output files.
#include <TrackTrim.hh>
Inheritance diagram for Garfield::TrackTrim:




Classes

- struct Cluster

Public Member Functions

- TrackTrim ()
  Default constructor.
- TrackTrim (Sensor *sensor)
  Constructor.
- virtual ~TrackTrim ()
  Destructor.
- bool ReadFile (const std::string &file, const unsigned int nIons=0, const unsigned int nSkip=0)
  Load data from an EXYZ.txt file.
- void Print ()
  Print a summary of the available TRIM data.
- void SetCharge (const double q)
  Set the projectile charge [-].
- void SetWorkFunction (const double w)
  Set the W value [eV].
- double GetWorkFunction () const
  Get the W value [eV].
- void SetFanoFactor (const double f)
  Set the Fano factor.
- void UnsetFanoFactor ()
  Use the default Fano factor.
- double GetFanoFactor () const
  Get the Fano factor.
- void EnableStepSizeLimit (const double dmax)
  Set a max. distance between subsequent steps.
- void DisableStepSizeLimit ()
  Do not reduce the steps with respect to the TRIM input file (default).
- void EnableEnergyLossLimit (const double emax)
  Set a max. energy loss per cluster.
- void DisableEnergyLossLimit ()
  Do not limit the energy loss per cluster (default).
- void SetParticle (const std::string &part) override
  Set the type of charged particle.
- bool NewTrack (const double x0, const double y0, const double z0, const double t0, const double dx0, const double dy0, const double dz0) override
  Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).
- const std::vector< Cluster > & GetClusters () const
- bool GetCluster (double &xc, double &yc, double &zc, double &tc, int &nc, double &ec, double &extra)

Public Member Functions inherited from Garfield::Track

- Track ()=delete

  Default constructor.
- Track (const std::string &name)

  Constructor.
- virtual ~Track ()

  Destructor.
- void SetEnergy (const double e)

  Set the particle energy.
- void SetBetaGamma (const double bg)

  Set the relative momentum of the particle.
- void SetBeta (const double beta)

  Set the speed ( $\beta = v/c$) of the particle.
- void SetGamma (const double gamma)

  Set the Lorentz factor of the particle.
- void SetMomentum (const double p)

  Set the particle momentum.
- void SetKineticEnergy (const double ekin)

  Set the kinetic energy of the particle.
- double GetEnergy () const

  Return the particle energy.
- double GetBetaGamma () const

  Return the $\beta\gamma$ of the projectile.
- double GetBeta () const

  Return the speed ( $\beta = v/c$) of the projectile.
- double GetGamma () const

  Return the Lorentz factor of the projectile.
- double GetMomentum () const

  Return the particle momentum.
- double GetKineticEnergy () const

  Return the kinetic energy of the projectile.
- double GetCharge () const

  Get the charge of the projectile.
- double GetMass () const

  Get the mass [eV / c2] of the projectile.
- void SetSensor (Sensor *s)

  Set the sensor through which to transport the particle.
- virtual double GetClusterDensity ()

  Get the cluster density (number of ionizing collisions per cm or inverse mean free path for ionization).
- virtual double GetStoppingPower ()

  Get the stopping power (mean energy loss [eV] per cm).
- void EnablePlotting (ViewDrift *viewer)

  Switch on plotting.
- void DisablePlotting ()

  Switch off plotting.
- void EnableDebugging ()

  Switch on debugging messages.
- void DisableDebugging ()

  Switch off debugging messages.

Protected Member Functions

- void AddIon (const std::vector< float > &x, const std::vector< float > &y, const std::vector< float > &z, const std::vector< float > &dedx, const std::vector< float > &ekin)

Protected Member Functions inherited from Garfield::Track

- void PlotNewTrack (const double x0, const double y0, const double z0)
- void PlotCluster (const double x0, const double y0, const double z0)

Protected Attributes

- double m_work = -1.

  Work function [eV] of the target.
- bool m_fset = false

  Has the Fano factor been set?
- double m_fano = -1.

  Fano factor [-] of the target.
- double m_ekin = 0.

  Projectile energy [eV].
- std::vector< std::vector< std::array< float, 6 > > > m_ions

  List of tracks.
- size_t m_ion = 0

  Index of the current track.
- std::vector< Cluster > m_clusters

  Clusters on the current track.
- size_t m_cluster = 0

  Index of the next cluster to be returned.
- double m_maxStepSize = -1.

  Step size limit.
- double m_maxLossPerStep = -1.

  Energy loss limit per step.

Protected Attributes inherited from Garfield::Track

- std::string m_className = "Track"
- double m_q = -1.
- int m_spin = 1
- double m_mass
- double m_energy = 0.
- double m_beta2 = 1.
- bool m_isElectron = false
- std::string m_particleName = "mu-"
- Sensor * m_sensor = nullptr
- bool m_isChanged = true
- ViewDrift * m_viewer = nullptr
- bool m_debug = false
- size_t m_plotId = 0

Additional Inherited Members

Static Protected Member Functions inherited from Garfield::Track

- static std::array< double, 3 > StepBfield (const double dt, const double qoverm, const double vmag, double bx, double by, double bz, std::array< double, 3 > &dir)

### 7.116.1 Detailed Description

Generate tracks based on TRIM output files.

- http://www.srim.org

Definition at line 15 of file TrackTrim.hh.

### 7.116.2 Constructor & Destructor Documentation

#### TrackTrim() [1/2]

Garfield::TrackTrim::TrackTrim () [inline]
Default constructor.
Definition at line 18 of file TrackTrim.hh.
00018 : TrackTrim(nullptr) {}

#### TrackTrim() [2/2]

Garfield::TrackTrim::TrackTrim (
Sensor * sensor)
Constructor.

#### ~TrackTrim()

virtual Garfield::TrackTrim::~TrackTrim () [inline], [virtual]
Destructor.
Definition at line 22 of file TrackTrim.hh.
00022 {}

### 7.116.3 Member Function Documentation

#### AddIon()

void Garfield::TrackTrim::AddIon (
const std::vector< float > & x,
const std::vector< float > & y,
const std::vector< float > & z,
const std::vector< float > & dedx,
const std::vector< float > & ekin) [protected]

#### DisableEnergyLossLimit()

void Garfield::TrackTrim::DisableEnergyLossLimit () [inline]
Do not limit the energy loss per cluster (default).
Each segment in the TRIM input file corresponds to one cluster.
Definition at line 56 of file TrackTrim.hh.
00056 { m_maxLossPerStep = -1.; }

#### DisableStepSizeLimit()

void Garfield::TrackTrim::DisableStepSizeLimit () [inline]
Do not reduce the steps with respect to the TRIM input file (default).
Definition at line 51 of file TrackTrim.hh.
00051 { m_maxStepSize = -1.; }

#### EnableEnergyLossLimit()

void Garfield::TrackTrim::EnableEnergyLossLimit (
const double emax) [inline]
Set a max. energy loss per cluster.
Definition at line 53 of file TrackTrim.hh.
00053 { m_maxLossPerStep = emax; }

### EnableStepSizeLimit()

void Garfield::TrackTrim::EnableStepSizeLimit (
const double dmax) [inline]

Set a max. distance between subsequent steps.

Definition at line 49 of file TrackTrim.hh.

```
00049 { m_maxStepSize = dmax; }
```

### GetCluster()

bool Garfield::TrackTrim::GetCluster (
double & xc,
double & yc,
double & zc,
double & tc,
int & nc,
double & ec,
double & extra)

### GetClusters()

const std::vector< Cluster > & Garfield::TrackTrim::GetClusters () const [inline]

Definition at line 71 of file TrackTrim.hh.

```
00071 { return m_clusters; }
```

### GetFanoFactor()

double Garfield::TrackTrim::GetFanoFactor () const [inline]

Get the Fano factor.

Definition at line 46 of file TrackTrim.hh.

```
00046 { return m_fano; }
```

### GetWorkFunction()

double Garfield::TrackTrim::GetWorkFunction () const [inline]

Get the W value [eV].

Definition at line 37 of file TrackTrim.hh.

```
00037 { return m_work; }
```

### NewTrack()

bool Garfield::TrackTrim::NewTrack (
const double x0,
const double y0,
const double z0,
const double t0,
const double dx0,
const double dy0,
const double dz0) [override], [virtual]

Calculate a new track starting from (x0, y0, z0) at time t0 in direction (dx0, dy0, dz0).

Implements Garfield::Track.

### Print()

void Garfield::TrackTrim::Print ()

Print a summary of the available TRIM data.

### ReadFile()

bool Garfield::TrackTrim::ReadFile (
const std::string & file,

const unsigned int nIons = 0,
const unsigned int nSkip = 0)

Load data from an EXYZ.txt file.

### SetCharge()

void Garfield::TrackTrim::SetCharge (
const double q) [inline]

Set the projectile charge [-].

Definition at line 31 of file TrackTrim.hh.

```
00031                                {
00032     if (fabs(q) > 0.) m_q = q;
00033   }
```

### SetFanoFactor()

void Garfield::TrackTrim::SetFanoFactor (
const double f) [inline]

Set the Fano factor.

Definition at line 39 of file TrackTrim.hh.

```
00039                                    {
00040     m_fano = f;
00041     m_fset = true;
00042   }
```

### SetParticle()

void Garfield::TrackTrim::SetParticle (
const std::string & part) [override], [virtual]

Set the type of charged particle.

- electron,e-
- positron,e+
- muon,mu-
- mu+
- pion,pi-
- pi+
- kaon,K-
- K+
- proton,p
- anti-proton,p-bar
- deuteron,d
- alpha

Reimplemented from Garfield::Track.

### SetWorkFunction()

void Garfield::TrackTrim::SetWorkFunction (
const double w) [inline]

Set the W value [eV].

Definition at line 35 of file TrackTrim.hh.

```
00035 { m_work = w; }
```

UnsetFanoFactor()

void Garfield::TrackTrim::UnsetFanoFactor () [inline]
Use the default Fano factor.
Definition at line 44 of file TrackTrim.hh.
00044 { m_fset = false; }

### 7.116.4 Member Data Documentation

m_cluster

size_t Garfield::TrackTrim::m_cluster = 0 [protected]
Index of the next cluster to be returned.
Definition at line 93 of file TrackTrim.hh.

m_clusters

std::vector<Cluster> Garfield::TrackTrim::m_clusters [protected]
Clusters on the current track.
Definition at line 91 of file TrackTrim.hh.

m_ekin

double Garfield::TrackTrim::m_ekin = 0. [protected]
Projectile energy [eV].
Definition at line 84 of file TrackTrim.hh.

m_fano

double Garfield::TrackTrim::m_fano = -1. [protected]
Fano factor [-] of the target.
Definition at line 81 of file TrackTrim.hh.

m_fset

bool Garfield::TrackTrim::m_fset = false [protected]
Has the Fano factor been set?
Definition at line 79 of file TrackTrim.hh.

m_ion

size_t Garfield::TrackTrim::m_ion = 0 [protected]
Index of the current track.
Definition at line 88 of file TrackTrim.hh.

m_ions

std::vector<std::vector<std::array<float, 6> > > Garfield::TrackTrim::m_ions [protected]
List of tracks.
Definition at line 86 of file TrackTrim.hh.

m_maxLossPerStep

double Garfield::TrackTrim::m_maxLossPerStep = -1. [protected]
Energy loss limit per step.
Definition at line 98 of file TrackTrim.hh.

m_maxStepSize

double Garfield::TrackTrim::m_maxStepSize = -1. [protected]
Step size limit.
Definition at line 96 of file TrackTrim.hh.

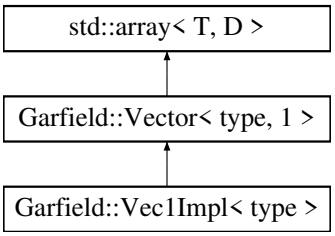
m_work

double Garfield::TrackTrim::m_work = -1. [protected]
Work function [eV] of the target.
Definition at line 77 of file TrackTrim.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/TrackTrim.hh

## 7.117 Garfield::Vec1Impl< type > Class Template Reference

#include <Vector.hh>
Inheritance diagram for Garfield::Vec1Impl< type >:




Public Member Functions

- Vec1Impl ()=default
- template<typename... xs>
  ___host___ ___device___ Vec1Impl (xs... values)
- ___host___ ___device___ type & x () noexcept
- ___host___ ___device___ type x () const noexcept

Public Member Functions inherited from Garfield::Vector< type, 1 >

- ___host___ ___device___ Vector (Args &&... args)
- ___host___ ___device___ Vector operator* (const double &i)
- ___host___ ___device___ Vector operator/ (const double &i)
- ___host___ ___device___ Vector operator+ (const Vector< T, D > &vec)
- ___host___ ___device___ Vector operator- (const Vector< T, D > &vec)

### 7.117.1 Detailed Description

template<typename type>
class Garfield::Vec1Impl< type >

Definition at line 63 of file Vector.hh.

### 7.117.2 Constructor & Destructor Documentation

Vec1Impl() [1/2]

template<typename type>
Garfield::Vec1Impl< type >::Vec1Impl () [default]

Vec1Impl() [2/2]

template<typename type>
template<typename... xs>
___host___ ___device___ Garfield::Vec1Impl< type >::Vec1Impl (
xs... values) [inline]
Definition at line 67 of file Vector.hh.
00067 : Vector<type, 3>({values...}) {}

### 7.117.3 Member Function Documentation

x() [1/2]

template<typename type>
___host___ ___device___ type Garfield::Vec1Impl< type >::x () const [inline], [noexcept]
Definition at line 69 of file Vector.hh.
00069 { return this->operator[](0); }

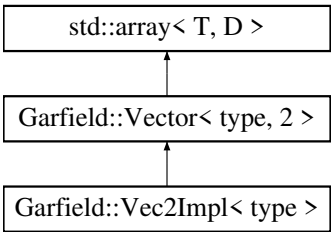
x() [2/2]

template<typename type>
___host___ ___device___ type & Garfield::Vec1Impl< type >::x () [inline], [noexcept]
Definition at line 68 of file Vector.hh.
00068 { return this->operator[](0); }
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Vector.hh

## 7.118 Garfield::Vec2Impl< type > Class Template Reference

#include <Vector.hh>
Inheritance diagram for Garfield::Vec2Impl< type >:




Public Member Functions

- Vec2Impl ()=default
- template<typename... xs>
  ___host___ ___device___ Vec2Impl (xs... values)
- ___host___ ___device___ type & x () noexcept
- ___host___ ___device___ type x () const noexcept
- ___host___ ___device___ type & y () noexcept
- ___host___ ___device___ type y () const noexcept

Public Member Functions inherited from Garfield::Vector< type, 2 >

- ___host___ ___device___ Vector (Args &&... args)
- ___host___ ___device___ Vector operator* (const double &i)
- ___host___ ___device___ Vector operator/ (const double &i)
- ___host___ ___device___ Vector operator+ (const Vector< T, D > &vec)
- ___host___ ___device___ Vector operator- (const Vector< T, D > &vec)

### 7.118.1 Detailed Description

template<typename type>
class Garfield::Vec2Impl< type >

Definition at line 73 of file Vector.hh.

### 7.118.2 Constructor & Destructor Documentation

#### Vec2Impl() [1/2]

```
template<typename type>
Garfield::Vec2Impl< type >::Vec2Impl ()  [default]
```

#### Vec2Impl() [2/2]

```
template<typename type>
template<typename... xs>
___host___ ___device___ Garfield::Vec2Impl< type >::Vec2Impl (
                xs... values)  [inline]
```

Definition at line 77 of file Vector.hh.

```
00077 : Vector<type, 3>({values...}) {}
```

### 7.118.3 Member Function Documentation

#### x() [1/2]

```
template<typename type>
___host___ ___device___ type Garfield::Vec2Impl< type >::x () const  [inline], [noexcept]
```

Definition at line 79 of file Vector.hh.

```
00079 { return this->operator[](0); }
```

#### x() [2/2]

```
template<typename type>
___host___ ___device___ type & Garfield::Vec2Impl< type >::x ()  [inline], [noexcept]
```

Definition at line 78 of file Vector.hh.

```
00078 { return this->operator[](0); }
```

#### y() [1/2]

```
template<typename type>
___host___ ___device___ type Garfield::Vec2Impl< type >::y () const  [inline], [noexcept]
```

Definition at line 81 of file Vector.hh.

```
00081 { return this->operator[](1); }
```

#### y() [2/2]

```
template<typename type>
___host___ ___device___ type & Garfield::Vec2Impl< type >::y ()  [inline], [noexcept]
```

Definition at line 80 of file Vector.hh.

```
00080 { return this->operator[](1); }
```

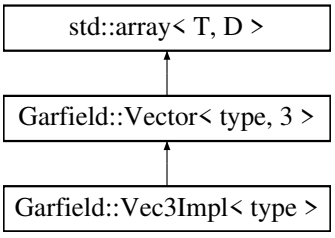
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Vector.hh

## 7.119 Garfield::Vec3Impl< type > Class Template Reference

#include <Vector.hh>

Inheritance diagram for Garfield::Vec3Impl< type >:



Public Member Functions

- Vec3Impl ()=default
- template<typename... xs>
  ___host___ ___device___ Vec3Impl (xs... values)
- ___host___ ___device___ type & x () noexcept
- ___host___ ___device___ type x () const noexcept
- ___host___ ___device___ type & y () noexcept
- ___host___ ___device___ type y () const noexcept
- ___host___ ___device___ type & z () noexcept
- ___host___ ___device___ type z () const noexcept

Public Member Functions inherited from Garfield::Vector< type, 3 >

- ___host___ ___device___ Vector (Args &&... args)
- ___host___ ___device___ Vector operator* (const double &i)
- ___host___ ___device___ Vector operator/ (const double &i)
- ___host___ ___device___ Vector operator+ (const Vector< T, D > &vec)
- ___host___ ___device___ Vector operator- (const Vector< T, D > &vec)

### 7.119.1 Detailed Description

template<typename type>
class Garfield::Vec3Impl< type >

Definition at line 85 of file Vector.hh.

### 7.119.2 Constructor & Destructor Documentation

#### Vec3Impl() [1/2]

template<typename type>
Garfield::Vec3Impl< type >::Vec3Impl () [default]

#### Vec3Impl() [2/2]

template<typename type>
template<typename... xs>
___host___ ___device___ Garfield::Vec3Impl< type >::Vec3Impl (
xs... values) [inline]

Definition at line 89 of file Vector.hh.

```
00089 : Vector<type, 3>({values...}) {}
```

### 7.119.3 Member Function Documentation

#### x() [1/2]

template<typename type>
___host___ ___device___ type Garfield::Vec3Impl< type >::x () const [inline], [noexcept]

Definition at line 91 of file Vector.hh.

```
00091 { return this->operator[](0); }
```

#### x() [2/2]

template<typename type>
___host___ ___device___ type & Garfield::Vec3Impl< type >::x () [inline], [noexcept]

Definition at line 90 of file Vector.hh.

```
00090 { return this->operator[](0); }
```

y() [1/2]

template<typename type>
___host___ ___device___ type Garfield::Vec3Impl< type >::y () const [inline], [noexcept]
Definition at line 93 of file Vector.hh.
00093 { return this->operator[](1); }

y() [2/2]

template<typename type>
___host___ ___device___ type & Garfield::Vec3Impl< type >::y () [inline], [noexcept]
Definition at line 92 of file Vector.hh.
00092 { return this->operator[](1); }

z() [1/2]

template<typename type>
___host___ ___device___ type Garfield::Vec3Impl< type >::z () const [inline], [noexcept]
Definition at line 95 of file Vector.hh.
00095 { return this->operator[](2); }

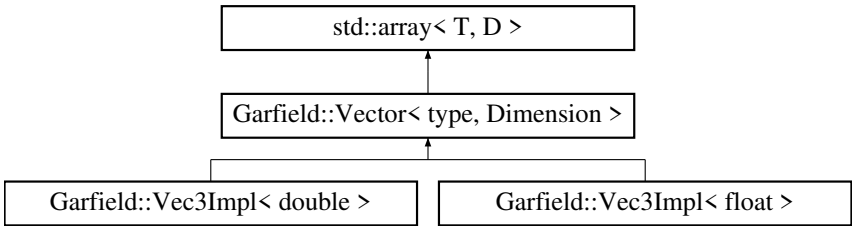
z() [2/2]

template<typename type>
___host___ ___device___ type & Garfield::Vec3Impl< type >::z () [inline], [noexcept]
Definition at line 94 of file Vector.hh.
00094 { return this->operator[](2); }
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Vector.hh

## 7.120 Garfield::Vector< type, Dimension > Class Template Reference

#include <Vector.hh>
Inheritance diagram for Garfield::Vector< type, Dimension >:




Public Member Functions

- template<typename... Args>
  ___host___ ___device___ Vector (Args &&... args)
- ___host___ ___device___ Vector ()
- ___host___ ___device___ Vector operator* (const double &i)
- ___host___ ___device___ Vector operator/ (const double &i)
- template<typename T, std::size_t D>
  ___host___ ___device___ Vector operator+ (const Vector< T, D > &vec)
- template<typename T, std::size_t D>
  ___host___ ___device___ Vector operator- (const Vector< T, D > &vec)

### 7.120.1 Detailed Description

template<typename type, std::size_t Dimension>
class Garfield::Vector< type, Dimension >

Definition at line 22 of file Vector.hh.

### 7.120.2 Constructor & Destructor Documentation

#### Vector() [1/2]

template<typename type, std::size_t Dimension>
template<typename... Args>
___host___ ___device___ Garfield::Vector< type, Dimension >::Vector (
Args &&... args) [inline]

Definition at line 25 of file Vector.hh.

```
00026       : GarfieldArray<type, Dimension>({std::forward<Args>(args)...}) {}
```

#### Vector() [2/2]

template<typename type, std::size_t Dimension>
___host___ ___device___ Garfield::Vector< type, Dimension >::Vector () [inline]

Definition at line 27 of file Vector.hh.

```
00027 { this->fill(0.); }
```

### 7.120.3 Member Function Documentation

#### operator*()

template<typename type, std::size_t Dimension>
___host___ ___device___ Vector Garfield::Vector< type, Dimension >::operator* (
const double & i) [inline]

Definition at line 28 of file Vector.hh.

```
00028                                                 {
00029     Vector ret = *this;
00030     for (std::size_t j = 0; j != ret.size(); ++j) {
00031       ret[j] *= i;
00032     }
00033     return ret;
00034   }
```

#### operator+()

template<typename type, std::size_t Dimension>
template<typename T, std::size_t D>
___host___ ___device___ Vector Garfield::Vector< type, Dimension >::operator+ (
const Vector< T, D > & vec) [inline]

Definition at line 43 of file Vector.hh.

```
00043                                                     {
00044     static_assert(Dimension == D, "The vectors don't have the same dimension!");
00045     Vector ret = *this;
00046     for (std::size_t j = 0; j != ret.size(); ++j) {
00047       ret[j] += vec[j];
00048     }
00049     return ret;
00050   }
```

#### operator-()

template<typename type, std::size_t Dimension>
template<typename T, std::size_t D>
___host___ ___device___ Vector Garfield::Vector< type, Dimension >::operator- (
const Vector< T, D > & vec) [inline]

Definition at line 52 of file Vector.hh.

```
00052                                                     {
00053     static_assert(Dimension == D, "The vectors don't have the same dimension!");
00054     Vector ret = *this;
```

```
00055     for (std::size_t j = 0; j != ret.size(); ++j) {
00056       ret[j] -= vec[j];
00057     }
00058     return ret;
00059   }
```

operator/()

```
template<typename type, std::size_t Dimension>
___host___ ___device___ Vector Garfield::Vector< type, Dimension >::operator/ (
                const double & i)   [inline]
```

Definition at line 35 of file Vector.hh.

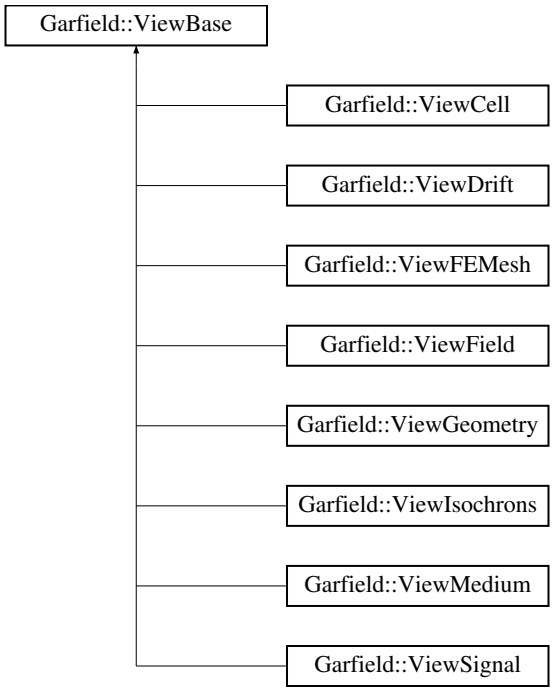
```
00035                                                 {
00036     Vector ret = *this;
00037     for (std::size_t j = 0; j != ret.size(); ++j) {
00038       ret[j] /= i;
00039     }
00040     return ret;
00041   }
```

The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/Vector.hh

## 7.121 Garfield::ViewBase Class Reference

Base class for visualization classes.
#include <ViewBase.hh>
Inheritance diagram for Garfield::ViewBase:




Public Member Functions

- ViewBase ()=delete
  Default constructor.
- ViewBase (const std::string &name)
  Constructor.
- virtual ~ViewBase ()=default
  Destructor.
- void SetCanvas (TPad *pad)

Set the canvas to be painted on.

- void SetCanvas ()

  Unset an external canvas.

- TPad * GetCanvas ()

  Retrieve the canvas.

- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).

- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).

- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).

- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.

- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.

- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).

- void SetPlaneXY ()

  Set the viewing plane to x-y.

- void SetPlaneXZ ()

  Set the viewing plane to x-z.

- void SetPlaneYZ ()

  Set the viewing plane to y-z.

- void SetPlaneZX ()

  Set the viewing plane to z-x.

- void SetPlaneZY ()

  Set the viewing plane to z-y.

- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Static Public Member Functions

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.

- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.

- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()

- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

Private Attributes

- TPad * m_pad = nullptr
- std::unique_ptr< TCanvas > m_canvas

### 7.121.1 Detailed Description

Base class for visualization classes.
Definition at line 21 of file ViewBase.hh.

### 7.121.2 Constructor & Destructor Documentation

ViewBase() [1/2]

Garfield::ViewBase::ViewBase () [delete]
Default constructor.

ViewBase() [2/2]

Garfield::ViewBase::ViewBase (
const std::string & name)
Constructor.

### ~ViewBase()

virtual Garfield::ViewBase::~ViewBase () [virtual], [default]
Destructor.

## 7.121.3 Member Function Documentation

### Clip()

void Garfield::ViewBase::Clip (
const std::array< float, 3 > & x0,
const std::array< float, 3 > & x1,
std::array< float, 3 > & xc) const [protected]

### DrawLine()

void Garfield::ViewBase::DrawLine (
const std::vector< std::array< float, 3 > > & xl,
const short col,
const short lw) [protected]

### EnableDebugging()

void Garfield::ViewBase::EnableDebugging (
const bool on = true) [inline]
Switch on/off debugging output.
Definition at line 75 of file ViewBase.hh.
00075 { m_debug = on; }

### FindUnusedCanvasName()

std::string Garfield::ViewBase::FindUnusedCanvasName (
const std::string & s) [static]
Find an unused canvas name.

### FindUnusedFunctionName()

std::string Garfield::ViewBase::FindUnusedFunctionName (
const std::string & s) [static]
Find an unused function name.

### FindUnusedHistogramName()

std::string Garfield::ViewBase::FindUnusedHistogramName (
const std::string & s) [static]
Find an unused histogram name.

### GetCanvas()

TPad * Garfield::ViewBase::GetCanvas ()
Retrieve the canvas.

### InBox()

template<typename T>
bool Garfield::ViewBase::InBox (
const std::array< T, 3 > & x) const [inline], [protected]
Definition at line 120 of file ViewBase.hh.

```
00120                                       {
00121     if (!m_userBox) return true;
00122     if (x[0] < m_xMinBox || x[0] > m_xMaxBox || x[1] < m_yMinBox ||
00123         x[1] > m_yMaxBox || x[2] < m_yMinBox || x[2] > m_zMaxBox)
00124       return false;
```

```
00125     return true;
00126   }
```

### LabelX()

std::string Garfield::ViewBase::LabelX () [protected]

### LabelY()

std::string Garfield::ViewBase::LabelY () [protected]

### PlaneDescription()

std::string Garfield::ViewBase::PlaneDescription () [protected]

### PlotLimits() [1/3]

```
bool Garfield::ViewBase::PlotLimits (
                Component * cmp,
                double & xmin,
                double & ymin,
                double & xmax,
                double & ymax) const   [protected]
```

### PlotLimits() [2/3]

```
bool Garfield::ViewBase::PlotLimits (
                Sensor * sensor,
                double & xmin,
                double & ymin,
                double & xmax,
                double & ymax) const   [protected]
```

### PlotLimits() [3/3]

```
bool Garfield::ViewBase::PlotLimits (
                std::array< double, 3 > & bbmin,
                std::array< double, 3 > & bbmax,
                double & xmin,
                double & ymin,
                double & xmax,
                double & ymax) const   [protected]
```

### PlotLimitsFromUserBox()

```
bool Garfield::ViewBase::PlotLimitsFromUserBox (
                double & xmin,
                double & ymin,
                double & xmax,
                double & ymax) const   [protected]
```

### RangeSet()

```
bool Garfield::ViewBase::RangeSet (
                TVirtualPad * )   [static], [protected]
```

### Rotate()

```
void Garfield::ViewBase::Rotate (
                const double angle)
```

Rotate the viewing plane (angle in radian).

SetArea() [1/3]

void Garfield::ViewBase::SetArea () [inline]
Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
Definition at line 46 of file ViewBase.hh.

```
00046                 {
00047     m_userBox = false;
00048     m_userPlotLimits = false;
00049   }
```

SetArea() [2/3]

void Garfield::ViewBase::SetArea (
const double xmin,
const double ymin,
const double xmax,
const double ymax)
Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).

SetArea() [3/3]

virtual void Garfield::ViewBase::SetArea (
const double xmin,
const double ymin,
const double zmin,
const double xmax,
const double ymax,
const double zmax) [virtual]
Set a bounding box (if applicable).

SetCanvas() [1/2]

void Garfield::ViewBase::SetCanvas () [inline]
Unset an external canvas.
Definition at line 33 of file ViewBase.hh.

```
00033 { m_pad = nullptr; }
```

SetCanvas() [2/2]

void Garfield::ViewBase::SetCanvas (
TPad * pad) [inline]
Set the canvas to be painted on.
Definition at line 31 of file ViewBase.hh.

```
00031 { m_pad = pad; }
```

SetPlane() [1/2]

virtual void Garfield::ViewBase::SetPlane (
const double fx,
const double fy,
const double fz,
const double x0,
const double y0,
const double z0) [virtual]
Set the projection (viewing plane), if applicable.

Parameters

| | |
|---|---|
| fx,fy,fz | normal vector |
| x0,y0,z0 | in-plane point |

Reimplemented in Garfield::ViewFEMesh.

SetPlane() [2/2]

virtual void Garfield::ViewBase::SetPlane (
const double fx,
const double fy,
const double fz,
const double x0,
const double y0,
const double z0,
const double hx,
const double hy,
const double hz) [virtual]

Set the projection plane specifying a hint for the in-plane x axis.
Reimplemented in Garfield::ViewFEMesh.

SetPlaneXY()

void Garfield::ViewBase::SetPlaneXY ()
Set the viewing plane to x-y.

SetPlaneXZ()

void Garfield::ViewBase::SetPlaneXZ ()
Set the viewing plane to x-z.

SetPlaneYZ()

void Garfield::ViewBase::SetPlaneYZ ()
Set the viewing plane to y-z.

SetPlaneZX()

void Garfield::ViewBase::SetPlaneZX ()
Set the viewing plane to z-x.

SetPlaneZY()

void Garfield::ViewBase::SetPlaneZY ()
Set the viewing plane to z-y.

SetRange()

void Garfield::ViewBase::SetRange (
TVirtualPad * pad,
const double x0,
const double y0,
const double x1,
const double y1) [static], [protected]

ToPlane()

template<typename T>
void Garfield::ViewBase::ToPlane (
const T x,
const T y,
const T z,

T & xp,
T & yp) const [inline], [protected]

Definition at line 114 of file ViewBase.hh.

```
00114                                                                 {
00115     xp = m_prmat[0][0] * x + m_prmat[0][1] * y + m_prmat[0][2] * z;
00116     yp = m_prmat[1][0] * x + m_prmat[1][1] * y + m_prmat[1][2] * z;
00117   }
```

### UpdateProjectionMatrix()

void Garfield::ViewBase::UpdateProjectionMatrix () [protected]

## 7.121.4 Member Data Documentation

### m_canvas

std::unique_ptr<TCanvas> Garfield::ViewBase::m_canvas [private]

Definition at line 157 of file ViewBase.hh.

### m_className

std::string Garfield::ViewBase::m_className = "ViewBase" [protected]

Definition at line 85 of file ViewBase.hh.

### m_debug

bool Garfield::ViewBase::m_debug = false [protected]

Definition at line 88 of file ViewBase.hh.

### m_pad

TPad* Garfield::ViewBase::m_pad = nullptr [private]

Definition at line 156 of file ViewBase.hh.

### m_plane

std::array<double, 4> Garfield::ViewBase::m_plane {{0, 0, 1, 0}} [protected]

Definition at line 105 of file ViewBase.hh.

```
00105 {{0, 0, 1, 0}};
```

### m_prmat

std::array<std::array<double, 3>, 3> Garfield::ViewBase::m_prmat [protected]

Initial value:

```
{
    {{{1, 0, 0}}, {{0, 1, 0}}, {{0, 0, 1}}}}
```

Definition at line 107 of file ViewBase.hh.

```
00107                                       {
00108       {{{1, 0, 0}}, {{0, 1, 0}}, {{0, 0, 1}}}};
```

### m_proj

std::array<std::array<double, 3>, 3> Garfield::ViewBase::m_proj [protected]

Initial value:

```
{
    {{{1, 0, 0}}, {{0, 1, 0}}, {{0, 0, 0}}}}
```

Definition at line 103 of file ViewBase.hh.

```
00103                                       {
00104       {{{1, 0, 0}}, {{0, 1, 0}}, {{0, 0, 0}}}};
```

### m_userBox

bool Garfield::ViewBase::m_userBox = false [protected]

Definition at line 96 of file ViewBase.hh.

### m_userPlotLimits

bool Garfield::ViewBase::m_userPlotLimits = false [protected]
Definition at line 91 of file ViewBase.hh.

### m_xMaxBox

double Garfield::ViewBase::m_xMaxBox = 1. [protected]
Definition at line 97 of file ViewBase.hh.

### m_xMaxPlot

double Garfield::ViewBase::m_xMaxPlot = 1. [protected]
Definition at line 92 of file ViewBase.hh.

### m_xMinBox

double Garfield::ViewBase::m_xMinBox = -1. [protected]
Definition at line 97 of file ViewBase.hh.

### m_xMinPlot

double Garfield::ViewBase::m_xMinPlot = -1. [protected]
Definition at line 92 of file ViewBase.hh.

### m_yMaxBox

double Garfield::ViewBase::m_yMaxBox = 1. [protected]
Definition at line 98 of file ViewBase.hh.

### m_yMaxPlot

double Garfield::ViewBase::m_yMaxPlot = 1. [protected]
Definition at line 93 of file ViewBase.hh.

### m_yMinBox

double Garfield::ViewBase::m_yMinBox = -1. [protected]
Definition at line 98 of file ViewBase.hh.

### m_yMinPlot

double Garfield::ViewBase::m_yMinPlot = -1. [protected]
Definition at line 93 of file ViewBase.hh.

### m_zMaxBox

double Garfield::ViewBase::m_zMaxBox = 1. [protected]
Definition at line 99 of file ViewBase.hh.

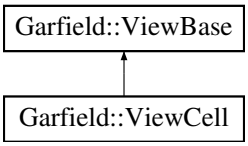
### m_zMinBox

double Garfield::ViewBase::m_zMinBox = -1. [protected]
Definition at line 99 of file ViewBase.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewBase.hh

## 7.122 Garfield::ViewCell Class Reference

Visualize the "cell" defined in an analytic-field component.
#include <ViewCell.hh>
Inheritance diagram for Garfield::ViewCell:




Public Member Functions

- ViewCell ()
  Default constructor.
- ViewCell (ComponentAnalyticField *cmp)
  Constructor from analytic-field component.
- ViewCell (ComponentNeBem2d *cmp)
  Constructor from two-dimenstional neBEM component.
- ~ViewCell ()=default
  Destructor.
- void SetComponent (ComponentAnalyticField *comp)
  Set the component for which to draw the cell geometry.
- void SetComponent (ComponentNeBem2d *comp)
- void Plot2d ()
  Make a two-dimensional drawing of the cell layout.
- void Plot3d ()
  Make a three-dimensional drawing of the cell layout (using TGeo).
- void EnableWireMarkers (const bool on=true)
  Visualize wirers using markers (default setting) or as a circle with the actual wire radius.
- void DisableWireMarkers ()

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete
  Default constructor.
- ViewBase (const std::string &name)
  Constructor.
- virtual ~ViewBase ()=default
  Destructor.
- void SetCanvas (TPad *pad)
  Set the canvas to be painted on.
- void SetCanvas ()
  Unset an external canvas.
- TPad * GetCanvas ()
  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)
  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)
  Set a bounding box (if applicable).
- void SetArea ()

Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).

- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

  Set the viewing plane to x-z.
- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

  Set the viewing plane to z-y.
- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Member Functions

- bool Plot (const bool twod)
- void PlotWire (const double x, const double y, const double d, const int type)
- void PlotWire (const double x, const double y, const double d, const int type, const double lz)
- void PlotTube (const double x0, const double y0, const double r, const int n)
- void PlotTube (const double x0, const double y0, const double r1, const double r2, const int n, const double lz)
- void PlotPlane (const double x0, const double y0, const double x1, const double y1)
- void PlotPlane (const double dx, const double dy, const double dz, const double x0, const double y0)
- bool PlotNeBem (const bool twod)
- void SetupGeo (const double dx, const double dy, const double dz)

Private Attributes

- bool m_useWireMarker = true
- ComponentAnalyticField * m_component = nullptr
- ComponentNeBem2d * m_nebem = nullptr
- std::unique_ptr< TGeoManager > m_geo

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.122.1 Detailed Description

Visualize the "cell" defined in an analytic-field component.
Definition at line 17 of file ViewCell.hh.

### 7.122.2 Constructor & Destructor Documentation

ViewCell() [1/3]

Garfield::ViewCell::ViewCell ()
Default constructor.

ViewCell() [2/3]

Garfield::ViewCell::ViewCell (
ComponentAnalyticField * cmp)
Constructor from analytic-field component.

ViewCell() [3/3]

Garfield::ViewCell::ViewCell (
ComponentNeBem2d * cmp)
Constructor from two-dimenstional neBEM component.

~ViewCell()

Garfield::ViewCell::~ViewCell () [default]
Destructor.

### 7.122.3 Member Function Documentation

DisableWireMarkers()

void Garfield::ViewCell::DisableWireMarkers () [inline]
Definition at line 40 of file ViewCell.hh.
00040 { EnableWireMarkers(false); }

EnableWireMarkers()

void Garfield::ViewCell::EnableWireMarkers (
const bool on = true) [inline]
Visualize wirers using markers (default setting) or as a circle with the actual wire radius.
Definition at line 39 of file ViewCell.hh.
00039 { m_useWireMarker = on; }

Plot()

bool Garfield::ViewCell::Plot (
const bool twod) [private]

Plot2d()

void Garfield::ViewCell::Plot2d ()
Make a two-dimensional drawing of the cell layout.

Plot3d()

void Garfield::ViewCell::Plot3d ()
Make a three-dimensional drawing of the cell layout (using TGeo).

PlotNeBem()

bool Garfield::ViewCell::PlotNeBem (
const bool twod) [private]

PlotPlane() [1/2]

void Garfield::ViewCell::PlotPlane (
const double dx,
const double dy,
const double dz,
const double x0,
const double y0) [private]

### PlotPlane() [2/2]

```
void Garfield::ViewCell::PlotPlane (
        const double x0,
        const double y0,
        const double x1,
        const double y1)   [private]
```

### PlotTube() [1/2]

```
void Garfield::ViewCell::PlotTube (
        const double x0,
        const double y0,
        const double r,
        const int n)   [private]
```

### PlotTube() [2/2]

```
void Garfield::ViewCell::PlotTube (
        const double x0,
        const double y0,
        const double r1,
        const double r2,
        const int n,
        const double lz)   [private]
```

### PlotWire() [1/2]

```
void Garfield::ViewCell::PlotWire (
        const double x,
        const double y,
        const double d,
        const int type)   [private]
```

### PlotWire() [2/2]

```
void Garfield::ViewCell::PlotWire (
        const double x,
        const double y,
        const double d,
        const int type,
        const double lz)   [private]
```

### SetComponent() [1/2]

```
void Garfield::ViewCell::SetComponent (
        ComponentAnalyticField * comp)
```

Set the component for which to draw the cell geometry.

### SetComponent() [2/2]

```
void Garfield::ViewCell::SetComponent (
        ComponentNeBem2d * comp)
```

### SetupGeo()

```
void Garfield::ViewCell::SetupGeo (
        const double dx,
        const double dy,
        const double dz)   [private]
```

### 7.122.4 Member Data Documentation

m_component

ComponentAnalyticField* Garfield::ViewCell::m_component = nullptr [private]
Definition at line 45 of file ViewCell.hh.

m_geo

std::unique_ptr<TGeoManager> Garfield::ViewCell::m_geo [private]
Definition at line 49 of file ViewCell.hh.

m_nebem

ComponentNeBem2d* Garfield::ViewCell::m_nebem = nullptr [private]
Definition at line 46 of file ViewCell.hh.

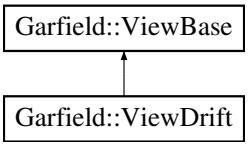
m_useWireMarker

bool Garfield::ViewCell::m_useWireMarker = true [private]
Definition at line 43 of file ViewCell.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewCell.hh

## 7.123 Garfield::ViewDrift Class Reference

Visualize drift lines and tracks.
#include <ViewDrift.hh>
Inheritance diagram for Garfield::ViewDrift:




Public Member Functions

- ViewDrift ()
  Constructor.
- ~ViewDrift ()=default
  Destructor.
- void Clear ()
  Delete existing drift lines, tracks and markers.
- void Plot (const bool twod=false, const bool axis=true, const bool snapshot=false)
  Draw the drift lines.
- void Plot2d (const bool axis=true, const bool snapshot=false)
  Make a 2D plot of the drift lines in the current viewing plane.
- void Plot3d (const bool axis=true, const bool ogl=true, const bool snapshot=false)
  Make a 3D plot of the drift lines.
- void EnableClusterMarkers (const bool on=true)
  Draw markers (or not) at every collision along a track.
- void SetClusterMarkerSize (const double size)
  Set the size of the cluster markers (see TAttMarker).
- void SetCollisionMarkerSize (const double size)
  Set the size of the collision markers (see TAttMarker).

- void SetColourElectrons (const short col)

  Set the colour with which to draw electron drift lines.
- void SetColourHoles (const short col)

  Set the colour with which to draw hole drift lines.
- void SetColourNegativeIons (const short col)

  Set the colour with which to draw negative ion drift lines.
- void SetColourIons (const short col)

  Set the colour with which to draw ion drift lines.
- void SetColourTracks (const short col)

  Set the colour with which to draw charged particle tracks.
- void SetColourPhotons (const short col)

  Set the colour with which to draw photons.
- void SetColourExcitations (const short col)

  Set the colour with which to draw excitation markers.
- void SetColourIonisations (const short col)

  Set the colour with which to draw ionisation markers.
- void SetColourAttachments (const short col)

  Set the colour with which to draw attachment markers.
- void SetElectronsToFront ()

  Put electron drift lines on top of the ion/hole drift lines.
- std::size_t GetNumberOfDriftLines () const

  Get the number of drift lines stored.
- void GetDriftLine (const std::size_t i, std::vector< std::array< float, 3 > > &driftLine, bool &electron) const

  Retrieve the coordinates of a given drift line.
- std::size_t NewDriftLine (const Particle particle, const std::size_t np, const float x0, const float y0, const float z0)
- void NewChargedParticleTrack (const std::size_t np, std::size_t &id, const float x0, const float y0, const float z0)
- void SetDriftLinePoint (const std::size_t iL, const std::size_t iP, const float x, const float y, const float z)
- void AddDriftLinePoint (const std::size_t iL, const float x, const float y, const float z)
- void SetTrackPoint (const std::size_t iL, const std::size_t iP, const float x, const float y, const float z)
- void AddTrackPoint (const std::size_t iL, const float x, const float y, const float z)
- void AddExcitation (const float x, const float y, const float z)
- void AddIonisation (const float x, const float y, const float z)
- void AddAttachment (const float x, const float y, const float z)
- void AddPhoton (const float x0, const float y0, const float z0, const float x1, const float y1, const float z1)

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.

- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

  Set the viewing plane to x-z.
- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

  Set the viewing plane to z-y.
- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Member Functions

- bool SetPlotLimits2d ()
- bool SetPlotLimits3d ()
- void DrawMarkers2d (const std::vector< std::array< float, 3 > > &points, const short col, const double size)
- void DrawMarkers3d (const std::vector< std::array< float, 3 > > &points, const short col, const double size)

Private Attributes

- std::mutex m_mutex
- std::vector< std::pair< std::vector< std::array< float, 3 > >, Particle > > m_driftLines
- std::vector< std::vector< std::array< float, 3 > > > m_tracks
- std::vector< std::array< std::array< float, 3 >, 2 > > m_photons
- std::vector< std::array< float, 3 > > m_exc
- std::vector< std::array< float, 3 > > m_ion
- std::vector< std::array< float, 3 > > m_att
- double m_markerSizeCluster = 0.01
- double m_markerSizeCollision = 0.5
- short m_colTrack = kGreen + 3
- short m_colPhoton = kBlue + 1
- short m_colElectron = kOrange - 3
- short m_colHole = kRed + 1

- short m_colIon = kRed + 1
- short m_colNegativeIon = kGray + 2
- short m_colExcitation = kGreen + 3
- short m_colIonisation = kOrange - 3
- short m_colAttachment = kCyan + 3
- bool m_drawClusters = false
- bool m_eTop = false

Friends

- class ViewFEMesh

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.

- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.123.1 Detailed Description

Visualize drift lines and tracks.
Definition at line 18 of file ViewDrift.hh.

### 7.123.2 Constructor & Destructor Documentation

#### ViewDrift()

Garfield::ViewDrift::ViewDrift ()
Constructor.

#### ~ViewDrift()

Garfield::ViewDrift::~ViewDrift () [default]
Destructor.

### 7.123.3 Member Function Documentation

#### AddAttachment()

void Garfield::ViewDrift::AddAttachment (
const float x,
const float y,
const float z)

#### AddDriftLinePoint()

void Garfield::ViewDrift::AddDriftLinePoint (
const std::size_t iL,
const float x,
const float y,
const float z)

#### AddExcitation()

void Garfield::ViewDrift::AddExcitation (
const float x,
const float y,
const float z)

#### AddIonisation()

void Garfield::ViewDrift::AddIonisation (
const float x,
const float y,
const float z)

### AddPhoton()

```
void Garfield::ViewDrift::AddPhoton (
                const float x0,
                const float y0,
                const float z0,
                const float x1,
                const float y1,
                const float z1)
```

### AddTrackPoint()

```
void Garfield::ViewDrift::AddTrackPoint (
                const std::size_t iL,
                const float x,
                const float y,
                const float z)
```

### Clear()

```
void Garfield::ViewDrift::Clear ()
```

Delete existing drift lines, tracks and markers.

### DrawMarkers2d()

```
void Garfield::ViewDrift::DrawMarkers2d (
                const std::vector< std::array< float, 3 > > & points,
                const short col,
                const double size)   [private]
```

### DrawMarkers3d()

```
void Garfield::ViewDrift::DrawMarkers3d (
                const std::vector< std::array< float, 3 > > & points,
                const short col,
                const double size)   [private]
```

### EnableClusterMarkers()

```
void Garfield::ViewDrift::EnableClusterMarkers (
                const bool on = true)   [inline]
```

Draw markers (or not) at every collision along a track.

Definition at line 38 of file ViewDrift.hh.

```
00038 { m_drawClusters = on; }
```

### GetDriftLine()

```
void Garfield::ViewDrift::GetDriftLine (
                const std::size_t i,
                std::vector< std::array< float, 3 > > & driftLine,
                bool & electron) const
```

Retrieve the coordinates of a given drift line.

### GetNumberOfDriftLines()

```
std::size_t Garfield::ViewDrift::GetNumberOfDriftLines () const   [inline]
```

Get the number of drift lines stored.

Definition at line 66 of file ViewDrift.hh.

```
00066 { return m_driftLines.size(); }
```

### NewChargedParticleTrack()

```
void Garfield::ViewDrift::NewChargedParticleTrack (
                const std::size_t np,
                std::size_t & id,
                const float x0,
                const float y0,
                const float z0)
```

### NewDriftLine()

```
std::size_t Garfield::ViewDrift::NewDriftLine (
                const Particle particle,
                const std::size_t np,
                const float x0,
                const float y0,
                const float z0)
```

### Plot()

```
void Garfield::ViewDrift::Plot (
                const bool twod = false,
                const bool axis = true,
                const bool snapshot = false)
```

Draw the drift lines.

### Plot2d()

```
void Garfield::ViewDrift::Plot2d (
                const bool axis = true,
                const bool snapshot = false)
```

Make a 2D plot of the drift lines in the current viewing plane.

### Plot3d()

```
void Garfield::ViewDrift::Plot3d (
                const bool axis = true,
                const bool ogl = true,
                const bool snapshot = false)
```

Make a 3D plot of the drift lines.

### SetClusterMarkerSize()

```
void Garfield::ViewDrift::SetClusterMarkerSize (
                const double size)
```

Set the size of the cluster markers (see TAttMarker).

### SetCollisionMarkerSize()

```
void Garfield::ViewDrift::SetCollisionMarkerSize (
                const double size)
```

Set the size of the collision markers (see TAttMarker).

### SetColourAttachments()

```
void Garfield::ViewDrift::SetColourAttachments (
                const short col)   [inline]
```

Set the colour with which to draw attachment markers.
Definition at line 61 of file ViewDrift.hh.

```
00061 { m_colAttachment = col; }
```

SetColourElectrons()

void Garfield::ViewDrift::SetColourElectrons (
const short col) [inline]
Set the colour with which to draw electron drift lines.
Definition at line 45 of file ViewDrift.hh.
00045 { m_colElectron = col; }

SetColourExcitations()

void Garfield::ViewDrift::SetColourExcitations (
const short col) [inline]
Set the colour with which to draw excitation markers.
Definition at line 57 of file ViewDrift.hh.
00057 { m_colExcitation = col; }

SetColourHoles()

void Garfield::ViewDrift::SetColourHoles (
const short col) [inline]
Set the colour with which to draw hole drift lines.
Definition at line 47 of file ViewDrift.hh.
00047 { m_colHole = col; }

SetColourIonisations()

void Garfield::ViewDrift::SetColourIonisations (
const short col) [inline]
Set the colour with which to draw ionisation markers.
Definition at line 59 of file ViewDrift.hh.
00059 { m_colIonisation = col; }

SetColourIons()

void Garfield::ViewDrift::SetColourIons (
const short col) [inline]
Set the colour with which to draw ion drift lines.
Definition at line 51 of file ViewDrift.hh.
00051 { m_colIon = col; }

SetColourNegativeIons()

void Garfield::ViewDrift::SetColourNegativeIons (
const short col) [inline]
Set the colour with which to draw negative ion drift lines.
Definition at line 49 of file ViewDrift.hh.
00049 { m_colNegativeIon = col; }

SetColourPhotons()

void Garfield::ViewDrift::SetColourPhotons (
const short col) [inline]
Set the colour with which to draw photons.
Definition at line 55 of file ViewDrift.hh.
00055 { m_colPhoton = col; }

SetColourTracks()

void Garfield::ViewDrift::SetColourTracks (
const short col) [inline]
Set the colour with which to draw charged particle tracks.

Definition at line 53 of file ViewDrift.hh.
00053 { m_colTrack = col; }

SetDriftLinePoint()

void Garfield::ViewDrift::SetDriftLinePoint (
const std::size_t iL,
const std::size_t iP,
const float x,
const float y,
const float z)

SetElectronsToFront()

void Garfield::ViewDrift::SetElectronsToFront () [inline]
Put electron drift lines on top of the ion/hole drift lines.
Definition at line 63 of file ViewDrift.hh.
00063 { m_eTop = true; }

SetPlotLimits2d()

bool Garfield::ViewDrift::SetPlotLimits2d () [private]

SetPlotLimits3d()

bool Garfield::ViewDrift::SetPlotLimits3d () [private]

SetTrackPoint()

void Garfield::ViewDrift::SetTrackPoint (
const std::size_t iL,
const std::size_t iP,
const float x,
const float y,
const float z)

### 7.123.4 Friends And Related Symbol Documentation

ViewFEMesh

friend class ViewFEMesh [friend]
Definition at line 93 of file ViewDrift.hh.

### 7.123.5 Member Data Documentation

m_att

std::vector<std::array<float, 3> > Garfield::ViewDrift::m_att [private]
Definition at line 106 of file ViewDrift.hh.

m_colAttachment

short Garfield::ViewDrift::m_colAttachment = kCyan + 3 [private]
Definition at line 119 of file ViewDrift.hh.

m_colElectron

short Garfield::ViewDrift::m_colElectron = kOrange - 3 [private]
Definition at line 113 of file ViewDrift.hh.

### m_colExcitation

short Garfield::ViewDrift::m_colExcitation = kGreen + 3 [private]
Definition at line 117 of file ViewDrift.hh.

### m_colHole

short Garfield::ViewDrift::m_colHole = kRed + 1 [private]
Definition at line 114 of file ViewDrift.hh.

### m_colIon

short Garfield::ViewDrift::m_colIon = kRed + 1 [private]
Definition at line 115 of file ViewDrift.hh.

### m_colIonisation

short Garfield::ViewDrift::m_colIonisation = kOrange - 3 [private]
Definition at line 118 of file ViewDrift.hh.

### m_colNegativeIon

short Garfield::ViewDrift::m_colNegativeIon = kGray + 2 [private]
Definition at line 116 of file ViewDrift.hh.

### m_colPhoton

short Garfield::ViewDrift::m_colPhoton = kBlue + 1 [private]
Definition at line 112 of file ViewDrift.hh.

### m_colTrack

short Garfield::ViewDrift::m_colTrack = kGreen + 3 [private]
Definition at line 111 of file ViewDrift.hh.

### m_drawClusters

bool Garfield::ViewDrift::m_drawClusters = false [private]
Definition at line 121 of file ViewDrift.hh.

### m_driftLines

std::vector<std::pair<std::vector<std::array<float, 3> >, Particle> > Garfield::ViewDrift::m_driftLines [private]
Definition at line 99 of file ViewDrift.hh.

### m_eTop

bool Garfield::ViewDrift::m_eTop = false [private]
Definition at line 123 of file ViewDrift.hh.

### m_exc

std::vector<std::array<float, 3> > Garfield::ViewDrift::m_exc [private]
Definition at line 104 of file ViewDrift.hh.

### m_ion

std::vector<std::array<float, 3> > Garfield::ViewDrift::m_ion [private]
Definition at line 105 of file ViewDrift.hh.

m_markerSizeCluster

double Garfield::ViewDrift::m_markerSizeCluster = 0.01 [private]
Definition at line 108 of file ViewDrift.hh.

m_markerSizeCollision

double Garfield::ViewDrift::m_markerSizeCollision = 0.5 [private]
Definition at line 109 of file ViewDrift.hh.

m_mutex

std::mutex Garfield::ViewDrift::m_mutex [private]
Definition at line 96 of file ViewDrift.hh.

m_photons

std::vector<std::array<std::array<float, 3>, 2> > Garfield::ViewDrift::m_photons [private]
Definition at line 102 of file ViewDrift.hh.

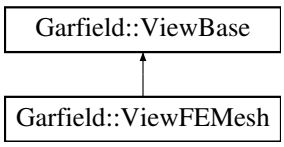
m_tracks

std::vector<std::vector<std::array<float, 3> > > Garfield::ViewDrift::m_tracks [private]
Definition at line 101 of file ViewDrift.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewDrift.hh

## 7.124 Garfield::ViewFEMesh Class Reference

Draw the mesh of a field-map component.
#include <ViewFEMesh.hh>
Inheritance diagram for Garfield::ViewFEMesh:




Public Member Functions

- ViewFEMesh ()
  Default constructor.
- ViewFEMesh (Component *cmp)
  Constructor.
- ~ViewFEMesh ()
  Destructor.
- void SetComponent (Component *cmp)
  Set the component from which to retrieve the mesh and field.
- void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0) override
  Set the projection (viewing plane), if applicable.
- void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz) override
  Set the projection plane specifying a hint for the in-plane x axis.
- void SetXaxis (TGaxis *ax)
- void SetYaxis (TGaxis *ay)

- void SetXaxisTitle (const std::string &xtitle)
- void SetYaxisTitle (const std::string &ytitle)
- void EnableAxes ()
- void DisableAxes ()
- bool Plot (const bool twod=true, const bool outline=false)

  Plot method to be called by user.
- void SetFillMesh (const bool f)

  Element fill switch; 2D only, set false for wireframe mesh.
- void SetDrawViewRegion (bool do_draw)

  Display intersection of projection plane with viewing area.
- bool GetDrawViewRegion (void) const
- void SetColor (int matID, int colorID)

  Associate a color with each element material map ID.
- void SetColors (const std::map< int, int > &colors)
- void SetFillColor (int matID, int colorID)
- void SetFillColors (const std::map< int, int > &colors)
- void SetViewDrift (ViewDrift *vd)

  Set the optional associated ViewDrift.
- void SetFillMeshWithBorders ()

  Show filled mesh elements.
- void CreateDefaultAxes ()

  Create a default set of custom-made axes.
- void DisableMaterial (int materialID)

  Disable a material so that its mesh cells are not drawn.

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.
- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

  Set the viewing plane to x-z.

- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

  Set the viewing plane to z-y.
- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Types

- typedef std::vector< size_t > Facet

Private Member Functions

- void DrawElements2d ()
- void DrawElements3d ()
- void DrawCST (ComponentCST *componentCST)
- void DrawBorders2d ()
- void DrawBorders3d ()
- void AddFacets (const size_t i, const std::vector< std::vector< Facet > > &elementFacets, const std::map< Facet, std::vector< size_t > > &facetElements, std::vector< Facet > &facets, std::vector< bool > &done) const
- bool FacetSign (const Facet &f, const size_t element) const
- void DrawDriftLines2d ()
- void DrawDriftLines3d ()
- bool GetPlotLimits ()
- bool InView (const double x, const double y) const

  Return true if the specified point is in the view region.
- bool LinesCrossed (double x1, double y1, double x2, double y2, double u1, double v1, double u2, double v2, double &xc, double &yc) const
- bool IntersectPlaneArea (double &xmin, double &ymin, double &xmax, double &ymax)
- bool OnLine (double x1, double y1, double x2, double y2, double u, double v) const
- void RemoveCrossings (std::vector< double > &x, std::vector< double > &y)
- bool PlaneCut (double x1, double y1, double z1, double x2, double y2, double z2, TMatrixD &xMat)
- void ClipToView (std::vector< double > &px, std::vector< double > &py, std::vector< double > &cx, std::vector< double > &cy)
- bool IsInPolygon (double x, double y, const std::vector< double > &px, const std::vector< double > &py, bool &edge) const
- void Reset ()

Private Attributes

- bool m_fillMesh = false
- bool m_drawViewRegion = false
- std::vector< double > m_viewRegionX
- std::vector< double > m_viewRegionY
- Component * m_cmp = nullptr
- ViewDrift * m_viewDrift = nullptr
- bool m_plotMeshBorders = false
- TGaxis * m_xaxis = nullptr
- TGaxis * m_yaxis = nullptr
- std::string m_xaxisTitle = ""
- std::string m_yaxisTitle = ""
- bool m_drawAxes = false
- std::map< int, int > m_colorMap

- std::map< int, int > m_colorMap_fill
- std::map< int, bool > m_disabledMaterial
- std::vector< TGeoVolume * > m_volumes
- std::vector< TGeoMedium * > m_media
- std::unique_ptr< TGeoManager > m_geoManager

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.

- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.124.1 Detailed Description

Draw the mesh of a field-map component.
Definition at line 25 of file ViewFEMesh.hh.

### 7.124.2 Member Typedef Documentation

Facet

typedef std::vector<size_t> Garfield::ViewFEMesh::Facet [private]
Definition at line 131 of file ViewFEMesh.hh.

### 7.124.3 Constructor & Destructor Documentation

ViewFEMesh() [1/2]

Garfield::ViewFEMesh::ViewFEMesh () [inline]
Default constructor.
Definition at line 28 of file ViewFEMesh.hh.
00028 : ViewFEMesh(nullptr) {}

ViewFEMesh() [2/2]

Garfield::ViewFEMesh::ViewFEMesh (
Component * cmp)
Constructor.

~ViewFEMesh()

Garfield::ViewFEMesh::~ViewFEMesh ()
Destructor.

### 7.124.4 Member Function Documentation

AddFacets()

void Garfield::ViewFEMesh::AddFacets (
const size_t i,
const std::vector< std::vector< Facet > > & elementFacets,
const std::map< Facet, std::vector< size_t > > & facetElements,
std::vector< Facet > & facets,
std::vector< bool > & done) const [private]

ClipToView()

void Garfield::ViewFEMesh::ClipToView (
std::vector< double > & px,
std::vector< double > & py,
std::vector< double > & cx,
std::vector< double > & cy) [private]

CreateDefaultAxes()

void Garfield::ViewFEMesh::CreateDefaultAxes ()
Create a default set of custom-made axes.

### DisableAxes()

void Garfield::ViewFEMesh::DisableAxes () [inline]
Definition at line 49 of file ViewFEMesh.hh.

```
00049 { m_drawAxes = false; }
```

### DisableMaterial()

void Garfield::ViewFEMesh::DisableMaterial (
int materialID) [inline]
Disable a material so that its mesh cells are not drawn.
Definition at line 86 of file ViewFEMesh.hh.

```
00086                                       {
00087     m_disabledMaterial[materialID] = true;
00088   }
```

### DrawBorders2d()

void Garfield::ViewFEMesh::DrawBorders2d () [private]

### DrawBorders3d()

void Garfield::ViewFEMesh::DrawBorders3d () [private]

### DrawCST()

void Garfield::ViewFEMesh::DrawCST (
ComponentCST * componentCST) [private]

### DrawDriftLines2d()

void Garfield::ViewFEMesh::DrawDriftLines2d () [private]

### DrawDriftLines3d()

void Garfield::ViewFEMesh::DrawDriftLines3d () [private]

### DrawElements2d()

void Garfield::ViewFEMesh::DrawElements2d () [private]

### DrawElements3d()

void Garfield::ViewFEMesh::DrawElements3d () [private]

### EnableAxes()

void Garfield::ViewFEMesh::EnableAxes () [inline]
Definition at line 48 of file ViewFEMesh.hh.

```
00048 { m_drawAxes = true; }
```

### FacetSign()

bool Garfield::ViewFEMesh::FacetSign (
const Facet & f,
const size_t element) const [private]

### GetDrawViewRegion()

bool Garfield::ViewFEMesh::GetDrawViewRegion (
void ) const [inline]
Definition at line 59 of file ViewFEMesh.hh.

```
00059 { return m_drawViewRegion; }
```

### GetPlotLimits()

bool Garfield::ViewFEMesh::GetPlotLimits () [private]

### IntersectPlaneArea()

bool Garfield::ViewFEMesh::IntersectPlaneArea (
double & xmin,
double & ymin,
double & xmax,
double & ymax) [private]

### InView()

bool Garfield::ViewFEMesh::InView (
const double x,
const double y) const [private]

Return true if the specified point is in the view region.

### IsInPolygon()

bool Garfield::ViewFEMesh::IsInPolygon (
double x,
double y,
const std::vector< double > & px,
const std::vector< double > & py,
bool & edge) const [private]

### LinesCrossed()

bool Garfield::ViewFEMesh::LinesCrossed (
double x1,
double y1,
double x2,
double y2,
double u1,
double v1,
double u2,
double v2,
double & xc,
double & yc) const [private]

### OnLine()

bool Garfield::ViewFEMesh::OnLine (
double x1,
double y1,
double x2,
double y2,
double u,
double v) const [private]

### PlaneCut()

bool Garfield::ViewFEMesh::PlaneCut (
double x1,
double y1,
double z1,
double x2,
double y2,

double z2,
TMatrixD & xMat) [private]

### Plot()

bool Garfield::ViewFEMesh::Plot (
const bool twod = true,
const bool outline = false)

Plot method to be called by user.

### RemoveCrossings()

void Garfield::ViewFEMesh::RemoveCrossings (
std::vector< double > & x,
std::vector< double > & y) [private]

### Reset()

void Garfield::ViewFEMesh::Reset () [private]

### SetColor()

void Garfield::ViewFEMesh::SetColor (
int matID,
int colorID) [inline]

Associate a color with each element material map ID.
Uses ROOT color numberings.
Definition at line 63 of file ViewFEMesh.hh.

```
00063 { m_colorMap[matID] = colorID; }
```

### SetColors()

void Garfield::ViewFEMesh::SetColors (
const std::map< int, int > & colors) [inline]

Definition at line 64 of file ViewFEMesh.hh.

```
00064                                                 {
00065     for (const auto& c : colors) SetColor(c.first, c.second);
00066   }
```

### SetComponent()

void Garfield::ViewFEMesh::SetComponent (
Component * cmp)

Set the component from which to retrieve the mesh and field.

### SetDrawViewRegion()

void Garfield::ViewFEMesh::SetDrawViewRegion (
bool do_draw) [inline]

Display intersection of projection plane with viewing area.
Definition at line 58 of file ViewFEMesh.hh.

```
00058 { m_drawViewRegion = do_draw; }
```

### SetFillColor()

void Garfield::ViewFEMesh::SetFillColor (
int matID,
int colorID) [inline]

Definition at line 67 of file ViewFEMesh.hh.

```
00067                                                 {
00068     m_colorMap_fill[matID] = colorID;
00069   }
```

### SetFillColors()

```
void Garfield::ViewFEMesh::SetFillColors (
                const std::map< int, int > & colors)   [inline]
```

Definition at line 70 of file ViewFEMesh.hh.

```
00070                                                  {
00071     for (const auto& c : colors) SetFillColor(c.first, c.second);
00072   }
```

### SetFillMesh()

```
void Garfield::ViewFEMesh::SetFillMesh (
                const bool f)   [inline]
```

Element fill switch; 2D only, set false for wireframe mesh.

Definition at line 55 of file ViewFEMesh.hh.

```
00055 { m_fillMesh = f; }
```

### SetFillMeshWithBorders()

```
void Garfield::ViewFEMesh::SetFillMeshWithBorders ()   [inline]
```

Show filled mesh elements.

Definition at line 77 of file ViewFEMesh.hh.

```
00077                                 {
00078     m_plotMeshBorders = true;
00079     m_fillMesh = true;
00080   }
```

### SetPlane() [1/2]

```
void Garfield::ViewFEMesh::SetPlane (
                const double fx,
                const double fy,
                const double fz,
                const double x0,
                const double y0,
                const double z0)   [override], [virtual]
```

Set the projection (viewing plane), if applicable.

Parameters

| | |
|---|---|
| fx,fy,fz | normal vector |
| x0,y0,z0 | in-plane point |

Reimplemented from Garfield::ViewBase.

### SetPlane() [2/2]

```
void Garfield::ViewFEMesh::SetPlane (
                const double fx,
                const double fy,
                const double fz,
                const double x0,
                const double y0,
                const double z0,
                const double hx,
                const double hy,
                const double hz)   [override], [virtual]
```

Set the projection plane specifying a hint for the in-plane x axis.

Reimplemented from Garfield::ViewBase.

SetViewDrift()

void Garfield::ViewFEMesh::SetViewDrift (
ViewDrift * vd) [inline]
Set the optional associated ViewDrift.
Definition at line 74 of file ViewFEMesh.hh.
00074 { m_viewDrift = vd; }

SetXaxis()

void Garfield::ViewFEMesh::SetXaxis (
TGaxis * ax)

SetXaxisTitle()

void Garfield::ViewFEMesh::SetXaxisTitle (
const std::string & xtitle) [inline]
Definition at line 46 of file ViewFEMesh.hh.
00046 { m_xaxisTitle = xtitle; }

SetYaxis()

void Garfield::ViewFEMesh::SetYaxis (
TGaxis * ay)

SetYaxisTitle()

void Garfield::ViewFEMesh::SetYaxisTitle (
const std::string & ytitle) [inline]
Definition at line 47 of file ViewFEMesh.hh.
00047 { m_yaxisTitle = ytitle; }

### 7.124.5 Member Data Documentation

m_cmp

Component* Garfield::ViewFEMesh::m_cmp = nullptr [private]
Definition at line 100 of file ViewFEMesh.hh.

m_colorMap

std::map<int, int> Garfield::ViewFEMesh::m_colorMap [private]
Definition at line 114 of file ViewFEMesh.hh.

m_colorMap_fill

std::map<int, int> Garfield::ViewFEMesh::m_colorMap_fill [private]
Definition at line 115 of file ViewFEMesh.hh.

m_disabledMaterial

std::map<int, bool> Garfield::ViewFEMesh::m_disabledMaterial [private]
Definition at line 118 of file ViewFEMesh.hh.

m_drawAxes

bool Garfield::ViewFEMesh::m_drawAxes = false [private]
Definition at line 111 of file ViewFEMesh.hh.

m_drawViewRegion

bool Garfield::ViewFEMesh::m_drawViewRegion = false [private]
Definition at line 95 of file ViewFEMesh.hh.

### m_fillMesh

bool Garfield::ViewFEMesh::m_fillMesh = false [private]
Definition at line 92 of file ViewFEMesh.hh.

### m_geoManager

std::unique_ptr<TGeoManager> Garfield::ViewFEMesh::m_geoManager [private]
Definition at line 122 of file ViewFEMesh.hh.

### m_media

std::vector<TGeoMedium*> Garfield::ViewFEMesh::m_media [private]
Definition at line 121 of file ViewFEMesh.hh.

### m_plotMeshBorders

bool Garfield::ViewFEMesh::m_plotMeshBorders = false [private]
Definition at line 104 of file ViewFEMesh.hh.

### m_viewDrift

ViewDrift* Garfield::ViewFEMesh::m_viewDrift = nullptr [private]
Definition at line 103 of file ViewFEMesh.hh.

### m_viewRegionX

std::vector<double> Garfield::ViewFEMesh::m_viewRegionX [private]
Definition at line 96 of file ViewFEMesh.hh.

### m_viewRegionY

std::vector<double> Garfield::ViewFEMesh::m_viewRegionY [private]
Definition at line 97 of file ViewFEMesh.hh.

### m_volumes

std::vector<TGeoVolume*> Garfield::ViewFEMesh::m_volumes [private]
Definition at line 120 of file ViewFEMesh.hh.

### m_xaxis

TGaxis* Garfield::ViewFEMesh::m_xaxis = nullptr [private]
Definition at line 107 of file ViewFEMesh.hh.

### m_xaxisTitle

std::string Garfield::ViewFEMesh::m_xaxisTitle = "" [private]
Definition at line 109 of file ViewFEMesh.hh.

### m_yaxis

TGaxis* Garfield::ViewFEMesh::m_yaxis = nullptr [private]
Definition at line 108 of file ViewFEMesh.hh.

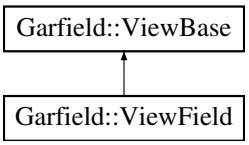
### m_yaxisTitle

std::string Garfield::ViewFEMesh::m_yaxisTitle = "" [private]
Definition at line 110 of file ViewFEMesh.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewFEMesh.hh

## 7.125 Garfield::ViewField Class Reference

Visualize the potential or electric field of a component or sensor.
#include <ViewField.hh>
Inheritance diagram for Garfield::ViewField:




Public Member Functions

- ViewField ()
  Default constructor.
- ViewField (Sensor *sensor)
  Constructor from sensor.
- ViewField (Component *component)
  Constructor from component.
- ~ViewField ()=default
  Destructor.
- void SetSensor (Sensor *s)
  Set the sensor for which to plot the field.
- void SetComponent (Component *c)
  Set the component for which to plot the field.
- void SetVoltageRange (const double vmin, const double vmax)
  Set the plot limits for the potential.
- void SetElectricFieldRange (const double emin, const double emax)
  Set the plot limits for the electric field.
- void SetWeightingFieldRange (const double wmin, const double wmax)
  Set the plot limits for the weighting field.
- void SetMagneticFieldRange (const double bmin, const double bmax)
  Set the plot limits for the magnetic field.
- void SetNumberOfContours (const unsigned int n)
  Set the number of contour levels.
- void SetNumberOfSamples1d (const unsigned int n)
  Set the number of points used for drawing 1D functions.
- void SetNumberOfSamples2d (const unsigned int nx, const unsigned int ny)
  Set the number of points used for drawing 2D functions.
- void PlotContour (const std::string &option="v")
  Make a contour plot of the electric potential, electric field, or magnetic field.
- void Plot (const std::string &option="v", const std::string &drawopt="arr")
  Make a 2D plot of the electric potential, electric field or magnetic field.
- void PlotProfile (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, const std::string &option="v", const bool normalised=true)
  Make a 1D plot of the potential or field along a line.
- void PlotContourWeightingField (const std::string &label, const std::string &option)
  Make a contour plot of the weighting potential or field.
- void PlotWeightingField (const std::string &label, const std::string &option, const std::string &drawopt, const double t=0.)
  Make a 2D plot of the weighting potential or field.

- void PlotProfileWeightingField (const std::string &label, const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, const std::string &option="v", const bool normalised=true)

  Make a 1D plot of the weighting potential or field along a line.
- void EnableAutoRange (const bool on=true, const bool samplePotential=true)

  Determine the range of the potential/field automatically (true) or set it explicitly (false).
- void AcknowledgeStatus (const bool on, const double v0=0.)

  Make use (or not) of the status flag returned by the sensor/component.
- void PlotFieldLines (const std::vector< double > &x0, const std::vector< double > &y0, const std::vector< double > &z0, const bool electron=true, const bool axis=true, const short col=kOrange - 3)

  Draw electric field lines from a set of starting points.
- bool EqualFluxIntervals (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, std::vector< double > &xf, std::vector< double > &yf, std::vector< double > &zf, const unsigned int nPoints=20) const

  Generates point along a line, spaced by equal flux intervals.
- bool FixedFluxIntervals (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, std::vector< double > &xf, std::vector< double > &yf, std::vector< double > &zf, const double interval=10.) const

  Generate points along a line, spaced by a given flux interval.

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.
- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

Set the viewing plane to x-z.

- void SetPlaneYZ ()

  Set the viewing plane to y-z.

- void SetPlaneZX ()

  Set the viewing plane to z-x.

- void SetPlaneZY ()

  Set the viewing plane to z-y.

- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Types

- enum class Parameter {
  Potential = 0 , Emag , Ex , Ey ,
  Ez , Bmag , Bx , By ,
  Bz , Unknown }

Private Member Functions

- bool SetPlotLimits ()
- void Draw2d (const std::string &option, const bool contour, const bool wfield, const std::string &electrode, const std::string &drawopt, const double t=0.)
- void DrawProfile (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1, const std::string &option, const bool wfield, const std::string &electrode, const bool normalised)
- Parameter GetPar (const std::string &option, std::string &title, bool &bfield) const
- double Efield (const double x, const double y, const double z, const Parameter par) const
- double Wfield (const double x, const double y, const double z, const Parameter par, const std::string &electrode, const double t=0.) const
- double Bfield (const double x, const double y, const double z, const Parameter par) const

Private Attributes

- bool m_useAutoRange = true
- bool m_samplePotential = true
- bool m_useStatus = false
- double m_vBkg = 0.
- Sensor * m_sensor = nullptr
- Component * m_component = nullptr
- double m_vmin = 0.
- double m_vmax = 100.
- double m_emin = 0.
- double m_emax = 10000.
- double m_wmin = 0.
- double m_wmax = 100.
- double m_bmin = 0.
- double m_bmax = 10.
- unsigned int m_nContours = 20
- unsigned int m_nSamples1d = 1000
- unsigned int m_nSamples2dX = 200
- unsigned int m_nSamples2dY = 200

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.125.1 Detailed Description

Visualize the potential or electric field of a component or sensor.
Definition at line 15 of file ViewField.hh.

### 7.125.2 Member Enumeration Documentation

Parameter

enum class Garfield::ViewField::Parameter [strong], [private]

Enumerator

| | |
|---|---|
| Potential | |
| Emag | |
| Ex | |
| Ey | |
| Ez | |
| Bmag | |
| Bx | |
| By | |
| Bz | |
| Unknown | |

Definition at line 143 of file ViewField.hh.

```
00143                   {
00144     Potential = 0,
00145     Emag,
00146     Ex,
00147     Ey,
00148     Ez,
00149     Bmag,
00150     Bx,
00151     By,
00152     Bz,
00153     Unknown
00154   };
```

### 7.125.3 Constructor & Destructor Documentation

ViewField() [1/3]

Garfield::ViewField::ViewField ()
Default constructor.

ViewField() [2/3]

Garfield::ViewField::ViewField (
Sensor * sensor)
Constructor from sensor.

ViewField() [3/3]

Garfield::ViewField::ViewField (
Component * component)
Constructor from component.

~ViewField()

Garfield::ViewField::~ViewField () [default]
Destructor.

### 7.125.4 Member Function Documentation

#### AcknowledgeStatus()

```
void Garfield::ViewField::AcknowledgeStatus (
                const bool on,
                const double v0 = 0.)   [inline]
```

Make use (or not) of the status flag returned by the sensor/component.

Parameters

| | |
|---|---|
| on | Take status flag into account (true) or ignore it (false). |
| v0 | Value to be used for regions with status != 0. |

Definition at line 119 of file ViewField.hh.

```
00119                                                {
00120     m_useStatus = on;
00121     m_vBkg = v0;
00122   }
```

#### Bfield()

```
double Garfield::ViewField::Bfield (
                const double x,
                const double y,
                const double z,
                const Parameter par) const   [private]
```

#### Draw2d()

```
void Garfield::ViewField::Draw2d (
                const std::string & option,
                const bool contour,
                const bool wfield,
                const std::string & electrode,
                const std::string & drawopt,
                const double t = 0.)   [private]
```

#### DrawProfile()

```
void Garfield::ViewField::DrawProfile (
                const double x0,
                const double y0,
                const double z0,
                const double x1,
                const double y1,
                const double z1,
                const std::string & option,
                const bool wfield,
                const std::string & electrode,
                const bool normalised)   [private]
```

#### Efield()

```
double Garfield::ViewField::Efield (
                const double x,
                const double y,
                const double z,
                const Parameter par) const   [private]
```

### EnableAutoRange()

```
void Garfield::ViewField::EnableAutoRange (
                const bool on = true,
                const bool samplePotential = true)   [inline]
```

Determine the range of the potential/field automatically (true) or set it explicitly (false).
Definition at line 109 of file ViewField.hh.

```
00110                                                {
00111     m_useAutoRange = on;
00112     m_samplePotential = samplePotential;
00113   }
```

### EqualFluxIntervals()

```
bool Garfield::ViewField::EqualFluxIntervals (
                const double x0,
                const double y0,
                const double z0,
                const double x1,
                const double y1,
                const double z1,
                std::vector< double > & xf,
                std::vector< double > & yf,
                std::vector< double > & zf,
                const unsigned int nPoints = 20) const
```

Generates point along a line, spaced by equal flux intervals.

### FixedFluxIntervals()

```
bool Garfield::ViewField::FixedFluxIntervals (
                const double x0,
                const double y0,
                const double z0,
                const double x1,
                const double y1,
                const double z1,
                std::vector< double > & xf,
                std::vector< double > & yf,
                std::vector< double > & zf,
                const double interval = 10.) const
```

Generate points along a line, spaced by a given flux interval.

### GetPar()

```
Parameter Garfield::ViewField::GetPar (
                const std::string & option,
                std::string & title,
                bool & bfield) const   [private]
```

### Plot()

```
void Garfield::ViewField::Plot (
                const std::string & option = "v",
                const std::string & drawopt = "arr")
```

Make a 2D plot of the electric potential, electric field or magnetic field.

#### Parameters

| | |
|---|---|
| option | quantity to be plotted (see PlotContour) |
| drawopt | option string passed to TF2::Draw |

PlotContour()

void Garfield::ViewField::PlotContour (
const std::string & option = "v")
Make a contour plot of the electric potential, electric field, or magnetic field.

Parameters

| | |
|---|---|
| option | quantity to be plotted<br>• potential: "v", "voltage", "p", "potential"<br>• magnitude of the electric field: "emag", "field"<br>• x-component of the electric field: "ex"<br>• y-component of the electric field: "ey"<br>• z-component of the electric field: "ez"<br>• magnitude of the magnetic field: "bmag"<br>• x-component of the magnetic field: "bx"<br>• y-component of the magnetic field: "by"<br>• z-component of the magnetic field: "bz" |

PlotContourWeightingField()

void Garfield::ViewField::PlotContourWeightingField (
const std::string & label,
const std::string & option)
Make a contour plot of the weighting potential or field.

Parameters

| | |
|---|---|
| label | identifier of the electrode |
| option | quantity to be plotted (see PlotContour) |

PlotFieldLines()

void Garfield::ViewField::PlotFieldLines (
const std::vector< double > & x0,
const std::vector< double > & y0,
const std::vector< double > & z0,
const bool electron = true,
const bool axis = true,
const short col = kOrange - 3)
Draw electric field lines from a set of starting points.

PlotProfile()

void Garfield::ViewField::PlotProfile (
const double x0,
const double y0,
const double z0,
const double x1,
const double y1,

const double z1,
const std::string & option = "v",
const bool normalised = true)

Make a 1D plot of the potential or field along a line.

Parameters

| | |
|---|---|
| x0,y0,z0 | starting point |
| x1,y1,z1 | end point |
| option | quantity to be plotted (see PlotContour) |
| normalised | flag whether to use normalised x-axis coordinates |

### PlotProfileWeightingField()

void Garfield::ViewField::PlotProfileWeightingField (
const std::string & label,
const double x0,
const double y0,
const double z0,
const double x1,
const double y1,
const double z1,
const std::string & option = "v",
const bool normalised = true)

Make a 1D plot of the weighting potential or field along a line.

Parameters

| | |
|---|---|
| label | identifier of the electrode |
| x0,y0,z0 | starting point |
| x1,y1,z1 | end point |
| option | quantity to be plotted (see PlotContour) |
| normalised | flag whether to use normalised x-axis coordinates |

### PlotWeightingField()

void Garfield::ViewField::PlotWeightingField (
const std::string & label,
const std::string & option,
const std::string & drawopt,
const double t = 0.)

Make a 2D plot of the weighting potential or field.

Parameters

| | |
|---|---|
| label | identifier of the electrode |
| option | quantity to be plotted (see PlotContour) |
| drawopt | option string passed to TF2::Draw |
| t | time slice of dynamic weighting potential [ns]. |

SetComponent()

void Garfield::ViewField::SetComponent (
Component * c)
Set the component for which to plot the field.

SetElectricFieldRange()

void Garfield::ViewField::SetElectricFieldRange (
const double emin,
const double emax)
Set the plot limits for the electric field.

SetMagneticFieldRange()

void Garfield::ViewField::SetMagneticFieldRange (
const double bmin,
const double bmax)
Set the plot limits for the magnetic field.

SetNumberOfContours()

void Garfield::ViewField::SetNumberOfContours (
const unsigned int n)
Set the number of contour levels.

SetNumberOfSamples1d()

void Garfield::ViewField::SetNumberOfSamples1d (
const unsigned int n)
Set the number of points used for drawing 1D functions.

SetNumberOfSamples2d()

void Garfield::ViewField::SetNumberOfSamples2d (
const unsigned int nx,
const unsigned int ny)
Set the number of points used for drawing 2D functions.

SetPlotLimits()

bool Garfield::ViewField::SetPlotLimits () [private]

SetSensor()

void Garfield::ViewField::SetSensor (
Sensor * s)
Set the sensor for which to plot the field.

SetVoltageRange()

void Garfield::ViewField::SetVoltageRange (
const double vmin,
const double vmax)
Set the plot limits for the potential.

SetWeightingFieldRange()

void Garfield::ViewField::SetWeightingFieldRange (
const double wmin,
const double wmax)
Set the plot limits for the weighting field.

### Wfield()

```
double Garfield::ViewField::Wfield (
                const double x,
                const double y,
                const double z,
                const Parameter par,
                const std::string & electrode,
                const double t = 0.) const   [private]
```

## 7.125.5 Member Data Documentation

### m_bmax

double Garfield::ViewField::m_bmax = 10. [private]
Definition at line 169 of file ViewField.hh.

### m_bmin

double Garfield::ViewField::m_bmin = 0. [private]
Definition at line 169 of file ViewField.hh.

### m_component

Component* Garfield::ViewField::m_component = nullptr [private]
Definition at line 163 of file ViewField.hh.

### m_emax

double Garfield::ViewField::m_emax = 10000. [private]
Definition at line 167 of file ViewField.hh.

### m_emin

double Garfield::ViewField::m_emin = 0. [private]
Definition at line 167 of file ViewField.hh.

### m_nContours

unsigned int Garfield::ViewField::m_nContours = 20 [private]
Definition at line 172 of file ViewField.hh.

### m_nSamples1d

unsigned int Garfield::ViewField::m_nSamples1d = 1000 [private]
Definition at line 174 of file ViewField.hh.

### m_nSamples2dX

unsigned int Garfield::ViewField::m_nSamples2dX = 200 [private]
Definition at line 175 of file ViewField.hh.

### m_nSamples2dY

unsigned int Garfield::ViewField::m_nSamples2dY = 200 [private]
Definition at line 176 of file ViewField.hh.

### m_samplePotential

bool Garfield::ViewField::m_samplePotential = true [private]
Definition at line 157 of file ViewField.hh.

m_sensor

Sensor* Garfield::ViewField::m_sensor = nullptr [private]
Definition at line 162 of file ViewField.hh.

m_useAutoRange

bool Garfield::ViewField::m_useAutoRange = true [private]
Definition at line 156 of file ViewField.hh.

m_useStatus

bool Garfield::ViewField::m_useStatus = false [private]
Definition at line 158 of file ViewField.hh.

m_vBkg

double Garfield::ViewField::m_vBkg = 0. [private]
Definition at line 159 of file ViewField.hh.

m_vmax

double Garfield::ViewField::m_vmax = 100. [private]
Definition at line 166 of file ViewField.hh.

m_vmin

double Garfield::ViewField::m_vmin = 0. [private]
Definition at line 166 of file ViewField.hh.

m_wmax

double Garfield::ViewField::m_wmax = 100. [private]
Definition at line 168 of file ViewField.hh.

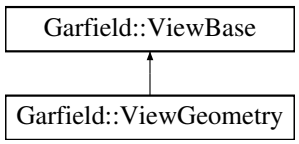
m_wmin

double Garfield::ViewField::m_wmin = 0. [private]
Definition at line 168 of file ViewField.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewField.hh

## 7.126 Garfield::ViewGeometry Class Reference

Visualize a geometry defined using the "native" shapes.
#include <ViewGeometry.hh>
Inheritance diagram for Garfield::ViewGeometry:




Public Member Functions

- ViewGeometry ()
  Default constructor.
- ViewGeometry (GeometrySimple *geo)

Constructor.

- ~ViewGeometry ()

  Destructor.
- void SetGeometry (GeometrySimple *geo)

  Set the geometry to be drawn.
- void Plot (const bool twod=false)

  Draw the geometry.
- void Plot2d ()

  Draw a cut through the geometry at the current viewing plane.
- void Plot3d ()

  Draw a three-dimensional view of the geometry.
- void PlotPanels ()

  Draw the surface panels.

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.
- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

  Set the viewing plane to x-z.
- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

Set the viewing plane to z-y.

- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Member Functions

- void Reset ()

Private Attributes

- GeometrySimple * m_geometry = nullptr
- std::vector< TGeoVolume * > m_volumes
- std::vector< TGeoMedium * > m_media
- std::unique_ptr< TGeoManager > m_geoManager

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.126.1 Detailed Description

Visualize a geometry defined using the "native" shapes.
Definition at line 17 of file ViewGeometry.hh.

### 7.126.2 Constructor & Destructor Documentation

ViewGeometry() [1/2]

Garfield::ViewGeometry::ViewGeometry () [inline]
Default constructor.
Definition at line 20 of file ViewGeometry.hh.

```
00020 : ViewGeometry(nullptr) {}
```

ViewGeometry() [2/2]

Garfield::ViewGeometry::ViewGeometry (
GeometrySimple * geo)
Constructor.

~ViewGeometry()

Garfield::ViewGeometry::~ViewGeometry ()
Destructor.

### 7.126.3 Member Function Documentation

Plot()

void Garfield::ViewGeometry::Plot (
const bool twod = false)
Draw the geometry.

Plot2d()

void Garfield::ViewGeometry::Plot2d ()
Draw a cut through the geometry at the current viewing plane.

Plot3d()

void Garfield::ViewGeometry::Plot3d ()
Draw a three-dimensional view of the geometry.

PlotPanels()

void Garfield::ViewGeometry::PlotPanels ()
Draw the surface panels.

Reset()

void Garfield::ViewGeometry::Reset () [private]

SetGeometry()

void Garfield::ViewGeometry::SetGeometry (
GeometrySimple * geo)
Set the geometry to be drawn.

### 7.126.4 Member Data Documentation

m_geoManager

std::unique_ptr<TGeoManager> Garfield::ViewGeometry::m_geoManager [private]
Definition at line 43 of file ViewGeometry.hh.

m_geometry

GeometrySimple* Garfield::ViewGeometry::m_geometry = nullptr [private]
Definition at line 38 of file ViewGeometry.hh.

m_media

std::vector<TGeoMedium*> Garfield::ViewGeometry::m_media [private]
Definition at line 41 of file ViewGeometry.hh.

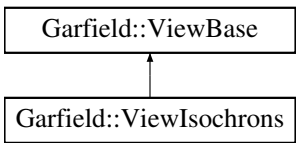
m_volumes

std::vector<TGeoVolume*> Garfield::ViewGeometry::m_volumes [private]
Definition at line 40 of file ViewGeometry.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewGeometry.hh

## 7.127 Garfield::ViewIsochrons Class Reference

Draw equal time contour lines.
#include <ViewIsochrons.hh>
Inheritance diagram for Garfield::ViewIsochrons:




Public Member Functions

- ViewIsochrons ()
  Constructor.
- ~ViewIsochrons ()=default
  Destructor.
- void SetSensor (Sensor *s)
  Set the sensor.

- void SetComponent (Component *c)

  Set the component.
- void PlotIsochrons (const double tstep, const std::vector< std::array< double, 3 > > &points, const bool rev=false, const bool colour=false, const bool markers=false, const bool plotDriftLines=true)

  Draw equal time contour lines.
- void DriftElectrons (const bool positive=false)

  Request electron drift lines with negative (default) or positive charge.
- void DriftIons (const bool negative=false)

  Request ion drift lines with positive (default) or negative charge.
- void EnableSorting (const bool on=true)

  Sort (or not) the points on a contour line (default: sorting is done).
- void CheckCrossings (const bool on=true)

  Check (or not) that drift-lines do not cross isochrons (default: check is done).
- void SetAspectRatioSwitch (const double ar)

  Set the aspect ratio above which an isochron is considered linear (as opposed to circular).
- void SetLoopThreshold (const double thr)

  Fractional distance between two points for closing a circular isochron (default: 0.2).
- void SetConnectionThreshold (const double thr)

  Fractional distance over which isochron segments are connected (default: 0.2).

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.
- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

Set the viewing plane to x-z.

- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

  Set the viewing plane to z-y.
- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Member Functions

- bool SetPlotLimits ()
- void ComputeDriftLines (const double tstep, const std::vector< std::array< double, 3 > > &points, std::vector< std::vector< std::array< double, 3 > > > &driftLines, std::vector< std::array< double, 3 > > &startPoints, std::vector< std::array< double, 3 > > &endPoints, std::vector< int > &statusCodes, const bool rev=false)
- void SortContour (std::vector< std::pair< std::array< double, 4 >, unsigned int > > &contour, bool &circle)

Private Attributes

- Sensor * m_sensor = nullptr
- Component * m_component = nullptr
- Particle m_particle = Particle::Electron
- bool m_positive = false
- short m_markerStyle = 5
- short m_lineStyle = 2
- bool m_sortContours = true
- double m_aspectRatio = 3.
- double m_loopThreshold = 0.2
- double m_connectionThreshold = 0.2
- bool m_checkCrossings = true

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()

- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.127.1 Detailed Description

Draw equal time contour lines.
Definition at line 16 of file ViewIsochrons.hh.

### 7.127.2 Constructor & Destructor Documentation

ViewIsochrons()

Garfield::ViewIsochrons::ViewIsochrons ()
Constructor.

~ViewIsochrons()

Garfield::ViewIsochrons::~ViewIsochrons () [default]
Destructor.

### 7.127.3 Member Function Documentation

CheckCrossings()

void Garfield::ViewIsochrons::CheckCrossings (
const bool on = true) [inline]
Check (or not) that drift-lines do not cross isochrons (default: check is done).
Definition at line 58 of file ViewIsochrons.hh.

```
00058 { m_checkCrossings = on; }
```

### ComputeDriftLines()

```
void Garfield::ViewIsochrons::ComputeDriftLines (
                const double tstep,
                const std::vector< std::array< double, 3 > > & points,
                std::vector< std::vector< std::array< double, 3 > > > & driftLines,
                std::vector< std::array< double, 3 > > & startPoints,
                std::vector< std::array< double, 3 > > & endPoints,
                std::vector< int > & statusCodes,
                const bool rev = false)   [private]
```

### DriftElectrons()

```
void Garfield::ViewIsochrons::DriftElectrons (
                const bool positive = false)   [inline]
```

Request electron drift lines with negative (default) or positive charge.
Definition at line 45 of file ViewIsochrons.hh.

```
00045                                          {
00046     m_particle = Particle::Electron;
00047     m_positive = positive;
00048   }
```

### DriftIons()

```
void Garfield::ViewIsochrons::DriftIons (
                const bool negative = false)   [inline]
```

Request ion drift lines with positive (default) or negative charge.
Definition at line 50 of file ViewIsochrons.hh.

```
00050                                          {
00051     m_particle = Particle::Ion;
00052     m_positive = !negative;
00053   }
```

### EnableSorting()

```
void Garfield::ViewIsochrons::EnableSorting (
                const bool on = true)   [inline]
```

Sort (or not) the points on a contour line (default: sorting is done).
Definition at line 55 of file ViewIsochrons.hh.

```
00055 { m_sortContours = on; }
```

### PlotIsochrons()

```
void Garfield::ViewIsochrons::PlotIsochrons (
                const double tstep,
                const std::vector< std::array< double, 3 > > & points,
                const bool rev = false,
                const bool colour = false,
                const bool markers = false,
                const bool plotDriftLines = true)
```

Draw equal time contour lines.

Parameters

| | |
|---|---|
| tstep | Time interval. |
| points | List of starting points. |
| rev | If true, the drift time is measured from the end points of the drift lines. |
| colour | Draw contour lines using colours. |
| markers | Draw markers (as opposed to lines). |
| plotDriftLines | Draw drift lines together with the isochrons. |

SetAspectRatioSwitch()

void Garfield::ViewIsochrons::SetAspectRatioSwitch (
const double ar)
Set the aspect ratio above which an isochron is considered linear (as opposed to circular).

SetComponent()

void Garfield::ViewIsochrons::SetComponent (
Component * c)
Set the component.

SetConnectionThreshold()

void Garfield::ViewIsochrons::SetConnectionThreshold (
const double thr)
Fractional distance over which isochron segments are connected (default: 0.2).

SetLoopThreshold()

void Garfield::ViewIsochrons::SetLoopThreshold (
const double thr)
Fractional distance between two points for closing a circular isochron (default: 0.2).

SetPlotLimits()

bool Garfield::ViewIsochrons::SetPlotLimits () [private]

SetSensor()

void Garfield::ViewIsochrons::SetSensor (
Sensor * s)
Set the sensor.

SortContour()

void Garfield::ViewIsochrons::SortContour (
std::vector< std::pair< std::array< double, 4 >, unsigned int > > & contour,
bool & circle) [private]

### 7.127.4 Member Data Documentation

m_aspectRatio

double Garfield::ViewIsochrons::m_aspectRatio = 3. [private]
Definition at line 81 of file ViewIsochrons.hh.

m_checkCrossings

bool Garfield::ViewIsochrons::m_checkCrossings = true [private]
Definition at line 84 of file ViewIsochrons.hh.

m_component

Component* Garfield::ViewIsochrons::m_component = nullptr [private]
Definition at line 71 of file ViewIsochrons.hh.

m_connectionThreshold

double Garfield::ViewIsochrons::m_connectionThreshold = 0.2 [private]
Definition at line 83 of file ViewIsochrons.hh.

m_lineStyle

short Garfield::ViewIsochrons::m_lineStyle = 2 [private]
Definition at line 78 of file ViewIsochrons.hh.

m_loopThreshold

double Garfield::ViewIsochrons::m_loopThreshold = 0.2 [private]
Definition at line 82 of file ViewIsochrons.hh.

m_markerStyle

short Garfield::ViewIsochrons::m_markerStyle = 5 [private]
Definition at line 77 of file ViewIsochrons.hh.

m_particle

Particle Garfield::ViewIsochrons::m_particle = Particle::Electron [private]
Definition at line 74 of file ViewIsochrons.hh.

m_positive

bool Garfield::ViewIsochrons::m_positive = false [private]
Definition at line 75 of file ViewIsochrons.hh.

m_sensor

Sensor* Garfield::ViewIsochrons::m_sensor = nullptr [private]
Definition at line 70 of file ViewIsochrons.hh.

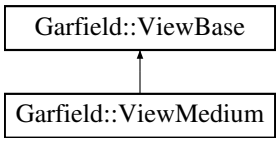
m_sortContours

bool Garfield::ViewIsochrons::m_sortContours = true [private]
Definition at line 80 of file ViewIsochrons.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewIsochrons.hh

## 7.128 Garfield::ViewMedium Class Reference

Plot transport coefficients as function of electric and magnetic field.
#include <ViewMedium.hh>
Inheritance diagram for Garfield::ViewMedium:




Public Member Functions

- ViewMedium ()
  Default constructor.
- ViewMedium (Medium *medium)
  Constructor.
- ~ViewMedium ()=default
  Destructor.
- void SetMedium (Medium *m)
  Set the medium from which to retrieve the transport coefficients.

- void EnableAutoRangeX (const bool on=true)

  Try to choose the x-axis range based on the field grid.
- void SetRangeE (const double emin, const double emax, const bool logscale)

  Set the limits of the electric field.
- void SetRangeB (const double bmin, const double bmax, const bool logscale)

  Set the limits of the magnetic field.
- void SetRangeA (const double amin, const double amax, const bool logscale)

  Set the limits of the angle between electric and magnetic field.
- void SetRangeEN (const double emin, const double emax, const bool logscale)

  Set the limits of the reduced electric field (E/N).
- void SetRangeEP (const double emin, const double emax, const bool logscale)

  Set the limits of the reduced electric field (E/P).
- void EnableAutoRangeY (const bool on=true)

  Choose the y-axis range based on the function's minima/maxima.
- void SetRangeY (const double ymin, const double ymax, const bool logscale=false)

  Set the range of the function (velocity etc.) to be plotted.
- void SetElectricField (const double efield)

  Set the electric field to use when plotting as function of B or angle.
- void SetMagneticField (const double bfield)

  Set the magnetic field to use when plotting as function of E or angle.
- void SetAngle (const double angle)

  Set the angle to use when plotting as function of E or B.
- void EnableExport (const std::string &txtfile)
- void DisableExport ()
- void PlotVelocity (const std::string &carriers, const char xaxis)

  Plot the drift velocity components.
- void PlotDiffusion (const std::string &carriers, const char xaxis)

  Plot the transverse and longitudinal diffusion coefficients.
- void PlotTownsend (const std::string &carriers, const char xaxis)

  Plot the Townsend coefficient.
- void PlotAttachment (const std::string &carriers, const char xaxis)

  Plot the attachment coefficient.
- void PlotAlphaEta (const std::string &carriers, const char xaxis)

  Plot Townsend and attachment coefficients.
- void PlotElectronVelocity (const char xaxis='e', const bool same=false)

  Plot the drift velocity components of electrons in the medium.
- void PlotElectronVelocityFluxBulk (const char xaxis='e', const bool same=false)

  Plot Flux and Bulk drift velocity.
- void PlotHoleVelocity (const char xaxis='e', const bool same=false)

  Plot the drift velocity components of holes in the medium.
- void PlotIonVelocity (const char xaxis='e', const bool same=false)

  Plot the ion drift velocity in the gas.
- void PlotElectronDiffusion (const char xaxis='e', const bool same=false)

  Plot the diffusion coefficients in the medium.
- void PlotHoleDiffusion (const char xaxis='e', const bool same=false)

  Plot the diffusion coefficients of holes in the medium.
- void PlotIonDiffusion (const char xaxis='e', const bool same=false)

  Plot the diffusion coefficients of ions in the gas.
- void PlotElectronTownsend (const char xaxis='e', const bool same=false)

  Plot the Townsend coefficient for electrons.
- void PlotElectronReducedTownsendN (const char xaxis='r', const bool same=false)

Plot the Townsend coefficient for electrons.

- void PlotElectronReducedTownsendP (const char xaxis='p', const bool same=false)

  Plot the Townsend coefficient for electrons.
- void PlotElectronTOFIonization (const char xaxis='e', const bool same=false)

  Plot the TOF ionization rate.
- void PlotHoleTownsend (const char xaxis='e', const bool same=false)

  Plot the Townsend coefficient for holes.
- void PlotElectronAttachment (const char xaxis='e', const bool same=false)

  Plot the attachment coefficient for electrons.
- void PlotElectronTOFAttachment (const char xaxis='e', const bool same=false)

  Plot the TOF attachment rate.
- void PlotHoleAttachment (const char xaxis='e', const bool same=false)

  Plot the attachment coefficient for holes.
- void PlotElectronLorentzAngle (const char xaxis='e', const bool same=false)

  Plot the angle between drift velocity and field.
- void SetColours (const std::vector< short > &cols)

  Set the (ROOT) colours to be used in the plots.
- void SetLabels (const std::vector< std::string > &labels)

  Set user-defined plot labels.

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.
- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).
- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

Set the viewing plane to x-z.

- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

  Set the viewing plane to z-y.
- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Types

- enum class Parameter {
  VelocityE , VelocityB , VelocityExB , VelocityWv ,
  VelocityWr , TransverseDiffusion , LongitudinalDiffusion , Townsend ,
  Attachment , LorentzAngle , RIonTof , RAttTof ,
  AlphaN , AlphaP }
- enum class Charge { Electron , Hole , Ion }
- enum class Axis {
  E , B , Angle , EoverN ,
  EoverP , None }

Private Member Functions

- void PlotVelocity (const Axis xaxis, const Charge particle, const bool same)
- void PlotVelocityFluxBulk (const Axis xaxis, const Charge particle, const bool same)
- void PlotDiffusion (const Axis xaxis, const Charge particle, const bool same)
- void Plot (const Axis xaxis, const Charge particle, const Parameter par, const bool same)
- void PlotLorentzAngle (const Axis xaxis, const Charge particle, const bool same)
- void ResetY ()
- void ResetX (const Axis xaxis)
- void Draw ()
- void Export ()
- Axis GetAxis (const char xaxis) const
- bool GetGrid (std::array< std::vector< double >, 3 > &grid, int &ie, int &ib, int &ia, const Axis xaxis) const
- double ConvertToEN (const double e0Vcm)
- double UnConvertFromEN (const double e0Td)
- double ConvertToEP (const double e0Vcm)
- double UnConvertFromEP (const double e0VcmTorr)

Private Attributes

- Medium * m_medium = nullptr
- double m_eMin = 100.
- double m_eMax = 100000.
- double m_bMin = 0.
- double m_bMax = 2.
- double m_aMin = 0.
- double m_aMax = Pi
- double m_enMin = 0.25
- double m_enMax = 400.
- double m_epMin = 0.13
- double m_epMax = 132.
- bool m_logE = true
- bool m_logB = false

- bool m_logA = false
- bool m_logEN = true
- bool m_logEP = true
- bool m_logX = true
- bool m_autoRangeX = true
- Axis m_xaxis = Axis::None
- double m_yMin = 0.
- double m_yMax = 1.
- bool m_logY = false
- bool m_autoRangeY = true
- double m_efield = 1000.
- double m_bfield = 0.
- double m_angle = HalfPi
- std::vector< double > m_xPlot
- std::vector< std::vector< double > > m_yPlot
- std::vector< Parameter > m_par
- std::vector< Charge > m_q
- std::vector< std::vector< double > > m_xGraph
- std::vector< std::vector< double > > m_yGraph
- std::vector< short > m_colours
- std::vector< std::string > m_labels
- std::string m_outfile

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()
- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.128.1 Detailed Description

Plot transport coefficients as function of electric and magnetic field.
Definition at line 17 of file ViewMedium.hh.

### 7.128.2 Member Enumeration Documentation

#### Axis

enum class Garfield::ViewMedium::Axis [strong], [private]

Enumerator

| | |
|---|---|
| E | |
| B | |
| Angle | |
| EoverN | |
| EoverP | |
| None | |

Definition at line 186 of file ViewMedium.hh.
00186 { E, B, Angle, EoverN, EoverP, None };

#### Charge

enum class Garfield::ViewMedium::Charge [strong], [private]

Enumerator

| | |
|---|---|
| Electron | |
| Hole | |
| Ion | |

Definition at line 184 of file ViewMedium.hh.

```
00184 { Electron, Hole, Ion };
```

### Parameter

enum class Garfield::ViewMedium::Parameter [strong], [private]

Enumerator

| | |
|---|---|
| VelocityE | |
| VelocityB | |
| VelocityExB | |
| VelocityWv | |
| VelocityWr | |
| TransverseDiffusion | |
| LongitudinalDiffusion | |
| Townsend | |
| Attachment | |
| LorentzAngle | |
| RIonTof | |
| RAttTof | |
| AlphaN | |
| AlphaP | |

Definition at line 167 of file ViewMedium.hh.

```
00167                 {
00168     VelocityE,
00169     VelocityB,
00170     VelocityExB,
00171     VelocityWv,
00172     VelocityWr,
00173     TransverseDiffusion,
00174     LongitudinalDiffusion,
00175     Townsend,
00176     Attachment,
00177     LorentzAngle,
00178     RIonTof,
00179     RAttTof,
00180     AlphaN,
00181     AlphaP
00182   };
```

## 7.128.3 Constructor & Destructor Documentation

### ViewMedium() [1/2]

Garfield::ViewMedium::ViewMedium () [inline]

Default constructor.

Definition at line 20 of file ViewMedium.hh.

```
00020 : ViewMedium(nullptr) {}
```

### ViewMedium() [2/2]

Garfield::ViewMedium::ViewMedium (
Medium * medium)

Constructor.

### ~ViewMedium()

Garfield::ViewMedium::~ViewMedium () [default]
Destructor.

## 7.128.4 Member Function Documentation

### ConvertToEN()

double Garfield::ViewMedium::ConvertToEN (
const double e0Vcm) [private]

### ConvertToEP()

double Garfield::ViewMedium::ConvertToEP (
const double e0Vcm) [private]

### DisableExport()

void Garfield::ViewMedium::DisableExport () [inline]
Definition at line 55 of file ViewMedium.hh.
00055 { m_outfile = ""; }

### Draw()

void Garfield::ViewMedium::Draw () [private]

### EnableAutoRangeX()

void Garfield::ViewMedium::EnableAutoRangeX (
const bool on = true) [inline]
Try to choose the x-axis range based on the field grid.
Definition at line 30 of file ViewMedium.hh.
00030 { m_autoRangeX = on; }

### EnableAutoRangeY()

void Garfield::ViewMedium::EnableAutoRangeY (
const bool on = true) [inline]
Choose the y-axis range based on the function's minima/maxima.
Definition at line 42 of file ViewMedium.hh.
00042 { m_autoRangeY = on; }

### EnableExport()

void Garfield::ViewMedium::EnableExport (
const std::string & txtfile) [inline]
Definition at line 54 of file ViewMedium.hh.
00054 { m_outfile = txtfile; }

### Export()

void Garfield::ViewMedium::Export () [private]

### GetAxis()

Axis Garfield::ViewMedium::GetAxis (
const char xaxis) const [private]

### GetGrid()

```
bool Garfield::ViewMedium::GetGrid (
                std::array< std::vector< double >, 3 > & grid,
                int & ie,
                int & ib,
                int & ia,
                const Axis xaxis) const    [private]
```

### Plot()

```
void Garfield::ViewMedium::Plot (
                const Axis xaxis,
                const Charge particle,
                const Parameter par,
                const bool same)    [private]
```

### PlotAlphaEta()

```
void Garfield::ViewMedium::PlotAlphaEta (
                const std::string & carriers,
                const char xaxis)
```

Plot Townsend and attachment coefficients.

### PlotAttachment()

```
void Garfield::ViewMedium::PlotAttachment (
                const std::string & carriers,
                const char xaxis)
```

Plot the attachment coefficient.

### PlotDiffusion() [1/2]

```
void Garfield::ViewMedium::PlotDiffusion (
                const Axis xaxis,
                const Charge particle,
                const bool same)    [private]
```

### PlotDiffusion() [2/2]

```
void Garfield::ViewMedium::PlotDiffusion (
                const std::string & carriers,
                const char xaxis)
```

Plot the transverse and longitudinal diffusion coefficients.

### PlotElectronAttachment()

```
void Garfield::ViewMedium::PlotElectronAttachment (
                const char xaxis = 'e',
                const bool same = false)    [inline]
```

Plot the attachment coefficient for electrons.
Definition at line 142 of file ViewMedium.hh.

```
00142                                                                {
00143     Plot(GetAxis(xaxis), Charge::Electron, Parameter::Attachment, same);
00144   }
```

### PlotElectronDiffusion()

```
void Garfield::ViewMedium::PlotElectronDiffusion (
                const char xaxis = 'e',
                const bool same = false)    [inline]
```

Plot the diffusion coefficients in the medium.

Definition at line 107 of file ViewMedium.hh.

```
00107                                                                 {
00108     PlotDiffusion(GetAxis(xaxis), Charge::Electron, same);
00109   }
```

### PlotElectronLorentzAngle()

```
void Garfield::ViewMedium::PlotElectronLorentzAngle (
                const char xaxis = 'e',
                const bool same = false)   [inline]
```

Plot the angle between drift velocity and field.
Definition at line 156 of file ViewMedium.hh.

```
00157                                                {
00158     PlotLorentzAngle(GetAxis(xaxis), Charge::Electron, same);
00159   }
```

### PlotElectronReducedTownsendN()

```
void Garfield::ViewMedium::PlotElectronReducedTownsendN (
                const char xaxis = 'r',
                const bool same = false)   [inline]
```

Plot the Townsend coefficient for electrons.
Definition at line 123 of file ViewMedium.hh.

```
00124                                                    {
00125     Plot(GetAxis(xaxis), Charge::Electron, Parameter::AlphaN, same);
00126   }
```

### PlotElectronReducedTownsendP()

```
void Garfield::ViewMedium::PlotElectronReducedTownsendP (
                const char xaxis = 'p',
                const bool same = false)   [inline]
```

Plot the Townsend coefficient for electrons.
Definition at line 128 of file ViewMedium.hh.

```
00129                                                    {
00130     Plot(GetAxis(xaxis), Charge::Electron, Parameter::AlphaP, same);
00131   }
```

### PlotElectronTOFAttachment()

```
void Garfield::ViewMedium::PlotElectronTOFAttachment (
                const char xaxis = 'e',
                const bool same = false)   [inline]
```

Plot the TOF attachment rate.
Definition at line 146 of file ViewMedium.hh.

```
00147                                                    {
00148     Plot(GetAxis(xaxis), Charge::Electron, Parameter::RAttTof, same);
00149   }
```

### PlotElectronTOFIonization()

```
void Garfield::ViewMedium::PlotElectronTOFIonization (
                const char xaxis = 'e',
                const bool same = false)   [inline]
```

Plot the TOF ionization rate.
Definition at line 133 of file ViewMedium.hh.

```
00134                                                    {
00135     Plot(GetAxis(xaxis), Charge::Electron, Parameter::RIonTof, same);
00136   }
```

### PlotElectronTownsend()

```
void Garfield::ViewMedium::PlotElectronTownsend (
                const char xaxis = 'e',
```

const bool same = false) [inline]

Plot the Townsend coefficient for electrons.

Definition at line 119 of file ViewMedium.hh.

```
00119                                                                 {
00120     Plot(GetAxis(xaxis), Charge::Electron, Parameter::Townsend, same);
00121   }
```

### PlotElectronVelocity()

void Garfield::ViewMedium::PlotElectronVelocity (
const char xaxis = 'e',
const bool same = false) [inline]

Plot the drift velocity components of electrons in the medium.

Parameters

| | |
|---|---|
| xaxis | abscissa.<br>• 'e': electric field,<br>• 'b': magnetic field,<br>• 'a': angle between E and B,<br>• 'r': reduced electric field (E/N),<br>• 'p': reduced electric field (E/p). |
| same | flag to keep existing plots (true) or not. |

Definition at line 90 of file ViewMedium.hh.

```
00090                                                                 {
00091     PlotVelocity(GetAxis(xaxis), Charge::Electron, same);
00092   }
```

### PlotElectronVelocityFluxBulk()

void Garfield::ViewMedium::PlotElectronVelocityFluxBulk (
const char xaxis = 'e',
const bool same = false) [inline]

Plot Flux and Bulk drift velocity.

Definition at line 94 of file ViewMedium.hh.

```
00095                                                                 {
00096     PlotVelocityFluxBulk(GetAxis(xaxis), Charge::Electron, same);
00097   }
```

### PlotHoleAttachment()

void Garfield::ViewMedium::PlotHoleAttachment (
const char xaxis = 'e',
const bool same = false) [inline]

Plot the attachment coefficient for holes.

Definition at line 151 of file ViewMedium.hh.

```
00151                                                                 {
00152     Plot(GetAxis(xaxis), Charge::Hole, Parameter::Attachment, same);
00153   }
```

### PlotHoleDiffusion()

void Garfield::ViewMedium::PlotHoleDiffusion (
const char xaxis = 'e',
const bool same = false) [inline]

Plot the diffusion coefficients of holes in the medium.

Definition at line 111 of file ViewMedium.hh.

```
00111                                                 {
00112     PlotDiffusion(GetAxis(xaxis), Charge::Hole, same);
00113   }
```

### PlotHoleTownsend()

```
void Garfield::ViewMedium::PlotHoleTownsend (
                const char xaxis = 'e',
                const bool same = false)    [inline]
```

Plot the Townsend coefficient for holes.
Definition at line 138 of file ViewMedium.hh.

```
00138                                                 {
00139     Plot(GetAxis(xaxis), Charge::Hole, Parameter::Townsend, same);
00140   }
```

### PlotHoleVelocity()

```
void Garfield::ViewMedium::PlotHoleVelocity (
                const char xaxis = 'e',
                const bool same = false)    [inline]
```

Plot the drift velocity components of holes in the medium.
Definition at line 99 of file ViewMedium.hh.

```
00099                                                 {
00100     PlotVelocity(GetAxis(xaxis), Charge::Hole, same);
00101   }
```

### PlotIonDiffusion()

```
void Garfield::ViewMedium::PlotIonDiffusion (
                const char xaxis = 'e',
                const bool same = false)    [inline]
```

Plot the diffusion coefficients of ions in the gas.
Definition at line 115 of file ViewMedium.hh.

```
00115                                                 {
00116     PlotDiffusion(GetAxis(xaxis), Charge::Ion, same);
00117   }
```

### PlotIonVelocity()

```
void Garfield::ViewMedium::PlotIonVelocity (
                const char xaxis = 'e',
                const bool same = false)    [inline]
```

Plot the ion drift velocity in the gas.
Definition at line 103 of file ViewMedium.hh.

```
00103                                                 {
00104     PlotVelocity(GetAxis(xaxis), Charge::Ion, same);
00105   }
```

### PlotLorentzAngle()

```
void Garfield::ViewMedium::PlotLorentzAngle (
                const Axis xaxis,
                const Charge particle,
                const bool same)    [private]
```

### PlotTownsend()

```
void Garfield::ViewMedium::PlotTownsend (
                const std::string & carriers,
                const char xaxis)
```

Plot the Townsend coefficient.

### PlotVelocity() [1/2]

void Garfield::ViewMedium::PlotVelocity (
const Axis xaxis,
const Charge particle,
const bool same) [private]

### PlotVelocity() [2/2]

void Garfield::ViewMedium::PlotVelocity (
const std::string & carriers,
const char xaxis)

Plot the drift velocity components.

Parameters

| | |
|---|---|
| carriers | string indicating the carriers for which to plot the drift velocity<br>• ”e”: electrons,<br>• ”h”: holes,<br>• ”i”: ions Options can be concatenated (e. g. ”ei”, ”eh”). |
| xaxis | abscissa.<br>• 'e': electric field,<br>• 'b': magnetic field,<br>• 'a': angle between E and B,<br>• 'r': reduced electric field (E/N),<br>• 'p': reduced electric field (E/p) |

### PlotVelocityFluxBulk()

void Garfield::ViewMedium::PlotVelocityFluxBulk (
const Axis xaxis,
const Charge particle,
const bool same) [private]

### ResetX()

void Garfield::ViewMedium::ResetX (
const Axis xaxis) [private]

### ResetY()

void Garfield::ViewMedium::ResetY () [private]

### SetAngle()

void Garfield::ViewMedium::SetAngle (
const double angle) [inline]

Set the angle to use when plotting as function of E or B.
Definition at line 52 of file ViewMedium.hh.

```
00052 { m_angle = angle; }
```

SetColours()

void Garfield::ViewMedium::SetColours (
const std::vector< short > & cols) [inline]
Set the (ROOT) colours to be used in the plots.
Definition at line 162 of file ViewMedium.hh.
00162 { m_colours = cols; }

SetElectricField()

void Garfield::ViewMedium::SetElectricField (
const double efield) [inline]
Set the electric field to use when plotting as function of B or angle.
Definition at line 48 of file ViewMedium.hh.
00048 { m_efield = efield; }

SetLabels()

void Garfield::ViewMedium::SetLabels (
const std::vector< std::string > & labels) [inline]
Set user-defined plot labels.
Definition at line 164 of file ViewMedium.hh.
00164 { m_labels = labels; }

SetMagneticField()

void Garfield::ViewMedium::SetMagneticField (
const double bfield) [inline]
Set the magnetic field to use when plotting as function of E or angle.
Definition at line 50 of file ViewMedium.hh.
00050 { m_bfield = bfield; }

SetMedium()

void Garfield::ViewMedium::SetMedium (
Medium * m)
Set the medium from which to retrieve the transport coefficients.

SetRangeA()

void Garfield::ViewMedium::SetRangeA (
const double amin,
const double amax,
const bool logscale)
Set the limits of the angle between electric and magnetic field.

SetRangeB()

void Garfield::ViewMedium::SetRangeB (
const double bmin,
const double bmax,
const bool logscale)
Set the limits of the magnetic field.

SetRangeE()

void Garfield::ViewMedium::SetRangeE (
const double emin,
const double emax,
const bool logscale)
Set the limits of the electric field.

SetRangeEN()

void Garfield::ViewMedium::SetRangeEN (
const double emin,
const double emax,
const bool logscale)
Set the limits of the reduced electric field (E/N).

SetRangeEP()

void Garfield::ViewMedium::SetRangeEP (
const double emin,
const double emax,
const bool logscale)
Set the limits of the reduced electric field (E/P).

SetRangeY()

void Garfield::ViewMedium::SetRangeY (
const double ymin,
const double ymax,
const bool logscale = false)
Set the range of the function (velocity etc.) to be plotted.

UnConvertFromEN()

double Garfield::ViewMedium::UnConvertFromEN (
const double e0Td) [private]

UnConvertFromEP()

double Garfield::ViewMedium::UnConvertFromEP (
const double e0VcmTorr) [private]

### 7.128.5 Member Data Documentation

m_aMax

double Garfield::ViewMedium::m_aMax = Pi [private]
Definition at line 193 of file ViewMedium.hh.

m_aMin

double Garfield::ViewMedium::m_aMin = 0. [private]
Definition at line 193 of file ViewMedium.hh.

m_angle

double Garfield::ViewMedium::m_angle = HalfPi [private]
Definition at line 215 of file ViewMedium.hh.

m_autoRangeX

bool Garfield::ViewMedium::m_autoRangeX = true [private]
Definition at line 202 of file ViewMedium.hh.

m_autoRangeY

bool Garfield::ViewMedium::m_autoRangeY = true [private]
Definition at line 208 of file ViewMedium.hh.

m_bfield

double Garfield::ViewMedium::m_bfield = 0. [private]
Definition at line 213 of file ViewMedium.hh.

m_bMax

double Garfield::ViewMedium::m_bMax = 2. [private]
Definition at line 192 of file ViewMedium.hh.

m_bMin

double Garfield::ViewMedium::m_bMin = 0. [private]
Definition at line 192 of file ViewMedium.hh.

m_colours

std::vector<short> Garfield::ViewMedium::m_colours [private]
Definition at line 225 of file ViewMedium.hh.

m_efield

double Garfield::ViewMedium::m_efield = 1000. [private]
Definition at line 211 of file ViewMedium.hh.

m_eMax

double Garfield::ViewMedium::m_eMax = 100000. [private]
Definition at line 191 of file ViewMedium.hh.

m_eMin

double Garfield::ViewMedium::m_eMin = 100. [private]
Definition at line 191 of file ViewMedium.hh.

m_enMax

double Garfield::ViewMedium::m_enMax = 400. [private]
Definition at line 194 of file ViewMedium.hh.

m_enMin

double Garfield::ViewMedium::m_enMin = 0.25 [private]
Definition at line 194 of file ViewMedium.hh.

m_epMax

double Garfield::ViewMedium::m_epMax = 132. [private]
Definition at line 195 of file ViewMedium.hh.

m_epMin

double Garfield::ViewMedium::m_epMin = 0.13 [private]
Definition at line 195 of file ViewMedium.hh.

m_labels

std::vector<std::string> Garfield::ViewMedium::m_labels [private]
Definition at line 226 of file ViewMedium.hh.

### m_logA

bool Garfield::ViewMedium::m_logA = false [private]
Definition at line 198 of file ViewMedium.hh.

### m_logB

bool Garfield::ViewMedium::m_logB = false [private]
Definition at line 197 of file ViewMedium.hh.

### m_logE

bool Garfield::ViewMedium::m_logE = true [private]
Definition at line 196 of file ViewMedium.hh.

### m_logEN

bool Garfield::ViewMedium::m_logEN = true [private]
Definition at line 199 of file ViewMedium.hh.

### m_logEP

bool Garfield::ViewMedium::m_logEP = true [private]
Definition at line 200 of file ViewMedium.hh.

### m_logX

bool Garfield::ViewMedium::m_logX = true [private]
Definition at line 201 of file ViewMedium.hh.

### m_logY

bool Garfield::ViewMedium::m_logY = false [private]
Definition at line 207 of file ViewMedium.hh.

### m_medium

Medium* Garfield::ViewMedium::m_medium = nullptr [private]
Definition at line 188 of file ViewMedium.hh.

### m_outfile

std::string Garfield::ViewMedium::m_outfile [private]
Definition at line 228 of file ViewMedium.hh.

### m_par

std::vector<Parameter> Garfield::ViewMedium::m_par [private]
Definition at line 219 of file ViewMedium.hh.

### m_q

std::vector<Charge> Garfield::ViewMedium::m_q [private]
Definition at line 220 of file ViewMedium.hh.

### m_xaxis

Axis Garfield::ViewMedium::m_xaxis = Axis::None [private]
Definition at line 203 of file ViewMedium.hh.

m_xGraph

std::vector<std::vector<double> > Garfield::ViewMedium::m_xGraph [private]
Definition at line 222 of file ViewMedium.hh.

m_xPlot

std::vector<double> Garfield::ViewMedium::m_xPlot [private]
Definition at line 217 of file ViewMedium.hh.

m_yGraph

std::vector<std::vector<double> > Garfield::ViewMedium::m_yGraph [private]
Definition at line 223 of file ViewMedium.hh.

m_yMax

double Garfield::ViewMedium::m_yMax = 1. [private]
Definition at line 206 of file ViewMedium.hh.

m_yMin

double Garfield::ViewMedium::m_yMin = 0. [private]
Definition at line 206 of file ViewMedium.hh.

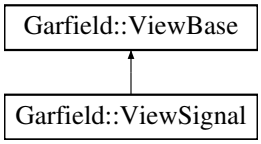
m_yPlot

std::vector<std::vector<double> > Garfield::ViewMedium::m_yPlot [private]
Definition at line 218 of file ViewMedium.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewMedium.hh

## 7.129 Garfield::ViewSignal Class Reference

Plot the signal computed by a sensor as a ROOT histogram.
#include <ViewSignal.hh>
Inheritance diagram for Garfield::ViewSignal:




Public Member Functions

- ViewSignal ()
  Default constructor.
- ViewSignal (Sensor *sensor)
  Constructor.
- ~ViewSignal ()=default
  Destructor.
- void SetSensor (Sensor *s)
  Set the sensor from which to retrieve the signal.
- void PlotSignal (const std::string &label, const std::string &optTotal="t", const std::string &optPrompt="", const std::string &optDelayed="", const bool same=false)
  Plot the signal.
- void SetRangeX (const double xmin, const double xmax)

Set the x-axis limits explicitly.

- void UnsetRangeX ()

  Remove the user-defined x-axis limits.
- void SetRangeY (const double ymin, const double ymax)

  Set the y-axis limits explicitly.
- void UnsetRangeY ()

  Remove the user-defined y-axis limits.
- void SetLabelY (const std::string &label)

  Override the default y-axis label.
- void EnableLegend (const bool on=true)

  Draw a legend on the plot or not.
- void SetColourTotal (const short col)

  Set the (ROOT) colour with which to draw the total signal.
- void SetColourElectrons (const short col)

  Set the (ROOT) colour with which to draw the electron component.
- void SetColourIons (const short col)

  Set the (ROOT) colour with which to draw the hole/ion component.
- void SetColourHoles (const short col)

  Set the (ROOT) colour with which to draw the hole/ion component.
- void SetColourDelayed (const short colTotal, const short colElectrons=kYellow - 7, const short colIons=kRed - 9)

  Set the (ROOT) colours with which to draw the delayed signal(s).

Public Member Functions inherited from Garfield::ViewBase

- ViewBase ()=delete

  Default constructor.
- ViewBase (const std::string &name)

  Constructor.
- virtual ~ViewBase ()=default

  Destructor.
- void SetCanvas (TPad *pad)

  Set the canvas to be painted on.
- void SetCanvas ()

  Unset an external canvas.
- TPad * GetCanvas ()

  Retrieve the canvas.
- void SetArea (const double xmin, const double ymin, const double xmax, const double ymax)

  Set the x- and y-axis limits (in local coordinates of the current viewing plane, if applicable).
- virtual void SetArea (const double xmin, const double ymin, const double zmin, const double xmax, const double ymax, const double zmax)

  Set a bounding box (if applicable).
- void SetArea ()

  Use default x- and y-axis limits (based on the bounding box of the sensor/component, if applicable).
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0)

  Set the projection (viewing plane), if applicable.
- virtual void SetPlane (const double fx, const double fy, const double fz, const double x0, const double y0, const double z0, const double hx, const double hy, const double hz)

  Set the projection plane specifying a hint for the in-plane x axis.
- void Rotate (const double angle)

  Rotate the viewing plane (angle in radian).

- void SetPlaneXY ()

  Set the viewing plane to x-y.
- void SetPlaneXZ ()

  Set the viewing plane to x-z.
- void SetPlaneYZ ()

  Set the viewing plane to y-z.
- void SetPlaneZX ()

  Set the viewing plane to z-x.
- void SetPlaneZY ()

  Set the viewing plane to z-y.
- void EnableDebugging (const bool on=true)

  Switch on/off debugging output.

Private Member Functions

- TH1 * DrawHistogram (TH1D &h, const std::string &opt, const std::string &ylabel)

Private Attributes

- Sensor * m_sensor = nullptr
- double m_xmin = 0.
- double m_xmax = 0.
- bool m_userRangeX = false
- double m_ymin = 0.
- double m_ymax = 0.
- bool m_userRangeY = false
- std::string m_labelY = ""
- bool m_legend = false
- short m_colTotal = kBlue + 3
- short m_colElectrons = kOrange - 3
- short m_colIons = kRed + 1
- std::array< short, 6 > m_colDelayed
- std::array< short, 3 > m_colPrompt {{kAzure + 10, kRed - 4, kMagenta + 2}}

Additional Inherited Members

Static Public Member Functions inherited from Garfield::ViewBase

- static std::string FindUnusedFunctionName (const std::string &s)

  Find an unused function name.
- static std::string FindUnusedHistogramName (const std::string &s)

  Find an unused histogram name.
- static std::string FindUnusedCanvasName (const std::string &s)

  Find an unused canvas name.

Protected Member Functions inherited from Garfield::ViewBase

- void UpdateProjectionMatrix ()
- template<typename T>
  void ToPlane (const T x, const T y, const T z, T &xp, T &yp) const
- template<typename T>
  bool InBox (const std::array< T, 3 > &x) const
- void Clip (const std::array< float, 3 > &x0, const std::array< float, 3 > &x1, std::array< float, 3 > &xc) const
- void DrawLine (const std::vector< std::array< float, 3 > > &xl, const short col, const short lw)
- std::string LabelX ()

- std::string LabelY ()
- std::string PlaneDescription ()
- bool PlotLimits (Sensor *sensor, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (Component *cmp, double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimitsFromUserBox (double &xmin, double &ymin, double &xmax, double &ymax) const
- bool PlotLimits (std::array< double, 3 > &bbmin, std::array< double, 3 > &bbmax, double &xmin, double &ymin, double &xmax, double &ymax) const

Static Protected Member Functions inherited from Garfield::ViewBase

- static bool RangeSet (TVirtualPad *)
- static void SetRange (TVirtualPad *pad, const double x0, const double y0, const double x1, const double y1)

Protected Attributes inherited from Garfield::ViewBase

- std::string m_className = "ViewBase"
- bool m_debug = false
- bool m_userPlotLimits = false
- double m_xMinPlot = -1.
- double m_xMaxPlot = 1.
- double m_yMinPlot = -1.
- double m_yMaxPlot = 1.
- bool m_userBox = false
- double m_xMinBox = -1.
- double m_xMaxBox = 1.
- double m_yMinBox = -1.
- double m_yMaxBox = 1.
- double m_zMinBox = -1.
- double m_zMaxBox = 1.
- std::array< std::array< double, 3 >, 3 > m_proj
- std::array< double, 4 > m_plane {{0, 0, 1, 0}}
- std::array< std::array< double, 3 >, 3 > m_prmat

### 7.129.1 Detailed Description

Plot the signal computed by a sensor as a ROOT histogram.
Definition at line 20 of file ViewSignal.hh.

### 7.129.2 Constructor & Destructor Documentation

ViewSignal() [1/2]

Garfield::ViewSignal::ViewSignal () [inline]
Default constructor.
Definition at line 23 of file ViewSignal.hh.

```
00023 : ViewSignal(nullptr) {}
```

ViewSignal() [2/2]

Garfield::ViewSignal::ViewSignal (
Sensor * sensor)
Constructor.

~ViewSignal()

Garfield::ViewSignal::~ViewSignal () [default]
Destructor.

### 7.129.3 Member Function Documentation

#### DrawHistogram()

```
TH1 * Garfield::ViewSignal::DrawHistogram (
            TH1D & h,
            const std::string & opt,
            const std::string & ylabel)   [private]
```

#### EnableLegend()

```
void Garfield::ViewSignal::EnableLegend (
            const bool on = true)   [inline]
```

Draw a legend on the plot or not.

Definition at line 67 of file ViewSignal.hh.

```
00067 { m_legend = on; }
```

#### PlotSignal()

```
void Garfield::ViewSignal::PlotSignal (
            const std::string & label,
            const std::string & optTotal = "t",
            const std::string & optPrompt = "",
            const std::string & optDelayed = "",
            const bool same = false)
```

Plot the signal.

Parameters

| | |
|---|---|
| label | Identifier (weighting field) of the signal to be plotted. |
| optTotal | String containing information about the total signals you want to plot. The syntax is the first letter of the charge carrier signal component you want to plot: "t" for total, "e" for electron and "i" for ion/hole. "tei" enables all three components. The total signal is always plotted. |
| optPrompt | String containing information about the prompt signal components you want to plot. The syntax is identical to the one described above. |
| optDelayed | String containing information about the delayed signal components you want to plot. The syntax is identical to the one described above. |
| same | Flag to keep existing plots on the canvas or not. |

#### SetColourDelayed()

```
void Garfield::ViewSignal::SetColourDelayed (
            const short colTotal,
            const short colElectrons = kYellow - 7,
            const short colIons = kRed - 9)   [inline]
```

Set the (ROOT) colours with which to draw the delayed signal(s).

Definition at line 79 of file ViewSignal.hh.

```
00081                                                 {
00082     m_colDelayed = {colTotal, colElectrons, colIons};
00083   }
```

#### SetColourElectrons()

```
void Garfield::ViewSignal::SetColourElectrons (
            const short col)   [inline]
```

Set the (ROOT) colour with which to draw the electron component.

Definition at line 72 of file ViewSignal.hh.

```
00072 { m_colElectrons = col; }
```

### SetColourHoles()

void Garfield::ViewSignal::SetColourHoles (
const short col) [inline]

Set the (ROOT) colour with which to draw the hole/ion component.
Definition at line 76 of file ViewSignal.hh.

```
00076 { m_colIons = col; }
```

### SetColourIons()

void Garfield::ViewSignal::SetColourIons (
const short col) [inline]

Set the (ROOT) colour with which to draw the hole/ion component.
Definition at line 74 of file ViewSignal.hh.

```
00074 { m_colIons = col; }
```

### SetColourTotal()

void Garfield::ViewSignal::SetColourTotal (
const short col) [inline]

Set the (ROOT) colour with which to draw the total signal.
Definition at line 70 of file ViewSignal.hh.

```
00070 { m_colTotal = col; }
```

### SetLabelY()

void Garfield::ViewSignal::SetLabelY (
const std::string & label) [inline]

Override the default y-axis label.
Definition at line 64 of file ViewSignal.hh.

```
00064 { m_labelY = label; }
```

### SetRangeX()

void Garfield::ViewSignal::SetRangeX (
const double xmin,
const double xmax)

Set the x-axis limits explicitly.

### SetRangeY()

void Garfield::ViewSignal::SetRangeY (
const double ymin,
const double ymax)

Set the y-axis limits explicitly.

### SetSensor()

void Garfield::ViewSignal::SetSensor (
Sensor * s)

Set the sensor from which to retrieve the signal.

### UnsetRangeX()

void Garfield::ViewSignal::UnsetRangeX () [inline]

Remove the user-defined x-axis limits.
Definition at line 56 of file ViewSignal.hh.

```
00056 { m_userRangeX = false; }
```

### UnsetRangeY()

void Garfield::ViewSignal::UnsetRangeY () [inline]
Remove the user-defined y-axis limits.
Definition at line 61 of file ViewSignal.hh.

```
00061 { m_userRangeY = false; }
```

## 7.129.4 Member Data Documentation

### m_colDelayed

std::array<short, 6> Garfield::ViewSignal::m_colDelayed [private]
Initial value:

```
{
    {kCyan + 2, kYellow - 7, kRed - 9, kGreen + 1, kYellow - 4, kRed - 9}}
```

Definition at line 106 of file ViewSignal.hh.

```
00106                                {
00107     {kCyan + 2, kYellow - 7, kRed - 9, kGreen + 1, kYellow - 4, kRed - 9}};
```

### m_colElectrons

short Garfield::ViewSignal::m_colElectrons = kOrange - 3 [private]
Definition at line 104 of file ViewSignal.hh.

### m_colIons

short Garfield::ViewSignal::m_colIons = kRed + 1 [private]
Definition at line 105 of file ViewSignal.hh.

### m_colPrompt

std::array<short, 3> Garfield::ViewSignal::m_colPrompt {{kAzure + 10, kRed - 4, kMagenta + 2}} [private]
Definition at line 109 of file ViewSignal.hh.

```
00109 {{kAzure + 10, kRed - 4, kMagenta + 2}};
```

### m_colTotal

short Garfield::ViewSignal::m_colTotal = kBlue + 3 [private]
Definition at line 103 of file ViewSignal.hh.

### m_labelY

std::string Garfield::ViewSignal::m_labelY = "" [private]
Definition at line 98 of file ViewSignal.hh.

### m_legend

bool Garfield::ViewSignal::m_legend = false [private]
Definition at line 100 of file ViewSignal.hh.

### m_sensor

Sensor* Garfield::ViewSignal::m_sensor = nullptr [private]
Definition at line 87 of file ViewSignal.hh.

### m_userRangeX

bool Garfield::ViewSignal::m_userRangeX = false [private]
Definition at line 92 of file ViewSignal.hh.

m_userRangeY

bool Garfield::ViewSignal::m_userRangeY = false [private]
Definition at line 95 of file ViewSignal.hh.

m_xmax

double Garfield::ViewSignal::m_xmax = 0. [private]
Definition at line 91 of file ViewSignal.hh.

m_xmin

double Garfield::ViewSignal::m_xmin = 0. [private]
Definition at line 90 of file ViewSignal.hh.

m_ymax

double Garfield::ViewSignal::m_ymax = 0. [private]
Definition at line 94 of file ViewSignal.hh.

m_ymin

double Garfield::ViewSignal::m_ymin = 0. [private]
Definition at line 93 of file ViewSignal.hh.
The documentation for this class was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ViewSignal.hh

## 7.130 Garfield::ComponentFieldMap::WeightingFieldCopy Struct Reference

#include <ComponentFieldMap.hh>

Public Attributes

- std::string source
- TMatrixD rot = TMatrixD(3, 3)
- TVectorD trans = TVectorD(3)

### 7.130.1 Detailed Description

Definition at line 263 of file ComponentFieldMap.hh.

### 7.130.2 Member Data Documentation

rot

TMatrixD Garfield::ComponentFieldMap::WeightingFieldCopy::rot = TMatrixD(3, 3)
Definition at line 266 of file ComponentFieldMap.hh.

source

std::string Garfield::ComponentFieldMap::WeightingFieldCopy::source
Definition at line 265 of file ComponentFieldMap.hh.

trans

TVectorD Garfield::ComponentFieldMap::WeightingFieldCopy::trans = TVectorD(3)
Definition at line 267 of file ComponentFieldMap.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh

## 7.131 Garfield::ComponentAnalyticField::Wire Struct Reference

Public Attributes

- double x
- double y

  Location.
- double r

  Radius.
- double v

  Potential.
- double e

  Charge.
- std::string type

  Label.
- double u

  Length.
- int ind

  Readout group.
- int nTrap

  Trap radius. Particle is ”trapped” if within nTrap ∗ radius of wire.
- double tension

  Stretching weight.
- double density

  Density.

### 7.131.1 Detailed Description

Definition at line 427 of file ComponentAnalyticField.hh.

### 7.131.2 Member Data Documentation

density

double Garfield::ComponentAnalyticField::Wire::density
Density.
Definition at line 438 of file ComponentAnalyticField.hh.

e

double Garfield::ComponentAnalyticField::Wire::e
Charge.
Definition at line 431 of file ComponentAnalyticField.hh.

ind

int Garfield::ComponentAnalyticField::Wire::ind
Readout group.
Definition at line 434 of file ComponentAnalyticField.hh.

nTrap

int Garfield::ComponentAnalyticField::Wire::nTrap
Trap radius. Particle is ”trapped” if within nTrap ∗ radius of wire.
Definition at line 436 of file ComponentAnalyticField.hh.

r

double Garfield::ComponentAnalyticField::Wire::r
Radius.
Definition at line 429 of file ComponentAnalyticField.hh.

tension

double Garfield::ComponentAnalyticField::Wire::tension
Stretching weight.
Definition at line 437 of file ComponentAnalyticField.hh.

type

std::string Garfield::ComponentAnalyticField::Wire::type
Label.
Definition at line 432 of file ComponentAnalyticField.hh.

u

double Garfield::ComponentAnalyticField::Wire::u
Length.
Definition at line 433 of file ComponentAnalyticField.hh.

v

double Garfield::ComponentAnalyticField::Wire::v
Potential.
Definition at line 430 of file ComponentAnalyticField.hh.

x

double Garfield::ComponentAnalyticField::Wire::x
Definition at line 428 of file ComponentAnalyticField.hh.

y

double Garfield::ComponentAnalyticField::Wire::y
Location.
Definition at line 428 of file ComponentAnalyticField.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh

## 7.132 Garfield::ComponentNeBem2d::Wire Struct Reference

Public Attributes

- double x
- double y

  Coordinates of the centre.
- double r

  Radius.
- double v

  Potential.
- double q

  Charge.
- int ntrap

  Trap radius (in units of the wire radius).

### 7.132.1 Detailed Description

Definition at line 159 of file ComponentNeBem2d.hh.

### 7.132.2 Member Data Documentation

ntrap

int Garfield::ComponentNeBem2d::Wire::ntrap
Trap radius (in units of the wire radius).
Definition at line 164 of file ComponentNeBem2d.hh.

q

double Garfield::ComponentNeBem2d::Wire::q
Charge.
Definition at line 163 of file ComponentNeBem2d.hh.

r

double Garfield::ComponentNeBem2d::Wire::r
Radius.
Definition at line 161 of file ComponentNeBem2d.hh.

v

double Garfield::ComponentNeBem2d::Wire::v
Potential.
Definition at line 162 of file ComponentNeBem2d.hh.

x

double Garfield::ComponentNeBem2d::Wire::x
Definition at line 160 of file ComponentNeBem2d.hh.

y

double Garfield::ComponentNeBem2d::Wire::y
Coordinates of the centre.
Definition at line 160 of file ComponentNeBem2d.hh.
The documentation for this struct was generated from the following file:

- /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh

# 8 File Documentation

## 8.1 /builds/garfield/docs/source/Include/Garfield/AvalancheGrid.hh File Reference

#include <array>
#include <cmath>
#include <string>
#include <vector>

Classes

- class Garfield::AvalancheGrid
  Calculate avalanches in a uniform electric field using avalanche statistics.
- struct Garfield::AvalancheGrid::Path
- struct Garfield::AvalancheGrid::AvalancheNode

Namespaces

- namespace Garfield

## 8.2 AvalancheGrid.hh

Go to the documentation of this file.

```
00001 #ifndef G__AVALANCHE__GRID__H
00002 #define G__AVALANCHE__GRID__H
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <string>
00007 #include <vector>
00008
00009 namespace Garfield {
00010
00011 class Sensor;
00012 class AvalancheMicroscopic;
00013 class ComponentParallelPlate;
00014
00016 class AvalancheGrid {
00017  public:
00019   AvalancheGrid() : AvalancheGrid(nullptr) {}
00021   AvalancheGrid(Sensor *sensor);
00023   ~AvalancheGrid() = default;
00024
00026   void SetSensor(Sensor *sensor) { m__sensor = sensor; }
00027
00029   void StartGridAvalanche();
00031   void SetElectronVelocity(const double vx, const double vy, const double vz) {
00032     double vel = std::sqrt(vx * vx + vy * vy + vz * vz);
00033     if (vel != std::abs(vx) && vel != std::abs(vy) && vel != std::abs(vz))
00034       return;
00035     int nx = (int)vx / vel;
00036     int ny = (int)vy / vel;
00037     int nz = (int)vz / vel;
00038     m__velNormal = {nx, ny, nz};
00039     m__Velocity = -std::abs(vel);
00040   }
00042   void SetElectronTownsend(const double town) { m__Townsend = town; }
00044   void SetElectronAttachment(const double att) { m__Attachment = att; }
00046   void SetMaxAvalancheSize(const double size) { m__MaxSize = size; }
00047
00056   void AddElectron(const double x, const double y, const double z,
00057                    const double t = 0, const int n = 1);
00059   void AddElectrons(AvalancheMicroscopic *avmc);
00060
00070   void SetGrid(const double xmin, const double xmax, const int xsteps,
00071                const double ymin, const double ymax, const int ysteps,
00072                const double zmin, const double zmax, const int zsteps);
00073
00075   int GetAmountOfStartingElectrons() { return m__nestart; }
00077   int GetAvalancheSize() { return m__nTotal; }
00078
00079   void EnableDebugging() { m__debug = true; }
00080
00081   void Reset();
00082
00083  private:
00084   bool m__debug = false;
00085
00086   double m__Townsend = -1;  // [1/cm]
00087
00088   double m__Attachment = -1;  // [1/cm]
00089
00090   double m__Velocity = 0.;  // [cm/ns]
00091
00092   std::vector<int> m__velNormal = {0, 0, 0};
00093
00094   double m__MaxSize = 1.6e7;  // Saturations size
00095   // Check if avalanche has reached maximum size
00096   bool m__Saturated = false;
00097   // Time when the avalanche has reached maximum size
00098   double m__SaturationTime = -1.;
00099
00100   int m__nestart = 0.;
00101
00102   std::vector<double> m__nLayer;
00103
00104   std::string m__className = "AvalancheGrid";
00105
00106   Sensor *m__sensor = nullptr;
00107
00108   bool m__printPar = false;
```

```
00109
00110   std::vector<double> m_zgrid;
00111   double m_zStepSize = 0.;
00112
00113   std::vector<double> m_ygrid;
00114   double m_yStepSize = 0.;
00115
00116   std::vector<double> m_xgrid;
00117   double m_xStepSize = 0.;
00118
00119   bool m_gridset = false;
00120   int m_nTotal = 0;
00121   double m_time = 0;
00122   bool m_run = true;
00123
00124   struct Path {
00125     std::vector<double> ts = {};
00126     std::vector<std::array<double, 3> > xs = {};
00127     std::vector<double> qs = {};
00128   };
00129
00130   struct AvalancheNode {
00131     double ix = 0;
00132     double iy = 0;
00133     double iz = 0;
00134
00135     int n = 1;
00136
00137     int layer = 0;
00138
00139     double townsend = 0;
00140     double attachment = 0;
00141     double velocity = 0;
00142
00143     double stepSize = 0;
00144     std::vector<int> velNormal = {0, 0, 0};
00145
00146     double time = 0.;
00147     double dt = -1.;
00148
00149     bool active = true;
00150
00151     Path path;
00152   };
00153
00154   std::vector<AvalancheNode> m_activeNodes = {};
00155
00156   // Assign electron to the closest grid point.
00157   bool SnapToGrid(const double x, const double y, const double z,
00158                   const double v, const int n = 1);
00159   // Go to next time step.
00160   void NextAvalancheGridPoint();
00161   // Obtain the Townsend coef., Attachment coef. and velocity vector from
00162   // sensor class.
00163   bool GetParameters(AvalancheNode &node);
00164
00165   void DeactivateNode(AvalancheNode &node);
00166 };
00167 }  // namespace Garfield
00168
00169 #endif
```

## 8.3 /builds/garfield/docs/source/Include/Garfield/AvalancheGridSpaceCharge.hh File Reference

#include <algorithm>
#include <string>
#include <utility>
#include <vector>

Classes

- class Garfield::AvalancheGridSpaceCharge

  Propagates avalanches with the 2d (axi-symmetric) space-charge routine from Lippmann, Riegler (2004) in uniform background fields.
- struct Garfield::AvalancheGridSpaceCharge::GridNode
- struct Garfield::AvalancheGridSpaceCharge::Point

Namespaces

- namespace Garfield

## 8.4 AvalancheGridSpaceCharge.hh

Go to the documentation of this file.

```
00001 #ifndef GARFIELD_AVALANCHEGRIDSPACECHARGE_HH
00002 #define GARFIELD_AVALANCHEGRIDSPACECHARGE_HH
00003
00004 #include <algorithm>
00005 #include <string>
00006 #include <utility>
00007 #include <vector>
00008
00009 namespace Garfield {
00010 class Sensor;
00011 class AvalancheMicroscopic;
00012 class ComponentParallelPlate;
00013
00017 class AvalancheGridSpaceCharge {
00018  public:
00020   AvalancheGridSpaceCharge() : AvalancheGridSpaceCharge(nullptr) {}
00022   AvalancheGridSpaceCharge(Sensor *sensor);
00024   ~AvalancheGridSpaceCharge() = default;
00025
00027   void Reset();
00028
00030   void EnableDebugging(const bool option = true) { m_bDebug = option; }
00031
00033   void EnableStickyAnode(const bool option = true) { m_bStick = option; }
00034
00036   void EnableTOF(const bool option = true) { m_bUseTOF = option; }
00037
00039   void EnableDiffusion(const bool option = true) { m_bDiffusion = option; }
00040
00042   void EnableSpaceChargeEffect(const bool option = true) {
00043     m_bSpaceCharge = option;
00044   }
00045
00047   void EnableAdaptiveTimeStepping(const bool option = true) {
00048     m_bAdaptiveTime = option;
00049   }
00050
00055   void EnableMC(const bool option = true) { m_bMC = option; }
00056
00059   void SetNCrit(const long NCrit = 1e8) { m_lNCrit = NCrit; }
00060
00066   void SetFieldCalculation(const std::string &option = "coulomb",
00067                            const int nof_approx = 1.) {
00068     m_sFieldOption = std::move(option);
00069     m_iFieldApprox = std::move(nof_approx);
00070   }
00071
00074   void SetK(float option = 0.95) { m_fStreamerK = option; }
00075
00077   void SetStopAtK(bool option = true) { m_bStopAtK = option; }
00078
00080   void SetSensor(Sensor *sensor);
00081
00090   void Set2dGrid(double zmin, double zmax, int zsteps, double rmax, int rsteps);
00091
00094   void AddElectrons(AvalancheMicroscopic *avmc);
00095
00097   void AddElectron(double x, double y, double z, double t = 0, int n = 1);
00098
00101   void AddExtraElectron(double y, int n = 1);
00102
00105   void StartGridAvalanche(double dtime = -1);
00106
00108   long GetAvalancheSize() const { return m_nTotPosIons; }
00109
00111   double GetMeanDistance();
00112
00114   [[nodiscard]] long ReachedKPercent() const {
00115     if (m_bFieldK)
00116       return m_lElectronsK;
00117     else
00118       return -1;
00119   }
00120
00122   [[nodiscard]] const std::vector<std::pair<double, long»
00123   GetElectronEvolution() const {
00124     return m_vNElectronEvolution;
00125   }
```

```
00126
00131   void ExportGrid(const std::string &filename);
00132
00133  private:
00134   struct GridNode {
00135     long nElectron = 0;
00136     double nPosIon = 0;
00137     double nNegIon = 0;
00138     // holder memories for stepping in time:
00139     long nElectronHolder = 0;
00140     double nPosIonHolder = 0;
00141     double nNegIonHolder = 0;
00142
00143     double townsend = 0;
00144     double attachment = 0;
00146     double velocity = 0.;
00148     double dSigmaL = 0;
00150     double dSigmaT = 0;
00151
00152     double Wv = 0;
00153     double Wr = 0;
00155     double townsendPT = 0;
00157     double attachmentPT = 0;
00159     double eFieldR = 0;
00161     double eFieldZ = 0;
00162
00163     double time = 0.;
00164
00165     bool anode = false;
00167     int layerIndex = 0;
00169     int gasGapIndex = 0;
00170     bool isGasGap = true;
00171   };
00172
00173   struct Point {
00174     double x, y, z;
00175     double t;
00176
00177     int gasLayerIndex;
00178   };
00179
00180   // Prepare grid and place stored electrons from AvalancheMicroscopic import
00181   void PrepareElectronsFromMicroscopicAvalanche();
00182
00183   // Assign electron to the closest grid point
00184   bool SnapTo2dGrid(double x, double y, double z, long n = 1, int gasLayer = 0);
00185
00186   // Prepare the mesh with the ComponentParallelPlate
00187   void Prepare2dMesh();
00188
00189   // Transports the electrons/nodes a timestep
00190   bool TransportTimeStep();
00191
00192   // Diffuses the electrons/nodes a timestep
00193   void DiffuseTimeStep(double dx, long nElectron, double nPosIon,
00194                        double nNegIon, int iz, int ir, int gasGap);
00195
00196   // Redistributes the charges
00197   void DistributeCharges(long nElectron, double nPosIon, double nNegIon, int iz,
00198                          int ir, double stepZ, double stepR, int gasGap);
00199
00200   // Calculate the field from all the contributions to the bin of interest. May
00201   // need much more functionalities/tables.
00202   void GetLocalField(int iz, int ir, double &eFieldZ, double &eFieldR,
00203                      const std::string &fieldOption, int gasGap);
00204
00205   // Calculate the field of charged ring in vacuum using coulomb potentials and
00206   // indices
00207   void GetFreeChargedRing(int iz, int ir, int fz, int fr, double &eFieldZ,
00208                           double &eFieldR);
00209
00210   // Calculate the field of charged ring in vacuum using coulomb potentials and
00211   // coordinates
00212   void GetFreeChargedRing(double zi, double ri, double zf, double rf,
00213                           double &eFieldZ, double &eFieldR);
00214
00215   // Get field at (zi, ri) from N charges at (zf, rf) either as a ring or a
00216   // coulomb ball (rf = 0) if i and f are too close it is considered as self
00217   // interaction and not included
00218   bool AddFieldFromChargeAt(int iz, int ir, int fz, int fr, double N,
00219                             double &eFieldZ, double &eFieldR);
00220
00221   // Get field at (zi, ri) from N charges at (zf, rf) either as a ring or a
00222   // coulomb ball (rf = 0) if i and f are too close it is considered as self
00223   // interaction and not included
00224   bool AddFieldFromChargeAt(int iz, int ir, double zf, double rf, double N,
00225                             double &eFieldZ, double &eFieldR);
```

```
00226
00227   // Get swarm parameters at electric field magnitude
00228   void GetSwarmParameters(double MagEField, double &alpha, double &eta,
00229                           double &drift, double &dSigmaL, double &dSigmaT,
00230                           double &wv, double &wr, double &alphaPT,
00231                           double &etaPT, int gasGap);
00232
00233   // Change from 2dGrid to Global coordinates
00234   void GetGlobalCoordinates(double r, double z, double phi, double &xg,
00235                             double &yg, double &zg, int gasGap);
00236
00237   // Import elliptic integral values
00238   void ImportEllipticIntegralValues(const std::string &filename);
00239
00240   // Gets elliptic integrals via list
00241   void GetEllipticIntegrals(double x, double &K, double &E);
00242
00243   // Get from index the gas gap number, else -1
00244   int GetGasGapNumber(int layerIndex) {
00245     auto it =
00246         std::find(m_vIndexGasGaps.begin(), m_vIndexGasGaps.end(), layerIndex);
00247     return (it != m_vIndexGasGaps.end())
00248                ? std::distance(m_vIndexGasGaps.begin(), it)
00249                : -1;
00250   }
00251
00252  private:
00253   std::string m_className = "AvalancheGridSpaceCharge";
00254
00255   bool m_bDebug = false;
00256   bool m_bDiffusion = false;
00257   bool m_bStick = true;
00258   // boolean for AvalancheElectron
00259   bool m_bDriftAvalanche = false;
00260   // boolean for ImportElectronsFromAvalancheMicroscopic
00261   bool m_bImportAvalanche = false;
00262   bool m_bPreparedImportAvalanche = false;
00263   long m_lNCrit = 1e8;
00264   bool m_bSpaceCharge = true;
00265
00266   float m_fStreamerK = 0.95;
00267   bool m_bStopAtK = false;
00268   bool m_bFieldK = false;
00269   long m_lElectronsK;
00270
00271   bool m_bAdaptiveTime = true;
00272   bool m_bImportElliptic = false;
00275   bool m_bUseTOF = true;
00277   bool m_bWrAvailable = true;
00279   bool m_bRatesAvailable = true;
00280
00281   bool m_bMC = true;
00282
00283   int m_iFieldApprox = 1;    //< order of approximation in Set(1,2,3,...)
00284   double m_dMinGroups = 50;  //< same values as lippmann
00285
00286   ComponentParallelPlate *m_pp = nullptr;
00287   Sensor *m_sensor = nullptr;
00288
00289   std::vector<double> m_zGrid;
00290   int m_zSteps = 0;
00291   double m_zStepSize = 0.;
00292
00293   std::vector<double> m_rGrid;
00294   int m_rSteps = 0.;
00295   double m_rStepSize = 0.;
00296
00297   bool m_isgridset = false;
00298   long m_nTotElectron = 0;
00299   long m_nTotPosIons = 0;
00300
00301   double m_time = 0.;
00302   double m_time0 = 0.;
00303   double m_dt = 0.;
00305   bool m_run = true;
00306
00307   std::vector<std::vector<int»
00308       m_zGasGapBoundaries;
00309
00310   std::vector<std::vector<GridNode» m_grid;
00312   std::vector<std::vector<Point» m_vElectrons;
00313
00314   std::vector<std::pair<double, long» m_vNElectronEvolution;
00315   std::vector<long> m_vGroupSizes = {
00316       1500, 800, 400, 200, 100,
00317       50,   20,  10,  5,   2};
00319   std::vector<int> m_vIndexGasGaps = {0};
```

```
00320
00323   std::vector<std::vector<double» m_vCoNGasLayer{};
00325   std::vector<double> m_vYPointInGasGap{};
00326
00328   std::vector<double> m_ezBkg = {0};
00330   std::vector<int> m_vSaturatedGaps{};
00331
00332   std::string m_sFieldOption = "coulomb";
00333   std::vector<double> m_vXElliptic;
00334   std::vector<double> m_vKElliptic;
00335   std::vector<double> m_vEElliptic;
00336 };
00337
00338 }  // namespace Garfield
00339
00340 #endif  // GARFIELD_AVALANCHEGRIDSPACECHARGE_HH
```

## 8.5 /builds/garfield/docs/source/Include/Garfield/AvalancheMC.hh File Reference

#include <array>
#include <string>
#include <utility>
#include <vector>
#include "Garfield/GarfieldConstants.hh"

Classes

- class Garfield::AvalancheMC

  Calculate drift lines and avalanches based on macroscopic transport coefficients, using Monte Carlo integration.
- struct Garfield::AvalancheMC::Point
- struct Garfield::AvalancheMC::EndPoint
- struct Garfield::AvalancheMC::Seed

Namespaces

- namespace Garfield

## 8.6 AvalancheMC.hh

Go to the documentation of this file.

```
00001 #ifndef G_AVALANCHE_MC_H
00002 #define G_AVALANCHE_MC_H
00003
00004 #include <array>
00005 #include <string>
00006 #include <utility>
00007 #include <vector>
00008
00009 #include "Garfield/GarfieldConstants.hh"
00010
00011 namespace Garfield {
00012
00013 class Sensor;
00014 class ViewDrift;
00015 class Medium;
00018 class AvalancheMC {
00019  public:
00021   AvalancheMC() : AvalancheMC(nullptr) {}
00023   AvalancheMC(Sensor* sensor);
00025   ~AvalancheMC() {}
00026
00028   void SetSensor(Sensor* s);
00029
00031   bool DriftElectron(const double x, const double y, const double z,
00032                      const double t, const size_t w = 1);
00034   bool DriftHole(const double x, const double y, const double z,
00035                  const double t, const size_t w = 1);
00037   bool DriftIon(const double x, const double y, const double z,
00038                 const double t, const size_t w = 1);
00040   bool DriftNegativeIon(const double x, const double y, const double z,
00041                         const double t, const size_t w = 1);
00047   bool AvalancheElectron(const double x, const double y, const double z,
```

```
00048                       const double t, const bool hole = false,
00049                       const size_t w = 1);
00051   bool AvalancheHole(const double x, const double y, const double z,
00052                      const double t, const bool electron = false,
00053                      const size_t w = 1);
00055   bool AvalancheElectronHole(const double x, const double y, const double z,
00056                              const double t, const size_t w = 1);
00057
00059   void AddElectron(const double x, const double y, const double z,
00060                    const double t, const size_t w = 1);
00062   void AddHole(const double x, const double y, const double z,
00063                const double t, const size_t w = 1);
00065   void AddIon(const double x, const double y, const double z,
00066               const double t, const size_t w = 1);
00068   void AddNegativeIon(const double x, const double y, const double z,
00069                       const double t, const size_t w = 1);
00071   bool ResumeAvalanche(const bool electron = true, const bool hole = true);
00072
00073   struct Point {
00074     double x, y, z;
00075     double t;
00076   };
00077
00078   struct EndPoint {
00079     int status;
00080     std::vector<Point> path;
00081     size_t weight;
00082   };
00083
00084   struct Seed {
00085     Point pt;
00086     Particle type;
00087     size_t w = 1;
00088   };
00089
00090   const std::vector<EndPoint>& GetElectrons() const { return m_electrons; }
00091   const std::vector<EndPoint>& GetHoles() const { return m_holes; }
00092   const std::vector<EndPoint>& GetIons() const { return m_ions; }
00093   const std::vector<EndPoint>& GetNegativeIons() const {
00094     return m_negativeIons;
00095   }
00096
00099   size_t GetNumberOfElectronEndpoints() const { return m_electrons.size(); }
00101   size_t GetNumberOfIonEndpoints() const { return m_ions.size(); }
00102
00110   void GetElectronEndpoint(const size_t i, double& x0, double& y0, double& z0,
00111                            double& t0, double& x1, double& y1, double& z1,
00112                            double& t1, int& status) const;
00113   void GetIonEndpoint(const size_t i, double& x0, double& y0, double& z0,
00114                       double& t0, double& x1, double& y1, double& z1,
00115                       double& t1, int& status) const;
00116   void GetNegativeIonEndpoint(const size_t i, double& x0, double& y0,
00117                               double& z0, double& t0, double& x1, double& y1,
00118                               double& z1, double& t1, int& status) const;
00119
00121   void EnablePlotting(ViewDrift* view);
00123   void DisablePlotting() { m_viewer = nullptr; }
00124
00126   void EnableDriftLines(const bool on = true) { m_storeDriftLines = on; }
00127
00129   void EnableSignalCalculation(const bool on = true) { m_doSignal = on; }
00134   void SetSignalAveragingOrder(const unsigned int navg) { m_navg = navg; }
00137   void UseWeightingPotential(const bool on = true) {
00138     m_useWeightingPotential = on;
00139   }
00140
00142   void EnableInducedChargeCalculation(const bool on = true) {
00143     m_doInducedCharge = on;
00144   }
00145
00148   void EnableRKFSteps(const bool on = true) { m_doRKF = on; }
00149
00152   void EnableProjectedPathIntegration(const bool on = true) {
00153     m_doEquilibration = on;
00154   }
00155
00157   void EnableDiffusion() { m_useDiffusion = true; }
00159   void DisableDiffusion() { m_useDiffusion = false; }
00160
00162   void EnableAttachment() { m_useAttachment = true; }
00164   void DisableAttachment() { m_useAttachment = false; }
00166   void EnableMultiplication(const bool on) { m_useMultiplication = on; }
00167
00169   void EnableTownsendMap(const bool on = true) { m_useTownsendMap = on; }
00171   void EnableAttachmentMap(const bool on = true) { m_useAttachmentMap = on; }
00173   void EnableMobilityMap(const bool on = true) { m_useMobilityMap = on; }
00175   void EnableVelocityMap(const bool on = true) { m_useVelocityMap = on; }
```

```
00176
00179   void EnableAvalancheSizeLimit(const unsigned int size) { m_sizeCut = size; }
00181   void DisableAvalancheSizeLimit() { m_sizeCut = 0; }
00183   int GetAvalancheSizeLimit() const { return m_sizeCut; }
00184
00186   void SetTimeSteps(const double d = 0.02);
00188   void SetDistanceSteps(const double d = 0.001);
00191   void SetCollisionSteps(const unsigned int n = 100);
00193   void SetStepDistanceFunction(double (*f)(double x, double y, double z));
00194
00196   void SetTimeWindow(const double t0, const double t1);
00198   void UnsetTimeWindow() { m_hasTimeWindow = false; }
00199
00201   void SetElectronSignalScalingFactor(const double scale) { m_scaleE = scale; }
00203   void SetHoleSignalScalingFactor(const double scale) { m_scaleH = scale; }
00205   void SetIonSignalScalingFactor(const double scale) { m_scaleI = scale; }
00206
00208   void GetAvalancheSize(unsigned int& ne, unsigned int& ni) const {
00209     ne = m_nElectrons;
00210     ni = std::max(m_nIons, m_nHoles);
00211   }
00213   std::pair<unsigned int, unsigned int> GetAvalancheSize() const {
00214     return std::make_pair(m_nElectrons, std::max(m_nIons, m_nHoles));
00215   }
00217   void EnableDebugging(const bool on = true) { m_debug = on; }
00218
00219  private:
00220   std::string m_className = "AvalancheMC";
00221
00222   Sensor* m_sensor = nullptr;
00223
00224   enum class StepModel {
00225     FixedTime,
00226     FixedDistance,
00227     CollisionTime,
00228     UserDistance
00229   };
00231   StepModel m_stepModel = StepModel::CollisionTime;
00232
00234   double m_tMc = 0.02;
00236   double m_dMc = 0.001;
00238   int m_nMc = 100;
00240   double (*m_fStep)(double x, double y, double z) = nullptr;
00241
00243   bool m_hasTimeWindow = false;
00245   double m_tMin = 0.;
00247   double m_tMax = 0.;
00248
00250   unsigned int m_sizeCut = 0;
00251
00253   unsigned int m_nElectrons = 0;
00255   unsigned int m_nHoles = 0;
00257   unsigned int m_nIons = 0;
00259   unsigned int m_nNegativeIons = 0;
00260
00263   std::vector<EndPoint> m_electrons;
00266   std::vector<EndPoint> m_holes;
00268   std::vector<EndPoint> m_ions;
00270   std::vector<EndPoint> m_negativeIons;
00271
00272   ViewDrift* m_viewer = nullptr;
00273
00274   bool m_storeDriftLines = false;
00275   bool m_doSignal = true;
00276   unsigned int m_navg = 1;
00277   bool m_useWeightingPotential = true;
00278   bool m_doInducedCharge = false;
00279   bool m_doEquilibration = true;
00280   bool m_doRKF = false;
00281   bool m_useDiffusion = true;
00282   bool m_useAttachment = true;
00283   bool m_useMultiplication = true;
00285   double m_scaleE = 1.;
00287   double m_scaleH = 1.;
00289   double m_scaleI = 1.;
00290
00292   bool m_useTownsendMap = false;
00294   bool m_useAttachmentMap = false;
00296   bool m_useMobilityMap = false;
00298   bool m_useVelocityMap = false;
00299
00300   bool m_debug = false;
00301
00303   int DriftLine(const Seed& seed, std::vector<Point>& path,
00304                 std::vector<Seed>& secondaries,
00305                 const bool aval, const bool signal) const;
00307   bool TransportParticles(std::vector<Seed>& stack,
```

```
00308                       const bool withElectrons, const bool withHoles,
00309                       const bool aval);
00310
00312   int GetField(const std::array<double, 3>& x, std::array<double, 3>& e,
00313                std::array<double, 3>& b, Medium*& medium) const;
00315   double GetMobility(const Particle particle, Medium* medium,
00316                      const std::array<double, 3>& x) const;
00318   bool GetVelocity(const Particle particle, Medium* medium,
00319                    const std::array<double, 3>& x,
00320                    const std::array<double, 3>& e,
00321                    const std::array<double, 3>& b,
00322                    std::array<double, 3>& v) const;
00324   bool GetDiffusion(const Particle particle, Medium* medium,
00325                     const std::array<double, 3>& e,
00326                     const std::array<double, 3>& b, double& dl,
00327                     double& dt) const;
00329   double GetAttachment(const Particle particle, Medium* medium,
00330                        const std::array<double, 3>& x,
00331                        const std::array<double, 3>& e,
00332                        const std::array<double, 3>& b) const;
00334   double GetTownsend(const Particle particle, Medium* medium,
00335                      const std::array<double, 3>& x,
00336                      const std::array<double, 3>& e,
00337                      const std::array<double, 3>& b) const;
00339   void StepRKF(const Particle particle, const std::array<double, 3>& x0,
00340                const std::array<double, 3>& v0, const double dt,
00341                std::array<double, 3>& xf, std::array<double, 3>& vf,
00342                int& status) const;
00344   void AddDiffusion(const double step, const double dl, const double dt,
00345                     std::array<double, 3>& x,
00346                     const std::array<double, 3>& v) const;
00348   void Terminate(const std::array<double, 3>& x0, const double t0,
00349                  std::array<double, 3>& x, double& t) const;
00351   bool ComputeGainLoss(const Particle ptype, const size_t w,
00352                        std::vector<Point>& path, int& status,
00353                        std::vector<Seed>& secondaries,
00354                        const bool semiconductor = false) const;
00356   bool ComputeAlphaEta(const Particle ptype, std::vector<Point>& path,
00357                        std::vector<double>& alphas,
00358                        std::vector<double>& etas) const;
00359   bool Equilibrate(std::vector<double>& alphas) const;
00361   void ComputeSignal(const double q, const std::vector<Point>& path) const;
00363   void ComputeInducedCharge(const double q,
00364                             const std::vector<Point>& path) const;
00365   void PrintError(const std::string& fcn, const std::string& par,
00366                   const Particle particle,
00367                   const std::array<double, 3>& x) const;
00368 };
00369 }  // namespace Garfield
00370
00371 #endif
```

## 8.7 /builds/garfield/docs/source/Include/Garfield/AvalancheMicroscopic.hh File Reference

#include <array>
#include <cstddef>
#include <string>
#include <utility>
#include <vector>
#include "Garfield/GarfieldConstants.hh"
#include "Garfield/MultiProcessInterface.hh"

### Classes

- class Garfield::AvalancheMicroscopic

  Calculate electron drift lines and avalanches using microscopic tracking.
- struct Garfield::AvalancheMicroscopic::Point
- struct Garfield::AvalancheMicroscopic::Electron
- struct Garfield::AvalancheMicroscopic::Seed
- struct Garfield::AvalancheMicroscopic::Statistics
- struct Garfield::AvalancheMicroscopic::Photon

Namespaces

- namespace Garfield

## 8.8 AvalancheMicroscopic.hh

Go to the documentation of this file.

```
00001 #ifndef G__AVALANCHE__MICROSCOPIC__H
00002 #define G__AVALANCHE__MICROSCOPIC__H
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <string>
00007 #include <utility>
00008 #include <vector>
00009
00010 #include "Garfield/GarfieldConstants.hh"
00011 #include "Garfield/MultiProcessInterface.hh"
00012
00013 class TH1;
00014
00015 namespace Garfield {
00016
00017 class AvalancheMicroscopicGPU;
00018 class Medium;
00019 class ViewDrift;
00020 class Sensor;
00021
00023
00024 class AvalancheMicroscopic {
00025  public:
00027   AvalancheMicroscopic() : AvalancheMicroscopic(nullptr) {}
00029   AvalancheMicroscopic(Sensor* sensor);
00031   ~AvalancheMicroscopic() {}
00032
00034   void SetSensor(Sensor* sensor);
00035
00037   void EnablePlotting(ViewDrift* view, const std::size_t nColl = 100);
00039   void DisablePlotting() { m_viewer = nullptr; }
00041   void EnableExcitationMarkers(const bool on = true) { m_plotExcitations = on; }
00043   void EnableIonisationMarkers(const bool on = true) { m_plotIonisations = on; }
00045   void EnableAttachmentMarkers(const bool on = true) { m_plotAttachments = on; }
00046
00048   void EnableSignalCalculation(const bool on = true) { m_doSignal = on; }
00051   void UseWeightingPotential(const bool on = true) {
00052     m_useWeightingPotential = on;
00053   }
00056   void EnableWeightingFieldIntegration(const bool on = true) {
00057     m_integrateWeightingField = on;
00058   }
00059
00061   void UseInducedCharge(const bool on = true) { m_doInducedCharge = on; }
00062
00064   void EnablePathLengthComputation(const bool on = true) {
00065     m_computePathLength = on;
00066   }
00068   void EnableElectronEnergyHistogramming(TH1* histo);
00070   void DisableElectronEnergyHistogramming() { m_histElectronEnergy = nullptr; }
00072   void EnableHoleEnergyHistogramming(TH1* histo);
00074   void DisableHoleEnergyHistogramming() { m_histHoleEnergy = nullptr; }
00075
00082   void SetDistanceHistogram(TH1* histo, const char opt = 'r');
00084   void EnableDistanceHistogramming(const int type);
00086   void DisableDistanceHistogramming(const int type);
00088   void DisableDistanceHistogramming();
00090   void EnableSecondaryEnergyHistogramming(TH1* histo);
00092   void DisableSecondaryEnergyHistogramming() { m_histSecondary = nullptr; }
00093
00095   void EnableDriftLines(const bool on = true) { m_storeDriftLines = on; }
00096
00099   void EnablePhotonTransport(const bool on = true) { m_usePhotons = on; }
00100
00102   void EnableBandStructure(const bool on = true) { m_useBandStructure = on; }
00103
00105   void EnableNullCollisionSteps(const bool on = true, const int nSteps = 1) {
00106     m_useNullCollisionSteps = on;
00107     m_nullCollScale = nSteps <= 1 ? 1. : 1. / nSteps;
00108   }
00109
00111   void EnableRKNSteps(const bool on = true) { m_rknSteps = on; }
00114   void SetRKNTolerance(const double sTol = 1.e-10,
00115                        const double sMinStep = 1.e-5) {
00116     m_rknsteperrortol = sTol;
00117     m_rknMinh = sMinStep;
00118   }
```

```
00119
00122   void SetElectronTransportCut(const double cut) { m_deltaCut = cut; }
00124   double GetElectronTransportCut() const { return m_deltaCut; }
00125
00127   void SetPhotonTransportCut(const double cut) { m_gammaCut = cut; }
00129   double GetPhotonTransportCut() const { return m_gammaCut; }
00130
00133   void EnableAvalancheSizeLimit(const unsigned int size) { m_sizeCut = size; }
00135   void DisableAvalancheSizeLimit() { m_sizeCut = 0; }
00137   int GetAvalancheSizeLimit() const { return m_sizeCut; }
00138
00140   void EnableMagneticField(const bool on = true) {
00141     m_useBfieldAuto = false;
00142     m_useBfield = on;
00143   }
00144
00146   void SetCollisionSteps(const unsigned int n) { m_nCollSkip = n; }
00147
00149   void SetTimeWindow(const double t0, const double t1);
00151   void UnsetTimeWindow() { m_hasTimeWindow = false; }
00152
00154   void GetAvalancheSize(int& ne, int& ni) const {
00155     ne = m_nElectrons;
00156     ni = m_nIons;
00157   }
00159   std::pair<int, int> GetAvalancheSize() const {
00160     return std::make_pair(m_nElectrons, m_nIons);
00161   }
00162   void GetAvalancheSize(int& ne, int& nh, int& ni) const {
00163     ne = m_nElectrons;
00164     nh = m_nHoles;
00165     ni = m_nIons;
00166   }
00168   void GetAvalancheSizeGPU(int& ne, int& ni) const {
00169     ne = m_nElectronsGPU;
00170     ni = m_nIonsGPU;
00171   }
00172
00173   struct Point {
00174     double x, y, z;
00175     double t;
00176     double energy;
00177     double kx, ky, kz;
00178     int band;
00179   };
00180
00181   struct Electron {
00182     int status = 0;
00183     std::vector<Point> path;
00184     size_t weight = 1;
00185     double pathLength = 0.;
00186   };
00187
00188   struct Seed {
00189     Point pt;
00190     Particle type;
00191     size_t w = 1;
00192   };
00193
00194   const std::vector<Electron>& GetElectrons() const { return m_electrons; }
00195   const std::vector<Electron>& GetHoles() const { return m_holes; }
00198   size_t GetNumberOfElectronEndpoints() const { return m_electrons.size(); }
00199   unsigned int GetNumberOfElectronEndpointsGPU() const {
00200     return m_electrons_gpu.size();
00201   }
00210   void GetElectronEndpoint(const size_t i, double& x0, double& y0, double& z0,
00211                            double& t0, double& e0, double& x1, double& y1,
00212                            double& z1, double& t1, double& e1,
00213                            int& status) const;
00214   void GetElectronEndpointGPU(const size_t i, double& x0, double& y0,
00215                               double& z0, double& t0, double& e0, double& x1,
00216                               double& y1, double& z1, double& t1, double& e1,
00217                               int& status) const;
00218   size_t GetNumberOfElectronDriftLinePoints(const size_t i = 0) const;
00219   void GetElectronDriftLinePoint(double& x, double& y, double& z, double& t,
00220                                  const size_t ip, const size_t ie = 0) const;
00221
00222   size_t GetNumberOfPhotons() const { return m_photons.size(); }
00223   // Status codes:
00224   //   -2: photon absorbed by gas molecule
00225   void GetPhoton(const size_t i, double& e, double& x0, double& y0, double& z0,
00226                  double& t0, double& x1, double& y1, double& z1, double& t1,
00227                  int& status) const;
00228
00236   bool DriftElectron(const double x, const double y, const double z,
00237                      const double t, const double e, const double dx = 0.,
00238                      const double dy = 0., const double dz = 0.,
```

```
00239                   const size_t w = 1);
00240
00242   bool AvalancheElectron(const double x, const double y, const double z,
00243                          const double t, const double e, const double dx = 0.,
00244                          const double dy = 0., const double dz = 0.,
00245                          const size_t w = 1);
00247   void AddElectron(const double x, const double y, const double z,
00248                    const double t, const double e, const double dx = 0.,
00249                    const double dy = 0., const double dz = 0.,
00250                    const size_t w = 1);
00252   bool ResumeAvalanche();
00253
00255   void SetUserHandleStep(void (*f)(double x, double y, double z, double t,
00256                                    double e, double dx, double dy, double dz,
00257                                    bool hole));
00259   void UnsetUserHandleStep() { m_userHandleStep = nullptr; }
00261   void SetUserHandleCollision(void (*f)(double x, double y, double z, double t,
00262                                         int type, int level, Medium* m,
00263                                         double e0, double e1, double dx0,
00264                                         double dy0, double dz0, double dx1,
00265                                         double dy1, double dz1));
00267   void UnsetUserHandleCollision() { m_userHandleCollision = nullptr; }
00269   void SetUserHandleAttachment(void (*f)(double x, double y, double z, double t,
00270                                          int type, int level, Medium* m));
00272   void UnsetUserHandleAttachment() { m_userHandleAttachment = nullptr; }
00274   void SetUserHandleInelastic(void (*f)(double x, double y, double z, double t,
00275                                         int type, int level, Medium* m));
00277   void UnsetUserHandleInelastic() { m_userHandleInelastic = nullptr; }
00280   void SetUserHandleIonisation(void (*f)(double x, double y, double z, double t,
00281                                          int type, int level, Medium* m));
00283   void UnsetUserHandleIonisation() { m_userHandleIonisation = nullptr; }
00284
00286   void EnableDebugging() { m_debug = true; }
00287   void DisableDebugging() { m_debug = false; }
00288
00289   // class to store CPU/GPU benchmark comparisons
00290   struct Statistics {
00291     std::vector<double> gpu_stack_process_time;
00292     std::vector<double> cpu_stack_process_time;
00293     std::vector<double> gpu_stack_transport_time;
00294     std::vector<double> cpu_stack_transport_time;
00295
00296     std::vector<size_t> stack_old_size;
00297     std::vector<size_t> stack_new_size;
00298   };
00299
00300   Statistics GetStatistics() { return m_stats; }
00301
00302   void SetRunModeOptions(MPRunMode mode, int device = -1);
00303   void SetMaxNumShowerLoops(int max_loops) { m_maxNumShowerLoops = max_loops; }
00304   void SetShowProgress(bool show_progress) { m_showProgress = show_progress; }
00305   void SetDebugShowerIterationAndElectronID(int iter_num, int elec_id) {
00306     m_debugShowerLoopNum = iter_num;
00307     m_debugElectronID = elec_id;
00308   }
00309
00310 private:
00311   std::string m_className = "AvalancheMicroscopic";
00312
00313   MPRunMode m_runMode{MPRunMode::Normal};
00314   int m_cudaDevice{-1};
00315   int m_maxNumShowerLoops{-1};
00316   int m_debugElectronID{-1};
00317   int m_debugShowerLoopNum{-1};
00318   bool m_showProgress{false};
00319   AvalancheMicroscopicGPU* m_gpuInterface{nullptr};
00320
00321   Statistics m_stats;
00322
00323   Sensor* m_sensor = nullptr;
00324
00325   std::vector<Electron> m_electrons;
00326   std::vector<Electron> m_electrons_gpu;
00327   std::vector<Electron> m_holes;
00328
00329   std::vector<Seed> m_stackStoreCPU;
00330   std::vector<Electron> m_stackStoreGPU;
00331
00332   struct Photon {
00333     int status;
00334     double energy;
00335     double x0, y0, z0, t0;
00336     double x1, y1, z1, t1;
00337   };
00338   std::vector<Photon> m_photons;
00339
00341   int m_nElectrons = 0;
```

```
00343   int m_nHoles = 0;
00345   int m_nIons = 0;
00346
00348   int m_nElectronsGPU = 0;
00350   int m_nHolesGPU = 0;
00352   int m_nIonsGPU = 0;
00353
00354   ViewDrift* m_viewer = nullptr;
00355   bool m_plotExcitations = true;
00356   bool m_plotIonisations = true;
00357   bool m_plotAttachments = true;
00358
00359   TH1* m_histElectronEnergy = nullptr;
00360   TH1* m_histHoleEnergy = nullptr;
00361   TH1* m_histDistance = nullptr;
00362   char m_distanceOption = 'r';
00363   std::vector<int> m_distanceHistogramType;
00364
00365   TH1* m_histSecondary = nullptr;
00366
00367   bool m_doSignal = true;
00368   bool m_useWeightingPotential = true;
00369   bool m_integrateWeightingField = false;
00370   bool m_doInducedCharge = false;
00371
00372   bool m_computePathLength = false;
00373   bool m_storeDriftLines = false;
00374   bool m_usePhotons = false;
00375   bool m_useBandStructure = true;
00376   bool m_useNullCollisionSteps = false;
00377   bool m_useBfieldAuto = true;
00378   bool m_useBfield = false;
00379
00380   bool m_rknSteps = false;
00381   double m_rknsteperrortol = 1.e-10;
00382   double m_rknMinh = 1.e-5;
00383   double m_nullCollScale = 1.;
00384
00385   // Transport cuts
00386   double m_deltaCut = 0.;
00387   double m_gammaCut = 0.;
00388
00389   // Max. avalanche size
00390   unsigned int m_sizeCut = 0;
00391
00392   size_t m_nCollSkip = 100;
00393   size_t m_nCollPlot = 100;
00394
00395   bool m_hasTimeWindow = false;
00396   double m_tMin = 0.;
00397   double m_tMax = 0.;
00398
00399   // User procedures
00400   void (*m_userHandleStep)(double x, double y, double z, double t, double e,
00401                            double dx, double dy, double dz,
00402                            bool hole) = nullptr;
00403   void (*m_userHandleCollision)(double x, double y, double z, double t,
00404                                 int type, int level, Medium* m, double e0,
00405                                 double e1, double dx0, double dy0, double dz0,
00406                                 double dx1, double dy1, double dz1) = nullptr;
00407   void (*m_userHandleAttachment)(double x, double y, double z, double t,
00408                                  int type, int level, Medium* m) = nullptr;
00409   void (*m_userHandleInelastic)(double x, double y, double z, double t,
00410                                 int type, int level, Medium* m) = nullptr;
00411   void (*m_userHandleIonisation)(double x, double y, double z, double t,
00412                                  int type, int level, Medium* m) = nullptr;
00413
00414   // Switch on/off debugging messages
00415   bool m_debug = false;
00416
00417   bool TransportElectrons(std::vector<Seed>& stack, const bool aval);
00418   int TransportElectron(
00419       const Seed& seed, const bool signal,
00420       std::vector<double>& ts, std::vector<std::array<double, 3> >& xs,
00421       std::vector<Point>& path, std::vector<Seed>& stack);
00422   int TransportElectronBfield(
00423       const Seed& seed, const bool signal,
00424       std::vector<double>& ts, std::vector<std::array<double, 3> >& xs,
00425       std::vector<Point>& path, std::vector<Seed>& stack);
00426   int TransportElectronSc(
00427       const Seed& seed, const bool signal,
00428       std::vector<double>& ts, std::vector<std::array<double, 3> >& xs,
00429       std::vector<Point>& path, std::vector<Seed>& stack);
00430   void TransportPhoton(const double x, const double y, const double z,
00431                        const double t, const double e, const size_t w,
00432                        std::vector<Seed>& stack);
00433
```

```
00434   bool transportParticleStack(
00435       const bool aval, std::vector<Seed>& stack,
00436       std::vector<Seed>& newParticles, const bool signal,
00437       const bool useBfield, const bool sc);
00438   void Terminate(double x0, double y0, double z0, double t0, double& x1,
00439                  double& y1, double& z1, double& t1) const;
00440
00441   void CreatePenningElectron(const double x, const double y, const double z,
00442                              const double t, const size_t w,
00443                              const double ds, const double dt, const double ep,
00444                              const int level, std::vector<Seed>& stack) const;
00445   void PlotCollision(const int cstype, const size_t did, const double x,
00446                      const double y, const double z, size_t& nCollPlot) const;
00447   void CallUserHandles(
00448       const int cstype, const double x, const double y, const double z,
00449       const double t, const int level, Medium* medium,
00450       const double en1, const double en,
00451       const double kx, const double ky, const double kz,
00452       const double kx1, const double ky1, const double kz1) const;
00453   void FillDistanceHistogram(const int cstype, const double x, const double y,
00454                              const double z, double& xLast, double& yLast,
00455                              double& zLast) const;
00456
00457  public:
00458   std::vector<Seed> GetStackOld() {
00459     return m_stackStoreCPU;
00460   }
00461   std::vector<Electron> GetStackOldGPU() { return m_stackStoreGPU; }
00462
00463   friend class AvalancheMicroscopicGPU;
00464 };
00465 }  // namespace Garfield
00466
00467 #endif
```

## 8.9 /builds/garfield/docs/source/Include/Garfield/Component.hh File Reference

#include ”Garfield/HelperMacros.hh”
#include <array>
#include <string>
#include <vector>

### Namespaces

- namespace Garfield

### Functions

- class Garfield::GARFIELD_CLASS_NAME (Component)

  Abstract base class for components.

## 8.10 Component.hh

Go to the documentation of this file.

```
00001 // Include this header if we're compiling with the GPU or this is the first time
00002 // without
00003 #if defined(___GPUCOMPILE___) || !defined(G_COMPONENT_H)
00004
00005 #if !defined(___GPUCOMPILE___) && !defined(G_COMPONENT_H)
00006 #define G_COMPONENT_H
00007 #endif
00008
00009 #include ”Garfield/HelperMacros.hh”
00010
00011 #ifdef ___GPUCOMPILE___
00012 #else
00013
00014 #include <array>
00015 #include <string>
00016 #include <vector>
00017
00018 #endif
00019
00020 namespace Garfield {
00021
00022 // setup class names depending on if this is compiling the GPU static version or
```

```
00023 // not
00024 #if !defined(___GPUCOMPILE___)
00025 class ComponentGPU;
00026 class Geometry;
00027 class Medium;
00028 #endif
00029
00031 class GARFIELD__CLASS__NAME(Component) {
00032  public:
00033 #ifdef ___GPUCOMPILE___
00034   GARFIELD__CLASS__NAME(Component)() = default;
00035 #else
00037   GARFIELD__CLASS__NAME(Component)() = delete;
00039   GARFIELD__CLASS__NAME(Component)(const std::string& name);
00040 #endif
00042   virtual ~GARFIELD__CLASS__NAME(Component)() {};
00043
00044 #ifndef ___GPUCOMPILE___
00046   virtual void SetGeometry(Geometry* geo);
00048   virtual void Clear();
00049
00051   virtual Medium* GetMedium(const double x, const double y, const double z);
00052
00070   virtual void ElectricField(const double x, const double y, const double z,
00071                              double& ex, double& ey, double& ez, Medium*& m,
00072                              int& status) = 0;
00074   virtual void ElectricField(const double x, const double y, const double z,
00075                              double& ex, double& ey, double& ez, double& v,
00076                              Medium*& m, int& status) = 0;
00077 #else
00078   ___device___ void ElectricField(const cuda__t xin, const cuda__t yin,
00079                                   const cuda__t zin, cuda__t& ex, cuda__t& ey,
00080                                   cuda__t& ez, MediumGPU*& m, int& status);
00081 #endif
00082
00083 #ifndef ___GPUCOMPILE___
00085   std::array<double, 3> ElectricField(const double x, const double y,
00086                                       const double z);
00088   virtual double ElectricPotential(const double x, const double y,
00089                                    const double z);
00091   virtual bool GetVoltageRange(double& vmin, double& vmax) = 0;
00092
00098   virtual void WeightingField(const double x, const double y, const double z,
00099                               double& wx, double& wy, double& wz,
00100                               const std::string& label);
00106   virtual double WeightingPotential(const double x, const double y,
00107                                     const double z, const std::string& label);
00108
00112   virtual const std::vector<double>& DelayedSignalTimes(
00113       const std::string& /*label*/) {
00114     return m__wdtimes;
00115   }
00123   virtual void DelayedWeightingField(const double x, const double y,
00124                                      const double z, const double t, double& wx,
00125                                      double& wy, double& wz,
00126                                      const std::string& label);
00127
00134   virtual double DelayedWeightingPotential(const double x, const double y,
00135                                            const double z, const double t,
00136                                            const std::string& label);
00137
00141   virtual void DelayedWeightingPotentials(const double x, const double y,
00142                                           const double z,
00143                                           const std::string& label,
00144                                           std::vector<double>& dwp);
00151   virtual void MagneticField(const double x, const double y, const double z,
00152                              double& bx, double& by, double& bz, int& status);
00154   void SetMagneticField(const double bx, const double by, const double bz);
00155
00157   virtual bool IsReady() { return m__ready; }
00159   virtual bool Is3d() { return true; }
00160
00162   virtual bool GetBoundingBox(double& xmin, double& ymin, double& zmin,
00163                               double& xmax, double& ymax, double& zmax);
00164
00166   virtual bool GetElementaryCell(double& xmin, double& ymin, double& zmin,
00167                                  double& xmax, double& ymax, double& zmax);
00168   // Get the x-length of the elementary cell.
00169   double CellSizeX();
00170   // Get the y-length of the elementary cell.
00171   double CellSizeY();
00172   // Get the z-length of the elementary cell.
00173   double CellSizeZ();
00174
00176   virtual size__t GetNumberOfElements() const { return 0; }
00178   virtual bool GetElementNodes(const size__t /*i*/,
00179                                std::vector<size__t>& /*nodes*/) const {
```

```
00180   return false;
00181 }
00184 virtual bool GetElementRegion(const size_t /*i*/, size_t& /*mat*/,
00185                                bool& /*drift*/) const {
00186   return false;
00187 }
00189 virtual size_t GetNumberOfNodes() const { return 0; }
00191 virtual bool GetNode(const size_t i, double& x, double& y, double& z) const;
00192
00200 double IntegrateFluxCircle(const double xc, const double yc, const double r,
00201                            const unsigned int nI = 50);
00209 double IntegrateFluxSphere(const double xc, const double yc, const double zc,
00210                            const double r, const unsigned int nI = 20);
00211
00220 double IntegrateFluxParallelogram(
00221     const double x0, const double y0, const double z0, const double dx1,
00222     const double dy1, const double dz1, const double dx2, const double dy2,
00223     const double dz2, const unsigned int nU = 20, const unsigned int nV = 20);
00224
00227 double IntegrateWeightingFluxParallelogram(
00228     const std::string& label, const double x0, const double y0,
00229     const double z0, const double dx1, const double dy1, const double dz1,
00230     const double dx2, const double dy2, const double dz2,
00231     const unsigned int nU = 20, const unsigned int nV = 20);
00232
00242 double IntegrateFluxLine(const double x0, const double y0, const double z0,
00243                          const double x1, const double y1, const double z1,
00244                          const double xp, const double yp, const double zp,
00245                          const unsigned int nI, const int isign = 0);
00246
00256 virtual bool CrossedWire(const double x0, const double y0, const double z0,
00257                          const double x1, const double y1, const double z1,
00258                          double& xc, double& yc, double& zc,
00259                          const bool centre, double& rc);
00266 virtual bool InTrapRadius(const double q0, const double x0, const double y0,
00267                           const double z0, double& xw, double& yw,
00268                           double& rw);
00271 virtual bool CrossedPlane(const double x0, const double y0, const double z0,
00272                           const double x1, const double y1, const double z1,
00273                           double& xc, double& yc, double& zc);
00274
00276 void EnablePeriodicityX(const bool on = true) {
00277   m_periodic[0] = on;
00278   UpdatePeriodicity();
00279 }
00281 void EnablePeriodicityY(const bool on = true) {
00282   m_periodic[1] = on;
00283   UpdatePeriodicity();
00284 }
00286 void EnablePeriodicityZ(const bool on = true) {
00287   m_periodic[2] = on;
00288   UpdatePeriodicity();
00289 }
00291 void IsPeriodic(bool& perx, bool& pery, bool& perz) {
00292   perx = m_periodic[0];
00293   pery = m_periodic[1];
00294   perz = m_periodic[2];
00295 }
00296
00298 void EnableMirrorPeriodicityX(const bool on = true) {
00299   m_mirrorPeriodic[0] = on;
00300   UpdatePeriodicity();
00301 }
00303 void EnableMirrorPeriodicityY(const bool on = true) {
00304   m_mirrorPeriodic[1] = on;
00305   UpdatePeriodicity();
00306 }
00308 void EnableMirrorPeriodicityZ(const bool on = true) {
00309   m_mirrorPeriodic[2] = on;
00310   UpdatePeriodicity();
00311 }
00313 void IsMirrorPeriodic(bool& perx, bool& pery, bool& perz) {
00314   perx = m_mirrorPeriodic[0];
00315   pery = m_mirrorPeriodic[1];
00316   perz = m_mirrorPeriodic[2];
00317 }
00318
00320 void EnableAxialPeriodicityX(const bool on = true) {
00321   m_axiallyPeriodic[0] = on;
00322   UpdatePeriodicity();
00323 }
00325 void EnableAxialPeriodicityY(const bool on = true) {
00326   m_axiallyPeriodic[1] = on;
00327   UpdatePeriodicity();
00328 }
00330 void EnableAxialPeriodicityZ(const bool on = true) {
00331   m_axiallyPeriodic[2] = on;
```

```
00332   UpdatePeriodicity();
00333 }
00335 void IsAxiallyPeriodic(bool& perx, bool& pery, bool& perz) {
00336   perx = m_axiallyPeriodic[0];
00337   pery = m_axiallyPeriodic[1];
00338   perz = m_axiallyPeriodic[2];
00339 }
00340
00342 void EnableRotationSymmetryX(const bool on = true) {
00343   m_rotationSymmetric[0] = on;
00344   UpdatePeriodicity();
00345 }
00347 void EnableRotationSymmetryY(const bool on = true) {
00348   m_rotationSymmetric[1] = on;
00349   UpdatePeriodicity();
00350 }
00352 void EnableRotationSymmetryZ(const bool on = true) {
00353   m_rotationSymmetric[2] = on;
00354   UpdatePeriodicity();
00355 }
00357 void IsRotationSymmetric(bool& rotx, bool& roty, bool& rotz) {
00358   rotx = m_rotationSymmetric[0];
00359   roty = m_rotationSymmetric[1];
00360   rotz = m_rotationSymmetric[2];
00361 }
00362
00364 void EnableTriangleSymmetricXY(const bool on = true,
00365                                const bool oct = 2) {
00366   m_triangleSymmetric[0] = on;
00367   m_triangleSymmetricOct = oct;
00368   m_mirrorPeriodic[0] = on;
00369   m_mirrorPeriodic[1] = on;
00370 }
00372 void EnableTriangleSymmetricXZ(const bool on = true,
00373                                const bool oct = 2) {
00374   m_triangleSymmetric[1] = on;
00375   m_triangleSymmetricOct = oct;
00376   m_mirrorPeriodic[0] = on;
00377   m_mirrorPeriodic[2] = on;
00378 }
00380 void EnableTriangleSymmetricYZ(const bool on = true,
00381                                const bool oct = 2) {
00382   m_triangleSymmetric[2] = on;
00383    m_triangleSymmetricOct = oct;
00384   m_mirrorPeriodic[1] = on;
00385   m_mirrorPeriodic[2] = on;
00386 }
00387
00389 void EnableDebugging(const bool on = true) { m_debug = on; }
00391 void DisableDebugging() { m_debug = false; }
00392
00394 virtual bool HasMagneticField() const;
00395
00397 virtual bool HasTownsendMap() const { return false; }
00399 virtual bool HasAttachmentMap() const { return false; }
00401 virtual bool HasMobilityMap() const { return false; }
00403 virtual bool HasVelocityMap() const { return false; }
00404
00406 virtual bool ElectronAttachment(const double /*x*/, const double /*y*/,
00407                                 const double /*z*/, double& eta) {
00408   eta = 0;
00409   return false;
00410 }
00412 virtual bool HoleAttachment(const double /*x*/, const double /*y*/,
00413                             const double /*z*/, double& eta) {
00414   eta = 0;
00415   return false;
00416 }
00417
00418 // Get the electron mobility coefficient.
00419 virtual bool ElectronMobility(const double /*x*/, const double /*y*/,
00420                               const double /*z*/, double& mu) {
00421   mu = 0;
00422   return false;
00423 }
00425 virtual bool HoleMobility(const double /*x*/, const double /*y*/,
00426                           const double /*z*/, double& mu) {
00427   mu = 0;
00428   return false;
00429 }
00430
00432 virtual bool ElectronTownsend(const double /*x*/, const double /*y*/,
00433                               const double /*z*/, double& alpha) {
00434   alpha = 0;
00435   return false;
00436 }
00438 virtual bool HoleTownsend(const double /*x*/, const double /*y*/,
```

```
00439                         const double /*z*/, double& alpha) {
00440     alpha = 0;
00441     return false;
00442   }
00444   virtual bool ElectronVelocity(const double /*x*/, const double /*y*/,
00445                                 const double /*z*/, double& vx, double& vy,
00446                                 double& vz) {
00447     vx = vy = vz = 0;
00448     return false;
00449   }
00451   virtual bool HoleVelocity(const double /*x*/, const double /*y*/,
00452                             const double /*z*/, double& vx, double& vy,
00453                             double& vz) {
00454     vx = vy = vz = 0;
00455     return false;
00456   }
00457
00458   virtual double StepSizeHint() { return -1.; }
00459
00461   virtual double CreateGPUTransferObject(ComponentGPU*& comp_gpu);
00462
00463  protected:
00465   std::string m_className = "Component";
00466
00468   Geometry* m_geometry = nullptr;
00469
00471   std::array<double, 3> m_b0 = {{0., 0., 0.}};
00472 #endif
00474   bool m_ready = false;
00475
00476 #ifdef ___GPUCOMPILE___
00478   bool m_periodic[3] = {false, false, false};
00480   bool m_mirrorPeriodic[3] = {false, false, false};
00482   bool m_axiallyPeriodic[3] = {false, false, false};
00484   bool m_rotationSymmetric[3] = {false, false, false};
00486   bool m_triangleSymmetric[3] = {false, false, false};
00488   int m_triangleSymmetricOct = 0;
00490   const int m_triangleOctRules[4] = {1, 4, 5, 8};
00491   bool m_outsideCone = false;
00492 #else
00494   bool m_debug = false;
00495
00497   std::array<bool, 3> m_periodic = {{false, false, false}};
00499   std::array<bool, 3> m_mirrorPeriodic = {{false, false, false}};
00501   std::array<bool, 3> m_axiallyPeriodic = {{false, false, false}};
00503   std::array<bool, 3> m_rotationSymmetric = {{false, false, false}};
00505   std::array<bool, 3> m_triangleSymmetric = {{false, false, false}};
00507   int m_triangleSymmetricOct = 0;
00509   const std::array<int, 4> m_triangleOctRules = {1, 4, 5, 8};
00510   bool m_outsideCone = false;
00511 #endif
00512
00513 #ifndef ___GPUCOMPILE___
00515   std::vector<double> m_wdtimes;
00516
00518   virtual void Reset() = 0;
00520   virtual void UpdatePeriodicity() = 0;
00521
00522  private:
00523   double IntegrateFluxParallelogram(const double x0, const double y0,
00524                                     const double z0, const double dx1,
00525                                     const double dy1, const double dz1,
00526                                     const double dx2, const double dy2,
00527                                     const double dz2, const unsigned int nU,
00528                                     const unsigned int nV, const bool wfield,
00529                                     const std::string& label);
00530 #else
00531
00532 // include parts from derived class due to big performance hit from using
00533 // virtual methods
00534 // TODO GPU TN: It isn't clear to me that we actually need (at least) the Ansys
00535 // include - all the code is protected by an ifndef ___GPUCOMPILE___, whereas it
00536 // will be defined when these includes are made
00537 #include "ComponentAnsys123.hh"
00538 #include "ComponentFieldMap.hh"
00539
00540   friend class ComponentAnsys123;
00541   friend class ComponentComsol;
00542   friend class ComponentElmer;
00543   friend class ComponentFieldMap;
00544   friend class Component;
00545
00546   // enum to mimic polymorphism
00547   enum class ComponentType {
00548     Component = 0,
00549     ComponentFieldMap,
00550     ComponentAnsys123,
```

```
00551     ComponentComsol,
00552     ComponentElmer
00553   };
00554
00555   ComponentType m_ComponentType{ComponentType::Component};
00556
00557 #endif
00558 };
00559 }  // namespace Garfield
00560
00561 #endif
```

## 8.11 /builds/garfield/docs/source/Include/Garfield/ComponentAnalyticField.hh File Reference

#include <array>
#include <complex>
#include <mutex>
#include <string>
#include <vector>
#include "Garfield/Component.hh"

Classes

- class Garfield::ComponentAnalyticField

  Semi-analytic calculation of two-dimensional configurations consisting of wires, planes, and tubes.
- struct Garfield::ComponentAnalyticField::Wire
- struct Garfield::ComponentAnalyticField::Strip
- struct Garfield::ComponentAnalyticField::Pixel
- struct Garfield::ComponentAnalyticField::Plane
- struct Garfield::ComponentAnalyticField::Charge3d

Namespaces

- namespace Garfield

## 8.12 ComponentAnalyticField.hh

Go to the documentation of this file.

```
00001 #ifndef G_COMPONENT_ANALYTIC_FIELD_H
00002 #define G_COMPONENT_ANALYTIC_FIELD_H
00003
00004 #include <array>
00005 #include <complex>
00006 #include <mutex>
00007 #include <string>
00008 #include <vector>
00009
00010 #include "Garfield/Component.hh"
00011
00012 class TPad;
00013
00014 namespace Garfield {
00015
00016 class Medium;
00017
00020
00021 class ComponentAnalyticField : public Component {
00022  public:
00024   ComponentAnalyticField();
00026   ~ComponentAnalyticField() {}
00027
00029   void SetMedium(Medium* medium) { m_medium = medium; }
00031   void AddWire(const double x, const double y, const double diameter,
00032                const double voltage, const std::string& label = "",
00033                const double length = 100., const double tension = 50.,
00034                const double rho = 19.3, const int ntrap = 5);
00036   void AddTube(const double radius, const double voltage, const int nEdges,
00037                const std::string& label = "");
00039   void AddPlaneX(const double x, const double voltage,
00040                  const std::string& label = "");
```

```
00042   void AddPlaneY(const double y, const double voltage,
00043                  const std::string& label = "");
00045   void AddPlaneR(const double r, const double voltage,
00046                  const std::string& label = "");
00048   void AddPlanePhi(const double phi, const double voltage,
00049                    const std::string& label = "");
00058   void AddStripOnPlaneX(const char direction, const double x, const double smin,
00059                         const double smax, const std::string& label,
00060                         const double gap = -1.);
00062   void AddStripOnPlaneY(const char direction, const double y, const double smin,
00063                         const double smax, const std::string& label,
00064                         const double gap = -1.);
00067   void AddStripOnPlaneR(const char direction, const double r, const double smin,
00068                         const double smax, const std::string& label,
00069                         const double gap = -1.);
00071   void AddStripOnPlanePhi(const char direction, const double phi,
00072                           const double smin, const double smax,
00073                           const std::string& label, const double gap = -1.);
00084   void AddPixelOnPlaneX(const double x, const double ymin, const double ymax,
00085                         const double zmin, const double zmax,
00086                         const std::string& label, const double gap = -1.,
00087                         const double rot = 0.);
00089   void AddPixelOnPlaneY(const double y, const double xmin, const double xmax,
00090                         const double zmin, const double zmax,
00091                         const std::string& label, const double gap = -1.,
00092                         const double rot = 0.);
00094   void AddPixelOnPlaneR(const double r, const double phimin,
00095                         const double phimax, const double zmin,
00096                         const double zmax, const std::string& label,
00097                         const double gap = -1.);
00099   void AddPixelOnPlanePhi(const double phi, const double rmin,
00100                           const double rmax, const double zmin,
00101                           const double zmax, const std::string& label,
00102                           const double gap = -1.);
00103
00105   void SetPeriodicityX(const double s);
00107   void SetPeriodicityY(const double s);
00109   void SetPeriodicityPhi(const double phi);
00111   bool GetPeriodicityX(double& s);
00113   bool GetPeriodicityY(double& s);
00115   bool GetPeriodicityPhi(double& s);
00116
00118   void SetCartesianCoordinates();
00120   void SetPolarCoordinates();
00122   bool IsPolar() const { return m_polar; }
00123
00125   void PrintCell();
00127   void PlotCell(TPad* pad);
00128
00130   void AddCharge(const double x, const double y, const double z,
00131                  const double q);
00133   void ClearCharges();
00135   void PrintCharges() const;
00136
00156   std::string GetCellType() {
00157     if (!m_cellset) {
00158       if (CellCheck()) CellType();
00159     }
00160     return GetCellType(m_cellType);
00161   }
00162
00165   void AddReadout(const std::string& label, const bool silent = false);
00166
00167   void SetNumberOfCellCopies(const unsigned int nfourier);
00168
00179   bool MultipoleMoments(const unsigned int iw, const unsigned int order = 4,
00180                         const bool print = false, const bool plot = false,
00181                         const double rmult = 1., const double eps = 1.e-4,
00182                         const unsigned int nMaxIter = 20);
00184   void EnableDipoleTerms(const bool on = true);
00186   void EnableChargeCheck(const bool on = true) { m_chargeCheck = on; }
00187
00189   unsigned int GetNumberOfWires() const { return m_nWires; }
00191   bool GetWire(const unsigned int i, double& x, double& y, double& diameter,
00192                double& voltage, std::string& label, double& length,
00193                double& charge, int& ntrap) const;
00194
00196   unsigned int GetNumberOfPlanesX() const;
00198   unsigned int GetNumberOfPlanesY() const;
00200   unsigned int GetNumberOfPlanesR() const;
00202   unsigned int GetNumberOfPlanesPhi() const;
00204   bool GetPlaneX(const unsigned int i, double& x, double& voltage,
00205                  std::string& label) const;
00207   bool GetPlaneY(const unsigned int i, double& y, double& voltage,
00208                  std::string& label) const;
00210   bool GetPlaneR(const unsigned int i, double& r, double& voltage,
00211                  std::string& label) const;
```

```
00213   bool GetPlanePhi(const unsigned int i, double& phi, double& voltage,
00214                    std::string& label) const;
00216   bool GetTube(double& r, double& voltage, int& nEdges,
00217                std::string& label) const;
00218
00232   bool OptimiseOnTrack(const std::vector<std::string>& groups,
00233                        const std::string& field_function, const double target,
00234                        const double x0, const double y0, const double x1,
00235                        const double y1, const unsigned int nP = 20,
00236                        const bool print = true);
00239   bool OptimiseOnGrid(const std::vector<std::string>& groups,
00240                       const std::string& field_function, const double target,
00241                       const double x0, const double y0, const double x1,
00242                       const double y1, const unsigned int nX = 10,
00243                       const unsigned int nY = 10, const bool print = true);
00246   bool OptimiseOnWires(const std::vector<std::string>& groups,
00247                        const std::string& field_function, const double target,
00248                        const std::vector<unsigned int>& wires,
00249                        const bool print = true);
00250
00257   void SetOptimisationParameters(const double dist = 1.,
00258                                  const double eps = 1.e-4,
00259                                  const unsigned int nMaxIter = 10);
00260
00263   bool ElectricFieldAtWire(const unsigned int iw, double& ex, double& ey);
00264
00267   void SetScanningGrid(const unsigned int nX, const unsigned int nY);
00270   void SetScanningArea(const double xmin, const double xmax, const double ymin,
00271                        const double ymax);
00274   void SetScanningAreaLargest() { m_scanRange = ScanningRange::Largest; }
00277   void SetScanningAreaFirstOrder(const double scale = 2.);
00280   void EnableExtrapolation(const bool on = true) { m_extrapolateForces = on; }
00281
00283   void SetGravity(const double dx, const double dy, const double dz);
00285   void GetGravity(double& dx, double& dy, double& dz) const;
00287   void EnableGravity(const bool on = true) { m_useGravitationalForce = on; }
00289   void EnableElectrostaticForce(const bool on = true) {
00290     m_useElectrostaticForce = on;
00291   }
00299   bool ForcesOnWire(const unsigned int iw, std::vector<double>& xMap,
00300                     std::vector<double>& yMap,
00301                     std::vector<std::vector<double> >& fxMap,
00302                     std::vector<std::vector<double> >& fyMap);
00313   bool WireDisplacement(const unsigned int iw, const bool detailed,
00314                         std::vector<double>& csag, std::vector<double>& xsag,
00315                         std::vector<double>& ysag, double& stretch,
00316                         const bool print = true);
00319   void SetNumberOfShots(const unsigned int n) { m_nShots = n; }
00321   void SetNumberOfSteps(const unsigned int n);
00322
00323   Medium* GetMedium(const double x, const double y, const double z) override;
00324   void ElectricField(const double x, const double y, const double z, double& ex,
00325                      double& ey, double& ez, Medium*& m, int& status) override;
00326
00327   void ElectricField(const double x, const double y, const double z, double& ex,
00328                      double& ey, double& ez, double& v, Medium*& m,
00329                      int& status) override;
00330
00331   using Component::ElectricField;
00332   bool GetVoltageRange(double& pmin, double& pmax) override;
00333
00334   void WeightingField(const double x, const double y, const double z,
00335                       double& wx, double& wy, double& wz,
00336                       const std::string& label) override {
00337     wx = wy = wz = 0.;
00338     if (!m_sigset) PrepareSignals();
00339     Wfield(x, y, z, wx, wy, wz, label);
00340   }
00341   double WeightingPotential(const double x, const double y, const double z,
00342                             const std::string& label) override {
00343     if (!m_sigset) PrepareSignals();
00344     return Wpot(x, y, z, label);
00345   }
00346
00347   bool GetBoundingBox(double& x0, double& y0, double& z0, double& x1,
00348                       double& y1, double& z1) override;
00349   bool GetElementaryCell(double& x0, double& y0, double& z0, double& x1,
00350                          double& y1, double& z1) override;
00351
00352   bool CrossedWire(const double x0, const double y0, const double z0,
00353                    const double x1, const double y1, const double z1,
00354                    double& xc, double& yc, double& zc, const bool centre,
00355                    double& rc) override;
00356
00357   bool InTrapRadius(const double q0, const double x0, const double y0,
00358                     const double z0, double& xw, double& yx,
00359                     double& rw) override;
```

```
00360
00361   bool CrossedPlane(const double x0, const double y0, const double z0,
00362                     const double x1, const double y1, const double z1,
00363                     double& xc, double& yc, double& zc) override;
00364
00365   enum Cell {
00366     A00,
00367     B1X,
00368     B1Y,
00369     B2X,
00370     B2Y,
00371     C10,
00372     C2X,
00373     C2Y,
00374     C30,
00375     D10,
00376     D20,
00377     D30,
00378     D40,
00379     Unknown
00380   };
00381
00382   double StepSizeHint() override;
00383
00384  private:
00385   std::mutex m_mutex;
00386
00387   Medium* m_medium = nullptr;
00388
00389   bool m_chargeCheck = false;
00390
00391   bool m_cellset = false;
00392   bool m_sigset = false;
00393
00394   bool m_polar = false;
00395
00396   // Cell type.
00397   Cell m_cellType;
00398
00399   // Bounding box
00400   double m_xmin, m_xmax;
00401   double m_ymin, m_ymax;
00402   double m_zmin, m_zmax;
00403
00404   // Voltage range
00405   double m_vmin, m_vmax;
00406
00407   // Periodicities
00408   bool m_perx = false;
00409   bool m_pery = false;
00410   double m_sx, m_sy;
00411
00412   // Signals
00413   int m_nFourier = 1;
00414   Cell m_cellTypeFourier = A00;
00415   bool m_fperx = false;
00416   bool m_fpery = false;
00417   int m_mxmin = 0;
00418   int m_mxmax = 0;
00419   int m_mymin = 0;
00420   int m_mymax = 0;
00421   int m_mfexp = 0;
00422
00423   std::vector<std::string> m_readout;
00424
00425   // Wires
00426   unsigned int m_nWires;
00427   struct Wire {
00428     double x, y;
00429     double r;
00430     double v;
00431     double e;
00432     std::string type;
00433     double u;
00434     int ind;
00436     int nTrap;
00437     double tension;
00438     double density;
00439   };
00440   std::vector<Wire> m_w;
00441
00442   // Option for computation of dipole terms
00443   bool m_dipole = false;
00444   // Dipole angle and amplitude
00445   std::vector<double> m_cosph2;
00446   std::vector<double> m_sinph2;
00447   std::vector<double> m_amp2;
```

```
00448
00449   // Parameters for B2 type cells
00450   std::vector<double> m_b2sin;
00451   // Parameters for C type cells
00452   int m_mode;
00453   std::complex<double> m_zmult;
00454   double m_p1, m_p2, m_c1;
00455   // Parameters for D3 type cells
00456   // Wire positions in conformal mapping
00457   std::vector<std::complex<double> > m_zw;
00458   double m_kappa;
00459
00460   // Reference potential
00461   double m_v0;
00462   double m_corvta, m_corvtb, m_corvtc;
00463
00464   // Planes
00465   // Existence
00466   bool m_ynplan[4];
00467   bool m_ynplax, m_ynplay;
00468   // Coordinates
00469   double m_coplan[4];
00470   double m_coplax, m_coplay;
00471   // Voltages
00472   double m_vtplan[4];
00473
00474   struct Strip {
00475     std::string type;
00476     int ind;
00477     double smin, smax;
00478     double gap;
00479   };
00480
00481   struct Pixel {
00482     std::string type;
00483     int ind = 0;
00484     double smin = 0., smax = 0.;
00485     double zmin = 0., zmax = 0.;
00486     double gap = -1.;
00487     double cphi = 1.;
00488     double sphi = 0.;
00489   };
00490
00491   struct Plane {
00492     std::string type;
00493     int ind;
00494     double ewxcor, ewycor;
00495     std::vector<Strip> strips1;
00496     std::vector<Strip> strips2;
00497     std::vector<Pixel> pixels;
00498   };
00499   std::array<Plane, 5> m_planes;
00500
00501   // Tube
00502   bool m_tube = false;
00503   int m_mtube = 0;
00504   int m_ntube = 1;
00505   double m_cotube = 1.;
00506   double m_cotube2 = 1.;
00507   double m_vttube = 0.;
00508
00509   // Smallest dimension.
00510   double m_dmin = -1.;
00511
00512   // Wire weighting charges.
00513   std::vector<std::vector<std::vector<double> > > m_qwire;
00514   // Plane weighting charges.
00515   std::vector<std::vector<std::vector<double> > > m_qplane;
00516
00517   // Point charges
00518   struct Charge3d {
00519     double x, y, z;
00520     double e;
00521   };
00522   std::vector<Charge3d> m_ch3d;
00523   unsigned int m_nTermBessel = 10;
00524   unsigned int m_nTermPoly = 100;
00525
00526   bool m_useElectrostaticForce = true;
00527   bool m_useGravitationalForce = true;
00528   // Gravity
00529   std::array<double, 3> m_down{{0, 0, 1}};
00530   // Number of shots used for solving the wire sag differential equations
00531   unsigned int m_nShots = 2;
00532   // Number of integration steps within each shot.
00533   unsigned int m_nSteps = 20;
00534   // Options for setting the range of wire shifts
```

```
00535   // for which the forces are computed.
00536   enum class ScanningRange { Largest = 0, FirstOrder, User };
00537   ScanningRange m_scanRange = ScanningRange::FirstOrder;
00538   // Limits of the user-specified scanning range.
00539   double m_xScanMin = 0.;
00540   double m_xScanMax = 0.;
00541   double m_yScanMin = 0.;
00542   double m_yScanMax = 0.;
00543   // Scaling factor for first-order estimate of the scanning range.
00544   double m_scaleRange = 2.;
00545   // Number of grid lines at which the forces are stored.
00546   unsigned int m_nScanX = 11;
00547   unsigned int m_nScanY = 11;
00548   // Extrapolate beyond the scanning range or not.
00549   bool m_extrapolateForces = false;
00550
00551   // Maximum deviation between target and field function at which
00552   // to allow the iteration to stop.
00553   double m_optDist = 1.;
00554   // Relative change in Euclidean distance between target and
00555   // field function at which to allow the iteration to stop.
00556   double m_optEps = 1.e-4;
00557   // Maximum number of iterations in the optimisation fit.
00558   unsigned int m_optNitmax = 10;
00559
00560   void UpdatePeriodicity() override;
00561   void Reset() override {
00562     CellInit();
00563     m_medium = nullptr;
00564   }
00565
00566   void CellInit();
00567   bool Prepare();
00568   bool CellCheck();
00569   bool WireCheck() const;
00570   bool CellType();
00571   std::string GetCellType(const Cell) const;
00572   bool PrepareStrips();
00573   bool PrepareSignals();
00574   bool SetupWireSignals();
00575   bool SetupPlaneSignals();
00576
00577   // Calculation of charges
00578   bool Setup();
00579   bool Update(const std::vector<double>& vw, const std::array<double, 5>& vp);
00580   bool SetupA00();
00581   bool SetupB1X();
00582   bool SetupB1Y();
00583   bool SetupB2X();
00584   bool SetupB2Y();
00585   bool SetupC10();
00586   bool SetupC2X();
00587   bool SetupC2Y();
00588   bool SetupC30();
00589   bool SetupD10();
00590   bool SetupD20();
00591   bool SetupD30();
00592
00593   bool IprA00(const int mx, const int my,
00594               std::vector<std::vector<std::complex<double> > >& mat);
00595   bool IprB2X(const int my,
00596               std::vector<std::vector<std::complex<double> > >& mat);
00597   bool IprB2Y(const int mx,
00598               std::vector<std::vector<std::complex<double> > >& mat);
00599   bool IprC2X(std::vector<std::vector<std::complex<double> > >& mat);
00600   bool IprC2Y(std::vector<std::vector<std::complex<double> > >& mat);
00601   bool IprC30(std::vector<std::vector<std::complex<double> > >& mat);
00602   bool IprD10(std::vector<std::vector<std::complex<double> > >& mat);
00603   bool IprD30(std::vector<std::vector<std::complex<double> > >& mat);
00604
00605   bool SetupDipoleTerms();
00606
00607   // Inversion of the capacitance matrix
00608   bool Charge(std::vector<std::vector<double> >& mat);
00609
00610   // Evaluation of the electric field
00611   int Field(const double xin, const double yin, const double zin, double& ex,
00612             double& ey, double& ez, double& volt, const bool opt);
00613   void FieldA00(const double xpos, const double ypos, double& ex, double& ey,
00614                 double& volt, const bool opt) const;
00615   void FieldB1X(const double xpos, const double ypos, double& ex, double& ey,
00616                 double& volt, const bool opt) const;
00617   void FieldB1Y(const double xpos, const double ypos, double& ex, double& ey,
00618                 double& volt, const bool opt) const;
00619   void FieldB2X(const double xpos, const double ypos, double& ex, double& ey,
00620                 double& volt, const bool opt) const;
00621   void FieldB2Y(const double xpos, const double ypos, double& ex, double& ey,
```

```
00622               double& volt, const bool opt) const;
00623   void FieldC10(const double xpos, const double ypos, double& ex, double& ey,
00624               double& volt, const bool opt) const;
00625   void FieldC2X(const double xpos, const double ypos, double& ex, double& ey,
00626               double& volt, const bool opt) const;
00627   void FieldC2Y(const double xpos, const double ypos, double& ex, double& ey,
00628               double& volt, const bool opt) const;
00629   void FieldC30(const double xpos, const double ypos, double& ex, double& ey,
00630               double& volt, const bool opt) const;
00631   void FieldD10(const double xpos, const double ypos, double& ex, double& ey,
00632               double& volt, const bool opt) const;
00633   void FieldD20(const double xpos, const double ypos, double& ex, double& ey,
00634               double& volt, const bool opt) const;
00635   void FieldD30(const double xpos, const double ypos, double& ex, double& ey,
00636               double& volt, const bool opt) const;
00637
00638   // Field due to point charges
00639   void Field3dA00(const double x, const double y, const double z, double& ex,
00640                 double& ey, double& ez, double& volt) const;
00641   void Field3dB2X(const double x, const double y, const double z, double& ex,
00642                 double& ey, double& ez, double& volt) const;
00643   void Field3dB2Y(const double x, const double y, const double z, double& ex,
00644                 double& ey, double& ez, double& volt) const;
00645   void Field3dD10(const double x, const double y, const double z, double& ex,
00646                 double& ey, double& ez, double& volt) const;
00647   // Evaluation of the weighting field
00648   bool Wfield(const double x, const double y, const double z, double& ex,
00649             double& ey, double& ez, const std::string& label) const;
00650   void WfieldWireA00(const double x, const double y, double& ex, double& ey,
00651                    const int mx, const int my,
00652                    const std::vector<double>& qw) const;
00653   void WfieldWireB2X(const double x, const double y, double& ex, double& ey,
00654                    const int my, const std::vector<double>& qw) const;
00655   void WfieldWireB2Y(const double x, const double y, double& ex, double& ey,
00656                    const int mx, const std::vector<double>& qw) const;
00657   void WfieldWireC2X(const double x, const double y, double& ex, double& ey,
00658                    const std::vector<double>& qw) const;
00659   void WfieldWireC2Y(const double x, const double y, double& ex, double& ey,
00660                    const std::vector<double>& qw) const;
00661   void WfieldWireC30(const double x, const double y, double& ex, double& ey,
00662                    const std::vector<double>& qw) const;
00663   void WfieldWireD10(const double x, const double y, double& ex, double& ey,
00664                    const std::vector<double>& qw) const;
00665   void WfieldWireD30(const double x, const double y, double& ex, double& ey,
00666                    const std::vector<double>& qw) const;
00667   void WfieldPlaneA00(const double x, const double y, double& ex, double& ey,
00668                     const int mx, const int my,
00669                     const std::vector<double>& qp) const;
00670   void WfieldPlaneB2X(const double x, const double y, double& ex, double& ey,
00671                     const int my, const std::vector<double>& qp) const;
00672   void WfieldPlaneB2Y(const double x, const double ypos, double& ex, double& ey,
00673                     const int mx, const std::vector<double>& qp) const;
00674   void WfieldPlaneC2X(const double x, const double y, double& ex, double& ey,
00675                     const std::vector<double>& qp) const;
00676   void WfieldPlaneC2Y(const double x, const double y, double& ex, double& ey,
00677                     const std::vector<double>& qp) const;
00678   void WfieldPlaneC30(const double x, const double y, double& ex, double& ey,
00679                     const std::vector<double>& qp) const;
00680   void WfieldPlaneD10(const double x, const double y, double& ex, double& ey,
00681                     const std::vector<double>& qp) const;
00682   void WfieldPlaneD30(const double x, const double y, double& ex, double& ey,
00683                     const std::vector<double>& qp) const;
00684   void WfieldStripZ(const double x, const double y, double& ex, double& ey,
00685                   const int ip, const Strip& strip) const;
00686   void WfieldStripXy(const double x, const double y, const double z, double& ex,
00687                    double& ey, double& ez, const int ip,
00688                    const Strip& strip) const;
00689   void WfieldStrip(const double x, const double y, const double g,
00690                  const double w, double& fx, double& fy) const;
00691   void WfieldPixel(const double x, const double y, const double z, double& ex,
00692                  double& ey, double& ez, const int ip,
00693                  const Pixel& pixel) const;
00694
00695   // Evaluation of the weighting potential.
00696   double Wpot(const double x, const double y, const double z,
00697             const std::string& label) const;
00698   double WpotWireA00(const double x, const double y, const int mx, const int my,
00699                    const std::vector<double>& qw) const;
00700   double WpotWireB2X(const double x, const double y, const int my,
00701                    const std::vector<double>& qw) const;
00702   double WpotWireB2Y(const double x, const double y, const int mx,
00703                    const std::vector<double>& qw) const;
00704   double WpotWireC2X(const double x, const double y,
00705                    const std::vector<double>& qw) const;
00706   double WpotWireC2Y(const double x, const double y,
00707                    const std::vector<double>& qw) const;
00708   double WpotWireC30(const double x, const double y,
```

```
00709                   const std::vector<double>& qw) const;
00710   double WpotWireD10(const double x, const double y,
00711                   const std::vector<double>& qw) const;
00712   double WpotWireD30(const double x, const double y,
00713                   const std::vector<double>& qw) const;
00714   double WpotPlaneA00(const double x, const double y, const int mx,
00715                    const int my, const std::vector<double>& qp) const;
00716   double WpotPlaneB2X(const double x, const double y, const int my,
00717                    const std::vector<double>& qp) const;
00718   double WpotPlaneB2Y(const double x, const double y, const int mx,
00719                    const std::vector<double>& qp) const;
00720   double WpotPlaneC2X(const double x, const double y,
00721                    const std::vector<double>& qp) const;
00722   double WpotPlaneC2Y(const double x, const double y,
00723                    const std::vector<double>& qp) const;
00724   double WpotPlaneC30(const double x, const double y,
00725                    const std::vector<double>& qp) const;
00726   double WpotPlaneD10(const double x, const double y,
00727                    const std::vector<double>& qp) const;
00728   double WpotPlaneD30(const double x, const double y,
00729                    const std::vector<double>& qp) const;
00730   double WpotStripZ(const double x, const double y, const int ip,
00731                  const Strip& strip) const;
00732   double WpotStripXy(const double x, const double y, const double z,
00733                    const int ip, const Strip& strip) const;
00734   double WpotPixel(const double x, const double y, const double z, const int ip,
00735                  const Pixel& pixel) const;
00736
00737   // Functions for calculating the electric field at a given wire position,
00738   // as if the wire itself were not there but with the presence
00739   // of its mirror images.
00740   void FieldAtWireA00(const double xpos, const double ypos, double& ex,
00741                    double& ey, const std::vector<bool>& cnalso) const;
00742   void FieldAtWireB1X(const double xpos, const double ypos, double& ex,
00743                    double& ey, const std::vector<bool>& cnalso) const;
00744   void FieldAtWireB1Y(const double xpos, const double ypos, double& ex,
00745                    double& ey, const std::vector<bool>& cnalso) const;
00746   void FieldAtWireB2X(const double xpos, const double ypos, double& ex,
00747                    double& ey, const std::vector<bool>& cnalso) const;
00748   void FieldAtWireB2Y(const double xpos, const double ypos, double& ex,
00749                    double& ey, const std::vector<bool>& cnalso) const;
00750   void FieldAtWireC10(const double xpos, const double ypos, double& ex,
00751                    double& ey, const std::vector<bool>& cnalso) const;
00752   void FieldAtWireC2X(const double xpos, const double ypos, double& ex,
00753                    double& ey, const std::vector<bool>& cnalso) const;
00754   void FieldAtWireC2Y(const double xpos, const double ypos, double& ex,
00755                    double& ey, const std::vector<bool>& cnalso) const;
00756   void FieldAtWireC30(const double xpos, const double ypos, double& ex,
00757                    double& ey, const std::vector<bool>& cnalso) const;
00758   void FieldAtWireD10(const double xpos, const double ypos, double& ex,
00759                    double& ey, const std::vector<bool>& cnalso) const;
00760   void FieldAtWireD20(const double xpos, const double ypos, double& ex,
00761                    double& ey, const std::vector<bool>& cnalso) const;
00762   void FieldAtWireD30(const double xpos, const double ypos, double& ex,
00763                    double& ey, const std::vector<bool>& cnalso) const;
00764
00765   void DipoleFieldA00(const double xpos, const double ypos, double& ex,
00766                    double& ey, double& volt, const bool opt) const;
00767   void DipoleFieldB1X(const double xpos, const double ypos, double& ex,
00768                    double& ey, double& volt, const bool opt) const;
00769   void DipoleFieldB1Y(const double xpos, const double ypos, double& ex,
00770                    double& ey, double& volt, const bool opt) const;
00771   void DipoleFieldB2X(const double xpos, const double ypos, double& ex,
00772                    double& ey, double& volt, const bool opt) const;
00773   void DipoleFieldB2Y(const double xpos, const double ypos, double& ex,
00774                    double& ey, double& volt, const bool opt) const;
00775
00776   // Auxiliary functions for C type cells
00777   double Ph2(const double xpos, const double ypos) const;
00778   double Ph2Lim(const double radius) const {
00779     return -log(abs(m_zmult) * radius * (1. - 3. * m_p1 + 5. * m_p2));
00780   }
00781   void E2Sum(const double xpos, const double ypos, double& ex,
00782              double& ey) const;
00783
00784   // Mapping function for D30 type cells
00785   void ConformalMap(const std::complex<double>& z, std::complex<double>& ww,
00786                     std::complex<double>& wd) const;
00787
00788   static bool InTube(const double x0, const double y0, const double a,
00789                      const int n);
00790
00791   bool SagDetailed(const Wire& wire, const std::vector<double>& xMap,
00792                    const std::vector<double>& yMap,
00793                    const std::vector<std::vector<double> >& fxMap,
00794                    const std::vector<std::vector<double> >& fyMap,
00795                    std::vector<double>& csag, std::vector<double>& xsag,
```

```
00796                 std::vector<double>& ysag) const;
00797   bool GetForceRatio(const Wire& wire, const double coor,
00798                      const std::array<double, 2>& bend,
00799                      const std::array<double, 2>& dbend,
00800                      std::array<double, 2>& f, const std::vector<double>& xMap,
00801                      const std::vector<double>& yMap,
00802                      const std::vector<std::vector<double> >& fxMap,
00803                      const std::vector<std::vector<double> >& fyMap) const;
00804   bool FindZeroes(const Wire& wire, const double h, std::vector<double>& x,
00805                   const std::vector<double>& xMap,
00806                   const std::vector<double>& yMap,
00807                   const std::vector<std::vector<double> >& fxMap,
00808                   const std::vector<std::vector<double> >& fyMap) const;
00809   bool StepRKN(const Wire& wire, const double h, double& x,
00810                std::array<double, 2>& y, std::array<double, 2>& yp,
00811                const std::vector<double>& xMap, const std::vector<double>& yMap,
00812                const std::vector<std::vector<double> >& fxMap,
00813                const std::vector<std::vector<double> >& fyMap) const;
00814   bool Trace(const Wire& wire, const double h, const std::vector<double>& xx,
00815              std::vector<double>& f, const std::vector<double>& xMap,
00816              const std::vector<double>& yMap,
00817              const std::vector<std::vector<double> >& fxMap,
00818              const std::vector<std::vector<double> >& fyMap) const;
00819   size_t SignalLayer(const int mx, const int my) const;
00820
00821   void InitialiseFitParameters(
00822       const std::vector<std::string>& groups, std::vector<double>& vw0,
00823       std::array<double, 5>& vp0, std::vector<double>& aFit,
00824       std::vector<std::vector<unsigned int> >& wiresInGroup,
00825       std::vector<std::vector<unsigned int> >& planesInGroup);
00826 };
00827 }  // namespace Garfield
00828
00829 #endif
```

## 8.13 /builds/garfield/docs/source/Include/Garfield/ComponentAnsys121.hh File Reference

#include "Garfield/ComponentFieldMap.hh"

### Classes

- class Garfield::ComponentAnsys121

  Component for importing and interpolating two-dimensional ANSYS field maps.

### Namespaces

- namespace Garfield

## 8.14 ComponentAnsys121.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__ANSYS121__H
00002 #define G__COMPONENT__ANSYS121__H
00003
00004 #include "Garfield/ComponentFieldMap.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class ComponentAnsys121 : public ComponentFieldMap {
00011  public:
00013   ComponentAnsys121();
00015   ~ComponentAnsys121() {}
00016
00024   bool Initialise(const std::string& elist = "ELIST.lis",
00025                   const std::string& nlist = "NLIST.lis",
00026                   const std::string& mplist = "MPLIST.lis",
00027                   const std::string& prnsol = "PRNSOL.lis",
00028                   const std::string& unit = "cm");
00030   bool SetWeightingPotential(const std::string& prnsol,
00031                              const std::string& label) {
00032     return SetWeightingField(prnsol, label);
00033   }
00034   bool SetWeightingField(const std::string& prnsol, const std::string& label);
00036   void SetRangeZ(const double zmin, const double zmax);
00037
```

```
00038  private:
00039   bool LoadPotentials(const std::string& prnsol, std::vector<double>& pot);
00040 };
00041 }  // namespace Garfield
00042
00043 #endif
```

## 8.15 /builds/garfield/docs/source/Include/Garfield/ComponentAnsys123.hh File Reference

#include ”Garfield/ComponentFieldMap.hh”

Classes

- class Garfield::ComponentAnsys123

  Component for importing and interpolating three-dimensional ANSYS field maps.

Namespaces

- namespace Garfield

## 8.16 ComponentAnsys123.hh

Go to the documentation of this file.

```
00001 #if defined(___GPUCOMPILE___) || !defined(G__COMPONENT__ANSYS123__H)
00002
00003 #if !defined(___GPUCOMPILE___) && !defined(G__COMPONENT__ANSYS123__H)
00004 #define G__COMPONENT__ANSYS123__H
00005 #endif
00006
00007 #ifndef ___GPUCOMPILE___
00008
00009 #include ”Garfield/ComponentFieldMap.hh”
00010
00011 namespace Garfield {
00012
00015
00016 class ComponentAnsys123 : public ComponentFieldMap {
00017  public:
00019   ComponentAnsys123();
00021   ~ComponentAnsys123() {}
00022
00030   bool Initialise(const std::string& elist = ”ELIST.lis”,
00031                   const std::string& nlist = ”NLIST.lis”,
00032                   const std::string& mplist = ”MPLIST.lis”,
00033                   const std::string& prnsol = ”PRNSOL.lis”,
00034                   const std::string& unit = ”cm”);
00035
00037   bool SetWeightingPotential(const std::string& prnsol,
00038                              const std::string& label) {
00039     return SetWeightingField(prnsol, label);
00040   }
00041   bool SetWeightingField(const std::string& prnsol, const std::string& label);
00042
00044   double CreateGPUTransferObject(ComponentGPU*& comp_gpu) override;
00045
00046  private:
00047   bool LoadPotentials(const std::string prnsol, std::vector<double>& pot);
00048 };
00049 }  // namespace Garfield
00050 #endif
00051 #endif
```

## 8.17 /builds/garfield/docs/source/Include/Garfield/ComponentComsol.hh File Reference

#include ”Garfield/ComponentFieldMap.hh”

Classes

- class Garfield::ComponentComsol

Component for importing and interpolating Comsol field maps.

- struct Garfield::ComponentComsol::Range

Namespaces

- namespace Garfield

## 8.18 ComponentComsol.hh

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Garfield/ComponentFieldMap.hh"
00004
00005 namespace Garfield {
00006
00008
00009 class ComponentComsol : public ComponentFieldMap {
00010  public:
00012   ComponentComsol();
00014   ComponentComsol(const std::string &mesh, const std::string &mplist,
00015                   const std::string &field, const std::string &unit = "m");
00017   ~ComponentComsol() {}
00018
00019   void SetImportRange(const double xmin, const double xmax, const double ymin,
00020                       const double ymax, const double zmin, const double zmax) {
00021     // TODO: Must happen before initialise function
00022     m_range.set = true;
00023     m_range.xmin = xmin;
00024     m_range.ymin = ymin;
00025     m_range.zmin = zmin;
00026
00027     m_range.xmax = xmax;
00028     m_range.ymax = ymax;
00029     m_range.zmax = zmax;
00030   }
00031
00038   bool Initialise(const std::string &header = "mesh.mphtxt",
00039                   const std::string &mplist = "dielectrics.dat",
00040                   const std::string &field = "field.txt",
00041                   const std::string &unit = "m");
00042
00044   bool SetWeightingPotential(const std::string &file, const std::string &label);
00045   bool SetWeightingField(const std::string &file, const std::string &label) {
00046     return SetWeightingPotential(file, label);
00047   }
00049   bool SetDynamicWeightingPotential(const std::string &file,
00050                                     const std::string &label);
00052   void SetTimeInterval(const double mint, const double maxt,
00053                        const double stept);
00055   void GetTimeInterval(std::vector<double> &delayedTimes) {
00056     delayedTimes = m_wdtimes;
00057   }
00058
00061   void SetImportMaterial(const double epsr = 1.) {
00062     m_materialSelect = true;
00063     m_epsr = epsr;
00064   }
00065
00066 #ifdef USEGPU
00068   double CreateGPUTransferObject(ComponentGPU *&comp_gpu) override;
00069
00070 #endif
00071  private:
00072   double m_unit = 100.;
00073   bool m_timeset = false;
00074   static constexpr double MaxNodeDistance = 1.e-8;
00075
00076   bool m_materialSelect = false;
00077   double m_epsr = 1.;
00078
00079   bool GetTimeInterval(const std::string &file);
00080
00081   struct Range {
00082     bool set = false;
00083
00084     double xmin = 0;
00085     double xmax = 0;
00086
00087     double ymin = 0;
00088     double ymax = 0;
00089
00090     double zmin = 0;
```

```
00091     double zmax = 0;
00092   };
00093
00094   Range m_range;
00095
00096   bool CheckInRange(const double x, const double y, const double z) const {
00097     if (!m_range.set) return true;
00098
00099     if (x < m_range.xmin || x > m_range.xmax || y < m_range.ymin ||
00100         y > m_range.ymax || z < m_range.zmin || z > m_range.zmax)
00101       return false;
00102
00103     return true;
00104   }
00105
00106   bool ElementInRange(const Element &element,
00107                       const std::vector<Node> &nodes) const {
00108     if (m_range.set) {
00109       for (size_t i = 0; i < 10; i++) {
00110         const Node &node = nodes[element.emap[i]];
00111         if (!CheckInRange(node.x, node.y, node.z)) return false;
00112       }
00113     }
00114     if (m_materialSelect && m_materials[element.matmap].eps != m_epsr)
00115       return false;
00116     return true;
00117   }
00118   bool LoadPotentials(const std::string &field, std::vector<double> &pot);
00119 };
00120 }  // namespace Garfield
```

## 8.19 /builds/garfield/docs/source/Include/Garfield/ComponentConstant.hh File Reference

#include <array>
#include <string>
#include "Garfield/Component.hh"

### Classes

- class Garfield::ComponentConstant

  Component with constant electric field.

### Namespaces

- namespace Garfield

## 8.20 ComponentConstant.hh

Go to the documentation of this file.

```
00001 #ifndef G_COMPONENT_CONSTANT_H
00002 #define G_COMPONENT_CONSTANT_H
00003
00004 #include <array>
00005 #include <string>
00006
00007 #include "Garfield/Component.hh"
00008
00009 namespace Garfield {
00010
00012
00013 class ComponentConstant : public Component {
00014  public:
00016   ComponentConstant();
00018   ~ComponentConstant() {}
00019
00021   void SetElectricField(const double ex, const double ey, const double ez);
00023   void SetPotential(const double x, const double y, const double z,
00024                     const double v = 0.);
00025
00027   void SetWeightingField(const double wx, const double wy, const double wz,
00028                          const std::string label);
00030   void SetWeightingPotential(const double x, const double y, const double z,
00031                              const double v = 0.);
00032
00035   void SetArea(const double xmin, const double ymin, const double zmin,
```

```
00036               const double xmax, const double ymax, const double zmax);
00038   void UnsetArea();
00040   void SetMedium(Medium* medium) { m_medium = medium; }
00041
00042   Medium* GetMedium(const double x, const double y, const double z) override;
00043   void ElectricField(const double x, const double y, const double z, double& ex,
00044                      double& ey, double& ez, Medium*& m, int& status) override;
00045   void ElectricField(const double x, const double y, const double z, double& ex,
00046                      double& ey, double& ez, double& v, Medium*& m,
00047                      int& status) override;
00048   using Component::ElectricField;
00049   bool GetVoltageRange(double& vmin, double& vmax) override;
00050   void WeightingField(const double x, const double y, const double z,
00051                       double& wx, double& wy, double& wz,
00052                       const std::string& label) override;
00053   double WeightingPotential(const double x, const double y, const double z,
00054                             const std::string& label) override;
00055
00056   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00057                       double& ymax, double& zmax) override;
00058
00059  private:
00060   // Electric field.
00061   std::array<double, 3> m_efield = {{0., 0., 0.}};
00062
00063   // Is the potential defined?
00064   bool m_hasPotential = false;
00065   // Point where potential was specified.
00066   double m_x0 = 0., m_y0 = 0., m_z0 = 0.;
00067   // Potential at this point.
00068   double m_v0 = 0.;
00069
00070   // Is the weighting field defined?
00071   bool m_hasWeightingField = false;
00072   // Identifier of the weighting field.
00073   std::string m_label = "";
00074   // Weighting field.
00075   std::array<double, 3> m_wfield = {{0., 0., 0.}};
00076   // Is the weighting potential defined?
00077   bool m_hasWeightingPotential = false;
00078   // Point where the weighting potential was specified.
00079   double m_wx0 = 0., m_wy0 = 0., m_wz0 = 0.;
00080   // Weighting potential at this point.
00081   double m_w0 = 0.;
00082
00083   // Active area.
00084   std::array<double, 3> m_xmin = {{0., 0., 0.}};
00085   std::array<double, 3> m_xmax = {{0., 0., 0.}};
00086   // Did we specify the active area explicitly?
00087   bool m_hasArea = false;
00088   // Medium in the active area.
00089   Medium* m_medium = nullptr;
00090
00091   void Reset() override;
00092   void UpdatePeriodicity() override;
00093
00094   bool InArea(const double x, const double y, const double z) {
00095     if (x < m_xmin[0] || x > m_xmax[0] || y < m_xmin[1] || y > m_xmax[1] ||
00096         z < m_xmin[2] || z > m_xmax[2]) {
00097       return false;
00098     }
00099     return true;
00100   }
00101 };
00102 }  // namespace Garfield
00103 #endif
```

## 8.21 /builds/garfield/docs/source/Include/Garfield/ComponentCST.hh File Reference

#include <map>
#include <vector>
#include "Garfield/ComponentFieldMap.hh"

### Classes

- class Garfield::ComponentCST

  Component for importing and interpolating field maps from CST.

Namespaces

- namespace Garfield

## 8.22 ComponentCST.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT_CST_H
00002 #define G__COMPONENT_CST_H
00003
00004 #include <map>
00005 #include <vector>
00006
00007 #include "Garfield/ComponentFieldMap.hh"
00008
00009 namespace Garfield {
00010
00015
00016 class ComponentCST : public ComponentFieldMap {
00017  public:
00019   ComponentCST();
00021   ~ComponentCST() {}
00022
00023   void ShiftComponent(const double xShift, const double yShift,
00024                       const double zShift);
00025
00026   void GetNumberOfMeshLines(unsigned int& nx, unsigned int& ny,
00027                             unsigned int& nz) const;
00028   size_t GetNumberOfElements() const override { return m_nElements; }
00029   bool GetElementNodes(const size_t i,
00030                        std::vector<size_t>& nodes) const override;
00031   bool GetElementRegion(const size_t i, size_t& mat,
00032                         bool& drift) const override;
00033   size_t GetNumberOfNodes() const override { return m_nNodes; }
00034   bool GetNode(const size_t i, double& x, double& y, double& z) const override;
00035
00036   void GetElementBoundaries(unsigned int element, double& xmin, double& xmax,
00037                             double& ymin, double& ymax, double& zmin,
00038                             double& zmax) const;
00039
00040   Medium* GetMedium(const double x, const double y, const double z) override;
00041   void ElectricField(const double x, const double y, const double z, double& ex,
00042                      double& ey, double& ez, Medium*& m, int& status) override;
00043   void ElectricField(const double x, const double y, const double z, double& ex,
00044                      double& ey, double& ez, double& v, Medium*& m,
00045                      int& status) override;
00046   void WeightingField(const double x, const double y, const double z,
00047                       double& wx, double& wy, double& wz,
00048                       const std::string& label) override;
00049
00050   double WeightingPotential(const double x, const double y, const double z,
00051                             const std::string& label) override;
00052   /* FIX DOXYGEN ERROR ON PDF GENERATION
00053    * Deprecated version of the interface based on text file import of field
00054    * data.
00055    * \param elist Information about the element material of mesh cells. Each
00056    * line contains the element number and the material index:
00057    * \code
00058    * 0  3
00059    * ...
00060    * \endcode
00061    *
00062    * \param nlist Information about the mesh like this:
00063    * \code
00064    *  xmax 136 ymax 79 zmax 425
00065    *  x-lines
00066    * 0
00067    * 8.9285e-07
00068    * 1.7857e-06
00069    * ...
00070    * y-lines
00071    * 0
00072    * 8.9285e-07
00073    * 1.7857e-06
00074    * ...
00075    * z-lines
00076    * 0.0027
00077    * 0.00270674
00078    * ...
00079    * \endcode
00080    *
00081    * \param mplist Information about material properties used in the simulation:
00082    * \code
00083    *  Materials 4
00084    *  Material 1 PERX 1.0000000
```

```
00085  *  Material 2 RSVX 0 . 0 0 0 0 0 0 0 PERX 0 . 1 0 0 0 0 0 0 0E+11
00086  *  Material 3 PERX 3 . 5 0 0 0 0 0 0
00087  *  Material 4 PERX 4 . 8 0 0 0 0 0 0
00088  *  \endcode
00089  *
00090  * \param prnsol Information about the node potentials. Each line contains the
00091  * node number and the potential:
00092  *  \code
00093  *  0 1000.00
00094  *  ...
00095  *  \endcode
00096  * \param unit The units used in the nlist input file
00097  */
00098  bool Initialise(std::string elist, std::string nlist, std::string mplist,
00099                  std::string prnsol, std::string unit = "cm");
00109
00110  bool Initialise(std::string dataFile, std::string unit = "cm");
00123  bool SetWeightingField(std::string prnsol, std::string label,
00124                         bool isBinary = true);
00125
00126  // Range
00127
00128  void SetRangeZ(const double zmin, const double zmax);
00139  void DisableXField() { disableFieldComponent[0] = true; }
00140  void DisableYField() { disableFieldComponent[1] = true; }
00141  void DisableZField() { disableFieldComponent[2] = true; }
00185  void EnableShaping() { doShaping = true; }
00186  void DisableShaping() { doShaping = false; }
00192  int Index2Element(const unsigned int i, const unsigned int j,
00193                    const unsigned int k) const;
00201  bool Coordinate2Index(const double x, const double y, const double z,
00202                        unsigned int& i, unsigned int& j,
00203                        unsigned int& k) const;
00204
00205 protected:
00206  void SetRange() override;
00207
00208  double GetElementVolume(const size_t i) const override;
00209  void GetAspectRatio(const size_t i, double& dmin,
00210                      double& dmax) const override;
00211
00229  bool Coordinate2Index(const double x, const double y, const double z,
00230                        unsigned int& i, unsigned int& j, unsigned int& k,
00231                        double* position_mapped, bool* mirrored) const;
00232
00233 private:
00234  std::vector<double> m_xlines;
00235  std::vector<double> m_ylines;
00236  std::vector<double> m_zlines;
00238  std::vector<float> m_potential;
00240  std::map<std::string, std::vector<float> > m_weightingFields;
00242  std::vector<unsigned char> m_elementMaterial;
00243
00244  unsigned int m_nx = 0;
00245  unsigned int m_ny = 0;
00246  unsigned int m_nz = 0;
00247  size_t m_nElements = 0;
00248  size_t m_nNodes = 0;
00249  // If true x,y,z fields of this component are disabled (e=0 V/cm).
00250  bool disableFieldComponent[3] = {false, false, false};
00251  bool doShaping = false;
00252
00253  void ElectricFieldBinary(const double x, const double y, const double z,
00254                           double& ex, double& ey, double& ez, double& v,
00255                           Medium*& m, int& status,
00256                           const bool calculatePotential = false) const;
00257  float GetFieldComponent(const unsigned int i, const unsigned int j,
00258                          const unsigned int k, const double rx,
00259                          const double ry, const double rz,
00260                          const char component,
00261                          const std::vector<float>& potentials) const;
00262
00263  float GetPotential(const unsigned int i, const unsigned int j,
00264                     const unsigned int k, const double rx, const double ry,
00265                     const double rz,
00266                     const std::vector<float>& potentials) const;
00267
00268  void ShapeField(float& ex, float& ey, float& ez, const double rx,
00269                  const double ry, const double rz, const unsigned int i,
00270                  const unsigned int j, const unsigned int k,
00271                  const std::vector<float>& potentials) const;
00272
00273  /* Calculate the index (i,j,k) along x,y,z direction of the given element.
00274   * i,j,k start at 0 and reach at maximum
00275   * m_xlines-1,m_ylines-1,m_zlines-1
00276   */
00277  void Element2Index(const size_t element, unsigned int& i, unsigned int& j,
```

```
00278                   unsigned int& k) const;
00279
00280   int Index2Node(const unsigned int i, const unsigned int j,
00281                  const unsigned int k) const;
00282
00283   void Node2Index(const size_t node, unsigned int& i, unsigned int& j,
00284                   unsigned int& k) const;
00285 };
00286
00287 }  // namespace Garfield
00288
00289 #endif
```

## 8.23 /builds/garfield/docs/source/Include/Garfield/ComponentElmer.hh File Reference

#include <string>
#include "Garfield/ComponentFieldMap.hh"

Classes

- class Garfield::ComponentElmer

  Component for importing field maps computed by Elmer.

Namespaces

- namespace Garfield

## 8.24 ComponentElmer.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__ELMER__H
00002 #define G__COMPONENT__ELMER__H
00003
00004 #include <string>
00005
00006 #include "Garfield/ComponentFieldMap.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class ComponentElmer : public ComponentFieldMap {
00013  public:
00015   ComponentElmer();
00017   ComponentElmer(const std::string& header, const std::string& elist,
00018                  const std::string& nlist, const std::string& mplist,
00019                  const std::string& volt, const std::string& unit);
00021   ~ComponentElmer() {}
00022
00032   bool Initialise(const std::string& header = "mesh.header",
00033                   const std::string& elist = "mesh.elements",
00034                   const std::string& nlist = "mesh.nodes",
00035                   const std::string& mplist = "dielectrics.dat",
00036                   const std::string& volt = "out.result",
00037                   const std::string& unit = "cm");
00039   bool SetWeightingPotential(const std::string& prnsol,
00040                              const std::string& label) {
00041     return SetWeightingField(prnsol, label);
00042   }
00043   bool SetWeightingField(const std::string& prnsol, const std::string& label);
00044
00045 #ifdef USEGPU
00047   double CreateGPUTransferObject(ComponentGPU*& comp_gpu) override;
00048 #endif
00049  private:
00050   bool LoadPotentials(const std::string& prnsol, std::vector<double>& pot);
00051 };
00052 }  // namespace Garfield
00053 #endif
```

## 8.25 /builds/garfield/docs/source/Include/Garfield/ComponentElmer2d.hh File Reference

#include <string>
#include "Garfield/ComponentFieldMap.hh"

Classes

- class Garfield::ComponentElmer2d

  Component for importing two-dimensional field maps computed by Elmer.

Namespaces

- namespace Garfield

## 8.26 ComponentElmer2d.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__ELMER__2D__H
00002 #define G__COMPONENT__ELMER__2D__H
00003
00004 #include <string>
00005
00006 #include "Garfield/ComponentFieldMap.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class ComponentElmer2d : public ComponentFieldMap {
00013  public:
00015   ComponentElmer2d();
00017   ComponentElmer2d(const std::string& header, const std::string& elist,
00018                    const std::string& nlist, const std::string& mplist,
00019                    const std::string& volt, const std::string& unit);
00021   ~ComponentElmer2d() {}
00022
00032   bool Initialise(const std::string& header = "mesh.header",
00033                   const std::string& elist = "mesh.elements",
00034                   const std::string& nlist = "mesh.nodes",
00035                   const std::string& mplist = "dielectrics.dat",
00036                   const std::string& volt = "out.result",
00037                   const std::string& unit = "cm");
00039   bool SetWeightingPotential(const std::string& prnsol,
00040                              const std::string& label) {
00041     return SetWeightingField(prnsol, label);
00042   }
00043   bool SetWeightingField(const std::string& prnsol, const std::string& label);
00044
00045   void SetRangeZ(const double zmin, const double zmax);
00046
00047  private:
00048   bool LoadPotentials(const std::string& prnsol, std::vector<double>& pot);
00049 };
00050 }  // namespace Garfield
00051 #endif
```

## 8.27 /builds/garfield/docs/source/Include/Garfield/ComponentFieldMap.hh File Reference

#include <array>
#include <map>
#include <memory>
#include <string>
#include <vector>
#include "Garfield/Component.hh"
#include "Garfield/TetrahedralTree.hh"
#include "TMatrixD.h"
#include "TVectorD.h"

Classes

- class Garfield::ComponentFieldMap

  Base class for components based on finite-element field maps.
- struct Garfield::ComponentFieldMap::Element

- struct Garfield::ComponentFieldMap::Node
- struct Garfield::ComponentFieldMap::Material
- struct Garfield::ComponentFieldMap::WeightingFieldCopy

Namespaces

- namespace Garfield

## 8.28 ComponentFieldMap.hh

Go to the documentation of this file.

```
00001 #if defined(___GPUCOMPILE___) || !defined(G_COMPONENT_FIELD_MAP_H)
00002
00003 #if !defined(___GPUCOMPILE___) && !defined(G_COMPONENT_FIELD_MAP_H)
00004 #define G_COMPONENT_FIELD_MAP_H
00005 #endif
00006
00007 #ifdef ___GPUCOMPILE___
00008 #else
00009
00010 #include <array>
00011 #include <map>
00012 #include <memory>
00013 #include <string>
00014 #include <vector>
00015
00016 #include "Garfield/Component.hh"
00017 #include "Garfield/TetrahedralTree.hh"
00018 #include "TMatrixD.h"
00019 #include "TVectorD.h"
00020
00021 #endif
00022
00023 #ifndef ___GPUCOMPILE___
00024
00025 namespace Garfield {
00026
00028
00029 class ComponentFieldMap : public Component {
00030  public:
00032   ComponentFieldMap() = delete;
00034   ComponentFieldMap(const std::string& name);
00036   virtual ~ComponentFieldMap();
00037
00039   bool Check();
00040
00042   void PrintRange();
00043
00045   void PrintMaterials();
00047   void DriftMedium(const size_t imat);
00049   void NotDriftMedium(const size_t imat);
00051   size_t GetNumberOfMaterials() const { return m_materials.size(); }
00053   double GetPermittivity(const size_t imat) const;
00055   double GetConductivity(const size_t imat) const;
00057   void SetMedium(const size_t imat, Medium* medium);
00059   Medium* GetMedium(const size_t imat) const;
00060   using Component::GetMedium;
00063   void SetGas(Medium* medium);
00064
00065   size_t GetNumberOfElements() const override { return m_elements.size(); }
00066   bool GetElementNodes(const size_t i,
00067                        std::vector<size_t>& nodes) const override;
00068   bool GetElementRegion(const size_t i, size_t& mat,
00069                         bool& drift) const override;
00071   bool GetElement(const size_t i, double& vol, double& dmin,
00072                   double& dmax) const;
00073   size_t GetNumberOfNodes() const override { return m_nodes.size(); }
00074   bool GetNode(const size_t i, double& x, double& y, double& z) const override;
00076   double GetPotential(const size_t i) const;
00077
00078   // Options
00079   void EnableCheckMapIndices(const bool on = true) {
00080     m_checkMultipleElement = on;
00081   }
00083   void EnableDeleteBackgroundElements(const bool on = true) {
00084     m_deleteBackground = on;
00085   }
00086
00089   void EnableTetrahedralTreeForElementSearch(const bool on = true) {
00090     m_useTetrahedralTree = on;
00091   }
00092
```

```
00095   void EnableConvergenceWarnings(const bool on = true) {
00096     m_printConvergenceWarnings = on;
00097   }
00098
00099   Medium* GetMedium(const double x, const double y, const double z) override;
00100   void ElectricField(const double x, const double y, const double z, double& ex,
00101                      double& ey, double& ez, Medium*& m, int& status) override;
00102   void ElectricField(const double x, const double y, const double z, double& ex,
00103                      double& ey, double& ez, double& v, Medium*& m,
00104                      int& status) override;
00105   using Component::ElectricField;
00106 #endif
00107   ___DEVICE___
00108   void WeightingField(const double x, const double y, const double z,
00109                       double& wx, double& wy, double& wz,
00110 #ifdef ___GPUCOMPILE___
00111                       size_t label);
00112 #else
00113                     const std::string& label) override;
00114 double WeightingPotential(const double x, const double y, const double z,
00115                           const std::string& label) override;
00116
00117 double DelayedWeightingPotential(double x, double y, double z, const double t,
00118                                  const std::string& label) override;
00119 void DelayedWeightingPotentials(const double x, const double y, const double z,
00120                                 const std::string& label,
00121                                 std::vector<double>& dwp) override;
00122
00123 bool IsInBoundingBox(const double x, const double y, const double z) const {
00124   return x >= m_minBoundingBox[0] && x <= m_maxBoundingBox[0] &&
00125          y >= m_minBoundingBox[1] && y <= m_maxBoundingBox[1] &&
00126          z >= m_minBoundingBox[2] && y <= m_maxBoundingBox[2];
00127 }
00128
00129 bool Is3d() override { return m_is3d; }
00130 bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00131                     double& ymax, double& zmax) override;
00132 bool GetElementaryCell(double& xmin, double& ymin, double& zmin, double& xmax,
00133                        double& ymax, double& zmax) override;
00134
00135 std::map<std::string, std::vector<double> > GetWeightingPotentials() {
00136   return m_wpot;
00137 }
00138
00139 bool GetVoltageRange(double& vmin, double& vmax) override {
00140   vmin = m_mapvmin;
00141   vmax = m_mapvmax;
00142   return true;
00143 }
00144
00152 void CopyWeightingPotential(const std::string& label,
00153                             const std::string& labelSource, const double x,
00154                             const double y, const double z, const double alpha,
00155                             const double beta, const double gamma);
00156
00158 double CreateGPUTransferObject(ComponentGPU*& comp_gpu) override;
00159 #endif
00160  protected:
00161   bool m_is3d = true;
00162
00163   enum class ElementType {
00164     Unknown = 0,
00165     Serendipity = 5,
00166     CurvedTetrahedron = 13
00167   };
00168   ElementType m_elementType = ElementType::CurvedTetrahedron;
00169
00170   // Elements
00171   struct Element {
00172     // Nodes
00173     int emap[10];
00174     // Material
00175     unsigned int matmap;
00176   };
00177 #ifdef ___GPUCOMPILE___
00178   Element* m_elements = nullptr;
00179   int* m_elementIndices = nullptr;
00180   int numElements = 0;
00181 #else
00182 std::vector<Element> m_elements;
00183 std::vector<int> m_elementIndices;
00184 #endif
00185
00186 // Degeneracy flags.
00187 #ifdef ___GPUCOMPILE___
00188   bool* m_degenerate;
00189 #else
```

```
00190 std::vector<bool> m_degenerate;
00191 #endif
00192
00193 // Bounding boxes of the elements.
00194 #ifdef ___GPUCOMPILE___
00195   cuda_t** m_bbMin = nullptr;
00196   cuda_t** m_bbMax = nullptr;
00197 #else
00198 std::vector<std::array<double, 3> > m_bbMin;
00199 std::vector<std::array<double, 3> > m_bbMax;
00200 #endif
00201
00202 #ifdef ___GPUCOMPILE___
00203   cuda_t*** m_w12 = nullptr;
00204 #else
00205 std::vector<std::array<std::array<double, 3>, 4> > m_w12;
00206 #endif
00207
00208   // Nodes
00209   struct Node {
00210     // Coordinates
00211     double x, y, z;
00212   };
00213 #ifdef ___GPUCOMPILE___
00214   Node* m_nodes = nullptr;
00215   int numNodes = 0;
00216 #else
00217 std::vector<Node> m_nodes;
00218 #endif
00219
00220 // TODO GPU: m_dwpot not yet implemented
00221 #ifndef ___GPUCOMPILE___
00222   // Potentials.
00223   std::vector<double> m_pot;
00224   // Weighting potentials.
00225   std::map<std::string, std::vector<double> > m_wpot;
00226   // Delayed weighting potentials.
00227   std::map<std::string, std::vector<std::vector<double> > > m_dwpot;
00228 #else
00229 double* m_pot = nullptr;
00230 int m_numpot = 0;
00231
00232 // The number of weighting potentials
00233 int m_num_wpots = 0;
00234 // The number of entries in each weighting potential
00235 int* m_num_entries_wpot = nullptr;
00236 double** m_wpot = nullptr;
00237 #endif
00238
00239   // Materials
00240   struct Material {
00241     // Permittivity
00242     double eps;
00243     // Resistivity
00244     double ohm;
00245     bool driftmedium;
00246 #ifdef ___GPUCOMPILE___
00247     MediumGPU* medium;
00248 #else
00249   // Associated medium
00250   Medium* medium;
00251 #endif
00252   };
00253
00254 #ifdef ___GPUCOMPILE___
00255   Material* m_materials;
00256   int numMaterials;
00257 #else
00258 std::vector<Material> m_materials;
00259 #endif
00260
00261 #ifndef ___GPUCOMPILE___
00262   // Weighting potential copy
00263   struct WeightingFieldCopy {
00264     // Source
00265     std::string source;
00266     TMatrixD rot = TMatrixD(3, 3);
00267     TVectorD trans = TVectorD(3);
00268   };
00269
00270   // Weighting potential copies.
00271   std::map<std::string, WeightingFieldCopy> m_wfieldCopies;
00272
00273   // Bounding box
00274   bool m_hasBoundingBox = false;
00275   std::array<double, 3> m_minBoundingBox = {{0., 0., 0.}};
00276   std::array<double, 3> m_maxBoundingBox = {{0., 0., 0.}};
```

```
00277
00279   bool m_cacheElemBoundingBoxes = false;
00280 #endif
00281
00282 #ifdef ___GPUCOMPILE___
00283   double m_mapmin[3];
00284   double m_mapmax[3];
00285   double m_mapamin[3];
00286   double m_mapamax[3];
00287 #else
00288 // Ranges and periodicities
00289 std::array<double, 3> m_mapmin = {{0., 0., 0.}};
00290 std::array<double, 3> m_mapmax = {{0., 0., 0.}};
00291 std::array<double, 3> m_mapamin = {{0., 0., 0.}};
00292 std::array<double, 3> m_mapamax = {{0., 0., 0.}};
00293 #endif
00294
00295 #ifndef ___GPUCOMPILE___
00296   std::array<double, 3> m_mapna = {{0., 0., 0.}};
00297   std::array<double, 3> m_cells = {{0., 0., 0.}};
00298
00299   double m_mapvmin = 0.;
00300   double m_mapvmax = 0.;
00301
00302   std::array<bool, 3> m_setang;
00303
00304   // Option to delete meshing in conductors
00305   bool m_deleteBackground = true;
00306
00307   // Warnings flag
00308   bool m_warning = false;
00309   unsigned int m_nWarnings = 0;
00310
00311   // Print warnings about failed convergence when refining
00312   // isoparametric coordinates.
00313   bool m_printConvergenceWarnings = true;
00314
00315   // Get the scaling factor for a given length unit.
00316   static double ScalingFactor(std::string unit);
00317
00318   // Reset the component.
00319   void Reset() override;
00320
00321   void Prepare();
00322
00323   // Calculate x, y, z, V and angular ranges.
00324   virtual void SetRange();
00325
00326   // Periodicities
00327   void UpdatePeriodicity() override {
00328     if (!m_is3d) UpdatePeriodicity2d();
00329     UpdatePeriodicityCommon();
00330   }
00331   void UpdatePeriodicity2d();
00332   void UpdatePeriodicityCommon();
00333
00335   bool SetDefaultDriftMedium();
00336
00337 #endif
00339   ___DEVICE___
00340   int Field(const double x, const double y, const double z, double& fx,
00341             double& fy, double& fz, int& iel,
00342 #ifdef ___GPUCOMPILE___
00343             const double* potentials, const int numPotentials
00344 #else
00345           const std::vector<double>& potentials
00346 #endif
00347   ) const;
00348 #ifndef ___GPUCOMPILE___
00350   double Potential(const double x, const double y, const double z,
00351                    const std::vector<double>& potentials) const;
00353   static double Potential3(const std::array<double, 6>& v,
00354                            const std::array<double, 3>& t);
00356   static void Field3(const std::array<double, 6>& v,
00357                      const std::array<double, 3>& t, double jac[4][4],
00358                      const double det, double& ex, double& ey);
00360   static double Potential5(const std::array<double, 8>& v,
00361                            const std::array<double, 2>& t);
00363   static void Field5(const std::array<double, 8>& v,
00364                      const std::array<double, 2>& t, double jac[4][4],
00365                      const double det, double& ex, double& ey);
00367   static double Potential13(const std::array<double, 10>& v,
00368                             const std::array<double, 4>& t);
00369 #endif
00371   ___DEVICE___ static void Field13(
00372 #ifndef ___GPUCOMPILE___
00373       const std::array<double, 10>& v, const std::array<double, 4>& t,
```

```
00374 #else
00375     const double v[10], const double t[4],
00376 #endif
00377       double jac[4][4], const double det, double& ex, double& ey, double& ez);
00378 #ifndef ___GPUCOMPILE___
00380   int FindElement5(const double x, const double y, double& t1, double& t2,
00381                    double& t3, double& t4, double jac[4][4], double& det) const;
00382 #endif
00384   ___DEVICE___
00385   int FindElement13(const double x, const double y, const double z, double& t1,
00386                     double& t2, double& t3, double& t4, double jac[4][4],
00387                     double& det) const;
00388 #ifndef ___GPUCOMPILE___
00390   int FindElementCube(const double x, const double y, const double z,
00391                       double& t1, double& t2, double& t3, TMatrixD*& jac,
00392                       std::vector<TMatrixD*>& dN) const;
00393 #endif
00394
00396   ___DEVICE___
00397   void MapCoordinates(double& xpos, double& ypos, double& zpos, bool& xmirrored,
00398                       bool& ymirrored, bool& zmirrored, double& rcoordinate,
00399                       double& rotation) const;
00401   ___DEVICE___
00402   void UnmapFields(double& ex, double& ey, double& ez, const double xpos,
00403                    const double ypos, const double zpos, const bool xmirrored,
00404                    const bool ymirrored, const bool zmirrored,
00405                    const double rcoordinate, const double rotation) const;
00406
00407 #ifndef ___GPUCOMPILE___
00408   static int ReadInteger(char* token, int def, bool& error);
00409   static double ReadDouble(char* token, double def, bool& error);
00410
00411   virtual double GetElementVolume(const size_t i) const;
00412   virtual void GetAspectRatio(const size_t i, double& dmin, double& dmax) const;
00413
00414   void PrintWarning(const std::string& header);
00415   void PrintNotReady(const std::string& header) const;
00416   void PrintCouldNotOpen(const std::string& header,
00417                          const std::string& filename) const;
00418   void PrintElement(const std::string& header, const double x, const double y,
00419                     const double z, const double t1, const double t2,
00420                     const double t3, const double t4, const size_t i,
00421                     const std::vector<double>& potential) const;
00423   void TimeInterpolation(const double t, double& f0, double& f1, int& i0,
00424                          int& i1);
00425 #endif
00426
00427  protected:
00429   bool m_checkMultipleElement = false;
00430
00431   // Tetrahedral tree
00432   bool m_useTetrahedralTree = true;
00433 #ifdef ___GPUCOMPILE___
00434   GARFIELD_CLASS_NAME(TetrahedralTree) * m_octree = nullptr;
00435 #else
00436 std::unique_ptr<GARFIELD_CLASS_NAME(TetrahedralTree)> m_octree;
00437 #endif
00438
00439  protected:
00440 #ifndef ___GPUCOMPILE___
00442   int Coordinates3(const double x, const double y, double& t1, double& t2,
00443                    double& t3, double& t4, double jac[4][4], double& det,
00444                    const std::array<double, 8>& xn,
00445                    const std::array<double, 8>& yn) const;
00447   int Coordinates4(const double x, const double y, double& t1, double& t2,
00448                    double& t3, double& t4, double& det,
00449                    const std::array<double, 8>& xn,
00450                    const std::array<double, 8>& yn) const;
00452   int Coordinates5(const double x, const double y, double& t1, double& t2,
00453                    double& t3, double& t4, double jac[4][4], double& det,
00454                    const std::array<double, 8>& xn,
00455                    const std::array<double, 8>& yn) const;
00456 #endif
00457
00459   ___DEVICE___
00460   void Coordinates12(const double x, const double y, const double z, double& t1,
00461                      double& t2, double& t3, double& t4,
00462 #ifdef ___GPUCOMPILE___
00463                      const double xn[10], const double yn[10],
00464                      const double zn[10], const double w[4][3]
00465 #else
00466                      const std::array<double, 10>& xn,
00467                      const std::array<double, 10>& yn,
00468                      const std::array<double, 10>& zn,
00469                      const std::array<std::array<double, 3>, 4>& w
00470 #endif
00471   ) const;
```

```
00472
00474   ___DEVICE___
00475   int Coordinates13(const double x, const double y, const double z, double& t1,
00476                     double& t2, double& t3, double& t4, double jac[4][4],
00477                     double& det,
00478 #ifdef ___GPUCOMPILE___
00479                     const double xn[10], const double yn[10],
00480                     const double zn[10], cuda_t** w
00481 #else
00482                   const std::array<double, 10>& xn,
00483                   const std::array<double, 10>& yn,
00484                   const std::array<double, 10>& zn,
00485                   const std::array<std::array<double, 3>, 4>& w
00486 #endif
00487   ) const;
00488
00489 #ifndef ___GPUCOMPILE___
00491   int CoordinatesCube(const double x, const double y, const double z,
00492                       double& t1, double& t2, double& t3, TMatrixD*& jac,
00493                       std::vector<TMatrixD*>& dN, const Element& element) const;
00494
00496   static void Jacobian3(const std::array<double, 8>& xn,
00497                         const std::array<double, 8>& yn, const double u,
00498                         const double v, const double w, double& det,
00499                         double jac[4][4]);
00501   static void Jacobian5(const std::array<double, 8>& xn,
00502                         const std::array<double, 8>& yn, const double u,
00503                         const double v, double& det, double jac[4][4]);
00504 #endif
00505
00507   ___DEVICE___
00508   static void Jacobian13(
00509 #ifdef ___GPUCOMPILE___
00510       const double xn[10], const double yn[10], const double zn[10],
00511 #else
00512     const std::array<double, 10>& xn, const std::array<double, 10>& yn,
00513     const std::array<double, 10>& zn,
00514 #endif
00515       const double fourt0, const double fourt1, const double fourt2,
00516       const double fourt3, double& det, double jac[4][4]);
00517
00518 #ifndef ___GPUCOMPILE___
00520   void JacobianCube(const Element& element, const double t1, const double t2,
00521                     const double t3, TMatrixD*& jac,
00522                     std::vector<TMatrixD*>& dN) const;
00523
00524   static std::array<std::array<double, 3>, 4> Weights12(
00525       const std::array<double, 10>& xn, const std::array<double, 10>& yn,
00526       const std::array<double, 10>& zn);
00527
00529   void CalculateElementBoundingBoxes();
00530
00532   bool InitializeTetrahedralTree();
00533 };
00534 }  // namespace Garfield
00535
00536 #endif
00537 #endif
```

## 8.29 /builds/garfield/docs/source/Include/Garfield/ComponentGrid.hh File Reference

#include <array>
#include <string>
#include <vector>
#include "Garfield/Component.hh"

### Classes

- class Garfield::ComponentGrid

  Component for interpolating field maps on a regular mesh.
- struct Garfield::ComponentGrid::Node

### Namespaces

- namespace Garfield

## 8.30 ComponentGrid.hh

Go to the documentation of this file.

```
00001 #ifndef G_COMPONENT_GRID_H
00002 #define G_COMPONENT_GRID_H
00003
00004 #include <array>
00005 #include <string>
00006 #include <vector>
00007
00008 #include "Garfield/Component.hh"
00009
00010 namespace Garfield {
00011
00013
00014 class ComponentGrid : public Component {
00015  public:
00017   ComponentGrid();
00019   ~ComponentGrid() {}
00020
00027   bool SetMesh(const unsigned int nx, const unsigned int ny,
00028                const unsigned int nz, const double xmin, const double xmax,
00029                const double ymin, const double ymax, const double zmin,
00030                const double zmax);
00032   bool GetMesh(unsigned int& nx, unsigned int& ny, unsigned int& nz,
00033                double& xmin, double& xmax, double& ymin, double& ymax,
00034                double& zmin, double& zmax) const;
00036   void SetCartesianCoordinates() { m_coordinates = Coordinates::Cartesian; }
00038   void SetCylindricalCoordinates();
00039
00061   bool LoadElectricField(const std::string& filename, const std::string& format,
00062                          const bool withPotential, const bool withFlag,
00063                          const double scaleX = 1., const double scaleE = 1.,
00064                          const double scaleP = 1.);
00065
00067   bool LoadWeightingField(const std::string& filename,
00068                           const std::string& format, const bool withPotential,
00069                           const double scaleX = 1., const double scaleE = 1.,
00070                           const double scaleP = 1.);
00072   bool LoadWeightingField(const std::string& filename,
00073                           const std::string& format, const double time,
00074                           const bool withPotential, const double scaleX = 1.,
00075                           const double scaleE = 1., const double scaleP = 1.);
00079   void SetWeightingFieldOffset(const double x, const double y, const double z);
00080
00082   bool LoadMagneticField(const std::string& filename, const std::string& format,
00083                          const double scaleX = 1., const double scaleB = 1.);
00084
00090   bool SaveElectricField(Component* cmp, const std::string& filename,
00091                          const std::string& fmt);
00098   bool SaveWeightingField(Component* cmp, const std::string& id,
00099                           const std::string& filename, const std::string& fmt);
00100
00102   bool GetElectricField(const unsigned int i, const unsigned int j,
00103                         const unsigned int k, double& v, double& ex, double& ey,
00104                         double& ez) const;
00105
00107   void SetMedium(Medium* m);
00109   Medium* GetMedium() const { return m_medium; }
00110
00112   void Print();
00113
00120   bool LoadElectronAttachment(const std::string& fname, const std::string& fmt,
00121                               const unsigned int col, const double scaleX = 1.);
00123   bool LoadHoleAttachment(const std::string& fname, const std::string& fmt,
00124                           const unsigned int col, const double scaleX = 1.);
00125
00132   bool LoadElectronVelocity(const std::string& fname, const std::string& fmt,
00133                             const double scaleX = 1.,
00134                             const double scaleV = 1.e-9);
00136   bool LoadHoleVelocity(const std::string& fname, const std::string& fmt,
00137                         const double scaleX = 1., const double scaleV = 1.e-9);
00138
00139   void Clear() override { Reset(); }
00140   void ElectricField(const double x, const double y, const double z, double& ex,
00141                      double& ey, double& ez, double& v, Medium*& m,
00142                      int& status) override;
00143   void ElectricField(const double x, const double y, const double z, double& ex,
00144                      double& ey, double& ez, Medium*& m, int& status) override;
00145   using Component::ElectricField;
00146   void WeightingField(const double x, const double y, const double z,
00147                       double& wx, double& wy, double& wz,
00148                       const std::string& label) override;
00149   double WeightingPotential(const double x, const double y, const double z,
00150                             const std::string& label) override;
00151   void DelayedWeightingField(const double x, const double y, const double z,
```

```
00152                           const double t, double& wx, double& wy, double& wz,
00153                           const std::string& label) override;
00154   double DelayedWeightingPotential(const double x, const double y,
00155                                    const double z, const double t,
00156                                    const std::string& label) override;
00157   void MagneticField(const double x, const double y, const double z, double& bx,
00158                      double& by, double& bz, int& status) override;
00159
00160   Medium* GetMedium(const double x, const double y, const double z) override;
00161
00162   bool GetVoltageRange(double& vmin, double& vmax) override;
00163   bool GetElectricFieldRange(double& exmin, double& exmax, double& eymin,
00164                              double& eymax, double& ezmin, double& ezmax);
00165   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00166                       double& ymax, double& zmax) override;
00167   bool GetElementaryCell(double& xmin, double& ymin, double& zmin, double& xmax,
00168                          double& ymax, double& zmax) override;
00169
00170   bool HasMagneticField() const override;
00171
00172   bool HasAttachmentMap() const override {
00173     return !(m_eAttachment.empty() && m_hAttachment.empty());
00174   }
00175   bool ElectronAttachment(const double x, const double y, const double z,
00176                           double& att) override;
00177   bool HoleAttachment(const double x, const double y, const double z,
00178                       double& att) override;
00179   bool HasMobilityMap() const override {
00180     return !(m_eMobility.empty() && m_hMobility.empty());
00181   }
00182   bool ElectronMobility(const double x, const double y, const double z,
00183                         double& mu) override;
00184   bool HoleMobility(const double x, const double y, const double z,
00185                     double& mu) override;
00186
00187   bool HasVelocityMap() const override {
00188     return !(m_eVelocity.empty() && m_hVelocity.empty());
00189   }
00190   bool ElectronVelocity(const double x, const double y, const double z,
00191                         double& vx, double& vy, double& vz) override;
00192   bool HoleVelocity(const double x, const double y, const double z, double& vx,
00193                     double& vy, double& vz) override;
00194
00195  private:
00196   enum class Format { Unknown, XY, XZ, XYZ, IJ, IK, IJK, YXZ };
00197   enum class Coordinates { Cartesian, Cylindrical };
00198   Coordinates m_coordinates = Coordinates::Cartesian;
00199
00200   Medium* m_medium = nullptr;
00201   struct Node {
00202     double fx, fy, fz;
00203     double v;
00204   };
00205
00207   std::vector<std::vector<std::vector<Node> > > m_efields;
00209   std::vector<std::vector<std::vector<Node> > > m_bfields;
00211   std::vector<std::vector<std::vector<Node> > > m_wfields;
00213   std::vector<std::vector<std::vector<std::vector<Node> > > > m_wdfields;
00215   std::vector<std::vector<std::vector<double> > > m_eAttachment;
00216   std::vector<std::vector<std::vector<double> > > m_hAttachment;
00218   std::vector<std::vector<std::vector<double> > > m_eMobility;
00219   std::vector<std::vector<std::vector<double> > > m_hMobility;
00221   std::vector<std::vector<std::vector<Node> > > m_eVelocity;
00222   std::vector<std::vector<std::vector<Node> > > m_hVelocity;
00224   std::vector<std::vector<std::vector<bool> > > m_active;
00225
00226   // Dimensions of the mesh
00227   std::array<unsigned int, 3> m_nX = {{1, 1, 1}};
00228   std::array<double, 3> m_xMin = {{0., 0., 0.}};
00229   std::array<double, 3> m_xMax = {{0., 0., 0.}};
00230   std::array<double, 3> m_sX = {{0., 0., 0.}};
00231
00232   bool m_hasMesh = false;
00233   bool m_hasPotential = false;
00234
00235   // Offset for weighting field
00236   std::array<double, 3> m_wFieldOffset = {{0., 0., 0.}};
00237
00238   // Voltage range
00239   double m_pMin = 0., m_pMax = 0.;
00240
00242   bool LoadMesh(const std::string& filename, std::string format,
00243                 const double scaleX);
00244
00246   bool LoadData(const std::string& filename, std::string format,
00247                 const bool withPotential, const bool withFlag,
00248                 const double scaleX, const double scaleF, const double scaleP,
```

```
00249               std::vector<std::vector<std::vector<Node> > >& field);
00251   bool LoadData(const std::string& filename, std::string format,
00252               const double scaleX,
00253               std::vector<std::vector<std::vector<double> > >& tab,
00254               const unsigned int col);
00255
00256   void Reset() override;
00257   void UpdatePeriodicity() override;
00258
00260   bool GetField(const double x, const double y, const double z,
00261               const std::vector<std::vector<std::vector<Node> > >& field,
00262               double& fx, double& fy, double& fz, double& p, bool& active);
00264   bool GetData(const double x, const double y, const double z,
00265              const std::vector<std::vector<std::vector<double> > >& table,
00266              double& value);
00267
00269   double Reduce(const double xin, const double xmin, const double xmax,
00270               const bool simplePeriodic, const bool mirrorPeriodic,
00271               bool& isMirrored) const;
00273   void Initialise(std::vector<std::vector<std::vector<Node> > >& fields);
00275   Format GetFormat(std::string fmt);
00276 };
00277 }  // namespace Garfield
00278 #endif
```

## 8.31 /builds/garfield/docs/source/Include/Garfield/ComponentNeBem2d.hh File Reference

#include <array>
#include "Garfield/Component.hh"

### Classes

- class Garfield::ComponentNeBem2d

  Two-dimensional implementation of the nearly exact Boundary Element Method.
- struct Garfield::ComponentNeBem2d::Region
- struct Garfield::ComponentNeBem2d::Segment
- struct Garfield::ComponentNeBem2d::Wire
- struct Garfield::ComponentNeBem2d::Element
- struct Garfield::ComponentNeBem2d::SpaceCharge

### Namespaces

- namespace Garfield

## 8.32 ComponentNeBem2d.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__NEBEM__2D__H
00002 #define G__COMPONENT__NEBEM__2D__H
00003
00004 #include <array>
00005
00006 #include "Garfield/Component.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class ComponentNeBem2d : public Component {
00013  public:
00015   ComponentNeBem2d();
00017   ~ComponentNeBem2d() {}
00018
00020   void SetMedium(Medium* medium) { m__medium = medium; }
00021
00027   bool AddSegment(const double x0, const double y0, const double x1,
00028                 const double y1, const double v, const int ndiv = -1);
00036   bool AddWire(const double x, const double y, const double d, const double v,
00037              const int ntrap = 5);
00045   bool AddRegion(const std::vector<double>& xp, const std::vector<double>& yp,
00046                Medium* medium, const unsigned int bctype = 4,
00047                const double v = 0., const int ndiv = -1);
00048   void AddChargeDistribution(const double x, const double y, const double a,
```

```
00049                         const double b, const double rho);
00051   void SetRangeZ(const double zmin, const double zmax);
00052
00054   bool Initialise();
00055
00057   void SetNumberOfDivisions(const unsigned int ndiv);
00058
00059   void SetNumberOfCollocationPoints(const unsigned int ncoll);
00060   void EnableAutoResizing(const bool on = true) { m_autoSize = on; }
00061   void EnableRandomCollocation(const bool on = true) {
00062     m_randomCollocation = on;
00063   }
00064   void SetMaxNumberOfIterations(const unsigned int niter);
00065
00067   unsigned int GetNumberOfRegions() const { return m_regions.size(); }
00069   bool GetRegion(const unsigned int i, std::vector<double>& xv,
00070                  std::vector<double>& yv, Medium*& medium, unsigned int& bctype,
00071                  double& v);
00073   unsigned int GetNumberOfSegments() const { return m_segments.size(); }
00075   bool GetSegment(const unsigned int i, double& x0, double& y0, double& x1,
00076                   double& x2, double& v) const;
00078   unsigned int GetNumberOfWires() const { return m_wires.size(); }
00080   bool GetWire(const unsigned int i, double& x, double& y, double& d, double& v,
00081                double& q) const;
00083   size_t GetNumberOfElements() const override { return m_elements.size(); }
00085   bool GetElement(const unsigned int i, double& x0, double& y0, double& x1,
00086                   double& y1, double& q) const;
00087
00088   Medium* GetMedium(const double x, const double y, const double z) override;
00089
00090   void ElectricField(const double x, const double y, const double z, double& ex,
00091                      double& ey, double& ez, Medium*& m, int& status) override;
00092   void ElectricField(const double x, const double y, const double z, double& ex,
00093                      double& ey, double& ez, double& v, Medium*& m,
00094                      int& status) override;
00095   using Component::ElectricField;
00096   bool GetVoltageRange(double& vmin, double& vmax) override;
00097
00098   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00099                       double& ymax, double& zmax) override;
00100   bool GetElementaryCell(double& xmin, double& ymin, double& zmin, double& xmax,
00101                          double& ymax, double& zmax) override;
00102
00103   bool CrossedWire(const double x0, const double y0, const double z0,
00104                    const double x1, const double y1, const double z1,
00105                    double& xc, double& yc, double& zc, const bool centre,
00106                    double& rc) override;
00107   bool InTrapRadius(const double q0, const double x0, const double y0,
00108                     const double z0, double& xw, double& yx,
00109                     double& rw) override;
00110
00111 private:
00112   static const double InvEpsilon0;
00113   static const double InvTwoPiEpsilon0;
00114
00116   unsigned int m_nDivisions = 5;
00117   unsigned int m_nCollocationPoints = 1;
00118   bool m_autoSize = false;
00119   bool m_randomCollocation = false;
00120   unsigned int m_nMaxIterations = 3;
00121
00123   Medium* m_medium = nullptr;
00125   bool m_useRangeZ = false;
00127   double m_zmin = -1.;
00129   double m_zmax = 1.;
00131   enum BC {
00132     Voltage = 1,
00133     Charge,
00134     Floating,
00135     Dielectric
00136   };
00137   struct Region {
00138     std::vector<double> xv;
00139     std::vector<double> yv;
00140     Medium* medium;
00141     std::pair<BC, double> bc;
00142     unsigned int depth;
00143     int ndiv;
00144   };
00146   std::vector<Region> m_regions;
00147
00148   struct Segment {
00149     std::array<double, 2> x0;
00150     std::array<double, 2> x1;
00151     int region1;
00152     int region2;
00153     std::pair<BC, double> bc;
```

```
00154     int ndiv;
00155   };
00157   std::vector<Segment> m_segments;
00158
00159   struct Wire {
00160     double x, y;
00161     double r;
00162     double v;
00163     double q;
00164     int ntrap;
00165   };
00167   std::vector<Wire> m_wires;
00168
00169   struct Element {
00170     double x, y;
00171     double a;
00172     double cphi;
00173     double sphi;
00174     double q;
00175     std::pair<BC, double> bc;
00176     double lambda;
00177   };
00179   std::vector<Element> m_elements;
00180
00181   struct SpaceCharge {
00182     double x, y;
00183     double a, b;
00184     double q;
00185     double v0;
00186   };
00187   std::vector<SpaceCharge> m_spaceCharge;
00188
00190   void EliminateOverlaps(std::vector<Segment>& segments);
00192   bool Discretise(const Segment& segment, std::vector<Element>& elements,
00193                   const double lambda, const unsigned int ndiv);
00194
00195   bool ComputeInfluenceMatrix(std::vector<std::vector<double> >& infmat) const;
00196   bool InvertMatrix(std::vector<std::vector<double> >& influenceMatrix,
00197                     std::vector<std::vector<double> >& inverseMatrix) const;
00198   bool LUDecomposition(std::vector<std::vector<double> >& mat,
00199                        std::vector<int>& index) const;
00200   void LUSubstitution(const std::vector<std::vector<double> >& mat,
00201                       const std::vector<int>& index,
00202                       std::vector<double>& col) const;
00203
00204   bool Solve(const std::vector<std::vector<double> >& inverseMatrix,
00205              const std::vector<double>& bc);
00206   bool CheckConvergence(const double tol, std::vector<bool>& ok);
00207   void SplitElement(Element& oldElement, std::vector<Element>& elements);
00208
00210   double LinePotential(const double a, const double x, const double y) const;
00212   double WirePotential(const double r0, const double x, const double y) const;
00214   void LineField(const double a, const double x, const double y, double& ex,
00215                  double& ey) const;
00217   void WireField(const double r0, const double x, const double y, double& ex,
00218                  double& ey) const;
00219
00221   double BoxPotential(const double a, const double b, const double x,
00222                       const double y, const double v0) const;
00224   void BoxField(const double a, const double b, const double x, const double y,
00225                 double& ex, double& ey) const;
00226
00227   void Reset() override;
00228   void UpdatePeriodicity() override;
00230   void ToLocal(const double xIn, const double yIn, const double cphi,
00231                const double sphi, double& xOut, double& yOut) const;
00233   void ToGlobal(const double xIn, const double yIn, const double cphi,
00234                 const double sphi, double& xOut, double& yOut) const;
00235
00237   int Field(const double x, const double y, const double z, double& ex,
00238             double& ey, double& ez, double& v, Medium*& m, const bool opt);
00239 };
00240 }  // namespace Garfield
00241
00242 #endif
```

## 8.33 /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3d.hh File Reference

#include <array>
#include <map>
#include "Garfield/Component.hh"
#include "Garfield/Solid.hh"

Classes

- class Garfield::ComponentNeBem3d

  Interface to neBEM.
- struct Garfield::ComponentNeBem3d::Primitive
- struct Garfield::ComponentNeBem3d::Element

Namespaces

- namespace Garfield

Variables

- ComponentNeBem3d * Garfield::gComponentNeBem3d

## 8.34 ComponentNeBem3d.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__NEBEM__3D__H
00002 #define G__COMPONENT__NEBEM__3D__H
00003
00004 #include <array>
00005 #include <map>
00006
00007 #include "Garfield/Component.hh"
00008 #include "Garfield/Solid.hh"
00009
00010 namespace Garfield {
00011
00013
00014 class ComponentNeBem3d : public Component {
00015  public:
00017   ComponentNeBem3d();
00019   ~ComponentNeBem3d() {}
00020
00021   Medium* GetMedium(const double x, const double y, const double z) override;
00022
00024   void AddPlaneX(const double x, const double voltage);
00026   void AddPlaneY(const double y, const double voltage);
00028   void AddPlaneZ(const double z, const double voltage);
00030   unsigned int GetNumberOfPlanesX() const;
00032   unsigned int GetNumberOfPlanesY() const;
00034   unsigned int GetNumberOfPlanesZ() const;
00036   bool GetPlaneX(const unsigned int i, double& x, double& v) const;
00038   bool GetPlaneY(const unsigned int i, double& y, double& v) const;
00040   bool GetPlaneZ(const unsigned int i, double& z, double& v) const;
00041
00042   unsigned int GetNumberOfPrimitives() const { return m_primitives.size(); }
00043   bool GetPrimitive(const unsigned int i, double& a, double& b, double& c,
00044                     std::vector<double>& xv, std::vector<double>& yv,
00045                     std::vector<double>& zv, int& interface, double& v,
00046                     double& q, double& lambda) const;
00047   bool GetPrimitive(const unsigned int i, double& a, double& b, double& c,
00048                     std::vector<double>& xv, std::vector<double>& yv,
00049                     std::vector<double>& zv, int& vol1, int& vol2) const;
00050   bool GetVolume(const unsigned int vol, int& shape, int& material, double& eps,
00051                  double& potential, double& charge, int& bc);
00052   int GetVolume(const double x, const double y, const double z);
00053
00054   size_t GetNumberOfElements() const override { return m_elements.size(); }
00055   bool GetElement(const unsigned int i, std::vector<double>& xv,
00056                   std::vector<double>& yv, std::vector<double>& zv,
00057                   int& interface, double& bc, double& lambda) const;
00058
00061   bool Initialise();
00062
00065   void SetTargetElementSize(const double length);
00068   void SetMinMaxNumberOfElements(const unsigned int nmin,
00069                                  const unsigned int nmax);
00070
00071   void SetNewModel(const unsigned int NewModel);
00072   void SetNewMesh(const unsigned int NewMesh);
00073   void SetNewBC(const unsigned int NewBC);
00074   void SetNewPP(const unsigned int NewPP);
00075   void SetModelOptions(const unsigned int NewModel, const unsigned int NewMesh,
00076                        const unsigned int NewBC, const unsigned int NewPP);
```

```
00077
00083   void SetStoreInflMatrix(const unsigned int OptStoreInflMatrix);
00084   void SetReadInflMatrix(const unsigned int OptReadInflMatrix);
00085   void SetStoreInvMatrix(const unsigned int OptStoreInvMatrix);
00086   void SetReadInvMatrix(const unsigned int OptReadInvMatrix);
00087   void SetStorePrimitives(const unsigned int OptStorePrimitives);
00088   void SetReadPrimitives(const unsigned int OptReadPrimitives);
00089   void SetStoreElements(const unsigned int OptStoreElements);
00090   void SetReadElements(const unsigned int OptReadElements);
00091   void SetFormattedFile(const unsigned int OptFormattedFile);
00092   void SetUnformattedFile(const unsigned int OptUnformattedFile);
00093   void SetStoreReadOptions(const unsigned int OptStoreInflMatrix,
00094                            const unsigned int OptReadInflMatrix,
00095                            const unsigned int OptStoreInvMatrix,
00096                            const unsigned int OptReadInvMatrix,
00097                            const unsigned int OptStorePrimitives,
00098                            const unsigned int OptReadPrimitives,
00099                            const unsigned int OptStoreElements,
00100                            const unsigned int OptReadElements,
00101                            const unsigned int OptFormattedFile,
00102                            const unsigned int OptUnformattedFile);
00103
00104   // Set whether an older model is to be re-used. Expects an inverted matrix
00105   // stored during an earlier computation that had identical model and mesh.
00106   // This execution considers only a change in the boundary conditions.
00107   void SetReuseModel(void);
00108
00113
00114   // Functions that set computation details and constraints
00115   void SetSystemChargeZero(const unsigned int OptSystemChargeZero);
00116   void SetValidateSolution(const unsigned int OptValidateSolution);
00117   void SetForceValidation(const unsigned int OptForceValidation);
00118   void SetRepeatLHMatrix(const unsigned int OptRepeatLHMatrix);
00119   void SetComputeOptions(const unsigned int OptSystemChargeZero,
00120                          const unsigned int OptValidateSolution,
00121                          const unsigned int OptForceValidation,
00122                          const unsigned int OptRepeatLHMatrix);
00123
00124   // Fast volume related information (physical potential and fields)
00125   void SetFastVolOptions(const unsigned int OptFastVol,
00126                          const unsigned int OptCreateFastPF,
00127                          const unsigned int OptReadFastPF);
00128   void SetFastVolVersion(const unsigned int VersionFV);
00129   void SetFastVolBlocks(const unsigned int NbBlocksFV);
00130
00131   // Needs to include IdWtField information for each of these WtFld functions
00132   // Weighting potential and field related Fast volume information
00133   void SetWtFldFastVolOptions(const unsigned int IdWtField,
00134                               const unsigned int OptWtFldFastVol,
00135                               const unsigned int OptCreateWtFldFastPF,
00136                               const unsigned int OptReadWtFldFastPF);
00137   void SetWtFldFastVolVersion(const unsigned int IdWtField,
00138                               const unsigned int VersionWtFldFV);
00139   void SetWtFldFastVolBlocks(const unsigned int IdWtField,
00140                              const unsigned int NbBlocksWtFldFV);
00141
00142   // Known charge options
00143   void SetKnownChargeOptions(const unsigned int OptKnownCharge);
00144
00145   // Charging up options
00146   void SetChargingUpOptions(const unsigned int OptChargingUp);
00147
00149   void UseLUInversion() { m_inversion = Inversion::LU; }
00151   void UseSVDInversion() { m_inversion = Inversion::SVD; }
00152
00156   void SetPeriodicCopies(const unsigned int nx, const unsigned int ny,
00157                          const unsigned int nz);
00159   void GetPeriodicCopies(unsigned int& nx, unsigned int& ny,
00160                          unsigned int& nz) const {
00161     nx = m_nCopiesX;
00162     ny = m_nCopiesY;
00163     nz = m_nCopiesZ;
00164   }
00166   void SetPeriodicityX(const double s);
00168   void SetPeriodicityY(const double s);
00170   void SetPeriodicityZ(const double s);
00172   void SetMirrorPeriodicityX(const double s);
00174   void SetMirrorPeriodicityY(const double s);
00176   void SetMirrorPeriodicityZ(const double s);
00178   bool GetPeriodicityX(double& s) const;
00180   bool GetPeriodicityY(double& s) const;
00182   bool GetPeriodicityZ(double& s) const;
00183
00185   void SetNumberOfThreads(const unsigned int n) { m_nThreads = n > 0 ? n : 1; }
00186
00190   void SetPrimAfter(const int n) { m_primAfter = n; }
00191
```

```
00195   void SetWtFldPrimAfter(const int n) { m_wtFldPrimAfter = n; }
00196
00198   void SetOptRmPrim(const unsigned int n) { m_optRmPrim = n; }
00199
00200   void ElectricField(const double x, const double y, const double z, double& ex,
00201                      double& ey, double& ez, Medium*& m, int& status) override;
00202   void ElectricField(const double x, const double y, const double z, double& ex,
00203                      double& ey, double& ez, double& v, Medium*& m,
00204                      int& status) override;
00205   using Component::ElectricField;
00206   bool GetVoltageRange(double& vmin, double& vmax) override;
00207
00208   void WeightingField(const double x, const double y, const double z,
00209                       double& wx, double& wy, double& wz,
00210                       const std::string& label) override;
00211   double WeightingPotential(const double x, const double y, const double z,
00212                             const std::string& label) override;
00213
00214 protected:
00215   void Reset() override;
00216   void UpdatePeriodicity() override;
00217
00218 private:
00219   struct Primitive {
00221     double a, b, c;
00223     std::vector<double> xv;
00225     std::vector<double> yv;
00227     std::vector<double> zv;
00229     int interface = 0;
00231     double v = 0.;
00233     double q = 0.;
00235     double lambda = 0.;
00237     double elementSize;
00239     int vol1, vol2;
00240   };
00242   std::vector<Primitive> m_primitives;
00243
00244   struct Element {
00246     std::array<double, 3> origin;
00247     double lx;
00248     double lz;
00250     double dA;
00252     std::array<std::array<double, 3>, 3> dcos;
00254     std::vector<double> xv;
00256     std::vector<double> yv;
00258     std::vector<double> zv;
00260     int interface = 0;
00262     double lambda = 0.;
00264     std::array<double, 3> collocationPoint;
00266     double bc = 0.;
00268     double assigned = 0.;
00270     double solution = 0.;
00271   };
00273   std::vector<Element> m_elements;
00274
00276   std::array<bool, 6> m_ynplan{{false, false, false, false, false, false}};
00278   std::array<double, 6> m_coplan{{0., 0., 0., 0., 0., 0.}};
00280   std::array<double, 6> m_vtplan{{0., 0., 0., 0., 0., 0.}};
00281
00282   // Model specifications
00283   unsigned int m_newModel = 1;
00284   unsigned int m_newMesh = 1;
00285   unsigned int m_newBC = 1;
00286   unsigned int m_newPP = 1;
00287
00288   // Store and read options
00289   unsigned int m_optStoreInflMatrix = 0;
00290   unsigned int m_optReadInflMatrix = 0;
00291   unsigned int m_optStoreInvMatrix = 1;
00292   unsigned int m_optReadInvMatrix = 0;
00293   unsigned int m_optStorePrimitives = 0;
00294   unsigned int m_optReadPrimitives = 0;
00295   unsigned int m_optStoreElements = 0;
00296   unsigned int m_optReadElements = 0;
00297   unsigned int m_optStoreFormatted = 1;
00298   unsigned int m_optStoreUnformatted = 0;
00299
00300   // Plot options
00301   // unsigned int m_optGnuplotPrimitives = 0;
00302   // unsigned int m_optGnuplotElements = 0;
00303   // unsigned int m_optPrimitiveFiles = 0;
00304   // unsigned int m_optElementFiles = 0;
00305
00306   // Compute options
00307   unsigned int m_optSystemChargeZero = 1;
00308   unsigned int m_optValidateSolution = 1;
00309   unsigned int m_optForceValidation = 0;
```

```
00310   unsigned int m_optRepeatLHMatrix = 0;
00311
00312   // Fast volume information (physical potential and fields)
00313   unsigned int m_optFastVol = 0;
00314   unsigned int m_optCreateFastPF = 0;
00315   unsigned int m_optReadFastPF = 0;
00316   unsigned int m_versionFV = 0;
00317   unsigned int m_nbBlocksFV = 0;
00318
00319   // Weighting potential and field related Fast volume information
00320   unsigned int m_idWtField = 0;
00321   unsigned int m_optWtFldFastVol[11];
00322   unsigned int m_optCreateWtFldFastPF[11];
00323   unsigned int m_optReadWtFldFastPF[11];
00324   unsigned int m_versionWtFldFV[11];
00325   unsigned int m_nbBlocksWtFldFV[11];
00326
00327   // Known charge options
00328   unsigned int m_optKnownCharge = 0;
00329
00330   // Charging up options
00331   unsigned int m_optChargingUp = 0;
00332
00333   // Number of threads to be used by neBEM.
00334   unsigned int m_nThreads = 1;
00335
00336   // Number of repetitions, after which only primitive properties are used.
00337   // a negative value implies elements are used always.
00338   int m_primAfter = -1;
00339
00340   // Number of repetitions, after which only primitive properties are used
00341   // for weighting field calculations.
00342   // A negative value implies only elements are used.
00343   int m_wtFldPrimAfter = -1;
00344
00345   // Option for removing primitives from a device geometry.
00346   // Zero implies none to be removed.
00347   unsigned int m_optRmPrim = 0;
00348
00349   static constexpr double MinDist = 1.e-6;
00351   double m_targetElementSize = 50.0e-4;
00353   unsigned int m_minNbElementsOnLength = 1;
00355   unsigned int m_maxNbElementsOnLength = 100;
00357   std::array<double, 3> m_periodicLength{{0., 0., 0.}};
00359   unsigned int m_nCopiesX = 5;
00361   unsigned int m_nCopiesY = 5;
00363   unsigned int m_nCopiesZ = 5;
00364
00365   enum class Inversion { LU = 0, SVD };
00366   Inversion m_inversion = Inversion::LU;
00367
00369   std::map<std::string, int> m_wfields;
00370
00371   void InitValues();
00373   void ShiftPanels(std::vector<Panel>& panels) const;
00375   bool EliminateOverlaps(const Panel& panel1, const Panel& panel2,
00376                          std::vector<Panel>& panelsOut,
00377                          std::vector<int>& itypo);
00378
00379   bool TraceEnclosed(const std::vector<double>& xl1,
00380                      const std::vector<double>& yl1,
00381                      const std::vector<double>& xl2,
00382                      const std::vector<double>& yl2, const Panel& originalPanel,
00383                      std::vector<Panel>& newPanels) const;
00384
00385   void TraceNonOverlap(
00386       const std::vector<double>& xp1, const std::vector<double>& yp1,
00387       const std::vector<double>& xl1, const std::vector<double>& yl1,
00388       const std::vector<double>& xl2, const std::vector<double>& yl2,
00389       const std::vector<int>& flags1, const std::vector<int>& flags2,
00390       const std::vector<int>& links1, const std::vector<int>& links2,
00391       std::vector<bool>& mark1, int ip1, const Panel& originalPanel,
00392       std::vector<Panel>& newPanels) const;
00393
00394   void TraceOverlap(
00395       const std::vector<double>& xp1, const std::vector<double>& yp1,
00396       const std::vector<double>& xp2, const std::vector<double>& yp2,
00397       const std::vector<double>& xl1, const std::vector<double>& yl1,
00398       const std::vector<double>& xl2, const std::vector<double>& yl2,
00399       const std::vector<int>& flags1, const std::vector<int>& links1,
00400       const std::vector<int>& links2, std::vector<bool>& mark1, int ip1,
00401       int ip2, const Panel& originalPanel, std::vector<Panel>& newPanels) const;
00402
00404   bool MakePrimitives(const Panel& panelIn,
00405                       std::vector<Panel>& panelsOut) const;
00406
00409   bool SplitTrapezium(const Panel panelIn, std::vector<Panel>& stack,
```

```
00410                     std::vector<Panel>& panelsOut, const double epsang) const;
00411
00412   unsigned int NbOfSegments(const double length, const double target) const;
00413   bool DiscretizeWire(const Primitive& primitive, const double targetSize,
00414                       std::vector<Element>& elements) const;
00415   bool DiscretizeTriangle(const Primitive& primitive, const double targetSize,
00416                           std::vector<Element>& elements) const;
00417   bool DiscretizeRectangle(const Primitive& prim, const double targetSize,
00418                            std::vector<Element>& elements) const;
00419   int InterfaceType(const Solid::BoundaryCondition bc) const;
00420 };
00421
00422 extern ComponentNeBem3d* gComponentNeBem3d;
00423
00424 }  // namespace Garfield
00425
00426 #endif
```

## 8.35 /builds/garfield/docs/source/Include/Garfield/ComponentNeBem3dMap.hh File Reference

#include "Garfield/Component.hh"

### Classes

- class Garfield::ComponentNeBem3dMap

  Component for interpolating field maps stored in a mesh generated by neBEM.
- struct Garfield::ComponentNeBem3dMap::Element

### Namespaces

- namespace Garfield

## 8.36 ComponentNeBem3dMap.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__NEBEM3DMAP__H
00002 #define G__COMPONENT__NEBEM3DMAP__H
00003
00004 #include "Garfield/Component.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class ComponentNeBem3dMap : public Component {
00011  public:
00013   ComponentNeBem3dMap();
00015   ~ComponentNeBem3dMap() {}
00016
00017   void ElectricField(const double x, const double y, const double z, double& ex,
00018                      double& ey, double& ez, double& v, Medium*& m,
00019                      int& status) override;
00020   void ElectricField(const double x, const double y, const double z, double& ex,
00021                      double& ey, double& ez, Medium*& m, int& status) override;
00022
00023   void WeightingField(const double x, const double y, const double z,
00024                       double& wx, double& wy, double& wz,
00025                       const std::string& label) override;
00026   double WeightingPotential(const double x, const double y, const double z,
00027                             const std::string& label) override;
00028
00029   void MagneticField(const double x, const double y, const double z, double& bx,
00030                      double& by, double& bz, int& status) override;
00031
00035   void SetWeightingFieldOffset(const double x, const double y, const double z);
00036
00037   Medium* GetMedium(const double x, const double y, const double z) override;
00038
00039   bool GetVoltageRange(double& vmin, double& vmax) override;
00040   bool GetElectricFieldRange(double& exmin, double& exmax, double& eymin,
00041                              double& eymax, double& ezmin, double& ezmax);
00042   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00043                       double& ymax, double& zmax) override;
00044
00046   bool LoadMapInfo(const std::string& MapInfoFile, std::string& MapVersion,
00047                    int& OptMap, int& OptStaggerMap, unsigned int& NbOfXCells,
```

```
00048                 unsigned int& NbOfYCells, unsigned int& NbOfZCells,
00049                 double& Xmin, double& Xmax, double& Ymin, double& Ymax,
00050                 double& Zmin, double& Zmax, double& XStagger,
00051                 double& YStagger, double& ZStagger,
00052                 std::string& MapDataFile);
00059   void SetMesh(const unsigned int nx, const unsigned int ny,
00060                const unsigned int nz, const double xmin, const double xmax,
00061                const double ymin, const double ymax, const double zmin,
00062                const double zmax);
00079   bool LoadElectricField(const std::string& filename, const std::string& format,
00080                          const bool withPotential, const bool withRegion,
00081                          const double scaleX = 1., const double scaleE = 1.,
00082                          const double scaleP = 1.);
00084   bool LoadMagneticField(const std::string& filename, const std::string& format,
00085                          const double scaleX = 1., const double scaleB = 1.);
00086
00088   bool GetElement(const double xi, const double yi, const double zi,
00089                   unsigned int& i, unsigned int& j, unsigned int& k,
00090                   bool& xMirrored, bool& yMirrored, bool& zMirrored) const;
00092   bool GetElement(const unsigned int i, const unsigned int j,
00093                   const unsigned int k, double& v, double& ex, double& ey,
00094                   double& ez) const;
00095
00097   void SetMedium(const unsigned int i, Medium* m);
00099   Medium* GetMedium(const unsigned int i) const;
00101   void PrintRegions() const;
00102
00103  private:
00104   std::vector<Medium*> m_media;
00105   struct Element {
00106     double fx, fy, fz;
00107     double v;
00108   };
00110   std::vector<std::vector<std::vector<Element> > > m_efields;
00112   std::vector<std::vector<std::vector<Element> > > m_bfields;
00114   std::vector<std::vector<std::vector<int> > > m_regions;
00115   // Dimensions of the mesh
00116   unsigned int m_nX = 0, m_nY = 0, m_nZ = 0;
00117   double m_xMin = 0., m_yMin = 0., m_zMin = 0.;
00118   double m_xMax = 0., m_yMax = 0., m_zMax = 0.;
00119
00120   bool m_hasMesh = false;
00121   bool m_hasPotential = false;
00122   bool m_hasEfield = false;
00123   bool m_hasBfield = false;
00124
00125   // Offset for weighting field
00126   double m_wField_xOffset = 0.;
00127   double m_wField_yOffset = 0.;
00128   double m_wField_zOffset = 0.;
00129
00130   // Voltage range
00131   double m_pMin = 0., m_pMax = 0.;
00132
00134   bool LoadData(const std::string& filename, std::string format,
00135                 const bool withPotential, const bool withRegion,
00136                 const double scaleX, const double scaleF, const double scaleP,
00137                 const char field);
00138
00139   void Reset() override;
00140   void UpdatePeriodicity() override;
00141
00143   double Reduce(const double xin, const double xmin, const double xmax,
00144                 const bool simplePeriodic, const bool mirrorPeriodic,
00145                 bool& isMirrored) const;
00146
00148   double TriLinInt(const double xd, const double yd, const double zd,
00149                    const double c000, const double c100, const double c010,
00150                    const double c001, const double c110, const double c101,
00151                    const double c011, const double c111);
00152 };
00153 }  // namespace Garfield
00154 #endif
```

## 8.37 /builds/garfield/docs/source/Include/Garfield/ComponentParallelPlate.hh File Reference

#include <TF1.h>
#include <TF2.h>
#include <string>
#include "Garfield/Component.hh"
#include "Garfield/ComponentGrid.hh"

Classes

- class Garfield::ComponentParallelPlate

  Component for parallel-plate geometries.
- struct Garfield::ComponentParallelPlate::Electrode

  Structure that captures the information of the electrodes under study.

Namespaces

- namespace Garfield

## 8.38 ComponentParallelPlate.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__PP__H
00002 #define G__COMPONENT__PP__H
00003
00004 #include <TF1.h>
00005 #include <TF2.h>
00006
00007 #include <string>
00008
00009 #include "Garfield/Component.hh"
00010 #include "Garfield/ComponentGrid.hh"
00011
00012 namespace Garfield {
00013
00014 class Medium;
00015 class ComponentGrid;
00016
00018
00019 class ComponentParallelPlate : public Component {
00020  public:
00022   ComponentParallelPlate();
00024   ~ComponentParallelPlate() {}
00025
00038   void Setup(const unsigned int N, std::vector<double> eps,
00039              std::vector<double> d,
00040              const double V, std::vector<int> sigmaIndex = {});
00041
00042   void ElectricField(const double x, const double y, const double z, double &ex,
00043                      double &ey, double &ez, Medium *&m, int &status) override;
00044   void ElectricField(const double x, const double y, const double z, double &ex,
00045                      double &ey, double &ez, double &v, Medium *&m,
00046                      int &status) override;
00047   using Component::ElectricField;
00048   double WeightingPotential(const double x, const double y, const double z,
00049                             const std::string &label) override;
00050
00051   bool GetVoltageRange(double &vmin, double &vmax) override;
00052
00061   void AddPixel(double x, double z, double lx, double lz,
00062                 const std::string &label, bool fromAnode = true);
00064   void AddStrip(double z, double lz, const std::string &label,
00065                 bool fromAnode = true);
00066
00069   void AddPlane(const std::string &label, bool fromAnode = true);
00070
00072   void SetMedium(Medium *medium) { m_medium = medium; }
00073
00081   void SetWeightingPotentialGrid(const double xmin, const double xmax,
00082                                  const double xsteps, const double ymin,
00083                                  const double ymax, const double ysteps,
00084                                  const double zmin, const double zmax,
00085                                  const double zsteps, const std::string &label);
00086
00089   void SetWeightingPotentialGrids(const double xmin, const double xmax,
00090                                   const double xsteps, const double ymin,
00091                                   const double ymax, const double ysteps,
00092                                   const double zmin, const double zmax,
00093                                   const double zsteps);
00094
00097   void LoadWeightingPotentialGrid(const std::string &label) {
00098     for (auto &electrode : m_readout_p) {
00099       if (electrode.label != label) continue;
00100       if (electrode.grid.LoadWeightingField(label + "map", "xyz", true)) {
00101         std::cout << m_className << "::LoadWeightingPotentialGrid: "
00102                   << "Weighting potential set for " << label << ".\n";
```

```
00103         electrode.m_usegrid = true;
00104         return;
00105       }
00106     }
00107     std::cerr « m_className
00108               « "::LoadWeightingPotentialGrid: Could not find file for "
00109               « label « ".\n";
00110   }
00111
00112   Medium *GetMedium(const double x, const double y, const double z) override;
00113
00114   bool GetBoundingBox(double &xmin, double &ymin, double &zmin, double &xmax,
00115                       double &ymax, double &zmax) override;
00116
00117   // Obtain the index and permitivity of the layer at height z.
00118   // TODO: getLayer -> GetLayer
00119   bool getLayer(const double y, int &m, double &epsM) {
00120     m = -1;
00121     if (y < m_z[0]) return false;
00122     for (int i = 1; i < m_N; i++) {
00123       if (y <= m_z[i]) {
00124         m = i;
00125         break;
00126       }
00127     }
00128     if (m == -1) return false;
00129     epsM = m_epsHolder[m - 1];
00130     return true;
00131   }
00132   // Obtain the relative permittivity from layer at index m
00133   void getPermittivityFromLayer(int m, double &eps) {
00134     eps = m_epsHolder.at(m - 1);
00135   }
00136   // Obtain the z-coordinate bounds of layer m
00137   void getZBoundFromLayer(int m, double &zbottom, double &ztop) {
00138     ztop = m_z.at(m);
00139     zbottom = m_z.at(m - 1);
00140   }
00141   // Obtain number of layers
00142   int NumberOfLayers() { return m_N - 1; }
00143   // Get the indices of the gas gaps
00144   void IndexOfGasGaps(std::vector<int> &indexGasGap) {
00145     indexGasGap = {};
00146     for (int i = 1; i < m_N; i++) {
00147       if (!m_conductive[i]) indexGasGap.push_back(i);
00148     }
00149   }
00150
00151   void SetIntegrationPrecision(const double eps) { m_precision = eps; }
00152
00153   void SetIntegrationUpperbound(const double p) { m_upperBoundIntegration = p; }
00154
00155   void DisablePotentialCalculationOutsideGasGap() {
00156     m_getPotentialInPlate = false;
00157   }
00158
00159 private:
00160   double m_precision = 1.e-12;
00161   static constexpr double m_Vw = 1.;
00163   double m_V = 0.;
00164
00165   bool m_getPotentialInPlate = true;
00166
00167   int m_N = 0;
00168
00169   double m_upperBoundIntegration = 30;
00170
00171   std::vector<double> m_eps;
00172   std::vector<double> m_epsHolder;
00173   std::vector<double> m_d;
00174   std::vector<double> m_z;
00175
00177   std::vector<bool> m_conductive;
00178
00179   TF2 m_hIntegrand;
00180
00181   TF1 m_wpStripIntegral;
00182   TF2 m_wpPixelIntegral;
00183
00184   std::vector<std::vector<std::vector<int»> m_sigmaMatrix;  // sigma_{i,j}^n,
00185                                                             // where n goes
00186                                                             // from 1 to N;
00187   std::vector<std::vector<std::vector<int»> m_thetaMatrix;  // theta_{i,j}^n,
00188                                                             // where n goes
00189                                                             // from 1 to N;
00190
00191   std::vector<std::vector<double» m_cMatrix;
```

```
00192   std::vector<std::vector<double» m_vMatrix;
00193   std::vector<std::vector<double» m_gMatrix;
00194   std::vector<std::vector<double» m_wMatrix;
00195
00196   int m_currentLayer = 0;
00197   double m_currentPosition = -1;
00198
00199   Medium *m_medium = nullptr;
00200
00202   struct Electrode {
00203     std::string label;
00204     int ind = structureelectrode::NotSet;
00205     double xpos, ypos;
00206     double lx, ly;
00207     bool formAnode = true;
00208
00209     bool m_usegrid = false;
00210     ComponentGrid grid;
00211   };
00212
00214   enum structureelectrode { NotSet = -1, Plane, Strip, Pixel };
00215
00216   // Vectors storing the readout electrodes.
00217   std::vector<std::string> m_readout;
00218   std::vector<Electrode> m_readout_p;
00219
00220   // Functions that calculate the weighting potential
00221   double IntegratePromptPotential(const Electrode &el, const double x,
00222                                   const double y, const double z);
00223
00224   void CalculateDynamicalWeightingPotential(const Electrode &el);
00225
00226   double FindWeightingPotentialInGrid(Electrode &el, const double x,
00227                                       const double y, const double z);
00228
00229   // Construct the sigma matrix needed to calculate the w, v, c and g
00230   // matrices
00231   bool Nsigma(int N, std::vector<std::vector<int» &sigmaMatrix);
00232
00233   // Construct the theta matrix needed to calculate the w, v, c and g
00234   // matrices
00235   bool Ntheta(int N, std::vector<std::vector<int» &thetaMatrix,
00236               std::vector<std::vector<int» &sigmaMatrix);
00237
00238   // Construct the sigma and theta matrices.
00239   void constructGeometryMatrices(const int N);
00240
00241   // Construct the w, v, c and g matrices needed for constructing
00242   // the weighting potentials equations.
00243   void constructGeometryFunction(const int N, const std::vector<double> &d);
00244
00245   // Build function h needed for the integrand of the weighting potential of a
00246   // strip and pixel
00247   void setHIntegrand();
00248
00249   // build integrand of weighting potential of a strip
00250   void setwpPixelIntegrand();
00251
00252   // build integrand of weighting potential of a pixel
00253   void setwpStripIntegrand();
00254
00255   // weighting field of a plane in layer with index "indexLayer"
00256   double constWEFieldLayer(const int indexLayer) {
00257     double invEz = 0;
00258     for (int i = 1; i <= m_N - 1; i++) {
00259       invEz += m_d[i - 1] / m_epsHolder[i - 1];
00260     }
00261     return 1 / (m_epsHolder[indexLayer - 1] * invEz);
00262   }
00263
00264   // weighting potential of a plane
00265   double wpPlane(const double z) {
00266     int im = -1;
00267     double epsM = -1;
00268     if (!getLayer(z, im, epsM)) return 0.;
00269     double v = 1 - (z - m_z[im - 1]) * constWEFieldLayer(im);
00270     for (int i = 1; i <= im - 1; i++) {
00271       v -= m_d[i - 1] * constWEFieldLayer(i);
00272     }
00273
00274     return v;
00275   }
00276
00277   // electric field in layer with index "indexLayer"
00278   double constEFieldLayer(const int indexLayer) {
00279     if (m_conductive[indexLayer]) return 0.;
00280     double invEz = 0;
```

```
00281     for (int i = 1; i <= m_N - 1; i++) {
00282       // TODO!
00283       if (m_conductive[indexLayer]) continue;
00284       invEz -= m_d[i - 1] / m_epsHolder[i - 1];
00285     }
00286     return m_V / (m_epsHolder[indexLayer - 1] * invEz);
00287   }
00288
00289   // function to convert decimal to binary expressed in n digits.
00290   bool decToBinary(int n, std::vector<int> &binaryNum);
00291
00292   // Rebuilds c, v, g and w matrix.
00293   void LayerUpdate(const double z, const int im, const double epsM) {
00294     if (z == m_currentPosition) return;
00295
00296     m_currentPosition = z;
00297
00298     if (im != m_currentLayer) {
00299       m_currentLayer = im;
00300       for (int i = 0; i < im - 1; i++) m_eps[i] = m_epsHolder[i];
00301       m_eps[im - 1] = epsM;
00302       m_eps[im] = epsM;
00303       for (int i = im + 1; i < m_N; i++) m_eps[i] = m_epsHolder[i - 1];
00304     }
00305
00306     double diff1 = m_z[im] - z;
00307     double diff2 = z - m_z[im - 1];
00308
00309     std::vector<double> d(m_N, 0.);
00310     for (int i = 0; i < im - 1; i++) d[i] = m_d[i];
00311     d[im - 1] = diff2;
00312     d[im] = diff1;
00313     for (int i = im + 1; i < m_N; i++) d[i] = m_d[i - 1];
00314     // TODO::Construct c and g matrices only for im != m_currentLayer.
00315     constructGeometryFunction(m_N, d);
00316   };
00317
00318   void UpdatePeriodicity() override;
00319   void Reset() override;
00320 };
00321 }  // namespace Garfield
00322 #endif
```

## 8.39 /builds/garfield/docs/source/Include/Garfield/ComponentTcad2d.hh File Reference

#include <array>
#include <memory>
#include "Garfield/ComponentTcadBase.hh"
#include "QuadTree.hh"

### Classes

- class Garfield::ComponentTcad2d

  Interpolation in a two-dimensional field map created by Sentaurus Device.

### Namespaces

- namespace Garfield

## 8.40 ComponentTcad2d.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__TCAD__2D__H
00002 #define G__COMPONENT__TCAD__2D__H
00003
00004 #include <array>
00005 #include <memory>
00006
00007 #include "Garfield/ComponentTcadBase.hh"
00008 #include "QuadTree.hh"
00009
00010 namespace Garfield {
00011
00013
```

```
00014 class ComponentTcad2d : public ComponentTcadBase<2> {
00015  public:
00017   ComponentTcad2d();
00019   ~ComponentTcad2d() {}
00020
00022   void SetRangeZ(const double zmin, const double zmax);
00023
00031   bool GetElement(const size_t i, double& vol, double& dmin, double& dmax,
00032                   int& type) const;
00034   bool GetNode(const size_t i, double& x, double& y, double& z) const override;
00035
00036   void ElectricField(const double x, const double y, const double z, double& ex,
00037                      double& ey, double& ez, double& v, Medium*& m,
00038                      int& status) override;
00039
00040   void ElectricField(const double x, const double y, const double z, double& ex,
00041                      double& ey, double& ez, Medium*& m, int& status) override {
00042     double v = 0.;
00043     ElectricField(x, y, z, ex, ey, ez, v, m, status);
00044   }
00045   using Component::ElectricField;
00046   Medium* GetMedium(const double x, const double y, const double z) override;
00047   void DelayedWeightingPotentials(const double x, const double y,
00048                                   const double z, const std::string& label,
00049                                   std::vector<double>& dwp) override;
00050
00051   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00052                       double& ymax, double& zmax) override;
00053   bool GetElementaryCell(double& xmin, double& ymin, double& zmin, double& xmax,
00054                          double& ymax, double& zmax) override;
00055
00056  private:
00057   // Bounding box
00058   bool m_hasRangeZ = false;
00059
00060   // Tetrahedral tree.
00061   std::unique_ptr<QuadTree> m_tree;
00062
00063   void Reset() override {
00064     Cleanup();
00065     m_hasRangeZ = false;
00066     m_ready = false;
00067   }
00068
00069   size_t FindElement(const double x, const double y,
00070                      std::array<double, nMaxVertices>& w) const;
00071   // Check whether a point is inside a given element and calculate the
00072   // shape functions if it is.
00073   bool InElement(const double x, const double y, const Element& element,
00074                  std::array<double, nMaxVertices>& w) const;
00075   bool InRectangle(const double x, const double y, const Element& element,
00076                    std::array<double, nMaxVertices>& w) const;
00077   bool InTriangle(const double x, const double y, const Element& element,
00078                   std::array<double, nMaxVertices>& w) const;
00079   bool OnLine(const double x, const double y, const Element& element,
00080               std::array<double, nMaxVertices>& w) const;
00081   bool AtPoint(const double x, const double y, const Element& element,
00082                std::array<double, nMaxVertices>& w) const;
00083
00084   bool Interpolate(const double x, const double y, const double z,
00085                    const std::vector<double>& field, double& f) override;
00086   bool Interpolate(const double x, const double y, const double z,
00087                    const std::vector<std::array<double, 2> >& field, double& fx,
00088                    double& fy, double& fz) override;
00089   void FillTree() override;
00090 };
00091 }  // namespace Garfield
00092 #endif
```

## 8.41 /builds/garfield/docs/source/Include/Garfield/ComponentTcad3d.hh File Reference

#include <array>
#include <iostream>
#include <memory>
#include "Garfield/ComponentTcadBase.hh"
#include "Garfield/TetrahedralTree.hh"

Classes

- class Garfield::ComponentTcad3d

  Interpolation in a three-dimensional field map created by Sentaurus Device.

Namespaces

- namespace Garfield

## 8.42 ComponentTcad3d.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__TCAD__3D__H
00002 #define G__COMPONENT__TCAD__3D__H
00003
00004 #include <array>
00005 #include <iostream>
00006 #include <memory>
00007
00008 #include "Garfield/ComponentTcadBase.hh"
00009 #include "Garfield/TetrahedralTree.hh"
00010
00011 namespace Garfield {
00012
00014
00015 class ComponentTcad3d : public ComponentTcadBase<3> {
00016  public:
00018   ComponentTcad3d();
00020   ~ComponentTcad3d() {}
00021
00029   bool GetElement(const size_t i, double& vol, double& dmin, double& dmax,
00030                   int& type) const;
00032   bool GetNode(const size_t i, double& x, double& y, double& z) const override;
00033
00034   void ElectricField(const double x, const double y, const double z, double& ex,
00035                      double& ey, double& ez, double& v, Medium*& m,
00036                      int& status) override;
00037   void ElectricField(const double x, const double y, const double z, double& ex,
00038                      double& ey, double& ez, Medium*& m, int& status) override;
00039   using Component::ElectricField;
00040   Medium* GetMedium(const double x, const double y, const double z) override;
00041   void DelayedWeightingPotentials(const double x, const double y,
00042                                   const double z, const std::string& label,
00043                                   std::vector<double>& dwp) override;
00044
00045   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00046                       double& ymax, double& zmax) override;
00047   bool GetElementaryCell(double& xmin, double& ymin, double& zmin, double& xmax,
00048                          double& ymax, double& zmax) override;
00049
00050  private:
00051   // Tetrahedral tree.
00052   std::unique_ptr<TetrahedralTree> m_tree;
00053
00054   void Reset() override {
00055     Cleanup();
00056     m_ready = false;
00057   }
00058
00059   size_t FindElement(const double x, const double y, const double z,
00060                      std::array<double, nMaxVertices>& w) const;
00061   bool InElement(const double x, const double y, const double z,
00062                  const Element& element,
00063                  std::array<double, nMaxVertices>& w) const {
00064     if (x < element.bbMin[0] || x > element.bbMax[0] || y < element.bbMin[1] ||
00065         y > element.bbMax[1] || z < element.bbMin[2] || z > element.bbMax[2]) {
00066       return false;
00067     }
00068     bool inside = false;
00069     switch (element.type) {
00070       case 2:
00071         if (InTriangle(x, y, z, element, w)) inside = true;
00072         break;
00073       case 5:
00074         if (InTetrahedron(x, y, z, element, w)) inside = true;
00075         break;
00076       default:
00077         std::cerr << m_className << "::InElement:\n"
00078                   << "    Invalid element type (" << element.type << ").\n";
00079         break;
00080     }
00081     return inside;
```

```
00082   }
00083   bool InTetrahedron(const double x, const double y, const double z,
00084                      const Element& element,
00085                      std::array<double, nMaxVertices>& w) const;
00086   bool InTriangle(const double x, const double y, const double z,
00087                   const Element& element,
00088                   std::array<double, nMaxVertices>& w) const;
00089
00090   bool Interpolate(const double x, const double y, const double z,
00091                    const std::vector<double>& field, double& f) override;
00092   bool Interpolate(const double x, const double y, const double z,
00093                    const std::vector<std::array<double, 3> >& field, double& fx,
00094                    double& fy, double& fz) override;
00095   void FillTree() override;
00096 };
00097 }  // namespace Garfield
00098 #endif
```

## 8.43 /builds/garfield/docs/source/Include/Garfield/ComponentTcadBase.hh File Reference

#include <algorithm>
#include <array>
#include <map>
#include "Garfield/Component.hh"

### Classes

- class Garfield::ComponentTcadBase< N >

  Interpolation in a field map created by Sentaurus Device.
- struct Garfield::ComponentTcadBase< N >::Region
- struct Garfield::ComponentTcadBase< N >::Element
- struct Garfield::ComponentTcadBase< N >::Defect

### Namespaces

- namespace Garfield

## 8.44 ComponentTcadBase.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__TCAD__BASE__H
00002 #define G__COMPONENT__TCAD__BASE__H
00003
00004 #include <algorithm>
00005 #include <array>
00006 #include <map>
00007
00008 #include "Garfield/Component.hh"
00009
00010 namespace Garfield {
00011
00013
00014 template <size_t N>
00015 class ComponentTcadBase : public Component {
00016  public:
00018   ComponentTcadBase() = delete;
00020   ComponentTcadBase(const std::string& name) : Component(name) {
00021     m_regions.reserve(10);
00022     m_vertices.reserve(10000);
00023     m_elements.reserve(10000);
00024   }
00026   virtual ~ComponentTcadBase() {}
00027
00032   bool Initialise(const std::string& gridfilename,
00033                   const std::string& datafilename);
00034
00045   bool SetWeightingField(const std::string& datfile1,
00046                          const std::string& datfile2, const double dv,
00047                          const std::string& label);
00048
00050   bool SetWeightingPotential(const std::string& datfile1,
00051                              const std::string& datfile2, const double dv,
00052                              const std::string& label) {
```

```
00053   return SetWeightingField(datfile1, datfile2, dv, label);
00054 }
00058 bool SetWeightingFieldShift(const std::string& label, const double x,
00059                             const double y, const double z);
00061 bool SetDynamicWeightingPotential(const std::string& datfile1,
00062                                   const std::string& datfile2,
00063                                   const double dv, const double t,
00064                                   const std::string& label);
00066 bool SetDynamicWeightingField(const std::string& datfile1,
00067                               const std::string& datfile2, const double dv,
00068                               const double t, const std::string& label);
00069
00071 void PrintRegions() const;
00073 size_t GetNumberOfRegions() const { return m_regions.size(); }
00075 void GetRegion(const size_t ireg, std::string& name, bool& active) const;
00077 void SetDriftRegion(const size_t ireg);
00079 void UnsetDriftRegion(const size_t ireg);
00081 void SetMedium(const size_t ireg, Medium* m);
00083 void SetMedium(const std::string& material, Medium* m);
00084
00085 size_t GetNumberOfElements() const override { return m_elements.size(); }
00086 bool GetElementNodes(const size_t i,
00087                      std::vector<size_t>& nodes) const override;
00088 bool GetElementRegion(const size_t i, size_t& region,
00089                       bool& active) const override;
00090 size_t GetNumberOfNodes() const override { return m_vertices.size(); }
00091
00093 void EnableVelocityMap(const bool on);
00094
00096 size_t GetNumberOfDonors() { return m_donors.size(); }
00098 size_t GetNumberOfAcceptors() { return m_acceptors.size(); }
00099
00106 bool SetDonor(const size_t donorNumber, const double exsec,
00107               const double hxsec, const double concentration);
00109 bool SetAcceptor(const size_t acceptorNumber, const double exsec,
00110                  const double hxsec, const double concentration);
00111
00113 void EnableAlphaMap(const bool on = true) { m_useAlphaMap = on; }
00114
00116 void EnableTrapOccupationMap(const bool on = true);
00118 void EnableLifetimeMap(const bool on = true);
00119
00121 bool GetElectronMobility(const double x, const double y, const double z,
00122                          double& mob);
00124 bool GetHoleMobility(const double x, const double y, const double z,
00125                      double& mob);
00126
00127 double ElectronLifetime(const double x, const double y, const double z);
00128 double HoleLifetime(const double x, const double y, const double z);
00129
00130 void WeightingField(const double x, const double y, const double z,
00131                     double& wx, double& wy, double& wz,
00132                     const std::string& label) override;
00133 double WeightingPotential(const double x, const double y, const double z,
00134                           const std::string& label) override;
00135 const std::vector<double>& DelayedSignalTimes(
00136     const std::string& label) override {
00137   if (m_dwtp.count(label) > 0) return m_dwtp[label];
00138   static const std::vector<double> emptyVector;
00139   return emptyVector;
00140 }
00141 void DelayedWeightingField(const double x, const double y, const double z,
00142                            const double t, double& wx, double& wy, double& wz,
00143                            const std::string& label) override;
00144 double DelayedWeightingPotential(const double x, const double y,
00145                                  const double z, const double t,
00146                                  const std::string& label) override;
00147
00148 bool GetVoltageRange(double& vmin, double& vmax) override;
00149
00150 bool HasVelocityMap() const override {
00151   return m_useVelocityMap && !(m_eVelocity.empty() && m_hVelocity.empty());
00152 }
00153 bool ElectronVelocity(const double x, const double y, const double z,
00154                       double& vx, double& vy, double& vz) override;
00155 bool HoleVelocity(const double x, const double y, const double z, double& vx,
00156                   double& vy, double& vz) override;
00157
00158 bool HasTownsendMap() const override {
00159   return m_useAlphaMap && !(m_eAlpha.empty() && m_hAlpha.empty());
00160 }
00161 bool HasAttachmentMap() const override {
00162   return ((m_useTrapOccMap || m_useLifetimeMap) &&
00163           !(m_eEta.empty() && m_hEta.empty()));
00164 }
00165 bool HasMobilityMap() const override {
00166   return !(m_eMobility.empty() && m_hMobility.empty());
```

```
00167   }
00168   bool ElectronAttachment(const double x, const double y, const double z,
00169                           double& eta) override;
00170   bool HoleAttachment(const double x, const double y, const double z,
00171                       double& eta) override;
00172   bool ElectronMobility(const double x, const double y, const double z,
00173                         double& mu) override;
00174   bool HoleMobility(const double x, const double y, const double z,
00175                     double& mu) override;
00176
00177   bool ElectronTownsend(const double x, const double y, const double z,
00178                         double& alpha) override;
00179   bool HoleTownsend(const double x, const double y, const double z,
00180                     double& alpha) override;
00181
00182  protected:
00183   // Max. number of vertices per element
00184   static constexpr size_t nMaxVertices = 4;
00185
00186   // Regions
00187   struct Region {
00188     // Name of the region (from Tcad)
00189     std::string name;
00190     // Material of the region (from Tcad)
00191     std::string material;
00192     // Flag indicating if the region is active (i. e. a drift medium)
00193     bool drift;
00194     // Medium object associated to the region
00195     Medium* medium;
00196   };
00197   std::vector<Region> m_regions;
00198
00199   // Vertex coordinates [cm].
00200   std::vector<std::array<double, N> > m_vertices;
00201
00202   // Elements
00203   struct Element {
00204     // Indices of vertices
00205     unsigned int vertex[nMaxVertices];
00206     // Type of element
00207     // 0: Point
00208     // 1: Segment (line)
00209     // 2: Triangle
00210     // 3: Rectangle
00211     // 4: Polygon
00212     // 5: Tetrahedron
00213     // 6: Pyramid
00214     // 7: Prism
00215     // 8: Brick
00216     // 9: Tetrabrick
00217     // 10: Polyhedron
00218     // In 2D, types 1 - 3 are supported.
00219     // In 3D, only types 2 and 5 are supported.
00220     unsigned int type;
00221     // Associated region
00222     unsigned int region;
00223     // Bounding box
00224     std::array<float, N> bbMin;
00225     std::array<float, N> bbMax;
00226   };
00227   std::vector<Element> m_elements;
00228
00229   // Potential [V] at each vertex.
00230   std::vector<double> m_epot;
00231   // Electric field [V / cm].
00232   std::vector<std::array<double, N> > m_efield;
00233
00234   // Weighting field and potential at each vertex.
00235   std::map<std::string, std::vector<std::array<double, N> > > m_wfield;
00236   std::map<std::string, std::vector<double> > m_wpot;
00237   // Weighting field labels and offsets.
00238   std::map<std::string, std::vector<double> > m_wshift;
00239
00240   // Delayed weighting field and potential.
00241   std::map<std::string, std::vector<std::vector<std::array<double, N> > > >
00242       m_dwf;
00243   std::map<std::string, std::vector<std::vector<double> > > m_dwp;
00244   // Times corresponding to the delayed weighting fields/potentials.
00245   std::map<std::string, std::vector<double> > m_dwtf;
00246   std::map<std::string, std::vector<double> > m_dwtp;
00247
00248   // Velocities [cm / ns]
00249   std::vector<std::array<double, N> > m_eVelocity;
00250   std::vector<std::array<double, N> > m_hVelocity;
00251   // Mobilities [cm2 / (V ns)]
00252   std::vector<double> m_eMobility;
00253   std::vector<double> m_hMobility;
```

```
00254   // Impact ionisation coefficients [1 / cm]
00255   std::vector<double> m_eAlpha;
00256   std::vector<double> m_hAlpha;
00257   // Lifetimes [ns]
00258   std::vector<double> m_eLifetime;
00259   std::vector<double> m_hLifetime;
00260   // Trap occupations [dimensionless]
00261   std::vector<std::vector<float> > m_donorOcc;
00262   std::vector<std::vector<float> > m_acceptorOcc;
00263   // Attachment coefficients [1 / cm]
00264   std::vector<double> m_eEta;
00265   std::vector<double> m_hEta;
00266
00267   struct Defect {
00268     // Electron cross-section
00269     double xsece;
00270     // Hole cross-section
00271     double xsech;
00272     // Concentration
00273     double conc;
00274   };
00275   std::vector<Defect> m_donors;
00276   std::vector<Defect> m_acceptors;
00277
00278   // Use velocity map or not.
00279   bool m_useVelocityMap = false;
00280   // Use trap occupation probability map or not.
00281   bool m_useTrapOccMap = false;
00282   // Use lifetime map or not.
00283   bool m_useLifetimeMap = false;
00284   // Use impact ionisation map or not.
00285   bool m_useAlphaMap = false;
00286
00287   // Bounding box.
00288   std::array<double, 3> m_bbMin = {{0., 0., 0.}};
00289   std::array<double, 3> m_bbMax = {{0., 0., 0.}};
00290
00291   // Voltage range
00292   double m_pMin = 0.;
00293   double m_pMax = 0.;
00294
00295   void UpdatePeriodicity() override;
00296
00297   void Cleanup();
00298
00299   static unsigned int ElementVertices(const Element& element) {
00300     return std::min(element.type + 1, 4U);
00301   }
00302   virtual bool Interpolate(const double x, const double y, const double z,
00303                            const std::vector<double>& field, double& f) = 0;
00304   virtual bool Interpolate(const double x, const double y, const double z,
00305                            const std::vector<std::array<double, N> >& field,
00306                            double& fx, double& fy, double& fz) = 0;
00307   virtual void FillTree() = 0;
00308
00309   size_t FindRegion(const std::string& name) const;
00310   void MapCoordinates(std::array<double, N>& x,
00311                       std::array<bool, N>& mirr) const;
00312   bool InBoundingBox(const std::array<double, N>& x) const {
00313     for (size_t i = 0; i < N; ++i) {
00314       if (x[i] < m_bbMin[i] || x[i] > m_bbMax[i]) return false;
00315     }
00316     return true;
00317   }
00318   void UpdateAttachment();
00319   void ComputeEtaFromLifetime();
00320   void ComputeEtaFromTraps();
00321
00322   bool GetOffset(const std::string& label, double& dx, double& dy,
00323                  double& dz) const;
00324   bool LoadGrid(const std::string& gridfilename);
00325   bool LoadData(const std::string& datafilename);
00326   bool ReadDataset(std::ifstream& datafile, const std::string& dataset);
00327   bool LoadWeightingField(const std::string& datafilename,
00328                           std::vector<std::array<double, N> >& wf,
00329                           std::vector<double>& wp);
00330 };
00331 }  // namespace Garfield
00332 #endif
```

## 8.45 /builds/garfield/docs/source/Include/Garfield/ComponentUser.hh File Reference

#include <array>
#include <functional>

#include <map>
#include "Garfield/Component.hh"

Classes

- class Garfield::ComponentUser

  Simple component with electric field given by a user function.

Namespaces

- namespace Garfield

## 8.46 ComponentUser.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__USER__H
00002 #define G__COMPONENT__USER__H
00003
00004 #include <array>
00005 #include <functional>
00006 #include <map>
00007
00008 #include "Garfield/Component.hh"
00009
00010 namespace Garfield {
00011
00013
00014 class ComponentUser : public Component {
00015  public:
00017   ComponentUser();
00019   ~ComponentUser() {}
00020
00022   void SetElectricField(
00023       std::function<void(const double, const double, const double, double&,
00024                          double&, double&)>);
00026   void SetElectricField(const std::string& expression);
00027
00029   void SetPotential(
00030       std::function<double(const double, const double, const double)>);
00032   void SetPotential(const std::string& expression);
00033
00035   void SetWeightingField(
00036       std::function<void(const double, const double, const double, double&,
00037                          double&, double&)>,
00038       const std::string& label);
00040   void SetWeightingField(const std::string& expression,
00041                          const std::string& label);
00042
00044   void SetWeightingPotential(
00045       std::function<double(const double, const double, const double)>,
00046       const std::string& label);
00048   void SetWeightingPotential(const std::string& expression,
00049                              const std::string& label);
00050
00052   void SetDelayedWeightingField(
00053       std::function<void(const double, const double, const double, const double,
00054                          double&, double&, double&)>,
00055       const std::string& label);
00057   void SetDelayedWeightingField(const std::string& expression,
00058                                 const std::string& label);
00059
00062   void SetDelayedWeightingPotential(
00063       std::function<double(const double, const double, const double,
00064                            const double)>,
00065       const std::string& label);
00068   void SetDelayedWeightingPotential(const std::string& expression,
00069                                     const std::string& label);
00072   void SetDelayedSignalTimes(const std::vector<double>& ts);
00074   void SetMagneticField(
00075       std::function<void(const double, const double, const double, double&,
00076                          double&, double&)>);
00078   void SetMagneticField(const std::string& expression);
00079
00082   void SetArea(const double xmin, const double ymin, const double zmin,
00083                const double xmax, const double ymax, const double zmax);
00085   void UnsetArea();
00087   void SetMedium(Medium* medium) { m_medium = medium; }
00088
00089   Medium* GetMedium(const double x, const double y, const double z) override {
```

```
00090     return !m_hasArea        ? Component::GetMedium(x, y, z)
00091           : InArea(x, y, z) ? m_medium
00092                             : nullptr;
00093   }
00094   void ElectricField(const double x, const double y, const double z, double& ex,
00095                      double& ey, double& ez, Medium*& m, int& status) override;
00096   void ElectricField(const double x, const double y, const double z, double& ex,
00097                      double& ey, double& ez, double& v, Medium*& m,
00098                      int& status) override;
00099   using Component::ElectricField;
00100   bool GetVoltageRange(double& vmin, double& vmax) override;
00101   void MagneticField(const double x, const double y, const double z, double& bx,
00102                      double& by, double& bz, int& status) override;
00103   void WeightingField(const double x, const double y, const double z,
00104                       double& wx, double& wy, double& wz,
00105                       const std::string& label) override;
00106   double WeightingPotential(const double x, const double y, const double z,
00107                             const std::string& label) override;
00108   void DelayedWeightingField(const double x, const double y, const double z,
00109                              const double t, double& wx, double& wy, double& wz,
00110                              const std::string& label) override;
00111   double DelayedWeightingPotential(const double x, const double y,
00112                                    const double z, const double t,
00113                                    const std::string& label) override;
00114   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00115                       double& ymax, double& zmax) override;
00116
00117   bool HasMagneticField() const override;
00118
00119  private:
00121   std::function<void(const double, const double, const double, double&, double&,
00122                      double&)>
00123       m_efield;
00125   std::function<double(const double, const double, const double)> m_epot;
00126
00128   std::map<std::string,
00129           std::function<void(const double, const double, const double, double&,
00130                              double&, double&)> >
00131       m_wfield;
00132
00134   std::map<std::string,
00135           std::function<double(const double, const double, const double)> >
00136       m_wpot;
00137
00139   std::map<std::string,
00140           std::function<void(const double, const double, const double,
00141                              const double, double&, double&, double&)> >
00142       m_dwfield;
00143
00145   std::map<std::string, std::function<double(const double, const double,
00146                                              const double, const double)> >
00147       m_dwpot;
00148
00150   std::function<void(const double, const double, const double, double&, double&,
00151                      double&)>
00152       m_bfield;
00153
00154   // Active area.
00155   std::array<double, 3> m_xmin = {{0., 0., 0.}};
00156   std::array<double, 3> m_xmax = {{0., 0., 0.}};
00157   // Did we specify the active area explicitly?
00158   bool m_hasArea = false;
00159   // Medium in the active area.
00160   Medium* m_medium = nullptr;
00161
00163   void Reset() override;
00164   // Verify periodicities
00165   void UpdatePeriodicity() override;
00166
00167   bool InArea(const double x, const double y, const double z) {
00168     return (x >= m_xmin[0] && x <= m_xmax[0] && y >= m_xmin[1] &&
00169             y <= m_xmax[1] && z >= m_xmin[2] && z <= m_xmax[2]);
00170   }
00171 };
00172 }  // namespace Garfield
00173 #endif
```

## 8.47 /builds/garfield/docs/source/Include/Garfield/ComponentVoxel.hh File Reference

#include ”Garfield/Component.hh”

Classes

- class Garfield::ComponentVoxel

  Component for interpolating field maps stored in a regular mesh.
- struct Garfield::ComponentVoxel::Element

Namespaces

- namespace Garfield

## 8.48 ComponentVoxel.hh

Go to the documentation of this file.

```
00001 #ifndef G__COMPONENT__VOXEL__H
00002 #define G__COMPONENT__VOXEL__H
00003
00004 #include "Garfield/Component.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class ComponentVoxel : public Component {
00011  public:
00013   ComponentVoxel();
00015   ~ComponentVoxel() {}
00016
00018   void EnableInterpolation(const bool on = true) { m_interpolate = on; }
00019
00026   void SetMesh(const unsigned int nx, const unsigned int ny,
00027                const unsigned int nz, const double xmin, const double xmax,
00028                const double ymin, const double ymax, const double zmin,
00029                const double zmax);
00046   bool LoadElectricField(const std::string& filename, const std::string& format,
00047                          const bool withPotential, const bool withRegion,
00048                          const double scaleX = 1., const double scaleE = 1.,
00049                          const double scaleP = 1.);
00051   bool LoadWeightingField(const std::string& filename,
00052                           const std::string& format, const bool withPotential,
00053                           const double scaleX = 1., const double scaleE = 1.,
00054                           const double scaleP = 1.);
00058   void SetWeightingFieldOffset(const double x, const double y, const double z);
00059
00061   bool LoadWeightingField(const std::string& filename,
00062                           const std::string& format, const double time,
00063                           const bool withPotential, const double scaleX = 1.,
00064                           const double scaleE = 1., const double scaleP = 1.);
00066   bool LoadMagneticField(const std::string& filename, const std::string& format,
00067                          const double scaleX = 1., const double scaleB = 1.);
00068
00070   void SetMedium(const unsigned int i, Medium* m);
00072   Medium* GetMedium(const unsigned int i) const;
00074   void PrintRegions() const;
00075
00077   bool GetElement(const double xi, const double yi, const double zi,
00078                   unsigned int& i, unsigned int& j, unsigned int& k,
00079                   bool& xMirrored, bool& yMirrored, bool& zMirrored) const;
00081   bool GetElement(const unsigned int i, const unsigned int j,
00082                   const unsigned int k, double& v, double& ex, double& ey,
00083                   double& ez) const;
00084
00085   void Clear() override { Reset(); }
00086   void ElectricField(const double x, const double y, const double z, double& ex,
00087                      double& ey, double& ez, double& v, Medium*& m,
00088                      int& status) override;
00089   void ElectricField(const double x, const double y, const double z, double& ex,
00090                      double& ey, double& ez, Medium*& m, int& status) override;
00091   using Component::ElectricField;
00092   void WeightingField(const double x, const double y, const double z,
00093                       double& wx, double& wy, double& wz,
00094                       const std::string& label) override;
00095   double WeightingPotential(const double x, const double y, const double z,
00096                             const std::string& label) override;
00097   void DelayedWeightingField(const double x, const double y, const double z,
00098                              const double t, double& wx, double& wy, double& wz,
00099                              const std::string& label) override;
00100   double DelayedWeightingPotential(const double x, const double y,
00101                                    const double z, const double t,
00102                                    const std::string& label) override;
00103   void MagneticField(const double x, const double y, const double z, double& bx,
00104                      double& by, double& bz, int& status) override;
00105   bool HasMagneticField() const override;
00106
```

```
00107   Medium* GetMedium(const double x, const double y, const double z) override;
00108
00109   bool GetVoltageRange(double& vmin, double& vmax) override;
00110   bool GetElectricFieldRange(double& exmin, double& exmax, double& eymin,
00111                              double& eymax, double& ezmin, double& ezmax);
00112   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00113                       double& ymax, double& zmax) override;
00114   bool GetElementaryCell(double& xmin, double& ymin, double& zmin, double& xmax,
00115                          double& ymax, double& zmax) override;
00116
00117 private:
00118   std::vector<Medium*> m_media;
00119   struct Element {
00120     double fx, fy, fz;
00121     double v;
00122   };
00123
00125   std::vector<std::vector<std::vector<int> > > m_regions;
00127   std::vector<std::vector<std::vector<Element> > > m_efields;
00129   std::vector<std::vector<std::vector<Element> > > m_bfields;
00131   std::vector<std::vector<std::vector<Element> > > m_wfields;
00133   std::vector<std::vector<std::vector<std::vector<Element> > > > m_wdfields;
00134
00135   // Dimensions of the mesh
00136   unsigned int m_nX = 0, m_nY = 0, m_nZ = 0;
00137   double m_xMin = 0., m_yMin = 0., m_zMin = 0.;
00138   double m_xMax = 0., m_yMax = 0., m_zMax = 0.;
00139   double m_dx = 0., m_dy = 0., m_dz = 0.;
00140
00141   bool m_interpolate = false;
00142
00143   bool m_hasMesh = false;
00144   bool m_hasPotential = false;
00145   bool m_hasEfield = false;
00146   bool m_hasBfield = false;
00147   bool m_hasWfield = false;
00148
00149   // Offset for weighting field
00150   double m_wField_xOffset = 0.;
00151   double m_wField_yOffset = 0.;
00152   double m_wField_zOffset = 0.;
00153
00154   // Voltage range
00155   double m_pMin = 0., m_pMax = 0.;
00156
00158   bool LoadData(const std::string& filename, std::string format,
00159                 const bool withPotential, const bool withRegion,
00160                 const double scaleX, const double scaleF, const double scaleP,
00161                 std::vector<std::vector<std::vector<Element> > >& field);
00162
00163   void Reset() override;
00164   void UpdatePeriodicity() override;
00165
00167   bool GetField(const double x, const double y, const double z,
00168                 const std::vector<std::vector<std::vector<Element> > >& field,
00169                 double& fx, double& fy, double& fz, double& p, int& region);
00171   double Reduce(const double xin, const double xmin, const double xmax,
00172                 const bool simplePeriodic, const bool mirrorPeriodic,
00173                 bool& isMirrored) const;
00175   void Initialise(std::vector<std::vector<std::vector<Element> > >& fields);
00176   void InitialiseRegions();
00177 };
00178 }  // namespace Garfield
00179 #endif
```

## 8.49 /builds/garfield/docs/source/Include/Garfield/DegradeInterface.hh File Reference

#include <cstdint>

### Namespaces

- namespace Garfield
- namespace Garfield::Degrade

### Functions

- void Garfield::Degrade::deginit (int64_t *ng, int64_t *nevt, int64_t *mip, int64_t *idvec, int32_t *iseed, double *e0, double *et, double *ec, int64_t *ngas1, int64_t *ngas2, int64_t *ngas3, int64_t *ngas4, int64_t *ngas5, int64_t *ngas6, double *frac1, double *frac2, double *frac3, double *frac4,

double *frac5, double *frac6, double *t0, double *p0, double *etot, double *btot, double *bang, int64_t *jcmp, int64_t *jray, int64_t *jpap, int64_t *jbrm, int64_t *jecasc, int64_t *iverb)

- void Garfield::Degrade::getdedx (double *dedxi, double *cldensi)
- int64_t Garfield::Degrade::getie (double *e)
- double Garfield::Degrade::gettcf (int64_t *ie)
- double Garfield::Degrade::gettcfn (int64_t *ie)
- void Garfield::Degrade::getlevel (int64_t *ie, double *r1, int64_t *izbr, double *rgas, double *ein, int64_t *ia, double *wpl, int64_t *index, double *an, double *ps, double *wklm, int64_t *nc0, double *ec0, int64_t *ng1, double *eg1, int64_t *ng2, double *eg2, double *dstfl, int64_t *jpn, int64_t *kg1, int64_t *lg1, int64_t *igshel, int64_t *ionmdl, int64_t *ilvl)
- int64_t Garfield::Degrade::getgas (int64_t *ilvl)
- double Garfield::Degrade::ionpot ()
- void Garfield::Degrade::getebrem (int64_t *k, double *ee, double *xe, double *ye, double *ze, double *te, double *dxe, double *dye, double *dze, int64_t *iok)
- void Garfield::Degrade::getecasc (int64_t *k, double *ee, double *xe, double *ye, double *ze, double *te, double *dxe, double *dye, double *dze, int64_t *iok)
- void Garfield::Degrade::degrade ()
- void Garfield::Degrade::brems (int64_t *iz, double *ein, double *dx, double *dy, double *dz, double *eout, double *dxe, double *dye, double *dze, double *egamma, double *dxg, double *dyg, double *dzg)
- void Garfield::Degrade::bremscasc (int64_t *j11, double *egamma, double *x0, double *y0, double *z0, double *t0, double *gdcx, double *gdcy, double *gdcz, int64_t *ilow)
- void Garfield::Degrade::cascadee (int64_t *j11, int64_t *kgas, int64_t *lgas, double *x0, double *y0, double *z0, double *t0, double *einit, int64_t *ishell)
- void Garfield::Degrade::drcos (double *drx, double *dry, double *drz, double *theta, double *phi, double *drxx, double *dryy, double *drzz)
- void Garfield::Degrade::ionsplit (int64_t *i, double *e, double *ei, double *esec)

## 8.50 DegradeInterface.hh

Go to the documentation of this file.

```
00001 // Interface to Degrade
00002
00003 #ifndef G__DEGRADE__INTERFACE
00004 #define G__DEGRADE__INTERFACE
00005
00006 #include <cstdint>
00007
00008 #ifndef ___CINT___
00009
00010 namespace Garfield {
00011
00012 namespace Degrade {
00013
00014 extern "C" {
00015
00016 void deginit(int64_t* ng, int64_t* nevt, int64_t* mip, int64_t* idvec,
00017             int32_t* iseed, double* e0, double* et, double* ec, int64_t* ngas1,
00018             int64_t* ngas2, int64_t* ngas3, int64_t* ngas4, int64_t* ngas5,
00019             int64_t* ngas6, double* frac1, double* frac2, double* frac3,
00020             double* frac4, double* frac5, double* frac6, double* t0,
00021             double* p0, double* etot, double* btot, double* bang,
00022             int64_t* jcmp, int64_t* jray, int64_t* jpap, int64_t* jbrm,
00023             int64_t* jecasc, int64_t* iverb);
00024 void getdedx(double* dedxi, double* cldensi);
00025 int64_t getie(double* e);
00026 double gettcf(int64_t* ie);
00027 double gettcfn(int64_t* ie);
00028 void getlevel(int64_t* ie, double* r1, int64_t* izbr, double* rgas, double* ein,
00029              int64_t* ia, double* wpl, int64_t* index, double* an, double* ps,
00030              double* wklm, int64_t* nc0, double* ec0, int64_t* ng1,
00031              double* eg1, int64_t* ng2, double* eg2, double* dstfl,
00032              int64_t* jpn, int64_t* kg1, int64_t* lg1, int64_t* igshel,
00033              int64_t* ionmdl, int64_t* ilvl);
00034 int64_t getgas(int64_t* ilvl);
00035 double ionpot();
00036 void getebrem(int64_t* k, double* ee, double* xe, double* ye, double* ze,
00037              double* te, double* dxe, double* dye, double* dze, int64_t* iok);
00038 void getecasc(int64_t* k, double* ee, double* xe, double* ye, double* ze,
00039              double* te, double* dxe, double* dye, double* dze, int64_t* iok);
```

```
00040 void degrade();
00041 void brems(int64_t* iz, double* ein, double* dx, double* dy, double* dz,
00042           double* eout, double* dxe, double* dye, double* dze, double* egamma,
00043           double* dxg, double* dyg, double* dzg);
00044 void bremscasc(int64_t* j11, double* egamma, double* x0, double* y0, double* z0,
00045               double* t0, double* gdcx, double* gdcy, double* gdcz,
00046               int64_t* ilow);
00047 void cascadee(int64_t* j11, int64_t* kgas, int64_t* lgas, double* x0,
00048              double* y0, double* z0, double* t0, double* einit,
00049              int64_t* ishell);
00050 void drcos(double* drx, double* dry, double* drz, double* theta, double* phi,
00051           double* drxx, double* dryy, double* drzz);
00052 void ionsplit(int64_t* i, double* e, double* ei, double* esec);
00053 }
00054 }  // namespace Degrade
00055 }  // namespace Garfield
00056 #endif
00057 #endif
```

## 8.51 /builds/garfield/docs/source/Include/Garfield/DriftLineRKF.hh File Reference

#include <array>
#include <string>
#include <utility>
#include <vector>
#include "Garfield/GarfieldConstants.hh"

Classes

- class Garfield::DriftLineRKF

  Calculation of drift lines based on macroscopic transport coefficients using Runge-Kutta-Fehlberg integration.

Namespaces

- namespace Garfield

## 8.52 DriftLineRKF.hh

Go to the documentation of this file.

```
00001 #ifndef G__DRIFTLINE__RKF__H
00002 #define G__DRIFTLINE__RKF__H
00003
00004 #include <array>
00005 #include <string>
00006 #include <utility>
00007 #include <vector>
00008
00009 #include "Garfield/GarfieldConstants.hh"
00010
00011 namespace Garfield {
00012
00013 class Sensor;
00014 class ViewDrift;
00015 class Medium;
00018 class DriftLineRKF {
00019  public:
00021   DriftLineRKF() : DriftLineRKF(nullptr) {}
00023   DriftLineRKF(Sensor* sensor);
00025   ~DriftLineRKF() {}
00026
00028   void SetSensor(Sensor* s);
00029
00031   void EnablePlotting(ViewDrift* view);
00033   void DisablePlotting();
00034
00036   void EnableSignalCalculation(const bool on = true) { m_doSignal = on; }
00041   void SetSignalAveragingOrder(const unsigned int navg) { m_navg = navg; }
00045   void UseWeightingPotential(const bool on = true) {
00046     m_useWeightingPotential = on;
00047   }
00048
00050   void SetIntegrationAccuracy(const double eps);
00052   void SetMaximumStepSize(const double ms);
00055   void SetMaximumStepSize();
```

```
00057   void UnsetMaximumStepSize() { m_useStepSizeLimit = false; }
00060   void RejectKinks(const bool on = true) { m_rejectKinks = on; }
00061
00063   void SetElectronSignalScalingFactor(const double scale) { m_scaleE = scale; }
00065   void SetHoleSignalScalingFactor(const double scale) { m_scaleH = scale; }
00067   void SetIonSignalScalingFactor(const double scale) { m_scaleI = scale; }
00068
00070   void EnableAvalanche(const bool on = true) { m_doAvalanche = on; }
00072   void EnableIonTail(const bool on = true) {
00073     m_doIonTail = on;
00074     m_doIonTailAuto = false;
00075   }
00077   void EnableNegativeIonTail(const bool on = true) { m_doNegativeIonTail = on; }
00079   void SetGainFluctuationsFixed(const double gain = -1.);
00082   void SetGainFluctuationsPolya(const double theta, const double mean = -1.,
00083                                 const bool quiet = false);
00084
00086   void EnableTownsendMap(const bool on = true) { m_useTownsendMap = on; }
00088   void EnableVelocityMap(const bool on = true) { m_useVelocityMap = on; }
00089
00091   bool DriftElectron(const double x, const double y, const double z,
00092                      const double t, const size_t w = 1);
00094   bool DriftHole(const double x, const double y, const double z,
00095                  const double t, const size_t w = 1);
00097   bool DriftIon(const double x, const double y, const double z,
00098                 const double t, const size_t w = 1);
00101   bool DriftPositron(const double x, const double y, const double z,
00102                      const double t, const size_t w = 1);
00105   bool DriftNegativeIon(const double x, const double y, const double z,
00106                         const double t, const size_t w = 1);
00107
00109   void PrintDriftLine() const;
00111   void GetEndPoint(double& x, double& y, double& z, double& t, int& st) const;
00113   size_t GetNumberOfDriftLinePoints() const { return m_x.size(); }
00115   void GetDriftLinePoint(const size_t i, double& x, double& y, double& z,
00116                          double& t) const;
00117
00120   double GetArrivalTimeSpread(const double eps = 1.e-4) const;
00122   double GetGain(const double eps = 1.e-4) const;
00124   double GetLoss(const double eps = 1.e-4) const;
00126   double GetDriftTime() const {
00127     return m_t.empty() ? 0. : m_t.back() - m_t.front();
00128   }
00130   double GetPathLength() const;
00131
00133   void GetAvalancheSize(double& ne, double& ni) const {
00134     ne = m_nE;
00135     ni = m_nI;
00136   }
00138   std::pair<double, double> GetAvalancheSize() const {
00139     return std::make_pair(m_nE, m_nI);
00140   }
00141
00147   bool FieldLine(const double xi, const double yi, const double zi,
00148                  std::vector<std::array<float, 3> >& xl,
00149                  const bool electron = true) const;
00150
00152   void EnableDebugging(const bool on = true) { m_debug = on; }
00153
00154 private:
00155   std::string m_className = "DriftLineRKF";
00156
00157   // Pointer to sensor.
00158   Sensor* m_sensor = nullptr;
00159
00160   // Particle type of the most recent drift line.
00161   Particle m_particle = Particle::Electron;
00162
00163   // Maximum allowed step size.
00164   double m_maxStepSize = 0.;
00165   // Precision of the stepping algorithm.
00166   double m_accuracy = 1.e-8;
00167   // Flag to reject bends > 90 degrees or not.
00168   bool m_rejectKinks = true;
00169   // Flag to apply a cut on the maximum allowed step size or not.
00170   bool m_useStepSizeLimit = false;
00171
00172   // Pointer to the drift viewer.
00173   ViewDrift* m_view = nullptr;
00174
00175   // Points along the current drift line.
00176   std::vector<std::array<double, 3> > m_x;
00177   // Times corresponding to the points along the current drift line.
00178   std::vector<double> m_t;
00179   // Status flag of the current drift line.
00180   int m_status = 0;
00181
```

```
00182   // Flag whether to calculate induced signals or not.
00183   bool m_doSignal = true;
00184   // Averaging order used when projecting the signal on the time bins.
00185   unsigned int m_navg = 2;
00186   // Use weighting potential or weighting field for calculating the signal.
00187   bool m_useWeightingPotential = true;
00188
00189   // Scaling factor for electron signals.
00190   double m_scaleE = 1.;
00191   // Scaling factor for hole signals.
00192   double m_scaleH = 1.;
00193   // Scaling factor for ion signals.
00194   double m_scaleI = 1.;
00195
00196   // Use drift velocity maps?
00197   bool m_useVelocityMap = false;
00198   // Use maps for the Townsend coefficient?
00199   bool m_useTownsendMap = false;
00200
00201   // Flag wether to simulate electron multiplication or not.
00202   bool m_doAvalanche = true;
00203   enum class GainFluctuations { None = 0, Polya };
00204   // Model to be used for randomizing the avalanche size.
00205   GainFluctuations m_gainFluctuations = GainFluctuations::None;
00206   // Polya shape parameter.
00207   double m_theta = 0.;
00208   // Mean avalanche size (only used if > 1).
00209   double m_gain = -1.;
00210
00211   // Flag whether to simulate the ion tail or not.
00212   bool m_doIonTail = true;
00213   // Simulate the ion tail automatically if the medium has mobility data?
00214   bool m_doIonTailAuto = true;
00215   // Flag whether to simulate the negative ion tail or not.
00216   bool m_doNegativeIonTail = false;
00217   // Avalanche size.
00218   double m_nE = 0., m_nI = 0.;
00219
00220   // Debug flag.
00221   bool m_debug = false;
00222
00223   // Calculate a drift line starting at a given position.
00224   bool DriftLine(const std::array<double, 3>& x0, const double t0,
00225                  const Particle particle, std::vector<double>& ts,
00226                  std::vector<std::array<double, 3> >& xs, int& status) const;
00227   // Calculate the number of electrons and ions at each point along a
00228   // drift line.
00229   bool Avalanche(const Particle particle,
00230                  const std::vector<std::array<double, 3> >& xs,
00231                  std::vector<double>& ne, std::vector<double>& ni,
00232                  std::vector<double>& nn, double& scale) const;
00233   // Calculate drift lines and induced signals of the positive ions
00234   // produced in an avalanche.
00235   bool AddIonTail(const std::vector<double>& te,
00236                   const std::vector<std::array<double, 3> >& xe,
00237                   const std::vector<double>& ni, const double scale) const;
00238   // Calculate drift lines and induced signals of the negative ions
00239   // produced by electron attachment.
00240   bool AddNegativeIonTail(const std::vector<double>& te,
00241                           const std::vector<std::array<double, 3> >& xe,
00242                           const std::vector<double>& nn,
00243                           const double scale) const;
00244   // Compute electric and magnetic field at a given position.
00245   int GetField(const std::array<double, 3>& x, double& ex, double& ey,
00246                double& ez, double& bx, double& by, double& bz,
00247                Medium*& medium) const;
00248   // Calculate transport parameters for a given point and particle type.
00249   std::array<double, 3> GetVelocity(const std::array<double, 3>& x,
00250                                     const Particle particle, int& status) const;
00251   bool GetDiffusion(const std::array<double, 3>& x, const Particle particle,
00252                     double& dl, double& dt) const;
00253   double GetVar(const std::array<double, 3>& x, const Particle particle) const;
00254   double GetAlpha(const std::array<double, 3>& x,
00255                   const Particle particle) const;
00256   double GetEta(const std::array<double, 3>& x, const Particle particle) const;
00257
00258   // Terminate a drift line at the edge of a boundary.
00259   bool Terminate(const std::array<double, 3>& xx0,
00260                  const std::array<double, 3>& xx1, const Particle particle,
00261                  std::vector<double>& ts,
00262                  std::vector<std::array<double, 3> >& xs) const;
00263
00264   // Drift a particle to a wire
00265   bool DriftToWire(const double xw, const double yw, const double rw,
00266                    const Particle particle, std::vector<double>& ts,
00267                    std::vector<std::array<double, 3> >& xs, int& stat) const;
00268   // Compute the arrival time spread along a drift line.
```

```
00269   double ComputeSigma(const std::vector<std::array<double, 3> >& x,
00270                       const Particle particle, const double eps) const;
00271   // Compute the gain factor along a drift line.
00272   double ComputeGain(const std::vector<std::array<double, 3> >& x,
00273                      const Particle particle, const double eps) const;
00274   // Compute the loss factor along a drift line.
00275   double ComputeLoss(const std::vector<std::array<double, 3> >& x,
00276                      const Particle particle, const double eps) const;
00277   // Integrate the longitudinal diffusion over a step.
00278   double IntegrateDiffusion(const std::array<double, 3>& xi,
00279                             const std::array<double, 3>& xe,
00280                             const Particle particle, const double tol) const;
00281   // Integrate the Townsend coefficient over a step.
00282   double IntegrateAlpha(const std::array<double, 3>& xi,
00283                         const std::array<double, 3>& xe,
00284                         const Particle particle, const double tol) const;
00285   // Integrate the attachment coefficient over a step.
00286   double IntegrateEta(const std::array<double, 3>& xi,
00287                       const std::array<double, 3>& xe, const Particle particle,
00288                       const double tol) const;
00289
00290   // Calculate the signal for a given drift line.
00291   void ComputeSignal(const Particle particle, const double scale,
00292                      const std::vector<double>& ts,
00293                      const std::vector<std::array<double, 3> >& xs,
00294                      const std::vector<double>& ne) const;
00295
00296   // Terminate a field line.
00297   void Terminate(const std::array<double, 3>& xx0,
00298                  const std::array<double, 3>& xx1,
00299                  std::vector<std::array<float, 3> >& xs) const;
00300
00301   static double Charge(const Particle particle) {
00302     if (particle == Particle::Electron || particle == Particle::NegativeIon) {
00303       return -1.;
00304     }
00305     return 1.;
00306   }
00307 };
00308 }  // namespace Garfield
00309
00310 #endif
```

## 8.53 /builds/garfield/docs/source/Include/Garfield/FundamentalConstants.hh File Reference

Namespaces

- namespace Garfield

## 8.54 FundamentalConstants.hh

Go to the documentation of this file.

```
00001 // Fundamental mathematical and physical constants
00002 // Source: K. Nakamura et al., The Review of Particle Physics,
00003 //         Journal of Physics G 37 075021 (2010)
00004
00005 #ifndef G_FUNDAMENTAL_CONSTANTS_H
00006 #define G_FUNDAMENTAL_CONSTANTS_H
00007
00008 namespace Garfield {
00009
00010 static constexpr double Pi = 3.14159265358979323846264338327950;
00011 static constexpr double TwoPi = 2. * Pi;
00012 static constexpr double HalfPi = 0.5 * Pi;
00013 static constexpr double Pi2 = Pi * Pi;
00014
00015 static constexpr double DegreeToRad = Pi / 180.;
00016 static constexpr double RadToDegree = 180. / Pi;
00017
00019 static constexpr double Gamma = 0.577215664901532861;
00020
00021 static constexpr double CLog2 = 0.693147180559945309417;
00022 static constexpr double Sqrt2 = 1.41421356237309504880168872420970;
00023
00024 // Elementary particle masses [eV / c2]
00025 static constexpr double ElectronMass = 0.510998910e6;
00026 static constexpr double MuonMass = 105.658367e6;
00027 static constexpr double ProtonMass = 938.272013e6;
00028 static constexpr double NeutronMass = 939.56536e6;
00029
00031 static constexpr double FineStructureConstant = 7.2973525376e-3;
```

```
00033 static constexpr double Hbar = 6.58211899e-7;
00035 static constexpr double HbarC = 197.3269631e-7;
00037 static constexpr double ElectronMassGramme = 9.10938215e-28;
00039 static constexpr double AtomicMassUnit = 1.660538782e-24;
00040 static constexpr double AtomicMassUnitElectronVolt = 931.494028e6;
00042 static constexpr double BohrRadius = 0.52917720859e-8;
00044 static constexpr double RydbergEnergy = 13.60569193;
00046 static constexpr double SpeedOfLight = 29.9792458;
00048 static constexpr double ClassicalElectronRadius = 2.8179402894e-13;
00050 static constexpr double ElementaryCharge = 1.602176487e-4;
00052 static constexpr double OmegaCyclotronOverB = 1.758820150e-3;
00054 static constexpr double BoltzmannConstant = 8.617343e-5;
00056 static constexpr double VacuumPermittivity = 88.54187817;
00057 static constexpr double FourPiEpsilon0 = 4. * Pi * VacuumPermittivity;
00058 static constexpr double TwoPiEpsilon0 = TwoPi * VacuumPermittivity;
00060 static constexpr double LoschmidtNumber = 2.6867774e19;
00062 static constexpr double AvogadroConstant = 6.02214179e23;
00064 static constexpr double AtmosphericPressure = 760.;
00066 static constexpr double ZeroCelsius = 273.15;
00067
00069 static constexpr double GravitationalAcceleration = 9.80665;
00070 }  // namespace Garfield
00071
00072 #endif
```

## 8.55 /builds/garfield/docs/source/Include/Garfield/GarfieldConstants.hh File Reference

Namespaces

- namespace Garfield

Enumerations

- enum class Garfield::Particle {
  Garfield::Electron = 0 , Garfield::Ion , Garfield::Hole , Garfield::Positron ,
  Garfield::NegativeIon , Garfield::Photon }

## 8.56 GarfieldConstants.hh

Go to the documentation of this file.

```
00001
00002 #ifndef G__GARFIELD__CONSTANTS__H
00003 #define G__GARFIELD__CONSTANTS__H
00004
00005 namespace Garfield {
00006
00007 enum class Particle { Electron = 0, Ion, Hole, Positron, NegativeIon, Photon };
00008
00009 // Collision types
00010 static const int ElectronCollisionTypeElastic = 0;
00011 static const int ElectronCollisionTypeIonisation = 1;
00012 static const int ElectronCollisionTypeAttachment = 2;
00013 static const int ElectronCollisionTypeInelastic = 3;
00014 static const int ElectronCollisionTypeExcitation = 4;
00015 static const int ElectronCollisionTypeSuperelastic = 5;
00016 static const int ElectronCollisionTypeVirtual = 6;
00017
00018 static const int ElectronCollisionTypeAcousticPhonon = 10;
00019 static const int ElectronCollisionTypeOpticalPhonon = 11;
00020 static const int ElectronCollisionTypeIntervalleyG = 12;
00021 static const int ElectronCollisionTypeIntervalleyF = 13;
00022 static const int ElectronCollisionTypeInterbandXL = 14;
00023 static const int ElectronCollisionTypeInterbandXG = 15;
00024 static const int ElectronCollisionTypeInterbandLG = 16;
00025 static const int ElectronCollisionTypeImpurity = 17;
00026
00027 static const int PhotonCollisionTypeElastic = 0;
00028 static const int PhotonCollisionTypeIonisation = 1;
00029 static const int PhotonCollisionTypeInelastic = 2;
00030 static const int PhotonCollisionTypeExcitation = 3;
00031
00032 // Status codes for drift lines
00033 static const int StatusAlive = 0;
00034 static const int StatusLeftDriftArea = -1;
00035 static const int StatusTooManySteps = -2;
00036 static const int StatusCalculationAbandoned = -3;
00037 static const int StatusLeftDriftMedium = -5;
00038 static const int StatusAttached = -7;
00039 static const int StatusSharpKink = -8;
00040 static const int StatusHitPlane = -11;
```

```
00041 static const int StatusBelowTransportCut = -16;
00042 static const int StatusOutsideTimeWindow = -17;
00043 static const double Small = 1.e-20;
00044
00045 static const double BoundaryDistance = 1.e-8;
00046
00047 // Conversion from Tesla to internal magnetic field units
00048 // compatible with cm and ns.
00049 static const double Tesla2Internal = 1.e5;
00050 }  // namespace Garfield
00051
00052 #endif
```

## 8.57 /builds/garfield/docs/source/Include/Garfield/Geometry.hh File Reference

#include <cstddef>
#include <string>

Classes

- class Garfield::Geometry

  Abstract base class for geometry classes.

Namespaces

- namespace Garfield

## 8.58 Geometry.hh

Go to the documentation of this file.

```
00001 #ifndef G_GEOMETRY_H
00002 #define G_GEOMETRY_H
00003
00004 #include <cstddef>
00005 #include <string>
00006
00007 namespace Garfield {
00008
00009 class Medium;
00010 class Solid;
00011
00013 class Geometry {
00014  public:
00016   Geometry() = delete;
00018   Geometry(const std::string& name) : m_className(name) {}
00020   virtual ~Geometry() {}
00021
00023   virtual Medium* GetMedium(const double x, const double y, const double z,
00024                             const bool tesselated = false) const = 0;
00025
00027   virtual std::size_t GetNumberOfSolids() const { return 0; }
00029   virtual Solid* GetSolid(const std::size_t /*i*/) const { return nullptr; }
00031   virtual Solid* GetSolid(const std::size_t /*i*/, Medium*& /*medium*/) const {
00032     return nullptr;
00033   }
00035   virtual bool IsInside(const double x, const double y, const double z,
00036                         const bool tesselated = false) const = 0;
00037
00039   virtual bool GetBoundingBox(double& xmin, double& ymin, double& zmin,
00040                               double& xmax, double& ymax, double& zmax) = 0;
00041
00042  protected:
00043   std::string m_className = "Geometry";
00044 };
00045 }  // namespace Garfield
00046
00047 #endif
```

## 8.59 /builds/garfield/docs/source/Include/Garfield/GeometryRoot.hh File Reference

#include <TGeoManager.h>
#include <TGeoMaterial.h>
#include <map>

#include ”Garfield/Geometry.hh”

Classes

- class Garfield::GeometryRoot

  Use a geometry defined using the ROOT TGeo package.

Namespaces

- namespace Garfield

## 8.60 GeometryRoot.hh

Go to the documentation of this file.

```
00001 #ifndef G_GEOMETRY_ROOT_H
00002 #define G_GEOMETRY_ROOT_H
00003
00004 #include <TGeoManager.h>
00005 #include <TGeoMaterial.h>
00006
00007 #include <map>
00008
00009 #include ”Garfield/Geometry.hh”
00010
00011 namespace Garfield {
00012
00014
00015 class GeometryRoot : public Geometry {
00016  public:
00018   GeometryRoot();
00020   ~GeometryRoot() {}
00021
00023   void SetGeometry(TGeoManager* geoman);
00024
00025   Medium* GetMedium(const double x, const double y, const double z,
00026                     const bool tesselated = false) const override;
00027
00029   unsigned int GetNumberOfMaterials();
00031   TGeoMaterial* GetMaterial(const unsigned int i);
00033   TGeoMaterial* GetMaterial(const char* name);
00035   void SetMedium(const unsigned int imat, Medium* med);
00037   void SetMedium(const char* mat, Medium* med);
00038
00039   bool IsInside(const double x, const double y, const double z,
00040                 const bool tesselated = false) const override;
00041   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00042                       double& ymax, double& zmax) override;
00043
00045   void EnableDebugging(const bool on = true) { m_debug = on; }
00046
00047  protected:
00048   // ROOT geometry manager
00049   TGeoManager* m_geoManager = nullptr;
00050
00051   // List of ROOT materials associated to Garfield media
00052   std::map<std::string, Medium*> m_materials;
00053
00054   // Switch on/off debugging messages.
00055   bool m_debug = false;
00056   void PrintGeoNotDefined(const std::string& fcn) const;
00057 };
00058 }  // namespace Garfield
00059
00060 #endif
```

## 8.61 /builds/garfield/docs/source/Include/Garfield/GeometrySimple.hh File Reference

#include <array>
#include <vector>
#include ”Garfield/Geometry.hh”

Classes

- class Garfield::GeometrySimple

”Native” geometry, using simple shapes.

Namespaces

- namespace Garfield

## 8.62 GeometrySimple.hh

Go to the documentation of this file.

```
00001 #ifndef G__GEOMETRY__SIMPLE__H
00002 #define G__GEOMETRY__SIMPLE__H
00003
00004 #include <array>
00005 #include <vector>
00006
00007 #include ”Garfield/Geometry.hh”
00008
00009 namespace Garfield {
00010
00012
00013 class GeometrySimple : public Geometry {
00014  public:
00016   GeometrySimple();
00018   virtual ~GeometrySimple() {}
00019
00020   Medium* GetMedium(const double x, const double y, const double z,
00021                     const bool tesselated = false) const override;
00022
00023   size_t GetNumberOfSolids() const override { return m_solids.size(); }
00024   Solid* GetSolid(const size_t i) const override;
00025   Solid* GetSolid(const size_t i, Medium*& medium) const override;
00026
00028   void AddSolid(Solid* s, Medium* m);
00030   Solid* GetSolid(const double x, const double y, const double z,
00031                   const bool tesselated = false) const;
00032
00034   void SetMedium(Medium* medium) { m_medium = medium; }
00035
00037   void Clear();
00039   void PrintSolids();
00040
00041   bool IsInside(const double x, const double y, const double z,
00042                 const bool tesselated = false) const override;
00044   bool IsInBoundingBox(const double x, const double y, const double z) const;
00045   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00046                       double& ymax, double& zmax) override {
00047     xmin = m_bbMin[0];
00048     ymin = m_bbMin[1];
00049     zmin = m_bbMin[2];
00050     xmax = m_bbMax[0];
00051     ymax = m_bbMax[1];
00052     zmax = m_bbMax[2];
00053     return true;
00054   }
00055
00057   void EnableDebugging(const bool on = true) { m_debug = on; }
00058
00059  protected:
00061   std::vector<std::pair<Solid*, Medium*> > m_solids;
00063   Medium* m_medium = nullptr;
00064
00065   // Bounding box ranges
00066   bool m_hasBoundingBox = false;
00067   std::array<double, 3> m_bbMin = {{0., 0., 0.}};
00068   std::array<double, 3> m_bbMax = {{0., 0., 0.}};
00069
00071   bool m_debug = false;
00072 };
00073 }  // namespace Garfield
00074
00075 #endif
```

## 8.63 /builds/garfield/docs/source/Include/Garfield/HelperMacros.hh File Reference

Macros

- #define GARFIELD_CLASS_NAME(name)
- #define ___DEVICE___

### 8.63.1 Macro Definition Documentation

___DEVICE___

#define ___DEVICE___
Definition at line 12 of file HelperMacros.hh.

GARFIELD_CLASS_NAME

#define GARFIELD_CLASS_NAME(
name)
Value:
name
Definition at line 10 of file HelperMacros.hh.

## 8.64 HelperMacros.hh

Go to the documentation of this file.

```
00001 #if defined(GARFIELD_CLASS_NAME)
00002 #undef GARFIELD_CLASS_NAME
00003 #endif
00004
00005 #if defined(___GPUCOMPILE___)
00006 #define GARFIELD_CLASS_NAME(name) name##GPU
00007 #undef ___DEVICE___
00008 #define ___DEVICE___ ___device___
00009 #else
00010 #define GARFIELD_CLASS_NAME(name) name
00011 #undef ___DEVICE___
00012 #define ___DEVICE___
00013 #endif
```

## 8.65 /builds/garfield/docs/source/Include/Garfield/KDTree.hh File Reference

#include <array>
#include <queue>
#include <vector>
#include <cstddef>

Classes

- struct Garfield::KDTreeResult
  Search result.
- class Garfield::KDTree
  Main k-d tree class.
- class Garfield::KDTreeNode
  A node in the k-d tree.

Namespaces

- namespace Garfield

Typedefs

- typedef std::vector< std::vector< double > > Garfield::KDTreeArray

## 8.66 KDTree.hh

Go to the documentation of this file.

```
00001 #ifndef G__KDTREE2__H
00002 #define G__KDTREE2__H
00003
00004 // (c) Matthew B. Kennel, Institute for Nonlinear Science, UCSD (2004)
00005 //
00006 // Licensed under the Academic Free License version 1.1 found in file LICENSE
```

```
00007 // with additional provisions in that same file.
00008
00009 // Implement a kd tree for fast searching of points in a fixed data base
00010 // in k-dimensional Euclidean space.
00011
00012 #include <array>
00013 #include <queue>
00014 #include <vector>
00015 #include <cstddef>
00016
00017 namespace Garfield {
00018
00019 typedef std::vector<std::vector<double> > KDTreeArray;
00020
00021 class KDTreeNode;
00022
00024
00025 struct KDTreeResult {
00026   double dis;  //< square Euclidean distance
00027   std::size_t idx;  //< index
00028 };
00029
00032
00033 class KDTree {
00034  public:
00035   // Reference to the underlying data to be included in the tree.
00036   const KDTreeArray& m_data;
00037
00038   std::size_t m_dim = 3;
00039   bool sort_results = false;
00040
00041  public:
00042   KDTree() = delete;
00044   KDTree(KDTreeArray& data_in);
00046   ~KDTree();
00047
00053   void n_nearest(const std::vector<double>& qv, const unsigned int nn,
00054                  std::vector<KDTreeResult>& result) const;
00055
00063   void n_nearest_around_point(const unsigned int idx,
00064                               const unsigned int ndecorrel,
00065                               const unsigned int nn,
00066                               std::vector<KDTreeResult>& result) const;
00067
00073   void r_nearest(const std::vector<double>& qv, const double r2,
00074                  std::vector<KDTreeResult>& result) const;
00075
00078   void r_nearest_around_point(const unsigned int idx,
00079                               const unsigned int ndecorrel, const double r2,
00080                               std::vector<KDTreeResult>& result) const;
00081
00082   friend class KDTreeNode;
00083
00084  private:
00085   KDTreeNode* m_root = nullptr;
00086
00087   // Index for the tree leaves. Data in a leaf with bounds [l,u] are
00088   // in 'data[ind[l],*] to data[ind[u],*]
00089   std::vector<std::size_t> m_ind;
00090
00091   static constexpr int bucketsize = 12;  // global constant.
00092
00093  private:
00094   KDTreeNode* build_tree_for_range(int l, int u, KDTreeNode* parent);
00095   int select_on_coordinate_value(int c, double alpha, int l, int u);
00096   std::array<double, 2> spread_in_coordinate(const int c, const int l,
00097                                              const int u) const;
00098 };
00099
00101
00102 class KDTreeNode {
00103  public:
00105   KDTreeNode(int dim);
00107   ~KDTreeNode();
00108
00109  private:
00110   friend class KDTree;
00111
00112   // Dimension to cut.
00113   std::size_t cut_dim = 0;
00114   // Cut value.
00115   double cut_val = 0.;
00116   double cut_val_left = 0.;
00117   double cut_val_right = 0.;
00118   // Extents in index array for searching
00119   int m_l = 0;
00120   int m_u = 0;
```

```
00121   // [min,max] of the box enclosing all points
00122   std::vector<std::array<double, 2> > box;
00123
00124   // Pointers to left and right nodes.
00125   KDTreeNode* left = nullptr;
00126   KDTreeNode* right = nullptr;
00127
00128   // Recursive innermost core routines for searching.
00129   void search_n(const int idx0, const int nd, const unsigned int nn, double& r2,
00130                 const std::vector<double>& qv, const KDTree& tree,
00131                 std::priority_queue<KDTreeResult>& res) const;
00132   void search_r(const int idx0, const int nd, const double r2,
00133                 const std::vector<double>& qv, const KDTree& tree,
00134                 std::vector<KDTreeResult>& res) const;
00135
00136   // Return true if the bounding box for this node is within the
00137   // search range around a point.
00138   bool box_in_search_range(const double r2,
00139                            const std::vector<double>& qv) const;
00140
00141   // For processing final buckets.
00142   void process_terminal_node_n(const int idx0, const int nd,
00143                                const unsigned int nn, double& r2,
00144                                const std::vector<double>& qv,
00145                                const KDTree& tree,
00146                                std::priority_queue<KDTreeResult>& res) const;
00147   void process_terminal_node_r(const int idx0, const int nd, const double r2,
00148                                const std::vector<double>& qv,
00149                                const KDTree& tree,
00150                                std::vector<KDTreeResult>& res) const;
00151 };
00152
00153 }  // namespace Garfield
00154
00155 #endif
```

## 8.67 /builds/garfield/docs/source/Include/Garfield/LinkDef.h File Reference

## 8.68 LinkDef.h

Go to the documentation of this file.

```
00001 #ifdef ___CINT___
00002
00003 #pragma link off all globals;
00004 #pragma link off all classes;
00005 #pragma link off all functions;
00006 #pragma link C++ nestedclasses;
00007
00008 #pragma link C++ namespace Garfield;
00009
00010 #pragma link C++ global Garfield::plottingEngine;
00011
00012 #pragma link C++ class Garfield::AvalancheMicroscopic;
00013 #pragma link C++ class Garfield::AvalancheMC;
00014 #pragma link C++ class Garfield::DriftLineRKF;
00015
00016 #pragma link C++ class Garfield::Medium;
00017 #pragma link C++ class Garfield::MediumGas;
00018 #pragma link C++ class Garfield::MediumMagboltz;
00019 #pragma link C++ class Garfield::MediumSilicon;
00020 #pragma link C++ class Garfield::MediumDiamond;
00021 #pragma link C++ class Garfield::MediumGaAs;
00022 #pragma link C++ class Garfield::MediumGaN;
00023 #pragma link C++ class Garfield::MediumCdTe;
00024 #pragma link C++ class Garfield::OpticalData;
00025
00026 #pragma link C++ class Garfield::Solid;
00027 #pragma link C++ class Garfield::SolidBox;
00028 #pragma link C++ class Garfield::SolidHole;
00029 #pragma link C++ class Garfield::SolidExtrusion;
00030 #pragma link C++ class Garfield::SolidRidge;
00031 #pragma link C++ class Garfield::SolidSphere;
00032 #pragma link C++ class Garfield::SolidTube;
00033 #pragma link C++ class Garfield::SolidWire;
00034
00035 #pragma link C++ class Garfield::Geometry;
00036 #pragma link C++ class Garfield::GeometrySimple;
00037 #pragma link C++ class Garfield::GeometryRoot;
00038
00039 #pragma link C++ class Garfield::Component;
00040 #pragma link C++ class Garfield::ComponentAnalyticField;
00041 #pragma link C++ class Garfield::ComponentFieldMap;
00042 #pragma link C++ class Garfield::ComponentAnsys123;
00043 #pragma link C++ class Garfield::ComponentAnsys121;
```

```
00044 #pragma link C++ class Garfield::ComponentCST;
00045 #pragma link C++ class Garfield::ComponentComsol;
00046 #pragma link C++ class Garfield::ComponentElmer;
00047 #pragma link C++ class Garfield::ComponentElmer2d;
00048 #pragma link C++ class Garfield::ComponentConstant;
00049 #pragma link C++ class Garfield::ComponentGrid;
00050 #pragma link C++ class Garfield::ComponentNeBem2d;
00051 #pragma link C++ class Garfield::ComponentNeBem3d;
00052 #pragma link C++ class Garfield::ComponentParallelPlate;
00053 #pragma link C++ class Garfield::ComponentTcad2d;
00054 #pragma link C++ class Garfield::ComponentTcad3d;
00055 #pragma link C++ class Garfield::ComponentUser;
00056
00057 #pragma link C++ class Garfield::Sensor;
00058
00059 #pragma link C++ class Garfield::ViewCell;
00060 #pragma link C++ class Garfield::ViewDrift;
00061 #pragma link C++ class Garfield::ViewFEMesh;
00062 #pragma link C++ class Garfield::ViewField;
00063 #pragma link C++ class Garfield::ViewGeometry;
00064 #pragma link C++ class Garfield::ViewIsochrons;
00065 #pragma link C++ class Garfield::ViewMedium;
00066 #pragma link C++ class Garfield::ViewSignal;
00067
00068 #pragma link C++ class Garfield::Track;
00069 #pragma link C++ class Garfield::TrackHeed;
00070 #pragma link C++ class Garfield::TrackSrim;
00071 #pragma link C++ class Garfield::TrackBichsel;
00072
00073 #pragma link C++ function Garfield::RndmUniform();
00074 #pragma link C++ function Garfield::SetDefaultStyle();
00075 #pragma link C++ function Garfield::SetSerif();
00076 #pragma link C++ function Garfield::SetSansSerif();
00077
00078 #pragma link C++ enum Garfield::Particle;
00079
00080 #endif
```

## 8.69 /builds/garfield/docs/source/Include/Garfield/MagboltzInterface.hh File Reference

#include <cstdint>

### Namespaces

- namespace Garfield
- namespace Garfield::Magboltz

### Functions

- void Garfield::Magboltz::gasmix_ (std::int64_t *ngs, double *q, double *qin, std::int64_t *nin, double *e, double *ei, char *name, double *virl, double *eb, double *peqel, double *peqin, double *penfra, std::int64_t *kel, std::int64_t *kin, double *qion, double *peqion, double *eion, std::int64_t *nion, double *qatt, std::int64_t *natt, double *qnull, std::int64_t *nnull, double *scln, std::int64_t *nc0, double *ec0, double *wk, double *efl, std::int64_t *ng1, double *eg1, std::int64_t *ng2, double *eg2, char scrpt[nMaxLevelsPerComponent][nCharDescr], char scrptn[nMaxNullTerms][nCharDescr], short namelen, short scrpt_len, short scrptn_len)
- void Garfield::Magboltz::colf_ (double *freq, double *freel, double *freion, double *freatt, double *frein, std::int64_t *ntotal)
- void Garfield::Magboltz::colft_ (double *freq, double *freel, double *freion, double *freatt, double *frein, std::int64_t *ntotal)
- void Garfield::Magboltz::magboltz_ ()

### Variables

- constexpr unsigned int Garfield::Magboltz::nEnergySteps = 4000
- constexpr unsigned int Garfield::Magboltz::nMaxIonisationTerms = 30
- constexpr unsigned int Garfield::Magboltz::nMaxInelasticTerms = 250
- constexpr unsigned int Garfield::Magboltz::nMaxAttachmentTerms = 8
- constexpr unsigned int Garfield::Magboltz::nMaxNullTerms = 10
- constexpr unsigned int Garfield::Magboltz::nMaxLevelsPerComponent = 300

- constexpr unsigned int Garfield::Magboltz::nCharName = 25
- constexpr unsigned int Garfield::Magboltz::nCharDescr = 50
- constexpr unsigned int Garfield::Magboltz::nMaxLevels = 960
- constexpr unsigned int Garfield::Magboltz::nMaxComponents = 6
- struct {
  double Garfield::Magboltz::eovb
  double Garfield::Magboltz::wb
  double Garfield::Magboltz::btheta
  double Garfield::Magboltz::bmag
  } Garfield::Magboltz::bfld_

- struct {
  std::int64_t Garfield::Magboltz::nGas
  std::int64_t Garfield::Magboltz::nStep
  std::int64_t Garfield::Magboltz::nAniso
  double Garfield::Magboltz::efinal
  double Garfield::Magboltz::estep
  double Garfield::Magboltz::akt
  double Garfield::Magboltz::ary
  double Garfield::Magboltz::tempc
  double Garfield::Magboltz::torr
  std::int64_t Garfield::Magboltz::ipen
  } Garfield::Magboltz::inpt_

- struct {
  double Garfield::Magboltz::tmax
  double Garfield::Magboltz::small
  double Garfield::Magboltz::api
  double Garfield::Magboltz::estart
  double Garfield::Magboltz::theta
  double Garfield::Magboltz::phi
  double Garfield::Magboltz::rstart
  double Garfield::Magboltz::efield
  std::int64_t Garfield::Magboltz::nmax
  } Garfield::Magboltz::setp_

- struct {
  double Garfield::Magboltz::amgas [6]
  double Garfield::Magboltz::vtmb [6]
  double Garfield::Magboltz::tcfmx
  double Garfield::Magboltz::tcfmxg [6]
  std::int64_t Garfield::Magboltz::ithrm
  } Garfield::Magboltz::thrm_

- struct {
  double Garfield::Magboltz::echarg
  double Garfield::Magboltz::emass
  double Garfield::Magboltz::amu
  double Garfield::Magboltz::pir2
  } Garfield::Magboltz::cnsts_

- struct {
  double Garfield::Magboltz::eg [nEnergySteps]
  double Garfield::Magboltz::eroot [nEnergySteps]
  double Garfield::Magboltz::qt1 [nEnergySteps]
  double Garfield::Magboltz::qt2 [nEnergySteps]
  double Garfield::Magboltz::qt3 [nEnergySteps]
  double Garfield::Magboltz::qt4 [nEnergySteps]

} Garfield::Magboltz::mix2_

- struct {
  double Garfield::Magboltz::den [nEnergySteps]
  } Garfield::Magboltz::dens_

- struct {
  double Garfield::Magboltz::time [300]
  std::int64_t Garfield::Magboltz::icoll [30]
  double Garfield::Magboltz::spec [nEnergySteps]
  double Garfield::Magboltz::tmax1
  double Garfield::Magboltz::ave
  double Garfield::Magboltz::den
  double Garfield::Magboltz::xid
  double Garfield::Magboltz::x
  double Garfield::Magboltz::y
  double Garfield::Magboltz::z
  double Garfield::Magboltz::st
  std::int64_t Garfield::Magboltz::nnull
  std::int64_t Garfield::Magboltz::icoln [nMaxLevels]
  std::int64_t Garfield::Magboltz::icolnn [60]
  } Garfield::Magboltz::outpt_

- struct {
  double Garfield::Magboltz::time [300]
  std::int64_t Garfield::Magboltz::icoll [5][nMaxComponents]
  double Garfield::Magboltz::spec [nEnergySteps]
  double Garfield::Magboltz::tmax1
  double Garfield::Magboltz::ave
  double Garfield::Magboltz::den
  double Garfield::Magboltz::xid
  double Garfield::Magboltz::x
  double Garfield::Magboltz::y
  double Garfield::Magboltz::z
  double Garfield::Magboltz::st
  std::int64_t Garfield::Magboltz::nnull
  std::int64_t Garfield::Magboltz::icoln [290][nMaxComponents]
  std::int64_t Garfield::Magboltz::icolnn [10][nMaxComponents]
  } Garfield::Magboltz::outptt_

- struct {
  char Garfield::Magboltz::dscrpt [nMaxLevels][nCharDescr]
  char Garfield::Magboltz::dscrptn [60][nCharDescr]
  } Garfield::Magboltz::scrip_

- struct {
  char Garfield::Magboltz::dscrpt [nMaxLevelsPerComponent][nMaxComponents][nCharDescr]
  char Garfield::Magboltz::dscrptn [10][nMaxComponents][nCharDescr]
  } Garfield::Magboltz::script_

- struct {
  double Garfield::Magboltz::cf [nMaxLevels][nEnergySteps]
  double Garfield::Magboltz::ein [nMaxLevels]
  double Garfield::Magboltz::tcf [nEnergySteps]
  std::int64_t Garfield::Magboltz::iarry [nMaxLevels]
  double Garfield::Magboltz::rgas [nMaxLevels]
  double Garfield::Magboltz::ipn [nMaxLevels]
  double Garfield::Magboltz::wpl [nMaxLevels]

std::int64_t Garfield::Magboltz::last
std::int64_t Garfield::Magboltz::isize
double Garfield::Magboltz::penfra [nMaxLevels][3]
double Garfield::Magboltz::tcfmax [8]
} Garfield::Magboltz::large_

- struct {
  double Garfield::Magboltz::cf [290][nEnergySteps][nMaxComponents]
  double Garfield::Magboltz::ein [290][nMaxComponents]
  double Garfield::Magboltz::tcf [nEnergySteps][nMaxComponents]
  std::int64_t Garfield::Magboltz::iarry [290][nMaxComponents]
  double Garfield::Magboltz::rgas [290][nMaxComponents]
  double Garfield::Magboltz::ipn [290][nMaxComponents]
  double Garfield::Magboltz::wpl [290][nMaxComponents]
  std::int64_t Garfield::Magboltz::last [nMaxComponents]
  std::int64_t Garfield::Magboltz::isize [nMaxComponents]
  double Garfield::Magboltz::penfra [290][3][nMaxComponents]
  double Garfield::Magboltz::tcfmax [nMaxComponents]
  } Garfield::Magboltz::larget_

- struct {
  std::int64_t Garfield::Magboltz::ngasn [6]
  } Garfield::Magboltz::gasn_

- struct {
  double Garfield::Magboltz::an1
  double Garfield::Magboltz::an2
  double Garfield::Magboltz::an3
  double Garfield::Magboltz::an4
  double Garfield::Magboltz::an5
  double Garfield::Magboltz::an6
  double Garfield::Magboltz::an
  double Garfield::Magboltz::frac [6]
  } Garfield::Magboltz::ratio_

- struct {
  double Garfield::Magboltz::wx
  double Garfield::Magboltz::wy
  double Garfield::Magboltz::wz
  } Garfield::Magboltz::vel_

- struct {
  double Garfield::Magboltz::dwx
  double Garfield::Magboltz::dwy
  double Garfield::Magboltz::dwz
  } Garfield::Magboltz::velerr_

- struct {
  double Garfield::Magboltz::difxx
  double Garfield::Magboltz::difyy
  double Garfield::Magboltz::difzz
  double Garfield::Magboltz::difyz
  double Garfield::Magboltz::difxy
  double Garfield::Magboltz::difxz
  } Garfield::Magboltz::diflab_

- struct {
  double Garfield::Magboltz::dxxer

double Garfield::Magboltz::dyyer
double Garfield::Magboltz::dzzer
double Garfield::Magboltz::dyzer
double Garfield::Magboltz::dxyer
double Garfield::Magboltz::dxzer
} Garfield::Magboltz::diferb_

- struct {
  double Garfield::Magboltz::difln
  double Garfield::Magboltz::diftr
  } Garfield::Magboltz::difvel_

- struct {
  double Garfield::Magboltz::dfler
  double Garfield::Magboltz::dfter
  } Garfield::Magboltz::diferl_

- struct {
  double Garfield::Magboltz::alpha
  double Garfield::Magboltz::att
  } Garfield::Magboltz::ctowns_

- struct {
  double Garfield::Magboltz::alper
  double Garfield::Magboltz::atter
  } Garfield::Magboltz::ctwner_

- struct {
  double Garfield::Magboltz::ralpha
  double Garfield::Magboltz::ralper
  double Garfield::Magboltz::tofene
  double Garfield::Magboltz::tofener
  double Garfield::Magboltz::tofwv
  double Garfield::Magboltz::tofwver
  double Garfield::Magboltz::tofdl
  double Garfield::Magboltz::tofdler
  double Garfield::Magboltz::tofdt
  double Garfield::Magboltz::tofdter
  double Garfield::Magboltz::tofwr
  double Garfield::Magboltz::tofwrer
  double Garfield::Magboltz::rattof
  double Garfield::Magboltz::ratofer
  } Garfield::Magboltz::tofout_

- struct {
  double Garfield::Magboltz::vdout
  double Garfield::Magboltz::vderr
  double Garfield::Magboltz::wsout
  double Garfield::Magboltz::wserr
  double Garfield::Magboltz::dlout
  double Garfield::Magboltz::dlerr
  double Garfield::Magboltz::dtout
  double Garfield::Magboltz::dterr
  double Garfield::Magboltz::alphasst
  double Garfield::Magboltz::alphaerr
  double Garfield::Magboltz::attsst
  double Garfield::Magboltz::atterr
  } Garfield::Magboltz::sstout_

## 8.70 MagboltzInterface.hh

Go to the documentation of this file.

```
00001 // Interface to Magboltz (version 11.19)
00002
00003 #ifndef G__MAGBOLTZ__INTERFACE
00004 #define G__MAGBOLTZ__INTERFACE
00005
00006 #include <cstdint>
00007
00008 #ifndef ___CINT___
00009
00010 namespace Garfield {
00011
00012 namespace Magboltz {
00013
00014 constexpr unsigned int nEnergySteps = 4000;
00015 constexpr unsigned int nMaxIonisationTerms = 30;
00016 constexpr unsigned int nMaxInelasticTerms = 250;
00017 constexpr unsigned int nMaxAttachmentTerms = 8;
00018 constexpr unsigned int nMaxNullTerms = 10;
00019 constexpr unsigned int nMaxLevelsPerComponent = 300;
00020 constexpr unsigned int nCharName = 25;
00021 constexpr unsigned int nCharDescr = 50;
00022 constexpr unsigned int nMaxLevels = 960;
00023 constexpr unsigned int nMaxComponents = 6;
00024
00025 extern "C" {
00026
00027 // Magboltz COMMON blocks
00028
00029 // Magnetic field
00030 extern struct {
00031   double eovb;
00032   double wb;
00033   double btheta, bmag;
00034 } bfld__;
00035
00036 extern struct {
00037   std::int64_t nGas;
00038   std::int64_t nStep;
00039   std::int64_t nAniso;
00040   double efinal;
00041   double estep;
00042   double akt;
00043   double ary;
00044   double tempc;
00045   double torr;
00046   std::int64_t ipen;
00047 } inpt__;
00048
00049 extern struct {
00050   double tmax;
00051   double small;
00052   double api;
00053   double estart;
00054   double theta, phi;
00055   double rstart;
00056   double efield;
00057   std::int64_t nmax;
00058 } setp__;
00059
00060 extern struct {
00061   double amgas[6];
00062   double vtmb[6];
00063   double tcfmx;
00064   double tcfmxg[6];
00065   std::int64_t ithrm;
00066 } thrm__;
00067
00068 // Physical constants
00069 extern struct {
00070   double echarg;
00071   double emass;
00072   double amu;
00073   double pir2;
00074 } cnsts__;
00075
00076 extern struct {
00077   double eg[nEnergySteps];
00078   double eroot[nEnergySteps];
00079   double qt1[nEnergySteps];
00080   double qt2[nEnergySteps];
```

```
00081   double qt3[nEnergySteps];
00082   double qt4[nEnergySteps];
00083 } mix2_;
00084
00085 extern struct {
00086   double den[nEnergySteps];
00087 } dens_;
00088
00089 extern struct {
00090   double time[300];
00091   std::int64_t icoll[30];
00092   double spec[nEnergySteps];
00093   double tmax1;
00094   double ave;
00095   double den;
00096   double xid;
00097   double x;
00098   double y;
00099   double z;
00100   double st;
00101   std::int64_t nnull;
00102   std::int64_t icoln[nMaxLevels];
00103   std::int64_t icolnn[60];
00104 } outpt_;
00105
00106 extern struct {
00107   double time[300];
00108   std::int64_t icoll[5][nMaxComponents];
00109   double spec[nEnergySteps];
00110   double tmax1;
00111   double ave;
00112   double den;
00113   double xid;
00114   double x;
00115   double y;
00116   double z;
00117   double st;
00118   std::int64_t nnull;
00119   std::int64_t icoln[290][nMaxComponents];
00120   std::int64_t icolnn[10][nMaxComponents];
00121 } outptt_;
00122
00123 extern struct {
00124   char dscrpt[nMaxLevels][nCharDescr];
00125   char dscrptn[60][nCharDescr];
00126 } scrip_;
00127
00128 extern struct {
00129   char dscrpt[nMaxLevelsPerComponent][nMaxComponents][nCharDescr];
00130   char dscrptn[10][nMaxComponents][nCharDescr];
00131 } script_;
00132
00133 extern struct {
00134   double cf[nMaxLevels][nEnergySteps];
00135   double ein[nMaxLevels];
00136   double tcf[nEnergySteps];
00137   std::int64_t iarry[nMaxLevels];
00138   double rgas[nMaxLevels];
00139   double ipn[nMaxLevels];
00140   double wpl[nMaxLevels];
00141   std::int64_t last;
00142   std::int64_t isize;
00143   double penfra[nMaxLevels][3];
00144   double tcfmax[8];
00145 } large_;
00146
00147 extern struct {
00148   double cf[290][nEnergySteps][nMaxComponents];
00149   double ein[290][nMaxComponents];
00150   double tcf[nEnergySteps][nMaxComponents];
00151   std::int64_t iarry[290][nMaxComponents];
00152   double rgas[290][nMaxComponents];
00153   double ipn[290][nMaxComponents];
00154   double wpl[290][nMaxComponents];
00155   std::int64_t last[nMaxComponents];
00156   std::int64_t isize[nMaxComponents];
00157   double penfra[290][3][nMaxComponents];
00158   double tcfmax[nMaxComponents];
00159 } larget_;
00160
00161 // Definition of the gas mixture
00162 extern struct {
00163   std::int64_t ngasn[6];
00164 } gasn_;
00165
00166 extern struct {
00167   double an1, an2, an3, an4, an5, an6, an;
```

```
00168   double frac[6];
00169 } ratio_;
00170
00171 // Calculation results
00172 // Drift velocity
00173 extern struct {
00174   double wx, wy, wz;
00175 } vel_;
00176 extern struct {
00177   double dwx, dwy, dwz;
00178 } velerr_;
00179
00180 // Diffusion
00181 extern struct {
00182   double difxx, difyy, difzz;
00183   double difyz, difxy, difxz;
00184 } diflab_;
00185 extern struct {
00186   double dxxer, dyyer, dzzer;
00187   double dyzer, dxyer, dxzer;
00188 } diferb_;
00189 extern struct {
00190   double difln, diftr;
00191 } difvel_;
00192 extern struct {
00193   double dfler, dfter;
00194 } diferl_;
00195
00196 // Townsend and attachment coefficient
00197 extern struct {
00198   double alpha, att;
00199 } ctowns_;
00200 extern struct {
00201   double alper, atter;
00202 } ctwner_;
00203
00204 // Time of flight swarm parameters
00205 extern struct {
00206   double ralpha, ralper;
00207   double tofene, tofener, tofwv, tofwver;
00208   double tofdl, tofdler, tofdt, tofdter;
00209   double tofwr, tofwrer;
00210   double rattof, ratofer;
00211 } tofout_;
00212
00213 // Steady state townsend swarm parameters
00214 extern struct {
00215   double vdout, vderr;              //< diffusion modified drift
00216   double wsout, wserr, dlout, dlerr;  //< seems to be WS SST velocities i.e.
00217                                     // mean arrival time drift velocity?
00218   double dtout, dterr, alphasst, alphaerr;
00219   double attsst, atterr;
00220 } sstout_;
00221
00222 void gasmix_(std::int64_t* ngs, double* q, double* qin, std::int64_t* nin,
00223              double* e, double* ei, char* name, double* virl, double* eb,
00224              double* peqel, double* peqin, double* penfra, std::int64_t* kel,
00225              std::int64_t* kin, double* qion, double* peqion, double* eion,
00226              std::int64_t* nion, double* qatt, std::int64_t* natt,
00227              double* qnull, std::int64_t* nnull, double* scln,
00228              std::int64_t* nc0, double* ec0, double* wk, double* efl,
00229              std::int64_t* ng1, double* eg1, std::int64_t* ng2, double* eg2,
00230              char scrpt[nMaxLevelsPerComponent][nCharDescr],
00231              char scrptn[nMaxNullTerms][nCharDescr], short namelen,
00232              short scrpt_len, short scrptn_len);
00233
00234 void colf_(double* freq, double* freel, double* freion, double* freatt,
00235            double* frein, std::int64_t* ntotal);
00236
00237 void colft_(double* freq, double* freel, double* freion, double* freatt,
00238             double* frein, std::int64_t* ntotal);
00239
00240 void magboltz_();
00241 }
00242 }  // namespace Magboltz
00243 }  // namespace Garfield
00244 #endif
00245 #endif
```

## 8.71 /builds/garfield/docs/source/Include/Garfield/Medium.hh File Reference

#include <string>
#include <vector>
#include "Garfield/FundamentalConstants.hh"

#include ”Garfield/GarfieldConstants.hh”
#include ”Garfield/HelperMacros.hh”

Namespaces

- namespace Garfield

Functions

- class Garfield::GARFIELD_CLASS_NAME (Medium)

  Abstract base class for components.

## 8.72 Medium.hh

Go to the documentation of this file.

```
00001 // Include this header if we're compiling with the GPU or this is the first time
00002 // without
00003 #if defined(___GPUCOMPILE___) || !defined(G_MEDIUM_H)
00004
00005 #if !defined(___GPUCOMPILE___) && !defined(G_MEDIUM_H)
00006 #define G_MEDIUM_H
00007 #endif
00008
00009 #ifdef ___GPUCOMPILE___
00010 #include ”GPUInterface.hh”
00011 #else
00012
00013 #include <string>
00014 #include <vector>
00015
00016 #include ”Garfield/FundamentalConstants.hh”
00017 #endif
00018
00019 #include ”Garfield/GarfieldConstants.hh”
00020 #include ”Garfield/HelperMacros.hh”
00021
00022 class TPad;
00023
00024 namespace Garfield {
00025
00026 // setup class names depending on if this is compiling the GPU static version or
00027 // not
00028 #ifdef ___GPUCOMPILE___
00029 #else
00030 class MediumGPU;
00031 #endif
00032
00034
00035 class GARFIELD_CLASS_NAME(Medium) {
00036  public:
00037 #ifdef ___GPUCOMPILE___
00039   GARFIELD_CLASS_NAME(Medium)() = default;
00041   ~GARFIELD_CLASS_NAME(Medium)() {};
00042 #else
00044   GARFIELD_CLASS_NAME(Medium)();
00046   virtual ~GARFIELD_CLASS_NAME(Medium)();
00047 #endif
00048
00050   ___DEVICE___ int GetId() const { return m_id; }
00051
00052 #ifndef ___GPUCOMPILE___
00054   const std::string& GetName() const { return m_name; }
00056   virtual bool IsGas() const { return false; }
00058   virtual bool IsSemiconductor() const { return false; }
00060   virtual bool IsConductor() const { return false; }
00061
00063   void SetTemperature(const double t);
00065   double GetTemperature() const { return m_temperature; }
00066   // Set the pressure [Torr].
00067   void SetPressure(const double p);
00068   // Get the pressure [Torr].
00069   double GetPressure() const { return m_pressure; }
00071   void SetDielectricConstant(const double eps);
00073   double GetDielectricConstant() const { return m_epsilon; }
00074
00076   unsigned int GetNumberOfComponents() const { return m_nComponents; }
00078   virtual void GetComponent(const unsigned int i, std::string& label,
00079                             double& f);
00081   virtual void SetAtomicNumber(const double z);
```

```
00083   virtual double GetAtomicNumber() const { return m_z; }
00085   virtual void SetAtomicWeight(const double a);
00087   virtual double GetAtomicWeight() const { return m_a; }
00089   virtual void SetNumberDensity(const double n);
00091   virtual double GetNumberDensity() const { return m_density; }
00093   virtual void SetMassDensity(const double rho);
00095   virtual double GetMassDensity() const;
00096
00098   virtual void EnableDrift(const bool on = true) { m_driftable = on; }
00100   virtual void EnablePrimaryIonisation(const bool on = true) {
00101     m_ionisable = on;
00102   }
00103 #endif
00104
00106   ___DEVICE___ bool IsDriftable() const { return m_driftable; }
00108   ___DEVICE___ bool IsMicroscopic() const { return m_microscopic; }
00109
00110 #ifndef ___GPUCOMPILE___
00112   bool IsIonisable() const { return m_ionisable; }
00113
00115   void SetW(const double w) { m_w = w; }
00117   double GetW() const { return m_w; }
00119   void SetFanoFactor(const double f) { m_fano = f; }
00121   double GetFanoFactor() const { return m_fano; }
00122
00124   void PlotVelocity(const std::string& carriers, TPad* pad);
00126   void PlotDiffusion(const std::string& carriers, TPad* pad);
00128   void PlotTownsend(const std::string& carriers, TPad* pad);
00130   void PlotAttachment(const std::string& carriers, TPad* pad);
00132   void PlotAlphaEta(const std::string& carriers, TPad* pad);
00133
00134   // Transport parameters for electrons
00136   virtual bool ElectronVelocity(const double ex, const double ey,
00137                                 const double ez, const double bx,
00138                                 const double by, const double bz, double& vx,
00139                                 double& vy, double& vz);
00142   virtual bool ElectronVelocityFluxBulk(const double ex, const double ey,
00143                                         const double ez, const double bx,
00144                                         const double by, const double bz,
00145                                         double& wv, double& wr);
00147   virtual bool ElectronDiffusion(const double ex, const double ey,
00148                                  const double ez, const double bx,
00149                                  const double by, const double bz, double& dl,
00150                                  double& dt);
00154   virtual bool ElectronDiffusion(const double ex, const double ey,
00155                                  const double ez, const double bx,
00156                                  const double by, const double bz,
00157                                  double cov[3][3]);
00159   virtual bool ElectronTownsend(const double ex, const double ey,
00160                                 const double ez, const double bx,
00161                                 const double by, const double bz,
00162                                 double& alpha);
00164   virtual bool ElectronAttachment(const double ex, const double ey,
00165                                   const double ez, const double bx,
00166                                   const double by, const double bz,
00167                                   double& eta);
00169   virtual bool ElectronTOFIonisation(const double ex, const double ey,
00170                                      const double ez, const double bx,
00171                                      const double by, const double bz,
00172                                      double& riontof);
00174   virtual bool ElectronTOFAttachment(const double ex, const double ey,
00175                                      const double ez, const double bx,
00176                                      const double by, const double bz,
00177                                      double& ratttof);
00179   virtual bool ElectronLorentzAngle(const double ex, const double ey,
00180                                     const double ez, const double bx,
00181                                     const double by, const double bz,
00182                                     double& lor);
00184   virtual double ElectronMobility();
00185   // Microscopic electron transport properties
00186
00188   virtual double GetElectronEnergy(const double px, const double py,
00189                                    const double pz, double& vx, double& vy,
00190                                    double& vz, const int band = 0);
00193   virtual void GetElectronMomentum(const double e, double& px, double& py,
00194                                    double& pz, int& band);
00195
00197   virtual double GetElectronNullCollisionRate(const int band = 0);
00198 #endif
00199
00200 #ifdef ___GPUCOMPILE___
00201
00202   ___device___ cuda_t GetElectronCollisionRate(const cuda_t e, const int band);
00203
00204   ___device___ bool ElectronCollision(const cuda_t e, int& type, int& level,
00205                                       cuda_t& e1, cuda_t& dx, cuda_t& dy,
00206                                       cuda_t& dz, Particle* secondaries_type,
```

```
00207                                 cuda_t* secondaries_energy,
00208                                 int& num_secondaries, int& ndxc, int& band);
00209 #else
00211   virtual double GetElectronCollisionRate(const double e, const int band = 0);
00212   struct Secondary {
00213     Particle type = Particle::Electron;
00214     double energy = 0.;
00215     double time = 0.;
00216     double distance = 0.;
00217   };
00219   virtual bool ElectronCollision(const double e, int& type, int& level,
00220                                  double& e1, double& dx, double& dy, double& dz,
00221                                  std::vector<Secondary>& secondaries,
00222                                  int& band);
00223 #endif
00224
00225 #ifndef ___GPUCOMPILE___
00226   // Transport parameters for holes
00228   virtual bool HoleVelocity(const double ex, const double ey, const double ez,
00229                             const double bx, const double by, const double bz,
00230                             double& vx, double& vy, double& vz);
00232   virtual bool HoleDiffusion(const double ex, const double ey, const double ez,
00233                              const double bx, const double by, const double bz,
00234                              double& dl, double& dt);
00236   virtual bool HoleDiffusion(const double ex, const double ey, const double ez,
00237                              const double bx, const double by, const double bz,
00238                              double cov[3][3]);
00240   virtual bool HoleTownsend(const double ex, const double ey, const double ez,
00241                             const double bx, const double by, const double bz,
00242                             double& alpha);
00244   virtual bool HoleAttachment(const double ex, const double ey, const double ez,
00245                               const double bx, const double by, const double bz,
00246                               double& eta);
00248   virtual double HoleMobility();
00249
00250   // Transport parameters for ions
00252   virtual bool IonVelocity(const double ex, const double ey, const double ez,
00253                            const double bx, const double by, const double bz,
00254                            double& vx, double& vy, double& vz);
00255   bool HasIonVelocity() const { return !(m_iVel.empty() && m_iMob.empty()); }
00257   virtual bool IonDiffusion(const double ex, const double ey, const double ez,
00258                             const double bx, const double by, const double bz,
00259                             double& dl, double& dt);
00261   virtual bool IonDissociation(const double ex, const double ey,
00262                                const double ez, const double bx,
00263                                const double by, const double bz, double& diss);
00265   virtual double IonMobility();
00266
00268   virtual bool NegativeIonVelocity(const double ex, const double ey,
00269                                    const double ez, const double bx,
00270                                    const double by, const double bz, double& vx,
00271                                    double& vy, double& vz);
00273   virtual double NegativeIonMobility();
00274
00276   void SetFieldGrid(double emin, double emax, const size_t ne, bool logE,
00277                     double bmin = 0., double bmax = 0., const size_t nb = 1,
00278                     double amin = HalfPi, double amax = HalfPi,
00279                     const size_t na = 1);
00281   void SetFieldGrid(const std::vector<double>& efields,
00282                     const std::vector<double>& bfields,
00283                     const std::vector<double>& angles);
00285   void GetFieldGrid(std::vector<double>& efields, std::vector<double>& bfields,
00286                     std::vector<double>& angles);
00287
00289   bool SetElectronVelocityE(const size_t ie, const size_t ib, const size_t ia,
00290                             const double v) {
00291     return SetEntry(ie, ib, ia, "ElectronVelocityE", m_eVelE, v);
00292   }
00294   bool GetElectronVelocityE(const size_t ie, const size_t ib, const size_t ia,
00295                             double& v) {
00296     return GetEntry(ie, ib, ia, "ElectronVelocityE", m_eVelE, v);
00297   }
00299   bool SetElectronVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00300                               const double v) {
00301     return SetEntry(ie, ib, ia, "ElectronVelocityExB", m_eVelX, v);
00302   }
00304   bool GetElectronVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00305                               double& v) {
00306     return GetEntry(ie, ib, ia, "ElectronVelocityExB", m_eVelX, v);
00307   }
00309   bool SetElectronVelocityB(const size_t ie, const size_t ib, const size_t ia,
00310                             const double v) {
00311     return SetEntry(ie, ib, ia, "ElectronVelocityB", m_eVelB, v);
00312   }
00314   bool GetElectronVelocityB(const size_t ie, const size_t ib, const size_t ia,
00315                             double& v) {
00316     return GetEntry(ie, ib, ia, "ElectronVelocityB", m_eVelB, v);
```

```
00317 }
00319 bool SetElectronFluxVelocity(const size_t ie, const size_t ib,
00320                               const size_t ia, const double v) {
00321   return SetEntry(ie, ib, ia, "ElectronFluxVelocity", m_eVelWv, v);
00322 }
00324 bool GetElectronFluxVelocity(const size_t ie, const size_t ib,
00325                               const size_t ia, double& v) {
00326   return GetEntry(ie, ib, ia, "ElectronFluxVelocity", m_eVelWv, v);
00327 }
00329 bool SetElectronBulkVelocity(const size_t ie, const size_t ib,
00330                               const size_t ia, const double v) {
00331   return SetEntry(ie, ib, ia, "ElectronBulkVelocity", m_eVelWr, v);
00332 }
00334 bool GetElectronBulkVelocity(const size_t ie, const size_t ib,
00335                               const size_t ia, double& v) {
00336   return GetEntry(ie, ib, ia, "ElectronBulkVelocity", m_eVelWr, v);
00337 }
00339 bool SetElectronLongitudinalDiffusion(const size_t ie, const size_t ib,
00340                                       const size_t ia, const double dl) {
00341   return SetEntry(ie, ib, ia, "ElectronLongitudinalDiffusion", m_eDifL, dl);
00342 }
00344 bool GetElectronLongitudinalDiffusion(const size_t ie, const size_t ib,
00345                                       const size_t ia, double& dl) {
00346   return GetEntry(ie, ib, ia, "ElectronLongitudinalDiffusion", m_eDifL, dl);
00347 }
00349 bool SetElectronTransverseDiffusion(const size_t ie, const size_t ib,
00350                                     const size_t ia, const double dt) {
00351   return SetEntry(ie, ib, ia, "ElectronTransverseDiffusion", m_eDifT, dt);
00352 }
00354 bool GetElectronTransverseDiffusion(const size_t ie, const size_t ib,
00355                                     const size_t ia, double& dt) {
00356   return GetEntry(ie, ib, ia, "ElectronTransverseDiffusion", m_eDifT, dt);
00357 }
00359 bool SetElectronTownsend(const size_t ie, const size_t ib, const size_t ia,
00360                          const double alpha) {
00361   return SetEntry(ie, ib, ia, "ElectronTownsend", m_eAlp, alpha);
00362 }
00364 bool GetElectronTownsend(const size_t ie, const size_t ib, const size_t ia,
00365                          double& alpha) {
00366   return GetEntry(ie, ib, ia, "ElectronTownsend", m_eAlp, alpha);
00367 }
00369 bool SetElectronAttachment(const size_t ie, const size_t ib, const size_t ia,
00370                            const double eta) {
00371   return SetEntry(ie, ib, ia, "ElectronAttachment", m_eAtt, eta);
00372 }
00374 bool GetElectronAttachment(const size_t ie, const size_t ib, const size_t ia,
00375                            double& eta) {
00376   return GetEntry(ie, ib, ia, "ElectronAttachment", m_eAtt, eta);
00377 }
00379 bool SetElectronTOFIonisation(const size_t ie, const size_t ib,
00380                               const size_t ia, const double v) {
00381   return SetEntry(ie, ib, ia, "ElectronTOFIonisation", m_eRIon, v);
00382 }
00384 bool GetElectronTOFIonisation(const size_t ie, const size_t ib,
00385                               const size_t ia, double& v) {
00386   return GetEntry(ie, ib, ia, "ElectronTOFIonisation", m_eRIon, v);
00387 }
00389 bool SetElectronTOFAttachment(const size_t ie, const size_t ib,
00390                               const size_t ia, const double v) {
00391   return SetEntry(ie, ib, ia, "ElectronTOFAttachment", m_eRAtt, v);
00392 }
00394 bool GetElectronTOFAttachment(const size_t ie, const size_t ib,
00395                               const size_t ia, double& v) {
00396   return GetEntry(ie, ib, ia, "ElectronTOFAttachment", m_eRAtt, v);
00397 }
00398
00400 bool SetElectronLorentzAngle(const size_t ie, const size_t ib,
00401                              const size_t ia, const double lor) {
00402   return SetEntry(ie, ib, ia, "ElectronLorentzAngle", m_eLor, lor);
00403 }
00405 bool GetElectronLorentzAngle(const size_t ie, const size_t ib,
00406                              const size_t ia, double& lor) {
00407   return GetEntry(ie, ib, ia, "ElectronLorentzAngle", m_eLor, lor);
00408 }
00409
00411 bool SetHoleVelocityE(const size_t ie, const size_t ib, const size_t ia,
00412                       const double v) {
00413   return SetEntry(ie, ib, ia, "HoleVelocityE", m_hVelE, v);
00414 }
00416 bool GetHoleVelocityE(const size_t ie, const size_t ib, const size_t ia,
00417                       double& v) {
00418   return GetEntry(ie, ib, ia, "HoleVelocityE", m_hVelE, v);
00419 }
00421 bool SetHoleVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00422                         const double v) {
00423   return SetEntry(ie, ib, ia, "HoleVelocityExB", m_hVelX, v);
00424 }
```

```
00426 bool GetHoleVelocityExB(const size_t ie, const size_t ib, const size_t ia,
00427                         double& v) {
00428   return GetEntry(ie, ib, ia, "HoleVelocityExB", m_hVelX, v);
00429 }
00431 bool SetHoleVelocityB(const size_t ie, const size_t ib, const size_t ia,
00432                       const double v) {
00433   return SetEntry(ie, ib, ia, "HoleVelocityB", m_hVelB, v);
00434 }
00436 bool GetHoleVelocityB(const size_t ie, const size_t ib, const size_t ia,
00437                       double& v) {
00438   return GetEntry(ie, ib, ia, "HoleVelocityB", m_hVelB, v);
00439 }
00440
00442 bool SetHoleLongitudinalDiffusion(const size_t ie, const size_t ib,
00443                                   const size_t ia, const double dl) {
00444   return SetEntry(ie, ib, ia, "HoleLongitudinalDiffusion", m_hDifL, dl);
00445 }
00447 bool GetHoleLongitudinalDiffusion(const size_t ie, const size_t ib,
00448                                   const size_t ia, double& dl) {
00449   return GetEntry(ie, ib, ia, "HoleLongitudinalDiffusion", m_hDifL, dl);
00450 }
00452 bool SetHoleTransverseDiffusion(const size_t ie, const size_t ib,
00453                                 const size_t ia, const double dt) {
00454   return SetEntry(ie, ib, ia, "HoleTransverseDiffusion", m_hDifT, dt);
00455 }
00457 bool GetHoleTransverseDiffusion(const size_t ie, const size_t ib,
00458                                 const size_t ia, double& dt) {
00459   return GetEntry(ie, ib, ia, "HoleTransverseDiffusion", m_hDifT, dt);
00460 }
00462 bool SetHoleTownsend(const size_t ie, const size_t ib, const size_t ia,
00463                      const double alpha) {
00464   return SetEntry(ie, ib, ia, "HoleTownsend", m_hAlp, alpha);
00465 }
00467 bool GetHoleTownsend(const size_t ie, const size_t ib, const size_t ia,
00468                      double& alpha) {
00469   return GetEntry(ie, ib, ia, "HoleTownsend", m_hAlp, alpha);
00470 }
00472 bool SetHoleAttachment(const size_t ie, const size_t ib, const size_t ia,
00473                        const double eta) {
00474   return SetEntry(ie, ib, ia, "HoleAttachment", m_hAtt, eta);
00475 }
00477 bool GetHoleAttachment(const size_t ie, const size_t ib, const size_t ia,
00478                        double& eta) {
00479   return GetEntry(ie, ib, ia, "HoleAttachment", m_hAtt, eta);
00480 }
00481
00486 bool SetIonMobility(const std::vector<double>& fields,
00487                     const std::vector<double>& mobilities,
00488                     const bool negativeIons = false);
00490 bool SetIonMobility(const size_t ie, const size_t ib, const size_t ia,
00491                     const double mu);
00493 bool GetIonMobility(const size_t ie, const size_t ib, const size_t ia,
00494                     double& mu) {
00495   return GetEntry(ie, ib, ia, "IonMobility", m_iMob, mu);
00496 }
00497
00499 bool SetIonLongitudinalDiffusion(const size_t ie, const size_t ib,
00500                                  const size_t ia, const double dl) {
00501   return SetEntry(ie, ib, ia, "IonLongitudinalDiffusion", m_iDifL, dl);
00502 }
00504 bool GetIonLongitudinalDiffusion(const size_t ie, const size_t ib,
00505                                  const size_t ia, double& dl) {
00506   return GetEntry(ie, ib, ia, "IonLongitudinalDiffusion", m_iDifL, dl);
00507 }
00509 bool SetIonTransverseDiffusion(const size_t ie, const size_t ib,
00510                                const size_t ia, const double dt) {
00511   return SetEntry(ie, ib, ia, "IonTransverseDiffusion", m_iDifT, dt);
00512 }
00514 bool GetIonTransverseDiffusion(const size_t ie, const size_t ib,
00515                                const size_t ia, double& dt) {
00516   return GetEntry(ie, ib, ia, "IonTransverseDiffusion", m_iDifT, dt);
00517 }
00519 bool SetIonDissociation(const size_t ie, const size_t ib, const size_t ia,
00520                         const double diss) {
00521   return SetEntry(ie, ib, ia, "IonDissociation", m_iDis, diss);
00522 }
00524 bool GetIonDissociation(const size_t ie, const size_t ib, const size_t ia,
00525                         double& diss) {
00526   return GetEntry(ie, ib, ia, "IonDissociation", m_iDis, diss);
00527 }
00528
00530 bool SetNegativeIonMobility(const size_t ie, const size_t ib, const size_t ia,
00531                             const double mu);
00533 bool GetNegativeIonMobility(const size_t ie, const size_t ib, const size_t ia,
00534                             double& mu) {
00535   return GetEntry(ie, ib, ia, "NegativeIonMobility", m_nMob, mu);
00536 }
```

```
00537
00539   virtual void ResetTables();
00540
00541   void ResetElectronVelocity() {
00542     m_eVelE.clear();
00543     m_eVelB.clear();
00544     m_eVelX.clear();
00545     m_eVelWv.clear();
00546     m_eVelWr.clear();
00547   }
00548   void ResetElectronDiffusion() {
00549     m_eDifL.clear();
00550     m_eDifT.clear();
00551     m_eDifM.clear();
00552   }
00553   void ResetElectronTownsend() { m_eAlp.clear(); }
00554   void ResetElectronAttachment() { m_eAtt.clear(); }
00555   void ResetElectronTOFRates() {
00556     m_eRIon.clear();
00557     m_eRAtt.clear();
00558   }
00559   void ResetElectronLorentzAngle() { m_eLor.clear(); }
00560
00561   void ResetHoleVelocity() {
00562     m_hVelE.clear();
00563     m_hVelB.clear();
00564     m_hVelX.clear();
00565   }
00566   void ResetHoleDiffusion() {
00567     m_hDifL.clear();
00568     m_hDifT.clear();
00569     m_hDifM.clear();
00570   }
00571   void ResetHoleTownsend() { m_hAlp.clear(); }
00572   void ResetHoleAttachment() { m_hAtt.clear(); }
00573
00574   void ResetIonMobility() {
00575     m_iMob.clear();
00576     m_iVel.clear();
00577   }
00578   void ResetIonDiffusion() {
00579     m_iDifL.clear();
00580     m_iDifT.clear();
00581   }
00582   void ResetIonDissociation() { m_iDis.clear(); }
00583   void ResetNegativeIonMobility() {
00584     m_nMob.clear();
00585     m_nVel.clear();
00586   }
00587
00588   void VelocityFromMobility(
00589       const std::vector<std::vector<std::vector<double> > >& mob,
00590       std::vector<std::vector<std::vector<double> > >& vel);
00591
00594   void SetExtrapolationMethodVelocity(const std::string& extrLow,
00595                                       const std::string& extrHigh);
00596   void SetExtrapolationMethodDiffusion(const std::string& extrLow,
00597                                        const std::string& extrHigh);
00598   void SetExtrapolationMethodTownsend(const std::string& extrLow,
00599                                       const std::string& extrHigh);
00600   void SetExtrapolationMethodAttachment(const std::string& extrLow,
00601                                         const std::string& extrHigh);
00602   void SetExtrapolationMethodIonMobility(const std::string& extrLow,
00603                                          const std::string& extrHigh);
00604   void SetExtrapolationMethodIonDissociation(const std::string& extrLow,
00605                                              const std::string& extrHigh);
00606
00608   void SetInterpolationMethodVelocity(const unsigned int intrp);
00609   void SetInterpolationMethodDiffusion(const unsigned int intrp);
00610   void SetInterpolationMethodTownsend(const unsigned int intrp);
00611   void SetInterpolationMethodAttachment(const unsigned int intrp);
00612   void SetInterpolationMethodIonMobility(const unsigned int intrp);
00613   void SetInterpolationMethodIonDissociation(const unsigned int intrp);
00614
00615   // Scaling of fields and transport parameters.
00616   virtual double ScaleElectricField(const double e) const { return e; }
00617   virtual double UnScaleElectricField(const double e) const { return e; }
00618   virtual double ScaleVelocity(const double v) const { return v; }
00619   virtual double ScaleDiffusion(const double d) const { return d; }
00620   virtual double ScaleDiffusionTensor(const double d) const { return d; }
00621   virtual double ScaleTownsend(const double alpha) const { return alpha; }
00622   virtual double ScaleAttachment(const double eta) const { return eta; }
00623   virtual double ScaleLorentzAngle(const double lor) const { return lor; }
00624   virtual double ScaleDissociation(const double diss) const { return diss; }
00625
00626   // Optical properties
00628   virtual bool GetOpticalDataRange(double& emin, double& emax,
```

```
00629                                 const unsigned int i = 0);
00631   virtual bool GetDielectricFunction(const double e, double& eps1, double& eps2,
00632                                      const unsigned int i = 0);
00633   // Get the photoabsorption cross-section [cm2] at a given energy.
00634   virtual bool GetPhotoAbsorptionCrossSection(const double e, double& sigma,
00635                                               const unsigned int i = 0);
00636   virtual double GetPhotonCollisionRate(const double e);
00637   virtual bool PhotonCollision(const double e, int& type, int& level,
00638                                double& e1, double& ctheta,
00639                                std::vector<Secondary>& secondaries);
00640
00642   void EnableDebugging() { m_debug = true; }
00643   void DisableDebugging() { m_debug = false; }
00644
00646   virtual double CreateGPUTransferObject(MediumGPU*& med_gpu);
00647
00648  protected:
00649   std::string m_className = "Medium";
00650
00651   static int m_idCounter;
00652
00653   // Number of components
00654   unsigned int m_nComponents = 1;
00655   // Name
00656   std::string m_name = "";
00657   // Temperature [K]
00658   double m_temperature = 293.15;
00659   // Pressure [Torr]
00660   double m_pressure = 760.;
00661   // Static dielectric constant
00662   double m_epsilon = 1.;
00663   // (Effective) atomic number Z
00664   double m_z = 1.;
00665   // Atomic weight A
00666   double m_a = 0.;
00667   // Number density [cm-3]
00668   double m_density = 0.;
00669 #endif
00670
00671   // Id number
00672   int m_id;
00673
00674   // Transport flags
00675   bool m_driftable = false;
00676   bool m_microscopic = false;
00677   bool m_ionisable = false;
00678
00679 #ifndef ___GPUCOMPILE___
00680
00681   // W value
00682   double m_w = 0.;
00683   // Fano factor
00684   double m_fano = 0.;
00685
00686   // Update flag
00687   bool m_isChanged = true;
00688
00689   // Switch on/off debugging messages
00690   bool m_debug = false;
00691
00692   // Tables of transport parameters
00693   bool m_tab2d = false;
00694
00695   // Field grids
00696   std::vector<double> m_eFields;
00697   std::vector<double> m_bFields;
00698   std::vector<double> m_bAngles;
00699
00700   // Electrons
00701   std::vector<std::vector<std::vector<double> > > m_eVelE;
00702   std::vector<std::vector<std::vector<double> > > m_eVelX;
00703   std::vector<std::vector<std::vector<double> > > m_eVelB;
00704   std::vector<std::vector<std::vector<double> > > m_eDifL;
00705   std::vector<std::vector<std::vector<double> > > m_eDifT;
00706   std::vector<std::vector<std::vector<double> > > m_eAlp;
00707   std::vector<std::vector<std::vector<double> > > m_eAtt;
00708   std::vector<std::vector<std::vector<double> > > m_eLor;
00709   std::vector<std::vector<std::vector<double> > > m_eVelWv;
00710   std::vector<std::vector<std::vector<double> > > m_eVelWr;
00711   std::vector<std::vector<std::vector<double> > > m_eRIon;
00712   std::vector<std::vector<std::vector<double> > > m_eRAtt;
00713
00714   std::vector<std::vector<std::vector<std::vector<double> > > > m_eDifM;
00715
00716   // Holes
00717   std::vector<std::vector<std::vector<double> > > m_hVelE;
00718   std::vector<std::vector<std::vector<double> > > m_hVelX;
```

```
00719   std::vector<std::vector<std::vector<double> > > m_hVelB;
00720   std::vector<std::vector<std::vector<double> > > m_hDifL;
00721   std::vector<std::vector<std::vector<double> > > m_hDifT;
00722   std::vector<std::vector<std::vector<double> > > m_hAlp;
00723   std::vector<std::vector<std::vector<double> > > m_hAtt;
00724
00725   std::vector<std::vector<std::vector<std::vector<double> > > > m_hDifM;
00726
00727   // Ions
00728   std::vector<std::vector<std::vector<double> > > m_iMob;
00729   std::vector<std::vector<std::vector<double> > > m_iVel;
00730   std::vector<std::vector<std::vector<double> > > m_iDifL;
00731   std::vector<std::vector<std::vector<double> > > m_iDifT;
00732   std::vector<std::vector<std::vector<double> > > m_iDis;
00733   // Negative ions
00734   std::vector<std::vector<std::vector<double> > > m_nMob;
00735   std::vector<std::vector<std::vector<double> > > m_nVel;
00736
00737   // Thresholds for Townsend, attachment and dissociation coefficients.
00738   unsigned int m_eThrAlp = 0;
00739   unsigned int m_eThrAtt = 0;
00740   unsigned int m_hThrAlp = 0;
00741   unsigned int m_hThrAtt = 0;
00742   unsigned int m_iThrDis = 0;
00743
00744   // Extrapolation methods (TODO: enum).
00745   std::pair<unsigned int, unsigned int> m_extrVel = {0, 1};
00746   std::pair<unsigned int, unsigned int> m_extrDif = {0, 1};
00747   std::pair<unsigned int, unsigned int> m_extrAlp = {0, 1};
00748   std::pair<unsigned int, unsigned int> m_extrAtt = {0, 1};
00749   std::pair<unsigned int, unsigned int> m_extrLor = {0, 1};
00750   std::pair<unsigned int, unsigned int> m_extrMob = {0, 1};
00751   std::pair<unsigned int, unsigned int> m_extrDis = {0, 1};
00752
00753   // Interpolation methods
00754   unsigned int m_intpVel = 2;
00755   unsigned int m_intpDif = 2;
00756   unsigned int m_intpAlp = 2;
00757   unsigned int m_intpAtt = 2;
00758   unsigned int m_intpLor = 2;
00759   unsigned int m_intpMob = 2;
00760   unsigned int m_intpDis = 2;
00761
00762   bool Velocity(const double ex, const double ey, const double ez,
00763                 const double bx, const double by, const double bz,
00764                 const std::vector<std::vector<std::vector<double> > >& velE,
00765                 const std::vector<std::vector<std::vector<double> > >& velB,
00766                 const std::vector<std::vector<std::vector<double> > >& velX,
00767                 const double q, double& vx, double& vy, double& vz) const;
00768   bool VelocityFluxBulk(
00769       const double ex, const double ey, const double ez, const double bx,
00770       const double by, const double bz,
00771       const std::vector<std::vector<std::vector<double> > >& velWv,
00772       const std::vector<std::vector<std::vector<double> > >& velWr,
00773       double& wv, double& wr) const;
00774   static void Langevin(const double ex, const double ey, const double ez,
00775                        double bx, double by, double bz, const double mu,
00776                        double& vx, double& vy, double& vz);
00777   static void Langevin(const double ex, const double ey, const double ez,
00778                        double bx, double by, double bz, const double mu,
00779                        const double muH, double& vx, double& vy, double& vz);
00780   bool Diffusion(const double ex, const double ey, const double ez,
00781                  const double bx, const double by, const double bz,
00782                  const std::vector<std::vector<std::vector<double> > >& difL,
00783                  const std::vector<std::vector<std::vector<double> > >& difT,
00784                  double& dl, double& dt) const;
00785   bool Diffusion(
00786       const double ex, const double ey, const double ez, const double bx,
00787       const double by, const double bz,
00788       const std::vector<std::vector<std::vector<std::vector<double> > > >& diff,
00789       double cov[3][3]) const;
00790   bool Alpha(const double ex, const double ey, const double ez, const double bx,
00791              const double by, const double bz,
00792              const std::vector<std::vector<std::vector<double> > >& tab,
00793              unsigned int intp, const unsigned int thr,
00794              const std::pair<unsigned int, unsigned int>& extr,
00795              double& alpha) const;
00796   double GetAngle(const double ex, const double ey, const double ez,
00797                   const double bx, const double by, const double bz,
00798                   const double e, const double b) const;
00799   bool Interpolate(const double e, const double b, const double a,
00800                    const std::vector<std::vector<std::vector<double> > >& table,
00801                    double& y, const unsigned int intp,
00802                    const std::pair<unsigned int, unsigned int>& extr,
00803                    const bool logval = false) const;
00804
00805   double Interpolate1D(const double e, const std::vector<double>& table,
```

```
00806                     const std::vector<double>& fields,
00807                     const unsigned int intpMeth,
00808                     const std::pair<unsigned int, unsigned int>& extr,
00809                     const bool logval = false) const;
00810
00811   bool SetEntry(const size_t i, const size_t j, const size_t k,
00812                 const std::string& fcn,
00813                 std::vector<std::vector<std::vector<double> > >& tab,
00814                 const double val);
00815   bool GetEntry(const size_t i, const size_t j, const size_t k,
00816                 const std::string& fcn,
00817                 const std::vector<std::vector<std::vector<double> > >& tab,
00818                 double& val) const;
00819
00820   void SetExtrapolationMethod(const std::string& low, const std::string& high,
00821                               std::pair<unsigned int, unsigned int>& extr,
00822                               const std::string& fcn);
00823   bool GetExtrapolationIndex(std::string str, unsigned int& nb) const;
00824   size_t SetThreshold(
00825       const std::vector<std::vector<std::vector<double> > >& tab) const;
00826
00827   void Clone(std::vector<std::vector<std::vector<double> > >& tab,
00828              const std::vector<double>& efields,
00829              const std::vector<double>& bfields,
00830              const std::vector<double>& angles, const unsigned int intp,
00831              const std::pair<unsigned int, unsigned int>& extr,
00832              const double init, const std::string& label);
00833   void Clone(std::vector<std::vector<std::vector<std::vector<double> > > >& tab,
00834              const size_t n, const std::vector<double>& efields,
00835              const std::vector<double>& bfields,
00836              const std::vector<double>& angles, const unsigned int intp,
00837              const std::pair<unsigned int, unsigned int>& extr,
00838              const double init, const std::string& label);
00839
00840   void Init(const size_t nE, const size_t nB, const size_t nA,
00841             std::vector<std::vector<std::vector<double> > >& tab,
00842             const double val);
00843   void Init(const size_t nE, const size_t nB, const size_t nA, const size_t nT,
00844             std::vector<std::vector<std::vector<std::vector<double> > > >& tab,
00845             const double val);
00846
00847 #else
00848
00849 #include "MediumMagboltz.hh"
00850
00851   friend class MediumGas;
00852   friend class MediumMagboltz;
00853
00854   // enum to mimic polymorphism
00855   enum class MediumType { Medium = 0, MediumGas, MediumMagboltz };
00856
00857   MediumType m_MediumType{MediumType::Medium};
00858
00859 #endif
00860 };
00861 }  // namespace Garfield
00862
00863 #endif
```

## 8.73 /builds/garfield/docs/source/Include/Garfield/MediumCdTe.hh File Reference

#include "Garfield/Medium.hh"

### Classes

- class Garfield::MediumCdTe

  Cadmium-Telluride.

### Namespaces

- namespace Garfield

## 8.74 MediumCdTe.hh

Go to the documentation of this file.

```
00001 #ifndef G_MEDIUM_CDTE_H
00002 #define G_MEDIUM_CDTE_H
00003
```

```
00004 #include "Garfield/Medium.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class MediumCdTe : public Medium {
00011  public:
00013   MediumCdTe();
00015   virtual ~MediumCdTe() {}
00016
00017   bool IsSemiconductor() const override { return true; }
00018
00019   void GetComponent(const unsigned int i, std::string& label,
00020                     double& f) override;
00021
00022   // Electron transport parameters
00023   bool ElectronVelocity(const double ex, const double ey, const double ez,
00024                         const double bx, const double by, const double bz,
00025                         double& vx, double& vy, double& vz) override;
00026   bool ElectronTownsend(const double ex, const double ey, const double ez,
00027                         const double bx, const double by, const double bz,
00028                         double& alpha) override;
00029   bool ElectronAttachment(const double ex, const double ey, const double ez,
00030                           const double bx, const double by, const double bz,
00031                           double& eta) override;
00032   double ElectronMobility() override { return m_eMobility; }
00033   // Hole transport parameters
00034   bool HoleVelocity(const double ex, const double ey, const double ez,
00035                     const double bx, const double by, const double bz,
00036                     double& vx, double& vy, double& vz) override;
00037   bool HoleTownsend(const double ex, const double ey, const double ez,
00038                     const double bx, const double by, const double bz,
00039                     double& alpha) override;
00040   bool HoleAttachment(const double ex, const double ey, const double ez,
00041                       const double bx, const double by, const double bz,
00042                       double& eta) override;
00043   double HoleMobility() override { return m_hMobility; }
00044
00045   void SetLowFieldMobility(const double mue, const double muh);
00046   void UnsetLowFieldMobility();
00047
00048  private:
00049   // Band gap energy [eV]
00050   // m_bandGap = 1.44;
00051   // Low-field mobility
00052   double m_eMobility = 1.05e-6;
00053   double m_hMobility = 0.1e-6;
00054   // Hall factor
00055   double m_eHallFactor = 1.15;
00056   double m_hHallFactor = 0.7;
00057
00058   bool m_userMobility = false;
00059
00060   void UpdateTransportParameters();
00061 };
00062 }  // namespace Garfield
00063
00064 #endif
```

## 8.75 /builds/garfield/docs/source/Include/Garfield/MediumConductor.hh File Reference

#include "Garfield/Medium.hh"

### Classes

- class Garfield::MediumConductor

  Conducting medium.

### Namespaces

- namespace Garfield

## 8.76 MediumConductor.hh

Go to the documentation of this file.

```
00001 #ifndef G_MEDIUM_CONDUCTOR_H
```

```
00002 #define G_MEDIUM_CONDUCTOR_H
00003
00004 #include "Garfield/Medium.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class MediumConductor : public Medium {
00011  public:
00013   MediumConductor() : Medium() {
00014     m_className = "MediumConductor";
00015     m_name = "Conductor";
00016   }
00018   virtual ~MediumConductor() {}
00019
00020   bool IsConductor() const override { return true; }
00021   void EnableDrift(const bool /*on*/) override {}
00022   void EnablePrimaryIonisation(const bool /*on*/) override {}
00023 };
00024 }  // namespace Garfield
00025
00026 #endif
```

## 8.77 /builds/garfield/docs/source/Include/Garfield/MediumDiamond.hh File Reference

#include <mutex>
#include "Garfield/Medium.hh"

### Classes

- class Garfield::MediumDiamond

  Diamond.

### Namespaces

- namespace Garfield

## 8.78 MediumDiamond.hh

Go to the documentation of this file.

```
00001 #ifndef G_MEDIUM_DIAMOND_H
00002 #define G_MEDIUM_DIAMOND_H
00003
00004 #include <mutex>
00005
00006 #include "Garfield/Medium.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class MediumDiamond : public Medium {
00013  public:
00015   MediumDiamond();
00017   virtual ~MediumDiamond() {}
00018
00019   bool IsSemiconductor() const override { return true; }
00020
00021   void GetComponent(const unsigned int i, std::string& label,
00022                     double& f) override;
00023
00024   // Electron transport parameters
00025   bool ElectronVelocity(const double ex, const double ey, const double ez,
00026                         const double bx, const double by, const double bz,
00027                         double& vx, double& vy, double& vz) override;
00028   bool ElectronTownsend(const double ex, const double ey, const double ez,
00029                         const double bx, const double by, const double bz,
00030                         double& alpha) override;
00031   bool ElectronAttachment(const double ex, const double ey, const double ez,
00032                           const double bx, const double by, const double bz,
00033                           double& eta) override;
00034   double ElectronMobility() override { return m_eMobility; }
00035   // Hole transport parameters
00036   bool HoleVelocity(const double ex, const double ey, const double ez,
00037                     const double bx, const double by, const double bz,
00038                     double& vx, double& vy, double& vz) override;
00039   bool HoleTownsend(const double ex, const double ey, const double ez,
00040                     const double bx, const double by, const double bz,
```

```
00041                   double& alpha) override;
00042   bool HoleAttachment(const double ex, const double ey, const double ez,
00043                       const double bx, const double by, const double bz,
00044                       double& eta) override;
00045   double HoleMobility() override { return m_hMobility; }
00046
00047   void SetLowFieldMobility(const double mue, const double muh);
00048   void UnsetLowFieldMobility();
00049
00050   void SetSaturationVelocity(const double vsate, const double vsath);
00051   void UnsetSaturationVelocity();
00052
00053  private:
00054   std::mutex m_mutex;
00055
00056   // Low-field mobility
00057   double m_eMobility = 4.551e-6;
00058   double m_hMobility = 2.750e-6;
00059   // Hall factor
00060   double m_eHallFactor = 1.;
00061   double m_hHallFactor = 1.;
00062   // Saturation velocity
00063   double m_eSatVel = 2.6e-2;
00064   double m_hSatVel = 1.6e-2;
00065
00066   bool m_userMobility = false;
00067
00068   void UpdateTransportParameters();
00069 };
00070 }  // namespace Garfield
00071
00072 #endif
```

## 8.79 /builds/garfield/docs/source/Include/Garfield/MediumGaAs.hh File Reference

#include "Garfield/Medium.hh"

### Classes

- class Garfield::MediumGaAs

  Gallium-Arsenide.

### Namespaces

- namespace Garfield

## 8.80 MediumGaAs.hh

Go to the documentation of this file.

```
00001 #ifndef G_MEDIUM_GAAS_H
00002 #define G_MEDIUM_GAAS_H
00003
00004 #include "Garfield/Medium.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class MediumGaAs : public Medium {
00011  public:
00013   MediumGaAs();
00015   virtual ~MediumGaAs() {}
00016
00017   bool IsSemiconductor() const override { return true; }
00018
00019   void GetComponent(const unsigned int i, std::string& label,
00020                     double& f) override;
00021
00022   // Electron transport parameters
00023   bool ElectronVelocity(const double ex, const double ey, const double ez,
00024                         const double bx, const double by, const double bz,
00025                         double& vx, double& vy, double& vz) override;
00026   bool ElectronTownsend(const double ex, const double ey, const double ez,
00027                         const double bx, const double by, const double bz,
00028                         double& alpha) override;
00029   bool ElectronAttachment(const double ex, const double ey, const double ez,
00030                           const double bx, const double by, const double bz,
00031                           double& eta) override;
00032   double ElectronMobility() override { return m_eMobility; }
```

```
00033   // Hole transport parameters
00034   bool HoleVelocity(const double ex, const double ey, const double ez,
00035                     const double bx, const double by, const double bz,
00036                     double& vx, double& vy, double& vz) override;
00037   bool HoleTownsend(const double ex, const double ey, const double ez,
00038                     const double bx, const double by, const double bz,
00039                     double& alpha) override;
00040   bool HoleAttachment(const double ex, const double ey, const double ez,
00041                       const double bx, const double by, const double bz,
00042                       double& eta) override;
00043   double HoleMobility() override { return m_hMobility; }
00044
00045   void SetLowFieldMobility(const double mue, const double muh);
00046   void UnsetLowFieldMobility();
00047
00048  private:
00049   // Band-gap energy [eV]
00050   // double m_bandGap = 1.42;
00051   // Low-field mobility
00052   double m_eMobility = 8.0e-6;
00053   double m_hMobility = 0.4e-6;
00054   // Saturation velocity
00055   double m_eSatVel = 7.7e-3;
00056   double m_hSatVel = 7.7e-3;
00057   // Hall factor
00058   double m_eHallFactor = 1.05;
00059   double m_hHallFactor = 1.25;
00060   // Impact ionization parameters
00061   double m_eImpactA = 1.889e5;
00062   double m_hImpactA = 2.215e5;
00063   double m_eImpactB = 5.75e5;
00064   double m_hImpactB = 6.57e5;
00065
00066   bool m_userMobility = false;
00067   void UpdateTransportParameters();
00068 };
00069 }  // namespace Garfield
00070
00071 #endif
```

## 8.81 /builds/garfield/docs/source/Include/Garfield/MediumGaN.hh File Reference

#include ”Garfield/Medium.hh”

### Classes

- class Garfield::MediumGaN

  Gallium-Nitride.

### Namespaces

- namespace Garfield

## 8.82 MediumGaN.hh

Go to the documentation of this file.

```
00001 #ifndef G_MEDIUM_GAN_H
00002 #define G_MEDIUM_GAN_H
00003
00004 #include ”Garfield/Medium.hh”
00005
00006 namespace Garfield {
00007
00009
00010 class MediumGaN : public Medium {
00011  public:
00013   MediumGaN();
00015   virtual ~MediumGaN() {}
00016
00017   bool IsSemiconductor() const override { return true; }
00018
00019   void GetComponent(const unsigned int i, std::string& label,
00020                     double& f) override;
00021
00022   // Electron transport parameters
00023   bool ElectronVelocity(const double ex, const double ey, const double ez,
00024                         const double bx, const double by, const double bz,
00025                         double& vx, double& vy, double& vz) override;
```

```
00026   bool ElectronTownsend(const double ex, const double ey, const double ez,
00027                         const double bx, const double by, const double bz,
00028                         double& alpha) override;
00029   bool ElectronAttachment(const double ex, const double ey, const double ez,
00030                           const double bx, const double by, const double bz,
00031                           double& eta) override;
00032   double ElectronMobility() override { return m_eMobility; }
00033   // Hole transport parameters
00034   bool HoleVelocity(const double ex, const double ey, const double ez,
00035                     const double bx, const double by, const double bz,
00036                     double& vx, double& vy, double& vz) override;
00037   bool HoleTownsend(const double ex, const double ey, const double ez,
00038                     const double bx, const double by, const double bz,
00039                     double& alpha) override;
00040   bool HoleAttachment(const double ex, const double ey, const double ez,
00041                       const double bx, const double by, const double bz,
00042                       double& eta) override;
00043   double HoleMobility() override { return m_hMobility; }
00044
00046   void SetElectronConcentration(const double c);
00047
00049   void SetLowFieldMobility(const double mue, const double muh);
00051   void UnsetLowFieldMobility();
00052
00053  private:
00054   // Low-field mobility.
00055   double m_eMobility = 1.405e-6;
00056   double m_hMobility = 0.170e-6;
00057
00058   // Electron concentration.
00059   double m_eDensity = 7.78e16;
00060
00061   // Hall factors.
00062   double m_eHallFactor = 1.;
00063   double m_hHallFactor = 1.;
00064
00065   // Impact ionization parameters.
00066   // J. Baliga, Semicond. Sci. Technol. 28 (2013) 074011,
00067   // https://doi-org.ezproxy.cern.ch/10.1088/0268-1242/28/7/074011
00068   double m_eImpactA = 1.5e5;
00069   double m_hImpactA = 6.4e5;
00070   double m_eImpactB = 1.41e7;
00071   double m_hImpactB = 1.46e7;
00072
00073   bool m_userMobility = false;
00074   void UpdateTransportParameters();
00075 };
00076 }  // namespace Garfield
00077
00078 #endif
```

## 8.83 /builds/garfield/docs/source/Include/Garfield/MediumGas.hh File Reference

#include <array>
#include <bitset>
#include <cmath>
#include <vector>
#include "Garfield/Medium.hh"

### Classes

- class Garfield::MediumGas

  Base class for gas media.
- struct Garfield::MediumGas::ExcLevel
- struct Garfield::MediumGas::IonLevel

### Namespaces

- namespace Garfield

## 8.84 MediumGas.hh

Go to the documentation of this file.

```
00001 #if defined(__GPUCOMPILE__) || !defined(G_MEDIUM_GAS_H)
```

```
00002
00003 #if !defined(___GPUCOMPILE___) && !defined(G__MEDIUM__GAS__H)
00004 #define G__MEDIUM__GAS__H
00005 #endif
00006
00007 #ifdef ___GPUCOMPILE___
00008
00009 #else
00010
00011 #include <array>
00012 #include <bitset>
00013 #include <cmath>
00014 #include <vector>
00015
00016 #include "Garfield/Medium.hh"
00017
00018 #endif
00019
00020 #ifndef ___GPUCOMPILE___
00021 namespace Garfield {
00022
00024
00025 class MediumGas : public Medium {
00026  public:
00028   MediumGas();
00030   virtual ~MediumGas() {}
00031
00033   bool SetComposition(const std::string& gas1, const double f1 = 1.,
00034                       const std::string& gas2 = "", const double f2 = 0.,
00035                       const std::string& gas3 = "", const double f3 = 0.,
00036                       const std::string& gas4 = "", const double f4 = 0.,
00037                       const std::string& gas5 = "", const double f5 = 0.,
00038                       const std::string& gas6 = "", const double f6 = 0.);
00040   void GetComposition(std::string& gas1, double& f1, std::string& gas2,
00041                       double& f2, std::string& gas3, double& f3,
00042                       std::string& gas4, double& f4, std::string& gas5,
00043                       double& f5, std::string& gas6, double& f6) const;
00044
00046   bool LoadGasFile(const std::string& filename, const bool quiet = false);
00048   bool WriteGasFile(const std::string& filename);
00050   bool MergeGasFile(const std::string& filename, const bool replaceOld);
00051
00054   virtual bool EnablePenningTransfer();
00061   virtual bool EnablePenningTransfer(const double r, const double lambda);
00064   virtual bool EnablePenningTransfer(const double r, const double lambda,
00065                                      std::string gasname);
00067   virtual void DisablePenningTransfer();
00069   virtual bool DisablePenningTransfer(std::string gasname);
00070
00073   bool GetPenningTransfer(const std::string& gasname, double& r,
00074                           double& lambda);
00075
00077   virtual void PrintGas();
00079   static void PrintGases();
00080
00082   bool LoadIonMobility(const std::string& filename, const bool quiet = false);
00084   bool LoadNegativeIonMobility(const std::string& filename,
00085                                const bool quiet = false);
00086
00089   bool AdjustTownsendCoefficient();
00090
00092   size_t GetNumberOfIonisationLevels() const { return m_ionLevels.size(); }
00094   size_t GetNumberOfExcitationLevels() const { return m_excLevels.size(); }
00096   void GetIonisationLevel(const size_t level, std::string& label,
00097                           double& energy) const;
00099   void GetExcitationLevel(const size_t level, std::string& label,
00100                           double& energy) const;
00102   bool GetElectronIonisationRate(const size_t level, const size_t ie,
00103                                  const size_t ib, const size_t ia,
00104                                  double& f) const;
00106   bool GetElectronExcitationRate(const size_t level, const size_t ie,
00107                                  const size_t ib, const size_t ia,
00108                                  double& f) const;
00109
00110   bool IsGas() const override { return true; }
00111
00112   void GetComponent(const unsigned int i, std::string& label,
00113                     double& f) override;
00114
00115   void SetAtomicNumber(const double z) override;
00116   double GetAtomicNumber() const override;
00117   void SetAtomicWeight(const double a) override;
00118   double GetAtomicWeight() const override;
00119   void SetNumberDensity(const double n) override;
00120   double GetNumberDensity() const override;
00121   void SetMassDensity(const double rho) override;
00122   double GetMassDensity() const override;
```

```
00123
00124   void ResetTables() override;
00125
00126   void SetExtrapolationMethodExcitationRates(const std::string& low,
00127                                              const std::string& high) {
00128     SetExtrapolationMethod(low, high, m_extrExc, "ExcitationRates");
00129   }
00130   void SetExtrapolationMethodIonisationRates(const std::string& low,
00131                                              const std::string& high) {
00132     SetExtrapolationMethod(low, high, m_extrIon, "IonisationRates");
00133   }
00134   void SetInterpolationMethodExcitationRates(const unsigned int intrp) {
00135     if (intrp > 0) m_intpExc = intrp;
00136   }
00137   void SetInterpolationMethodIonisationRates(const unsigned int intrp) {
00138     if (intrp > 0) m_intpIon = intrp;
00139   }
00140
00141   // Scaling laws.
00142   // TODO: cache scaling factors.
00143   double ScaleElectricField(const double e) const override {
00144     return e * m_pressureTable / m_pressure;
00145   }
00146   double UnScaleElectricField(const double e) const override {
00147     return e * m_pressure / m_pressureTable;
00148   }
00149   double ScaleDiffusion(const double d) const override {
00150     return d * sqrt(m_pressureTable / m_pressure);
00151   }
00152   double ScaleDiffusionTensor(const double d) const override {
00153     return d * m_pressureTable / m_pressure;
00154   }
00155   double ScaleTownsend(const double alpha) const override {
00156     return alpha * m_pressure / m_pressureTable;
00157   }
00158   double ScaleAttachment(const double eta) const override {
00159     return eta * m_pressure / m_pressureTable;
00160   }
00161   double ScaleLorentzAngle(const double lor) const override {
00162     return lor * m_pressure / m_pressureTable;
00163   }
00164
00165   bool GetPhotoAbsorptionCrossSection(const double e, double& sigma,
00166                                       const unsigned int i) override;
00167
00169   virtual double CreateGPUTransferObject(MediumGPU*& med_gpu) override;
00170
00171  protected:
00172 #endif
00173
00174   static constexpr unsigned int m_nMaxGases = 6;
00175
00176 #ifndef ___GPUCOMPILE___
00177   // Gas mixture
00178   std::array<std::string, m_nMaxGases> m_gas;
00179   std::array<double, m_nMaxGases> m_fraction;
00180   std::array<double, m_nMaxGases> m_atWeight;
00181   std::array<double, m_nMaxGases> m_atNum;
00182
00183   // Penning transfer
00184   // Flag enabling/disabling Penning transfer
00185   bool m_usePenning = false;
00186   // Penning transfer probability
00187   double m_rPenningGlobal = 0.;
00188   // Mean distance of Penning ionisation
00189   double m_lambdaPenningGlobal = 0.;
00190   // Penning transfer probability per component
00191   std::array<double, m_nMaxGases> m_rPenningGas;
00192   // Penning transfer distance per component
00193   std::array<double, m_nMaxGases> m_lambdaPenningGas;
00194
00195   // Pressure at which the transport parameter table was calculated
00196   double m_pressureTable;
00197   // Temperature at which the transport parameter table was calculated
00198   double m_temperatureTable;
00199
00200   // Table of Townsend coefficients without Penning transfer
00201   std::vector<std::vector<std::vector<double> > > m_eAlp0;
00202
00203   // Tables for excitation and ionisation rates
00204   std::vector<std::vector<std::vector<std::vector<double> > > > m_excRates;
00205   std::vector<std::vector<std::vector<std::vector<double> > > > m_ionRates;
00206
00207   // Store excitation and ionization information
00208   struct ExcLevel {
00209     std::string label;
00210     double energy;
```

```
00211     double prob;
00212     double rms;
00213     double dt;
00214   };
00215   std::vector<ExcLevel> m_excLevels;
00216
00217   struct IonLevel {
00218     std::string label;
00219     double energy;
00220   };
00221   std::vector<IonLevel> m_ionLevels;
00222
00223   // Extrapolation/interpolation for excitation and ionisation rates.
00224   std::pair<unsigned int, unsigned int> m_extrExc = {0, 1};
00225   std::pair<unsigned int, unsigned int> m_extrIon = {0, 1};
00226   unsigned int m_intpExc = 2;
00227   unsigned int m_intpIon = 2;
00228
00229   bool LoadMobility(const std::string& filename, const bool quiet,
00230                     const bool negative);
00231   bool ReadHeader(std::ifstream& gasfile, int& version, std::bitset<20>& gasok,
00232                   bool& is3d, std::vector<double>& mixture,
00233                   std::vector<double>& efields, std::vector<double>& bfields,
00234                   std::vector<double>& angles, std::vector<ExcLevel>& excLevels,
00235                   std::vector<IonLevel>& ionLevels);
00236   void ReadFooter(std::ifstream& gasfile, std::array<unsigned int, 13>& extrapH,
00237                   std::array<unsigned int, 13>& extrapL,
00238                   std::array<unsigned int, 13>& interp, unsigned int& thrAlp,
00239                   unsigned int& thrAtt, unsigned int& thrDis, double& ionDiffL,
00240                   double& ionDiffT, double& pgas, double& tgas);
00241   void ReadRecord3D(std::ifstream& gasfile, double& ve, double& vb, double& vx,
00242                     double& dl, double& dt, double& alpha, double& alpha0,
00243                     double& eta, double& mu, double& lor, double& dis,
00244                     std::array<double, 6>& dif, std::vector<double>& rexc,
00245                     std::vector<double>& rion, std::bitset<20> gasok);
00246   void ReadRecord1D(std::ifstream& gasfile, double& ve, double& vb, double& vx,
00247                     double& wv, double& wr, double& dl, double& dt,
00248                     double& alpha, double& alpha0, double& eta, double& riontof,
00249                     double& ratttof, double& mu, double& lor, double& dis,
00250                     std::array<double, 6>& dif, std::vector<double>& rexc,
00251                     std::vector<double>& rion, std::bitset<20> gasok);
00252   void InsertE(const int ie, const int ne, const int nb, const int na);
00253   void InsertB(const int ib, const int ne, const int nb, const int na);
00254   void InsertA(const int ia, const int ne, const int nb, const int na);
00255   void ZeroRowE(const int ie, const int nb, const int na);
00256   void ZeroRowB(const int ib, const int ne, const int na);
00257   void ZeroRowA(const int ia, const int ne, const int nb);
00258   bool GetMixture(const std::vector<double>& mixture, const int version,
00259                   std::vector<std::string>& gasnames,
00260                   std::vector<double>& percentages) const;
00261   void GetGasBits(std::bitset<20>& gasok) const;
00262
00263   static bool GetGasInfo(const std::string& gasname, double& a, double& z,
00264                          double& w, double& f);
00265   static std::string GetGasName(const int gasnumber, const int version);
00266   static std::string GetGasName(std::string input);
00267   static int GetGasNumberGasFile(const std::string& input);
00268   static const std::vector<std::string> GetAliases(const std::string& gas);
00269 };
00270 }
00271
00272 #endif
00273 #endif
```

## 8.85 /builds/garfield/docs/source/Include/Garfield/MediumMagboltz.hh File Reference

#include <array>
#include <mutex>
#include "Garfield/MagboltzInterface.hh"
#include "Garfield/MediumGas.hh"

Classes

- class Garfield::MediumMagboltz

  Interface to Magboltz (version 11).
- struct Garfield::MediumMagboltz::Deexcitation

Namespaces

- namespace Garfield

## 8.86 MediumMagboltz.hh

Go to the documentation of this file.

```
00001 #if defined(___GPUCOMPILE___) || !defined(G_MEDIUM_MAGBOLTZ_11)
00002
00003 #if !defined(___GPUCOMPILE___) && !defined(G_MEDIUM_MAGBOLTZ_11)
00004 #define G_MEDIUM_MAGBOLTZ_11
00005 #endif
00006
00007 #ifndef ___GPUCOMPILE___
00008 #include <array>
00009 #include <mutex>
00010 #endif
00011 #include "Garfield/MagboltzInterface.hh"
00012 #include "Garfield/MediumGas.hh"
00013 #ifndef ___GPUCOMPILE___
00014
00015 class TPad;
00016
00017 namespace Garfield {
00018
00021
00022 class MediumMagboltz : public MediumGas {
00023  public:
00025   MediumMagboltz();
00027   MediumMagboltz(const std::string& gas1, const double f1 = 1.,
00028                  const std::string& gas2 = "", const double f2 = 0.,
00029                  const std::string& gas3 = "", const double f3 = 0.,
00030                  const std::string& gas4 = "", const double f4 = 0.,
00031                  const std::string& gas5 = "", const double f5 = 0.,
00032                  const std::string& gas6 = "", const double f6 = 0.);
00034   virtual ~MediumMagboltz() {}
00035
00038   bool SetMaxElectronEnergy(const double e);
00040   double GetMaxElectronEnergy() const { return m_eMax; }
00041
00044   bool SetMaxPhotonEnergy(const double e);
00046   double GetMaxPhotonEnergy() const { return m_eFinalGamma; }
00047
00049   void EnableAnisotropicScattering(const bool on = true) {
00050     m_useAnisotropic = on;
00051     m_isChanged = true;
00052   }
00053
00055   void SetSplittingFunctionOpalBeaty();
00057   void SetSplittingFunctionGreenSawada();
00059   void SetSplittingFunctionFlat();
00060
00062   void EnableDeexcitation();
00064   void DisableDeexcitation() { m_useDeexcitation = false; }
00066   void EnableRadiationTrapping();
00068   void DisableRadiationTrapping() { m_useRadTrap = false; }
00071   void SetLineWidth(const double n) { m_nAbsWidths = n; }
00072
00073   bool EnablePenningTransfer() override;
00074   bool EnablePenningTransfer(const double r, const double lambda) override;
00075   bool EnablePenningTransfer(const double r, const double lambda,
00076                              std::string gasname) override;
00077   void DisablePenningTransfer() override;
00078   bool DisablePenningTransfer(std::string gasname) override;
00079
00081   void EnableCrossSectionOutput(const bool on = true) { m_useCsOutput = on; }
00082
00084   void SetExcitationScaling(const double r, std::string gasname);
00085
00089   bool Initialise(const bool verbose = false);
00090
00091   void PrintGas() override;
00092
00093 #endif
00094
00095 #ifdef ___GPUCOMPILE___
00097   ___device___ double GetElectronNullCollisionRate(const int band);
00098 #else
00099 double GetElectronNullCollisionRate(const int band) override;
00100 #endif
00101
00102 #ifdef ___GPUCOMPILE___
00103   ___device___ double GetElectronCollisionRate___MediumMagboltz(const double e,
00104                                                 const int band);
00105 #endif
```

```
00106
00107 #ifndef ___GPUCOMPILE___
00109   double GetElectronCollisionRate(const double e, const int band) override;
00111   double GetElectronCollisionRate(const double e, const unsigned int level,
00112                                   const int band);
00114   bool ElectronCollision(const double e, int& type, int& level, double& e1,
00115                          double& dx, double& dy, double& dz,
00116                          std::vector<Secondary>& secondaries,
00117                          int& band) override;
00118   void ComputeDeexcitation(int iLevel, int& fLevel,
00119                            std::vector<Secondary>& secondaries);
00120
00121   double GetPhotonCollisionRate(const double e) override;
00122   bool PhotonCollision(const double e, int& type, int& level, double& e1,
00123                        double& ctheta,
00124                        std::vector<Secondary>& secondaries) override;
00125
00127   void ResetCollisionCounters();
00129   unsigned int GetNumberOfElectronCollisions() const;
00131   unsigned int GetNumberOfElectronCollisions(unsigned int& nElastic,
00132                                              unsigned int& nIonising,
00133                                              unsigned int& nAttachment,
00134                                              unsigned int& nInelastic,
00135                                              unsigned int& nExcitation,
00136                                              unsigned int& nSuperelastic) const;
00138   unsigned int GetNumberOfLevels();
00140   bool GetLevel(const unsigned int i, int& ngas, int& type, std::string& descr,
00141                 double& e);
00143   bool GetPenningTransfer(const unsigned int i, double& r, double& lambda);
00144
00146   unsigned int GetNumberOfElectronCollisions(const unsigned int level) const;
00147
00149   unsigned int GetNumberOfPenningTransfers() const { return m_nPenning; }
00150
00152   unsigned int GetNumberOfPhotonCollisions() const;
00154   unsigned int GetNumberOfPhotonCollisions(unsigned int& nElastic,
00155                                            unsigned int& nIonising,
00156                                            unsigned int& nInelastic) const;
00157
00161   void EnableThermalMotion(const bool on = true) { m_useGasMotion = on; }
00164   void EnableAutoEnergyLimit(const bool on = true) { m_autoEnergyLimit = on; }
00165
00192   void RunMagboltz(const double e, const double b, const double btheta,
00193                    const int ncoll, bool verbose, double& vx, double& vy,
00194                    double& vz, double& wv, double& wr, double& dl, double& dt,
00195                    double& alpha, double& eta, double& riontof, double& ratttof,
00196                    double& lor, double& vxerr, double& vyerr, double& vzerr,
00197                    double& wverr, double& wrerr, double& dlerr, double& dterr,
00198                    double& alphaerr, double& etaerr, double& riontoferr,
00199                    double& ratttoferr, double& lorerr, double& alphatof,
00200                    std::array<double, 6>& difftens);
00201
00205   void GenerateGasTable(const int numCollisions = 10,
00206                         const bool verbose = true);
00207
00208   void PlotElectronCrossSections(const unsigned int i, TPad* pad);
00209   void PlotElectronCollisionRates(TPad* pad);
00210   void PlotElectronInverseMeanFreePath(TPad* pad);
00211
00212   static int GetGasNumberMagboltz(const std::string& input);
00213   double CreateGPUTransferObject(MediumGPU*& med_gpu) override;
00214
00215  private:
00216 #endif
00217   static constexpr int nEnergyStepsLog = 1000;
00218   static constexpr int nEnergyStepsGamma = 5000;
00219   static constexpr int nCsTypes = 7;
00220
00221 #ifndef ___GPUCOMPILE___
00222   static constexpr int nCsTypesGamma = 4;
00223
00224   static const int DxcTypeRad;
00225   static const int DxcTypeCollIon;
00226   static const int DxcTypeCollNonIon;
00227
00229   std::mutex m_mutex;
00230
00232   bool m_useGasMotion = false;
00234   bool m_autoEnergyLimit = true;
00235 #endif
00236
00238   double m_eMax;
00240   double m_eStep;
00242   double m_eStepInv;
00243   double m_eHigh, m_eHighLog;
00244   double m_lnStep;
00245
```

```
00247   bool m_useCsOutput = false;
00249   unsigned int m_nTerms = 0;
00250 #ifndef ___GPUCOMPILE___
00252   std::array<double, m_nMaxGases> m_mgas;
00254   std::array<double, m_nMaxGases> m_rgas;
00255   std::array<double, m_nMaxGases> m_s2;
00257   std::array<double, Magboltz::nMaxLevels> m_wOpalBeaty;
00260   std::array<std::array<double, 5>, m_nMaxGases> m_parGreenSawada;
00261   std::array<bool, m_nMaxGases> m_hasGreenSawada;
00262 #endif
00264   bool m_useOpalBeaty = true;
00266   bool m_useGreenSawada = false;
00267
00268 #ifdef ___GPUCOMPILE___
00269   int m_csType[Magboltz::nMaxLevels];
00270   double m_energyLoss[Magboltz::nMaxLevels];
00271
00272   double** m_scatPar{nullptr};
00273   int* m_numscatParIdx{nullptr};
00274   int m_numscatPar{0};
00275
00276   double** m_scatCut{nullptr};
00277   int* m_numscatCutIdx{nullptr};
00278   int m_numscatCut{0};
00279
00280   double** m_scatParLog{nullptr};
00281   int* m_numscatParLogIdx{nullptr};
00282   int m_numscatParLog{0};
00283
00284   double** m_scatCutLog{nullptr};
00285   int* m_numscatCutLogIdx{nullptr};
00286   int m_numscatCutLog{0};
00287   int m_scatModel[Magboltz::nMaxLevels];
00288
00289   double* m_cfTot{nullptr};
00290   int m_numcfTot{0};
00291   double* m_cfTotLog{nullptr};
00292   int m_numcfTotLog{0};
00293
00294   double** m_cf{nullptr};
00295   int* m_numcfIdx{nullptr};
00296   int m_numcf{0};
00297
00298   double** m_cfLog{nullptr};
00299   int* m_numcfLogIdx{nullptr};
00300   int m_numcfLog{0};
00301 #else
00303 std::array<double, Magboltz::nMaxLevels> m_energyLoss;
00305 std::array<int, Magboltz::nMaxLevels> m_csType;
00306
00308 std::array<double, Magboltz::nMaxLevels> m_yFluorescence;
00310 std::array<unsigned int, Magboltz::nMaxLevels> m_nAuger1;
00311 std::array<unsigned int, Magboltz::nMaxLevels> m_nAuger2;
00313 std::array<double, Magboltz::nMaxLevels> m_eAuger1;
00314 std::array<double, Magboltz::nMaxLevels> m_eAuger2;
00315 std::array<unsigned int, Magboltz::nMaxLevels> m_nFluorescence;
00316 std::array<double, Magboltz::nMaxLevels> m_eFluorescence;
00317
00318 // Parameters for calculation of scattering angles
00319 std::vector<std::vector<double> > m_scatPar;
00320 std::vector<std::vector<double> > m_scatCut;
00321 std::vector<std::vector<double> > m_scatParLog;
00322 std::vector<std::vector<double> > m_scatCutLog;
00323 std::array<int, Magboltz::nMaxLevels> m_scatModel;
00324
00326 std::vector<std::string> m_description;
00327
00328 // Total collision frequency
00329 std::vector<double> m_cfTot;
00330 std::vector<double> m_cfTotLog;
00331 // Collision frequencies
00332 std::vector<std::vector<double> > m_cf;
00333 std::vector<std::vector<double> > m_cfLog;
00334
00335 #endif
00336   bool m_useAnisotropic = true;
00338   double m_cfNull = 0.;
00339 #ifdef ___GPUCOMPILE___
00340   double m_wOpalBeaty[Magboltz::nMaxLevels];
00341   double m_yFluorescence[Magboltz::nMaxLevels];
00342
00343   unsigned int m_nAuger1[Magboltz::nMaxLevels];
00344   unsigned int m_nAuger2[Magboltz::nMaxLevels];
00346   double m_eAuger1[Magboltz::nMaxLevels];
00347   double m_eAuger2[Magboltz::nMaxLevels];
00348
00349   unsigned int m_nFluorescence[Magboltz::nMaxLevels];
```

```
00350   double m_eFluorescence[Magboltz::nMaxLevels];
00351   double m_rgas[m_nMaxGases];
00352   double m_s2[m_nMaxGases];
00353 #endif
00354
00355 #ifndef ___GPUCOMPILE___
00363   std::array<unsigned int, nCsTypes> m_nCollisions;
00365   std::vector<unsigned int> m_nCollisionsDetailed;
00366
00367   // Penning transfer
00369   std::array<double, Magboltz::nMaxLevels> m_rPenning;
00371   std::array<double, Magboltz::nMaxLevels> m_lambdaPenning;
00373   unsigned int m_nPenning = 0;
00374
00375   // Deexcitation
00377   bool m_useDeexcitation = false;
00380   bool m_useRadTrap = true;
00381
00382   struct Deexcitation {
00383     // Gas component
00384     int gas;
00385     // Associated cross-section term
00386     int level;
00387     // Level description
00388     std::string label;
00389     // Energy
00390     double energy;
00391     // Branching ratios
00392     std::vector<double> p;
00393     // Final levels
00394     std::vector<int> final;
00395     // Type of transition
00396     std::vector<int> type;
00397     // Oscillator strength
00398     double osc;
00399     // Total decay rate
00400     double rate;
00401     // Doppler broadening
00402     double sDoppler;
00403     // Pressure broadening
00404     double gPressure;
00405     // Effective width
00406     double width;
00407     // Integrated absorption collision rate
00408     double cf;
00409   };
00410   std::vector<Deexcitation> m_deexcitations;
00411   // Mapping between deexcitations and cross-section terms.
00412   std::array<int, Magboltz::nMaxLevels> m_iDeexcitation;
00413   // Number of emission widths within which to conosider discrete line
00414   // absorption.
00415   // TODO: default value?
00416   double m_nAbsWidths = 1000.;
00417
00419   std::array<double, m_nMaxGases> m_ionPot;
00421   double m_minIonPot = -1.;
00422
00423   // Scaling factor for excitation cross-sections
00424   std::array<double, m_nMaxGases> m_scaleExc;
00425
00426   // Energy spacing of photon collision rates table
00427   double m_eFinalGamma, m_eStepGamma;
00428   // Number of photon collision cross-section terms
00429   unsigned int m_nPhotonTerms = 0;
00430   // Total photon collision frequencies
00431   std::vector<double> m_cfTotGamma;
00432   // Photon collision frequencies
00433   std::vector<std::vector<double> > m_cfGamma;
00434   std::vector<int> csTypeGamma;
00435   // Photon collision counters
00436   // 0: elastic
00437   // 1: ionisation
00438   // 2: inelastic
00439   // 3: excitation
00440   std::array<unsigned int, nCsTypesGamma> m_nPhotonCollisions;
00441
00442   bool Update(const bool verbose = false);
00443   bool Mixer(const bool verbose = false);
00444   void SetupGreenSawada();
00445
00446   void GetExcitationIonisationLevels();
00447
00448   void ComputeDeexcitationTable(const bool verbose);
00449   void AddPenningDeexcitation(Deexcitation& dxc, const double rate,
00450                               const double pPenning) {
00451     dxc.p.push_back(rate * (1. - pPenning));
00452     dxc.type.push_back(DxcTypeCollNonIon);
```

```
00453     dxc.final.push_back(-1);
00454     if (pPenning > 0.) {
00455       dxc.p.push_back(rate * pPenning);
00456       dxc.type.push_back(DxcTypeCollIon);
00457       dxc.final.push_back(-1);
00458     }
00459   }
00460   double RateConstantWK(const double energy, const double osc,
00461                         const double pacs, const int igas1,
00462                         const int igas2) const;
00463   double RateConstantHardSphere(const double r1, const double r2,
00464                                 const int igas1, const int igas2) const;
00465   double CalcDiscreteLineCf(const Deexcitation& dxc, double e,
00466                             double cfOth) const;
00467   void ComputeDeexcitationInternal(int iLevel, int& fLevel,
00468                                    std::vector<Secondary>& secondaries);
00469   bool ComputePhotonCollisionTable(const bool verbose);
00470 };
00471 }
00472 #endif
00473 #endif
```

## 8.87 /builds/garfield/docs/source/Include/Garfield/MediumPlastic.hh File Reference

#include ”Garfield/Medium.hh”

Classes

- class Garfield::MediumPlastic

  Plastic medium.

Namespaces

- namespace Garfield

## 8.88 MediumPlastic.hh

Go to the documentation of this file.

```
00001 #ifndef G__MEDIUM__PLASTIC__H
00002 #define G__MEDIUM__PLASTIC__H
00003
00004 #include ”Garfield/Medium.hh”
00005
00006 namespace Garfield {
00007
00009
00010 class MediumPlastic : public Medium {
00011  public:
00013   MediumPlastic() : MediumPlastic(1.) {}
00015   MediumPlastic(const double eps) : Medium() {
00016     m__className = ”MediumPlastic”;
00017     m__name = ”Plastic”;
00018     if (eps > 1.) m__epsilon = eps;
00019   }
00021   virtual ~MediumPlastic() {}
00022
00023   void EnableDrift(const bool /*on*/) override {}
00024   void EnablePrimaryIonisation(const bool /*on*/) override {}
00025 };
00026 }  // namespace Garfield
00027
00028 #endif
```

## 8.89 /builds/garfield/docs/source/Include/Garfield/MediumSilicon.hh File Reference

#include <array>
#include <mutex>
#include <string>
#include <vector>
#include ”Garfield/Medium.hh”

Classes

- class Garfield::MediumSilicon

  Solid crystalline silicon
- struct Garfield::MediumSilicon::Band

Namespaces

- namespace Garfield

## 8.90 MediumSilicon.hh

Go to the documentation of this file.

```
00001 #ifndef G__MEDIUM__SILICON__H
00002 #define G__MEDIUM__SILICON__H
00003
00004 #include <array>
00005 #include <mutex>
00006 #include <string>
00007 #include <vector>
00008
00009 #include "Garfield/Medium.hh"
00010
00011 namespace Garfield {
00013
00014 class MediumSilicon : public Medium {
00015  public:
00017   MediumSilicon();
00019   virtual ~MediumSilicon() {}
00020
00021   bool IsSemiconductor() const override { return true; }
00022
00024   void SetDoping(const char type, const double c);
00026   void GetDoping(char& type, double& c) const;
00027
00029   void SetTrapCrossSection(const double ecs, const double hcs);
00031   void SetTrapDensity(const double n);
00033   void SetTrappingTime(const double etau, const double htau);
00034
00036   void SetLowFieldMobility(const double mue, const double muh);
00043   void SetLatticeMobilityModel(const std::string& model);
00045   void SetLatticeMobilityModelMinimos();
00047   void SetLatticeMobilityModelSentaurus();
00049   void SetLatticeMobilityModelReggiani();
00050
00052   void SetDopingMobilityModelMinimos();
00054   void SetDopingMobilityModelMasetti();
00055
00057   void SetSaturationVelocity(const double vsate, const double vsath);
00059   void SetSaturationVelocityModelMinimos();
00061   void SetSaturationVelocityModelCanali();
00063   void SetSaturationVelocityModelReggiani();
00064
00073   void SetHighFieldMobilityModel(const std::string& model);
00075   void SetHighFieldMobilityModelMinimos();
00077   void SetHighFieldMobilityModelCanali();
00079   void SetHighFieldMobilityModelReggiani();
00081   void SetHighFieldMobilityModelConstant();
00082
00090   void SetImpactIonisationModel(const std::string& model);
00092   void SetImpactIonisationModelVanOverstraetenDeMan();
00094   void SetImpactIonisationModelGrant();
00096   void SetImpactIonisationModelMassey();
00098   void SetImpactIonisationModelOkutoCrowell();
00099
00101   void SetDiffusionScaling(const double d) { m_diffScale = d; }
00102
00103   // Microscopic transport properties
00104   bool SetMaxElectronEnergy(const double e);
00105   double GetMaxElectronEnergy() const { return m_cb[2].eFinal; }
00106
00107   bool Initialise();
00108
00109   // When enabled, the scattering rates table is written to file
00110   // when loaded into memory.
00111   void EnableScatteringRateOutput(const bool on = true) { m_cfOutput = on; }
00112   void EnableNonParabolicity(const bool on = true) { m_nonParabolic = on; }
00113   void EnableFullBandDensityOfStates(const bool on = true) {
00114     m_fullBandDos = on;
00115   }
00116   void EnableAnisotropy(const bool on = true) { m_anisotropic = on; }
00117
```

```
00118   // Electron transport parameters
00119   bool ElectronVelocity(const double ex, const double ey, const double ez,
00120                         const double bx, const double by, const double bz,
00121                         double& vx, double& vy, double& vz) override;
00122   bool ElectronTownsend(const double ex, const double ey, const double ez,
00123                         const double bx, const double by, const double bz,
00124                         double& alpha) override;
00125   bool ElectronAttachment(const double ex, const double ey, const double ez,
00126                           const double bx, const double by, const double bz,
00127                           double& eta) override;
00128   double ElectronMobility() override { return m_eMu; }
00129
00130   // Hole transport parameters
00131   bool HoleVelocity(const double ex, const double ey, const double ez,
00132                     const double bx, const double by, const double bz,
00133                     double& vx, double& vy, double& vz) override;
00134   bool HoleTownsend(const double ex, const double ey, const double ez,
00135                     const double bx, const double by, const double bz,
00136                     double& alpha) override;
00137   bool HoleAttachment(const double ex, const double ey, const double ez,
00138                       const double bx, const double by, const double bz,
00139                       double& eta) override;
00140   double HoleMobility() override { return m_hMu; }
00141   // Get the electron energy (and its gradient)
00142   // for a given (crystal) momentum
00143   double GetElectronEnergy(const double px, const double py, const double pz,
00144                            double& vx, double& vy, double& vz,
00145                            const int band = 0) override;
00146   // Get the electron (crystal) momentum for a given kinetic energy
00147   void GetElectronMomentum(const double e, double& px, double& py, double& pz,
00148                            int& band) override;
00149
00150   // Get the null-collision rate [ns-1]
00151   double GetElectronNullCollisionRate(const int band) override;
00152   // Get the (real) collision rate [ns-1] at a given electron energy
00153   double GetElectronCollisionRate(const double e, const int band) override;
00154   // Sample the collision type
00155   bool ElectronCollision(const double e, int& type, int& level, double& e1,
00156                          double& dx, double& dy, double& dz,
00157                          std::vector<Secondary>& secondaries,
00158                          int& band) override;
00159
00160   // Reset the collision counters
00161   void ResetCollisionCounters();
00162   // Get the total number of electron collisions
00163   unsigned int GetNumberOfElectronCollisions() const;
00164   // Get number of scattering rate terms
00165   unsigned int GetNumberOfLevels() const;
00166   // Get number of collisions for a specific level
00167   unsigned int GetNumberOfElectronCollisions(const unsigned int level) const;
00168
00169   unsigned int GetNumberOfElectronBands() const;
00170   int GetElectronBandPopulation(const int band);
00171
00172   bool GetOpticalDataRange(double& emin, double& emax,
00173                            const unsigned int i = 0) override;
00174   bool GetDielectricFunction(const double e, double& eps1, double& eps2,
00175                              const unsigned int i = 0) override;
00176
00177   void ComputeSecondaries(const double e0, double& ee, double& eh);
00178
00179  private:
00180   enum class LatticeMobility { Sentaurus = 0, Minimos, Reggiani };
00181   enum class DopingMobility { Minimos = 0, Masetti };
00182   enum class SaturationVelocity { Minimos = 0, Canali, Reggiani };
00183   enum class HighFieldMobility { Minimos = 0, Canali, Reggiani, Constant };
00184   enum class ImpactIonisation { VanOverstraeten = 0, Grant, Massey, Okuto };
00185
00186   std::mutex m_mutex;
00187
00188   // Diffusion scaling factor
00189   double m_diffScale = 1.;
00190
00191   double m_bandGap = 1.12;
00192   // Doping
00193   char m_dopingType = 'i';
00194   // Doping concentration
00195   double m_cDop = 0.;
00196
00197   // Lattice mobility
00198   double m_eMuLat = 1.35e-6;
00199   double m_hMuLat = 0.45e-6;
00200   // Low-field mobility
00201   double m_eMu = 1.35e-6;
00202   double m_hMu = 0.45e-6;
00203   // High-field mobility parameters
00204   double m_eBetaCanali = 1.109;
```

```
00205   double m_hBetaCanali = 1.213;
00206   double m_eBetaCanaliInv = 1. / 1.109;
00207   double m_hBetaCanaliInv = 1. / 1.213;
00208   // Saturation velocity
00209   double m_eVs = 1.02e-2;
00210   double m_hVs = 0.72e-2;
00211   // Ratio between low-field mobility and saturation velocity
00212   double m_eRs = 1.35e-6 / 1.02e-2;
00213   double m_hRs = 0.45e-6 / 0.72e-2;
00214   // Hall factor
00215   double m_eHallFactor = 1.15;
00216   double m_hHallFactor = 0.7;
00217
00218   // Trapping parameters
00219   double m_eTrapCs = 1.e-15;
00220   double m_hTrapCs = 1.e-15;
00221   double m_eTrapDensity = 0.;
00222   double m_hTrapDensity = 0.;
00223   double m_eTrapTime = 0.;
00224   double m_hTrapTime = 0.;
00225   double m_eTrapRate = 0.;
00226   double m_hTrapRate = 0.;
00227   int m_trappingModel = 0;
00228
00229   // Impact ionisation parameters
00230   double m_eImpactA0 = 3.318e5;
00231   double m_eImpactA1 = 0.703e6;
00232   double m_eImpactA2 = 0.;
00233   double m_eImpactB0 = 1.135e6;
00234   double m_eImpactB1 = 1.231e6;
00235   double m_eImpactB2 = 0.;
00236   double m_hImpactA0 = 1.582e6;
00237   double m_hImpactA1 = 0.671e6;
00238   double m_hImpactB0 = 2.036e6;
00239   double m_hImpactB1 = 1.693e6;
00240
00241   // Models
00242   bool m_hasUserMobility = false;
00243   bool m_hasUserSaturationVelocity = false;
00244   LatticeMobility m_latticeMobilityModel = LatticeMobility::Sentaurus;
00245   DopingMobility m_dopingMobilityModel = DopingMobility::Masetti;
00246   SaturationVelocity m_saturationVelocityModel = SaturationVelocity::Canali;
00247   HighFieldMobility m_highFieldMobilityModel = HighFieldMobility::Canali;
00248   ImpactIonisation m_impactIonisationModel = ImpactIonisation::VanOverstraeten;
00249
00250   // Options
00251   bool m_cfOutput = false;
00252   bool m_nonParabolic = true;
00253   bool m_fullBandDos = false;
00254   bool m_anisotropic = true;
00255
00256   struct Band {
00257     int nEnergySteps = 2000;
00258     double eStep = 0.;
00259     double invStep = 0.;
00260     // Energy range of scattering rates.
00261     double eFinal;
00262     // Energy offset [eV].
00263     double eMin = 0.;
00264     // Index corresponding to the energy offset.
00265     int iMin = 0;
00266     // Density of states.
00267     std::vector<double> dos;
00268     // Multiplicity (number of valleys).
00269     int nValleys = 1;
00270     // Longitudinal mass.
00271     double mL = 1.;
00272     // Transverse mass.
00273     double mT = 1.;
00274     // Conduction effective mass.
00275     double mC = 1.;
00276     // Non-parabolicity parameter [1/eV].
00277     double alpha = 0.;
00278     // Null-collision rate.
00279     double cfNull;
00280     // Total scattering rate.
00281     std::vector<double> cfTot;
00282     // Scattering rates.
00283     std::vector<std::vector<double> > cf;
00284     std::vector<double> energyLoss;
00285     // Cross-section type.
00286     std::vector<int> scatType;
00287     // Number of scattering terms.
00288     int nLevels = 0;
00289   };
00290
00291   // Conduction bands.
```

```
00292   std::array<Band, 3> m_cb;
00293   std::vector<size_t> m_cbIndex;
00294   // Valence band.
00295   Band m_vb;
00296
00297   // Collision counters
00298   unsigned int m_nCollElectronAcoustic = 0;
00299   unsigned int m_nCollElectronOptical = 0;
00300   unsigned int m_nCollElectronIntervalley = 0;
00301   unsigned int m_nCollElectronImpurity = 0;
00302   unsigned int m_nCollElectronIonisation = 0;
00303   std::vector<unsigned int> m_nCollElectronDetailed;
00304   std::vector<unsigned int> m_nCollElectronBand;
00305
00306   // Density of states tables
00307   double m_eStepDos = 0.;
00308   double m_invStepDos = 0.;
00309   std::vector<double> m_fbDosV;
00310   std::vector<double> m_fbDosC;
00311   double m_fbDosMaxV, m_fbDosMaxC;
00312
00313   // Optical data
00314   std::string m_opticalDataFile = "OpticalData_Si.txt";
00315   std::vector<double> m_egamma;
00316   std::vector<double> m_eps1;
00317   std::vector<double> m_eps2;
00318
00319   bool Update();
00320   void UpdateLatticeMobility();
00321
00322   void UpdateDopingMobilityMinimos();
00323   void UpdateDopingMobilityMasetti();
00324
00325   void UpdateSaturationVelocity();
00326
00327   void UpdateHighFieldMobilityCanali();
00328
00329   void UpdateImpactIonisation();
00330
00331   double ElectronMobility(const double e) const;
00332   double ElectronAlpha(const double e) const;
00333
00334   double HoleMobility(const double e) const;
00335   double HoleAlpha(const double e) const;
00336
00337   bool LoadOpticalData(const std::string& filename);
00338
00339   bool ElectronScatteringRates();
00340   bool HoleScatteringRates();
00341   bool AcousticScatteringRates(const double rho, const double kbt,
00342                                const double dp, Band& band);
00343   bool OpticalScatteringRates(const double rho, const double kbt,
00344                               const double dtk, const double eph, Band& band);
00345   bool IntervalleyScatteringRates(const double rho, const double kbt,
00346                                   const double dtk, const double eph,
00347                                   Band& bndI, Band& bndF, const double zF,
00348                                   const int collType);
00349   bool IonisationRates(const std::vector<double>& p,
00350                        const std::vector<double>& eth,
00351                        const std::vector<double>& b, Band& band);
00352   bool ImpurityScatteringRates(const double kbt, Band& band);
00353
00354   void InitialiseDOS();
00355   void ComputeDOS();
00356 };
00357 }  // namespace Garfield
00358
00359 #endif
```

## 8.91 /builds/garfield/docs/source/Include/Garfield/MultiProcessInterface.hh File Reference

Namespaces

- namespace Garfield

Enumerations

- enum class Garfield::MPRunMode { Garfield::Normal = 0 , Garfield::GPUExclusive }

## 8.92 MultiProcessInterface.hh

Go to the documentation of this file.

```
00001 #ifndef G_MULTIPROCESS_INTERFACE
00002 #define G_MULTIPROCESS_INTERFACE
00003
00004 namespace Garfield {
00005
00006 // Structure to determine how multi-processing is done
00007 enum class MPRunMode { Normal = 0, GPUExclusive };
00008
00009 }  // namespace Garfield
00010
00011 #endif
```

## 8.93 /builds/garfield/docs/source/Include/Garfield/Numerics.hh File Reference

#include <array>
#include <complex>
#include <functional>
#include <vector>
#include "Garfield/FundamentalConstants.hh"

### Namespaces

- namespace Garfield
- namespace Garfield::Numerics

  Collection of numerical routines.
- namespace Garfield::Numerics::QUADPACK

  Functions for performing numerical integration (quadrature).
- namespace Garfield::Numerics::CERNLIB

  Linear algebra routines from CERNLIB.

### Functions

- constexpr std::array< double, 2 > Garfield::Numerics::GaussLegendreNodes2 ()
- constexpr std::array< double, 2 > Garfield::Numerics::GaussLegendreWeights2 ()
- constexpr std::array< double, 3 > Garfield::Numerics::GaussLegendreNodes3 ()
- constexpr std::array< double, 3 > Garfield::Numerics::GaussLegendreWeights3 ()
- constexpr std::array< double, 4 > Garfield::Numerics::GaussLegendreNodes4 ()
- constexpr std::array< double, 4 > Garfield::Numerics::GaussLegendreWeights4 ()
- constexpr std::array< double, 5 > Garfield::Numerics::GaussLegendreNodes5 ()
- constexpr std::array< double, 5 > Garfield::Numerics::GaussLegendreWeights5 ()
- constexpr std::array< double, 6 > Garfield::Numerics::GaussLegendreNodes6 ()
- constexpr std::array< double, 6 > Garfield::Numerics::GaussLegendreWeights6 ()
- void Garfield::Numerics::QUADPACK::qagi (std::function< double(double)> f, double bound, const int inf, const double epsabs, const double epsrel, double &result, double &abserr, unsigned int &status)

  Estimates an integral over a semi-infinite or infinite interval.
- void Garfield::Numerics::QUADPACK::qk15i (std::function< double(double)> f, double bound, const int inf, const double a, const double b, double &result, double &abserr, double &resabs, double &resasc)

  15-point Gauss-Kronrod integration with (semi-)infinite integration range.
- void Garfield::Numerics::QUADPACK::qk15 (std::function< double(double)> f, const double a, const double b, double &result, double &abserr, double &resabs, double &resasc)

  15-point Gauss-Kronrod integration with finite integration range.
- int Garfield::Numerics::CERNLIB::deqn (const int n, std::vector< std::vector< double > > &a, std::vector< double > &b)

  Replaces b by the solution x of Ax = b, after which A is undefined.

- int Garfield::Numerics::CERNLIB::deqinv (const int n, std::vector< std::vector< double > > &a, std::vector< double > &b)

  Replaces b by the solution x of Ax = b, and replace A by its inverse.
- void Garfield::Numerics::CERNLIB::dfact (const int n, std::vector< std::vector< double > > &a, std::vector< int > &ir, int &ifail, double &det, int &jfail)
- void Garfield::Numerics::CERNLIB::dfeqn (const int n, std::vector< std::vector< double > > &a, std::vector< int > &ir, std::vector< double > &b)
- void Garfield::Numerics::CERNLIB::dfinv (const int n, std::vector< std::vector< double > > &a, std::vector< int > &ir)
- int Garfield::Numerics::CERNLIB::dinv (const int n, std::vector< std::vector< double > > &a)

  Replace square matrix A by its inverse.
- void Garfield::Numerics::CERNLIB::cfact (const int n, std::vector< std::vector< std::complex< double > > > &a, std::vector< int > &ir, int &ifail, std::complex< double > &det, int &jfail)
- void Garfield::Numerics::CERNLIB::cfinv (const int n, std::vector< std::vector< std::complex< double > > > &a, std::vector< int > &ir)
- int Garfield::Numerics::CERNLIB::cinv (const int n, std::vector< std::vector< std::complex< double > > > &a)

  Replace square matrix A by its inverse.
- void Garfield::Numerics::CERNLIB::cfft (std::vector< std::complex< double > > &a, const int msign)
- double Garfield::Numerics::Legendre (const unsigned int n, const double x)

  Legendre polynomials.
- double Garfield::Numerics::BesselI0S (const double xx)

  Modified Bessel functions.
- double Garfield::Numerics::BesselI1S (const double xx)
- double Garfield::Numerics::BesselK0S (const double xx)
- double Garfield::Numerics::BesselK0L (const double xx)
- double Garfield::Numerics::BesselK1S (const double xx)
- double Garfield::Numerics::BesselK1L (const double xx)
- double Garfield::Numerics::Divdif (const std::vector< double > &f, const std::vector< double > &a, int nn, double x, int mm)

  C++ version of DIVDIF (CERN program library E105) which performs tabular interpolation using symmetrically placed argument points.
- double Garfield::Numerics::LinearInterpolation (const std::vector< double > &ytab, const std::vector< double > &xtab, const double x)
- bool Garfield::Numerics::Boxin2 (const std::vector< std::vector< double > > &value, const std::vector< double > &xAxis, const std::vector< double > &yAxis, const int nx, const int ny, const double xx, const double yy, double &f, const int iOrder)

  Interpolation of order 1 and 2 in an irregular rectangular two-dimensional grid.
- bool Garfield::Numerics::Boxin3 (const std::vector< std::vector< std::vector< double > > > &value, const std::vector< double > &xAxis, const std::vector< double > &yAxis, const std::vector< double > &zAxis, const int nx, const int ny, const int nz, const double xx, const double yy, const double zz, double &f, const int iOrder)

  Interpolation of order 1 and 2 in an irregular rectangular three-dimensional grid.
- bool Garfield::Numerics::LeastSquaresFit (std::function< double(double, const std::vector< double > &)> f, std::vector< double > &par, std::vector< double > &epar, const std::vector< double > &x, const std::vector< double > &y, const std::vector< double > &ey, const unsigned int nMaxIter, const double diff, double &chi2, const double eps, const bool debug, const bool verbose)

  Least-squares minimisation.

## 8.94 Numerics.hh

Go to the documentation of this file.

```
00001 #ifndef G__NUMERICS__H
00002 #define G__NUMERICS__H
00003
00004 #include <array>
00005 #include <complex>
00006 #include <functional>
00007 #include <vector>
00008
00009 #include "Garfield/FundamentalConstants.hh"
00010
00011 namespace Garfield {
00012
00014 namespace Numerics {
00015
00016 constexpr std::array<double, 2> GaussLegendreNodes2() {
00017   return {-0.577350269189625765, 0.577350269189625765};
00018 }
00019 constexpr std::array<double, 2> GaussLegendreWeights2() { return {1., 1.}; }
00020 constexpr std::array<double, 3> GaussLegendreNodes3() {
00021   return {-0.774596669241483377, 0., 0.774596669241483377};
00022 }
00023 constexpr std::array<double, 3> GaussLegendreWeights3() {
00024   return {5. / 9., 8. / 9., 5. / 9.};
00025 }
00026 constexpr std::array<double, 4> GaussLegendreNodes4() {
00027   return {-0.861136311594052575, -0.339981043584856265, 0.339981043584856265,
00028           0.861136311594052575};
00029 }
00030 constexpr std::array<double, 4> GaussLegendreWeights4() {
00031   return {0.347854845137453857, 0.652145154862546143, 0.652145154862546143,
00032           0.347854845137453857};
00033 }
00034 constexpr std::array<double, 5> GaussLegendreNodes5() {
00035   return {-0.906179845938663993, -0.538469310105683091, 0.,
00036           0.538469310105683091, 0.906179845938663993};
00037 }
00038 constexpr std::array<double, 5> GaussLegendreWeights5() {
00039   return {0.236926885056189088, 0.478628670499366468, 128. / 225.,
00040           0.478628670499366468, 0.236926885056189088};
00041 }
00042 constexpr std::array<double, 6> GaussLegendreNodes6() {
00043   return {-0.932469514203152028, -0.661209386466264514, -0.238619186083196909,
00044           0.238619186083196909,  0.661209386466264514,  0.932469514203152028};
00045 }
00046 constexpr std::array<double, 6> GaussLegendreWeights6() {
00047   return {0.171324492379170345, 0.360761573048138608, 0.467913934572691047,
00048           0.467913934572691047, 0.360761573048138608, 0.171324492379170345};
00049 }
00050
00057 namespace QUADPACK {
00058
00098 void qagi(std::function<double(double)> f, double bound, const int inf,
00099           const double epsabs, const double epsrel, double& result,
00100           double& abserr, unsigned int& status);
00101
00113 void qk15i(std::function<double(double)> f, double bound, const int inf,
00114            const double a, const double b, double& result, double& abserr,
00115            double& resabs, double& resasc);
00116
00118 void qk15(std::function<double(double)> f, const double a, const double b,
00119           double& result, double& abserr, double& resabs, double& resasc);
00120
00121 }  // namespace QUADPACK
00122
00124 namespace CERNLIB {
00125
00131 int deqn(const int n, std::vector<std::vector<double> >& a,
00132          std::vector<double>& b);
00134 int deqinv(const int n, std::vector<std::vector<double> >& a,
00135            std::vector<double>& b);
00136
00137 void dfact(const int n, std::vector<std::vector<double> >& a,
00138            std::vector<int>& ir, int& ifail, double& det, int& jfail);
00139 void dfeqn(const int n, std::vector<std::vector<double> >& a,
00140            std::vector<int>& ir, std::vector<double>& b);
00141 void dfinv(const int n, std::vector<std::vector<double> >& a,
00142            std::vector<int>& ir);
00144 int dinv(const int n, std::vector<std::vector<double> >& a);
00145
00146 void cfact(const int n, std::vector<std::vector<std::complex<double> > >& a,
00147            std::vector<int>& ir, int& ifail, std::complex<double>& det,
00148            int& jfail);
00149 void cfinv(const int n, std::vector<std::vector<std::complex<double> > >& a,
```

```
00150           std::vector<int>& ir);
00151
00153 int cinv(const int n, std::vector<std::vector<std::complex<double> > >& a);
00154
00155 void cfft(std::vector<std::complex<double> >& a, const int msign);
00156 }  // namespace CERNLIB
00157
00159 inline double Legendre(const unsigned int n, const double x) {
00160   if (std::abs(x) > 1.) return 0.;
00161   double p0 = 1.;
00162   double p1 = x;
00163   if (n == 0) return p0;
00164   if (n == 1) return p1;
00165   for (unsigned int k = 1; k < n; ++k) {
00166     p0 = ((2 * k + 1) * x * p1 - k * p0) / (k + 1);
00167     std::swap(p0, p1);
00168   }
00169   return p1;
00170 }
00171
00174 inline double BesselI0S(const double xx) {
00175   const double y = xx / 3.75;
00176   const double y2 = y * y;
00177   return 1. + 3.5156229 * y2 + 3.0899424 * y2 * y2 + 1.2067492 * pow(y2, 3) +
00178          0.2659732 * pow(y2, 4) + 0.0360768 * pow(y2, 5) +
00179          0.0045813 * pow(y2, 6);
00180 }
00181
00182 inline double BesselI1S(const double xx) {
00183   const double y = xx / 3.75;
00184   const double y2 = y * y;
00185   return xx * (0.5 + 0.87890594 * y2 + 0.51498869 * y2 * y2 +
00186                0.15084934 * pow(y2, 3) + 0.02658733 * pow(y2, 4) +
00187                0.00301532 * pow(y2, 5) + 0.00032411 * pow(y2, 6));
00188 }
00189
00190 inline double BesselK0S(const double xx) {
00191   const double y = xx / 2.;
00192   const double y2 = y * y;
00193   return -log(y) * BesselI0S(xx) - Gamma + 0.42278420 * y2 +
00194          0.23069756 * y2 * y2 + 0.03488590 * pow(y2, 3) +
00195          0.00262698 * pow(y2, 4) + 0.00010750 * pow(y2, 5) +
00196          0.00000740 * pow(y2, 6);
00197 }
00198
00199 inline double BesselK0L(const double xx) {
00200   const double y = 2. / xx;
00201   return (exp(-xx) / sqrt(xx)) *
00202          (1.25331414 - 0.07832358 * y + 0.02189568 * y * y -
00203           0.01062446 * pow(y, 3) + 0.00587872 * pow(y, 4) -
00204           0.00251540 * pow(y, 5) + 0.00053208 * pow(y, 6));
00205 }
00206
00207 inline double BesselK1S(const double xx) {
00208   const double y = xx / 2.;
00209   const double y2 = y * y;
00210   return log(y) * BesselI1S(xx) +
00211          (1. / xx) * (1. + 0.15443144 * y2 - 0.67278579 * y2 * y2 -
00212                       0.18156897 * pow(y2, 3) - 0.01919402 * pow(y2, 4) -
00213                       0.00110404 * pow(y2, 5) - 0.00004686 * pow(y2, 6));
00214 }
00215
00216 inline double BesselK1L(const double xx) {
00217   const double y = 2. / xx;
00218   return (exp(-xx) / sqrt(xx)) *
00219          (1.25331414 + 0.23498619 * y - 0.03655620 * y * y +
00220           0.01504268 * pow(y, 3) - 0.00780353 * pow(y, 4) +
00221           0.00325614 * pow(y, 5) - 0.00068245 * pow(y, 6));
00222 }
00223
00226 double Divdif(const std::vector<double>& f, const std::vector<double>& a,
00227               int nn, double x, int mm);
00228
00229 double LinearInterpolation(const std::vector<double>& ytab,
00230                            const std::vector<double>& xtab, const double x);
00231
00234 bool Boxin2(const std::vector<std::vector<double> >& value,
00235             const std::vector<double>& xAxis, const std::vector<double>& yAxis,
00236             const int nx, const int ny, const double xx, const double yy,
00237             double& f, const int iOrder);
00240 bool Boxin3(const std::vector<std::vector<std::vector<double> > >& value,
00241             const std::vector<double>& xAxis, const std::vector<double>& yAxis,
00242             const std::vector<double>& zAxis, const int nx, const int ny,
00243             const int nz, const double xx, const double yy, const double zz,
00244             double& f, const int iOrder);
00245
00247 bool LeastSquaresFit(
```

```
00248     std::function<double(double, const std::vector<double>&)> f,
00249     std::vector<double>& par, std::vector<double>& epar,
00250     const std::vector<double>& x, const std::vector<double>& y,
00251     const std::vector<double>& ey, const unsigned int nMaxIter,
00252     const double diff, double& chi2, const double eps, const bool debug,
00253     const bool verbose);
00254
00255 }  // namespace Numerics
00256 }  // namespace Garfield
00257
00258 #endif
```

## 8.95 /builds/garfield/docs/source/Include/Garfield/OpticalData.hh File Reference

#include <string>

### Classes

- class Garfield::OpticalData

  Photoabsorption cross-sections for some gases.

### Namespaces

- namespace Garfield

## 8.96 OpticalData.hh

Go to the documentation of this file.

```
00001 #ifndef G__OPTICAL__DATA__H
00002 #define G__OPTICAL__DATA__H
00003
00004 #include <string>
00005
00006 namespace Garfield {
00007
00009
00010 class OpticalData {
00011  public:
00013   OpticalData() = default;
00015   ~OpticalData() = default;
00016
00018   static bool IsAvailable(const std::string& material);
00020   static bool PhotoabsorptionCrossSection(const std::string& material,
00021                                           const double energy, double& cs,
00022                                           double& eta);
00024   static double PhotoabsorptionCrossSection(const std::string& material,
00025                                             const double energy);
00027   static double PhotoionisationYield(const std::string& material,
00028                                      const double energy);
00029
00030  private:
00031   static constexpr double OscToPacs = 8.067283e-18;
00032   static constexpr double Mbarn = 1.e-18;
00033
00034   static bool PhotoAbsorptionCsNeon(const double e, double& cs, double& eta);
00035   static bool PhotoAbsorptionCsArgon(const double e, double& cs, double& eta);
00036
00037   static bool PhotoAbsorptionCsCO2(const double e, double& cs, double& eta);
00038
00039   static bool PhotoAbsorptionCsMethane(const double e, double& cs, double& eta);
00040   static bool PhotoAbsorptionCsEthane(const double e, double& cs, double& eta);
00041   static bool PhotoAbsorptionCsButane(const double e, double& cs, double& eta);
00042   static bool PhotoAbsorptionCsAcetylene(const double e, double& cs,
00043                                          double& eta);
00044   static bool PhotoAbsorptionCsCF4(const double e, double& cs, double& eta);
00045
00046   static bool PhotoAbsorptionCsNitrogen(const double e, double& cs,
00047                                         double& eta);
00048 };
00049 }  // namespace Garfield
00050
00051 #endif
```

## 8.97 /builds/garfield/docs/source/Include/Garfield/Plotting.hh File Reference

#include "Garfield/PlottingEngine.hh"

Namespaces

- namespace Garfield

Functions

- void Garfield::SetDefaultStyle ()
- void Garfield::SetSerif ()
- void Garfield::SetSansSerif ()

Variables

- PlottingEngine Garfield::plottingEngine

## 8.98 Plotting.hh

Go to the documentation of this file.

```
00001 #ifndef G__PLOTTING__H
00002 #define G__PLOTTING__H
00003
00004 #include "Garfield/PlottingEngine.hh"
00005
00006 namespace Garfield {
00007
00008 extern PlottingEngine plottingEngine;
00009
00010 inline void SetDefaultStyle() { plottingEngine.SetDefaultStyle(); }
00011
00012 inline void SetSerif() { plottingEngine.SetSerif(); }
00013
00014 inline void SetSansSerif() { plottingEngine.SetSansSerif(); }
00015
00016 }  // namespace Garfield
00017
00018 #endif
```

## 8.99 /builds/garfield/docs/source/Include/Garfield/PlottingEngine.hh File Reference

#include <TStyle.h>

Classes

- class Garfield::PlottingEngine

  Plotting style.

Namespaces

- namespace Garfield

## 8.100 PlottingEngine.hh

Go to the documentation of this file.

```
00001 #ifndef G__PLOTTING__ENGINE__H
00002 #define G__PLOTTING__ENGINE__H
00003
00004 #include <TStyle.h>
00005
00006 namespace Garfield {
00007
00009
00010 class PlottingEngine {
00011  public:
00013   PlottingEngine();
00015   ~PlottingEngine() = default;
```

```
00016
00018   void SetSerif() { m_serif = true; }
00020   void SetSansSerif() { m_serif = false; }
00021
00023   void SetPalette(int ncol) { m_palette = ncol; }
00024
00026   void SetDefaultStyle();
00027
00028  private:
00029   std::string m_className = "PlottingEngine";
00030
00031   bool m_serif = false;
00032   int m_palette = 0;
00033
00034   TStyle m_garfieldStyle;
00035 };
00036 }  // namespace Garfield
00037
00038 #endif
```

## 8.101 /builds/garfield/docs/source/Include/Garfield/Polygon.hh File Reference

#include <vector>

### Namespaces

- namespace Garfield
- namespace Garfield::Polygon

### Functions

- void Garfield::Polygon::Inside (const std::vector< double > &xpl, const std::vector< double > &ypl, const double x, const double y, bool &inside, bool &edge)

  Determine whether the point (x, y) is located inside of the polygon (xpl, ypl).
- double Garfield::Polygon::Area (const std::vector< double > &xp, const std::vector< double > &yp)

  Determine the (signed) area of a polygon.
- bool Garfield::Polygon::NonTrivial (const std::vector< double > &xp, const std::vector< double > &yp)

  Check whether a set of points builds a non-trivial polygon.
- void Garfield::Polygon::EliminateButterflies (std::vector< double > &xp, std::vector< double > &yp, std::vector< double > &zp)

  Try to eliminate "butterflies" (crossing of two adjacent segments of a polygon), by point exchanges.

## 8.102 Polygon.hh

Go to the documentation of this file.

```
00001 #ifndef G_POLYGON_H
00002 #define G_POLYGON_H
00003
00004 #include <vector>
00005
00006 namespace Garfield {
00007
00008 namespace Polygon {
00009
00012 void Inside(const std::vector<double>& xpl, const std::vector<double>& ypl,
00013             const double x, const double y, bool& inside, bool& edge);
00014
00016 double Area(const std::vector<double>& xp, const std::vector<double>& yp);
00017
00019 bool NonTrivial(const std::vector<double>& xp, const std::vector<double>& yp);
00020
00023 void EliminateButterflies(std::vector<double>& xp, std::vector<double>& yp,
00024                           std::vector<double>& zp);
00025 }  // namespace Polygon
00026
00027 }  // namespace Garfield
00028
00029 #endif
```

## 8.103 /builds/garfield/docs/source/Include/Garfield/QuadTree.hh File Reference

#include <cstddef>
#include <tuple>
#include <vector>

Classes

- class Garfield::QuadTree

  Quadtree search.

Namespaces

- namespace Garfield

## 8.104 QuadTree.hh

Go to the documentation of this file.

```
00001 #ifndef G__QUAD__TREE__H
00002 #define G__QUAD__TREE__H
00003
00004 #include <cstddef>
00005 #include <tuple>
00006 #include <vector>
00007
00008 namespace Garfield {
00009
00011
00012 class QuadTree {
00013  public:
00015   QuadTree(const double x0, const double y0, const double hx, const double hy);
00016
00018   ~QuadTree();
00019
00021   void InsertMeshNode(const double x, const double y, const int index);
00022
00024   void InsertMeshElement(const double bb[4], const int index);
00025
00027   const std::vector<int>& GetElementsInBlock(const double x,
00028                                              const double y) const;
00029
00030  private:
00031   static std::vector<int> emptyBlock;
00032
00033   // Centre of this tree node.
00034   double m__x0, m__y0;
00035   // Half-width in x and y of this tree node.
00036   double m__hx, m__hy;
00037   // Bounding box of the tree node.
00038   double m__xmin, m__ymin, m__xmax, m__ymax;
00039
00040   // Pointers to child quadrants.
00041   QuadTree* children[4];
00042
00043   // Children follow a predictable pattern to make accesses simple.
00044   // Here, - means less than 'origin' in that dimension, + means greater than.
00045   // child: 0 1 2 3
00046   // x:     - - + +
00047   // y:     - + - +
00048
00049   std::vector<std::tuple<float, float, int> > nodes;
00050   std::vector<int> elements;
00051
00052   static const size__t BlockCapacity = 10;
00053
00054   // Check if the given box overlaps with this tree node.
00055   bool DoesBoxOverlap(const double bb[4]) const;
00056
00057   int GetQuadrant(const double x, const double y) const;
00058
00059   // Check if this tree node is a leaf or intermediate node.
00060   bool IsLeafNode() const;
00061
00062   // Get a block containing the input point
00063   const QuadTree* GetBlockFromPoint(const double x, const double y) const;
00064
00065   // A helper function used by the function above.
00066   // Called recursively on the child nodes.
```

```
00067   const QuadTree* GetBlockFromPointHelper(const double x, const double y) const;
00068 };
00069 }  // namespace Garfield
00070
00071 #endif
```

## 8.105 /builds/garfield/docs/source/Include/Garfield/Random.hh File Reference

#include <cmath>
#include <functional>
#include "Garfield/FundamentalConstants.hh"

### Classes

- class Garfield::Random

### Namespaces

- namespace Garfield

### Functions

- double Garfield::RndmUniform ()

  Draw a random number uniformly distributed in the range [0, 1).
- double Garfield::RndmUniformPos ()

  Draw a random number uniformly distributed in the range (0, 1).
- double Garfield::RndmGaussian ()

  Draw a Gaussian random variate with mean zero and standard deviation one.
- double Garfield::RndmGaussian (const double mu, const double sigma)

  Draw a Gaussian random variate with mean mu and standard deviation sigma.
- double Garfield::RndmLorentzian (const double mu, const double gamma)

  Draw a Lorentzian random variate with mean mu and half-width at half maximum gamma.
- double Garfield::RndmVoigt (const double mu, const double sigma, const double gamma)

  Draw a random number according to a Voigt function with mean mu.
- unsigned int Garfield::RndmYuleFurry (const double mean)

  Draw a random number from a geometric distribution.
- double Garfield::RndmPolya (const double theta)

  Draw a Polya distributed random number.
- double Garfield::RndmLandau ()

  Draw a random number from a Landau distribution.
- double Garfield::RndmVavilov (const double rkappa, const double beta2)

  Draw a random number from a Vavilov distribution.
- int Garfield::RndmPoisson (const double mean)

  Draw a random number from a Poisson distribution.
- double Garfield::RndmHeedWF (const double w, const double f)

  Draw a random energy needed to create a single electron in a material asymptotic work function W and Fano factor F, according to Igor Smirnov's phenomenological model.
- void Garfield::RndmDirection (double &dx, double &dy, double &dz, const double length=1.)

  Draw a random (isotropic) direction vector.

## 8.106 Random.hh

Go to the documentation of this file.

```
00001 #ifndef G_RANDOM_H
00002 #define G_RANDOM_H
00003
00004 #include <cmath>
00005 #include <functional>
00006
00007 #include "Garfield/FundamentalConstants.hh"
00008
00009 namespace Garfield {
00010
00011 class Random {
00012  public:
00013   Random() = default;
00014   template <typename T>
00015   static inline void SetEngine(T engine) {
00016     draw = std::bind(&T::Draw, engine);
00017   }
00018   inline double static Draw() noexcept { return draw(); };
00019
00020  private:
00021   static std::function<double(void)> draw;
00022 };
00023
00025 inline double RndmUniform() { return Random::Draw(); }
00026
00028 inline double RndmUniformPos() {
00029   double r = RndmUniform();
00030   while (r <= 0.) r = RndmUniform();
00031   return r;
00032 }
00033
00035 inline double RndmGaussian() {
00036   static bool cached = false;
00037   static double u = 0.;
00038   if (cached) {
00039     cached = false;
00040     return u;
00041   }
00042   // Box-Muller algorithm
00043   u = 2. * RndmUniform() - 1.;
00044   double v = 2. * RndmUniform() - 1.;
00045   double r2 = u * u + v * v;
00046   while (r2 > 1.) {
00047     u = 2. * RndmUniform() - 1.;
00048     v = 2. * RndmUniform() - 1.;
00049     r2 = u * u + v * v;
00050   }
00051   const double p = sqrt(-2. * log(r2) / r2);
00052   u *= p;
00053   cached = true;
00054   return v * p;
00055 }
00056
00058 inline double RndmGaussian(const double mu, const double sigma) {
00059   return mu + sigma * RndmGaussian();
00060 }
00061
00064 inline double RndmLorentzian(const double mu, const double gamma) {
00065   return mu + gamma * tan(Pi * (RndmUniform() - 0.5));
00066 }
00067
00072 inline double RndmVoigt(const double mu, const double sigma,
00073                         const double gamma) {
00074   if (sigma <= 0.) return RndmLorentzian(mu, gamma);
00075   const double a = gamma / (Sqrt2 * sigma);
00076   const double x = RndmLorentzian(0., a) + RndmGaussian(0., 1. / Sqrt2);
00077   return mu + x * Sqrt2 * sigma;
00078 }
00079
00081 inline unsigned int RndmYuleFurry(const double mean) {
00082   if (mean <= 0.) return 0;
00083   return 1 + static_cast<unsigned int>(std::log(RndmUniformPos()) /
00084                                        std::log1p(-1. / mean));
00085   /*
00086   const double u = RndmUniform();
00087   double p = 1. / mean;
00088   const double q = 1. - p;
00089   double sum = p;
00090   unsigned int k = 1;
00091   while (sum < u) {
00092     ++k;
00093     p *= q;
00094     sum += p;
```

```
00095   }
00096   return k;
00097   */
00098 }
00099
00101 inline double RndmPolya(const double theta) {
00102   // Algorithm from Review of Particle Physics
00103   // C. Amsler et al, Phys. Lett. B 667 (2008)
00104   if (theta <= 0.) return -log(RndmUniformPos());
00105   const double c = 3 * (theta + 1.) - 0.75;
00106   double u1, u2, v1, v2, v3;
00107   double x;
00108   while (1) {
00109     u1 = RndmUniformPos();
00110     v1 = u1 * (1. - u1);
00111     if (v1 == 0.) continue;
00112     v2 = (u1 - 0.5) * sqrt(c / v1);
00113     x = theta + v2;
00114     if (x <= 0.) continue;
00115     u2 = RndmUniformPos();
00116     v3 = 64 * u2 * u2 * pow(v1, 3);
00117     if (v3 <= 1. - 2 * v2 * v2 / x ||
00118         log(v3) <= 2 * (theta * log(x / theta) - v2)) {
00119       return x / (theta + 1.);
00120     }
00121   }
00122 }
00123
00125 double RndmLandau();
00127 double RndmVavilov(const double rkappa, const double beta2);
00128
00130 int RndmPoisson(const double mean);
00131
00135 double RndmHeedWF(const double w, const double f);
00136
00138 inline void RndmDirection(double& dx, double& dy, double& dz,
00139                           const double length = 1.) {
00140   const double phi = TwoPi * RndmUniform();
00141   const double ctheta = 2 * RndmUniform() - 1.;
00142   const double stheta = sqrt(1. - ctheta * ctheta);
00143   dx = length * cos(phi) * stheta;
00144   dy = length * sin(phi) * stheta;
00145   dz = length * ctheta;
00146 }
00147 }  // namespace Garfield
00148
00149 #endif
```

## 8.107 /builds/garfield/docs/source/Include/Garfield/RandomEngine.hh File Reference

Classes

- class Garfield::RandomEngine< Engine, SeedType >

  Abstract base class for random number generators.

Namespaces

- namespace Garfield

## 8.108 RandomEngine.hh

Go to the documentation of this file.

```
00001 #ifndef G__RANDOM__ENGINE__H
00002 #define G__RANDOM__ENGINE__H
00003
00004 namespace Garfield {
00005
00007 template <typename Engine, typename SeedType>
00008 class RandomEngine {
00009  public:
00010   using seed__t = SeedType;
00011   using engine__t = Engine;
00012   RandomEngine() = default;
00013   RandomEngine(const SeedType& seed) { this->SetSeed(seed); }
00015   inline void SetSeed(const seed__t& seed) {
00016     m__seed = seed;
00017     static__cast<Engine*>(this)->SetSeed(m__seed);
00018   }
00020   inline double operator()() { return Draw(); }
00022   inline double Draw() { return static__cast<Engine*>(this)->Draw(); }
```

```
00024   inline seed_t GetSeed() { return static_cast<Engine*>(this)->GetSeed(); }
00026   inline void Print() { return static_cast<Engine*>(this)->Print(); }
00027
00028  protected:
00029   seed_t m_seed{0};
00030 };
00031
00032 }  // namespace Garfield
00033 #endif
```

## 8.109 /builds/garfield/docs/source/Include/Garfield/RandomEngineRoot.hh File Reference

#include <TRandom3.h>
#include "Garfield/RandomEngine.hh"

Classes

- class Garfield::RandomEngineRoot

Namespaces

- namespace Garfield

## 8.110 RandomEngineRoot.hh

Go to the documentation of this file.

```
00001 #ifndef G_RANDOM_ENGINE_ROOT_H
00002 #define G_RANDOM_ENGINE_ROOT_H
00003
00004 #include <TRandom3.h>
00005
00006 #include "Garfield/RandomEngine.hh"
00007
00008 namespace Garfield {
00009
00010 class RandomEngineRoot : public RandomEngine<RandomEngineRoot, UInt_t> {
00011  public:
00012   RandomEngineRoot() { m_rng.SetSeed(); };
00013   RandomEngineRoot(const UInt_t& seed) : RandomEngine(seed) {}
00014   inline double Draw() { return m_rng.Rndm(); }
00015   inline void SetSeed(const seed_t& seed) { m_rng.SetSeed(seed); }
00016   inline seed_t GetSeed() { return m_seed; }
00017   void Print();
00018
00019  private:
00020   TRandom3 m_rng;
00021 };
00022
00023 }  // namespace Garfield
00024
00025 #endif
```

## 8.111 /builds/garfield/docs/source/Include/Garfield/RandomEngineSTL.hh File Reference

#include <random>
#include "Garfield/RandomEngine.hh"

Classes

- class Garfield::RandomEngineSTL

Namespaces

- namespace Garfield

## 8.112 RandomEngineSTL.hh

Go to the documentation of this file.

```
00001 #ifndef G__RANDOM__ENGINE__STL__H
00002 #define G__RANDOM__ENGINE__STL__H
00003
00004 #include <random>
00005
00006 #include "Garfield/RandomEngine.hh"
00007
00008 namespace Garfield {
00009
00010 class RandomEngineSTL
00011     : public RandomEngine<RandomEngineSTL, std::mt19937::result_type> {
00012  public:
00013   RandomEngineSTL() = default;
00014   RandomEngineSTL(const std::mt19937::result_type& seed) : RandomEngine(seed) {}
00015   inline double Draw() {
00016     return std::generate_canonical<double, std::numeric_limits<double>::digits>(
00017         m_rng);
00018   }
00019   inline void SetSeed(const seed_t& seed) { m_rng.seed(seed); }
00020   inline seed_t GetSeed() { return m_seed; }
00021   void Print();
00022
00023  private:
00024   std::mt19937 m_rng{std::random_device{}()};
00025 };
00026
00027 }  // namespace Garfield
00028
00029 #endif
```

## 8.113 /builds/garfield/docs/source/Include/Garfield/Sensor.hh File Reference

#include "Garfield/HelperMacros.hh"
#include <array>
#include <functional>
#include <mutex>
#include <tuple>
#include <utility>
#include <vector>
#include "Garfield/Component.hh"

### Namespaces

- namespace Garfield

### Macros

- #define ___GPUCONST___

### Functions

- class Garfield::GARFIELD_CLASS_NAME (Sensor)

  Sensor

### 8.113.1 Macro Definition Documentation

#### ___GPUCONST___

#define ___GPUCONST___
Definition at line 37 of file Sensor.hh.

## 8.114 Sensor.hh

Go to the documentation of this file.

```
00001 // Include this header if we're compiling with the GPU or this is the first time
```

```
00002 // without
00003 #if defined(___GPUCOMPILE___) || !defined(G_SENSOR_H)
00004
00005 #if !defined(___GPUCOMPILE___) && !defined(G_SENSOR_H)
00006 #define G_SENSOR_H
00007 #endif
00008
00009 #include "Garfield/HelperMacros.hh"
00010
00011 #ifdef ___GPUCOMPILE___
00012
00013 #else
00014 #include <array>
00015 #include <functional>
00016 #include <mutex>
00017 #include <tuple>
00018 #include <utility>
00019 #include <vector>
00020
00021 #include "Garfield/Component.hh"
00022 #endif
00023
00024 class TPad;
00025
00026 namespace Garfield {
00027
00028 #if defined(___GPUCONST___)
00029 #undef ___GPUCONST___
00030 #endif
00031
00032 // setup class names depending on if this is compiling the GPU static version or
00033 // not
00034 #ifdef ___GPUCOMPILE___
00035 #define ___GPUCONST___ const
00036 #else
00037 #define ___GPUCONST___
00038 class SensorGPU;
00039 class Shaper;
00040 #endif
00041
00043
00044 class GARFIELD_CLASS_NAME(Sensor) {
00045  public:
00047   GARFIELD_CLASS_NAME(Sensor)() = default;
00049   ~GARFIELD_CLASS_NAME(Sensor)() = default;
00050
00051 #ifndef ___GPUCOMPILE___
00053   Sensor(Component* comp);
00055   void AddComponent(Component* comp);
00057   size_t GetNumberOfComponents() const { return m_components.size(); }
00059   Component* GetComponent(const unsigned int i);
00061   void EnableComponent(const unsigned int i, const bool on);
00063   void EnableMagneticField(const unsigned int i, const bool on);
00065   bool HasMagneticField() const;
00066
00068   void AddElectrode(Component* comp, const std::string& label);
00070   size_t GetNumberOfElectrodes() const { return m_electrodes.size(); }
00072   void ClearElectrodes();
00074   void Clear();
00075
00077   void ElectricField(const double x, const double y, const double z, double& ex,
00078                      double& ey, double& ez, double& v, Medium*& medium,
00079                      int& status);
00080 #endif
00082   ___DEVICE___
00083   void ElectricField(const double x, const double y, const double z, double& ex,
00084                      double& ey, double& ez,
00085                      GARFIELD_CLASS_NAME(Medium) * &medium,
00086                      int& status) ___GPUCONST___;
00087
00088 #ifndef ___GPUCOMPILE___
00090   void MagneticField(const double x, const double y, const double z, double& bx,
00091                      double& by, double& bz, int& status);
00092
00094   void WeightingField(const double x, const double y, const double z,
00095                       double& wx, double& wy, double& wz,
00096                       const std::string& label);
00098   double WeightingPotential(const double x, const double y, const double z,
00099                             const std::string& label);
00100
00102   double DelayedWeightingPotential(const double x, const double y,
00103                                    const double z, const double t,
00104                                    const std::string& label);
00105
00107   Medium* GetMedium(const double x, const double y, const double z);
00108
00110   void EnableDebugging(const bool on = true) { m_debug = on; }
```

```
00111
00113   bool SetArea(const bool verbose = false);
00115   bool SetArea(const double xmin, const double ymin, const double zmin,
00116               const double xmax, const double ymax, const double zmax);
00118   bool GetArea(double& xmin, double& ymin, double& zmin, double& xmax,
00119               double& ymax, double& zmax);
00120 #endif
00122   ___DEVICE___
00123   bool IsInArea(const double x, const double y, const double z) ___GPUCONST___;
00124
00125 #ifndef ___GPUCOMPILE___
00127   bool IsInside(const double x, const double y, const double z);
00128
00130   bool GetVoltageRange(double& vmin, double& vmax);
00131
00133   void NewSignal() { ++m_nEvents; }
00135   void ClearSignal();
00136 #else
00138   ___device___ void AddSignal(const double q, const double t0, const double t1,
00139                               const double x0, const double y0, const double z0,
00140                               const double x1, const double y1, const double z1,
00141                               const bool integrateWeightingField,
00142                               const bool useWeightingPotential,
00143                               const int particle_idx);
00144 #endif
00145
00146 #ifndef ___GPUCOMPILE___
00152   void SetTimeWindow(const double tstart, const double tstep,
00153                      const unsigned int nsteps);
00155   void GetTimeWindow(double& tstart, double& tstep,
00156                      unsigned int& nsteps) const {
00157     tstart = m_tStart;
00158     tstep = m_tStep;
00159     nsteps = m_nTimeBins;
00160   }
00161
00163   void EnableDelayedSignal(const bool on = true) { m_delayedSignal = on; }
00165   void SetDelayedSignalTimes(const std::vector<double>& ts);
00170   void SetDelayedSignalAveragingOrder(const unsigned int navg) {
00171     m_nAvgDelayedSignal = navg;
00172   }
00173
00175   void SetSignal(const std::string& label, const unsigned int bin,
00176                  const double signal);
00178   void SetSignal(const std::string& label, const std::vector<double>& ts,
00179                  const std::vector<double>& is);
00181   double GetSignal(const std::string& label, const unsigned int bin);
00183   double GetSignal(const std::string& label, const unsigned int bin,
00184                    const int comp);
00186   double GetPromptSignal(const std::string& label, const unsigned int bin);
00188   double GetDelayedSignal(const std::string& label, const unsigned int bin);
00190   double GetElectronSignal(const std::string& label, const unsigned int bin);
00192   double GetIonSignal(const std::string& label, const unsigned int bin);
00194   double GetDelayedElectronSignal(const std::string& label,
00195                                   const unsigned int bin);
00197   double GetDelayedIonSignal(const std::string& label, const unsigned int bin);
00199   double GetInducedCharge(const std::string& label);
00200
00202   void SetTransferFunction(std::function<double(double)>);
00204   void SetTransferFunction(const std::vector<double>& times,
00205                            const std::vector<double>& values);
00207   void SetTransferFunction(Shaper& shaper);
00209   double GetTransferFunction(const double t);
00211   void PrintTransferFunction();
00213   void PlotTransferFunction();
00216   void EnableTransferFunctionCache(const bool on = true) {
00217     m_cacheTransferFunction = on;
00218   }
00221   bool ConvoluteSignal(const std::string& label, const bool fft = false);
00223   bool ConvoluteSignals(const bool fft = false);
00225   bool IntegrateSignal(const std::string& label);
00227   bool IntegrateSignals();
00229   bool IsIntegrated(const std::string& label) const;
00230
00236   bool DelayAndSubtractFraction(const double td, const double f);
00238   void SetNoiseFunction(double (*f)(double t));
00240   void AddNoise(const bool total = true, const bool electron = false,
00241                 const bool ion = false);
00250   void AddWhiteNoise(const std::string& label, const double enc,
00251                      const bool poisson = true, const double q0 = 1.);
00253   void AddWhiteNoise(const double enc, const bool poisson = true,
00254                      const double q0 = 1.);
00255
00261   bool ComputeThresholdCrossings(const double thr, const std::string& label,
00262                                  int& n);
00265   size_t GetNumberOfThresholdCrossings() const {
00266     return m_thresholdCrossings.size();
```

```
00267   }
00275   bool GetThresholdCrossing(const unsigned int i, double& time, double& level,
00276                             bool& rise) const;
00277
00280   void AddSignalWeightingPotential(
00281       const double q, const std::vector<double>& ts,
00282       const std::vector<std::array<double, 3> >& xs);
00285   void AddSignalWeightingPotential(
00286       const double q, const std::vector<double>& ts,
00287       const std::vector<std::array<double, 3> >& xs,
00288       const std::vector<double>& qs);
00291   void AddSignalWeightingField(const double q, const std::vector<double>& ts,
00292                                const std::vector<std::array<double, 3> >& xs,
00293                                const bool integrateWeightingField);
00297   void AddSignalWeightingField(const double q, const std::vector<double>& ts,
00298                                const std::vector<std::array<double, 3> >& xs,
00299                                const std::vector<std::array<double, 3> >& vs,
00300                                const std::vector<double>& ns, const int navg);
00301
00303   void PlotSignal(const std::string& label, TPad* pad,
00304                   const std::string optTotal = "t",
00305                   const std::string optPrompt = "",
00306                   const std::string optDelayed = "");
00308   void ExportSignal(const std::string& label, const std::string& filename,
00309                     const bool chargeCariers = false) const;
00310
00312   void AddInducedCharge(const double q, const double x0, const double y0,
00313                         const double z0, const double x1, const double y1,
00314                         const double z1);
00315
00318   bool CrossedWire(const double x0, const double y0, const double z0,
00319                    const double x1, const double y1, const double z1,
00320                    double& xc, double& yc, double& zc, const bool centre,
00321                    double& rc);
00323   bool InTrapRadius(const double q0, const double x0, const double y0,
00324                     const double z0, double& xw, double& yw, double& rw);
00327   bool CrossedPlane(const double x0, const double y0, const double z0,
00328                     const double x1, const double y1, const double z1,
00329                     double& xc, double& yc, double& zc);
00330
00333   double IntegrateFluxLine(const double x0, const double y0, const double z0,
00334                            const double x1, const double y1, const double z1,
00335                            const double xp, const double yp, const double zp,
00336                            const unsigned int nI, const int isign = 0);
00337   // TODO!
00338   double GetTotalInducedCharge(const std::string& label);
00339
00340   double StepSizeHint();
00341
00343   double CreateGPUTransferObject(SensorGPU*& sensor_gpu);
00344 #if USEGPU
00346   void TransferGPUElectrodeSignals(SensorGPU*& sensor_gpu);
00347 #endif
00348
00349  private:
00350   std::string m_className = "Sensor";
00352   std::mutex m_mutex;
00353 #endif
00354
00356 #ifdef ___GPUCOMPILE___
00357   ComponentGPU** m_components = nullptr;
00358   size_t m_numComponents;
00359   friend class Sensor;
00360 #else
00361   std::vector<std::tuple<Component*, bool, bool> > m_components;
00362 #endif
00363
00364 #ifndef ___GPUCOMPILE___
00365   struct Electrode {
00366     Component* comp;
00367     std::string label;
00368     std::vector<double> signal;
00369     std::vector<double> delayedSignal;
00370     std::vector<double> electronSignal;
00371     std::vector<double> ionSignal;
00372     std::vector<double> delayedElectronSignal;
00373     std::vector<double> delayedIonSignal;
00374     double charge;
00375     bool integrated;
00376   };
00378   std::vector<Electrode> m_electrodes;
00379 #else
00380   struct ElectrodeGPU {
00381     ComponentGPU* comp;
00382     int label;
00383     double* signal;
00384   };
```

```
00385
00386   ElectrodeGPU* m_electrodes = nullptr;
00387   size_t m_numElectrodes;
00388 #endif
00389
00390   // Time window for signals
00391   double m_tStart = 0.;
00392   double m_tStep = 10.;
00393   unsigned int m_nTimeBins = 200;
00394   unsigned int m_nEvents = 0;
00395 #ifndef ___GPUCOMPILE___
00396   bool m_delayedSignal = false;
00397   std::vector<double> m_delayedSignalTimes;
00398   unsigned int m_nAvgDelayedSignal = 0;
00399
00400   // Transfer function
00401   std::function<double(double)> m_fTransfer;
00402   Shaper* m_shaper = nullptr;
00403   std::vector<std::pair<double, double> > m_fTransferTab;
00404   bool m_cacheTransferFunction = true;
00405   // Integral of the transfer function squared.
00406   double m_fTransferSq = -1.;
00407   // FFT of the transfer function.
00408   std::vector<double> m_fTransferFFT;
00409
00410   // Noise
00411   double (*m_fNoise)(double t) = nullptr;
00412
00413   std::vector<std::pair<double, bool> > m_thresholdCrossings;
00414   double m_thresholdLevel = 0.;
00415 #endif
00416
00417   // User bounding box
00418   double m_xMinUser = 0., m_yMinUser = 0., m_zMinUser = 0.;
00419   double m_xMaxUser = 0., m_yMaxUser = 0., m_zMaxUser = 0.;
00420   bool m_hasUserArea = false;
00421
00422 #ifndef ___GPUCOMPILE___
00423   // Switch on/off debugging messages
00424   bool m_debug = false;
00425
00426   // Return the current sensor size
00427   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00428                       double& ymax, double& zmax);
00429
00430   void FillSignal(Electrode& electrode, const double q,
00431                   const std::vector<double>& ts, const std::vector<double>& is,
00432                   const int navg, const bool delayed = false);
00433   void FillBin(Electrode& electrode, const unsigned int bin,
00434                const double signal, const bool electron, const bool delayed) {
00435     std::lock_guard<std::mutex> guard(m_mutex);
00436     electrode.signal[bin] += signal;
00437     if (delayed) electrode.delayedSignal[bin] += signal;
00438     if (electron) {
00439       electrode.electronSignal[bin] += signal;
00440       if (delayed) electrode.delayedElectronSignal[bin] += signal;
00441     } else {
00442       electrode.ionSignal[bin] += signal;
00443       if (delayed) electrode.delayedIonSignal[bin] += signal;
00444     }
00445   }
00446 #else
00447   __device__ void FillBin(ElectrodeGPU& electrode, const unsigned int bin,
00448                           const double signal, const bool electron,
00449                           const bool delayed, const int particle_idx);
00450 #endif
00451
00452 #ifndef ___GPUCOMPILE___
00453   void IntegrateSignal(Electrode& electrode);
00454   void ConvoluteSignal(Electrode& electrode, const std::vector<double>& tab);
00455   bool ConvoluteSignalFFT();
00456   bool ConvoluteSignalFFT(const std::string& label);
00457   void ConvoluteSignalFFT(Electrode& electrode, const std::vector<double>& tab,
00458                           const unsigned int nn);
00459   // Evaluate the integral over the transfer function squared.
00460   double TransferFunctionSq();
00461   double InterpolateTransferFunctionTable(const double t) const;
00462   void MakeTransferFunctionTable(std::vector<double>& tab);
00463   void FFT(std::vector<double>& data, const bool inverse, const int nn);
00464 #endif
00465 };
00466 }  // namespace Garfield
00467
00468 #undef SENSORCLASS
00469 #endif
```

## 8.115 /builds/garfield/docs/source/Include/Garfield/Shaper.hh File Reference

#include <string>

Classes

- class Garfield::Shaper

  Class for signal processing.

Namespaces

- namespace Garfield

## 8.116 Shaper.hh

Go to the documentation of this file.

```
00001 #ifndef G__SHAPER__H
00002 #define G__SHAPER__H
00003
00004 #include <string>
00005
00006 namespace Garfield {
00007
00009
00010 class Shaper {
00011  public:
00013   Shaper() = delete;
00015   Shaper(const unsigned int n, const double tau, const double g,
00016          std::string shaperType);
00018   ~Shaper() {}
00019
00021   double Shape(const double t) const;
00023   double UnipolarShaper(const double t) const;
00025   double BipolarShaper(const double t) const;
00027   double PeakingTime() const { return m__tp; }
00028
00030   double TransferFuncSq() const { return m__transfer__func__sq; }
00031
00033   bool IsUnipolar() const { return (m__type == ShaperType::Unipolar); }
00035   bool IsBipolar() const { return (m__type == ShaperType::Bipolar); }
00037   void GetParameters(unsigned int& n, double& tp) {
00038     n = m__n;
00039     tp = m__tp;
00040   }
00041
00042  private:
00043   std::string m__className = "Shaper";
00044
00045   // Shaper type.
00046   enum class ShaperType { Unipolar = 0, Bipolar };
00047   ShaperType m__type = ShaperType::Unipolar;
00048   // Order of the shaper.
00049   unsigned int m__n = 1;
00050   // Time constant.
00051   double m__tau = 1.;
00052   // Peaking time.
00053   double m__tp = 1.;
00054   // Normalization factor.
00055   double m__prefactor = 1.;
00056   // Gain.
00057   double m__g = 1.;
00058   // Integral of the transfer function squared.
00059   double m__transfer__func__sq = -1.;
00060 };
00061 }  // namespace Garfield
00062
00063 #endif
```

## 8.117 /builds/garfield/docs/source/Include/Garfield/Solid.hh File Reference

#include <string>
#include <vector>

Classes

- struct Garfield::Panel

  Surface panel.
- class Garfield::Solid

  Abstract base class for solids.

Namespaces

- namespace Garfield

## 8.118 Solid.hh

Go to the documentation of this file.

```
00001 #ifndef G__SOLID__H
00002 #define G__SOLID__H
00003
00004 #include <string>
00005 #include <vector>
00006
00007 namespace Garfield {
00008
00010
00011 struct Panel {
00013   double a, b, c;
00015   std::vector<double> xv;
00017   std::vector<double> yv;
00019   std::vector<double> zv;
00021   int colour;
00023   int volume;
00024 };
00025
00027
00028 class Solid {
00029  public:
00031   Solid() = delete;
00033   Solid(const double cx, const double cy, const double cz,
00034         const std::string& name)
00035     : m_cX(cx), m_cY(cy), m_cZ(cz), m_className(name) {
00036     m_id = s_id++;
00037   }
00038
00040   virtual ~Solid() {}
00041
00044   virtual bool IsInside(const double x, const double y, const double z,
00045                         const bool tesselated = false) const = 0;
00047   virtual bool GetBoundingBox(double& xmin, double& ymin, double& zmin,
00048                               double& xmax, double& ymax,
00049                               double& zmax) const = 0;
00051   virtual bool IsBox() const { return false; }
00053   virtual bool IsTube() const { return false; }
00055   virtual bool IsSphere() const { return false; }
00057   virtual bool IsHole() const { return false; }
00059   virtual bool IsRidge() const { return false; }
00061   virtual bool IsExtrusion() const { return false; }
00063   virtual bool IsWire() const { return false; }
00064
00066   void SetLabel(const std::string& label) { m_label = label; }
00068   std::string GetLabel() const { return m_label; }
00069
00071   bool GetCentre(double& x, double& y, double& z) const {
00072     x = m_cX;
00073     y = m_cY;
00074     z = m_cZ;
00075     return true;
00076   }
00078   bool GetDirection(double& dx, double& dy, double& dz) const {
00079     dx = m_dX;
00080     dy = m_dY;
00081     dz = m_dZ;
00082     return true;
00083   }
00085   bool GetOrientation(double& ctheta, double& stheta, double& cphi,
00086                       double& sphi) const {
00087     ctheta = m_cTheta;
00088     stheta = m_sTheta;
00089     cphi = m_cPhi;
00090     sphi = m_sPhi;
00091     return true;
00092   }
00093
```

```
00095   virtual double GetHalfLengthX() const {
00096     return NotImplemented("GetHalfLengthX");
00097   }
00099   virtual double GetHalfLengthY() const {
00100     return NotImplemented("GetHalfLengthY");
00101   }
00103   virtual double GetHalfLengthZ() const {
00104     return NotImplemented("GetHalfLengthZ");
00105   }
00107   virtual double GetInnerRadius() const {
00108     return NotImplemented("GetInnerRadius");
00109   }
00111   virtual double GetOuterRadius() const {
00112     return NotImplemented("GetOuterRadius");
00113   }
00115   virtual double GetRadius() const { return NotImplemented("GetRadius"); }
00117   virtual double GetLowerRadius() const {
00118     return NotImplemented("GetLowerRadius");
00119   }
00121   virtual double GetUpperRadius() const {
00122     return NotImplemented("GetUpperRadius");
00123   }
00125   virtual double GetRidgeOffset() const {
00126     return NotImplemented("GetRidgeOffset");
00127   }
00129   virtual double GetRidgeHeight() const {
00130     return NotImplemented("GetRidgeHeight");
00131   }
00133   virtual bool GetProfile(std::vector<double>& xv,
00134                           std::vector<double>& yv) const;
00135
00137   unsigned int GetId() const { return m_id; }
00138
00140   virtual bool SolidPanels(std::vector<Panel>& panels) = 0;
00141
00143   virtual void SetDiscretisationLevel(const double dis) = 0;
00145   virtual double GetDiscretisationLevel(const Panel& panel) = 0;
00146
00147   virtual void Cut(const double x0, const double y0, const double z0,
00148                    const double xn, const double yn, const double zn,
00149                    std::vector<Panel>& panels) = 0;
00150
00151   enum BoundaryCondition {
00152     Unknown = 0,
00153     Voltage,
00154     Charge,
00155     Float,
00156     Dielectric,
00157     DielectricCharge,
00158     ParallelField,
00159     PerpendicularField
00160   };
00161
00163   void SetBoundaryPotential(const double v) {
00164     m_volt = v;
00165     m_bctype = Voltage;
00166   }
00168   void SetBoundaryChargeDensity(const double q) {
00169     m_charge = q;
00170     m_bctype = Charge;
00171   }
00173   void SetBoundaryFloat() { m_bctype = Float; }
00175   void SetBoundaryDielectric() { m_bctype = Dielectric; }
00176   void SetBoundaryParallelField() { m_bctype = ParallelField; }
00177   void SetBoundaryPerpendicularField() { m_bctype = PerpendicularField; }
00178
00180   BoundaryCondition GetBoundaryConditionType() const { return m_bctype; }
00182   double GetBoundaryPotential() const { return m_volt; }
00184   double GetBoundaryChargeDensity() const { return m_charge; }
00185
00187   void EnableDebugging(const bool on = true) { m_debug = on; }
00188
00190   void SetColour(const int col) { m_colour = col; }
00192   int GetColour() const { return m_colour; }
00193
00194   static bool Intersect(const double x1, const double y1, const double z1,
00195                         const double x2, const double y2, const double z2,
00196                         const double x0, const double y0, const double z0,
00197                         const double a, const double b, const double c,
00198                         double& xc, double& yc, double& zc);
00199
00200 protected:
00202   double m_cX = 0., m_cY = 0., m_cZ = 0.;
00203
00205   double m_dX = 0., m_dY = 0., m_dZ = 1.;
00207   double m_cPhi = 1., m_sPhi = 0.;
00209   double m_cTheta = 1., m_sTheta = 0.;
```

```
00210
00212   std::string m_className = "Solid";
00213
00215   std::string m_label = "";
00216
00218   bool m_debug = false;
00219
00221   BoundaryCondition m_bctype = Unknown;
00223   double m_volt = 0.;
00225   double m_charge = 0.;
00227   double m_eps = 0.;
00228
00230   int m_colour = -1;
00231
00234   void ToLocal(const double x, const double y, const double z, double& u,
00235                double& v, double& w) const {
00236     const double dx = x - m_cX;
00237     const double dy = y - m_cY;
00238     const double dz = z - m_cZ;
00239
00240     u = m_cPhi * m_cTheta * dx + m_sPhi * m_cTheta * dy - m_sTheta * dz;
00241     v = -m_sPhi * dx + m_cPhi * dy;
00242     w = m_cPhi * m_sTheta * dx + m_sPhi * m_sTheta * dy + m_cTheta * dz;
00243   }
00246   void ToGlobal(const double u, const double v, const double w, double& x,
00247                 double& y, double& z) const {
00248     x = m_cX + m_cPhi * m_cTheta * u - m_sPhi * v + m_cPhi * m_sTheta * w;
00249     y = m_cY + m_sPhi * m_cTheta * u + m_cPhi * v + m_sPhi * m_sTheta * w;
00250     z = m_cZ - m_sTheta * u + m_cTheta * w;
00251   }
00253   void VectorToLocal(const double x, const double y, const double z, double& u,
00254                      double& v, double& w) {
00255     u = m_cPhi * m_cTheta * x + m_sPhi * m_cTheta * y - m_sTheta * z;
00256     v = -m_sPhi * x + m_cPhi * y;
00257     w = m_cPhi * m_sTheta * x + m_sPhi * m_sTheta * y + m_cTheta * z;
00258   }
00259
00260   void SetDirection(const double dx, const double dy, const double dz);
00261
00262  private:
00263   double NotImplemented(const std::string& fcn) const;
00265   static unsigned int s_id;
00267   unsigned int m_id;
00268 };
00269 }  // namespace Garfield
00270
00271 #endif
```

## 8.119 /builds/garfield/docs/source/Include/Garfield/SolidBox.hh File Reference

#include <array>
#include "Garfield/Solid.hh"

### Classes

- class Garfield::SolidBox

  Box.

### Namespaces

- namespace Garfield

## 8.120 SolidBox.hh

Go to the documentation of this file.

```
00001 #ifndef G_SOLID_BOX_H
00002 #define G_SOLID_BOX_H
00003
00004 #include <array>
00005
00006 #include "Garfield/Solid.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class SolidBox : public Solid {
00013  public:
```

```
00015   SolidBox(const double cx, const double cy, const double cz, const double lx,
00016            const double ly, const double lz);
00018   SolidBox(const double cx, const double cy, const double cz, const double lx,
00019            const double ly, const double lz, const double dx, const double dy,
00020            const double dz);
00022   ~SolidBox() {}
00023
00024   bool IsInside(const double x, const double y, const double z,
00025                 const bool tesselated) const override;
00026   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00027                       double& ymax, double& zmax) const override;
00028   bool IsBox() const override { return true; }
00029
00030   double GetHalfLengthX() const override { return m_lX; }
00031   double GetHalfLengthY() const override { return m_lY; }
00032   double GetHalfLengthZ() const override { return m_lZ; }
00033
00034   void SetHalfLengthX(const double lx);
00035   void SetHalfLengthY(const double ly);
00036   void SetHalfLengthZ(const double lz);
00037
00038   bool SolidPanels(std::vector<Panel>& panels) override;
00039   void SetDiscretisationLevel(const double dis) override { m_dis.fill(dis); }
00040   double GetDiscretisationLevel(const Panel& panel) override;
00041
00042   void Cut(const double x0, const double y0, const double z0, const double xn,
00043            const double yn, const double zn,
00044            std::vector<Panel>& panels) override;
00045
00046  private:
00048   double m_lX = 0., m_lY = 0., m_lZ = 0.;
00050   std::array<double, 6> m_dis{{-1, -1, -1, -1, -1, -1}};
00051 };
00052 }  // namespace Garfield
00053
00054 #endif
```

## 8.121 /builds/garfield/docs/source/Include/Garfield/SolidExtrusion.hh File Reference

#include <array>
#include "Garfield/Solid.hh"

### Classes

- class Garfield::SolidExtrusion

  Extrusion.

### Namespaces

- namespace Garfield

## 8.122 SolidExtrusion.hh

Go to the documentation of this file.

```
00001 #ifndef G__SOLID__EXTRUSION__H
00002 #define G__SOLID__EXTRUSION__H
00003
00004 #include <array>
00005
00006 #include "Garfield/Solid.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class SolidExtrusion : public Solid {
00013  public:
00015   SolidExtrusion(const double lz, const std::vector<double>& xp,
00016                  const std::vector<double>& yp);
00018   SolidExtrusion(const double lz, const std::vector<double>& xp,
00019                  const std::vector<double>& yp, const double cx,
00020                  const double cy, const double cz, const double dx,
00021                  const double dy, const double dz);
00023   ~SolidExtrusion() {}
00024
00025   bool IsInside(const double x, const double y, const double z,
00026                 const bool tesselated) const override;
00027   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
```

```
00028                     double& ymax, double& zmax) const override;
00029   bool IsExtrusion() const override { return true; }
00030
00031   double GetHalfLengthZ() const override { return m_lZ; }
00032   bool GetProfile(std::vector<double>& xp,
00033                   std::vector<double>& yp) const override {
00034     xp = m_xp;
00035     yp = m_yp;
00036     return !m_xp.empty();
00037   }
00039   void SetHalfLengthZ(const double lz);
00041   void SetProfile(const std::vector<double>& xp, const std::vector<double>& yp);
00043   void SetTopLid(const bool closed) { m_toplid = closed; }
00045   void SetBottomLid(const bool closed) { m_botlid = closed; }
00046
00047   bool SolidPanels(std::vector<Panel>& panels) override;
00048   void SetDiscretisationLevel(const double dis) override { m_dis.fill(dis); }
00049   double GetDiscretisationLevel(const Panel& panel) override;
00050
00051   void Cut(const double x0, const double y0, const double z0, const double xn,
00052            const double yn, const double zn,
00053            std::vector<Panel>& panels) override;
00054
00055  private:
00057   double m_lZ = 0.;
00059   std::vector<double> m_xp;
00061   std::vector<double> m_yp;
00062
00064   bool m_toplid = true;
00066   bool m_botlid = true;
00067
00069   bool m_clockwise = true;
00070
00072   std::array<double, 3> m_dis{{-1, -1, -1}};
00073 };
00074 }  // namespace Garfield
00075
00076 #endif
```

## 8.123 /builds/garfield/docs/source/Include/Garfield/SolidHole.hh File Reference

#include <array>
#include <mutex>
#include "Garfield/Solid.hh"

### Classes

- class Garfield::SolidHole

  Box with a cylindrical hole.

### Namespaces

- namespace Garfield

## 8.124 SolidHole.hh

Go to the documentation of this file.

```
00001 #ifndef G_SOLID_HOLE_H
00002 #define G_SOLID_HOLE_H
00003
00004 #include <array>
00005 #include <mutex>
00006
00007 #include "Garfield/Solid.hh"
00008
00009 namespace Garfield {
00010
00012
00013 class SolidHole : public Solid {
00014  public:
00016   SolidHole(const double cx, const double cy, const double cz, const double rup,
00017             const double rlow, const double lx, const double ly,
00018             const double lz);
00021   SolidHole(const double cx, const double cy, const double cz, const double rup,
00022             const double rlow, const double lx, const double ly,
00023             const double lz, const double dx, const double dy, const double dz);
00025   ~SolidHole() {}
```

```
00026
00027   bool IsInside(const double x, const double y, const double z,
00028                 const bool tesselated) const override;
00029   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00030                       double& ymax, double& zmax) const override;
00031   bool IsHole() const override { return true; }
00032
00034   void SetHalfLengthX(const double lx);
00036   void SetHalfLengthY(const double ly);
00038   void SetHalfLengthZ(const double lz);
00040   void SetUpperRadius(const double r);
00042   void SetLowerRadius(const double r);
00043
00044   double GetHalfLengthX() const override { return m_lX; }
00045   double GetHalfLengthY() const override { return m_lY; }
00046   double GetHalfLengthZ() const override { return m_lZ; }
00047   double GetUpperRadius() const override { return m_rUp; }
00048   double GetLowerRadius() const override { return m_rLow; }
00049
00054   void SetSectors(const unsigned int n);
00060   void SetAverageRadius(const bool average) {
00061     m_average = average;
00062     Update();
00063   }
00064
00066   unsigned int GetSectors() const { return m_n; }
00068   bool GetAverage() const { return m_average; }
00069
00070   bool SolidPanels(std::vector<Panel>& panels) override;
00071   void SetDiscretisationLevel(const double dis) override { m_dis.fill(dis); }
00072   double GetDiscretisationLevel(const Panel& panel) override;
00073
00074   void Cut(const double x0, const double y0, const double z0, const double xn,
00075            const double yn, const double zn,
00076            std::vector<Panel>& panels) override;
00077
00078  private:
00080   std::mutex m_mutex;
00081
00083   double m_rUp;
00085   double m_rLow;
00087   double m_lX;
00089   double m_lY;
00091   double m_lZ;
00092
00094   unsigned int m_n = 2;
00096   bool m_average = false;
00097
00099   double m_fp = 1.;
00101   double m_fi = 1.;
00102
00104   std::array<double, 7> m_dis{{-1., -1., -1., -1., -1., -1., -1.}};
00105
00106   void Update();
00107 };
00108 }  // namespace Garfield
00109
00110 #endif
```

## 8.125 /builds/garfield/docs/source/Include/Garfield/SolidRidge.hh File Reference

#include <array>
#include "Garfield/Solid.hh"

Classes

- class Garfield::SolidRidge

  Triangular prism (Toblerone bar).

Namespaces

- namespace Garfield

## 8.126 SolidRidge.hh

Go to the documentation of this file.

```
00001 #ifndef G_SOLID_RIDGE_H
```

```
00002 #define G__SOLID__RIDGE__H
00003
00004 #include <array>
00005
00006 #include "Garfield/Solid.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class SolidRidge : public Solid {
00013  public:
00015   SolidRidge(const double cx, const double cy, const double cz, const double lx,
00016              const double ly, const double hz, const double offsetx);
00018   SolidRidge(const double cx, const double cy, const double cz, const double lx,
00019              const double ly, const double hz, const double offsetx,
00020              const double dx, const double dy, const double dz);
00022   ~SolidRidge() {}
00023
00024   bool IsInside(const double x, const double y, const double z,
00025                 const bool tesselated) const override;
00026   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00027                       double& ymax, double& zmax) const override;
00028   bool IsRidge() const override { return true; }
00029
00031   void SetHalfLengthX(const double lx);
00033   void SetHalfLengthY(const double ly);
00035   void SetRidgeHeight(const double hz);
00037   void SetRidgeOffset(const double dx) { m__hx = dx; }
00038
00039   double GetHalfLengthX() const override { return m__lX; }
00040   double GetHalfLengthY() const override { return m__lY; }
00041   double GetRidgeHeight() const override { return m__hz; }
00042   double GetRidgeOffset() const override { return m__hx; }
00043
00044   bool SolidPanels(std::vector<Panel>& panels) override;
00045   void SetDiscretisationLevel(const double dis) override { m__dis.fill(dis); }
00046   double GetDiscretisationLevel(const Panel& panel) override;
00047
00048   void Cut(const double x0, const double y0, const double z0, const double xn,
00049            const double yn, const double zn,
00050            std::vector<Panel>& panels) override;
00051
00052  private:
00054   double m__lX;
00056   double m__lY;
00058   double m__hz;
00060   double m__hx;
00061
00063   std::array<double, 5> m__dis{{-1., -1., -1., -1., -1.}};
00064 };
00065 }  // namespace Garfield
00066
00067 #endif
```

## 8.127 /builds/garfield/docs/source/Include/Garfield/SolidSphere.hh File Reference

#include <mutex>
#include "Garfield/Solid.hh"

### Classes

- class Garfield::SolidSphere

  Sphere.

### Namespaces

- namespace Garfield

## 8.128 SolidSphere.hh

Go to the documentation of this file.

```
00001 #ifndef G__SOLID__SPHERE__H
00002 #define G__SOLID__SPHERE__H
00003
00004 #include <mutex>
00005
00006 #include "Garfield/Solid.hh"
```

```
00007
00008 namespace Garfield {
00009
00011
00012 class SolidSphere : public Solid {
00013  public:
00015   SolidSphere(const double cx, const double cy, const double cz,
00016               const double r);
00018   SolidSphere(const double cx, const double cy, const double cz,
00019               const double rmin, const double rmax);
00021   ~SolidSphere() {}
00022
00023   bool IsInside(const double x, const double y, const double z,
00024                 const bool tesselated) const override;
00025   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00026                       double& ymax, double& zmax) const override;
00027   bool IsSphere() const override { return true; }
00028
00030   void SetRadius(const double r);
00032   void SetRadii(const double rmin, const double rmax);
00033
00034   double GetRadius() const override { return m_rMax; }
00035   double GetInnerRadius() const override { return m_rMin; }
00036   double GetOuterRadius() const override { return m_rMax; }
00037
00041   void SetMeridians(const unsigned int n);
00042
00043   bool SolidPanels(std::vector<Panel>& panels) override;
00044   void SetDiscretisationLevel(const double dis) override { m_dis = dis; }
00045   double GetDiscretisationLevel(const Panel& panel) override;
00046
00047   void Cut(const double x0, const double y0, const double z0, const double xn,
00048            const double yn, const double zn,
00049            std::vector<Panel>& panels) override;
00050
00051  private:
00053   std::mutex m_mutex;
00054
00056   double m_rMin = 0., m_rMax = 1.;
00057
00059   unsigned int m_n = 10;
00060
00062   double m_dis = -1.;
00063
00065   std::vector<Panel> m_panelsO;
00066   std::vector<Panel> m_panelsI;
00067
00068   void UpdatePanels();
00069   void MakePanels(const int vol, const double r, const bool out,
00070                   std::vector<Panel>& panels) const;
00071 };
00072 }  // namespace Garfield
00073
00074 #endif
```

## 8.129 /builds/garfield/docs/source/Include/Garfield/SolidTube.hh File Reference

#include <array>
#include <mutex>
#include "Garfield/Solid.hh"

### Classes

- class Garfield::SolidTube

  Cylindrical tube.

### Namespaces

- namespace Garfield

## 8.130 SolidTube.hh

Go to the documentation of this file.

```
00001 #ifndef G_SOLID_TUBE_H
00002 #define G_SOLID_TUBE_H
00003
00004 #include <array>
```

```
00005 #include <mutex>
00006
00007 #include "Garfield/Solid.hh"
00008
00009 namespace Garfield {
00010
00012
00013 class SolidTube : public Solid {
00014  public:
00016   SolidTube(const double cx, const double cy, const double cz, const double r,
00017             const double lz);
00019   SolidTube(const double cx, const double cy, const double cz, const double r,
00020             const double lz, const double dx, const double dy, const double dz);
00022   SolidTube(const double cx, const double cy, const double cz, const double ri,
00023             const double ro, const double lz);
00026   SolidTube(const double cx, const double cy, const double cz, const double ri,
00027             const double ro, const double lz, const double dx, const double dy,
00028             const double dz);
00030   ~SolidTube() {}
00031
00032   bool IsInside(const double x, const double y, const double z,
00033                 const bool tesselated) const override;
00034   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00035                       double& ymax, double& zmax) const override;
00036   bool IsTube() const override { return true; }
00037
00039   void SetHalfLength(const double lz);
00041   void SetRadius(const double r);
00043   double GetHalfLengthZ() const override { return m_lZ; }
00045   double GetRadius() const override { return m_rO; }
00047   double GetOuterRadius() const override { return m_rO; }
00049   double GetInnerRadius() const override { return m_rI; }
00050
00055   void SetSectors(const unsigned int n);
00059   void SetRotation(const double angle) { m_rot = angle; }
00065   void SetAverageRadius(const bool average) { m_average = average; }
00067   void SetTopLid(const bool closed) { m_toplid = closed; }
00069   void SetBottomLid(const bool closed) { m_botlid = closed; }
00070
00072   unsigned int GetSectors() const { return m_n; }
00074   double GetRotation() const { return m_rot; }
00076   bool GetAverage() const { return m_average; }
00077
00078   bool SolidPanels(std::vector<Panel>& panels) override;
00079   void SetDiscretisationLevel(const double dis) override { m_dis.fill(dis); }
00080   double GetDiscretisationLevel(const Panel& panel) override;
00081
00082   void Cut(const double x0, const double y0, const double z0, const double xn,
00083            const double yn, const double zn,
00084            std::vector<Panel>& panels) override;
00085
00086  private:
00088   std::mutex m_mutex;
00089
00091   double m_rO;
00093   double m_rI = 0.;
00095   double m_lZ;
00096
00098   double m_rot = 0.;
00100   unsigned int m_n = 2;
00102   bool m_average = false;
00104   double m_rpO;
00106   double m_rpI = 0.;
00108   double m_riO;
00110   double m_riI = 0.;
00112   std::vector<double> m_xpO;
00114   std::vector<double> m_ypO;
00116   std::vector<double> m_xpI;
00118   std::vector<double> m_ypI;
00119
00121   bool m_toplid = true;
00123   bool m_botlid = true;
00124
00126   std::array<double, 3> m_dis{{-1., -1., -1.}};
00127
00128   void UpdatePolygon();
00129 };
00130 }  // namespace Garfield
00131
00132 #endif
```

## 8.131 /builds/garfield/docs/source/Include/Garfield/SolidWire.hh File Reference

#include "Garfield/Solid.hh"

Classes

- class Garfield::SolidWire

  Wire.

Namespaces

- namespace Garfield

## 8.132 SolidWire.hh

Go to the documentation of this file.

```
00001 #ifndef G__SOLID__WIRE__H
00002 #define G__SOLID__WIRE__H
00003
00004 #include "Garfield/Solid.hh"
00005
00006 namespace Garfield {
00007
00009
00010 class SolidWire : public Solid {
00011  public:
00013   SolidWire(const double cx, const double cy, const double cz, const double r,
00014             const double lz);
00016   SolidWire(const double cx, const double cy, const double cz, const double r,
00017             const double lz, const double dx, const double dy, const double dz);
00019   ~SolidWire() {}
00020
00021   bool IsInside(const double x, const double y, const double z,
00022                 const bool tesselated) const override;
00023   bool GetBoundingBox(double& xmin, double& ymin, double& zmin, double& xmax,
00024                       double& ymax, double& zmax) const override;
00025   bool IsWire() const override { return true; }
00026
00027   void SetHalfLength(const double lz);
00028   void SetRadius(const double r);
00029
00030   double GetHalfLengthZ() const override { return m__lZ; }
00031   double GetRadius() const override { return m__r; }
00032
00033   bool SolidPanels(std::vector<Panel>& panels) override;
00034   void SetDiscretisationLevel(const double dis) override { m__dis = dis; }
00035   double GetDiscretisationLevel(const Panel& panel) override;
00036
00037   void Cut(const double x0, const double y0, const double z0, const double xn,
00038            const double yn, const double zn,
00039            std::vector<Panel>& panels) override;
00040
00041  private:
00043   double m__r;
00045   double m__lZ;
00046
00048   double m__dis = -1.;
00049 };
00050 }  // namespace Garfield
00051
00052 #endif
```

## 8.133 /builds/garfield/docs/source/Include/Garfield/TetrahedralTree.hh File Reference

#include "Garfield/HelperMacros.hh"
#include "Garfield/Vector.hh"
#include <cstddef>
#include <utility>
#include <vector>

Namespaces

- namespace Garfield

Typedefs

- using Garfield::Vec3 = Vec3Impl<double>

Functions

- class Garfield::GARFIELD_CLASS_NAME (TetrahedralTree)

  Helper class for searches in field maps.

## 8.134 TetrahedralTree.hh

Go to the documentation of this file.

```
00001 // Include this header if we're compiling with the GPU or this is the first time
00002 // without
00003 #if defined(___GPUCOMPILE___) || !defined(TETRAHEDRAL_TREE_H)
00004
00005 #if !defined(___GPUCOMPILE___) && !defined(TETRAHEDRAL_TREE_H)
00006 #define TETRAHEDRAL_TREE_H
00007 #endif
00008
00009 #include "Garfield/HelperMacros.hh"
00010 #include "Garfield/Vector.hh"
00011
00012 #if defined(___GPUCOMPILE___)
00013 #include "GPUInterface.hh"
00014 #else
00015 class TetrahedralTreeGPU;
00016 #endif
00017
00018 #include <cstddef>
00019 #include <utility>
00020 #include <vector>
00021
00022 namespace Garfield {
00023
00024 #ifdef ___GPUCOMPILE___
00025 using Vec3 = Vec3Impl<cuda_t>;
00026 #else
00027 using Vec3 = Vec3Impl<double>;
00028 #endif
00029
00042 class GARFIELD_CLASS_NAME(TetrahedralTree) {
00043  public:
00044 #ifdef ___GPUCOMPILE___
00045   // Constructor
00046   GARFIELD_CLASS_NAME(TetrahedralTree)() = default;
00047
00048   // Destructor
00049   ~GARFIELD_CLASS_NAME(TetrahedralTree)() = default;
00050 #else
00051   // Constructor
00052   GARFIELD_CLASS_NAME(TetrahedralTree)(const Vec3& origin,
00053                                        const Vec3& halfDimension);
00054
00056   ~GARFIELD_CLASS_NAME(TetrahedralTree)();
00057 #endif
00058
00059 #ifndef ___GPUCOMPILE___
00060   // Insert a mesh node (a vertex/point) to the tree
00061   void InsertMeshNode(Vec3 point, const int index);
00062
00064   void InsertMeshElement(const double bb[6], const int index);
00065
00067   double CreateGPUTransferObject(TetrahedralTreeGPU*& tree_gpu);
00068 #endif
00069
00070  private:
00071   static std::vector<int> emptyBlock;
00072
00073   // Physical centre of this tree node.
00074   Vec3 m_origin;
00075   // Half the width/height/depth of this tree node.
00076   Vec3 m_halfDimension;
00077   // Storing min and max points for convenience
00078   Vec3 m_min, m_max;
00079
00080   // The tree has up to eight children and can additionally store
00081   // a list of mesh nodes and mesh elements.
00082   // Pointers to child octants.
00083   GARFIELD_CLASS_NAME(TetrahedralTree) * children[8];
00084
00085   // Children follow a predictable pattern to make accesses simple.
00086   // Here, - means less than 'origin' in that dimension, + means greater than.
00087   // child: 0 1 2 3 4 5 6 7
00088   // x:     - - - - + + + +
00089   // y:     - - + + - - + +
00090   // z:     - + - + - + - +
00091
```

```
00092 #ifndef ___GPUCOMPILE___
00093   std::vector<std::pair<Vec3, int> > nodes;
00094 #endif
00095
00096 #ifdef ___GPUCOMPILE___
00097   int* elements{nullptr};
00098   int numelements{0};
00099 #else
00100   std::vector<int> elements;
00101 #endif
00102
00103   static const size_t BlockCapacity = 10;
00104
00105 #ifndef ___GPUCOMPILE___
00106   // Check if the given box overlaps with this tree node.
00107   bool DoesBoxOverlap(const double bb[6]) const;
00108 #endif
00109   // Check if this tree node is a leaf or intermediate node.
00110   ___DEVICE___ bool IsLeafNode() const;
00111
00112  public:
00113 // Get all tetrahedra linked to a block corresponding to the given point
00114 #ifdef ___GPUCOMPILE___
00115   ___device___ void GetElementsInBlock(const Vec3& point,
00116                                        const int*& tet_list_elems,
00117                                        int& num_elems) const;
00118 #else
00119   const std::vector<int>& GetElementsInBlock(const Vec3& point) const;
00120 #endif
00121
00122  private:
00123   ___DEVICE___ int GetOctantContainingPoint(const Vec3& point) const;
00124   // Get a block containing the input point
00125   ___DEVICE___ const GARFIELD_CLASS_NAME(TetrahedralTree) *
00126       GetBlockFromPoint(const Vec3& point) const;
00127   // A helper function used by the function above.
00128   // Called recursively on the child nodes.
00129   ___DEVICE___ const GARFIELD_CLASS_NAME(TetrahedralTree) *
00130       GetBlockFromPointHelper(const Vec3& point) const;
00131
00132 #ifdef ___GPUCOMPILE___
00133   friend class TetrahedralTree;
00134 #endif
00135 };
00136
00137 }  // namespace Garfield
00138
00139 #endif
```

## 8.135 /builds/garfield/docs/source/Include/Garfield/TGeoTet.hh File Reference

#include <array>
#include "TGeoBBox.h"

### Classes

- class TGeoTet

## 8.136 TGeoTet.hh

Go to the documentation of this file.

```
00001 #ifndef G_TGEOTET
00002 #define G_TGEOTET
00003
00004 #include <array>
00005
00006 #include "TGeoBBox.h"
00007
00008 class TGeoTet : public TGeoBBox {
00009  public:
00010   TGeoTet() {}
00012   TGeoTet(const char* name,
00013           const std::array<std::array<double, 3>, 4>& vertices);
00015   virtual ~TGeoTet() {}
00016
00017   void ComputeBBox() override;
00018   int DistancetoPrimitive(int, int) override { return 99999; }
00019   const TBuffer3D& GetBuffer3D(int reqSections, bool localFrame) const override;
00020   void GetMeshNumbers(int& nvert, int& nsegs, int& npols) const override {
00021     nvert = 4;
```

```
00022     nsegs = 12;
00023     npols = 4;
00024   }
00025   int GetNmeshVertices() const override { return 4; }
00026   void InspectShape() const override {}
00027   TBuffer3D* MakeBuffer3D() const override;
00028   void Print(Option_t* option = "") const override;
00029   void SavePrimitive(std::ostream&, Option_t*) override {}
00030   void SetPoints(double* points) const override;
00031   void SetPoints(float* points) const override;
00032   void SetSegsAndPols(TBuffer3D& buff) const override;
00033   void Sizeof3D() const override {}
00034
00035   // ClassDef(TGeoTet, 1)
00036
00037  private:
00038   std::array<std::array<double, 3>, 4> fVertices;
00039
00040   TGeoTet(const TGeoTet&) = delete;
00041   TGeoTet& operator=(const TGeoTet&) = delete;
00042 };
00043
00044 #endif
```

## 8.137 /builds/garfield/docs/source/Include/Garfield/Track.hh File Reference

```
#include <array>
#include <cmath>
#include <string>
```

### Classes

- class Garfield::Track

  Abstract base class for track generation.

### Namespaces

- namespace Garfield

## 8.138 Track.hh

Go to the documentation of this file.

```
00001 #ifndef G_TRACK_H
00002 #define G_TRACK_H
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <string>
00007
00008 namespace Garfield {
00009
00010 class Sensor;
00011 class ViewDrift;
00012
00014
00015 class Track {
00016  public:
00018   Track() = delete;
00020   Track(const std::string& name);
00022   virtual ~Track() {}
00023
00037   virtual void SetParticle(const std::string& part);
00038
00040   void SetEnergy(const double e);
00042   void SetBetaGamma(const double bg);
00044   void SetBeta(const double beta);
00046   void SetGamma(const double gamma);
00048   void SetMomentum(const double p);
00050   void SetKineticEnergy(const double ekin);
00051
00053   double GetEnergy() const { return m_energy; }
00055   double GetBetaGamma() const { return sqrt(m_beta2 / (1. - m_beta2)); }
00057   double GetBeta() const { return sqrt(m_beta2); }
00059   double GetGamma() const { return sqrt(1. / (1. - m_beta2)); }
00061   double GetMomentum() const { return m_mass * sqrt(m_beta2 / (1. - m_beta2)); }
00063   double GetKineticEnergy() const { return m_energy - m_mass; }
00064
```

```
00066   double GetCharge() const { return m_q; }
00068   double GetMass() const { return m_mass; }
00069
00071   void SetSensor(Sensor* s);
00072
00075   virtual bool NewTrack(const double x0, const double y0, const double z0,
00076                         const double t0, const double dx0, const double dy0,
00077                         const double dz0) = 0;
00078
00081   virtual double GetClusterDensity() { return 0.; }
00083   virtual double GetStoppingPower() { return 0.; }
00084
00086   void EnablePlotting(ViewDrift* viewer);
00088   void DisablePlotting();
00089
00091   void EnableDebugging() { m_debug = true; }
00093   void DisableDebugging() { m_debug = false; }
00094
00095  protected:
00096   std::string m_className = "Track";
00097
00098   double m_q = -1.;
00099   int m_spin = 1;
00100   double m_mass;
00101   double m_energy = 0.;
00102   double m_beta2 = 1.;
00103   bool m_isElectron = false;
00104   std::string m_particleName = "mu-";
00105
00106   Sensor* m_sensor = nullptr;
00107
00108   bool m_isChanged = true;
00109
00110   ViewDrift* m_viewer = nullptr;
00111
00112   bool m_debug = false;
00113
00114   size_t m_plotId = 0;
00115   void PlotNewTrack(const double x0, const double y0, const double z0);
00116   void PlotCluster(const double x0, const double y0, const double z0);
00117
00118   static std::array<double, 3> StepBfield(const double dt, const double qoverm,
00119                                           const double vmag, double bx,
00120                                           double by, double bz,
00121                                           std::array<double, 3>& dir);
00122 };
00123 }  // namespace Garfield
00124
00125 #endif
```

## 8.139 /builds/garfield/docs/source/Include/Garfield/TrackBichsel.hh File Reference

#include <array>
#include <vector>
#include "Garfield/FundamentalConstants.hh"
#include "Garfield/Track.hh"

Classes

- class Garfield::TrackBichsel

  Generate tracks using differential cross-sections for silicon computed by Hans Bichsel.
- struct Garfield::TrackBichsel::Cluster

Namespaces

- namespace Garfield

## 8.140 TrackBichsel.hh

Go to the documentation of this file.

```
00001 #ifndef G_TRACK_BICHSEL_H
00002 #define G_TRACK_BICHSEL_H
00003
00004 #include <array>
00005 #include <vector>
00006
```

```
00007 #include "Garfield/FundamentalConstants.hh"
00008 #include "Garfield/Track.hh"
00009
00010 namespace Garfield {
00011
00017
00018 class TrackBichsel : public Track {
00019  public:
00020   struct Cluster {
00021     double x, y, z, t;
00022     double energy;
00023   };
00024
00026   TrackBichsel() : TrackBichsel(nullptr) {}
00028   TrackBichsel(Sensor* sensor);
00030   virtual ~TrackBichsel() {}
00031
00032   bool NewTrack(const double x0, const double y0, const double z0,
00033                 const double t0, const double dx0, const double dy0,
00034                 const double dz0) override;
00035   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00036
00037   double GetClusterDensity() override;
00038   double GetStoppingPower() override;
00039
00040   bool Initialise();
00041   bool ComputeCrossSection();
00042
00043  private:
00044   constexpr static size_t NEnergyBins = 1250;
00045   std::array<double, NEnergyBins + 1> m_E;
00046
00048   std::array<double, NEnergyBins> m_dfdE;
00050   std::array<double, NEnergyBins> m_eps1;
00052   std::array<double, NEnergyBins> m_eps2;
00054   std::array<double, NEnergyBins> m_int;
00056   std::array<double, NEnergyBins> m_k1;
00057
00058   constexpr static size_t NCdfBins = 10000;
00059   std::array<double, NCdfBins> m_tab;
00060
00062   double m_density;
00064   double m_conv = 0.0092456;
00065
00066   bool m_initialised = false;
00067
00069   double m_imfp = 0.;
00071   double m_dEdx = 0.;
00072
00074   double m_speed = SpeedOfLight;
00075
00076   std::vector<Cluster> m_clusters;
00077 };
00078 }  // namespace Garfield
00079
00080 #endif
```

## 8.141 /builds/garfield/docs/source/Include/Garfield/TrackDegrade.hh File Reference

#include <array>
#include <vector>
#include "Garfield/Track.hh"

### Classes

- class Garfield::TrackDegrade

  Interface to Degrade.
- struct Garfield::TrackDegrade::Electron
- struct Garfield::TrackDegrade::Excitation
- struct Garfield::TrackDegrade::Cluster

### Namespaces

- namespace Garfield

## 8.142 TrackDegrade.hh

Go to the documentation of this file.

```
00001 #ifndef G_TRACK_DEGRADE_H
00002 #define G_TRACK_DEGRADE_H
00003
00004 #include <array>
00005 #include <vector>
00006
00007 #include "Garfield/Track.hh"
00008
00009 namespace Garfield {
00010 class Medium;
00011
00013
00014 class TrackDegrade : public Track {
00015  public:
00016   struct Electron {
00017     double x = 0.;
00018     double y = 0.;
00019     double z = 0.;
00020     double t = 0.;
00021     double energy = 0.;
00022     double dx = 0.;
00023     double dy = 0.;
00024     double dz = 0.;
00025   };
00026
00027   struct Excitation {
00028     double x = 0.;
00029     double y = 0.;
00030     double z = 0.;
00031     double t = 0.;
00032     double energy = 0.;
00033   };
00034
00035   struct Cluster {
00036     double x, y, z, t;
00037     std::vector<Electron> deltaElectrons;
00038     std::vector<Electron> electrons;
00039     std::vector<Excitation> excitations;
00040   };
00041
00043   TrackDegrade() : TrackDegrade(nullptr) {}
00045   TrackDegrade(Sensor* sensor);
00047   virtual ~TrackDegrade() {}
00048
00049   bool NewTrack(const double x0, const double y0, const double z0,
00050                 const double t0, const double dx0, const double dy0,
00051                 const double dz0) override;
00052   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00053   double GetClusterDensity() override;
00054   double GetStoppingPower() override;
00055
00056   bool Initialise(Medium* medium, const bool verbose = false);
00057
00059   void SetThresholdEnergy(const double eth);
00061   void StoreExcitations(const bool on = true, const double thr = 4.) {
00062     m_storeExcitations = on;
00063     m_ethrExc = std::max(thr, 1.e-3);
00064   }
00066   void EnableBremsstrahlung(const bool on = true) { m_bremsStrahlung = on; }
00068   void EnableFullCascade(const bool on = true) { m_fullCascade = on; }
00069
00070   void SetParticle(const std::string& particle) override;
00071
00072  protected:
00073   std::vector<Cluster> m_clusters;
00074
00075   bool m_penning = true;
00076   bool m_bremsStrahlung = true;
00077   bool m_fullCascade = true;
00078   bool m_storeExcitations = false;
00079   // Energy threshold for storing excitations.
00080   double m_ethrExc = 4.;
00081   // Energy threshold for tracking electrons.
00082   double m_ethr = 2.;
00083
00084   double m_pressure = -1.;
00085   double m_temperature = -1.;
00086   std::string m_mediumName = "";
00087   unsigned int m_nGas = 0;
00088
00089   double m_dedx = -1.;
00090   double m_clusterDensity = -1.;
00091
```

```
00092   std::array<double, 6> m_rPenning;
00093   std::array<double, 6> m_dPenning;
00094
00095   std::pair<std::vector<Electron>, std::vector<Excitation> >
00096   TransportDeltaElectron(const double x0, const double y0, const double z0,
00097                          const double t0, const double e0, const double dx,
00098                          const double dy, const double dz);
00099
00100   void SetupPenning(Medium* medium, std::array<double, 6>& rP,
00101                     std::array<double, 6>& dP);
00102   bool IsInside(const double x, const double y, const double z);
00103 };
00104 }  // namespace Garfield
00105
00106 #endif
```

## 8.143 /builds/garfield/docs/source/Include/Garfield/TrackElectron.hh File Reference

#include <string>
#include <vector>
#include "Garfield/Track.hh"

Classes

- class Garfield::TrackElectron

  [WIP] Ionization calculation based on MIP program (S. Biagi).
- struct Garfield::TrackElectron::Cluster
- struct Garfield::TrackElectron::Parameters

Namespaces

- namespace Garfield

## 8.144 TrackElectron.hh

Go to the documentation of this file.

```
00001 #ifndef G_TRACK_ELECTRON
00002 #define G_TRACK_ELECTRON
00003
00004 #include <string>
00005 #include <vector>
00006
00007 #include "Garfield/Track.hh"
00008
00009 namespace Garfield {
00010 class Medium;
00011
00013
00014 class TrackElectron : public Track {
00015  public:
00016   struct Cluster {
00017     double x, y, z, t;
00018     double esec;
00019   };
00020
00022   TrackElectron();
00024   virtual ~TrackElectron() {}
00025
00026   void SetParticle(const std::string& particle) override;
00027
00028   bool NewTrack(const double x0, const double y0, const double z0,
00029                 const double t0, const double dx0, const double dy0,
00030                 const double dz0) override;
00031
00032   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00033
00034   double GetClusterDensity() override;
00035   double GetStoppingPower() override;
00036
00037  private:
00038   struct Parameters {
00039     // Dipole moment
00040     double m2;
00041     // Constant in ionisation cross-section
00042     double cIon;
```

```
00043     // Density correction term
00044     double x0;
00045     double x1;
00046     double cDens;
00047     double aDens;
00048     double mDens;
00049     // Opal-Beaty-Peterson splitting factor
00050     double wSplit;
00051     // Ionisation threshold
00052     double ethr;
00053   };
00054
00055   std::vector<Cluster> m_clusters;
00056
00057   // Mean free path
00058   double m_mfp = 0.;
00059   // Stopping power
00060   double m_dedx = 0.;
00061
00062   static bool Setup(Medium* gas, std::vector<Parameters>& par,
00063                     std::vector<double>& frac);
00064   static bool Update(const double density, const double beta2,
00065                      const std::vector<Parameters>& par,
00066                      const std::vector<double>& frac, std::vector<double>& prob,
00067                      double& mfp, double& dedx);
00068   static double Delta(const double x, const Parameters& par);
00069   static double Esec(const double e0, const Parameters& par);
00070 };
00071 }  // namespace Garfield
00072
00073 #endif
```

## 8.145 /builds/garfield/docs/source/Include/Garfield/TrackHeed.hh File Reference

#include <memory>
#include <vector>
#include "Garfield/Track.hh"

Classes

- class Garfield::TrackHeed

  Generate tracks using Heed++.
- struct Garfield::TrackHeed::Electron
- struct Garfield::TrackHeed::Photon
- struct Garfield::TrackHeed::Cluster

Namespaces

- namespace Heed
- namespace Garfield

## 8.146 TrackHeed.hh

Go to the documentation of this file.

```
00001 #ifndef G__TRACK__HEED__H
00002 #define G__TRACK__HEED__H
00003
00004 #include <memory>
00005 #include <vector>
00006
00007 #include "Garfield/Track.hh"
00008
00009 namespace Heed {
00010 class gparticle;
00011 class particle_def;
00012 class HeedParticle;
00013 class HeedCondElectron;
00014 class HeedMatterDef;
00015 class GasDef;
00016 class MatterDef;
00017 class EnergyMesh;
00018 class EnTransfCS;
00019 class ElElasticScat;
00020 class ElElasticScatLowSigma;
```

```
00021 class PairProd;
00022 class HeedDeltaElectronCS;
00023 class HeedPhoton;
00024 }  // namespace Heed
00025
00026 namespace Garfield {
00027
00028 class HeedChamber;
00029 class HeedFieldMap;
00030 class Medium;
00031
00033
00034 class TrackHeed : public Track {
00035  public:
00036   struct Electron {
00037     double x = 0.;
00038     double y = 0.;
00039     double z = 0.;
00040     double t = 0.;
00041     double e = 0.;
00042     double dx = 0.;
00043     double dy = 0.;
00044     double dz = 0.;
00045   };
00046
00047   struct Photon {
00048     double x = 0.;
00049     double y = 0.;
00050     double z = 0.;
00051     double t = 0.;
00052     double e = 0.;
00053     double dx = 0.;
00054     double dy = 0.;
00055     double dz = 0.;
00056   };
00057
00058   struct Cluster {
00059     double x, y, z, t;
00060     double energy;
00061     double extra;
00062     std::vector<Photon> photons;
00063     std::vector<Electron> electrons;
00064     std::vector<Electron> ions;
00065   };
00066
00068   TrackHeed() : TrackHeed(nullptr) {}
00070   TrackHeed(Sensor* sensor);
00072   virtual ~TrackHeed();
00073
00074   bool NewTrack(const double x0, const double y0, const double z0,
00075                 const double t0, const double dx0, const double dy0,
00076                 const double dz0) override;
00077   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00078
00079   bool GetCluster(double& xc, double& yc, double& zc, double& tc, int& nc,
00080                   double& ec, double& extra);
00081   bool GetCluster(double& xc, double& yc, double& zc, double& tc, int& ne,
00082                   int& ni, double& ec, double& extra);
00092   bool GetCluster(double& xc, double& yc, double& zc, double& tc, int& ne,
00093                   int& ni, int& np, double& ec, double& extra);
00102   bool GetElectron(const unsigned int i, double& x, double& y, double& z,
00103                    double& t, double& e, double& dx, double& dy, double& dz);
00109   bool GetIon(const unsigned int i, double& x, double& y, double& z,
00110               double& t) const;
00111
00119   bool GetPhoton(const unsigned int i, double& x, double& y, double& z,
00120                  double& t, double& e, double& dx, double& dy,
00121                  double& dz) const;
00122
00123   double GetClusterDensity() override;
00124   double GetStoppingPower() override;
00126   double GetW() const;
00128   double GetFanoFactor() const;
00130   double GetPhotoAbsorptionCrossSection(const double e) const;
00131
00133   bool Initialise(Medium* medium, const bool verbose = false);
00134
00141   Cluster TransportDeltaElectron(const double x0, const double y0,
00142                                  const double z0, const double t0,
00143                                  const double e0, const double dx0,
00144                                  const double dy0, const double dz0);
00145
00153   void TransportDeltaElectron(const double x0, const double y0, const double z0,
00154                               const double t0, const double e0,
00155                               const double dx0, const double dy0,
00156                               const double dz0, int& ne, int& ni);
00164   void TransportDeltaElectron(const double x0, const double y0, const double z0,
```

```
00165                         const double t0, const double e0,
00166                         const double dx0, const double dy0,
00167                         const double dz0, int& ne);
00168
00175   Cluster TransportPhoton(const double x0, const double y0, const double z0,
00176                           const double t0, const double e0, const double dx0,
00177                           const double dy0, const double dz0);
00178
00188   void TransportPhoton(const double x0, const double y0, const double z0,
00189                        const double t0, const double e0, const double dx0,
00190                        const double dy0, const double dz0, int& ne, int& ni,
00191                        int& np);
00192
00201   void TransportPhoton(const double x0, const double y0, const double z0,
00202                        const double t0, const double e0, const double dx0,
00203                        const double dy0, const double dz0, int& ne, int& ni);
00211   void TransportPhoton(const double x0, const double y0, const double z0,
00212                        const double t0, const double e0, const double dx0,
00213                        const double dy0, const double dz0, int& ne);
00214
00216   void EnableElectricField();
00218   void DisableElectricField();
00220   void EnableMagneticField();
00222   void DisableMagneticField();
00223
00234   void SetSteppingLimits(const double maxStep, const double radStraight,
00235                          const double stepAngleStraight,
00236                          const double stepAngleCurved) {
00237     m_maxStep = maxStep;
00238     m_radStraight = radStraight;
00239     m_stepAngleStraight = stepAngleStraight;
00240     m_stepAngleCurved = stepAngleCurved;
00241   }
00242   void GetSteppingLimits(double& maxStep, double& radStraight,
00243                          double& stepAngleStraight, double& stepAngleCurved) {
00244     maxStep = m_maxStep;
00245     radStraight = m_radStraight;
00246     stepAngleStraight = m_stepAngleStraight;
00247     stepAngleCurved = m_stepAngleCurved;
00248   }
00249
00250   void CrossInactiveMedia(const bool on = true) { m_crossInactiveMedia = on; }
00251   void EnableCoulombScattering(const bool on = true) {
00252     m_coulombScattering = on;
00253   }
00255   void EnableDeltaElectronTransport() { m_doDeltaTransport = true; }
00257   void DisableDeltaElectronTransport() { m_doDeltaTransport = false; }
00258
00260   void EnablePhotonReabsorption(const bool on = true) {
00261     m_doPhotonReabsorption = on;
00262   }
00263
00265   void EnablePhotoAbsorptionCrossSectionOutput(const bool on) {
00266     m_usePacsOutput = on;
00267   }
00272   void SetEnergyMesh(const double e0, const double e1, const int nsteps);
00273
00276   void SetParticleUser(const double m, const double z);
00277
00278   void EnableOneStepFly(const bool on) { m_oneStepFly = on; }
00279
00280 private:
00281   // Prevent usage of copy constructor and assignment operator
00282   TrackHeed(const TrackHeed& heed);
00283   TrackHeed& operator=(const TrackHeed& heed);
00284
00285   bool m_oneStepFly = false;
00286
00287   bool m_ready = false;
00288   bool m_hasActiveTrack = false;
00289
00290   double m_mediumDensity = -1.;
00291   std::string m_mediumName = "";
00292
00293   bool m_usePacsOutput = false;
00294
00295   bool m_doPhotonReabsorption = true;
00296   bool m_crossInactiveMedia = false;
00297   bool m_coulombScattering = false;
00298   bool m_useBfieldAuto = true;
00299   bool m_doDeltaTransport = true;
00300
00301   std::vector<Cluster> m_clusters;
00302   size_t m_cluster = 0;
00303
00304   // Particle properties
00305   std::unique_ptr<Heed::particle_def> m_particle_def;
```

```
00306   // Material properties
00307   std::unique_ptr<Heed::HeedMatterDef> m_matter;
00308   std::unique_ptr<Heed::GasDef> m_gas;
00309   std::unique_ptr<Heed::MatterDef> m_material;
00310
00311   // Energy mesh
00312   double m_emin = 2.e-6;
00313   double m_emax = 2.e-1;
00314   unsigned int m_nEnergyIntervals = 200;
00315   std::unique_ptr<Heed::EnergyMesh> m_energyMesh;
00316
00317   // Cross-sections
00318   std::unique_ptr<Heed::EnTransfCS> m_transferCs;
00319   std::unique_ptr<Heed::ElElasticScat> m_elScat;
00320   std::unique_ptr<Heed::ElElasticScatLowSigma> m_lowSigma;
00321   std::unique_ptr<Heed::PairProd> m_pairProd;
00322   std::unique_ptr<Heed::HeedDeltaElectronCS> m_deltaCs;
00323
00324   // Interface classes
00325   std::unique_ptr<HeedChamber> m_chamber;
00326   std::unique_ptr<HeedFieldMap> m_fieldMap;
00327
00328   // Bounding box
00329   double m_lX = 0., m_lY = 0., m_lZ = 0.;
00330   double m_cX = 0., m_cY = 0., m_cZ = 0.;
00331
00332   // Stepping parameters.
00334   double m_maxStep = 100.;
00336   double m_radStraight = 1000.;
00338   double m_stepAngleStraight = 0.1;
00340   double m_stepAngleCurved = 0.2;
00341
00342   bool SetupGas(Medium* medium);
00343   bool SetupMaterial(Medium* medium);
00344   bool SetupDelta(const std::string& databasePath);
00345   bool AddCluster(Heed::HeedPhoton* virtualPhoton,
00346                   std::vector<Cluster>& clusters);
00347   void AddElectrons(
00348       const std::vector<Heed::HeedCondElectron>& conductionElectrons,
00349       std::vector<Electron>& electrons);
00350   bool IsInside(const double x, const double y, const double z);
00351   bool UpdateBoundingBox(bool& update);
00352 };
00353 }  // namespace Garfield
00354
00355 #endif
```

## 8.147 /builds/garfield/docs/source/Include/Garfield/TrackPAI.hh File Reference

#include <array>
#include <string>
#include <vector>
#include "Garfield/Track.hh"

### Classes

- class Garfield::TrackPAI
- struct Garfield::TrackPAI::Cluster

### Namespaces

- namespace Garfield

## 8.148 TrackPAI.hh

Go to the documentation of this file.

```
00001 #ifndef G_TRACK_PAI
00002 #define G_TRACK_PAI
00003
00004 #include <array>
00005 #include <string>
00006 #include <vector>
00007
00008 #include "Garfield/Track.hh"
00009
00010 namespace Garfield {
```

```
00011
00013 class Medium;
00014 class TrackPAI : public Track {
00015  public:
00016   struct Cluster {
00017     double x, y, z, t;
00018     double energy;
00019   };
00020
00021   // Constructor
00022   TrackPAI();
00023   // Destructor
00024   virtual ~TrackPAI() {}
00025
00026   bool NewTrack(const double x0, const double y0, const double z0,
00027                 const double t0, const double dx0, const double dy0,
00028                 const double dz0) override;
00029
00030   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00031
00032   double GetClusterDensity() override;
00033   double GetStoppingPower() override;
00034
00035  private:
00036   // Particle speed.
00037   double m_speed = 0.;
00038   // Max. energy transfer in a collision
00039   double m_emax = 0.;
00040
00041   // Total inelastic mean free path
00042   double m_imfp = 0.;
00043   // Stopping power
00044   double m_dedx = 0.;
00045
00046   // Dielectric function
00047   static constexpr size_t m_nSteps = 1000;
00048   std::array<double, m_nSteps> m_eps1;
00049   std::array<double, m_nSteps> m_eps2;
00050   std::array<double, m_nSteps> m_epsInt;
00051
00052   // Tables for interpolation of cumulative distribution functions
00053   std::array<double, m_nSteps> m_energies;
00054   std::array<double, m_nSteps> m_cdf;
00055   std::array<double, m_nSteps> m_rutherford;
00056
00057   std::vector<Cluster> m_clusters;
00058
00059   // Medium properties
00060   std::string m_mediumName = "";
00061   double m_mediumDensity = 0.;
00062   double m_electronDensity = 0.;
00063
00064   bool SetupMedium(Medium* medium);
00065   bool SetupCrossSectionTable();
00066
00067   double ComputeMaxTransfer() const;
00068
00069   double ComputeCsTail(const double emin, const double emax);
00070   double ComputeDeDxTail(const double emin, const double emax);
00071
00072   std::pair<double, double> SampleEnergyDeposit(const double u) const;
00073   double SampleAsymptoticCs(double u) const;
00074   double SampleAsymptoticCsSpinZero(const double emin, double u) const;
00075   double SampleAsymptoticCsSpinHalf(const double emin, double u) const;
00076   double SampleAsymptoticCsSpinOne(const double emin, double u) const;
00077   double SampleAsymptoticCsElectron(const double emin, double u) const;
00078   double SampleAsymptoticCsPositron(const double emin, double u) const;
00079
00080   double LossFunction(const double eps1, const double eps2) const {
00081     const double eps = eps1 * eps1 + eps2 * eps2;
00082     return eps > 0. ? eps2 / eps : 0.;
00083   }
00084 };
00085 }  // namespace Garfield
00086
00087 #endif
```

## 8.149 /builds/garfield/docs/source/Include/Garfield/TrackSimple.hh File Reference

#include <vector>
#include "Garfield/Track.hh"

Classes

- class Garfield::TrackSimple

  Generate tracks based on a cluster density given by the user.
- struct Garfield::TrackSimple::Cluster

Namespaces

- namespace Garfield

## 8.150 TrackSimple.hh

Go to the documentation of this file.

```
00001 #ifndef G__TRACK__SIMPLE__H
00002 #define G__TRACK__SIMPLE__H
00003
00004 #include <vector>
00005
00006 #include "Garfield/Track.hh"
00007
00008 namespace Garfield {
00009
00011
00012 class TrackSimple : public Track {
00013  public:
00014   struct Cluster {
00015     double x, y, z, t;
00016     double energy;
00017   };
00018
00020   TrackSimple() : TrackSimple(nullptr) {}
00022   TrackSimple(Sensor* sensor);
00024   virtual ~TrackSimple() {}
00025
00027   void SetEqualSpacing() { m__useEqualSpacing = true; }
00029   void SetExponentialSpacing() { m__useEqualSpacing = false; }
00030
00032   void SetClusterDensity(const double d);
00033   double GetClusterDensity() override;
00035   void SetStoppingPower(const double dedx);
00036   double GetStoppingPower() override;
00037
00038   bool NewTrack(const double x0, const double y0, const double z0,
00039                 const double t0, const double dx0, const double dy0,
00040                 const double dz0) override;
00041   const std::vector<Cluster>& GetClusters() const { return m__clusters; }
00042
00043  protected:
00044   // Mean free path (mean spacing between adjacent clusters)
00045   double m__mfp = 0.04;
00046   // Average energy per cluster
00047   double m__eloss = 2530.;
00048
00049   bool m__useEqualSpacing = false;
00050
00051   std::vector<Cluster> m__clusters;
00052 };
00053 }  // namespace Garfield
00054
00055 #endif
```

## 8.151 /builds/garfield/docs/source/Include/Garfield/TrackSrim.hh File Reference

#include <array>
#include <vector>
#include "Garfield/Track.hh"

Classes

- class Garfield::TrackSrim

  Generate tracks based on SRIM energy loss, range and straggling tables.
- struct Garfield::TrackSrim::Cluster

Namespaces

- namespace Garfield

## 8.152 TrackSrim.hh

Go to the documentation of this file.

```
00001 #ifndef G__TRACK__SRIM__H
00002 #define G__TRACK__SRIM__H
00003
00004 #include <array>
00005 #include <vector>
00006
00007 #include "Garfield/Track.hh"
00008
00009 namespace Garfield {
00010 class Medium;
00013
00014 class TrackSrim : public Track {
00015  public:
00017   TrackSrim() : TrackSrim(nullptr) {}
00019   TrackSrim(Sensor* sensor);
00021   virtual ~TrackSrim() {}
00022
00024   bool ReadFile(const std::string& file);
00026   void Print();
00029   void PlotEnergyLoss();
00031   void PlotRange();
00034   void PlotStraggling();
00035
00039   void SetModel(const int m) { m_model = m; }
00041   int GetModel() const { return m_model; }
00042
00044   void SetTargetClusterSize(const int n) { m_nsize = n; }
00046   int GetTargetClusterSize() const { return m_nsize; }
00047
00049   void SetClustersMaximum(const int n) { m_maxclusters = n; }
00051   int GetClustersMaximum() const { return m_maxclusters; }
00052
00054   void SetWorkFunction(const double w) { m_work = w; }
00056   double GetWorkFunction() const { return m_work; }
00058   void SetFanoFactor(const double f) {
00059     m_fano = f;
00060     m_fset = true;
00061   }
00063   void UnsetFanoFactor() { m_fset = false; }
00065   double GetFanoFactor() const { return m_fano; }
00067   void SetDensity(const double density) { m_rho = density; }
00069   double GetDensity() const { return m_rho; }
00071   void SetAtomicMassNumbers(const double a, const double z) {
00072     m_a = a;
00073     m_z = z;
00074   }
00076   void GetAtomicMassMumbers(double& a, double& z) const {
00077     a = m_a;
00078     z = m_z;
00079   }
00080
00082   void EnableTransverseStraggling(const bool on = true) {
00083     m_useTransStraggle = on;
00084   }
00086   void EnableLongitudinalStraggling(const bool on = true) {
00087     m_useLongStraggle = on;
00088   }
00089
00090   struct Cluster {
00091     double x, y, z, t;
00092     double energy;
00093     double kinetic;
00094     int n;
00095   };
00096
00097   bool NewTrack(const double x0, const double y0, const double z0,
00098                 const double t0, const double dx0, const double dy0,
00099                 const double dz0) override;
00100   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00101   bool GetCluster(double& xc, double& yc, double& zc, double& tc, int& nc,
00102                   double& ec, double& extra);
00103
00104  protected:
00106   bool m_useTransStraggle = true;
00108   bool m_useLongStraggle = false;
00109
00111   bool m_chargeset = false;
00113   double m_qion = 1.;
```

```
00115   double m_mion = -1.;
00117   double m_rho = -1.;
00119   double m_work = -1.;
00121   bool m_fset = false;
00123   double m_fano = -1.;
00125   double m_a = -1.;
00127   double m_z = -1.;
00128
00130   int m_maxclusters = -1;
00132   std::vector<double> m_ekin;
00134   std::vector<double> m_emloss;
00136   std::vector<double> m_hdloss;
00138   std::vector<double> m_range;
00140   std::vector<double> m_transstraggle;
00142   std::vector<double> m_longstraggle;
00143
00145   size_t m_currcluster = 0;
00148   unsigned int m_model = 4;
00150   int m_nsize = -1;
00151   std::vector<Cluster> m_clusters;
00152
00153   double Xi(const double x, const double beta2, const double edens) const;
00154   double DedxEM(const double e) const;
00155   double DedxHD(const double e) const;
00156   bool PreciseLoss(const double step, const double estart, double& deem,
00157                    double& dehd) const;
00158   bool EstimateRange(const double ekin, const double step,
00159                      double& stpmax) const;
00160   bool SmallestStep(const double ekin, const double edens, double de,
00161                     double step, double& stpmin);
00162   Medium* GetMedium(const std::array<double, 3>& x) const;
00163   double Terminate(const std::array<double, 3>& x0,
00164                    const std::array<double, 3>& v0, const double step0) const;
00165   double TerminateBfield(const std::array<double, 3>& x0,
00166                          const std::array<double, 3>& v0, const double dt0,
00167                          const double vmag) const;
00168   double RndmEnergyLoss(const double ekin, const double de, const double step,
00169                         const double edens) const;
00170 };
00171 }  // namespace Garfield
00172
00173 #endif
```

## 8.153 /builds/garfield/docs/source/Include/Garfield/TrackTrim.hh File Reference

#include <array>
#include <string>
#include <vector>
#include "Garfield/Track.hh"

Classes

- class Garfield::TrackTrim

  Generate tracks based on TRIM output files.
- struct Garfield::TrackTrim::Cluster

Namespaces

- namespace Garfield

## 8.154 TrackTrim.hh

Go to the documentation of this file.

```
00001 #ifndef G_TRACK_TRIM_H
00002 #define G_TRACK_TRIM_H
00003
00004 #include <array>
00005 #include <string>
00006 #include <vector>
00007
00008 #include "Garfield/Track.hh"
00009
00010 namespace Garfield {
00011
00014
00015 class TrackTrim : public Track {
```

```
00016  public:
00018   TrackTrim() : TrackTrim(nullptr) {}
00020   TrackTrim(Sensor* sensor);
00022   virtual ~TrackTrim() {}
00023
00025   bool ReadFile(const std::string& file, const unsigned int nIons = 0,
00026                 const unsigned int nSkip = 0);
00028   void Print();
00029
00031   void SetCharge(const double q) {
00032     if (fabs(q) > 0.) m_q = q;
00033   }
00035   void SetWorkFunction(const double w) { m_work = w; }
00037   double GetWorkFunction() const { return m_work; }
00039   void SetFanoFactor(const double f) {
00040     m_fano = f;
00041     m_fset = true;
00042   }
00044   void UnsetFanoFactor() { m_fset = false; }
00046   double GetFanoFactor() const { return m_fano; }
00047
00049   void EnableStepSizeLimit(const double dmax) { m_maxStepSize = dmax; }
00051   void DisableStepSizeLimit() { m_maxStepSize = -1.; }
00053   void EnableEnergyLossLimit(const double emax) { m_maxLossPerStep = emax; }
00056   void DisableEnergyLossLimit() { m_maxLossPerStep = -1.; }
00057
00058   void SetParticle(const std::string& part) override;
00059
00060   struct Cluster {
00061     double x, y, z;
00062     double t;
00063     double energy;
00064     double ekin;
00065     int n;
00066   };
00067
00068   bool NewTrack(const double x0, const double y0, const double z0,
00069                 const double t0, const double dx0, const double dy0,
00070                 const double dz0) override;
00071   const std::vector<Cluster>& GetClusters() const { return m_clusters; }
00072   bool GetCluster(double& xc, double& yc, double& zc, double& tc, int& nc,
00073                   double& ec, double& extra);
00074
00075  protected:
00077   double m_work = -1.;
00079   bool m_fset = false;
00081   double m_fano = -1.;
00082
00084   double m_ekin = 0.;
00086   std::vector<std::vector<std::array<float, 6> > > m_ions;
00088   size_t m_ion = 0;
00089
00091   std::vector<Cluster> m_clusters;
00093   size_t m_cluster = 0;
00094
00096   double m_maxStepSize = -1.;
00098   double m_maxLossPerStep = -1.;
00099
00100   void AddIon(const std::vector<float>& x, const std::vector<float>& y,
00101               const std::vector<float>& z, const std::vector<float>& dedx,
00102               const std::vector<float>& ekin);
00103 };
00104 }  // namespace Garfield
00105
00106 #endif
```

## 8.155 /builds/garfield/docs/source/Include/Garfield/Utilities.hh File Reference

#include <algorithm>
#include <sstream>
#include <string>
#include <vector>

### Namespaces

- namespace Garfield

Functions

- void Garfield::ltrim (std::string &line)
- void Garfield::rtrim (std::string &line)
- std::vector< std::string > Garfield::tokenize (const std::string &line)
- bool Garfield::startsWith (const std::string &line, const std::string &s)

## 8.156 Utilities.hh

Go to the documentation of this file.

```
00001 #ifndef G_UTILITIES_H
00002 #define G_UTILITIES_H
00003
00004 #include <algorithm>
00005 #include <sstream>
00006 #include <string>
00007 #include <vector>
00008
00009 namespace Garfield {
00010
00011 inline void ltrim(std::string& line) {
00012   line.erase(line.begin(), std::find_if(line.begin(), line.end(), [](int ch) {
00013                return !std::isspace(ch);
00014              }));
00015 }
00016
00017 inline void rtrim(std::string& line) {
00018   line.erase(std::find_if(line.rbegin(), line.rend(),
00019                           [](int ch) { return !std::isspace(ch); })
00020                .base(),
00021              line.end());
00022 }
00023
00024 inline std::vector<std::string> tokenize(const std::string& line) {
00025   std::vector<std::string> words;
00026   std::istringstream ss(line);
00027   for (std::string word; ss » word;) {
00028     words.push_back(word);
00029   }
00030   return words;
00031 }
00032
00033 inline bool startsWith(const std::string& line, const std::string& s) {
00034   return (line.rfind(s, 0) == 0);
00035 }
00036
00037 }  // namespace Garfield
00038
00039 #endif
```

## 8.157 /builds/garfield/docs/source/Include/Garfield/Vector.hh File Reference

#include <array>
#include <cstddef>

Classes

- class Garfield::Vector< type, Dimension >
- class Garfield::Vec1Impl< type >
- class Garfield::Vec2Impl< type >
- class Garfield::Vec3Impl< type >

Namespaces

- namespace Garfield

Macros

- #define ___GARFIELD_DEFINED_DEVICE___
- #define ___device___
- #define ___host___

Typedefs

- template<typename T, std::size_t D>
  using GarfieldArray = std::array<T, D>
- using Garfield::Vec3D = Vec3Impl<double>
- using Garfield::Vec3F = Vec3Impl<float>

### 8.157.1 Macro Definition Documentation

___device___

#define ___device___
Definition at line 12 of file Vector.hh.

___GARFIELD_DEFINED_DEVICE___

#define ___GARFIELD_DEFINED_DEVICE___
Definition at line 11 of file Vector.hh.

___host___

#define ___host___
Definition at line 13 of file Vector.hh.

### 8.157.2 Typedef Documentation

GarfieldArray

template<typename T, std::size_t D>
using GarfieldArray = std::array<T, D>
Definition at line 16 of file Vector.hh.

## 8.158 Vector.hh

Go to the documentation of this file.

```
00001 #ifndef G_VECTOR_H
00002 #define G_VECTOR_H
00003
00004 #if defined(___NVCC___)
00005 #include <cuda/std/array>
00006 template <typename T, std::size_t D>
00007 using GarfieldArray = cuda::std::array<T, D>;
00008 #else
00009 #include <array>
00010 #if !defined(___device___)
00011 #define ___GARFIELD_DEFINED_DEVICE___
00012 #define ___device___
00013 #define ___host___
00014 #endif
00015 template <typename T, std::size_t D>
00016 using GarfieldArray = std::array<T, D>;
00017 #endif
00018 #include <cstddef>
00019
00020 namespace Garfield {
00021 template <typename type, std::size_t Dimension>
00022 class Vector : public GarfieldArray<type, Dimension> {
00023  public:
00024   template <typename... Args>
00025   ___host___ ___device___ Vector(Args&&... args)
00026       : GarfieldArray<type, Dimension>({std::forward<Args>(args)...}) {}
00027   ___host___ ___device___ Vector() { this->fill(0.); }
00028   ___host___ ___device___ Vector operator*(const double& i) {
00029     Vector ret = *this;
00030     for (std::size_t j = 0; j != ret.size(); ++j) {
00031       ret[j] *= i;
00032     }
00033     return ret;
00034   }
00035   ___host___ ___device___ Vector operator/(const double& i) {
00036     Vector ret = *this;
00037     for (std::size_t j = 0; j != ret.size(); ++j) {
```

```
00038       ret[j] /= i;
00039     }
00040     return ret;
00041   }
00042   template <typename T, std::size_t D>
00043   ___host___ ___device___ Vector operator+(const Vector<T, D>& vec) {
00044     static_assert(Dimension == D, "The vectors don't have the same dimension!");
00045     Vector ret = *this;
00046     for (std::size_t j = 0; j != ret.size(); ++j) {
00047       ret[j] += vec[j];
00048     }
00049     return ret;
00050   }
00051   template <typename T, std::size_t D>
00052   ___host___ ___device___ Vector operator-(const Vector<T, D>& vec) {
00053     static_assert(Dimension == D, "The vectors don't have the same dimension!");
00054     Vector ret = *this;
00055     for (std::size_t j = 0; j != ret.size(); ++j) {
00056       ret[j] -= vec[j];
00057     }
00058     return ret;
00059   }
00060 };
00061
00062 template <typename type>
00063 class Vec1Impl : public Vector<type, 1> {
00064  public:
00065   Vec1Impl() = default;
00066   template <typename... xs>
00067   ___host___ ___device___ Vec1Impl(xs... values) : Vector<type, 3>({values...}) {}
00068   ___host___ ___device___ type& x() noexcept { return this->operator[](0); }
00069   ___host___ ___device___ type x() const noexcept { return this->operator[](0); }
00070 };
00071
00072 template <typename type>
00073 class Vec2Impl : public Vector<type, 2> {
00074  public:
00075   Vec2Impl() = default;
00076   template <typename... xs>
00077   ___host___ ___device___ Vec2Impl(xs... values) : Vector<type, 3>({values...}) {}
00078   ___host___ ___device___ type& x() noexcept { return this->operator[](0); }
00079   ___host___ ___device___ type x() const noexcept { return this->operator[](0); }
00080   ___host___ ___device___ type& y() noexcept { return this->operator[](1); }
00081   ___host___ ___device___ type y() const noexcept { return this->operator[](1); }
00082 };
00083
00084 template <typename type>
00085 class Vec3Impl : public Vector<type, 3> {
00086  public:
00087   Vec3Impl() = default;
00088   template <typename... xs>
00089   ___host___ ___device___ Vec3Impl(xs... values) : Vector<type, 3>({values...}) {}
00090   ___host___ ___device___ type& x() noexcept { return this->operator[](0); }
00091   ___host___ ___device___ type x() const noexcept { return this->operator[](0); }
00092   ___host___ ___device___ type& y() noexcept { return this->operator[](1); }
00093   ___host___ ___device___ type y() const noexcept { return this->operator[](1); }
00094   ___host___ ___device___ type& z() noexcept { return this->operator[](2); }
00095   ___host___ ___device___ type z() const noexcept { return this->operator[](2); }
00096 };
00097
00098 using Vec3D = Vec3Impl<double>;
00099 using Vec3F = Vec3Impl<float>;
00100
00101 }  // namespace Garfield
00102
00103 #if defined(___GARFIELD_DEFINED_DEVICE___)
00104 #undef ___device___
00105 #undef ___host___
00106 #undef ___GARFIELD_DEFINED_DEVICE___
00107 #endif
00108
00109 #endif
```

## 8.159 /builds/garfield/docs/source/Include/Garfield/ViewBase.hh File Reference

#include <array>
#include <memory>
#include <string>
#include <TCanvas.h>

Classes

- class Garfield::ViewBase

  Base class for visualization classes.

Namespaces

- namespace Garfield

## 8.160 ViewBase.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_BASE
00002 #define G_VIEW_BASE
00003
00004 #include <array>
00005 #include <memory>
00006 #include <string>
00007
00008 // #include <TPad.h>
00009 #include <TCanvas.h>
00010
00011 class TPad;
00012 class TVirtualPad;
00013
00014 namespace Garfield {
00015
00016 class Sensor;
00017 class Component;
00018
00020
00021 class ViewBase {
00022  public:
00024   ViewBase() = delete;
00026   ViewBase(const std::string& name);
00028   virtual ~ViewBase() = default;
00029
00031   void SetCanvas(TPad* pad) { m_pad = pad; }
00033   void SetCanvas() { m_pad = nullptr; }
00035   TPad* GetCanvas();
00036
00039   void SetArea(const double xmin, const double ymin, const double xmax,
00040                const double ymax);
00042   virtual void SetArea(const double xmin, const double ymin, const double zmin,
00043                        const double xmax, const double ymax, const double zmax);
00046   void SetArea() {
00047     m_userBox = false;
00048     m_userPlotLimits = false;
00049   }
00050
00055   virtual void SetPlane(const double fx, const double fy, const double fz,
00056                         const double x0, const double y0, const double z0);
00058   virtual void SetPlane(const double fx, const double fy, const double fz,
00059                         const double x0, const double y0, const double z0,
00060                         const double hx, const double hy, const double hz);
00062   void Rotate(const double angle);
00064   void SetPlaneXY();
00066   void SetPlaneXZ();
00068   void SetPlaneYZ();
00070   void SetPlaneZX();
00072   void SetPlaneZY();
00073
00075   void EnableDebugging(const bool on = true) { m_debug = on; }
00076
00078   static std::string FindUnusedFunctionName(const std::string& s);
00080   static std::string FindUnusedHistogramName(const std::string& s);
00082   static std::string FindUnusedCanvasName(const std::string& s);
00083
00084 protected:
00085   std::string m_className = "ViewBase";
00086
00087   // Options
00088   bool m_debug = false;
00089
00090   // Plot axis limits.
00091   bool m_userPlotLimits = false;
00092   double m_xMinPlot = -1., m_xMaxPlot = 1.;
00093   double m_yMinPlot = -1., m_yMaxPlot = 1.;
00094
00095   // Bounding box.
00096   bool m_userBox = false;
00097   double m_xMinBox = -1., m_xMaxBox = 1.;
```

```
00098   double m_yMinBox = -1., m_yMaxBox = 1.;
00099   double m_zMinBox = -1., m_zMaxBox = 1.;
00100
00101   // Viewing plane (FPROJ).
00102   // Default projection: x-y at z = 0.
00103   std::array<std::array<double, 3>, 3> m_proj{
00104       {{1, 0, 0}}, {{0, 1, 0}}, {{0, 0, 0}}};
00105   std::array<double, 4> m_plane{{0, 0, 1, 0}};
00106   // Matrix used for projections (FPRMAT).
00107   std::array<std::array<double, 3>, 3> m_prmat{
00108       {{1, 0, 0}}, {{0, 1, 0}}, {{0, 0, 1}}};
00109
00110   // Update and invert the projection matrix.
00111   void UpdateProjectionMatrix();
00112   // Determine plane coordinates.
00113   template <typename T>
00114   void ToPlane(const T x, const T y, const T z, T& xp, T& yp) const {
00115     xp = m_prmat[0][0] * x + m_prmat[0][1] * y + m_prmat[0][2] * z;
00116     yp = m_prmat[1][0] * x + m_prmat[1][1] * y + m_prmat[1][2] * z;
00117   }
00118   // Determine whether a point is inside the bounding box.
00119   template <typename T>
00120   bool InBox(const std::array<T, 3>& x) const {
00121     if (!m_userBox) return true;
00122     if (x[0] < m_xMinBox || x[0] > m_xMaxBox || x[1] < m_yMinBox ||
00123         x[1] > m_yMaxBox || x[2] < m_yMinBox || x[2] > m_zMaxBox)
00124       return false;
00125     return true;
00126   }
00127   // Clip the line x0 - x1 to the extent of the bounding box.
00128   void Clip(const std::array<float, 3>& x0, const std::array<float, 3>& x1,
00129             std::array<float, 3>& xc) const;
00130   // Draw the projection of a line onto the current viewing plane.
00131   void DrawLine(const std::vector<std::array<float, 3> >& xl, const short col,
00132                 const short lw);
00133
00134   // X-axis label for the current viewing plane.
00135   std::string LabelX();
00136   // Y-axis label for the current viewing plane.
00137   std::string LabelY();
00138   // Description of the current viewing plane.
00139   std::string PlaneDescription();
00140
00141   bool PlotLimits(Sensor* sensor, double& xmin, double& ymin, double& xmax,
00142                   double& ymax) const;
00143   bool PlotLimits(Component* cmp, double& xmin, double& ymin, double& xmax,
00144                   double& ymax) const;
00145   bool PlotLimitsFromUserBox(double& xmin, double& ymin, double& xmax,
00146                              double& ymax) const;
00147   bool PlotLimits(std::array<double, 3>& bbmin, std::array<double, 3>& bbmax,
00148                   double& xmin, double& ymin, double& xmax, double& ymax) const;
00149
00150   static bool RangeSet(TVirtualPad*);
00151   static void SetRange(TVirtualPad* pad, const double x0, const double y0,
00152                        const double x1, const double y1);
00153
00154 private:
00155   // Current pad.
00156   TPad* m_pad = nullptr;
00157   std::unique_ptr<TCanvas> m_canvas;
00158 };
00159 }  // namespace Garfield
00160 #endif
```

## 8.161 /builds/garfield/docs/source/Include/Garfield/ViewCell.hh File Reference

#include <TGeoManager.h>
#include <memory>
#include "Garfield/ViewBase.hh"

### Classes

- class Garfield::ViewCell

  Visualize the "cell" defined in an analytic-field component.

### Namespaces

- namespace Garfield

## 8.162 ViewCell.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_CELL
00002 #define G_VIEW_CELL
00003
00004 #include <TGeoManager.h>
00005
00006 #include <memory>
00007
00008 #include "Garfield/ViewBase.hh"
00009
00010 namespace Garfield {
00011
00012 class ComponentAnalyticField;
00013 class ComponentNeBem2d;
00014
00016
00017 class ViewCell : public ViewBase {
00018  public:
00020   ViewCell();
00022   ViewCell(ComponentAnalyticField* cmp);
00024   ViewCell(ComponentNeBem2d* cmp);
00026   ~ViewCell() = default;
00027
00029   void SetComponent(ComponentAnalyticField* comp);
00030   void SetComponent(ComponentNeBem2d* comp);
00031
00033   void Plot2d();
00035   void Plot3d();
00036
00039   void EnableWireMarkers(const bool on = true) { m_useWireMarker = on; }
00040   void DisableWireMarkers() { EnableWireMarkers(false); }
00041
00042  private:
00043   bool m_useWireMarker = true;
00044
00045   ComponentAnalyticField* m_component = nullptr;
00046   ComponentNeBem2d* m_nebem = nullptr;
00047
00048   // 3D geometry.
00049   std::unique_ptr<TGeoManager> m_geo;
00050
00051   bool Plot(const bool twod);
00052   // Draw a wire in 2D.
00053   void PlotWire(const double x, const double y, const double d, const int type);
00054   // Draw a wire in 3D.
00055   void PlotWire(const double x, const double y, const double d, const int type,
00056                 const double lz);
00057   // Draw a tube in 2D.
00058   void PlotTube(const double x0, const double y0, const double r, const int n);
00059   // Draw a tube in 3D.
00060   void PlotTube(const double x0, const double y0, const double r1,
00061                 const double r2, const int n, const double lz);
00062   // Draw a plane in 2D.
00063   void PlotPlane(const double x0, const double y0, const double x1,
00064                  const double y1);
00065   // Draw a plane in 3D.
00066   void PlotPlane(const double dx, const double dy, const double dz,
00067                  const double x0, const double y0);
00068   // Draw a neBEM 2D layout.
00069   bool PlotNeBem(const bool twod);
00070   // Setup the TGeoManager.
00071   void SetupGeo(const double dx, const double dy, const double dz);
00072 };
00073 }  // namespace Garfield
00074 #endif
```

## 8.163 /builds/garfield/docs/source/Include/Garfield/ViewDrift.hh File Reference

#include <Rtypes.h>
#include <array>
#include <cstddef>
#include <mutex>
#include <vector>
#include "Garfield/GarfieldConstants.hh"
#include "Garfield/ViewBase.hh"

Classes

- class Garfield::ViewDrift

  Visualize drift lines and tracks.

Namespaces

- namespace Garfield

## 8.164 ViewDrift.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_DRIFT
00002 #define G_VIEW_DRIFT
00003
00004 #include <Rtypes.h>
00005
00006 #include <array>
00007 #include <cstddef>
00008 #include <mutex>
00009 #include <vector>
00010
00011 #include "Garfield/GarfieldConstants.hh"
00012 #include "Garfield/ViewBase.hh"
00013
00014 namespace Garfield {
00015
00017
00018 class ViewDrift : public ViewBase {
00019  public:
00021   ViewDrift();
00023   ~ViewDrift() = default;
00024
00026   void Clear();
00027
00029   void Plot(const bool twod = false, const bool axis = true,
00030             const bool snapshot = false);
00032   void Plot2d(const bool axis = true, const bool snapshot = false);
00034   void Plot3d(const bool axis = true, const bool ogl = true,
00035               const bool snapshot = false);
00036
00038   void EnableClusterMarkers(const bool on = true) { m_drawClusters = on; }
00040   void SetClusterMarkerSize(const double size);
00042   void SetCollisionMarkerSize(const double size);
00043
00045   void SetColourElectrons(const short col) { m_colElectron = col; }
00047   void SetColourHoles(const short col) { m_colHole = col; }
00049   void SetColourNegativeIons(const short col) { m_colNegativeIon = col; }
00051   void SetColourIons(const short col) { m_colIon = col; }
00053   void SetColourTracks(const short col) { m_colTrack = col; }
00055   void SetColourPhotons(const short col) { m_colPhoton = col; }
00057   void SetColourExcitations(const short col) { m_colExcitation = col; }
00059   void SetColourIonisations(const short col) { m_colIonisation = col; }
00061   void SetColourAttachments(const short col) { m_colAttachment = col; }
00063   void SetElectronsToFront() { m_eTop = true; }
00064
00066   std::size_t GetNumberOfDriftLines() const { return m_driftLines.size(); }
00068   void GetDriftLine(const std::size_t i,
00069                     std::vector<std::array<float, 3> >& driftLine,
00070                     bool& electron) const;
00071
00072   // Functions used by the transport classes.
00073   std::size_t NewDriftLine(const Particle particle, const std::size_t np,
00074                            const float x0, const float y0, const float z0);
00075   void NewChargedParticleTrack(const std::size_t np, std::size_t& id,
00076                                const float x0, const float y0, const float z0);
00077
00078   void SetDriftLinePoint(const std::size_t iL, const std::size_t iP,
00079                          const float x, const float y, const float z);
00080   void AddDriftLinePoint(const std::size_t iL, const float x, const float y,
00081                          const float z);
00082   void SetTrackPoint(const std::size_t iL, const std::size_t iP, const float x,
00083                      const float y, const float z);
00084   void AddTrackPoint(const std::size_t iL, const float x, const float y,
00085                      const float z);
00086   void AddExcitation(const float x, const float y, const float z);
00087   void AddIonisation(const float x, const float y, const float z);
00088   void AddAttachment(const float x, const float y, const float z);
00089
00090   void AddPhoton(const float x0, const float y0, const float z0, const float x1,
00091                  const float y1, const float z1);
00092
```

```
00093   friend class ViewFEMesh;
00094
00095  private:
00096   std::mutex m_mutex;
00097
00098   std::vector<std::pair<std::vector<std::array<float, 3> >, Particle> >
00099       m_driftLines;
00100
00101   std::vector<std::vector<std::array<float, 3> > > m_tracks;
00102   std::vector<std::array<std::array<float, 3>, 2> > m_photons;
00103
00104   std::vector<std::array<float, 3> > m_exc;
00105   std::vector<std::array<float, 3> > m_ion;
00106   std::vector<std::array<float, 3> > m_att;
00107
00108   double m_markerSizeCluster = 0.01;
00109   double m_markerSizeCollision = 0.5;
00110
00111   short m_colTrack = kGreen + 3;
00112   short m_colPhoton = kBlue + 1;
00113   short m_colElectron = kOrange - 3;
00114   short m_colHole = kRed + 1;
00115   short m_colIon = kRed + 1;
00116   short m_colNegativeIon = kGray + 2;
00117   short m_colExcitation = kGreen + 3;
00118   short m_colIonisation = kOrange - 3;
00119   short m_colAttachment = kCyan + 3;
00120
00121   bool m_drawClusters = false;
00122
00123   bool m_eTop = false;
00124
00125   bool SetPlotLimits2d();
00126   bool SetPlotLimits3d();
00127   void DrawMarkers2d(const std::vector<std::array<float, 3> >& points,
00128                      const short col, const double size);
00129   void DrawMarkers3d(const std::vector<std::array<float, 3> >& points,
00130                      const short col, const double size);
00131 };
00132 }  // namespace Garfield
00133 #endif
```

## 8.165 /builds/garfield/docs/source/Include/Garfield/ViewFEMesh.hh File Reference

#include <TGeoManager.h>
#include <TMatrixDfwd.h>
#include <map>
#include <string>
#include <vector>
#include "Garfield/ViewBase.hh"

### Classes

- class Garfield::ViewFEMesh

  Draw the mesh of a field-map component.

### Namespaces

- namespace Garfield

### Typedefs

- using Garfield::ViewMesh = ViewFEMesh

## 8.166 ViewFEMesh.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_FE_MESH
00002 #define G_VIEW_FE_MESH
00003
00004 #include <TGeoManager.h>
00005 #include <TMatrixDfwd.h>
00006
00007 #include <map>
```

```
00008 #include <string>
00009 #include <vector>
00010
00011 #include "Garfield/ViewBase.hh"
00012
00013 class TGaxis;
00014 class TGeoVolume;
00015 class TGeoMedium;
00016
00017 namespace Garfield {
00018
00019 class Component;
00020 class ComponentCST;
00021 class ViewDrift;
00022
00024
00025 class ViewFEMesh : public ViewBase {
00026  public:
00028   ViewFEMesh() : ViewFEMesh(nullptr) {}
00030   ViewFEMesh(Component* cmp);
00032   ~ViewFEMesh();
00033
00035   void SetComponent(Component* cmp);
00036
00037   void SetPlane(const double fx, const double fy, const double fz,
00038                 const double x0, const double y0, const double z0) override;
00039   void SetPlane(const double fx, const double fy, const double fz,
00040                 const double x0, const double y0, const double z0,
00041                 const double hx, const double hy, const double hz) override;
00042
00043   // Axes
00044   void SetXaxis(TGaxis* ax);
00045   void SetYaxis(TGaxis* ay);
00046   void SetXaxisTitle(const std::string& xtitle) { m_xaxisTitle = xtitle; }
00047   void SetYaxisTitle(const std::string& ytitle) { m_yaxisTitle = ytitle; }
00048   void EnableAxes() { m_drawAxes = true; }
00049   void DisableAxes() { m_drawAxes = false; }
00050
00052   bool Plot(const bool twod = true, const bool outline = false);
00053
00055   void SetFillMesh(const bool f) { m_fillMesh = f; }
00056
00058   void SetDrawViewRegion(bool do_draw) { m_drawViewRegion = do_draw; }
00059   bool GetDrawViewRegion(void) const { return m_drawViewRegion; }
00060
00063   void SetColor(int matID, int colorID) { m_colorMap[matID] = colorID; }
00064   void SetColors(const std::map<int, int>& colors) {
00065     for (const auto& c : colors) SetColor(c.first, c.second);
00066   }
00067   void SetFillColor(int matID, int colorID) {
00068     m_colorMap_fill[matID] = colorID;
00069   }
00070   void SetFillColors(const std::map<int, int>& colors) {
00071     for (const auto& c : colors) SetFillColor(c.first, c.second);
00072   }
00074   void SetViewDrift(ViewDrift* vd) { m_viewDrift = vd; }
00075
00077   void SetFillMeshWithBorders() {
00078     m_plotMeshBorders = true;
00079     m_fillMesh = true;
00080   }
00081
00083   void CreateDefaultAxes();
00084
00086   void DisableMaterial(int materialID) {
00087     m_disabledMaterial[materialID] = true;
00088   }
00089
00090  private:
00091   // Options
00092   bool m_fillMesh = false;
00093
00094   // Intersection of viewing plane with plotted area in planar coordinates
00095   bool m_drawViewRegion = false;
00096   std::vector<double> m_viewRegionX;
00097   std::vector<double> m_viewRegionY;
00098
00099   // Field map object
00100   Component* m_cmp = nullptr;
00101
00102   // Optional associated ViewDrift object
00103   ViewDrift* m_viewDrift = nullptr;
00104   bool m_plotMeshBorders = false;
00105
00106   // Axes
00107   TGaxis* m_xaxis = nullptr;
00108   TGaxis* m_yaxis = nullptr;
```

```
00109   std::string m_xaxisTitle = "";
00110   std::string m_yaxisTitle = "";
00111   bool m_drawAxes = false;
00112
00113   // The color map
00114   std::map<int, int> m_colorMap;
00115   std::map<int, int> m_colorMap_fill;
00116
00117   // Disabled materials -> not shown in the mesh view
00118   std::map<int, bool> m_disabledMaterial;
00119
00120   std::vector<TGeoVolume*> m_volumes;
00121   std::vector<TGeoMedium*> m_media;
00122   std::unique_ptr<TGeoManager> m_geoManager;
00123
00124   // Element plotting methods
00125   void DrawElements2d();
00126   void DrawElements3d();
00127   void DrawCST(ComponentCST* componentCST);
00128   void DrawBorders2d();
00129   void DrawBorders3d();
00130
00131   typedef std::vector<size_t> Facet;
00132   void AddFacets(const size_t i,
00133                  const std::vector<std::vector<Facet> >& elementFacets,
00134                  const std::map<Facet, std::vector<size_t> >& facetElements,
00135                  std::vector<Facet>& facets, std::vector<bool>& done) const;
00136   bool FacetSign(const Facet& f, const size_t element) const;
00137   void DrawDriftLines2d();
00138   void DrawDriftLines3d();
00139
00140   bool GetPlotLimits();
00141
00143   bool InView(const double x, const double y) const;
00144
00145   bool LinesCrossed(double x1, double y1, double x2, double y2, double u1,
00146                     double v1, double u2, double v2, double& xc,
00147                     double& yc) const;
00148   bool IntersectPlaneArea(double& xmin, double& ymin, double& xmax,
00149                           double& ymax);
00150   bool OnLine(double x1, double y1, double x2, double y2, double u,
00151               double v) const;
00152   void RemoveCrossings(std::vector<double>& x, std::vector<double>& y);
00153   bool PlaneCut(double x1, double y1, double z1, double x2, double y2,
00154                 double z2, TMatrixD& xMat);
00155   void ClipToView(std::vector<double>& px, std::vector<double>& py,
00156                   std::vector<double>& cx, std::vector<double>& cy);
00157   bool IsInPolygon(double x, double y, const std::vector<double>& px,
00158                    const std::vector<double>& py, bool& edge) const;
00159
00160   void Reset();
00161 };
00162
00163 using ViewMesh = ViewFEMesh;
00164
00165 }  // namespace Garfield
00166 #endif
```

## 8.167 /builds/garfield/docs/source/Include/Garfield/ViewField.hh File Reference

#include <Rtypes.h>
#include "Garfield/ViewBase.hh"

### Classes

- class Garfield::ViewField

  Visualize the potential or electric field of a component or sensor.

### Namespaces

- namespace Garfield

## 8.168 ViewField.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_FIELD
00002 #define G_VIEW_FIELD
```

```
00003
00004 #include <Rtypes.h>
00005
00006 #include "Garfield/ViewBase.hh"
00007
00008 namespace Garfield {
00009
00010 class Sensor;
00011 class Component;
00012
00014
00015 class ViewField : public ViewBase {
00016  public:
00018   ViewField();
00020   ViewField(Sensor* sensor);
00022   ViewField(Component* component);
00024   ~ViewField() = default;
00025
00027   void SetSensor(Sensor* s);
00029   void SetComponent(Component* c);
00030
00032   void SetVoltageRange(const double vmin, const double vmax);
00034   void SetElectricFieldRange(const double emin, const double emax);
00036   void SetWeightingFieldRange(const double wmin, const double wmax);
00038   void SetMagneticFieldRange(const double bmin, const double bmax);
00039
00041   void SetNumberOfContours(const unsigned int n);
00043   void SetNumberOfSamples1d(const unsigned int n);
00045   void SetNumberOfSamples2d(const unsigned int nx, const unsigned int ny);
00046
00060   void PlotContour(const std::string& option = "v");
00066   void Plot(const std::string& option = "v",
00067             const std::string& drawopt = "arr");
00074   void PlotProfile(const double x0, const double y0, const double z0,
00075                    const double x1, const double y1, const double z1,
00076                    const std::string& option = "v",
00077                    const bool normalised = true);
00078
00083   void PlotContourWeightingField(const std::string& label,
00084                                  const std::string& option);
00091   void PlotWeightingField(const std::string& label, const std::string& option,
00092                           const std::string& drawopt, const double t = 0.);
00093
00101   void PlotProfileWeightingField(const std::string& label, const double x0,
00102                                  const double y0, const double z0,
00103                                  const double x1, const double y1,
00104                                  const double z1,
00105                                  const std::string& option = "v",
00106                                  const bool normalised = true);
00109   void EnableAutoRange(const bool on = true,
00110                        const bool samplePotential = true) {
00111     m_useAutoRange = on;
00112     m_samplePotential = samplePotential;
00113   }
00114
00119   void AcknowledgeStatus(const bool on, const double v0 = 0.) {
00120     m_useStatus = on;
00121     m_vBkg = v0;
00122   }
00123
00125   void PlotFieldLines(const std::vector<double>& x0,
00126                       const std::vector<double>& y0,
00127                       const std::vector<double>& z0, const bool electron = true,
00128                       const bool axis = true, const short col = kOrange - 3);
00130   bool EqualFluxIntervals(const double x0, const double y0, const double z0,
00131                           const double x1, const double y1, const double z1,
00132                           std::vector<double>& xf, std::vector<double>& yf,
00133                           std::vector<double>& zf,
00134                           const unsigned int nPoints = 20) const;
00136   bool FixedFluxIntervals(const double x0, const double y0, const double z0,
00137                           const double x1, const double y1, const double z1,
00138                           std::vector<double>& xf, std::vector<double>& yf,
00139                           std::vector<double>& zf,
00140                           const double interval = 10.) const;
00141
00142  private:
00143   enum class Parameter {
00144     Potential = 0,
00145     Emag,
00146     Ex,
00147     Ey,
00148     Ez,
00149     Bmag,
00150     Bx,
00151     By,
00152     Bz,
00153     Unknown
```

```
00154   };
00155
00156   bool m_useAutoRange = true;
00157   bool m_samplePotential = true;
00158   bool m_useStatus = false;
00159   double m_vBkg = 0.;
00160
00161   // Sensor
00162   Sensor* m_sensor = nullptr;
00163   Component* m_component = nullptr;
00164
00165   // Function range
00166   double m_vmin = 0., m_vmax = 100.;
00167   double m_emin = 0., m_emax = 10000.;
00168   double m_wmin = 0., m_wmax = 100.;
00169   double m_bmin = 0., m_bmax = 10.;
00170
00171   // Number of contours
00172   unsigned int m_nContours = 20;
00173   // Number of points used to draw the functions
00174   unsigned int m_nSamples1d = 1000;
00175   unsigned int m_nSamples2dX = 200;
00176   unsigned int m_nSamples2dY = 200;
00177
00178   bool SetPlotLimits();
00179   void Draw2d(const std::string& option, const bool contour, const bool wfield,
00180               const std::string& electrode, const std::string& drawopt,
00181               const double t = 0.);
00182   void DrawProfile(const double x0, const double y0, const double z0,
00183                    const double x1, const double y1, const double z1,
00184                    const std::string& option, const bool wfield,
00185                    const std::string& electrode, const bool normalised);
00186   Parameter GetPar(const std::string& option, std::string& title,
00187                    bool& bfield) const;
00188   double Efield(const double x, const double y, const double z,
00189                 const Parameter par) const;
00190   double Wfield(const double x, const double y, const double z,
00191                 const Parameter par, const std::string& electrode,
00192                 const double t = 0.) const;
00193   double Bfield(const double x, const double y, const double z,
00194                 const Parameter par) const;
00195 };
00196 }  // namespace Garfield
00197 #endif
```

## 8.169 /builds/garfield/docs/source/Include/Garfield/ViewGeometry.hh File Reference

#include <TGeoManager.h>
#include <memory>
#include <vector>
#include "Garfield/ViewBase.hh"

### Classes

- class Garfield::ViewGeometry

  Visualize a geometry defined using the "native" shapes.

### Namespaces

- namespace Garfield

## 8.170 ViewGeometry.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_GEOMETRY
00002 #define G_VIEW_GEOMETRY
00003
00004 #include <TGeoManager.h>
00005
00006 #include <memory>
00007 #include <vector>
00008
00009 #include "Garfield/ViewBase.hh"
00010
00011 namespace Garfield {
00012
```

```
00013 class GeometrySimple;
00014
00016
00017 class ViewGeometry : public ViewBase {
00018  public:
00020   ViewGeometry() : ViewGeometry(nullptr) {}
00022   ViewGeometry(GeometrySimple* geo);
00024   ~ViewGeometry();
00025
00027   void SetGeometry(GeometrySimple* geo);
00029   void Plot(const bool twod = false);
00031   void Plot2d();
00033   void Plot3d();
00035   void PlotPanels();
00036
00037  private:
00038   GeometrySimple* m_geometry = nullptr;
00039
00040   std::vector<TGeoVolume*> m_volumes;
00041   std::vector<TGeoMedium*> m_media;
00042
00043   std::unique_ptr<TGeoManager> m_geoManager;
00044
00045   void Reset();
00046 };
00047 }  // namespace Garfield
00048 #endif
```

## 8.171 /builds/garfield/docs/source/Include/Garfield/ViewIsochrons.hh File Reference

#include <array>
#include "Garfield/GarfieldConstants.hh"
#include "Garfield/ViewBase.hh"

### Classes

- class Garfield::ViewIsochrons

  Draw equal time contour lines.

### Namespaces

- namespace Garfield

## 8.172 ViewIsochrons.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_ISOCHRONS
00002 #define G_VIEW_ISOCHRONS
00003
00004 #include <array>
00005
00006 #include "Garfield/GarfieldConstants.hh"
00007 #include "Garfield/ViewBase.hh"
00008
00009 namespace Garfield {
00010
00011 class Sensor;
00012 class Component;
00013
00015
00016 class ViewIsochrons : public ViewBase {
00017  public:
00019   ViewIsochrons();
00021   ~ViewIsochrons() = default;
00022
00024   void SetSensor(Sensor* s);
00026   void SetComponent(Component* c);
00027
00037   void PlotIsochrons(const double tstep,
00038                      const std::vector<std::array<double, 3> >& points,
00039                      const bool rev = false, const bool colour = false,
00040                      const bool markers = false,
00041                      const bool plotDriftLines = true);
00042
00045   void DriftElectrons(const bool positive = false) {
00046     m_particle = Particle::Electron;
00047     m_positive = positive;
```

```
00048   }
00050   void DriftIons(const bool negative = false) {
00051     m_particle = Particle::Ion;
00052     m_positive = !negative;
00053   }
00055   void EnableSorting(const bool on = true) { m_sortContours = on; }
00058   void CheckCrossings(const bool on = true) { m_checkCrossings = on; }
00061   void SetAspectRatioSwitch(const double ar);
00064   void SetLoopThreshold(const double thr);
00067   void SetConnectionThreshold(const double thr);
00068
00069  private:
00070   Sensor* m_sensor = nullptr;
00071   Component* m_component = nullptr;
00072
00073   // Type of particle to be used for computing drift lines.
00074   Particle m_particle = Particle::Electron;
00075   bool m_positive = false;
00076
00077   short m_markerStyle = 5;
00078   short m_lineStyle = 2;
00079
00080   bool m_sortContours = true;
00081   double m_aspectRatio = 3.;
00082   double m_loopThreshold = 0.2;
00083   double m_connectionThreshold = 0.2;
00084   bool m_checkCrossings = true;
00085
00086   bool SetPlotLimits();
00087
00088   void ComputeDriftLines(
00089       const double tstep, const std::vector<std::array<double, 3> >& points,
00090       std::vector<std::vector<std::array<double, 3> > >& driftLines,
00091       std::vector<std::array<double, 3> >& startPoints,
00092       std::vector<std::array<double, 3> >& endPoints,
00093       std::vector<int>& statusCodes, const bool rev = false);
00094   void SortContour(
00095       std::vector<std::pair<std::array<double, 4>, unsigned int> >& contour,
00096       bool& circle);
00097 };
00098 }  // namespace Garfield
00099 #endif
```

## 8.173 /builds/garfield/docs/source/Include/Garfield/ViewMedium.hh File Reference

#include <array>
#include <string>
#include <vector>
#include "Garfield/FundamentalConstants.hh"
#include "Garfield/ViewBase.hh"

### Classes

- class Garfield::ViewMedium

  Plot transport coefficients as function of electric and magnetic field.

### Namespaces

- namespace Garfield

## 8.174 ViewMedium.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_MEDIUM
00002 #define G_VIEW_MEDIUM
00003
00004 #include <array>
00005 #include <string>
00006 #include <vector>
00007
00008 #include "Garfield/FundamentalConstants.hh"
00009 #include "Garfield/ViewBase.hh"
00010
00011 namespace Garfield {
00012
00013 class Medium;
```

```
00014
00016
00017 class ViewMedium : public ViewBase {
00018  public:
00020   ViewMedium() : ViewMedium(nullptr) {}
00022   ViewMedium(Medium* medium);
00024   ~ViewMedium() = default;
00025
00027   void SetMedium(Medium* m);
00028
00030   void EnableAutoRangeX(const bool on = true) { m_autoRangeX = on; }
00032   void SetRangeE(const double emin, const double emax, const bool logscale);
00034   void SetRangeB(const double bmin, const double bmax, const bool logscale);
00036   void SetRangeA(const double amin, const double amax, const bool logscale);
00038   void SetRangeEN(const double emin, const double emax, const bool logscale);
00040   void SetRangeEP(const double emin, const double emax, const bool logscale);
00042   void EnableAutoRangeY(const bool on = true) { m_autoRangeY = on; }
00044   void SetRangeY(const double ymin, const double ymax,
00045                  const bool logscale = false);
00046
00048   void SetElectricField(const double efield) { m_efield = efield; }
00050   void SetMagneticField(const double bfield) { m_bfield = bfield; }
00052   void SetAngle(const double angle) { m_angle = angle; }
00053
00054   void EnableExport(const std::string& txtfile) { m_outfile = txtfile; }
00055   void DisableExport() { m_outfile = ""; }
00056
00071   void PlotVelocity(const std::string& carriers, const char xaxis);
00073   void PlotDiffusion(const std::string& carriers, const char xaxis);
00075   void PlotTownsend(const std::string& carriers, const char xaxis);
00077   void PlotAttachment(const std::string& carriers, const char xaxis);
00079   void PlotAlphaEta(const std::string& carriers, const char xaxis);
00080
00090   void PlotElectronVelocity(const char xaxis = 'e', const bool same = false) {
00091     PlotVelocity(GetAxis(xaxis), Charge::Electron, same);
00092   }
00094   void PlotElectronVelocityFluxBulk(const char xaxis = 'e',
00095                                     const bool same = false) {
00096     PlotVelocityFluxBulk(GetAxis(xaxis), Charge::Electron, same);
00097   }
00099   void PlotHoleVelocity(const char xaxis = 'e', const bool same = false) {
00100     PlotVelocity(GetAxis(xaxis), Charge::Hole, same);
00101   }
00103   void PlotIonVelocity(const char xaxis = 'e', const bool same = false) {
00104     PlotVelocity(GetAxis(xaxis), Charge::Ion, same);
00105   }
00107   void PlotElectronDiffusion(const char xaxis = 'e', const bool same = false) {
00108     PlotDiffusion(GetAxis(xaxis), Charge::Electron, same);
00109   }
00111   void PlotHoleDiffusion(const char xaxis = 'e', const bool same = false) {
00112     PlotDiffusion(GetAxis(xaxis), Charge::Hole, same);
00113   }
00115   void PlotIonDiffusion(const char xaxis = 'e', const bool same = false) {
00116     PlotDiffusion(GetAxis(xaxis), Charge::Ion, same);
00117   }
00119   void PlotElectronTownsend(const char xaxis = 'e', const bool same = false) {
00120     Plot(GetAxis(xaxis), Charge::Electron, Parameter::Townsend, same);
00121   }
00123   void PlotElectronReducedTownsendN(const char xaxis = 'r',
00124                                     const bool same = false) {
00125     Plot(GetAxis(xaxis), Charge::Electron, Parameter::AlphaN, same);
00126   }
00128   void PlotElectronReducedTownsendP(const char xaxis = 'p',
00129                                     const bool same = false) {
00130     Plot(GetAxis(xaxis), Charge::Electron, Parameter::AlphaP, same);
00131   }
00133   void PlotElectronTOFIonization(const char xaxis = 'e',
00134                                  const bool same = false) {
00135     Plot(GetAxis(xaxis), Charge::Electron, Parameter::RIonTof, same);
00136   }
00138   void PlotHoleTownsend(const char xaxis = 'e', const bool same = false) {
00139     Plot(GetAxis(xaxis), Charge::Hole, Parameter::Townsend, same);
00140   }
00142   void PlotElectronAttachment(const char xaxis = 'e', const bool same = false) {
00143     Plot(GetAxis(xaxis), Charge::Electron, Parameter::Attachment, same);
00144   }
00146   void PlotElectronTOFAttachment(const char xaxis = 'e',
00147                                  const bool same = false) {
00148     Plot(GetAxis(xaxis), Charge::Electron, Parameter::RAttTof, same);
00149   }
00151   void PlotHoleAttachment(const char xaxis = 'e', const bool same = false) {
00152     Plot(GetAxis(xaxis), Charge::Hole, Parameter::Attachment, same);
00153   }
00154
00156   void PlotElectronLorentzAngle(const char xaxis = 'e',
00157                                 const bool same = false) {
00158     PlotLorentzAngle(GetAxis(xaxis), Charge::Electron, same);
```

```
00159   }
00160
00162   void SetColours(const std::vector<short>& cols) { m_colours = cols; }
00164   void SetLabels(const std::vector<std::string>& labels) { m_labels = labels; }
00165
00166  private:
00167   enum class Parameter {
00168     VelocityE,
00169     VelocityB,
00170     VelocityExB,
00171     VelocityWv,
00172     VelocityWr,
00173     TransverseDiffusion,
00174     LongitudinalDiffusion,
00175     Townsend,
00176     Attachment,
00177     LorentzAngle,
00178     RIonTof,
00179     RAttTof,
00180     AlphaN,
00181     AlphaP
00182   };
00183
00184   enum class Charge { Electron, Hole, Ion };
00185
00186   enum class Axis { E, B, Angle, EoverN, EoverP, None };
00187
00188   Medium* m_medium = nullptr;
00189
00190   // X axis
00191   double m_eMin = 100., m_eMax = 100000.;
00192   double m_bMin = 0., m_bMax = 2.;
00193   double m_aMin = 0., m_aMax = Pi;
00194   double m_enMin = 0.25, m_enMax = 400.;
00195   double m_epMin = 0.13, m_epMax = 132.;
00196   bool m_logE = true;
00197   bool m_logB = false;
00198   bool m_logA = false;
00199   bool m_logEN = true;
00200   bool m_logEP = true;
00201   bool m_logX = true;
00202   bool m_autoRangeX = true;
00203   Axis m_xaxis = Axis::None;
00204
00205   // Y axis
00206   double m_yMin = 0., m_yMax = 1.;
00207   bool m_logY = false;
00208   bool m_autoRangeY = true;
00209
00210   // E-field to use when plotting as function of B-field or angle.
00211   double m_efield = 1000.;
00212   // B-field to use when plotting as function of E-field or angle.
00213   double m_bfield = 0.;
00214   // Angle to use when plotting as function of E-field or B-field.
00215   double m_angle = HalfPi;
00216
00217   std::vector<double> m_xPlot;
00218   std::vector<std::vector<double> > m_yPlot;
00219   std::vector<Parameter> m_par;
00220   std::vector<Charge> m_q;
00221
00222   std::vector<std::vector<double> > m_xGraph;
00223   std::vector<std::vector<double> > m_yGraph;
00224
00225   std::vector<short> m_colours;
00226   std::vector<std::string> m_labels;
00227
00228   std::string m_outfile;
00229
00230   void PlotVelocity(const Axis xaxis, const Charge particle, const bool same);
00231   void PlotVelocityFluxBulk(const Axis xaxis, const Charge particle,
00232                             const bool same);
00233   void PlotDiffusion(const Axis xaxis, const Charge particle, const bool same);
00234   void Plot(const Axis xaxis, const Charge particle, const Parameter par,
00235             const bool same);
00236   void PlotLorentzAngle(const Axis xaxis, const Charge particle,
00237                         const bool same);
00238
00239   void ResetY();
00240   void ResetX(const Axis xaxis);
00241   void Draw();
00242   void Export();
00243
00244   Axis GetAxis(const char xaxis) const;
00245   bool GetGrid(std::array<std::vector<double>, 3>& grid, int& ie, int& ib,
00246                int& ia, const Axis xaxis) const;
00247
```

```
00248   double ConvertToEN(const double e0Vcm);
00249   double UnConvertFromEN(const double e0Td);
00250   double ConvertToEP(const double e0Vcm);
00251   double UnConvertFromEP(const double e0VcmTorr);
00252 };
00253 }  // namespace Garfield
00254 #endif
```

## 8.175 /builds/garfield/docs/source/Include/Garfield/ViewSignal.hh File Reference

#include <Rtypes.h>
#include <array>
#include <string>
#include "Garfield/ViewBase.hh"

Classes

- class Garfield::ViewSignal

  Plot the signal computed by a sensor as a ROOT histogram.

Namespaces

- namespace Garfield

## 8.176 ViewSignal.hh

Go to the documentation of this file.

```
00001 #ifndef G_VIEW_SIGNAL
00002 #define G_VIEW_SIGNAL
00003
00004 #include <Rtypes.h>
00005
00006 #include <array>
00007 #include <string>
00008
00009 #include "Garfield/ViewBase.hh"
00010
00011 class TH1;
00012 class TH1D;
00013
00014 namespace Garfield {
00015
00016 class Sensor;
00017
00019
00020 class ViewSignal : public ViewBase {
00021  public:
00023   ViewSignal() : ViewSignal(nullptr) {}
00025   ViewSignal(Sensor* sensor);
00027   ~ViewSignal() = default;
00028
00030   void SetSensor(Sensor* s);
00031
00048
00049   void PlotSignal(const std::string& label, const std::string& optTotal = "t",
00050                   const std::string& optPrompt = "",
00051                   const std::string& optDelayed = "", const bool same = false);
00052
00054   void SetRangeX(const double xmin, const double xmax);
00056   void UnsetRangeX() { m_userRangeX = false; }
00057
00059   void SetRangeY(const double ymin, const double ymax);
00061   void UnsetRangeY() { m_userRangeY = false; }
00062
00064   void SetLabelY(const std::string& label) { m_labelY = label; }
00065
00067   void EnableLegend(const bool on = true) { m_legend = on; }
00068
00070   void SetColourTotal(const short col) { m_colTotal = col; }
00072   void SetColourElectrons(const short col) { m_colElectrons = col; }
00074   void SetColourIons(const short col) { m_colIons = col; }
00076   void SetColourHoles(const short col) { m_colIons = col; }
00077
00079   void SetColourDelayed(const short colTotal,
00080                         const short colElectrons = kYellow - 7,
00081                         const short colIons = kRed - 9) {
```

```
00082     m_colDelayed = {colTotal, colElectrons, colIons};
00083   }
00084
00085  private:
00086   // Sensor.
00087   Sensor* m_sensor = nullptr;
00088
00089   // Axis range.
00090   double m_xmin = 0.;
00091   double m_xmax = 0.;
00092   bool m_userRangeX = false;
00093   double m_ymin = 0.;
00094   double m_ymax = 0.;
00095   bool m_userRangeY = false;
00096
00097   // Axis label.
00098   std::string m_labelY = "";
00099
00100   bool m_legend = false;
00101
00102   // Colours.
00103   short m_colTotal = kBlue + 3;
00104   short m_colElectrons = kOrange - 3;
00105   short m_colIons = kRed + 1;
00106   std::array<short, 6> m_colDelayed{
00107       {kCyan + 2, kYellow - 7, kRed - 9, kGreen + 1, kYellow - 4, kRed - 9}};
00108
00109   std::array<short, 3> m_colPrompt{{kAzure + 10, kRed - 4, kMagenta + 2}};
00110
00111   TH1* DrawHistogram(TH1D& h, const std::string& opt,
00112                      const std::string& ylabel);
00113 };
00114 }  // namespace Garfield
00115 #endif
```

## 8.177 /builds/garfield/docs/source/README.md File Reference