FiPy

Programmer's Reference

Daniel Wheeler
Jonathan E. Guyer
James A. Warren

Metallurgy Division
and the Center for Theoretical and Computational Materials Science
Materials Science and Engineering Laboratory

November 3, 2004

Version 0.1

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

This software was developed at the National Institute of Standards and Technology by employees of the Federal Government in the course of their official duties. Pursuant to title 17 section 105 of the United States Code this software is not subject to copyright protection and is in the public domain. FiPy is an experimental system. NIST assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. We would appreciate acknowledgement if the software is used.

This software can be redistributed and/or modified freely provided that any derivative works bear some notice that they are derived from it, and any modified versions bear some notice that they have been modified.

Contents

1 Module fipy.boundaryConditions **1**
- 1.1 Module fipy.boundaryConditions.boundaryCondition . . . 1
- 1.2 Module fipy.boundaryConditions.fixedFlux . . . 3
- 1.3 Module fipy.boundaryConditions.fixedValue . . . 4
- 1.4 Module fipy.boundaryConditions.nthOrderBoundaryCondition . . . 5

2 Module fipy.equations **7**
- 2.1 Module fipy.equations.advectionEquation . . . 7
- 2.2 Module fipy.equations.diffusionEquation . . . 8
- 2.3 Module fipy.equations.diffusionEquationWithSource . . . 9
- 2.4 Module fipy.equations.equation . . . 10
- 2.5 Module fipy.equations.explicitDiffusionEquation . . . 11
- 2.6 Module fipy.equations.matrixEquation . . . 12
- 2.7 Module fipy.equations.nthOrderDiffusionEquation . . . 13
- 2.8 Module fipy.equations.postRelaxationEquation . . . 14
- 2.9 Module fipy.equations.preRelaxationEquation . . . 15
- 2.10 Module fipy.equations.relaxationEquation . . . 16
- 2.11 Module fipy.equations.sourceEquation . . . 17
- 2.12 Module fipy.equations.stdyConvDiffEquation . . . 18
- 2.13 Module fipy.equations.stdyConvDiffScEquation . . . 19

3 Module fipy.iterators **21**
- 3.1 Module fipy.iterators.adaptiveIterator . . . 21
- 3.2 Module fipy.iterators.iterator . . . 23

4 Module fipy.meshes **25**
- 4.1 Module fipy.meshes.common.mesh . . . 25
- 4.2 Module fipy.meshes.numMesh.adaptiveMesh . . . 29
- 4.3 Module fipy.meshes.numMesh.cell . . . 32
- 4.4 Module fipy.meshes.numMesh.face . . . 33
- 4.5 Module fipy.meshes.numMesh.gmshExport . . . 34
- 4.6 Module fipy.meshes.numMesh.gmshImport . . . 35
- 4.7 Module fipy.meshes.numMesh.grid2D . . . 42
- 4.8 Module fipy.meshes.numMesh.grid3D . . . 44

4.9 Module fipy.meshes.numMesh.mesh . . . 47
4.10 Module fipy.meshes.numMesh.mesh2D . . . 55
4.11 Module fipy.meshes.numMesh.refinedMesh . . . 57
4.12 Module fipy.meshes.numMesh.skewedGrid2D . . . 60
4.13 Module fipy.meshes.numMesh.test . . . 62
4.14 Module fipy.meshes.numMesh.testGrid . . . 63
4.15 Module fipy.meshes.numMesh.testGrid3D . . . 66
4.16 Module fipy.meshes.numMesh.testMesh . . . 69
4.17 Module fipy.meshes.numMesh.testMesh3D . . . 71
4.18 Module fipy.meshes.numMesh.testMeshBase . . . 73
4.19 Module fipy.meshes.numMesh.testTri2D . . . 75
4.20 Module fipy.meshes.numMesh.tri2D . . . 77
4.21 Module fipy.meshes.pyMesh.cell . . . 80
4.22 Module fipy.meshes.pyMesh.face . . . 82
4.23 Module fipy.meshes.pyMesh.face2D . . . 85
4.24 Module fipy.meshes.pyMesh.grid2D . . . 86
4.25 Module fipy.meshes.pyMesh.mesh . . . 90
4.26 Module fipy.meshes.pyMesh.test . . . 93
4.27 Module fipy.meshes.pyMesh.vertex . . . 94

5 Module fipy.models **95**
5.1 Module fipy.models.cahnHilliard.cahnHilliardEquation . . . 95
5.2 Module fipy.models.elphf.componentVariable . . . 97
5.3 Module fipy.models.elphf.concentrationEquation . . . 99
5.4 Module fipy.models.elphf.elphf . . . 100
5.5 Module fipy.models.elphf.elphfIterator . . . 101
5.6 Module fipy.models.elphf.interstitialEquation . . . 102
5.7 Module fipy.models.elphf.phaseEquation . . . 103
5.8 Module fipy.models.elphf.phaseVariable . . . 104
5.9 Module fipy.models.elphf.poissonEquation . . . 105
5.10 Module fipy.models.elphf.scaledCellVariable . . . 106
5.11 Module fipy.models.elphf.semiImplicitPoissonEquation . . . 107
5.12 Module fipy.models.elphf.solventVariable . . . 108
5.13 Module fipy.models.elphf.substitutionalEquation . . . 109
5.14 Module fipy.models.elphf.substitutionalVariable . . . 111
5.15 Module fipy.models.elphf.test . . . 113
5.16 Module fipy.models.levelSet.advection.advectionEquation . . . 122
5.17 Module fipy.models.levelSet.advection.advectionTerm . . . 123
5.18 Module fipy.models.levelSet.advection.higherOrderAdvectionEquation . . . 125
5.19 Module fipy.models.levelSet.advection.higherOrderAdvectionTerm . . . 126
5.20 Module fipy.models.levelSet.distanceFunction.distanceEquation . . . 129
5.21 Module fipy.models.levelSet.distanceFunction.distanceVariable . . . 131
5.22 Module fipy.models.levelSet.distanceFunction.extensionEquation . . . 136
5.23 Module fipy.models.levelSet.distanceFunction.levelSetDiffusionEquation . . . 138
5.24 Module fipy.models.levelSet.distanceFunction.levelSetDiffusionVariable . . . 139
5.25 Module fipy.models.levelSet.electroChem.depositionRateVariable . . . 141

5.26 Module fipy.models.levelSet.electroChem.metalIonDiffusionEquation 144
5.27 Module fipy.models.levelSet.electroChem.metalIonSourceVariable 146
5.28 Module fipy.models.levelSet.electroChem.test 148
5.29 Module fipy.models.levelSet.surfactant.adsorbingSurfactantEquation 149
5.30 Module fipy.models.levelSet.surfactant.convectionCoeff 153
5.31 Module fipy.models.levelSet.surfactant.surfactantBulkDiffusionEquation 156
5.32 Module fipy.models.levelSet.surfactant.surfactantEquation 159
5.33 Module fipy.models.levelSet.surfactant.surfactantVariable 160
5.34 Module fipy.models.levelSet.test 163
5.35 Module fipy.models.phase.phase.addOverFacesVariable 164
5.36 Module fipy.models.phase.phase.anisotropyVariable 165
5.37 Module fipy.models.phase.phase.mPhiVariable 168
5.38 Module fipy.models.phase.phase.phaseDiffusionVariable 169
5.39 Module fipy.models.phase.phase.phaseEquation 170
5.40 Module fipy.models.phase.phase.phaseHalfAngleVariable 171
5.41 Module fipy.models.phase.phase.scSourceVariable 172
5.42 Module fipy.models.phase.phase.spSourceVariable 173
5.43 Module fipy.models.phase.phase.type1MPhiVariable 174
5.44 Module fipy.models.phase.phase.type2MPhiVariable 175
5.45 Module fipy.models.phase.temperature.temperatureEquation 176
5.46 Module fipy.models.phase.test 177
5.47 Module fipy.models.phase.theta.diffusionVariable 178
5.48 Module fipy.models.phase.theta.modCellGradVariable 179
5.49 Module fipy.models.phase.theta.modCellToFaceVariable 180
5.50 Module fipy.models.phase.theta.modFaceGradVariable 181
5.51 Module fipy.models.phase.theta.modPhysicalField 182
5.52 Module fipy.models.phase.theta.modularVariable 183
5.53 Module fipy.models.phase.theta.noModularVariable 184
5.54 Module fipy.models.phase.theta.sourceVariable 185
5.55 Module fipy.models.phase.theta.thetaEquation 186
5.56 Module fipy.models.phase.theta.thetaHalfAngleVariable 187
5.57 Module fipy.models.phase.theta.transientVariable 188
5.58 Module fipy.models.test 189

6 Module fipy.solvers 191
6.1 Module fipy.solvers.linearCGSSolver 191
6.2 Module fipy.solvers.linearGMRESSolver 192
6.3 Module fipy.solvers.linearLUSolver 193
6.4 Module fipy.solvers.linearPCGSolver 194
6.5 Module fipy.solvers.linearScipyLUSolver 195
6.6 Module fipy.solvers.linearScipySolver 196
6.7 Module fipy.solvers.solver 197

7 Module fipy.terms 199
7.1 Module fipy.terms.cellTerm 199
7.2 Module fipy.terms.centralDiffConvectionTerm 200

7.3 Module fipy.terms.convectionTerm . . . 201
7.4 Module fipy.terms.diffusionTerm . . . 202
7.5 Module fipy.terms.explicitDiffusionTerm . . . 203
7.6 Module fipy.terms.explicitUpwindConvectionTerm . . . 204
7.7 Module fipy.terms.exponentialConvectionTerm . . . 205
7.8 Module fipy.terms.faceTerm . . . 206
7.9 Module fipy.terms.hybridConvectionTerm . . . 207
7.10 Module fipy.terms.implicitDiffusionTerm . . . 208
7.11 Module fipy.terms.nthOrderDiffusionTerm . . . 209
7.12 Module fipy.terms.powerLawConvectionTerm . . . 212
7.13 Module fipy.terms.scSourceTerm . . . 213
7.14 Module fipy.terms.sourceTerm . . . 214
7.15 Module fipy.terms.spSourceTerm . . . 215
7.16 Module fipy.terms.term . . . 216
7.17 Module fipy.terms.test . . . 217
7.18 Module fipy.terms.transientTerm . . . 218
7.19 Module fipy.terms.upwindConvectionTerm . . . 219
7.20 Module fipy.terms.vanLeerConvectionTerm . . . 220

8 Module fipy.test 221
8.1 Module fipy.test . . . 221

9 Module fipy.tests 223
9.1 Module fipy.tests.blinky . . . 223
9.2 Module fipy.tests.doctestPlus . . . 224
9.3 Module fipy.tests.testBase . . . 225
9.4 Module fipy.tests.testProgram . . . 226

10 Module fipy.tools 229
10.1 Module fipy.tools.array . . . 229
10.2 Module fipy.tools.dimensions.DictWithDefault . . . 231
10.3 Module fipy.tools.dimensions.NumberDict . . . 232
10.4 Module fipy.tools.dimensions.physicalField . . . 234
10.5 Module fipy.tools.dump . . . 254
10.6 Module fipy.tools.inline.inline . . . 255
10.7 Module fipy.tools.memoryLeak . . . 256
10.8 Module fipy.tools.sparseMatrix . . . 257
10.9 Module fipy.tools.test . . . 261
10.10 Module fipy.tools.vector . . . 263

11 Module fipy.variables 265
11.1 Module fipy.variables.addOverFacesVariable . . . 265
11.2 Module fipy.variables.arithmeticCellToFaceVariable . . . 267
11.3 Module fipy.variables.cellGradVariable . . . 268
11.4 Module fipy.variables.cellToFaceVariable . . . 269
11.5 Module fipy.variables.cellVariable . . . 270
11.6 Module fipy.variables.cellVolumeAverageVariable . . . 274

11.7 Module fipy.variables.faceDifferenceVariable . . . 275
11.8 Module fipy.variables.faceGradContributionsVariable . . . 276
11.9 Module fipy.variables.faceGradVariable . . . 277
11.10 Module fipy.variables.faceVariable . . . 278
11.11 Module fipy.variables.harmonicCellToFaceVariable . . . 279
11.12 Module fipy.variables.magVariable . . . 280
11.13 Module fipy.variables.sumVariable . . . 281
11.14 Module fipy.variables.test . . . 282
11.15 Module fipy.variables.testInterpolation . . . 283
11.16 Module fipy.variables.testLaplacian . . . 292
11.17 Module fipy.variables.testPickle . . . 296
11.18 Module fipy.variables.transposeVariable . . . 297
11.19 Module fipy.variables.variable . . . 298
11.20 Module fipy.variables.vectorArithmeticCellToFaceVariable . . . 306
11.21 Module fipy.variables.vectorCellToFaceVariable . . . 307
11.22 Module fipy.variables.vectorCellVariable . . . 308
11.23 Module fipy.variables.vectorFaceVariable . . . 309

12 Module fipy.viewers 311
12.1 Module fipy.viewers.colorbar . . . 311
12.2 Module fipy.viewers.gist1DViewer . . . 312
12.3 Module fipy.viewers.gistVectorViewer . . . 313
12.4 Module fipy.viewers.gistViewer . . . 314
12.5 Module fipy.viewers.gnuplotViewer . . . 315
12.6 Module fipy.viewers.grid2DGistViewer . . . 316
12.7 Module fipy.viewers.numpyGistViewer . . . 319
12.8 Module fipy.viewers.numpyPyxViewer . . . 320
12.9 Module fipy.viewers.pyxviewer . . . 321
12.10 Module fipy.viewers.test . . . 326
12.11 Module fipy.viewers.varGistViewer . . . 327
12.12 Module fipy.viewers.viewer . . . 328

Index 329

Chapter 1

Module fipy.boundaryConditions

1.1 Module fipy.boundaryConditions.boundaryCondition

Generic boundary condition base class

1.1.1 Class BoundaryCondition

Known Subclasses: FixedFlux, FixedValue, NthOrderBoundaryCondition

Methods

__init__(*self*, *faces*, *value*)

Generic boundary condition base class.

Parameters

`faces:` `list` or `tuple` of `Face` objects to which this condition applies
`value:` the value to impose

getContribution(*self*, *cell1dia*, *cell1off*)

Return the effect of this boundary condition on the equation solution matrices.
`getContribution()` is called by each `Term` of each `Equation`.
A `tuple` of (`LL`, `bb`, `ids`) is calculated, to be added to the equation's (**L**, **b**) matrices at the cells specified by `ids`.

Parameters

`cell1dia:` contribution to adjacent cell diagonal by this exterior face
`cell1off:` contribution to **b**-vector by this exterior face

getDerivative(*self*, *order*)

getFaces(*self*)

Return the faces this boundary condition applies to.

1.2 Module fipy.boundaryConditions.fixedFlux

Fixed flux (Neumann) boundary condition

1.2.1 Class FixedFlux

fipy.boundaryConditions.boundaryCondition.BoundaryCondition
└
FixedFlux

Methods

__init__(*self*, *faces*, *value*)

Generic boundary condition base class.

Parameters

`faces: list` or `tuple` of `Face` objects to which this condition applies
`value:` the value to impose

Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.__init__
extit(inherited documentation)

getContribution(*self*, *cell1dia*, *cell1off*)

Leave **L** unchanged and add gradient to **b**

Parameters

`cell1dia:` *unused*
`cell1off:` *unused*

Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.getContribution

getDerivative(*self*, *order*)
Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.getDerivative

Inherited from BoundaryCondition: getFaces

1.3 Module fipy.boundaryConditions.fixedValue

Fixed value (Dirichlet) boundary condition

1.3.1 Class FixedValue

fipy.boundaryConditions.boundaryCondition.BoundaryCondition
└
FixedValue

Methods

getContribution(*self*, *cell1dia*, *cell1off*)

Set boundary equal to value.
A `tuple` of (`LL`, `bb`, `ids`) is calculated, to be added to the equation's (**L**, **b**) matrices at the cells specified by `ids`.

Parameters

`cell1dia`: contribution to adjacent cell diagonal by this exterior face
`cell1off`: contribution to **b**-vector by this exterior face

Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.getContribution

Inherited from BoundaryCondition: __init__, getDerivative, getFaces

1.4 Module fipy.boundaryConditions.nthOrderBoundaryCondition

Boundary condition to describe high-order derivatives.

This `BoundaryCondition` never has any direct effect on the solution matrices, but its derivatives do.

1.4.1 Class NthOrderBoundaryCondition

fipy.boundaryConditions.boundaryCondition.BoundaryCondition ┬
NthOrderBoundaryCondition

Methods

__init__(*self*, *faces*, *value*, *order*)
Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.__init__

getContribution(*self*, *cell1dia*, *cell1off*)

Leave `L` and `b` unchanged

Parameters

`cell1dia:` *unused*
`cell1off:` *unused*

Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.getContribution

getDerivative(*self*, *order*)
Overrides: fipy.boundaryConditions.boundaryCondition.BoundaryCondition.getDerivative

Inherited from BoundaryCondition: getFaces

Chapter 2

Module fipy.equations

2.1 Module fipy.equations.advectionEquation

2.1.1 Class AdvectionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

AdvectionEquation

Methods

__init__(*self*, *var*, *transientCoeff*=`1.0`, *convectionCoeff*=`1.0`, *solver*=`'default_solver'`, *convectionScheme*=`<class fipy.terms.upwindConvectionTerm.UpwindConvectionTe...`, *boundaryConditions*=`()`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.2 Module fipy.equations.diffusionEquation

2.2.1 Class DiffusionEquation

fipy.equations.equation.Equation ┐

fipy.equations.matrixEquation.MatrixEquation ┐

DiffusionEquation

Known Subclasses: DiffusionEquationWithSource, LevelSetDiffusionEquation

Diffusion equation is implicit.

Methods

__init__(*self*, *var*, *transientCoeff*=`1.0`, *diffusionCoeff*=`1.0`, *solver*=`'default_solver'`, *boundaryConditions*=`()`, *otherTerms*=`()`) Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.3 Module fipy.equations.diffusionEquationWithSource

2.3.1 Class DiffusionEquationWithSource

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.diffusionEquation.DiffusionEquation

DiffusionEquationWithSource

Equation consiting of a transient term, diffusion terms and source terms.

Methods

__init__(*self*, *var*, *transientCoeff*=`1.0`, *diffusionCoeff*=`0.0`, *scCoeff*=`0.0`, *spCoeff*=`0.0`, *solver*=`<fipy.solvers.linearPCGSolver.LinearPCGSolver instance at...`, *boundaryConditions*=`()`, *otherTerms*=`()`)
Overrides: fipy.equations.diffusionEquation.DiffusionEquation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.4 Module fipy.equations.equation

2.4.1 Class Equation

Known Subclasses: DistanceEquation, MatrixEquation

Methods

__init__(*self*, *var*, *terms*, *solver*, *solutionTolerance*=`0.0001`)

getFigureOfMerit(*self*)

getResidual(*self*)

getSolutionTolerance(*self*)

getVar(*self*)

isConverged(*self*)

solve(*self*, *dt*=`1.0`)

updateVar(*self*)

2.5 Module fipy.equations.explicitDiffusionEquation

2.5.1 Class ExplicitDiffusionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

ExplicitDiffusionEquation

Diffusion equation is explicit.

Methods

__**init**__(*self*, *var*, *transientCoeff*=`1.0`, *diffusionCoeff*=`1.0`, *solver*=`'default_solver'`, *boundaryConditions*=`()`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.6 Module fipy.equations.matrixEquation

2.6.1 Class MatrixEquation

fipy.equations.equation.Equation

MatrixEquation

Known Subclasses: AdvectionEquation, AdvectionEquation, CahnHilliardEquation, DiffusionEquation, ExplicitDiffusionEquation, NthOrderDiffusionEquation, PhaseEquation, RelaxationEquation, SourceEquation, SteadyConvectionDiffusionEquation, SteadyConvectionDiffusionScEquation, SurfactantEquation, TemperatureEquation, ThetaEquation

Methods

buildMatrix(*self*, *dt*)

getResidual(*self*)
Overrides: fipy.equations.equation.Equation.getResidual

getResidual2(*self*)

getVar(*self*)
Overrides: fipy.equations.equation.Equation.getVar

postSolve(*self*, *array*)

solve(*self*, *dt*=`1.0`)
Overrides: fipy.equations.equation.Equation.solve

Inherited from Equation: __init__, getFigureOfMerit, getSolutionTolerance, isConverged, updateVar

2.7 Module fipy.equations.nthOrderDiffusionEquation

2.7.1 Class NthOrderDiffusionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

NthOrderDiffusionEquation

Diffusion equation is implicit.

Methods

__**init**__(*self*, *var*, *transientCoeff*=`1.0`, *diffusionCoeff*=`(1.0,)`, *solver*=`'default_solver'`, *boundaryConditions*=`()`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.8 Module fipy.equations.postRelaxationEquation

2.8.1 Class PostRelaxationEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.relaxationEquation.RelaxationEquation

PostRelaxationEquation

Methods

postSolve(*self*, *residual*)
Overrides: fipy.equations.relaxationEquation.RelaxationEquation.postSolve

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: __init__, getRelaxation

2.9 Module fipy.equations.preRelaxationEquation

2.9.1 Class PreRelaxationEquation

fipy.equations.equation.Equation ┐

fipy.equations.matrixEquation.MatrixEquation ┐

fipy.equations.relaxationEquation.RelaxationEquation ┐

PreRelaxationEquation

Methods

buildMatrix(*self*)
Overrides: fipy.equations.matrixEquation.MatrixEquation.buildMatrix

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: __init__, getRelaxation, postSolve

2.10 Module fipy.equations.relaxationEquation

2.10.1 Class RelaxationEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

RelaxationEquation

Known Subclasses: ConcentrationEquation, PhaseEquation, PoissonEquation, PostRelaxationEquation, PreRelaxationEquation

Methods

__init__(*self*, *var*, *terms*, *solver*, *solutionTolerance*=`0.0001`, *relaxation*=`1.0`)
Overrides: fipy.equations.equation.Equation.__init__

getRelaxation(*self*)

postSolve(*self*, *residual*)
Overrides: fipy.equations.matrixEquation.MatrixEquation.postSolve

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve

2.11 Module fipy.equations.sourceEquation

2.11.1 Class SourceEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

SourceEquation

Equation consiting of a transient term, and source terms.

Methods

__**init**__(*self*, *var*, *transientCoeff*=`1.0`, *scCoeff*=`0.0`, *spCoeff*=`0.0`, *solver*=<`fipy.solvers.linearPCGSolver.LinearPCGSolver instance at...`, *boundaryConditions*=`()`, *otherTerms*=`()`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.12 Module fipy.equations.stdyConvDiffEquation

2.12.1 Class SteadyConvectionDiffusionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

SteadyConvectionDiffusionEquation

Diffusion equation is implicit.

Methods

__init__(*self*, *var*, *diffusionCoeff*=`1.0`, *convectionCoeff*=`1.0`, *solver*=`'default_solver'`, *convectionScheme*=`<class fipy.terms.powerLawConvectionTerm.PowerLawConvecti...`, *boundaryConditions*=`()`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

2.13 Module fipy.equations.stdyConvDiffScEquation

2.13.1 Class SteadyConvectionDiffusionScEquation

fipy.equations.equation.Equation ┐

fipy.equations.matrixEquation.MatrixEquation ┐

SteadyConvectionDiffusionScEquation

Diffusion equation is implicit.

Methods

__**init**__(*self*, *var*, *diffusionCoeff*=`1.0`, *convectionCoeff*=`1.0`, *sourceCoeff*=`0.0`, *solver*=`'default_solver'`, *convectionScheme*=`<class fipy.terms.powerLawConvectionTerm.PowerLawConvecti...`, *boundaryConditions*=`()`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

Chapter 3

Module fipy.iterators

3.1 Module fipy.iterators.adaptiveIterator

3.1.1 Class AdaptiveIterator

fipy.iterators.iterator.Iterator
└
AdaptiveIterator

Adaptive time-stepping equation iterator

Methods

__init__(*self*, *equations*, *timeStepDuration*=`None`, *viewers*=`()`)

:Parameters:
- ‘equations‘: ‘list‘ or ‘tuple‘ of ‘Equation‘ objects to iterate over

Overrides: fipy.iterators.iterator.Iterator.__init__

adjustTimeStep(*self*, *dt*)

resetTimeStep(*self*)

timestep(*self*, *maxSweeps*=`1`, *dt*=`1.0`)

Iterate the solution.

Parameters

`maxSweeps`: maximum number of sweeps to reach convergence

Overrides: fipy.iterators.iterator.Iterator.timestep extit(inherited documentation)

Inherited from Iterator: advanceTimeStep, elapseTime, sweep, sweeps

3.2 Module fipy.iterators.iterator

Generic equation iterator

3.2.1 Class ConvergenceError

exceptions.Exception
  exceptions.StandardError
    exceptions.ArithmeticError
      ConvergenceError

Methods

__init__(*self*, *equations*)
Overrides: exceptions.Exception.__init__

__str__(*self*)
Overrides: exceptions.Exception.__str__

Inherited from Exception: __getitem__

3.2.2 Class Iterator

Known Subclasses: AdaptiveIterator, ElPhFIterator

Generic equation iterator

Methods

__init__(*self*, *equations*, *timeStepDuration*=`None`)

Parameters

`equations:`	list or tuple of equations to iterate over
`timeStepDuration:`	duration of each timestep (`Variable`)

advanceTimeStep(*self*)

elapseTime(*self*, *desiredTime*, *maxSweepsPerStep*=`1`)

sweep(*self*, *dt*)

sweeps(*self*, *maxSweeps*=`1`, *dt*=`1.0`)

timestep(*self*, *maxSweeps*=`1`, *dt*=`1.0`)

Iterate the solution.

Parameters

`maxSweeps:` maximum number of sweeps to reach convergence

Chapter 4

Module fipy.meshes

4.1 Module fipy.meshes.common.mesh

Generic mesh class defining implementation-agnostic behavior.

Make changes to mesh here first, then implement specific implementations in ‘pyMesh‘ and ‘numMesh‘.

Meshes contain cells, faces, and vertices.

4.1.1 Class Mesh

Known Subclasses: Mesh, Mesh

Methods

__init__(*self*)

calcAdjacentCellIDs(*self*)

calcAreaProjections(*self*)

calcCellAreas(*self*)

calcCellCenters(*self*)

calcCellDistances(*self*)

calcCellToCellDistances(*self*)

calcCellToCellIDs(*self*)

calcCellToCellIDsFilled(*self*)

calcCellToFaceOrientations(*self*)

calcCellVolumes(*self*)

calcExteriorCellIDs(*self*)

calcFaceAreas(*self*)

calcFaceAspectRatios(*self*)

calcFaceNormals(*self*)

calcFaceTangents(*self*)

calcFaceToCellDistanceRatio(*self*)

calcFaceToCellDistances(*self*)

calcGeometry(*self*)

calcHigherOrderScalings(*self*)

calcInteriorAndExteriorCellIDs(*self*)

calcInteriorAndExteriorFaceIDs(*self*)

calcInteriorCellIDs(*self*)

calcOrientedAreaProjections(*self*)

calcOrientedFaceNormals(*self*)

calcScaledGeometry(*self*)

calcTopology(*self*)

getAdjacentCellIDs(*self*)

getAreaProjections(*self*)

getCellAreaProjections(*self*)

getCellAreas(*self*)

getCellCenters(*self*)

getCellDistances(*self*)

getCellFaceIDs(*self*)

getCellFaceOrientations(*self*)

getCellNormals(*self*)

getCells(*self*, *filter*=`None`, *ids*=`None`, ***args*)

Return ‘Cell‘ objects of ‘Mesh‘.

getCellsByID(*self*, *ids*=`None`)

getCellToCellDistances(*self*)

getCellToCellIDs(*self*)

getCellToCellIDsFilled(*self*)

getCellVolumes(*self*)

getDim(*self*)

getExteriorCellIDs(*self*)

getExteriorFaceIDs(*self*)

getExteriorFaces(*self*)

getFaceAreas(*self*)

getFaceAspectRatios(*self*)

getFaceNormals(*self*)

getFaces(*self*)

getFacesWithFilter(*self*, *filter*, ***args*)

Return ‘Face‘ objects of ‘Mesh‘.

getFaceTangents1(*self*)

getFaceTangents2(*self*)

getFaceToCellDistanceRatio(*self*)

getFaceToCellDistances(*self*)

getInteriorCellIDs(*self*)

getInteriorFaceIDs(*self*)

getInteriorFaces(*self*)

getMaxFacesPerCell(*self*)

getNearestCell(*self*, *point*)

getNearestCellID(*self*, *point*)

getNumberOfCells(*self*)

getNumberOfFaces(*self*)

getOrientedAreaProjections(*self*)

getOrientedFaceNormals(*self*)

getPointToCellDistances(*self*, *point*)

setScale(*self*, *value*=`1.0`)

4.2 Module fipy.meshes.numMesh.adaptiveMesh

The AdaptiveMesh classes allow meshes to be created "on-the-fly" using a variable specified by the user. The AdaptiveMesh classes use Gmsh, a free open-source meshing utility (http://www.geuz.org/gmsh). To do this, pass in a variable to the initializer function that has a mesh with the same boundaries (though not necessarily the same interior geometry) as the mesh you want to create and values that are equal to the approximate mesh sizes you want at any given point. For example, if you want the mesh to be finer in areas where a concentration gradient is steeper, create a variable whose value is equal to some constant times the reciprocal of the concentration gradient and pass it to the initializer. The following will create a mesh that is finer toward the upper right hand corner:

```
>>> baseMesh = Tri2D(nx = 2, ny = 2, dx = 1.0, dy = 1.0)
>>> var = CellVariable(mesh = baseMesh,
...     value = 0.05 - (0.01 * Numeric.add.reduce(baseMesh.getCellCenters(), axis = 1)),
....    name = "characteristic lengths")
```

Since the value of `var` is smaller in the upper right hand corner, the mesh will be finer there. To create the mesh, do this:

```
>>> newMesh = AdaptiveMesh2D(var)
```

Note

At present, AdaptiveMesh supports triangular meshes only.

4.2.1 Functions

bracedList(*list*)

orderVertices(*vertexCoords*, *vertices*)

parenList(*list*)

removeDuplicates(*list*)

4.2.2 Class AdaptiveMesh2D

fipy.meshes.common.mesh.Mesh
fipy.meshes.numMesh.mesh.Mesh
fipy.meshes.numMesh.mesh2D.Mesh2D
AdaptiveMesh2D

Methods

__init__(*self*, *variable*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

createBackgroundMesh(*self*)

finalInit(*self*)

getExteriorLines(*self*)

getExteriorVertexIDs(*self*)

getGeometryPoints(*self*)

writeBackgroundMesh(*self*)

writeGeometryFile(*self*)

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __getstate__, __mul__, __setstate__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNor-

mals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

4.3 Module fipy.meshes.numMesh.cell

4.3.1 Class Cell

Methods

__init__(*self*, *mesh*, *id*)

__cmp__(*self*, *cell*)

getCellToCellDistances(*self*)

getCellToCellIDs(*self*)

getCenter(*self*)

getID(*self*)

getMesh(*self*)

getNormal(*self*, *index*)

4.4 Module fipy.meshes.numMesh.face

4.4.1 Class Face

`Face` within a‘Mesh‘

`Face` objects are bounded by `Vertex` objects. `Face` objects separate `Cell` objects.

Methods

__init__(*self*, *mesh*, *id*)

`Face` is initialized by `Mesh`

Parameters

`mesh:` the `Mesh` that contains this `Face`
`id:` a unique identifier

getArea(*self*)

getCellID(*self*, *index*=`0`)

Return the `id` of the specified `Cell` on one side of this `Face`.

getCenter(*self*)

Return the coordinates of the `Face` center.

getID(*self*)

4.5 Module fipy.meshes.numMesh.gmshExport

This module takes a FiPy mesh and creates a mesh file that can be opened in Gmsh.

4.5.1 Functions

exportAsMesh(*mesh*, *filename*)

getElementType(*vertices*, *dimensions*)

orderVertices(*vertexCoords*, *vertices*)

4.5.2 Variables

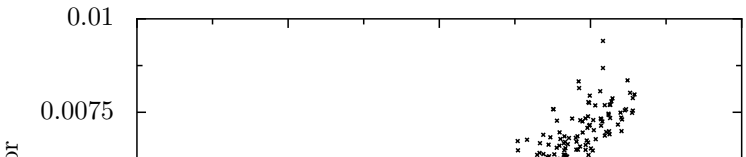
Name	Description
MeshExportError	**Value:** `'MeshExportError'` *(type=`str`)*

4.6 Module fipy.meshes.numMesh.gmshImport

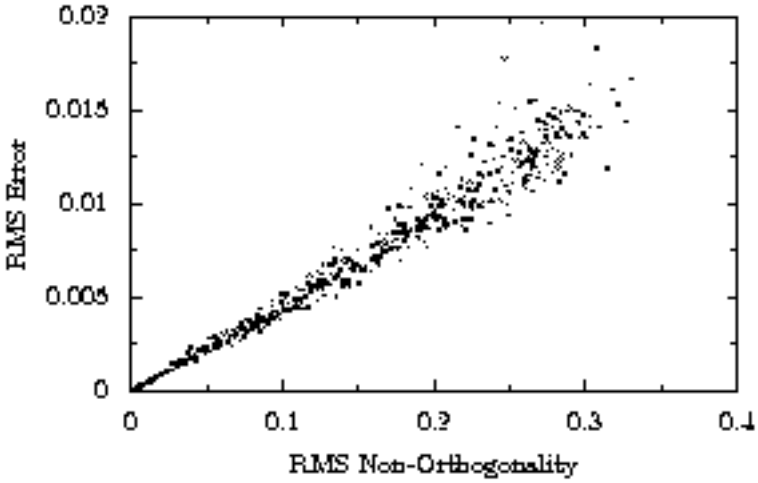
This module takes a Gmsh output file (`.msh`) and converts it into a FiPy mesh. This currently supports triangular and tetrahedral meshes only.

Gmsh generates unstructured meshes, which may contain a significant amount of non-orthogonality and it is very difficult to directly control the amount of non-orthogonality simply by manipulating Gmsh parameters. Therefore, it is necessary to take into account the possibility of errors arising due to the non-orthogonality of the mesh. To test the degree of error, an experiment was conducted using a simple 1D diffusion problem with constant diffusion coefficients and boundary conditions as follows: fixed value of 0 on the left side, fixed value of 1 on the right side, and a fixed flux of 0 on the top and bottom sides. The analytical solution is clearly a uniform gradient going from left to right. this problem was implemented using a Cartesian Grid2D mesh with each interior vertex displaced a short distance in a random direction to create non-orthogonality. Then the root-mean-square error was plotted against the root-mean-square non-orthogonality. The error in each cell was calculated by simply subtracting the analytical solution at each cell center from the calculated value for that cell. The non-orthogonality in each cell is the average, weighted by face area, of the sines of the angles between the face normals and the line segments joining the cells. Thus, the non-orthogonality of a cell can range from 0 (every face is orthogonal to its corresponding cell-to-cell line segment) to 1 (only possible in a degenerate case). This test was run using 500 separate 20x20 meshes and 500 separate 10x10 meshes, each with the interior vertices moved different amounts so as to created different levels of non-orthogonality. The results are shown below.

Results for 20x20 mesh:


0.01
0.0075

Results for 10x10 mesh:


0.02
0.015
0.01
0.005
0
RMS Error
0
0.1
0.2
0.3
0.4
RMS Non-Orthogonality

It is clear from the graphs that finer meshes decrease the error due to non-orthogonality, and that even with a reasonably coarse mesh the error is quite low. However, note that this test is only for a simple 1D diffusion problem with a constant diffusion coefficient, and it is unknown whether the results will be significantly different with more complicated problems.

Test cases:

```
>>> newmesh = GmshImporter3D('fipy/meshes/numMesh/testgmsh.msh')
>>> print newmesh.getVertexCoords().tolist()
[[0.0, 0.0, 0.0], [0.5, 0.5, 1.0], [1.0, 0.0, 0.0], [0.5, 1.0, 0.0], [0.5, 0.5, 0.5]]

>>> print newmesh.faceVertexIDs.tolist()
[[2, 1, 0], [4, 1, 0], [4, 2, 0], [4, 2, 1], [3, 1, 0], [4, 3, 0], [4, 3, 1], [3, 2, 0], [4, 3, 2]

>>> print newmesh.cellFaceIDs.tolist()
[[0, 1, 2, 3], [4, 1, 5, 6], [7, 2, 5, 8], [9, 3, 6, 8]]

>>> twomesh = GmshImporter2D('fipy/meshes/numMesh/GmshTest2D.msh')
>>> print twomesh.getVertexCoords().tolist()
[[0.0, 0.0], [1.0, 0.0], [0.5, 0.5], [0.0, 1.0], [1.0, 1.0], [0.5, 1.5], [0.0, 2.0], [1.0, 2.0]]

>>> print twomesh.faceVertexIDs.tolist()
[[2, 0], [0, 1], [1, 2], [0, 3], [3, 2], [1, 4], [4, 2], [4, 3], [3, 5], [5, 4], [3, 6], [6, 5], [5

>>> print twomesh.cellFaceIDs.tolist()
[[0, 1, 2], [0, 3, 4], [2, 5, 6], [7, 4, 6], [7, 8, 9], [8, 10, 11], [12, 13, 9], [14, 11, 12]]
```

4.6.1 Functions

listToString(*list*)

stringToList(*string*)

4.6.2 Variables

Name	Description
MeshImportError	**Value:** `'MeshImportError'` *(type=`str`)*

4.6.3 Class DataGetter

Methods

computeBaseFaceVertexIDs(*self*)

computeCellFaceIDs(*self*)

computeCellVertexIDs(*self*)

computeFaceVertexIDs(*self*)

computeVertexCoords(*self*)

getData(*self*, *fileName*, *dimensions*)

4.6.4 Class GmshImporter2D

fipy.meshes.common.mesh.Mesh

fipy.meshes.numMesh.mesh.Mesh

fipy.meshes.numMesh.mesh2D.Mesh2D

GmshImporter2D

Methods

__init__(*self*, *filename*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __getstate__, __mul__, __setstate__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

4.6.5 Class GmshImporter3D

fipy.meshes.common.mesh.Mesh

fipy.meshes.numMesh.mesh.Mesh

GmshImporter3D

Methods

__init__(*self*, *filename*) Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcHigherOrderScalings, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __getstate__, __mul__, __setstate__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceAreas, calcFaceCellIDs, calcFaceCenters, calcFaceNormals, calcFaceTangents, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, dilate, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords, meshAdd, translate

4.7 Module fipy.meshes.numMesh.grid2D

2D rectangular Mesh

4.7.1 Class Grid2D

fipy.meshes.common.mesh.Mesh
fipy.meshes.numMesh.mesh.Mesh
fipy.meshes.numMesh.mesh2D.Mesh2D
Grid2D

Known Subclasses: TestMesh

Creates a 2D grid mesh with horizontal faces numbered first and then vertical faces. Vertices and cells are numbered in the usual way.

Methods

__init__(*self*, *dx*=`1.0`, *dy*=`1.0`, *nx*=`None`, *ny*=`1`)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

__getstate__(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__getstate__

__setstate__(*self*, *dict*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__setstate__

createCells(*self*)

cells = (f1, f2, f3, f4) going anticlock wise. f1 etx refer to the faces

createFaces(*self*)

v1, v2 refer to the cells. Horizontel faces are first

createVertices(*self*)

getFacesBottom(*self*)

Return list of faces on bottom boundary of Grid2D.

getFacesLeft(*self*)

Return list of faces on left boundary of Grid2D.

getFacesRight(*self*)

Return list of faces on right boundary of Grid2D.

getFacesTop(*self*)

Return list of faces on top boundary of Grid2D.

getMeshSpacing(*self*)

getPhysicalShape(*self*)

Return physical dimensions of Grid2D.

getScale(*self*)

getShape(*self*)

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __mul__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

4.8 Module fipy.meshes.numMesh.grid3D

3D rectangular-prism Mesh

X axis runs from left to right. Y axis runs from bottom to top. Z axis runs from front to back.

Numbering System:

Vertices: Numbered in the usual way. X coordinate changes most quickly, then Y, then Z.

Cells: Same numbering system as vertices.

Faces: XY faces numbered first, then XZ faces, then YZ faces. Within each subcategory, it is numbered in the usual way.

4.8.1 Class Grid3D

fipy.meshes.common.mesh.Mesh

fipy.meshes.numMesh.mesh.Mesh

Grid3D

Creates a 3D grid mesh.

Methods

__init__(*self*, *dx*=`1.0`, *dy*=`1.0`, *dz*=`1.0`, *nx*=`None`, *ny*=`1`, *nz*=`1`)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

__getstate__(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__getstate__

__setstate__(*self*, *dict*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__setstate__

calcFaceAreas(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.calcFaceAreas

calcFaceNormals(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.calcFaceNormals

calcFaceTangents(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.calcFaceTangents

calcHigherOrderScalings(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcHigherOrderScalings

createCells(*self*)

cells = (front face, back face, left face, right face, bottom face, top face) front and back faces are YZ faces left and right faces are XZ faces top and bottom faces are XY faces

createFaces(*self*)

XY faces are first, then XZ faces, then YZ faces

createVertices(*self*)

getFacesBack(*self*)

Return list of faces on back boundary of Grid3D.

getFacesBottom(*self*)

Return list of faces on bottom boundary of Grid3D.

getFacesFront(*self*)

Return list of faces on front boundary of Grid3D.

getFacesLeft(*self*)

Return list of faces on left boundary of Grid3D.

getFacesRight(*self*)

Return list of faces on right boundary of Grid3D.

getFacesTop(*self*)

Return list of faces on top boundary of Grid3D.

getMeshSpacing(*self*)

getPhysicalShape(*self*)

Return physical dimensions of Grid2D.

getScale(*self*)

getShape(*self*)

repeatWithOffset(*self*, *array*, *offset*, *reps*)

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale

Inherited from Mesh: __add__, __mul__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, dilate, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords, meshAdd, translate

4.9 Module fipy.meshes.numMesh.mesh

Generic mesh class using Numeric to do the calculations

Meshes contain cells, faces, and vertices.

This is built for a non-mixed element mesh.

Test cases:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> from fipy.meshes.numMesh.grid3D import Grid3D
>>> from fipy.meshes.numMesh.tri2D import Tri2D
>>> basemesh = Grid2D(dx = 1.0, dy = 1.0, nx = 2, ny = 2)
>>> dilatedMesh = basemesh * (3, 2)
>>> print dilatedMesh.getVertexCoords().tolist()
[[0.0, 0.0], [3.0, 0.0], [6.0, 0.0], [0.0, 2.0], [3.0, 2.0], [6.0, 2.0], [0.0, 4.0], [3.0, 4.0], [6.0,

>>> translatedMesh = basemesh + (5, 10)
>>> print translatedMesh.getVertexCoords().tolist()
[[5.0, 10.0], [6.0, 10.0], [7.0, 10.0], [5.0, 11.0], [6.0, 11.0], [7.0, 11.0], [5.0, 12.0], [6.0, 12.0]

>>> addedMesh = basemesh + (basemesh + (2, 0))
>>> print addedMesh.getVertexCoords().tolist()
[[0.0, 0.0], [1.0, 0.0], [2.0, 0.0], [0.0, 1.0], [1.0, 1.0], [2.0, 1.0], [0.0, 2.0], [1.0, 2.0], [2.0,

>>> print addedMesh.getCellFaceIDs()
[[ 0, 7, 2, 6,]
 [ 1, 8, 3, 7,]
 [ 2,10, 4, 9,]
 [ 3,11, 5,10,]
 [12,18,14, 8,]
 [13,19,15,18,]
 [14,20,16,11,]
 [15,21,17,20,]]

>>> print addedMesh.faceVertexIDs
[[ 1, 0,]
 [ 2, 1,]
 [ 3, 4,]
 [ 4, 5,]
 [ 6, 7,]
 [ 7, 8,]
 [ 0, 3,]
 [ 4, 1,]
 [ 5, 2,]
 [ 3, 6,]
 [ 7, 4,]
 [ 8, 5,]
 [ 9, 2,]
```

```
 [10, 9,]
 [ 5,11,]
 [11,12,]
 [ 8,13,]
 [13,14,]
 [11, 9,]
 [12,10,]
 [13,11,]
 [14,12,]]
```

```
>>> addedMesh = basemesh + (basemesh + (3, 0))
Traceback (most recent call last):
...
MeshAdditionError: Vertices are not aligned
```

```
>>> addedMesh = basemesh + (basemesh + (2, 2))
Traceback (most recent call last):
...
MeshAdditionError: Faces are not aligned
```

```
>>> triMesh = Tri2D(dx = 1.0, dy = 1.0, nx = 2, ny = 1)
>>> triMesh = triMesh + (2, 0)
>>> triAddedMesh = basemesh + triMesh
>>> print triAddedMesh.getVertexCoords().tolist()
[[0.0, 0.0], [1.0, 0.0], [2.0, 0.0], [0.0, 1.0], [1.0, 1.0], [2.0, 1.0], [0.0, 2.0], [1.0, 2.0],
```

```
>>> print triAddedMesh.getCellFaceIDs()
[[0 ,7 ,2 ,6 ,]
 [1 ,8 ,3 ,7 ,]
 [2 ,10 ,4 ,9 ,]
 [3 ,11 ,5 ,10 ,]
 [16 ,20 ,24 ,-- ,]
 [17 ,21 ,25 ,-- ,]
 [14 ,22 ,24 ,-- ,]
 [15 ,23 ,25 ,-- ,]
 [8 ,18 ,22 ,-- ,]
 [16 ,19 ,23 ,-- ,]
 [12 ,18 ,20 ,-- ,]
 [13 ,19 ,21 ,-- ,]]
```

```
>>> print triAddedMesh.faceVertexIDs
[[ 1, 0,]
 [ 2, 1,]
 [ 3, 4,]
 [ 4, 5,]
 [ 6, 7,]
 [ 7, 8,]
 [ 0, 3,]
 [ 4, 1,]
```

```
 [ 5, 2,]
 [ 3, 6,]
 [ 7, 4,]
 [ 8, 5,]
 [ 9, 2,]
 [10, 9,]
 [ 5,11,]
 [11,12,]
 [11, 9,]
 [12,10,]
 [13, 2,]
 [14, 9,]
 [ 9,13,]
 [10,14,]
 [13, 5,]
 [14,11,]
 [13,11,]
 [14,12,]]

>>> ThreeDBaseMesh = Grid3D(dx = 1.0, dy = 1.0, dz = 1.0, nx = 2, ny = 2, nz = 2)
>>> ThreeDSecondMesh = Grid3D(dx = 1.0, dy = 1.0, dz = 1.0, nx = 1, ny = 1, nz = 1)
>>> ThreeDAddedMesh = ThreeDBaseMesh + (ThreeDSecondMesh + (2, 0, 0))
>>> print ThreeDAddedMesh.getVertexCoords().tolist()
[[0.0, 0.0, 0.0], [1.0, 0.0, 0.0], [2.0, 0.0, 0.0], [0.0, 1.0, 0.0], [1.0, 1.0, 0.0], [2.0, 1.0, 0.0],

>>> print ThreeDAddedMesh.getCellFaceIDs()
[[24,25,12,14, 0, 4,]
 [25,26,13,15, 1, 5,]
 [27,28,14,16, 2, 6,]
 [28,29,15,17, 3, 7,]
 [30,31,18,20, 4, 8,]
 [31,32,19,21, 5, 9,]
 [33,34,20,22, 6,10,]
 [34,35,21,23, 7,11,]
 [26,40,38,39,36,37,]]

>>> print ThreeDAddedMesh.faceVertexIDs
[[ 0, 1, 4, 3,]
 [ 1, 2, 5, 4,]
 [ 3, 4, 7, 6,]
 [ 4, 5, 8, 7,]
 [ 9,10,13,12,]
 [10,11,14,13,]
 [12,13,16,15,]
 [13,14,17,16,]
 [18,19,22,21,]
 [19,20,23,22,]
 [21,22,25,24,]
```

```
 [22,23,26,25,]
 [ 0, 1,10, 9,]
 [ 1, 2,11,10,]
 [ 3, 4,13,12,]
 [ 4, 5,14,13,]
 [ 6, 7,16,15,]
 [ 7, 8,17,16,]
 [ 9,10,19,18,]
 [10,11,20,19,]
 [12,13,22,21,]
 [13,14,23,22,]
 [15,16,25,24,]
 [16,17,26,25,]
 [ 0, 3,12, 9,]
 [ 1, 4,13,10,]
 [ 2, 5,14,11,]
 [ 3, 6,15,12,]
 [ 4, 7,16,13,]
 [ 5, 8,17,14,]
 [ 9,12,21,18,]
 [10,13,22,19,]
 [11,14,23,20,]
 [12,15,24,21,]
 [13,16,25,22,]
 [14,17,26,23,]
 [ 2,27,28, 5,]
 [11,29,30,14,]
 [ 2,27,29,11,]
 [ 5,28,30,14,]
 [27,28,30,29,]]


>>> InvalidMesh = ThreeDBaseMesh + basemesh
Traceback (most recent call last):
...
MeshAdditionError: Dimensions do not match
```

4.9.1 Functions

MAtake(*array*, *indices*, *fill*=0, *axis*=0)

4.9.2 Class Mesh

fipy.meshes.common.mesh.Mesh ┐
Mesh

Known Subclasses: GmshImporter3D, Grid3D, Mesh2D

Methods

__init__(*self*, *vertexCoords*, *faceVertexIDs*, *cellFaceIDs*)

faceVertexIds and cellFacesIds must be padded with minus ones.

Overrides: fipy.meshes.common.mesh.Mesh.__init__

__add__(*self*, *other*, *smallNumber*=`1.0000000000000001e-15`)

__getstate__(*self*)

__mul__(*self*, *other*)

__setstate__(*self*, *dict*)

calcAdjacentCellIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcAdjacentCellIDs

calcAreaProjections(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcAreaProjections

calcCellCenters(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellCenters

calcCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellDistances

calcCellNormals(*self*)

calcCellToCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellToCellDistances

calcCellToCellIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellToCellIDs

calcCellToFaceOrientations(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellToFaceOrientations

calcCellVolumes(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellVolumes

calcFaceAreas(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceAreas

calcFaceCellIDs(*self*)

calcFaceCenters(*self*)

calcFaceNormals(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceNormals

calcFaceTangents(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceTangents

calcFaceToCellDistanceRatio(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceToCellDistanceRatio

calcFaceToCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceToCellDistances

calcGeometry(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcGeometry

calcInteriorAndExteriorCellIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcInteriorAndExteriorCellIDs

calcInteriorAndExteriorFaceIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcInteriorAndExteriorFaceIDs

calcOrientedAreaProjections(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcOrientedAreaProjections

calcOrientedFaceNormals(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcOrientedFaceNormals

calcTopology(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcTopology

dilate(*self*, *factor*)

equalExceptOrder(*self*, *first*, *second*)

Determines if two lists contain the same set of elements, although they may be in different orders. Does not work if one list contains duplicates of an element.

getAddedMeshValues(*self*, *other*, *smallNumber*)

Returns a tuple with 3 elements: the new mesh vertexCoords, faceVertexIDs, and cellFaceIDs, in that order.

getCellsByID(*self*, *ids*=`None`)
Overrides: fipy.meshes.common.mesh.Mesh.getCellsByID

getExteriorFaces(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getExteriorFaces

getFaceCellIDs(*self*)

getFaceCenters(*self*)

getFaces(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getFaces

getInteriorFaces(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getInteriorFaces

getMaxFacesPerCell(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getMaxFacesPerCell

getVertexCoords(*self*)

meshAdd(*self*, *other*, *smallNumber*)

translate(*self*, *vector*)

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcHigherOrderScalings, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCel-

lIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale

4.10 Module fipy.meshes.numMesh.mesh2D

Generic mesh class using Numeric to do the calculations

Meshes contain cells, faces, and vertices.

This is built for a non-mixed element mesh.

4.10.1 Functions

orderVertices(*vertexCoords*, *vertices*)

4.10.2 Class Mesh2D

fipy.meshes.common.mesh.Mesh

fipy.meshes.numMesh.mesh.Mesh

Mesh2D

Known Subclasses: AdaptiveMesh2D, GmshImporter2D, Grid2D, RefinedMesh2D, SkewedGrid2D, Tri2D

Methods

calcFaceAreas(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.calcFaceAreas

calcFaceNormals(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.calcFaceNormals

calcFaceTangents(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.calcFaceTangents

calcHigherOrderScalings(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcHigherOrderScalings

dilate(*self*, *factor*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.dilate

getNonOrthogonality(*self*)

getOrderedCellVertexIDs(*self*)

meshAdd(*self*, *other*, *smallNumber*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.meshAdd

translate(*self*, *vector*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.translate

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __init__, __add__, __getstate__, __mul__, __setstate__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords

4.11 Module fipy.meshes.numMesh.refinedMesh

The `RefinedMesh` class provides an alternative way to adapt a mesh without using the `AdaptiveMesh` class. The `RefinedMesh` contructor takes as input an old mesh (baseMesh) as well as a list of cells to refine (nodeList). The cells are refined by putting an additional vertex in the center of the cell and dividng up the cell by drawing lines from the center to each vertex. After creating the new mesh, you can convert variables to use the new mesh by creating a `RefinedMeshCellVariable` (defined in this module) with the old variable and the new `RefinedMesh` as arguments. Note that if the mesh of the variable passed to `RefinedMeshCellVariable` is not the same as the old mesh used to generate the `RefinedMesh`, the results will be erroneous.

> **Note**
>
> This currently only works with triangular meshes.

Test case:

```
>>> baseMesh = Tri2D(dx = 6., dy = 6., nx = 1, ny = 1)
>>> baseVar = CellVariable(value = [0., 1., 2., 3.], mesh = baseMesh)
>>> refinedMesh = RefinedMesh2D(baseMesh, [1, 3])
>>> refinedVar = RefinedMeshCellVariable(baseVar, refinedMesh)
>>> print refinedMesh.getVertexCoords()
[[ 0., 0.,]
 [ 6., 0.,]
 [ 0., 6.,]
 [ 6., 6.,]
 [ 3., 3.,]
 [ 3., 5.,]
 [ 3., 1.,]]
>>> print refinedMesh.faceVertexIDs
[[1,0,]
 [2,3,]
 [0,2,]
 [3,1,]
 [4,0,]
 [1,4,]
 [4,2,]
 [4,3,]
 [5,4,]
 [5,3,]
 [5,2,]
 [6,1,]
 [6,4,]
 [6,0,]]
>>> print refinedMesh.cellFaceIDs
[[ 3, 5, 7,]
 [ 2, 4, 6,]
```

```
 [ 6,10, 8,]
 [ 7, 8, 9,]
 [ 1, 9,10,]
 [ 0,13,11,]
 [ 5,11,12,]
 [ 4,12,13,]]
>>> print Numeric.array(refinedVar)
[ 0., 2., 1., 1., 1., 3., 3., 3.,]
```

4.11.1 Functions

listToString(*list*)

4.11.2 Class RefinedMesh2D

fipy.meshes.common.mesh.Mesh
  fipy.meshes.numMesh.mesh.Mesh
    fipy.meshes.numMesh.mesh2D.Mesh2D
      RefinedMesh2D

Methods

__**init**__(*self*, *baseMesh*, *nodeList*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __getstate__, __mul__, __setstate__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, get-

FaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

4.11.3 Class RefinedMeshCellVariable

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

RefinedMeshCellVariable

Methods

__**init**__(*self*, *oldVar*, *newMesh*) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

4.12 Module fipy.meshes.numMesh.skewedGrid2D

2D rectangular Mesh with each interior vertex moved in a random distance and direction. The X and Y distances to move each vertex are determined independently, and are random values between -R and R, where R is the "rand" keyword argument to the constructor.

4.12.1 Class SkewedGrid2D

fipy.meshes.common.mesh.Mesh
fipy.meshes.numMesh.mesh.Mesh
fipy.meshes.numMesh.mesh2D.Mesh2D
SkewedGrid2D

Creates a 2D grid mesh with horizontal faces numbered first and then vertical faces. Vertices and cells are numbered in the usual way. The points are skewed by a random amount (up to the rand parameter) in the X and Y directions.

Methods

__init__(*self*, *dx*=`1.0`, *dy*=`1.0`, *nx*=`None`, *ny*=`1`, *rand*=`0`)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

__getstate__(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__getstate__

__setstate__(*self*, *dict*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__setstate__

createCells(*self*)

cells = (f1, f2, f3, f4) going anticlock wise. f1 etx refer to the faces

createFaces(*self*)

v1, v2 refer to the cells. Horizontel faces are first

createVertices(*self*)

getFacesBottom(*self*)

Return list of faces on bottom boundary of Grid2D.

getFacesLeft(*self*)

Return list of faces on left boundary of Grid2D.

getFacesRight(*self*)

Return list of faces on right boundary of Grid2D.

getFacesTop(*self*)

Return list of faces on top boundary of Grid2D.

getMeshSpacing(*self*)

getPhysicalShape(*self*)

Return physical dimensions of Grid2D.

getScale(*self*)

getShape(*self*)

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __mul__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

4.13 Module fipy.meshes.numMesh.test

Test numeric implementation of the mesh

4.13.1 Functions

suite()

4.14 Module fipy.meshes.numMesh.testGrid

Test numeric implementation of the mesh

4.14.1 Functions

suite()

4.14.2 Class TestGrid

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.meshes.numMesh.testMeshBase.TestMeshBase
TestGrid

Known Subclasses: TestGridPickle

Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

testCells(*self*)

testFaces(*self*)

testFacesBottom(*self*)

testFacesLeft(*self*)

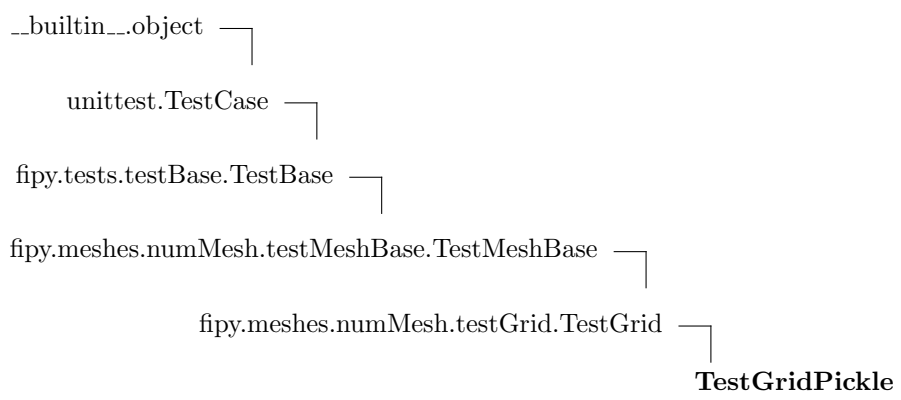
testFacesRight(*self*)

testFacesTop(*self*)

testVertices(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.14.3 Class TestGridPickle



Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: fipy.meshes.numMesh.testGrid.TestGrid.setUp extit(inherited documentation)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestGrid: testCells, testFaces, testFacesBottom, testFacesLeft, testFacesRight, testFacesTop, testVertices
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters,

testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2

Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues

Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.15 Module fipy.meshes.numMesh.testGrid3D

Test numeric implementation of the mesh

4.15.1 Functions

suite()

4.15.2 Class TestGrid

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.meshes.numMesh.testMeshBase.TestMeshBase
TestGrid

Known Subclasses: TestGridPickle

Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

testCells(*self*)

testFaces(*self*)

testFacesBack(*self*)

testFacesBottom(*self*)

testFacesFront(*self*)

testFacesLeft(*self*)

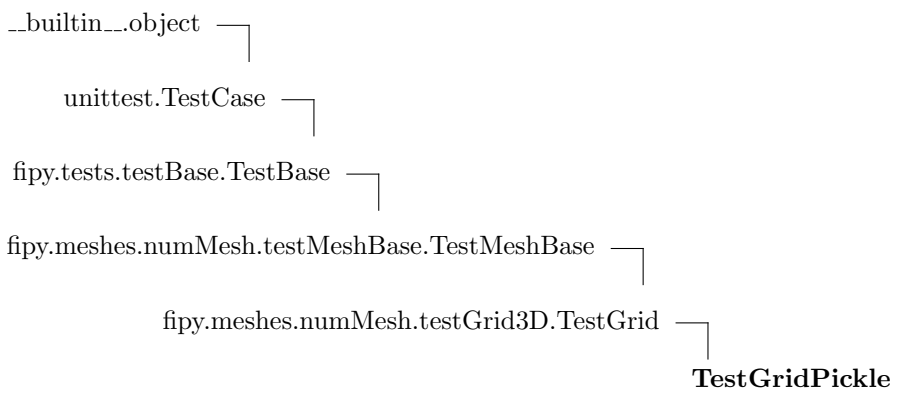
testFacesRight(*self*)

testFacesTop(*self*)

testVertices(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.15.3 Class TestGridPickle



Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: fipy.meshes.numMesh.testGrid3D.TestGrid.setUp extit(inherited documentation)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestGrid: testCells, testFaces, testFacesBack, testFacesBottom, testFacesFront, testFacesLeft, testFacesRight, testFacesTop, testVertices
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.16 Module fipy.meshes.numMesh.testMesh

Test numeric implementation of the mesh

4.16.1 Functions

suite()

4.16.2 Class TestMesh

__builtin__.object
  unittest.TestCase
    fipy.tests.testBase.TestBase
      fipy.meshes.numMesh.testMeshBase.TestMeshBase
        TestMesh

Known Subclasses: TestMeshPickle

Methods

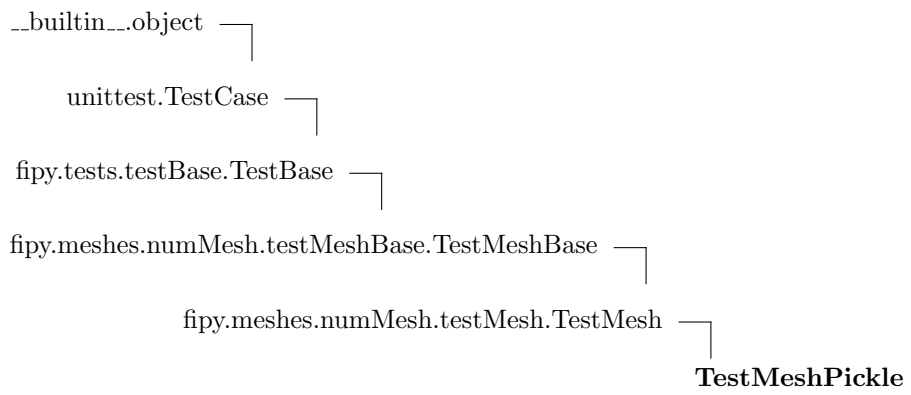
setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.16.3 Class TestMeshPickle



Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: fipy.meshes.numMesh.testMesh.TestMesh.setUp extit(inherited documentation)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.17 Module fipy.meshes.numMesh.testMesh3D

Test numeric implementation of the mesh

4.17.1 Functions

suite()

4.17.2 Class TestMesh3D

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.meshes.numMesh.testMeshBase.TestMeshBase
TestMesh3D

Known Subclasses: TestMesh3DPickle

Methods

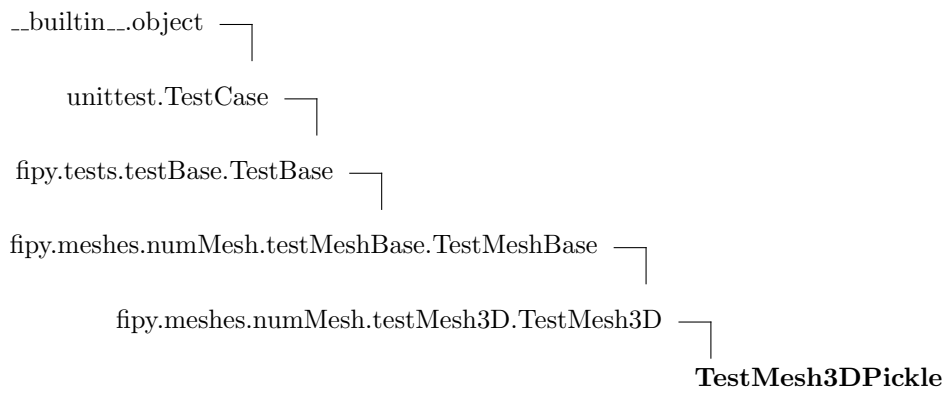
setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.17.3 Class TestMesh3DPickle



Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: fipy.meshes.numMesh.testMesh3D.TestMesh3D.setUp extit(inherited documentation)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellAreaProjections, testCellCenters, testCellDistances, testCellNormals, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.18 Module fipy.meshes.numMesh.testMeshBase

Test numeric implementation of the mesh

4.18.1 Class TestMeshBase

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
TestMeshBase

Known Subclasses: TestGrid, TestGrid, TestMesh, TestMesh3D, TestTri2D

Methods

testAreaProjections(*self*)

testCellAreaProjections(*self*)

testCellCenters(*self*)

testCellDistances(*self*)

testCellNormals(*self*)

testCellToCellDistances(*self*)

testCellToCellIDs(*self*)

testCellToFaceOrientations(*self*)

testCellVolumes(*self*)

testExteriorCellIDs(*self*)

testExteriorFaces(*self*)

testFaceAreas(*self*)

testFaceCellIds(*self*)

testFaceCenters(*self*)

testFaceNormals(*self*)

testFaceToCellDistanceRatios(*self*)

testFaceToCellDistances(*self*)

testInteriorCellIDs(*self*)

testInternalFaces(*self*)

testResult(*self*)
Overrides: fipy.tests.testBase.TestBase.testResult

testTangents1(*self*)

testTangents2(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, setUp, shortDescription, tearDown

4.19 Module fipy.meshes.numMesh.testTri2D

Test numeric implementation of the mesh

4.19.1 Functions

suite()

4.19.2 Class TestTri2D

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.meshes.numMesh.testMeshBase.TestMeshBase
TestTri2D

Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

testCellAreaProjections(*self*)
Overrides: fipy.meshes.numMesh.testMeshBase.TestMeshBase.testCellAreaProjections

testCellNormals(*self*)
Overrides: fipy.meshes.numMesh.testMeshBase.TestMeshBase.testCellNormals

testCells(*self*)

testFaces(*self*)

testFacesBottom(*self*)

testFacesLeft(*self*)

testFacesRight(*self*)

testFacesTop(*self*)

testVertices(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestMeshBase: testAreaProjections, testCellCenters, testCellDistances, testCellToCellDistances, testCellToCellIDs, testCellToFaceOrientations, testCellVolumes, testExteriorCellIDs, testExteriorFaces, testFaceAreas, testFaceCellIds, testFaceCenters, testFaceNormals, testFaceToCellDistanceRatios, testFaceToCellDistances, testInteriorCellIDs, testInternalFaces, testResult, testTangents1, testTangents2
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

4.20 Module fipy.meshes.numMesh.tri2D

2-dimensional triangular mesh.

This class creates a mesh made out of triangles. It does this by starting with a standard Cartesian mesh (Grid2D) and dividing each cell in that mesh (hereafter referred to as a 'box') into four equal partswith the dividing lines being the diagonals. A Tri2D mesh is constructed using the following keyword arguments:

dx, dy - The X and Y dimensions of each box. Note that if dx e dy, the line segments connecting the cell centers will not be orthogonal to the faces.

nx, ny - The number of boxes in the X direction and the Y direction. The total number of boxes will be equal to nx * ny, and the total number of cells will be equal to 4 * nx * ny.

The faces, cells, and vertices are numbered as follows:

Faces - Horizontal faces are numbered first, then vertical faces, then diagonal faces on the lower left of the boxes, then diagonal faces on the llower right of boxes, then diagonal faces on the upper left of boxes, then diagonal faces on the upper right of boxes.

Vertices - Vertices on the corners of boxes are numbered first, then vertices on the box centers.

Cells - Cells on the right of boxes are numbered first, then cells on the top of boxes, then cells on the left of boxes, then cells on the bottom of boxes.

Within each of the 'sub-categories' in the above, the vertices, cells and faces are numbered in the usual way.

Test cases:

```
>>> testmesh = Tri2D(dx = 0.5, dy = 0.5, nx = 2, ny = 2)
>>> list = testmesh.createVertices().tolist()
>>> print list
[[0.0, 0.0], [0.5, 0.0], [1.0, 0.0], [0.0, 0.5], [0.5, 0.5], [1.0, 0.5], [0.0, 1.0], [0.5, 1.0], [

>>> testmesh = Tri2D(dx = 0.5, dy = 0.5, nx = 2, ny = 2)
>>> list = testmesh.createFaces().tolist()
>>> print list
[[1, 0], [2, 1], [3, 4], [4, 5], [6, 7], [7, 8], [0, 3], [4, 1], [5, 2], [3, 6], [7, 4], [8, 5], [

>>> testmesh = Tri2D(dx = 0.5, dy = 0.5, nx = 2, ny = 2)
>>> list = testmesh.createCells().tolist()
>>> print list
[[7, 24, 16], [8, 25, 17], [10, 26, 18], [11, 27, 19], [2, 20, 24], [3, 21, 25], [4, 22, 26], [5,
```

4.20.1 Class Tri2D

fipy.meshes.common.mesh.Mesh
fipy.meshes.numMesh.mesh.Mesh
fipy.meshes.numMesh.mesh2D.Mesh2D
Tri2D

Creates a 2D triangular mesh with horizontal faces numbered first then vertical faces, then diagonal faces. Vertices are numbered starting with the vertices at the corners of boxes and then the vertices at the centers of boxes. Vertices and cells are numbered in the usual way.

Methods

__init__(*self*, *dx*=`1.0`, *dy*=`1.0`, *nx*=`None`, *ny*=`1`)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__init__

__getstate__(*self*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__getstate__

__setstate__(*self*, *dict*)
Overrides: fipy.meshes.numMesh.mesh.Mesh.__setstate__

createCells(*self*)

cells = (f1, f2, f3, f4) going anticlockwise. f1 etx refer to the faces

createFaces(*self*)

v1, v2 refer to the cells. Horizontel faces are first

createVertices(*self*)

getFacesBottom(*self*)

Return list of faces on bottom boundary of Grid2D.

getFacesLeft(*self*)

Return list of faces on left boundary of Grid2D.

getFacesRight(*self*)

Return list of faces on right boundary of Grid2D.

getFacesTop(*self*)

Return list of faces on top boundary of Grid2D.

getMeshSpacing(*self*)

getPhysicalShape(*self*)

Return physical dimensions of Grid2D.

getScale(*self*)

getShape(*self*)

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Mesh: __add__, __mul__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

4.21 Module fipy.meshes.pyMesh.cell

Cell within a mesh

4.21.1 Class Cell

Cell within a mesh

`Cell` objects are bounded by `Face` objects.

Methods

__init__(*self*, *faces*, *faceOrientations*, *id*)

`Cell` is initialized by `Mesh`

Parameters

`faces:`	`list` or `tuple` of bounding faces that define the cell
`faceOrientations:`	`list`, `tuple`, or `Numeric.array` of orientations (+/-1) to indicate whether a face points into this face or out of it. Can be calculated, but the mesh typically knows this information already.
`id:`	unique identifier

__repr__(*self*)

Textual representation of `Cell`.

calcCenter(*self*)

Calculate the coordinates of the `Cell` center.
Cell center is the average of the bounding Face centers.

Attention!

Is this right? Seems to give too much weight to small faces.

calcFaceIDs(*self*)

calcVolume(*self*)

Calculate the volume of the Cell.
Sums the projected volumes of the faces.
Projected volume of a face is a right-prism bounded by its center and the y-z plane, whose cross-section is the projection of the face on the y-z plane.

getBoundingCells(*self*)

getCenter(*self*)

Return the coordinates of the `Cell` center.

getFaceIDs(*self*)

getFaceOrientations(*self*)

getFaces(*self*)

Return the faces bounding the `Cell`.

getID(*self*)

Return the id of this Cell.

getVolume(*self*)

Return the volume of the `Cell`.

setID(*self*, *id*)

Set the `id` of the `Cell`.

4.22 Module fipy.meshes.pyMesh.face

`Face` within a `Mesh`

4.22.1 Class Face

Known Subclasses: Face2D

`Face` within a `Mesh`

`Face` objects are bounded by `Vertex` objects. `Face` objects separate `Cell` objects.

Methods

__init__(*self*, *vertices*, *id*)

`Face` is initialized by `Mesh`

Parameters

`vertices:` the `Vertex` points that bound the `Face`
`id:` a unique identifier

__repr__(*self*)

Textual representation of `Face`.

addBoundingCell(*self*, *cell*, *orientation*)

Add `cell` to the list of `Cell` objects which lie on either side of this Face.

calcArea(*self*)

Calculate the area of the `Face`.
Area is the signed sum of the area of the triangles bounded by each polygon edge and the origin.

calcCellDistance(*self*)

Calculate the distance between adjacent `Cell` centers.
If the Face is on a boundary and has only one bordering Cell, the distance is from the Cell center to the Face center.

calcCenter(*self*)

Calculate the coordinates of the `Face` center.
Cell center is the average of the bounding Vertex centers.

calcFaceToCellDistance(*self*, *id*)

calcNormal(*self*)

Calculate the unit normal vector.
Unit normal vector is calculated from cross-product of two tangent vectors.

> **Warning**
>
> Doesn't work if `t1` and `t2` are colinear!

calcTangent1(*self*)

calcTangent2(*self*)

getArea(*self*)

Return the area of the `Face`.

getCellDistance(*self*)

Return the distance between adjacent `Cell` centers.

getCellID(*self*, *index*=`0`)

Return the `id` of the specified `Cell` on one side of this `Face`.

getCells(*self*)

Return the `Cell` objects which lie on either side of this `Face`.

getCenter(*self*)

Return the coordinates of the `Face` center.

getFaceToCellDistance(*self*, *cellID*=`None`)

getID(*self*)

getNormal(*self*)

Return the unit normal vector

getOrientation(*self*)

removeBoundingCell(*self*, *cell*)

Remove `cell` from the list of bounding `Cell` objects.
Called by the Mesh when a Cell is removed.

setCellDistance(*self*)

Assign the cached distance between adjacent `Cell` centers.

setFaceToCellDistances(*self*)

setID(*self*, *id*)

Set the `id` of the `Face`.

setNormal(*self*)

Cache the unit normal vector.
Called by Cell initializer after bounding Cells have been added to Faces.

4.23 Module fipy.meshes.pyMesh.face2D

1D (edge) Face in a 2D Mesh

4.23.1 Class Face2D

fipy.meshes.pyMesh.face.Face ┐
Face2D

1D (edge) Face in a 2D Mesh

Face2D is bounded by two Vertices.

Methods

calcArea(*self*)

Area is length of vector between vertices.

Overrides: fipy.meshes.pyMesh.face.Face.calcArea

calcNormal(*self*)

Normal is perpendicular to vector between vertices.

Overrides: fipy.meshes.pyMesh.face.Face.calcNormal

calcTangent1(*self*)
Overrides: fipy.meshes.pyMesh.face.Face.calcTangent1

calcTangent2(*self*)
Overrides: fipy.meshes.pyMesh.face.Face.calcTangent2

Inherited from Face: __init__, __repr__, addBoundingCell, calcCellDistance, calcCenter, calcFaceToCellDistance, getArea, getCellDistance, getCellID, getCells, getCenter, getFaceToCellDistance, getID, getNormal, getOrientation, removeBoundingCell, setCellDistance, setFaceToCellDistances, setID, setNormal

4.24 Module fipy.meshes.pyMesh.grid2D

2D rectangular Mesh

4.24.1 Class Grid2D

fipy.meshes.common.mesh.Mesh

fipy.meshes.pyMesh.mesh.Mesh

Grid2D

2D rectangular Mesh

Numbering system

nx=5

ny=3

Cells:

```
************************************
*      *      *       *      *      *
* 10   * 11   * 12    * 13   * 14   *
************************************
*      *      *       *      *      *
* 5    * 6    * 7     * 8    * 9    *
************************************
*      *      *       *      *      *
* 0    * 1    * 2     * 3    * 4    *
************************************
```

Faces (before reordering):

```
***15******16*****17******18****19***
*      *      *       *      *      *
32    33      34      35     36     37
***10******11*****12******13*****14**
*      *      *       *      *      *
26     27     28      29     30     31
***5*******6******7*******8******9***
*      *      *       *      *      *
20     21     22      23     24     25
***0*******1******2*******3******4***
```

Faces (after reordering):

```
***27******28*****29******30****31***
```

```
*      *      *       *      *      *
34     18     19      20     21     37
***5*******6******7*******8******9***
*      *      *       *      *      *
33     14     15      16     17     36
***0*******1******2*******3******4***
*      *      *       *      *      *
32     10     11      12     13     35
***22******23*****24******25*****26**
```

Vertices:

```
18*****19*****20******21*****22****23
*      *      *       *      *      *
*      *      *       *      *      *
12*****13*****14******15*****16****17
*      *      *       *      *      *
*      *      *       *      *      *
6******7******8*******9******10****11
*      *      *       *      *      *
*      *      *       *      *      *
0******1******2*******3******4******5
```

Methods

__init__(*self*, *dx*, *dy*, *nx*, *ny*)

Grid2D is initialized by caller

Parameters

- `dx`: dimension of each cell in **x** direction
- `dy`: dimension of each cell in **y** direction
- `nx`: number of cells in **x** direction
- `ny`: number of cells in **y** direction

Overrides: fipy.meshes.pyMesh.mesh.Mesh.__init__

createCells(*self*, *rowFaces*, *colFaces*)

Return list of `Cell` objects.

createFaces(*self*, *vertices*)

Return 2-`tuple` of `Face` objects bounded by `vertices`.
First `tuple` are the `Face` objects that separate rows of `Cell` objects. Second `tuple` are the `Face` objects that separate columns of `Cell objects. These initial lists are layed out for efficiency of composing and indexing into the lists to compose ‘Cell` objects. They will subsequently be reordered for efficiency of computations.

createInteriorFaces(*self*, *faces*)

Return list of faces that are not on boundary of Grid2D.

createVertices(*self*)

Return list of `Vertex` objects

getCellCenters(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getCellCenters

getCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getCellDistances

getCellVolumes(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getCellVolumes

getFaceAreas(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getFaceAreas

getFacesBottom(*self*)

Return list of faces on bottom boundary of Grid2D.

getFacesLeft(*self*)

Return list of faces on left boundary of Grid2D.

getFacesRight(*self*)

Return list of faces on right boundary of Grid2D.

getFacesTop(*self*)

Return list of faces on top boundary of Grid2D.

getFaceToCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getFaceToCellDistances

getMaxFacesPerCell(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getMaxFacesPerCell

getMeshSpacing(*self*)

getPhysicalShape(*self*)

Return physical dimensions of Grid2D.

Overrides: fipy.meshes.pyMesh.mesh.Mesh.getPhysicalShape

getShape(*self*)

Return cell dimensions `Grid2D`.

makeGridData(*self*, *array*)

Return `array` data mapped onto cell geometry of `Grid2D`.

Overrides: fipy.meshes.pyMesh.mesh.Mesh.makeGridData

reorderFaces(*self*, *rowFaces*, *colFaces*)

Return a `tuple` of `Face` objects ordered for best efficiency.
Composed from `rowFaces` and `colFaces` such that all interior faces are listed contiguously, rows then columns, followed by all boundary faces, rows then columns.

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcFaceAspectRatios, calcGeometry, calcHigherOrderScalings, calcInteriorAndExteriorCellIDs, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances
Inherited from Mesh: calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellFaceIDs, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcExteriorCellIDs, calcFaceAreas, calcFaceNormals, calcFaceOrientations, calcFaceTangents, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, getCellsByID, getExteriorFaces, getFaceOrientations, getFaces, getInteriorFaces, getScale, setScale

4.25 Module fipy.meshes.pyMesh.mesh

Generic mesh class

Meshes contain cells, faces, and vertices.

4.25.1 Class Mesh

fipy.meshes.common.mesh.Mesh ─┐
Mesh

Known Subclasses: Grid2D

Methods

__init__(*self*, *cells*, *faces*, *interiorFaces*, *vertices*)
Overrides: fipy.meshes.common.mesh.Mesh.__init__

calcAdjacentCellIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcAdjacentCellIDs

calcAreaProjections(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcAreaProjections

calcCellCenters(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellCenters

calcCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellDistances

calcCellFaceIDs(*self*)

calcCellToCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellToCellDistances

calcCellToCellIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellToCellIDs

calcCellToFaceOrientations(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellToFaceOrientations

calcCellVolumes(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcCellVolumes

calcExteriorCellIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcExteriorCellIDs

calcFaceAreas(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceAreas

calcFaceNormals(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceNormals

calcFaceOrientations(*self*)

calcFaceTangents(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceTangents

calcFaceToCellDistanceRatio(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceToCellDistanceRatio

calcFaceToCellDistances(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcFaceToCellDistances

calcInteriorAndExteriorFaceIDs(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcInteriorAndExteriorFaceIDs

calcOrientedAreaProjections(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcOrientedAreaProjections

calcOrientedFaceNormals(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcOrientedFaceNormals

calcTopology(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.calcTopology

getCellsByID(*self*, *ids*=`None`)
Overrides: fipy.meshes.common.mesh.Mesh.getCellsByID

getExteriorFaces(*self*)

Return all Faces of Mesh that are on a Mesh boundary

Overrides: fipy.meshes.common.mesh.Mesh.getExteriorFaces

getFaceOrientations(*self*)

getFaces(*self*)
Overrides: fipy.meshes.common.mesh.Mesh.getFaces

getInteriorFaces(*self*)

Return Faces of Mesh that are not on a Mesh boundary.

Overrides: fipy.meshes.common.mesh.Mesh.getInteriorFaces

getPhysicalShape(*self*)

Return physical dimensions of Mesh.

getScale(*self*)

makeGridData(*self*, *array*)

Return array data mapped onto cell geometry of Mesh.

setScale(*self*, *scale*)
Overrides: fipy.meshes.common.mesh.Mesh.setScale

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcFaceAspectRatios, calcGeometry, calcHigherOrderScalings, calcInteriorAndExteriorCellIDs, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, getFaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getMaxFacesPerCell, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances

4.26 Module fipy.meshes.pyMesh.test

Test numeric implementation of the mesh

4.26.1 Functions

suite()

4.27 Module fipy.meshes.pyMesh.vertex

Vertex within a Mesh

Vertices bound Faces.

4.27.1 Class Vertex

Methods

__init__(*self*, *coordinates*)

Vertex is initialized by Mesh with its coordinates.

__repr__(*self*)

Textual representation of Vertex.

getCoordinates(*self*)

Return coordinates of Vertex.

Chapter 5

Module fipy.models

5.1 Module fipy.models.cahnHilliard.cahnHilliardEquation

The Cahn-Hilliard equation is given by:

$$\frac{\partial \phi}{\partial t} = \nabla \cdot D \nabla \left(\frac{\partial f}{\partial \phi} - \epsilon^2 \nabla^2 \phi \right)$$

where the free energy functional is given by,

$$f = \frac{a^2}{2} \phi^2 (1 - \phi)^2$$

In the `CahnHilliardEquation` object the equation is transformed into the following form,

$$\frac{\partial \phi}{\partial t} = \nabla \cdot D \frac{\partial^2 f}{\partial \phi^2} \nabla \phi - \nabla \cdot D \nabla \epsilon^2 \nabla^2 \phi$$

This form of the equation allows the `CahnHilliardEquation` to be constructed from a transient term, a diffusion term, and a fourth order diffusion term. Notice that the diffusion coefficient for the diffusion term does not always remain positive since,

$$\frac{\partial^2 f}{\partial \phi^2} = a^2 (1 - 6\phi(1 - \phi))$$

can be less than zero and thus unstable. The fourth order diffusion term acts to stabilize the problem.

5.1.1 Class CahnHilliardEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

CahnHilliardEquation

Methods

__**init**__(*self*, *var*, *solver*=`'default_solver'`, *transientCoeff*=`1.0`, *boundaryConditions*=`()`, *parameters*={})
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.2 Module fipy.models.elphf.componentVariable

5.2.1 Class ComponentVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.elphf.scaledCellVariable.ScaledCellVariable

ComponentVariable

Known Subclasses: SolventVariable, SubstitutionalVariable

Methods

__init__(*self*, *mesh*, *parameters*, *value*=`0.0`, *hasOld*=`1`)
Overrides: fipy.models.elphf.scaledCellVariable.ScaledCellVariable.__init__

copy(*self*)

Make an duplicate of the Variable

```
>>> a = Variable(value = 3)
>>> b = a.copy()
>>> b
Variable(value = 3)
```

The duplicate will not reflect changes made to the original

```
>>> a.setValue(5)
>>> b
Variable(value = 3)
```

Overrides: fipy.variables.cellVariable.CellVariable.copy extit(inherited documentation)

getBarrierHeight(*self*)

getDiffusivity(*self*)

getStandardPotential(*self*)

getValence(*self*)

Inherited from ScaledCellVariable: getScale, getScaled, setValue
Inherited from CellVariable: __call__, __getstate__, __setstate__, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue,

getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.3 Module fipy.models.elphf.concentrationEquation

5.3.1 Class ConcentrationEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.relaxationEquation.RelaxationEquation

ConcentrationEquation

Known Subclasses: InterstitialEquation, SubstitutionalEquation

Methods

__init__(*self*, *Cj*, *fields*={},
convectionScheme=<`class fipy.terms.powerLawConvectionTerm.PowerLawConvecti...`,
solver=`'default_solver'`, *relaxation*=`0.0`, *phaseRelaxation*=`1.0`, *solutionTolerance*=`1e-10`,
boundaryConditions=`()`)
Overrides: fipy.equations.relaxationEquation.RelaxationEquation.__init__

getConvectionCoeff(*self*, *Cj*, *fields*, *diffusivity*=`None`)

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: getRelaxation, postSolve

5.4 Module fipy.models.elphf.elphf

5.4.1 Functions

addScales(*mesh*, *parameters*)

makeEquations(*mesh*, *fields*, *parameters*, *phaseRelaxation*=`1.0`, *solutionTolerance*=`9.9999999999999995e-07`)

makeFields(*mesh*, *parameters*)

5.4.2 Variables

Name	Description
constant	**Value:** {} *(type=`dict`)*

5.5 Module fipy.models.elphf.elphfIterator

Generic equation iterator

5.5.1 Class ElPhFIterator

fipy.iterators.iterator.Iterator
ElPhFIterator

Methods

__init__(*self*, *equations*, *timeStepDuration*=`None`, *viewers*=`()`)

Arguments:
'equations' – list or tuple of equations to iterate over

Overrides: fipy.iterators.iterator.Iterator.__init__

sweep(*self*)
Overrides: fipy.iterators.iterator.Iterator.sweep

Inherited from Iterator: advanceTimeStep, elapseTime, sweeps, timestep

5.6 Module fipy.models.elphf.interstitialEquation

5.6.1 Class InterstitialEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.relaxationEquation.RelaxationEquation

fipy.models.elphf.concentrationEquation.ConcentrationEquation

InterstitialEquation

Represents the diffusion equation for interstitial species, such as electrons,

$$\underbrace{\frac{\partial C_j}{\partial t}}_{\text{transient}} = \underbrace{D_j \nabla^2 C_j}_{\text{diffusion}} + \underbrace{D_j \nabla \cdot C_j \left\{ \overbrace{\left[p'(\xi)\Delta\mu_j^\circ + g'(\xi)W_j\right]\nabla\xi}^{\text{phase transformation}} + \overbrace{z_j \nabla\phi}^{\text{electromigration}} \right\}}_{\text{convection}}$$

where, for a given species j, C_j is the concentration, D_j is the self diffusivity, $\Delta\mu_j^\circ$ is the standard chemical potential difference between the electrode and electrolyte for a pure material, W_j is the magnitude of the energy barrier in the double-well free energy function, and z_j is the valence.

In addition, t is time, ξ is the phase field variable, ϕ is the electrostatic potential, $p(\xi)$ describes the interpolation of the free energy, $g(\xi)$ describes the shape of the energy barrier between the electrode and electrolyte phases, $p'(\xi) = 30\xi^2\left(1-\xi\right)^2$, and $g'(\xi) = 2\xi\left(1-\xi\right)\left(1-2\xi\right)$.

Methods

getConvectionCoeff(*self*, *Cj*, *fields*, *diffusivity*=`None`)
Overrides: fipy.models.elphf.concentrationEquation.ConcentrationEquation.getConvectionCoeff

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: getRelaxation, postSolve
Inherited from ConcentrationEquation: __init__

5.7 Module fipy.models.elphf.phaseEquation

5.7.1 Class PhaseEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.relaxationEquation.RelaxationEquation

PhaseEquation

Represents the phase field equation

$$\underbrace{\frac{\partial \xi}{\partial t}}_{\text{transient}} = \underbrace{M_\xi \kappa_\xi \nabla^2 \xi}_{\text{diffusion}} - \underbrace{\overbrace{M_\xi \sum_{j=1}^{n} C_j \left[p'(\xi)\Delta\mu_j^\circ + g'(\xi)W_j\right]}^{\text{phase transformation}}}_{\text{source}}$$

where ξ is the phase field variable, t is time, M_ξ is the phase field mobility, κ_ξ is the phase field gradient energy coefficient. $p(\xi)$ describes the interpolation of the free energy, $g(\xi)$ describes the shape of the energy barrier between the electrode and electrolyte phases, $p'(\xi) = 30\xi^2 (1-\xi)^2$, and $g'(\xi) = 2\xi (1-\xi)(1-2\xi)$. For a given species j, C_j is the concentration, $\Delta\mu_j^\circ$ is the standard chemical potential difference between the electrode and electrolyte for a pure material, and W_j is the magnitude of the energy barrier in the double-well free energy function.

Methods

__init__(*self*, *phase*, *fields*={}, *phaseMobility*=`1.0`, *phaseGradientEnergy*=`1.0`, *solver*=`'default_solver'`, *relaxation*=`0.0`, *solutionTolerance*=`1e-10`, *boundaryConditions*=`()`)
Overrides: fipy.equations.relaxationEquation.RelaxationEquation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: getRelaxation, postSolve

5.8 Module fipy.models.elphf.phaseVariable

5.8.1 Class PhaseVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

PhaseVariable

Methods

__init__(*self*, *mesh*, *name*='', *value*=0.0, *hasOld*=1)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

get_g(*self*)

get_gPrime(*self*)

get_p(*self*)

get_pPrime(*self*)

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.9 Module fipy.models.elphf.poissonEquation

5.9.1 Class PoissonEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.relaxationEquation.RelaxationEquation

PoissonEquation

Known Subclasses: SemiImplicitPoissonEquation

Represents the Poisson equation

$$\underbrace{\nabla \cdot (\epsilon \nabla \phi)}_{\text{diffusion}} + \underbrace{\rho}_{\text{source}} = 0$$

where ϕ is the electrostatic potential, ϵ is the dielectric constant $\rho \equiv \sum_{j=1}^{n} z_j C_j$, is the total charge, C_j is the concentration of the jth species, and z_j is the valence of that species.

Methods

__init__(*self*, *potential*, *parameters*, *fields*={}, *solver*=`'default_solver'`, *solutionTolerance*=`1e-10`, *relaxation*=`1.0`, *boundaryConditions*=`()`)
Overrides: fipy.equations.relaxationEquation.RelaxationEquation.__init__

getConcentration(*self*, *component*)

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: getRelaxation, postSolve

5.10 Module fipy.models.elphf.scaledCellVariable

5.10.1 Class ScaledCellVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

ScaledCellVariable

Known Subclasses: ComponentVariable

Methods

__init__(*self*, *mesh*, *name*, *value*=`0.0`, *hasOld*=`1`, *scale*=`1`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

getScale(*self*)

getScaled(*self*)

setValue(*self*, *value*, *cells*=`()`)
Overrides: fipy.variables.cellVariable.CellVariable.setValue

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.11 Module fipy.models.elphf.semiImplicitPoissonEquation

5.11.1 Class SemiImplicitPoissonEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.relaxationEquation.RelaxationEquation

fipy.models.elphf.poissonEquation.PoissonEquation

SemiImplicitPoissonEquation

Methods

getConcentration(*self*, *component*)
Overrides: fipy.models.elphf.poissonEquation.PoissonEquation.getConcentration

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve
Inherited from RelaxationEquation: getRelaxation, postSolve
Inherited from PoissonEquation: __init__

5.12 Module fipy.models.elphf.solventVariable

5.12.1 Class SolventVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.elphf.scaledCellVariable.ScaledCellVariable

fipy.models.elphf.componentVariable.ComponentVariable

SolventVariable

Methods

__**init**__(*self*, *mesh*, *parameters*, *substitutionals*)
Overrides: fipy.models.elphf.componentVariable.ComponentVariable.__init__

Inherited from ComponentVariable: copy, getBarrierHeight, getDiffusivity, getStandardPotential, getValence
Inherited from ScaledCellVariable: getScale, getScaled, setValue
Inherited from CellVariable: __call__, __getstate__, __setstate__, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.13 Module fipy.models.elphf.substitutionalEquation

5.13.1 Class SubstitutionalEquation

fipy.equations.equation.Equation ┐

fipy.equations.matrixEquation.MatrixEquation ┐

fipy.equations.relaxationEquation.RelaxationEquation ┐

fipy.models.elphf.concentrationEquation.ConcentrationEquation ┐

SubstitutionalEquation

Represents the diffusion equation for substitutional species

$$\underbrace{\frac{\partial C_j}{\partial t}}_{\text{transient}} = \underbrace{D_j \nabla^2 C_j}_{\text{diffusion}} + \underbrace{D_j \nabla \cdot \frac{C_j}{1 - \sum_{\substack{k=2 \\ k\neq j}}^{n-1} C_k} \left\{ \overbrace{\sum_{\substack{i=2 \\ i\neq j}}^{n-1} \nabla C_i}^{\text{counter diffusion}} + \overbrace{C_n \left[p'(\xi) \Delta\mu_{jn}^{\circ} + g'(\xi) W_{jn} \right] \nabla \xi}^{\text{phase transformation}} + \overbrace{C_n z_{jn} \nabla \phi}^{\text{electromigration}} \right\}}_{\text{convection}}$$

where, for a given species j, C_j is the concentration, D_j is the self diffusivity, $\Delta\mu_j^{\circ}$ is the standard chemical potential difference between the electrode and electrolyte for a pure material, W_j is the magnitude of the energy barrier in the double-well free energy function, and z_j is the valence.

In addition, t is time, ξ is the phase field variable, ϕ is the electrostatic potential, $p(\xi)$ describes the interpolation of the free energy, $g(\xi)$ describes the shape of the energy barrier between the electrode and electrolyte phases, $p'(\xi) = 30\xi^2 (1-\xi)^2$, and $g'(\xi) = 2\xi (1-\xi)(1-2\xi)$.

The summation $\sum_{\substack{i=2 \\ i\neq j}}^{n-1}$ is over all substitutional species, excluding the species of interest and the designated solvent, and $\Delta\mu_{jn}^{\circ}$, W_{jn}, and z_{jn} are the differences of the respective quantities $\Delta\mu_j^{\circ}$, W_j, and z_j between substitutional species j and the solvent species n.

Methods

getConvectionCoeff(*self*, *Cj*, *fields*, *diffusivity*=`None`)
Overrides: fipy.models.elphf.concentrationEquation.ConcentrationEquation.getConvectionCoeff

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, solve

Inherited from RelaxationEquation: getRelaxation, postSolve
Inherited from ConcentrationEquation: __init__

5.14 Module fipy.models.elphf.substitutionalVariable

5.14.1 Class SubstitutionalVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.elphf.scaledCellVariable.ScaledCellVariable

fipy.models.elphf.componentVariable.ComponentVariable

SubstitutionalVariable

Methods

__init__(*self*, *mesh*, *parameters*, *solventParameters*, *name*=`''`, *value*=`0.0`, *hasOld*=`1`)
Overrides: fipy.models.elphf.componentVariable.ComponentVariable.__init__

copy(*self*)

Make an duplicate of the Variable

```
>>> a = Variable(value = 3)
>>> b = a.copy()
>>> b
Variable(value = 3)
```

The duplicate will not reflect changes made to the original

```
>>> a.setValue(5)
>>> b
Variable(value = 3)
```

Overrides: fipy.models.elphf.componentVariable.ComponentVariable.copy extit(inherited documentation)

Inherited from ComponentVariable: getBarrierHeight, getDiffusivity, getStandardPotential, getValence
Inherited from ScaledCellVariable: getScale, getScaled, setValue
Inherited from CellVariable: __call__, __getstate__, __setstate__, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnary-

OperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.15 Module fipy.models.elphf.test

Test steady-state diffusion solutions

5.15.1 Functions

suite()

5.15.2 Class TestElPhF

__builtin__.object
  unittest.TestCase
    fipy.tests.testBase.TestBase
      TestElPhF

Known Subclasses: TestElPhF1D, TestElPhF1Dphase, TestElPhF1DphaseBinary, TestElPhF1DphaseQuaternary, TestElPhF1DphaseTernaryAndElectrons, TestElPhF1DpoissonAllCharge, TestElPhF1DpoissonLeftCharge, TestElPhF1DpoissonRightCharge, TestElPhF2D, TestElPhF2Dcorner

Simple test case for the phase field equation.

Methods

assertFieldWithinTolerance(*self*, *field*, *final*)

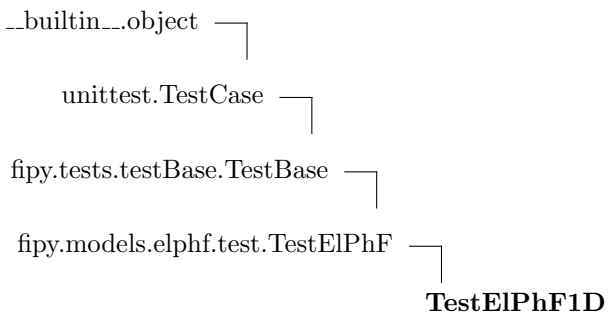
setUp(*self*, *input*)
Overrides: unittest.TestCase.setUp

testResult(*self*)
Overrides: fipy.tests.testBase.TestBase.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, fail-

IfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.3 Class TestElPhF1D

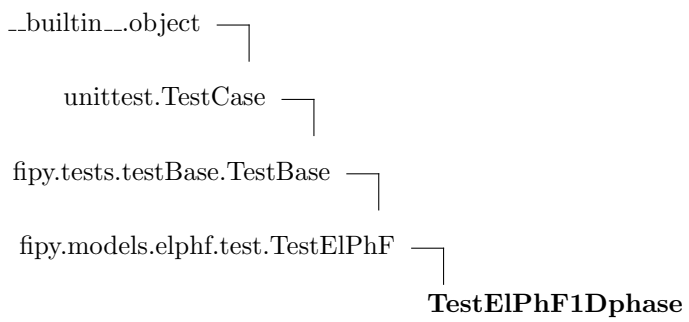


Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance, testResult
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.4 Class TestElPhF1Dphase



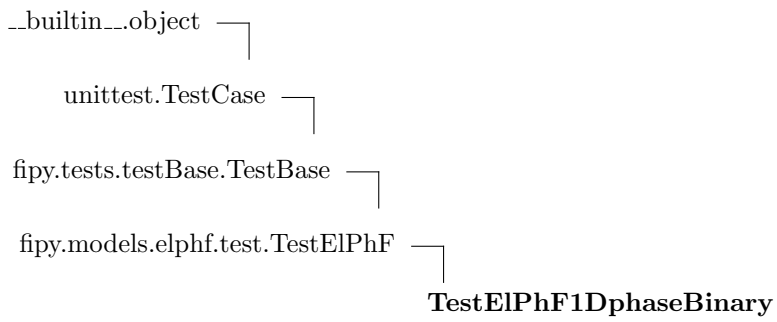
Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

testResult(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.5 Class TestElPhF1DphaseBinary

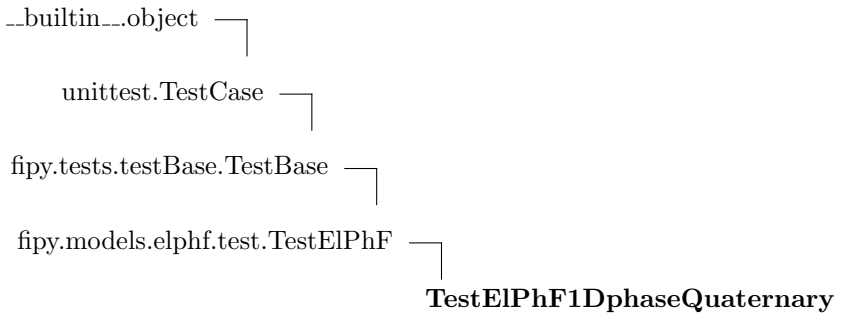


Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance, testResult
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.6 Class TestElPhF1DphaseQuaternary



Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance, testResult
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.7 Class TestElPhF1DphaseTernaryAndElectrons

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.models.elphf.test.TestElPhF
TestElPhF1DphaseTernaryAndElectrons

Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance, testResult
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.8 Class TestElPhF1DpoissonAllCharge

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.models.elphf.test.TestElPhF
TestElPhF1DpoissonAllCharge

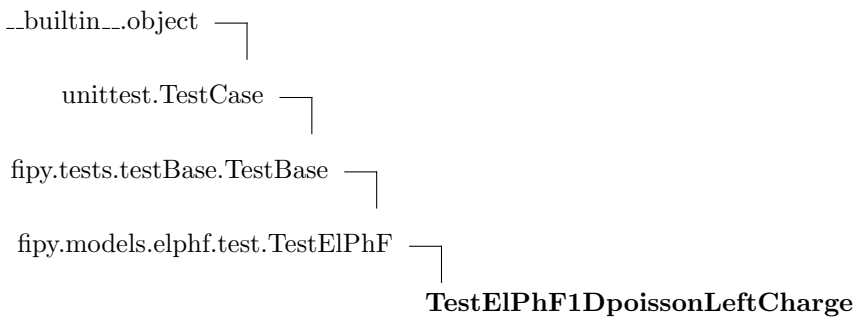
Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

testResult(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.9 Class TestElPhF1DpoissonLeftCharge



Methods

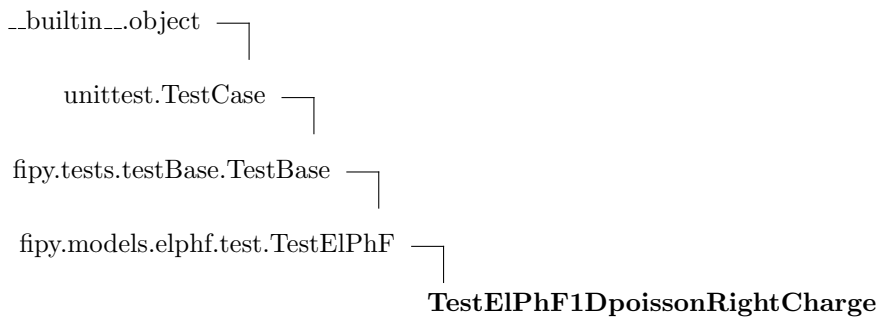
setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

testResult(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance

Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.10 Class TestElPhF1DpoissonRightCharge



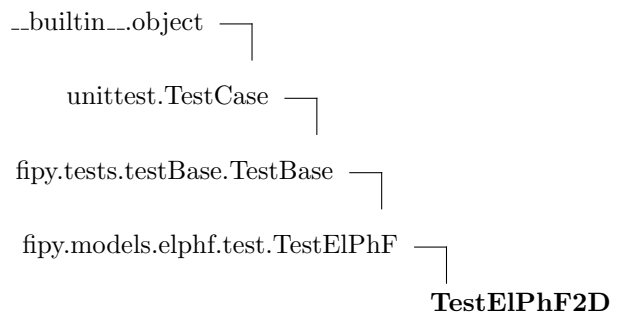
Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

testResult(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.11 Class TestElPhF2D

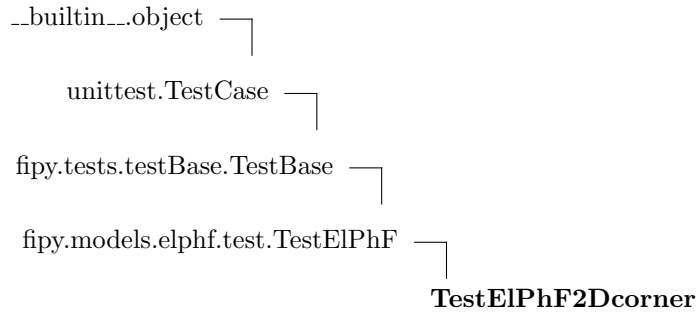


Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance, testResult
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.15.12 Class TestElPhF2Dcorner



Methods

setUp(*self*)
Overrides: fipy.models.elphf.test.TestElPhF.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestElPhF: assertFieldWithinTolerance, testResult
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

5.16 Module fipy.models.levelSet.advection.advectionEquation

The `AdvectionEquation` object constructs the b vector and L matrix necessary for the solution of the advection equation. The advection equation is given by:

$$\frac{\partial \phi}{\partial t} + u|\nabla \phi|$$

This solution method for the `AdvectionTerm` is set up specifically to evolve `var` while preserving `var` as a distance function. This equation is used in conjunction with the `DistanceFunction` object. Further details of the numerical method can be found in "Level Set Methods and Fast Marching Methods" by J.A. Sethian, Cambridge University Press, 1999. Testing for the advection equation is in `examples.levelSet.advection`

5.16.1 Class AdvectionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

AdvectionEquation

Known Subclasses: HigherOrderAdvectionEquation

Methods

__init__(*self*, *var*=`None`, *advectionCoeff*=`None`, *solver*=`None`, *advectionTerm*=<`class fipy.models.levelSet.advection.advectionTerm.Advec...`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.17 Module fipy.models.levelSet.advection.advectionTerm

The `AdvectionTerm` object constructs the b vector contribution for the advection term given by

$$u|\nabla\phi|$$

from the advection equation given by:

$$\frac{\partial\phi}{\partial t} + u|\nabla\phi| = 0$$

The construction of the gradient magnitude term requires upwinding. The formula used here is given by:

$$u_P|\nabla\phi|_P = \max(u_P, 0)\left[\sum_A \min\left(\frac{\phi_A - \phi_P}{d_{AP}}, 0\right)^2\right]^{1/2} + \min(u_P, 0)\left[\sum_A \max\left(\frac{\phi_A - \phi_P}{d_{AP}}, 0\right)^2\right]^{1/2}$$

Here are some simple test cases for this problem:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 3, ny = 1)
```

Trivial test: `>>> L, b = AdvectionTerm(0., mesh = mesh).buildMatrix(Numeric.zeros(3, 'd'))`

```
>>> Numeric.allclose(b, Numeric.zeros(3, 'd'), atol = 1e-10)
1
```

Less trivial test:

```
>>> L, b = AdvectionTerm(1., mesh = mesh).buildMatrix(Numeric.arange(3))
>>> Numeric.allclose(b, Numeric.array((0., -1., -1.)), atol = 1e-10)
1
```

Even less trivial

```
>>> L, b = AdvectionTerm(-1., mesh = mesh).buildMatrix(Numeric.arange(3))
>>> Numeric.allclose(b, Numeric.array((1., 1., 0.)), atol = 1e-10)
1
```

Another trivial test case (more trivial than a trivial test case standing on a harpsichord singing 'trivial test cases are here again')

```
>>> vel = Numeric.array((-1, 2, -3))
>>> L, b = AdvectionTerm(vel, mesh = mesh).buildMatrix(Numeric.array((4,6,1)))
>>> Numeric.allclose(b, -vel * Numeric.array((2, Numeric.sqrt(5**2 + 2**2), 5)), atol = 1e-10)
1
```

Somewhat less trivial test case:

```
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 2, ny = 2)
>>> vel = Numeric.array((3, -5, -6, -3))
>>> L, b = AdvectionTerm(vel, mesh = mesh).buildMatrix(Numeric.array((3 , 1, 6, 7)))
>>> answer = -vel * Numeric.array((2, Numeric.sqrt(2**2 + 6**2), 1, 0))
>>> Numeric.allclose(b, answer, atol = 1e-10)
1
```

5.17.1 Class AdvectionTerm

fipy.terms.term.Term
└
AdvectionTerm

Known Subclasses: HigherOrderAdvectionTerm

Methods

__init__(*self*, *coeff*=`None`, *mesh*=`None`)
Overrides: fipy.terms.term.Term.__init__

buildMatrix(*self*, *oldArray*, *coeffScale*=`None`, *varScale*=`None`, *dt*=`None`)
Overrides: fipy.terms.term.Term.buildMatrix

getDifferences(*self*, *adjacentValues*, *cellValues*, *oldArray*, *cellToCellIDs*)

Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

5.18 Module fipy.models.levelSet.advection.higherOrderAdvectionEquation

The `HigherOrderAdvectionEquation` is the same as and `AdvectionTerm` but uses a `HigherOrderAdvectionTerm`. The `HigherOrderAdvectionEquation` solves,

$$\frac{\partial \phi}{\partial t} + u|\nabla \phi|$$

5.18.1 Class HigherOrderAdvectionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.models.levelSet.advection.advectionEquation.AdvectionEquation

HigherOrderAdvectionEquation

Methods

__init__(*self*, *var*=`None`, *advectionCoeff*=`None`, *solver*=`None`)
Overrides: fipy.models.levelSet.advection.advectionEquation.AdvectionEquation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.19 Module fipy.models.levelSet.advection.higherOrderAdvectionTerm

The `HigherOrderAdvectionTerm` object constructs the `b` vector contribution for the advection term given by

$$u|\nabla\phi|$$

from the advection equation given by:

$$\frac{\partial\phi}{\partial t} + u|\nabla\phi| = 0$$

The construction of the gradient magnitude term requires upwinding as in the standard `AdvectionTerm`. The higher order terms are incorperated as follows. The formula used here is given by:

$$u_P|\nabla\phi|_P = \max\left(u_P, 0\right)\left[\sum_A \min\left(D_{AP}, 0\right)^2\right]^{1/2} + \min\left(u_P, 0\right)\left[\sum_A \max\left(D_{AP}, 0\right)^2\right]^{1/2}$$

where,

$$D_{AP} = \frac{\phi_A - \phi_P}{d_{AP}} - \frac{d_{AP}}{2} m\left(L_A, L_P\right)$$

and

$$m\left(x, y\right) = x \text{ if } |x| \leq |y| \;\; xy \geq 0$$

$$m\left(x, y\right) = y \text{ if } |x| > |y| \;\; xy \geq 0$$

$$m\left(x, y\right) = 0 \text{ if } xy < 0$$

also,

$$L_A = \frac{\phi_{AA} + \phi_P - 2\phi_A}{d_{AP}^2}$$

$$L_P = \frac{\phi_A + \phi_{PP} - 2\phi_P}{d_{AP}^2}$$

Here are some simple test cases for this problem:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 3, ny = 1)
```

Trivial test:

```
>>> from fipy.variables.cellVariable import CellVariable
>>> coeff = CellVariable(mesh = mesh, value = Numeric.zeros(3, 'd'))
>>> L, b = HigherOrderAdvectionTerm(0., mesh = mesh).buildMatrix(coeff)
>>> Numeric.allclose(b, Numeric.zeros(3, 'd'), atol = 1e-10)
1
```

Less trivial test:

```
>>> coeff = CellVariable(mesh = mesh, value = Numeric.arange(3))
>>> L, b = HigherOrderAdvectionTerm(1., mesh = mesh).buildMatrix(coeff)
>>> Numeric.allclose(b, Numeric.array((0., -1., -1.)), atol = 1e-10)
1
```

Even less trivial

```
>>> coeff = CellVariable(mesh = mesh, value = Numeric.arange(3))
>>> L, b = HigherOrderAdvectionTerm(-1., mesh = mesh).buildMatrix(coeff)
>>> Numeric.allclose(b, Numeric.array((1., 1., 0.)), atol = 1e-10)
1
```

Another trivial test case (more trivial than a trivial test case standing on a harpsichord singing 'trivial test cases are here again')

```
>>> vel = Numeric.array((-1, 2, -3))
>>> coeff = CellVariable(mesh = mesh, value = Numeric.array((4,6,1)))
>>> L, b = HigherOrderAdvectionTerm(vel, mesh = mesh).buildMatrix(coeff)
>>> Numeric.allclose(b, -vel * Numeric.array((2, Numeric.sqrt(5**2 + 2**2), 5)), atol = 1e-10)
1
```

Somewhat less trivial test case:

```
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 2, ny = 2)
>>> vel = Numeric.array((3, -5, -6, -3))
>>> coeff = CellVariable(mesh = mesh, value = Numeric.array((3 , 1, 6, 7)))
>>> L, b = HigherOrderAdvectionTerm(vel, mesh = mesh).buildMatrix(coeff)
>>> answer = -vel * Numeric.array((2, Numeric.sqrt(2**2 + 6**2), 1, 0))
>>> Numeric.allclose(b, answer, atol = 1e-10)
1
```

For the above test cases the `HigherOrderAdvectionTerm` gives the same result as the `AdvectionTerm`. The following test imposes a quadratic field. The higher order term can resolve this field correctly.

$$\phi = x^2$$

The returned vector `b` should have the value:

$$-|\nabla\phi| = -\left|\frac{\partial\phi}{\partial x}\right| = -2|x|$$

Build the test case in the following way,

```
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 5, ny = 1)
>>> vel = 1.
>>> coeff = CellVariable(mesh = mesh, value = mesh.getCellCenters()[:,0]**2)
>>> L, b = AdvectionTerm(vel, mesh = mesh).buildMatrix(coeff)
```

The first order term is not accurate. The first and last element are ignored because they don't have any neighbors for higher order evaluation

```
>>> Numeric.allclose(b[1:-1], -2 * mesh.getCellCenters()[:,0][1:-1])
0
```

The higher order term is spot on.

```
>>> L, b = HigherOrderAdvectionTerm(vel, mesh = mesh).buildMatrix(coeff)
>>> Numeric.allclose(b[1:-1], -2 * mesh.getCellCenters()[:,0][1:-1])
1
```

The `HigherOrderAdvectionTerm` will also resolve a circular field with more accuracy,

$$\phi = \left(x^2 + y^2\right)^{1/2}$$

Build the test case in the following way,

```
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 10, ny = 10)
>>> vel = 1.
>>> x = mesh.getCellCenters()[:,0]
>>> y = mesh.getCellCenters()[:,1]
>>> r = Numeric.sqrt(x**2 + y**2)
>>> coeff = CellVariable(mesh = mesh, value = r)
>>> L, b = AdvectionTerm(1., mesh = mesh).buildMatrix(coeff)
>>> error = Numeric.reshape(Numeric.reshape(b, (10,10))[2:-2,2:-2] + 1, (36,))
>>> print error[Numeric.argmax(error)]
0.123105625618
```

The maximum error is large (about 12 %) for the first order advection.

```
>>> L, b = HigherOrderAdvectionTerm(1., mesh = mesh).buildMatrix(coeff)
>>> error = Numeric.reshape(Numeric.reshape(b, (10,10))[2:-2,2:-2] + 1, (36,))
>>> print error[Numeric.argmax(error)]
0.0201715476597
```

The maximum error is 2 % when using a higher order contribution.

5.19.1 Class HigherOrderAdvectionTerm

fipy.terms.term.Term

fipy.models.levelSet.advection.advectionTerm.AdvectionTerm

HigherOrderAdvectionTerm

Methods

getDifferences(*self*, *adjacentValues*, *cellValues*, *oldArray*, *cellToCellIDs*)
Overrides: fipy.models.levelSet.advection.advectionTerm.AdvectionTerm.getDifferences

Inherited from AdvectionTerm: __init__, buildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

5.20 Module fipy.models.levelSet.distanceFunction.distanceEquation

A `DistanceEquation` object solves the equation,

$$|\nabla\phi| = 1$$

using the fast marching method with an initial condition defined by the zero level set.

Currently the solution is first order, This suffices for initial conditions with straight edges (e.g. trenches in electrodeposition). The method should work for unstructured 2D grids but testing on unstructured grids is untested thus far. This is a 2D implementation as it stands. Extending to 3D should be relatively simple.

Here we will define a few test cases. Firstly a 1D test case

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .2, nx = 8, ny = 1)
>>> from distanceVariable import DistanceVariable
>>> var = DistanceVariable(mesh = mesh, value = (-1, -1, -1, -1, 1, 1, 1, 1))
>>> eqn = DistanceEquation(var)
>>> eqn.solve()
>>> answer = (-1.75, -1.25, -.75, -0.25, 0.25, 0.75, 1.25, 1.75)
>>> Numeric.allclose(answer, Numeric.array(var))
1
```

A 1D test case with very small dimensions.

```
>>> dx = 1e-10
>>> mesh = Grid2D(dx = dx, dy = 1., nx = 8, ny = 1)
>>> var = DistanceVariable(mesh = mesh, value = (-1, -1, -1, -1, 1, 1, 1, 1))
>>> eqn = DistanceEquation(var)
>>> eqn.solve()
>>> answer = Numeric.arange(8) * dx - 3.5 * dx
>>> Numeric.allclose(answer, Numeric.array(var))
1
```

5.20.1 Class DistanceEquation

fipy.equations.equation.Equation
└
DistanceEquation

Known Subclasses: ExtensionEquation

Methods

__init__(*self*, *var*, *terminationValue*=`10000000000.0`, *maskedCells*=`()`)

The `var` argument must contain both positive and negative to define the zero level set.
The `terminationValue` represents the furthest distance from the interface that the signed distance function will be calculated.

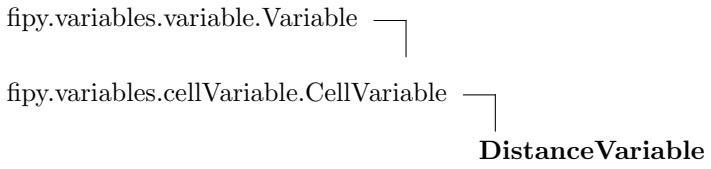
Overrides: fipy.equations.equation.Equation.__init__

solve(*self*, *terminationValue*=`None`)
Overrides: fipy.equations.equation.Equation.solve

Inherited from Equation: getFigureOfMerit, getResidual, getSolutionTolerance, getVar, isConverged, updateVar

5.21 Module fipy.models.levelSet.distanceFunction.distanceVariable

5.21.1 Class DistanceVariable



The 'DistanceVariable' evaluates quantities associated with the distance function. It is mainly evaluated in the 'DistanceFunctionEquation'.

Methods

getCellInterfaceAreas(*self*)

Returns the length of the interface that crosses the cell
A simple 1D test:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 4, ny = 1)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-1.5, -0.5, 0.5, 1.5))
>>> Numeric.allclose(distanceVariable.getCellInterfaceAreas(),
...                  (0, 0., 1., 0))
1
```

A 2D test case:

```
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 3, ny = 3)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (1.5, 0.5, 1.5,
...                                              0.5,-0.5, 0.5,
...                                              1.5, 0.5, 1.5))
>>> Numeric.allclose(distanceVariable.getCellInterfaceAreas(),
...                  (0, 1, 0, 1, 0, 1, 0, 1, 0))
1
```

Another 2D test case:

```
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> Numeric.allclose(distanceVariable.getCellInterfaceAreas(),
...                  (0, Numeric.sqrt(2) / 4,  Numeric.sqrt(2) / 4, 0))
1
```

Test to check that the circumfrence of a circle is, in fact, $2\pi r$.

```
>>> mesh = Grid2D(dx = 0.05, dy = 0.05, nx = 20, ny = 20)
>>> r = 0.25
>>> rad = Numeric.sqrt((mesh.getCellCenters()[:,0] - .5)**2
...                    + (mesh.getCellCenters()[:,1] - .5)**2) - r
>>> distanceVariable = DistanceVariable(mesh = mesh, value = rad)
>>> print Numeric.sum(distanceVariable.getCellInterfaceAreas())
1.57984690073
```

getCellInterfaceFlag(*self*)

Returns 1 for those faces on the interface:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> answer = Numeric.array((0, 1, 1, 0))
>>> Numeric.allclose(distanceVariable.getCellInterfaceFlag(), answer)
1
```

getCellInterfaceNormals(*self*)

Returns the interface normals over the cells.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> v = 1 / Numeric.sqrt(2)
>>> answer = Numeric.array((((0, 0), (0, 0), (0, 0), (0, 0)),
...                        ((0, 0), (0, 0), (0, 0), (v, v)),
...                        ((v, v), (0, 0), (0, 0), (0, 0)),
...                        ((0, 0), (0, 0), (0, 0), (0, 0))))
>>> Numeric.allclose(distanceVariable.getCellInterfaceNormals(), answer)
1
```

getCellValueOverFaces(*self*)

Returns the cells values at the faces.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> answer = Numeric.array(((-.5, -.5, -.5, -.5),
...                        (.5, .5, .5, .5),
...                        (.5, .5, .5, .5),
...                        (1.5, 1.5, 1.5, 1.5)))
>>> Numeric.allclose(distanceVariable.getCellValueOverFaces(), answer)
1
```

getDifferences(*self*, *adjacentValues*, *cellValues*, *oldArray*, *cellToCellIDs*)

getInterfaceFlag(*self*)

Returns 1 for faces on boundary and 0 otherwise.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> answer = Numeric.array((0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0))
>>> Numeric.allclose(distanceVariable.getInterfaceFlag(), answer)
1
```

getInterfaceNormals(*self*)

Returns the normals on the boundary faces only, the other are set to zero.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> v = 1 / Numeric.sqrt(2)
>>> answer = Numeric.array(((0, 0), (0, 0),
...                         (v, v), (0, 0),
...                         (0, 0), (0, 0),
...                         (0, 0), (v, v), (0, 0),
...                         (0, 0), (0, 0), (0, 0)))
>>> Numeric.allclose(distanceVariable.getInterfaceNormals(), answer)
1
```

getLevelSetNormals(*self*)

Return the face level set normals.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> v = 1 / Numeric.sqrt(2)
>>> answer = Numeric.array(((0, 0), (0, 0), (v, v), (v, v), (0, 0), (0, 0),
...                         (0, 0), (v, v), (0, 0), (0, 0), (v, v), (0, 0)))
>>> Numeric.allclose(distanceVariable.getLevelSetNormals(), answer)
1
```

getUpwindMag(*self*)

Inherited from CellVariable: __init__, __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld

Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__,

__gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.22 Module fipy.models.levelSet.distanceFunction.extensionEquation

A `ExtensionEquation` object solves the equation,

$$\nabla u \cdot \nabla \phi = 0$$

using the fast marching method with an initial condition defined at the zero level set. Essentially the equation solves a fake distance function equation to march out the velocity from the interface.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> from distanceVariable import DistanceVariable
>>> from fipy.variables.cellVariable import CellVariable
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 2, ny = 2)
>>> var = DistanceVariable(mesh = mesh, value = (-1, 1, 1, 1))
>>> extensionVar = CellVariable(mesh = mesh, value = (-1, .5, 2, -1))
>>> setValueFlag = ExtensionEquation(var, extensionVar).solve()
>>> Numeric.allclose((-1, 1, 1, 1), Numeric.array(var))
1
>>> Numeric.allclose((1.25, .5, 2, 1.25), Numeric.array(extensionVar))
1

>>> mesh = Grid2D(dx = 1., dy = 1., nx = 3, ny = 3)
>>> var = DistanceVariable(mesh = mesh, value = (-1, 1, 1,
...                                              1, 1, 1,
...                                              1, 1, 1))
>>> extensionVar = CellVariable(mesh = mesh, value = (-1, .5, -1,
...                                                    2, -1, -1,
...                                                   -1, -1, -1))
>>> setValueFlag = ExtensionEquation(var, extensionVar).solve()
>>> answer = (1.25, .5, .5, 2, 1.25, 0.9544, 2, 1.5456, 1.25)
>>> Numeric.allclose(answer, Numeric.array(extensionVar), atol = 1e-5)
1
```

5.22.1 Class ExtensionEquation

fipy.equations.equation.Equation

fipy.models.levelSet.distanceFunction.distanceEquation.DistanceEquation

ExtensionEquation

Methods

__init__(*self*, *var*=`None`, *extensionVar*=`None`, *terminationValue*=`10000000000.0`, *maskedCells*=`()`)

The `var` argument must contain both positive and negative values to define the zero level set.
The `extensionVar` must be defined on the positive interface cells.
The `terminationValue` represents the furthest distance from the interface that the signed distance function will be calculated.

Overrides: fipy.models.levelSet.distanceFunction.distanceEquation.DistanceEquation.__init__

solve(*self*, *dt*=`None`)
Overrides: fipy.models.levelSet.distanceFunction.distanceEquation.DistanceEquation.solve

Inherited from Equation: getFigureOfMerit, getResidual, getSolutionTolerance, getVar, isConverged, updateVar

5.23 Module fipy.models.levelSet.distanceFunction.levelSetDiffusionEq

The `LevelSetDiffusionEquation` solves the diffusion of a species in conjunction with the level set equation. Essentially the spcies is only transported in the electrolyte. The governing equation is given by,

$$\frac{\partial c}{\partial t} = \nabla \cdot D \nabla c$$

where,

$$D = D_c \text{ when } \phi > 0$$

$$D = 0 \text{ when } \phi \leq 0$$

5.23.1 Class LevelSetDiffusionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.diffusionEquation.DiffusionEquation

LevelSetDiffusionEquation

Known Subclasses: MetalIonDiffusionEquation, SurfactantBulkDiffusionEquation

Methods

__init__(*self*, *var*, *distanceVar*=`None`, *transientCoeff*=`1.0`, *diffusionCoeff*=`1.0`, *solver*=<`fipy.solvers.linearPCGSolver.LinearPCGSolver instance at...`, *boundaryConditions*=`()`, *otherTerms*=`()`)

A `LevelSetDiffusionEquation` is instantiated with the following arguments,
`var` - The species concentration variable.
`distanceVar` - A `DistanceVariable` object
`transientCoeff` - In general 1 is used.
`solver` - A given solver.
`boundaryConditions` - A tuple of `BoundaryCondition` objects.

Overrides: fipy.equations.diffusionEquation.DiffusionEquation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.24 Module fipy.models.levelSet.distanceFunction.levelSetDiffusionVariable

This variable sets it's face value to zero if either of the surrounding cell values are zero else it uses the value of the diffusion coefficient. The diffusion coefficient is given by,

$$D = D_c \text{ when } \phi > 0$$

$$D = 0 \text{ when } \phi \leq 0$$

Here is a simpel 1D test case:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 3, ny = 1)
>>> from fipy.variables.cellVariable import CellVariable
>>> var = CellVariable(mesh = mesh, value = (-1, 1, 1))
>>> arr = Numeric.array(LevelSetDiffusionVariable(var, 1))
>>> Numeric.allclose(arr, (0,1,1,0,1,1,0,0,1,1))
1
```

5.24.1 Class LevelSetDiffusionVariable

fipy.variables.variable.Variable

fipy.variables.faceVariable.FaceVariable

fipy.variables.cellToFaceVariable.CellToFaceVariable

LevelSetDiffusionVariable

Methods

__init__(*self*, *distanceVariable*=`None`, *diffusionCoeff*=`None`)

Requires the following arguments to instantiate,
`distanceVariable` - A `DistanceVariable` object
`diffusionCoeff` - Either a `CellVariable` or a single value

Overrides: fipy.variables.cellToFaceVariable.CellToFaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.25 Module fipy.models.levelSet.electroChem.depositionRateVariable

The deposition rate variable is the distance the interface moves in a unit of time. It is proportional to the current density such that,

$$v = \frac{i\Omega}{nF}$$

The density is given by,

$$i = i_0 \frac{c_c^i}{c_c^\infty} \exp\left(\frac{-\alpha F}{RT}\eta\right)$$

The exchange current density is an empirical function of accelerator coverage,

$$i_0(\theta) = b_0 + b_1\theta$$

Here is a small test,

```
>>> parameters = {
...     'experimental parameters' :
...         {
...             'transfer coefficient' :
...                 {
...                     'constant'               : 0.5,
...                     'accelerator dependence' : 0.
...                 },
...             'temperature'              : 298.,
...             'overpotential'            : -0.3,
...             'bulk metal ion concentration' : 278.,
...             'exchange current density' :
...                 {
...                     'constant'               : 0.26,
...                     'accelerator dependence' : 45.
...                 }
...         },
...     'material properties' :
...         {
...             'Faradays constant' : 9.6e4,
...             'gas constant'      : 8.314
...         },
...     'metal ion properties' :
...         {
...             'atomic volume' : 7.1e-6,
...             'ion charge'    : 2.
...         }
...     }
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(nx = 2, ny = 1, dx = 1., dy = 1.)
>>> from fipy.models.levelSet.distanceFunction.distanceVariable import DistanceVariable
```

```
>>> distanceVariable = DistanceVariable(mesh = mesh, value = (-0.5, 0.5))
>>> from fipy.models.levelSet.surfactant.surfactantVariable import SurfactantVariable
>>> acceleratorVariable = SurfactantVariable(distanceVar = distanceVariable, value = (0, 1)
>>> print DepositionRateVariable(metalIonVariable = 278., acceleratorVariable = accelerator
[  3.21448659e-09,  5.59567934e-07,]
>>> print DepositionRateVariable(metalIonVariable = parameters['experimental parameters']['
[  1.60724329e-09,  2.79783967e-07,]
```

5.25.1 Class DepositionRateVariable

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

DepositionRateVariable

Methods

__init__(*self*, *metalIonVariable*, *acceleratorVariable*=`None`, *overpotential*=`None`, *parameters*=`None`, *name*=`'deposition rate variable'`)

The following arguments are required to instatiate a `MetalIonSourceVariable`.
`ionVar` - A `CellVariable`.
`overpotential` - An `Variable` or float object
`parameters` - A dictionary with the correct form as given in the test.
`acceleratorCoverage` - the accelerator coverage close to the interface, must a `SurfactantVariable`

Overrides: fipy.variables.cellVariable.CellVariable.__init__

getCurrentDensity(*self*)

getExpoConstant(*self*)

getTransferCoefficient(*self*)

setOverpotential(*self*, *overpotential*)

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__,

__repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.26 Module fipy.models.levelSet.electroChem.metalIonDiffusionEquat

The `MetalIonDiffusionEquation` solves the diffusion of the metal species with a source term at the electrolyte interface. The governing equation is given by,

$$\frac{\partial c}{\partial t} = \nabla \cdot D \nabla c$$

where,

$$D = D_c \text{ when } \phi > 0$$

$$D = 0 \text{ when } \phi \leq 0$$

The velocity of the interface generally has a linear dependence on ion concentration. The following boundary condition applies at the zero level set,

$$D\hat{n} \cdot \nabla c = \frac{v(c)}{\Omega} \text{ at } \phi = 0$$

where

$$v(c) = cV_0$$

The test case below is for a 1D steady state problem. The solution is given by:

$$c(x) = \frac{c^\infty}{\Omega D / V_0 + L} (x - L) + c^\infty$$

This is the test case,

```
>>> import Numeric
>>> from fipy.meshes.grid2D import Grid2D
>>> nx = 11
>>> dx = 1.
>>> mesh = Grid2D(nx = nx, ny = 1, dx = dx, dy = 1)
>>> from fipy.variables.cellVariable import CellVariable
>>> ionVar = CellVariable(mesh = mesh, value = 1)
>>> from fipy.models.levelSet.distanceFunction.distanceVariable import DistanceVariable
>>> disVar = DistanceVariable(mesh = mesh, value = Numeric.arange(11) - 0.99)

>>> v = 1.
>>> diffusion = 1.
>>> omega = 1.
>>> cinf = 1.
>>> from fipy.boundaryConditions.fixedValue import FixedValue
>>> eqn = MetalIonDiffusionEquation(ionVar,
...                                 distanceVar = disVar,
...                                 depositionRate = v * ionVar,
...                                 diffusionCoeff = diffusion,
...                                 metalIonAtomicVolume = omega,
...                                 boundaryConditions = (
...                                     FixedValue(mesh.getFacesRight(), cinf),))
```

```
>>> for i in range(10):
...     eqn.solve(dt = 1000)
>>> L = (nx - 1) * dx - dx / 2
>>> gradient = cinf / (omega * diffusion / v + L)
>>> answer = gradient * (mesh.getCellCenters()[:,0] - L - dx * 3 / 2) + cinf
>>> answer[0] = 1
>>> Numeric.allclose(answer, Numeric.array(ionVar))
1
```

5.26.1 Class MetalIonDiffusionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.diffusionEquation.DiffusionEquation

fipy.models.levelSet.distanceFunction.levelSetDiffusionEquation.LevelSetDiffusionEquation

MetalIonDiffusionEquatio

Methods

__init__(*self*, *var*, *distanceVar*=`None`, *depositionRate*=`1`, *diffusionCoeff*=`1`, *transientCoeff*=`1`, *metalIonAtomicVolume*=`1`, *boundaryConditions*=`()`)

A `MetalIonDiffusionEquation` is instantiated with the following arguments,
`var` - The metal ion concentration variable.
`distanceVariable` - A `DistanceVariable` object
`depositionRate` - A float or a `CellVariable` representing the interface deposition rate.
`diffusionCoeff` - A float or a `FaceVariable`.
`transientCoeff` - In general 1 is used.
`metalIonAtomicVolume` - Atomic volume of the metal ions.
`solver` - A given solver.
`boundaryConditions` - A tuple of `BoundaryCondition` objects.

Overrides:
fipy.models.levelSet.distanceFunction.levelSetDiffusionEquation.LevelSetDiffusionEquation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.27 Module fipy.models.levelSet.electroChem.metalIonSourceVariable

The `MetalIonSourceVariable` object evaluates the source coefficient for the `MetalIonEquation`. The source only exists on the cells in front of the interface and simulates the loss of material from bulk diffusion as the interface advances. The source is given by,

$$D\hat{n} \cdot \nabla c = \frac{v(c)}{\Omega} \text{ at } \phi = 0$$

Here is a test,

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1, dy = 1, nx = 2, ny = 2)
>>> from fipy.models.levelSet.distanceFunction.distanceVariable import DistanceVariable
>>> distance = DistanceVariable(mesh = mesh, value = (-.5, .5, .5, 1.5))
>>> ionVar = CellVariable(mesh = mesh, value = (1, 1, 1, 1))
>>> arr = Numeric.array(MetalIonSourceVariable(ionVar, distance, (1, 1, 1, 1), 1))
>>> sqrt = Numeric.sqrt(2)
>>> Numeric.allclose(arr, (0, 1 / sqrt, 1 / sqrt, 0))
1
```

5.27.1 Class MetalIonSourceVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

MetalIonSourceVariable

Methods

__init__(*self*, *ionVar*=`None`, *distanceVar*=`None`, *depositionRate*=`None`, *metalIonAtomicVolume*=`None`)

The following arguments are required to instatiate a `MetalIonSourceVariable`,
`ionVar` - A `CellVariable`.
`distanceVar` - A `DistanceVariable` object.
`depositionRate` - Either a `CellVariable` or a float.
`metalIonAtomicVolume` - Atomic volume of the metal ions.

Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__,

__gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.28 Module fipy.models.levelSet.electroChem.test

5.28.1 Functions

suite()

5.29 Module fipy.models.levelSet.surfactant.adsorbingSurfactantEquation

The `AdsorbingSurfactantEquation` object solves the `SurfactantEquation` but with an adsorbing species from some bulk value. The equation that describes the surfactant adsorbing is given by,

$$\dot{\theta} = Jv\theta + kc(1 - \theta)$$

This last term in this equation accounts for Langmuir type adsorption from the bulk. It assumes a vacant proportion of surface sites. The adsorption term is added to the source by setting $S_c = kc$ and $S_p = kc$. The following is a test case:

```
>>> from fipy.models.levelSet.distanceFunction.distanceVariable import DistanceVariable
>>> from fipy.models.levelSet.surfactant.surfactantVariable import SurfactantVariable
>>> from fipy.meshes.grid2D import Grid2D
>>> dx = .5
>>> dy = 2.3
>>> dt = 0.25
>>> k = 0.56
>>> initialValue = 0.1
>>> c = 0.2
>>> mesh = Grid2D(dx = dx, dy = dy, nx = 5, ny = 1)
>>> distanceVar = DistanceVariable(mesh = mesh,
...                                value = (-dx*3/2, -dx/2, dx/2, 3*dx/2 ,5*dx/2))
>>> var = SurfactantVariable(value = (0, 0, initialValue, 0 ,0),
...                          distanceVar = distanceVar)
>>> bulkVar = CellVariable(mesh = mesh, value = (c , c, c, c, c))
>>> eqn = AdsorbingSurfactantEquation(var, distanceVar, bulkVar, k)
>>> eqn.solve(dt = dt)
>>> answer = (initialValue + dt * k * c) / (1 + dt * k * c)
>>> Numeric.allclose(var.getInterfaceVar(), Numeric.array((0, 0, answer, 0, 0)))
1
```

5.29.1 Class AdsorbingSurfactantEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.models.levelSet.surfactant.surfactantEquation.SurfactantEquation

AdsorbingSurfactantEquation

Methods

__init__(*self*, *var*, *distanceVar*, *bulkVar*, *rateConstant*, *scCoeff*=`None`, *spCoeff*=`None`)
Overrides: fipy.models.levelSet.surfactant.surfactantEquation.SurfactantEquation.__init__

solve(*self*, *dt*)
Overrides: fipy.models.levelSet.surfactant.surfactantEquation.SurfactantEquation.solve

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve

5.29.2 Class AdsorptionCoeff

fipy.variables.variable.Variable
fipy.variables.cellVariable.CellVariable
AdsorptionCoeff

Known Subclasses: AdsorptionCoeffAreaOverVolume, AdsorptionCoeffInterfaceFlag

Methods

__init__(*self*, *distanceVar*, *bulkVar*, *rateConstant*)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

updateDt(*self*, *dt*)

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.29.3 Class AdsorptionCoeffAreaOverVolume

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.levelSet.surfactant.adsorbingSurfactantEquation.AdsorptionCoeff

AdsorptionCoeffAreaOverVolume

Methods

multiplier(*self*)

Inherited from AdsorptionCoeff: __init__, updateDt
Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.29.4 Class AdsorptionCoeffInterfaceFlag

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.levelSet.surfactant.adsorbingSurfactantEquation.AdsorptionCoeff

AdsorptionCoeffInterfaceFlag

Methods

multiplier(*self*)

Inherited from AdsorptionCoeff: __init__, updateDt
Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.30 Module fipy.models.levelSet.surfactant.convectionCoeff

5.30.1 Class ConvectionCoeff

fipy.variables.variable.Variable

fipy.variables.vectorFaceVariable.VectorFaceVariable

ConvectionCoeff

Convection coefficient for the `ConservativeSurfactantEquation`. The coeff only has a value for a negative `distanceVar`.

Methods

__init__(*self*, *distanceVar*)

Simple one dimensional test:

```
>>> from fipy.variables.cellVariable import CellVariable
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(nx = 3, ny = 1, dx = 1., dy = 1.)
>>> distanceVar = CellVariable(mesh, value = (-.5, .5, 1.5))
>>> answer = Numeric.zeros((mesh.getNumberOfFaces(),2),'d')
>>> answer[7,0] = -1
>>> Numeric.allclose(ConvectionCoeff(distanceVar), answer)
1
```

Change the dimensions:

```
>>> mesh = Grid2D(nx = 3, ny = 1, dx = .5, dy = .25)
>>> distanceVar = CellVariable(mesh, value = (-.25, .25, .75))
>>> answer[7,0] = -.5
>>> Numeric.allclose(ConvectionCoeff(distanceVar), answer)
1
```

Two dimensional example:

```
>>> mesh = Grid2D(nx = 2, ny = 2, dx = 1., dy = 1.)
>>> distanceVar = CellVariable(mesh, value = (-1.5, -.5, -.5, .5))
>>> answer = Numeric.zeros((mesh.getNumberOfFaces(),2),'d')
>>> answer[2,1] = -.5
>>> answer[3,1] = -1
>>> answer[7,0] = -.5
>>> answer[10,0] = -1
>>> Numeric.allclose(ConvectionCoeff(distanceVar), answer)
1
```

Larger grid:

```
>>> mesh = Grid2D(nx = 3, ny = 3, dx = 1., dy = 1.)
>>> distanceVar = CellVariable(mesh, value = (1.5, .5 , 1.5,
...                                           .5 , -.5, .5 ,
...                                           1.5, .5 , 1.5))
>>> answer = Numeric.zeros((mesh.getNumberOfFaces(),2), 'd')
>>> answer[4,1] = .25
>>> answer[7,1] = -.25
>>> answer[7,1] = -.25
>>> answer[17,0] = .25
>>> answer[18,0] = -.25
>>> Numeric.allclose(ConvectionCoeff(distanceVar), answer)
1
```

Overrides: fipy.variables.vectorFaceVariable.VectorFaceVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__,

__rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.31 Module fipy.models.levelSet.surfactant.surfactantBulkDiffusionEq

The `SurfactantBulkDiffusionEquation` solves the bulk diffusion of a species with a source term for the jump from the bulk to an interface. The governing equation is given by,

$$\frac{\partial c}{\partial t} = \nabla \cdot D \nabla c$$

where,

$$D = D_c \text{ when } \phi > 0$$

$$D = 0 \text{ when } \phi \leq 0$$

The jump condition at the interface is defined by Langmuir adsorption. Langmuir adsorption essentially states that the ability for a species to jump from an electrolyte to an interface is proportional to the concentration in the electrolyte, available site density and a jump coefficient. The boundary condition at the interface is given by

$$D\hat{n} \cdot \nabla c = -kc(1-\theta) \text{ at } \phi = 0$$

5.31.1 Class AdsorptionCoeff

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

AdsorptionCoeff

Known Subclasses: ScAdsorptionCoeff

Methods

__init__(*self*, *rateConstant*=`None`, *distanceVar*=`None`) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.31.2 Class ScAdsorptionCoeff

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.levelSet.surfactant.surfactantBulkDiffusionEquation.AdsorptionCoeff

ScAdsorptionCoeff

Methods

__init__(*self*, *bulkVar*=`None`, *surfactantVar*=`None`, *rateConstant*=`None`, *distanceVar*=`None`)
Overrides:
fipy.models.levelSet.surfactant.surfactantBulkDiffusionEquation.AdsorptionCoeff.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.31.3 Class SurfactantBulkDiffusionEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

fipy.equations.diffusionEquation.DiffusionEquation

fipy.models.levelSet.distanceFunction.levelSetDiffusionEquation.LevelSetDiffusionEquation

SurfactantBulkDif

Methods

__init__(*self*, *var*, *distanceVar*=`None`, *surfactantVar*=`None`, *diffusionCoeff*=`None`, *transientCoeff*=`1.0`, *rateConstant*=`None`, *boundaryConditions*=`()`)

A `SurfactantBulkDiffusionEquation` is instantiated with the following arguments,
`var` - The bulk surfactant concentration variable.
`distanceVar` - A `DistanceVariable` object
`surfactantVariable` - A `SurfactantVariable` object
`diffusionCoeff` - A float or a `FaceVariable`.
`transientCoeff` - In general 1 is used.
`jumpRate` - The adsorption coefficient.
`boundaryConditions` - A tuple of `BoundaryCondition` objects.

Overrides:
fipy.models.levelSet.distanceFunction.levelSetDiffusionEquation.LevelSetDiffusionEquation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.32 Module fipy.models.levelSet.surfactant.surfactantEquation

5.32.1 Class SurfactantEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

SurfactantEquation

Known Subclasses: AdsorbingSurfactantEquation

A ‘SurfactantEquation‘ aims to evolve a surfactant on an interface defined by the zero level set of the ‘distanceVar‘. The method should completely conserve the total coverage of surfactant. The surfactant is only in the cells immediately in front of the advancing interface. The method only works for a positive velocity as it stands.

Methods

__init__(*self*, *var*, *distanceVar*,
solver=<`fipy.solvers.linearLUSolver.LinearLUSolver instance at 0...`,
boundaryConditions=`None`)
Overrides: fipy.equations.equation.Equation.__init__

solve(*self*, *dt*=`1.0`)
Overrides: fipy.equations.matrixEquation.MatrixEquation.solve

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve

5.33 Module fipy.models.levelSet.surfactant.surfactantVariable

5.33.1 Class InterfaceSurfactantVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

InterfaceSurfactantVariable

Methods

__init__(*self*, *surfactantVar*) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.33.2 Class SurfactantVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

SurfactantVariable

`SurfactantVariable` initializes itself across the zero level set interface using the delta function. The `value` argument corresponds to the initial concentration of surfactant on the interface (moles divided by area). The value held by the `SurfactantVariable` is actually a volume density (moles divided by volume).

Methods

__init__(*self*, *value*=0.0, *distanceVar*=None, *name*='surfactant variable')

A simple 1D test:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 4, ny = 1)
>>> from fipy.models.levelSet.distanceFunction.distanceVariable \
...     import DistanceVariable
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-1.5, -0.5, 0.5, 941.5))
>>> surfactantVariable = SurfactantVariable(value = 1,
...                                         distanceVar = distanceVariable)
>>> Numeric.allclose(surfactantVariable, (0, 0., 1., 0))
1
```

A 2D test case:

```
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 3, ny = 3)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (1.5, 0.5, 1.5,
...                                              0.5,-0.5, 0.5,
...                                              1.5, 0.5, 1.5))
>>> surfactantVariable = SurfactantVariable(value = 1,
...                                         distanceVar = distanceVariable)
>>> Numeric.allclose(surfactantVariable, (0, 1, 0, 1, 0, 1, 0, 1, 0))
1
```

Another 2D test case:

```
>>> mesh = Grid2D(dx = .5, dy = .5, nx = 2, ny = 2)
>>> distanceVariable = DistanceVariable(mesh = mesh,
...                                     value = (-0.5, 0.5, 0.5, 1.5))
>>> surfactantVariable = SurfactantVariable(value = 1,
...                                         distanceVar = distanceVariable)
>>> Numeric.allclose(surfactantVariable,
...                  (0, Numeric.sqrt(2), Numeric.sqrt(2), 0))
1
```

Overrides: fipy.variables.cellVariable.CellVariable.__init__

getDistanceVar(*self*)

getInterfaceVar(*self*)

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBi-

naryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.34 Module fipy.models.levelSet.test

5.34.1 Functions

suite()

5.35 Module fipy.models.phase.phase.addOverFacesVariable

5.35.1 Class AddOverFacesVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

AddOverFacesVariable

Methods

__init__(*self*, *faceGradient*=`None`, *faceVariable*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.36 Module fipy.models.phase.phase.anisotropyVariable

5.36.1 Class AnisotropyVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

AnisotropyVariable

Methods

__init__(*self*, *parameters*=`None`, *phase*=`None`, *halfAngle*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.36.2 Class DPhiReverse

fipy.variables.variable.Variable

fipy.variables.vectorFaceVariable.VectorFaceVariable

DPhiReverse

Methods

__init__(*self*, *phase*) Overrides: fipy.variables.vectorFaceVariable.VectorFaceVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.36.3 Class FFVariable

fipy.variables.variable.Variable
fipy.variables.faceVariable.FaceVariable
FFVariable

Methods

__init__(*self*, *parameters*=`None`, *halfAngle*=`None`) Overrides: fipy.variables.faceVariable.FaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.37 Module fipy.models.phase.phase.mPhiVariable

5.37.1 Class MPhiVariable

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

MPhiVariable

Known Subclasses: Type1MPhiVariable, Type2MPhiVariable

Methods

__**init**__(*self*, *phase*=`None`, *temperature*=`None`, *parameters*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.38 Module fipy.models.phase.phase.phaseDiffusionVariable

5.38.1 Class PhaseDiffusionVariable

fipy.variables.variable.Variable ┐

fipy.variables.faceVariable.FaceVariable ┐

PhaseDiffusionVariable

Methods

__init__(*self*, *parameters*=`None`, *halfAngle*=`None`) Overrides: fipy.variables.faceVariable.FaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.39 Module fipy.models.phase.phase.phaseEquation

5.39.1 Class PhaseEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

PhaseEquation

Methods

__**init**__(*self*, *var*, *solver*=`'default_solver'`, *boundaryConditions*=`()`, *fields*={}, *parameters*={}, *mPhi*=`0.0`)
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.40 Module fipy.models.phase.phase.phaseHalfAngleVariable

5.40.1 Class PhaseHalfAngleVariable

fipy.variables.variable.Variable

fipy.variables.faceVariable.FaceVariable

PhaseHalfAngleVariable

Methods

__init__(*self*, *parameters*=`None`, *phase*=`None`, *theta*=`None`)
Overrides: fipy.variables.faceVariable.FaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.41 Module fipy.models.phase.phase.scSourceVariable

5.41.1 Class ScSourceVariable

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

ScSourceVariable

Methods

__init__(*self*, *mPhi*=`None`, *phase*=`None`, *anisotropy*=`None`) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.42 Module fipy.models.phase.phase.spSourceVariable

5.42.1 Class SpSourceVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

SpSourceVariable

Methods

__init__(*self*, *theta*=`None`, *mPhi*=`None`, *phase*=`None`, *parameters*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.43 Module fipy.models.phase.phase.type1MPhiVariable

5.43.1 Class Type1MPhiVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.phase.phase.mPhiVariable.MPhiVariable

Type1MPhiVariable

Methods

Inherited from MPhiVariable: __init__
Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.44 Module fipy.models.phase.phase.type2MPhiVariable

5.44.1 Class Type2MPhiVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

fipy.models.phase.phase.mPhiVariable.MPhiVariable

Type2MPhiVariable

Methods

Inherited from MPhiVariable: __init__
Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.45 Module fipy.models.phase.temperature.temperatureEquation

5.45.1 Class TemperatureEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

TemperatureEquation

Methods

__init__(*self*, *var*, *solver*=`'default_solver'`, *boundaryConditions*=`()`, *fields*={}, *parameters*={})
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.46 Module fipy.models.phase.test

5.46.1 Functions

suite()

5.47 Module fipy.models.phase.theta.diffusionVariable

5.47.1 Class DiffusionVariable

fipy.variables.variable.Variable ┐

fipy.variables.faceVariable.FaceVariable ┐

DiffusionVariable

Methods

__init__(*self*, *phase*=`None`, *theta*=`None`, *parameters*=`None`) Overrides: fipy.variables.faceVariable.FaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.48 Module fipy.models.phase.theta.modCellGradVariable

5.48.1 Class ModCellGradVariable

fipy.variables.variable.Variable

fipy.variables.vectorCellVariable.VectorCellVariable

fipy.variables.cellGradVariable.CellGradVariable

ModCellGradVariable

Methods

__init__(*self*, *var*, *modIn*, *modPy*) Overrides: fipy.variables.cellGradVariable.CellGradVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorCellVariable: dot, getArithmeticFaceValue, getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.49 Module fipy.models.phase.theta.modCellToFaceVariable

5.49.1 Class ModCellToFaceVariable

fipy.variables.variable.Variable

fipy.variables.faceVariable.FaceVariable

fipy.variables.cellToFaceVariable.CellToFaceVariable

fipy.variables.arithmeticCellToFaceVariable.ArithmeticCellToFaceVariable

ModCellToFaceVariable

Methods

__init__(*self*, *var*, *modIn*)
Overrides: fipy.variables.cellToFaceVariable.CellToFaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.50 Module fipy.models.phase.theta.modFaceGradVariable

5.50.1 Class ModFaceGradVariable

fipy.variables.variable.Variable

fipy.variables.vectorFaceVariable.VectorFaceVariable

fipy.variables.faceGradVariable.FaceGradVariable

ModFaceGradVariable

Methods

__init__(*self*, *var*, *modIn*) Overrides: fipy.variables.faceGradVariable.FaceGradVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.51 Module fipy.models.phase.theta.modPhysicalField

5.51.1 Class ModPhysicalField

fipy.tools.dimensions.physicalField.PhysicalField
ModPhysicalField

Methods

__rsub__(*self*, *other*)
Overrides: fipy.tools.dimensions.physicalField.PhysicalField.__rsub__

__sub__(*self*, *other*)

Subtract two physical quantities, so long as their units are compatible. The unit of the result is the unit of the first operand.

```
>>> print PhysicalField(10., 'km') - PhysicalField(10., 'm')
9.99 km
>>> print PhysicalField(10., 'km') - PhysicalField(10., 'J')
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

Overrides: fipy.tools.dimensions.physicalField.PhysicalField.__sub__ extit(inherited documentation)

mod(*self*, *argument*)

Inherited from PhysicalField: __init__, __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __nonzero__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __setitem__, __str__, allclose, arctan, arctan2, convertToUnit, copy, cos, dot, getNumericValue, getUnit, inBaseUnits, inUnitsOf, isCompatible, put, reshape, sin, sqrt, sum, take, tan

5.52 Module fipy.models.phase.theta.modularVariable

5.52.1 Class ModularVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

ModularVariable

Methods

__init__(*self*, *mesh*, *name*=`''`, *value*=`0.0`, *unit*=`None`, *hasOld*=`0`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

getArithmeticFaceValue(*self*)
Overrides: fipy.variables.cellVariable.CellVariable.getArithmeticFaceValue

getFaceGrad(*self*)
Overrides: fipy.variables.cellVariable.CellVariable.getFaceGrad

getGrad(*self*)
Overrides: fipy.variables.cellVariable.CellVariable.getGrad

updateOld(*self*)
Overrides: fipy.variables.cellVariable.CellVariable.updateOld

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getCellVolumeAverage, getFaceDifference, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.53 Module fipy.models.phase.theta.noModularVariable

5.53.1 Class NoModularVariable

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

NoModularVariable

Methods

__init__(*self*, *modVar*) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.54 Module fipy.models.phase.theta.sourceVariable

5.54.1 Class SourceVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

SourceVariable

Methods

__init__(*self*, *phase*=`None`, *theta*=`None`, *diffCoeff*=`None`, *halfAngleVariable*=`None`, *parameters*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.55 Module fipy.models.phase.theta.thetaEquation

5.55.1 Class ThetaEquation

fipy.equations.equation.Equation

fipy.equations.matrixEquation.MatrixEquation

ThetaEquation

Methods

__**init**__(*self*, *var*=`None`, *solver*=`'default_solver'`, *boundaryConditions*=`()`, *fields*={}, *parameters*={})
Overrides: fipy.equations.equation.Equation.__init__

Inherited from Equation: getFigureOfMerit, getSolutionTolerance, isConverged, updateVar
Inherited from MatrixEquation: buildMatrix, getResidual, getResidual2, getVar, postSolve, solve

5.56 Module fipy.models.phase.theta.thetaHalfAngleVariable

5.56.1 Class ThetaHalfAngleVariable

fipy.variables.variable.Variable ┐

fipy.variables.cellVariable.CellVariable ┐

ThetaHalfAngleVariable

Methods

__init__(*self*, *parameters*=`None`, *phase*=`None`, *theta*=`None`) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.57 Module fipy.models.phase.theta.transientVariable

5.57.1 Class TransientVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

TransientVariable

Methods

__init__(*self*, *phase*=`None`, *theta*=`None`, *parameters*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

5.58 Module fipy.models.test

5.58.1 Functions

suite()

Chapter 6

Module fipy.solvers

6.1 Module fipy.solvers.linearCGSSolver

6.1.1 Class LinearCGSSolver

fipy.solvers.solver.Solver ┐
LinearCGSSolver

Methods

__init__(*self*, *tolerance*, *steps*)
Overrides: fipy.solvers.solver.Solver.__init__

solve(*self*, *L*, *x*, *b*)
Overrides: fipy.solvers.solver.Solver.solve

6.2 Module fipy.solvers.linearGMRESSolver

6.2.1 Class LinearGMRESSolver

fipy.solvers.solver.Solver ┐
LinearGMRESSolver

Methods

__init__(*self*, *tolerance*, *steps*)
Overrides: fipy.solvers.solver.Solver.__init__

solve(*self*, *L*, *x*, *b*)
Overrides: fipy.solvers.solver.Solver.solve

6.3 Module fipy.solvers.linearLUSolver

6.3.1 Class LinearLUSolver

fipy.solvers.solver.Solver
└
LinearLUSolver

Methods

__init__(*self*, *tolerance*=`1e-10`, *steps*=`10`)
Overrides: fipy.solvers.solver.Solver.__init__

solve(*self*, *L*, *x*, *b*)
Overrides: fipy.solvers.solver.Solver.solve

6.4 Module fipy.solvers.linearPCGSolver

6.4.1 Class LinearPCGSolver

fipy.solvers.solver.Solver
LinearPCGSolver

Methods

__init__(*self*, *tolerance*, *steps*)
Overrides: fipy.solvers.solver.Solver.__init__

solve(*self*, *L*, *x*, *b*)
Overrides: fipy.solvers.solver.Solver.solve

6.5 Module fipy.solvers.linearScipyLUSolver

6.5.1 Class LinearScipyLUSolver

fipy.solvers.solver.Solver
└
LinearScipyLUSolver

Methods

__**init**__(*self*, *tolerance*=`1e-10`, *steps*=`10`)
Overrides: fipy.solvers.solver.Solver.__init__

solve(*self*, *L*, *x*, *b*)
Overrides: fipy.solvers.solver.Solver.solve

6.6 Module fipy.solvers.linearScipySolver

6.6.1 Class LinearScipySolver

fipy.solvers.solver.Solver
└
LinearScipySolver

Methods

__init__(*self*, *tolerance*=`1e-10`, *steps*=`10`)
Overrides: fipy.solvers.solver.Solver.__init__

solve(*self*, *L*, *x*, *b*)
Overrides: fipy.solvers.solver.Solver.solve

6.7 Module fipy.solvers.solver

6.7.1 Class Solver

Known Subclasses: LinearCGSSolver, LinearGMRESSolver, LinearLUSolver, LinearPCGSolver, LinearScipyLUSolver, LinearScipySolver

Methods

__init__(*self*, *tolerance*, *steps*)

solve(*self*, *L*, *x*, *b*)

Chapter 7

Module fipy.terms

7.1 Module fipy.terms.cellTerm

7.1.1 Class CellTerm

fipy.terms.term.Term
└
CellTerm

Known Subclasses: SourceTerm, TransientTerm

Methods

__init__(*self*, *weight*, *mesh*) Overrides: fipy.terms.term.Term.__init__

buildMatrix(*self*, *oldArray*, *coeffScale*, *varScale*, *dt*) Overrides: fipy.terms.term.Term.buildMatrix

Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.2 Module fipy.terms.centralDiffConvectionTerm

7.2.1 Class CentralDifferenceConvectionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.convectionTerm.ConvectionTerm

CentralDifferenceConvectionTerm

Methods

Inherited from ConvectionTerm: __init__, getWeight
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

Class Variables

Name	Description

7.3 Module fipy.terms.convectionTerm

7.3.1 Class ConvectionTerm

fipy.terms.term.Term ┐

fipy.terms.faceTerm.FaceTerm ┐

ConvectionTerm

Known Subclasses: CentralDifferenceConvectionTerm, ExponentialConvectionTerm, HybridConvectionTerm, PowerLawConvectionTerm, UpwindConvectionTerm

Methods

__init__(*self*, *convCoeff*, *mesh*, *boundaryConditions*, *diffusionTerm*=`None`)
Overrides: fipy.terms.faceTerm.FaceTerm.__init__

getWeight(*self*, *alpha*)

Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.4 Module fipy.terms.diffusionTerm

7.4.1 Class DiffusionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

DiffusionTerm

Known Subclasses: ExplicitDiffusionTerm, ImplicitDiffusionTerm

Methods

__init__(*self*, *diffCoeff*, *mesh*, *boundaryConditions*, *weight*)
Overrides: fipy.terms.faceTerm.FaceTerm.__init__

getCoeff(*self*)

Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.5 Module fipy.terms.explicitDiffusionTerm

7.5.1 Class ExplicitDiffusionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.diffusionTerm.DiffusionTerm

ExplicitDiffusionTerm

Methods

__init__(*self*, *diffCoeff*, *mesh*, *boundaryConditions*) Overrides: fipy.terms.diffusionTerm.DiffusionTerm.__init__

Inherited from DiffusionTerm: getCoeff
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.6 Module fipy.terms.explicitUpwindConvectionTerm

7.6.1 Class ExplicitUpwindConvectionTerm

fipy.terms.term.Term ┐

fipy.terms.faceTerm.FaceTerm ┐

fipy.terms.convectionTerm.ConvectionTerm ┐

fipy.terms.upwindConvectionTerm.UpwindConvectionTerm ┐

ExplicitUpwindConvectionTerm

Known Subclasses: VanLeerConvectionTerm

Methods

getWeight(*self*, *alpha*)
Overrides: fipy.terms.convectionTerm.ConvectionTerm.getWeight

Inherited from ConvectionTerm: __init__
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.7 Module fipy.terms.exponentialConvectionTerm

7.7.1 Class ExponentialConvectionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.convectionTerm.ConvectionTerm

ExponentialConvectionTerm

Methods

Inherited from ConvectionTerm: __init__, getWeight
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

Class Variables

Name	Description

7.8 Module fipy.terms.faceTerm

7.8.1 Class FaceTerm

fipy.terms.term.Term
└
FaceTerm

Known Subclasses: ConvectionTerm, DiffusionTerm

Methods

__init__(*self*, *weight*, *mesh*, *boundaryConditions*)
Overrides: fipy.terms.term.Term.__init__

buildMatrix(*self*, *oldArray*, *coeffScale*, *varScale*, *dt*)

Implicit portion considers

Overrides: fipy.terms.term.Term.buildMatrix

explicitBuildMatrix(*self*, *oldArray*, *id1*, *id2*, *b*, *coeffScale*, *varScale*)

getOldAdjacentValues(*self*, *oldArray*, *id1*, *id2*)

implicitBuildMatrix(*self*, *L*, *coeffScale*, *id1*, *id2*, *b*, *varScale*)

Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.9 Module fipy.terms.hybridConvectionTerm

7.9.1 Class HybridConvectionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.convectionTerm.ConvectionTerm

HybridConvectionTerm

Methods

Inherited from ConvectionTerm: __init__, getWeight
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

Class Variables

Name	Description

7.10 Module fipy.terms.implicitDiffusionTerm

7.10.1 Class ImplicitDiffusionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.diffusionTerm.DiffusionTerm

ImplicitDiffusionTerm

Methods

__**init**__(*self*, *diffCoeff*, *mesh*, *boundaryConditions*)
Overrides: fipy.terms.diffusionTerm.DiffusionTerm.__init__

Inherited from DiffusionTerm: getCoeff
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.11 Module fipy.terms.nthOrderDiffusionTerm

This term represents a higher order diffusion term. The order of the term is determined by the number of `coeffs`, such that:

`NthOrderDiffusionTerm(D1, mesh, bcs)`

represents a typical 2nd-order diffusion term of the form

$$\nabla \cdot (D_1 \nabla \phi)$$

and:

`NthOrderDiffusionTerm((D1,D2), mesh, bcs)`

represents a 4th-order Cahn-Hilliard term of the form

$$\nabla \cdot [D_1 \nabla \cdot (D_2 \nabla \phi)]$$

and so on.

Test, 2nd order, 1 dimension, fixed flux of zero both ends.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(1., 1., 2, 1)
>>> term = NthOrderDiffusionTerm((1,), mesh, ())
>>> print term.getCoefficientMatrix()
 1.000000  -1.000000
-1.000000   1.000000
>>> L,b = term.buildMatrix((0,0), 1., 1.)
>>> print L
 1.000000  -1.000000
-1.000000   1.000000
>>> print b
[ 0., 0.,]
```

Test, 2nd order, 1 dimension, fixed flux 3, fixed value of 4

```
>>> from fipy.boundaryConditions.fixedFlux import FixedFlux
>>> from fipy.boundaryConditions.fixedValue import FixedValue
>>> bcLeft = FixedFlux(mesh.getFacesLeft(), 3.)
>>> bcRight = FixedValue(mesh.getFacesRight(), 4.)
>>> term = NthOrderDiffusionTerm((1.,), mesh, (bcLeft, bcRight))
>>> print term.getCoefficientMatrix()
 1.000000  -1.000000
-1.000000   1.000000
>>> L,b = term.buildMatrix((0.,0.), 1., 1.)
>>> print L
 1.000000  -1.000000
-1.000000   3.000000
>>> print b
[ 3., 8.,]
```

Test, 4th order, 1 dimension, x = 0; fixed flux 3, fixed curvatures 0, x = 2, fixed value 1, fixed curvature 0

```
>>> bcLeft1 = FixedFlux(mesh.getFacesLeft(), 3.)
>>> from fipy.boundaryConditions.nthOrderBoundaryCondition import NthOrderBoundaryCondition
>>> bcLeft2 =  NthOrderBoundaryCondition(mesh.getFacesLeft(), 0., 2)
>>> bcRight1 = FixedValue(mesh.getFacesRight(), 4.)
>>> bcRight2 =  NthOrderBoundaryCondition(mesh.getFacesRight(), 0., 2)
>>> term = NthOrderDiffusionTerm((1., 1.), mesh, (bcLeft1, bcLeft2, bcRight1, bcRight2))
>>> print term.getCoefficientMatrix()
-1.000000   1.000000
 1.000000  -1.000000
>>> L,b = term.buildMatrix((0.,0.), 1., 1.)
>>> print L
-4.000000   6.000000
 4.000000  -10.000000
>>> print b
[ -1.,-21.,]
```

Test, 4th order, 1 dimension, x = 0; fixed flux 3, fixed curvature 2, x = 2, fixed value 4, fixed 3rd order -1

```
>>> bcLeft1 = FixedFlux(mesh.getFacesLeft(), 3.)
>>> from fipy.boundaryConditions.nthOrderBoundaryCondition import NthOrderBoundaryCondition
>>> bcLeft2 =  NthOrderBoundaryCondition(mesh.getFacesLeft(), 2., 2)
>>> bcRight1 = FixedValue(mesh.getFacesRight(), 4.)
>>> bcRight2 =  NthOrderBoundaryCondition(mesh.getFacesRight(), -1., 3)
>>> term = NthOrderDiffusionTerm((1., 1.), mesh, (bcLeft1, bcLeft2, bcRight1, bcRight2))
>>> print term.getCoefficientMatrix()
-1.000000   1.000000
 1.000000  -1.000000
>>> L,b = term.buildMatrix((0.,0.), 1., 1.)
>>> print L
-4.000000   6.000000
 2.000000  -4.000000
>>> print b
[ 3.,-4.,]
```

Test when dx = 0.5.

```
>>> mesh = Grid2D(dx = .5, dy = 1., nx = 2, ny = 1)
>>> bcLeft1 = FixedValue(mesh.getFacesLeft(), 0.)
>>> from fipy.boundaryConditions.nthOrderBoundaryCondition import NthOrderBoundaryCondition
>>> bcLeft2 =  NthOrderBoundaryCondition(mesh.getFacesLeft(), 1., 2)
>>> bcRight1 = FixedFlux(mesh.getFacesRight(), 1.)
>>> bcRight2 =  NthOrderBoundaryCondition(mesh.getFacesRight(), 0., 3)
>>> term = NthOrderDiffusionTerm((1., 1.), mesh, (bcLeft1, bcLeft2, bcRight1, bcRight2))
>>> print term.getCoefficientMatrix()
-2.000000   2.000000
```

```
 2.000000  -2.000000
>>> L,b = term.buildMatrix((0.,0.), 1., 1.)
>>> print L
-8.00e+01  32.000000
32.000000  -16.000000
>>> print b
[ 8.,-4.,]
```

Test when dx = 0.25.

```
>>> mesh = Grid2D(dx = .25, dy = 1., nx = 2, ny = 1)
>>> bcLeft1 = FixedValue(mesh.getFacesLeft(), 0.)
>>> from fipy.boundaryConditions.nthOrderBoundaryCondition import NthOrderBoundaryCondition
>>> bcLeft2 =  NthOrderBoundaryCondition(mesh.getFacesLeft(), 1., 2)
>>> bcRight1 = FixedFlux(mesh.getFacesRight(), 1.)
>>> bcRight2 =  NthOrderBoundaryCondition(mesh.getFacesRight(), 0., 3)
>>> term = NthOrderDiffusionTerm((1., 1.), mesh, (bcLeft1, bcLeft2, bcRight1, bcRight2))
>>> print term.getCoefficientMatrix()
-4.000000   4.000000
 4.000000  -4.000000
>>> L,b = term.buildMatrix((0.,0.), 1., 1.)
>>> print L
-6.40e+02   2.56e+02
 2.56e+02  -1.28e+02
>>> print b
[ 24.,-16.,]
```

7.11.1 Class NthOrderDiffusionTerm

fipy.terms.term.Term
└
NthOrderDiffusionTerm

Methods

__init__(*self*, *coeffs*, *mesh*, *boundaryConditions*)
Overrides: fipy.terms.term.Term.__init__

buildMatrix(*self*, *oldArray*, *coeffScale*, *varScale*, *dt*=`1.0`)
Overrides: fipy.terms.term.Term.buildMatrix

getCoefficientMatrix(*self*)

Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.12 Module fipy.terms.powerLawConvectionTerm

7.12.1 Class PowerLawConvectionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.convectionTerm.ConvectionTerm

PowerLawConvectionTerm

Methods

Inherited from ConvectionTerm: __init__, getWeight
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

Class Variables

Name	Description

7.13 Module fipy.terms.scSourceTerm

7.13.1 Class ScSourceTerm

fipy.terms.term.Term

fipy.terms.cellTerm.CellTerm

fipy.terms.sourceTerm.SourceTerm

ScSourceTerm

Sc source term. This term in general should be positive for stability. Added to the b vector.

Methods

__init__(*self*, *sourceCoeff*, *mesh*) Overrides: fipy.terms.sourceTerm.SourceTerm.__init__

Inherited from CellTerm: buildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.14 Module fipy.terms.sourceTerm

7.14.1 Class SourceTerm

fipy.terms.term.Term ┐

fipy.terms.cellTerm.CellTerm ┐

SourceTerm

Known Subclasses: ScSourceTerm, SpSourceTerm

Methods

__**init**__(*self*, *sourceCoeff*, *weight*, *mesh*) Overrides: fipy.terms.cellTerm.CellTerm.__init__

Inherited from CellTerm: buildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.15 Module fipy.terms.spSourceTerm

7.15.1 Class SpSourceTerm

fipy.terms.term.Term

fipy.terms.cellTerm.CellTerm

fipy.terms.sourceTerm.SourceTerm

SpSourceTerm

Sp source term. This term in general should be positive for stability. Added to the matrix diagonal.

Methods

__init__(*self*, *sourceCoeff*, *mesh*)
Overrides: fipy.terms.sourceTerm.SourceTerm.__init__

Inherited from CellTerm: buildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.16 Module fipy.terms.term

7.16.1 Class Term

Known Subclasses: AdvectionTerm, CellTerm, FaceTerm, NthOrderDiffusionTerm

Methods

__init__(*self*, *mesh*, *weight*)

buildMatrix(*self*, *oldArray*, *coeffScale*, *varScale*, *dt*)

calcCoeffScale(*self*)

getCoeffScale(*self*)

getFigureOfMerit(*self*)

getMesh(*self*)

7.17 Module fipy.terms.test

7.17.1 Functions

suite()

7.18 Module fipy.terms.transientTerm

7.18.1 Class TransientTerm

fipy.terms.term.Term

fipy.terms.cellTerm.CellTerm

TransientTerm

Methods

__init__(*self*, *tranCoeff*, *mesh*) Overrides: fipy.terms.cellTerm.CellTerm.__init__

Inherited from CellTerm: buildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

7.19 Module fipy.terms.upwindConvectionTerm

7.19.1 Class UpwindConvectionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.convectionTerm.ConvectionTerm

UpwindConvectionTerm

Known Subclasses: ExplicitUpwindConvectionTerm

Methods

Inherited from ConvectionTerm: __init__, getWeight
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, getOldAdjacentValues, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getFigureOfMerit, getMesh

Class Variables

Name	Description

7.20 Module fipy.terms.vanLeerConvectionTerm

7.20.1 Class VanLeerConvectionTerm

fipy.terms.term.Term

fipy.terms.faceTerm.FaceTerm

fipy.terms.convectionTerm.ConvectionTerm

fipy.terms.upwindConvectionTerm.UpwindConvectionTerm

fipy.terms.explicitUpwindConvectionTerm.ExplicitUpwindConvectionTerm

VanLeerConvectionTerm

Methods

getFigureOfMerit(*self*)
Overrides: fipy.terms.term.Term.getFigureOfMerit

getGradient(*self*, *normalGradient*, *gradUpwind*)

getOldAdjacentValues(*self*, *oldArray*, *id1*, *id2*)
Overrides: fipy.terms.faceTerm.FaceTerm.getOldAdjacentValues

Inherited from ConvectionTerm: __init__
Inherited from ExplicitUpwindConvectionTerm: getWeight
Inherited from FaceTerm: buildMatrix, explicitBuildMatrix, implicitBuildMatrix
Inherited from Term: calcCoeffScale, getCoeffScale, getMesh

Chapter 8

Module fipy.test

8.1 Module fipy.test

8.1.1 Functions

suite()

Chapter 9

Module fipy.tests

9.1 Module fipy.tests.blinky

9.1.1 Variables

Name	Description
bottomLeft	**Value:** [<fipy.meshes.numMesh.cell.Cell instance at 0x15ebd00>, <fipy.meshes.numMesh.... *(type=list)*
bottomRight	**Value:** [<fipy.meshes.numMesh.cell.Cell instance at 0x15f1940>, <fipy.meshes.numMesh.... *(type=list)*
dx	**Value:** 0.01 *(type=float)*
L	**Value:** 0.20000000000000001 *(type=float)*
mesh	**Value:** <fipy.meshes.numMesh.grid2D.Grid2D instance at 0x151e800> *(type=Grid2D)*
nx	**Value:** 20 *(type=int)*
topLeft	**Value:** [<fipy.meshes.numMesh.cell.Cell instance at 0x16799e0>, <fipy.meshes.numMesh.... *(type=list)*
topRight	**Value:** [<fipy.meshes.numMesh.cell.Cell instance at 0x16420f8>, <fipy.meshes.numMesh.... *(type=list)*
var	**Value:** CellVariable(name = "test", value = [2.50000000e-05, 7.50000000e-05, 1.25... *(type=CellVariable)*

9.2 Module fipy.tests.doctestPlus

9.2.1 Functions

getScript(*name*='__main__')

9.3 Module fipy.tests.testBase

9.3.1 Class TestBase

__builtin__.object
  unittest.TestCase
    TestBase

Known Subclasses: TestElPhF, TestLaplacian, TestMean, TestMeshBase, TestVariablePickle

Methods

assertArrayEqual(*self*, *first*, *second*, *msg*=`None`)

Fail if the two objects are unequal by more than tol.

assertArrayWithinTolerance(*self*, *first*, *second*, *atol*=`1e-10`, *rtol*=`1e-10`, *msg*=`None`)

Fail if the two objects are unequal by more than tol.

assertWithinTolerance(*self*, *first*, *second*, *tol*=`1e-10`, *msg*=`None`)

Fail if the two objects are unequal by more than tol.

getTestValue(*self*, *cell*)

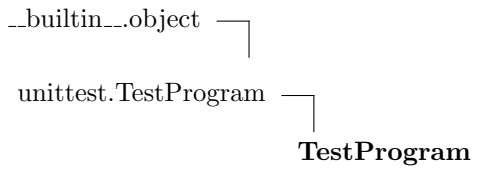
getTestValues(*self*)

testResult(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, setUp, shortDescription, tearDown

9.4 Module fipy.tests.testProgram

9.4.1 Class TestProgram



Methods

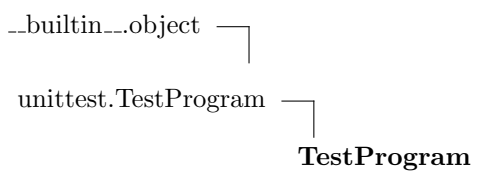
parseArgs(*self*, *argv*)
Overrides: unittest.TestProgram.parseArgs

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __repr__, __setattr__, __str__
Inherited from TestProgram: __init__, createTests, runTests, usageExit

Class Variables

Name	Description
Inherited from TestProgram: USAGE *(p. ??)*	

9.4.2 Class TestProgram



Methods

parseArgs(*self*, *argv*)
Overrides: unittest.TestProgram.parseArgs

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __repr__, __setattr__, __str__
Inherited from TestProgram: __init__, createTests, runTests, usageExit

Class Variables

Name	Description
Inherited from TestProgram: USAGE *(p. ??)*	

Chapter 10

Module fipy.tools

10.1 Module fipy.tools.array

10.1.1 Functions

allclose(*first*, *second*, *atol*=`1.0000000000000001e-05`, *rtol*=`1e-08`)

allequal(*first*, *second*)

arctan(*arr*)

arctan2(*arr*, *other*)

crossProd(*v1*, *v2*)

Return vector cross-product of v1 and v2.

dot(*a1*, *a2*, *axis*=`1`)

return array of vector dot-products of v1 and v2 for arrays a1 and a2 of vectors v1 and v2
We can't use Numeric.dot on an array of vectors

put(*arr*, *ids*, *values*)

reshape(*arr*, *shape*)

sqrt(*arr*)

sqrtDot(*a1*, *a2*)

Return array of square roots of vector dot-products for arrays a1 and a2 of vectors v1 and v2
Usually used with v1==v2 to return magnitude of v1.

sum(*arr*, *index*=0)

take(*arr*, *ids*)

tan(*arr*)

10.2 Module fipy.tools.dimensions.DictWithDefault

10.2.1 Class DictWithDefault

UserDict.UserDict
└
DictWithDefault

Known Subclasses: NumberDict

Dictionary with default values

Constructor: DictWithDefault(|default|)

Instances of this class act like standard Python dictionaries, except that they return a *copy* of |default| for a key that has no associated value.

Methods

__init__(*self*, *default*)
Overrides: UserDict.UserDict.__init__

__delitem__(*self*, *key*)
Overrides: UserDict.UserDict.__delitem__

__getitem__(*self*, *key*)
Overrides: UserDict.UserDict.__getitem__

Inherited from UserDict: __cmp__, __contains__, __len__, __repr__, __setitem__, clear, copy, get, has_key, items, iteritems, iterkeys, itervalues, keys, pop, popitem, setdefault, update, values

Class Methods

Inherited from UserDict: fromkeys

10.3 Module fipy.tools.dimensions.NumberDict

10.3.1 Class NumberDict

UserDict.UserDict

fipy.tools.dimensions.DictWithDefault.DictWithDefault

NumberDict

Dictionary storing numerical values

Constructor: NumberDict()

An instance of this class acts like an array of number with generalized (non-integer) indices. A value of zero is assumed for undefined entries. NumberDict instances support addition, and subtraction with other NumberDict instances, and multiplication and division by scalars.

Methods

__init__(*self*)
Overrides: fipy.tools.dimensions.DictWithDefault.DictWithDefault.__init__

__add__(*self*, *other*)

__coerce__(*self*, *other*)

__div__(*self*, *other*)

__mul__(*self*, *other*)

__radd__(*self*, *other*)

__repr__(*self*)
Overrides: UserDict.UserDict.__repr__

__rmul__(*self*, *other*)

__rsub__(*self*, *other*)

__str__(*self*)

__sub__(*self*, *other*)

Inherited from UserDict: __cmp__, __contains__, __len__, __setitem__, clear, copy, get, has_key, items, iteritems, iterkeys, itervalues, keys, pop, popitem, setdefault, update, values
Inherited from DictWithDefault: __delitem__, __getitem__

Class Methods

Inherited from UserDict: fromkeys

10.4 Module fipy.tools.dimensions.physicalField

Physical quantities with units.

This module provides a data type that represents a physical quantity together with its unit. It is possible to add and subtract these quantities if the units are compatible, and a quantity can be converted to another compatible unit. Multiplication, subtraction, and raising to integer powers is allowed without restriction, and the result will have the correct unit. A quantity can be raised to a non-integer power only if the result can be represented by integer powers of the base units.

The values of physical constants are taken from the 2002 recommended values from CODATA. Other conversion factors (e.g. for British units) come from Appendix B of NIST Special Publication 811.

Warning

We can't guarantee for the correctness of all entries in the unit table, so use this at your own risk!

10.4.1 Functions

Scale(*quantity*, *scaling*)

Normalize `quantity` by `scaling`.
`quantity` can be a `PhysicalField`

```
>>> print round(Scale(PhysicalField("1. inch"), PhysicalField("1. mm")), 6)
25.4
```

or a value-unit string convertable to a `PhysicalField`

```
>>> print round(Scale("1. inch", PhysicalField("1. mm")), 6)
25.4
```

or a dimensionless number. A dimensionless number is left alone.

```
>>> print round(Scale(PhysicalField(2.), PhysicalField("1. mm")), 6)
2.0
```

It is an error for the result to have dimensions.

```
>>> print Scale(PhysicalField("1. s"), PhysicalField("1. mm"))
Traceback (most recent call last):
    ...
TypeError: <PhysicalUnit s> and <PhysicalUnit m> are incompatible
```

10.4.2 Class PhysicalField

Known Subclasses: ModPhysicalField

Physical field or quantity with units

`PhysicalField` instances allow addition, subtraction, multiplication, and division with each other as well as multiplication, division, and exponentiation with numbers. Addition and subtraction

check that the units of the two operands are compatible and return the result in the units of the first operand. A limited set of mathematical functions (from module `Numeric`) is applicable as well:

sqrt equivalent to exponentiation with 0.5.

sin, cos, tan applicable only to objects whose unit is compatible with `rad`.

Methods

__init__(*self*, *value*, *unit*=`None`, *array*=`None`)

Physical Fields can be constructed in one of two ways:

- `PhysicalField(*value*, *unit*)`, where `*value*` is a number of arbitrary type and `*unit*` is a string containing the unit name
```
>>> print PhysicalField(value = 10., unit = 'm')
10.0 m
```
- `PhysicalField(*string*)`, where `*string*` contains both the value and the unit. This form is provided to make interactive use more convenient
```
>>> print PhysicalField(value = "10. m")
10.0 m
```

Dimensionless quantities, with a `unit` of 1, can be specified in several ways
```
>>> print PhysicalField(value = "1")
1.0 1
>>> print PhysicalField(value = 2., unit = " ")
2.0 1
>>> print PhysicalField(value = 2.)
2.0 1
```
Physical arrays are also possible (and are the reason this code was adapted from Konrad Hinsen's original PhysicalQuantity). The `value` can be a Numeric `array`:
```
>>> a = Numeric.array(((3.,4.),(5.,6.)))
>>> print PhysicalField(value = a, unit = "m")
[[ 3., 4.,]
 [ 5., 6.,]] m
```
or a `tuple`:
```
>>> print PhysicalField(value = ((3.,4.),(5.,6.)), unit = "m")
[[ 3., 4.,]
 [ 5., 6.,]] m
```
or as a single value to be applied to every element of a supplied array:
```
>>> print PhysicalField(value = 2., unit = "m", array = a)
[[ 2., 2.,]
 [ 2., 2.,]] m
```
Every element in an array has the same unit, which is stored only once for the whole array.

__abs__(*self*)

Return the absolute value of the quantity. The `unit` is unchanged.

```
>>> print abs(PhysicalField(((3.,-2.),(-1.,4.)), 'm'))
[[ 3., 2.,]
 [ 1., 4.,]] m
```

__add__(*self*, *other*)

Add two physical quantities, so long as their units are compatible. The unit of the result is the unit of the first operand.

```
>>> print PhysicalField(10., 'km') + PhysicalField(10., 'm')
10.01 km
>>> print PhysicalField(10., 'km') + PhysicalField(10., 'J')
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

__array__(*self*, *t*=`None`)

Return a dimensionless `PhysicalField` as a Numeric `array`.

```
>>> Numeric.array(PhysicalField(((2.,3.),(4.,5.)),"m/m"))
[[ 2., 3.,]
 [ 4., 5.,]]
```

As a special case, fields with angular units are converted to base units (radians) and then assumed dimensionless.

```
>>> Numeric.array(PhysicalField(((2.,3.),(4.,5.)),"deg"))
[[ 0.03490659, 0.05235988,]
 [ 0.06981317, 0.08726646,]]
```

If the array is not dimensionless, the conversion fails.

```
>>> Numeric.array(PhysicalField(((2.,3.),(4.,5.)),"m"))
Traceback (most recent call last):
    ...
TypeError: Numeric array value must be dimensionless
```

__div__(*self*, *other*)

Divide two physical quantities. The unit of the result is the unit of the first operand divided by the unit of the second.

```
>>> print PhysicalField(10., 'm') / PhysicalField(2., 's')
5.0 m/s
```

As a special case, if the result is dimensionless, the value is returned without units, rather than with a dimensionless unit of `1`. This facilitates passing physical quantities to packages such as Numeric that cannot use units, while ensuring the quantities have the desired units

```
>>> print Numeric.array(PhysicalField(1., 'inch') / PhysicalField(1., 'mm'))
25.4
>>> print Numeric.array(PhysicalField(1., 'inch') / PhysicalField(1., 'kg'))
Traceback (most recent call last):
    ...
TypeError: Numeric array value must be dimensionless
```

__eq__(*self*, *other*)

__float__(*self*)

Return a dimensionless PhysicalField quantity as a float.

```
>>> float(PhysicalField("2. m/m"))
2.0
```

As a special case, quantities with angular units are converted to base units (radians) and then assumed dimensionless.

```
>>> print round(float(PhysicalField("2. deg")), 6)
0.034907
```

If the quantity is not dimensionless, the conversion fails.

```
>>> float(PhysicalField("2. m"))
Traceback (most recent call last):
    ...
TypeError: Not possible to convert a PhysicalField with dimensions to float
```

Just as a Numeric `array` cannot be cast to float, neither can PhysicalField arrays

```
>>> float(PhysicalField(((2.,3.),(4.,5.)),"m/m"))
Traceback (most recent call last):
    ...
TypeError: only rank-0 arrays can be converted to Python scalars.
```

__ge__(*self*, *other*)

__getitem__(*self*, *index*)

Return the specified element of the array. The unit of the result will be the unit of the array.

```
>>> a = PhysicalField(((3.,4.),(5.,6.)),"m")
>>> print a[1,1]
6.0 m
```

__gt__(*self*, *other*)

Compare `self` to `other`, returning an array of boolean values corresponding to the test against each element.

```
>>> a = PhysicalField(((3.,4.),(5.,6.)),"m")
>>> print a > PhysicalField("13 ft")
[[0,1,]
 [1,1,]]
```

Appropriately formatted dimensional quantity strings can also be compared.

```
>>> print a > "13 ft"
[[0,1,]
 [1,1,]]
```

Arrays are compared element to element

```
>>> print a > PhysicalField(((3.,13.),(17.,6.)),"ft")
[[1,1,]
 [0,1,]]
```

Units must be compatible

```
>>> print a > PhysicalField("1 lb")
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

And so must array dimensions

```
>>> print a > PhysicalField(((3.,13.,4.),(17.,6.,2.)),"ft")
Traceback (most recent call last):
    ...
ValueError: frames are not aligned
```

__le__(*self*, *other*)

__len__(*self*)

__lt__(*self*, *other*)

__mod__(*self*, *other*)

Return the remainder of dividing two physical quantities. The unit of the result is the unit of the first operand divided by the unit of the second.

```
>>> print PhysicalField(11., 'm') % PhysicalField(2., 's')
1.0 m/s
```

__mul__(*self*, *other*)

Multiply two physical quantities. The unit of the result is the product of the units of the operands.

```
>>> print PhysicalField(10., 'N') * PhysicalField(10., 'm')
100.0 m*N
```

As a special case, if the result is dimensionless, the value is returned without units, rather than with a dimensionless unit of `1`. This facilitates passing physical quantities to packages such as Numeric that cannot use units, while ensuring the quantities have the desired units.

```
>>> print Numeric.array(PhysicalField(10., 's') * PhysicalField(2., 'Hz'))
20.0
>>> print Numeric.array(PhysicalField(10., 's') * PhysicalField(2., 's'))
Traceback (most recent call last):
    ...
TypeError: Numeric array value must be dimensionless
```

__ne__(*self*, *other*)

__neg__(*self*)

Return the negative of the quantity. The `unit` is unchanged.

```
>>> print -PhysicalField(((3.,-2.),(-1.,4.)), 'm')
[[-3., 2.,]
 [ 1.,-4.,]] m
```

__nonzero__(*self*)

Test if the quantity is zero.
Should this only pass if the unit offset is zero?

__pos__(*self*)

__pow__(*self*, *other*)

Raise a `PhysicalField` to a power. The unit is raised to the same power.

```
>>> print PhysicalField(10., 'm')**2
100.0 m**2
```

__radd__(*self*, *other*)

Add two physical quantities, so long as their units are compatible. The unit of the result is the unit of the first operand.

```
>>> print PhysicalField(10., 'km') + PhysicalField(10., 'm')
10.01 km
>>> print PhysicalField(10., 'km') + PhysicalField(10., 'J')
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

__rdiv__(*self*, *other*)

__repr__(*self*)

Return representation of a physical quantity suitable for re-use

```
>>> PhysicalField(value = 3., unit = "eV")
PhysicalField(3.0,'eV')
```

__rmul__(*self*, *other*)

Multiply two physical quantities. The unit of the result is the product of the units of the operands.

```
>>> print PhysicalField(10., 'N') * PhysicalField(10., 'm')
100.0 m*N
```

As a special case, if the result is dimensionless, the value is returned without units, rather than with a dimensionless unit of `1`. This facilitates passing physical quantities to packages such as Numeric that cannot use units, while ensuring the quantities have the desired units.

```
>>> print Numeric.array(PhysicalField(10., 's') * PhysicalField(2., 'Hz'))
20.0
>>> print Numeric.array(PhysicalField(10., 's') * PhysicalField(2., 's'))
Traceback (most recent call last):
    ...
TypeError: Numeric array value must be dimensionless
```

__rpow__(*self*, *other*)

__rsub__(*self*, *other*)

__setitem__(*self*, *index*, *value*)

Assign the specified element of the array, performing apropriate conversions.

```
>>> a = PhysicalField(((3.,4.),(5.,6.)),"m")
>>> a[0,1] = PhysicalField("6 ft")
>>> print a
[[ 3.    , 1.8288,]
 [ 5.    , 6.    ,]] m
>>> a[1,0] = PhysicalField("2 min")
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

__str__(*self*)

Return human-readable form of a physical quantity

```
>>> print PhysicalField(value = 3., unit = "eV")
3.0 eV
```

__sub__(*self*, *other*)

Subtract two physical quantities, so long as their units are compatible. The unit of the result is the unit of the first operand.

```
>>> print PhysicalField(10., 'km') - PhysicalField(10., 'm')
9.99 km
>>> print PhysicalField(10., 'km') - PhysicalField(10., 'J')
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

allclose(*self*, *other*, *atol*=`None`, *rtol*=`1e-08`)

This function tests whether or not `self` and `other` are equal subject to the given relative and absolute tolerances. The formula used is

$$|self - other| < atol + rtol|other|$$

This means essentially that both elements are small compared to `atol` or their difference divided by `other`'s value is small compared to `rtol`.

arctan(*self*)

Return the arctangent of the `PhysicalField` in radians

```
>>> print round(PhysicalField(1).arctan(), 6)
0.785398
```

The input `PhysicalField` must be dimensionless

```
>>> print round(PhysicalField("1 m").arctan(), 6)
Traceback (most recent call last):
    ...
TypeError: Numeric array value must be dimensionless
```

arctan2(*self*, *other*)

Return the arctangent of `self` divided by `other` in radians

```
>>> print round(PhysicalField(2.).arctan2(PhysicalField(5.)), 6)
0.380506
```

The input `PhysicalField` objects must be dimensionless

```
>>> print round(PhysicalField(2.).arctan2(PhysicalField("5. m")), 6)
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

convertToUnit(*self*, *unit*)

Changes the unit to `unit` and adjusts the value such that the combination is equivalent. The new unit is by a string containing its name. The new unit must be compatible with the previous unit of the object.

```
>>> e = PhysicalField('2.7 Hartree*Nav')
>>> e.convertToUnit('kcal/mol')
>>> print e
1694.27557621 kcal/mol
```

copy(*self*)

cos(*self*)

Return the cosine of the `PhysicalField`

```
>>> print round(PhysicalField(2*Numeric.pi/6,"rad").cos(), 6)
0.5
>>> print round(PhysicalField(60.,"deg").cos(), 6)
0.5
```

The units of the `PhysicalField` must be an angle

```
>>> PhysicalField(60.,"m").cos()
Traceback (most recent call last):
    ...
TypeError: Argument of cos must be an angle
```

dot(*self*, *other*)

Return the dot product of `self` with `other`. The resulting unit is the product of the units of `self` and `other`.

```
>>> v = PhysicalField(((5.,6.),(7.,8.)), "m")
>>> print PhysicalField(((1.,2.),(3.,4.)), "m").dot(v)
[[ 19., 22.,]
 [ 43., 50.,]] m**2
```

getNumericValue(*self*)

Return the `PhysicalField` without units, after conversion to base SI units.

```
>>> print round(PhysicalField("1 inch").getNumericValue(), 6)
0.0254
```

getUnit(*self*)

Return the unit object of `self`.

```
>>> PhysicalField("1 m").getUnit()
<PhysicalUnit m>
```

inBaseUnits(*self*)

Return the quantity with all units reduced to their base SI elements.

```
>>> e = PhysicalField('2.7 Hartree*Nav')
>>> print e.inBaseUnits()
7088849.01085 kg*m**2/s**2/mol
```

inUnitsOf(*self*, **units*)

Returns one or more `PhysicalField` objects that express the same physical quantity in different units. The units are specified by strings containing their names. The units must be compatible with the unit of the object. If one unit is specified, the return value is a single `PhysicalField`.

```
>>> freeze = PhysicalField('0 degC')
>>> print freeze.inUnitsOf('degF')
32.0 degF
```

If several units are specified, the return value is a tuple of `PhysicalField` instances with with one element per unit such that the sum of all quantities in the tuple equals the the original quantity and all the values except for the last one are integers. This is used to convert to irregular unit systems like hour/minute/second. The original object will not be changed.

```
>>> t = PhysicalField(314159., 's')
>>> [str(element) for element in t.inUnitsOf('d','h','min','s')]
['3.0 d', '15.0 h', '15.0 min', '59.0 s']
```

isCompatible(*self*, *unit*)

put(*self*, *indices*, *values*)

`put` is the opposite of `take`. The values of `self` at the locations specified in `indices` are set to the corresponding value of `values`.
The `indices` can be any integer sequence object with values suitable for indexing into the flat form of `self`. The `values` must be any sequence of values that can be converted to the typecode of `self`.

```
>>> f = PhysicalField((1.,2.,3.),"m")
>>> f.put((2,0), PhysicalField((2.,3.),"inch"))
>>> print f
[ 0.0762, 2.    , 0.0508,] m
```

The units of `values` must be compatible with `self`.

```
>>> f.put(1, PhysicalField(3,"kg"))
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

reshape(*self*, *shape*)

Changes the shape of `self` to that specified in `shape`

```
>>> print PhysicalField((1.,2.,3.,4.),"m").reshape((2,2))
[[ 1., 2.,]
 [ 3., 4.,]] m
```

The new shape must have the same size as the existing one.

```
>>> print PhysicalField((1.,2.,3.,4.),"m").reshape((2,3))
Traceback (most recent call last):
    ...
ValueError: total size of new array must be unchanged
```

sin(*self*)

Return the sine of the `PhysicalField`

```
>>> print PhysicalField(Numeric.pi/6,"rad").sin()
0.5
>>> print PhysicalField(30.,"deg").sin()
0.5
```

The units of the PhysicalField must be an angle

```
>>> PhysicalField(30.,"m").sin()
Traceback (most recent call last):
    ...
TypeError: Argument of sin must be an angle
```

sqrt(*self*)

Return the square root of the `PhysicalField`

```
>>> print PhysicalField("100. m**2").sqrt()
10.0 m
```

The resulting unit must be integral

```
>>> print PhysicalField("100. m").sqrt()
Traceback (most recent call last):
    ...
TypeError: Illegal exponent
```

sum(*self*, *index*=0)

Returns the sum of all of the elements in `self` along the specified axis (first axis by default).

```
>>> print PhysicalField(((1.,2.),(3.,4.)), "m").sum()
[ 4., 6.,] m
>>> print PhysicalField(((1.,2.),(3.,4.)), "m").sum(1)
[ 3., 7.,] m
```

take(*self*, *indices*, *axis*=0)

Return the elements of `self` specified by the elements of `indices`. The resulting `PhysicalField` array has the same units as the original.

```
>>> print PhysicalField((1.,2.,3.),"m").take((2,0))
[ 3., 1.,] m
```

The optional third argument specifies the axis along which the selection occurs, and the default value (as in the example above) is 0, the first axis.

```
>>> print PhysicalField(((1.,2.,3.),(4.,5.,6.)),"m").take((2,0), axis = 1)
[[ 3., 1.,]
 [ 6., 4.,]] m
```

tan(*self*)

Return the tangent of the `PhysicalField`

```
>>> round(PhysicalField(Numeric.pi/4,"rad").tan(), 6)
1.0
>>> round(PhysicalField(45,"deg").tan(), 6)
1.0
```

The units of the `PhysicalField` must be an angle

```
>>> PhysicalField(45.,"m").tan()
Traceback (most recent call last):
    ...
TypeError: Argument of tan must be an angle
```

10.4.3 Class PhysicalUnit

A `PhysicalUnit` represents the units of a `PhysicalField`.

Methods

__init__(*self*, *names*, *factor*, *powers*, *offset*=0)

This class is not generally not instantiated by users of this module, but rather it is created in the process of constructing a `PhysicalField`.

Parameters

- `names:` the name of the unit
- `factor:` the multiplier between the unit and the fundamental SI unit
- `powers:` a nine-element `list`, `tuple`, or Numeric `array` representing the fundamental SI units of [“m”, “kg”, “s”, “A”, “K”, “mol”, “cd”, “rad”, “sr”]
- `offset:` the displacement between the zero-point of the unit and the zero-point of the corresponding fundamental SI unit.

__cmp__(*self*, *other*)

Determine if units are identical

```
>>> a = PhysicalField("1. m")
>>> b = PhysicalField("3. ft")
>>> a.getUnit() == b.getUnit()
0
>>> a.getUnit() == b.inBaseUnits().getUnit()
1
```

Units can only be compared with other units

```
>>> a.getUnit() == 3
Traceback (most recent call last):
    ...
TypeError: PhysicalUnits can only be compared with other PhysicalUnits
```

__div__(*self*, *other*)

Divide one unit by another

```
>>> a = PhysicalField("1. m")
>>> b = PhysicalField("3. ft")
>>> a.getUnit() / b.getUnit()
<PhysicalUnit m/ft>
>>> a.getUnit() / b.inBaseUnits().getUnit()
<PhysicalUnit 1>
>>> c = PhysicalField("1. s")
>>> d = PhysicalField("3. Hz")
>>> c.getUnit() / d.getUnit()
<PhysicalUnit s/Hz>
>>> c.getUnit() / d.inBaseUnits().getUnit()
<PhysicalUnit s**2/1>
```

or divide units by numbers

```
>>> a.getUnit() / 3.
<PhysicalUnit m/3.0>
```

Units must have zero offset to be divided

```
>>> e = PhysicalField("1. J")
>>> f = PhysicalField("25. degC")
>>> e.getUnit() / f.getUnit()
Traceback (most recent call last):
    ...
TypeError: cannot divide units with non-zero offset
>>> e.getUnit() / f.inBaseUnits().getUnit()
<PhysicalUnit J/K>
```

__mul__(*self*, *other*)

Multiply units together

```
>>> a = PhysicalField("1. m")
>>> b = PhysicalField("3. ft")
>>> a.getUnit() * b.getUnit()
<PhysicalUnit ft*m>
>>> a.getUnit() * b.inBaseUnits().getUnit()
<PhysicalUnit m**2>
>>> c = PhysicalField("1. s")
>>> d = PhysicalField("3. Hz")
>>> c.getUnit() * d.getUnit()
<PhysicalUnit Hz*s>
>>> c.getUnit() * d.inBaseUnits().getUnit()
<PhysicalUnit 1>
```

or multiply units by numbers

```
>>> a.getUnit() * 3.
<PhysicalUnit m*3.0>
```

Units must have zero offset to be multiplied

```
>>> e = PhysicalField("1. kB")
>>> f = PhysicalField("25. degC")
>>> e.getUnit() * f.getUnit()
Traceback (most recent call last):
    ...
TypeError: cannot multiply units with non-zero offset
>>> e.getUnit() * f.inBaseUnits().getUnit()
<PhysicalUnit kB*K>
```

__pow__(*self*, *other*)

Raise a unit to an integer power

```
>>> a = PhysicalField("1. m")
>>> a.getUnit()**2
<PhysicalUnit m**2>
>>> a.getUnit()**-2
<PhysicalUnit 1/m**2>
```

Non-integer powers are not supported

```
>>> a.getUnit()**0.5
Traceback (most recent call last):
    ...
TypeError: Illegal exponent
```

Units must have zero offset to be exponentiated

```
>>> b = PhysicalField("25. degC")
>>> b.getUnit()**2
Traceback (most recent call last):
    ...
TypeError: cannot exponentiate units with non-zero offset
>>> b.inBaseUnits().getUnit()**2
<PhysicalUnit K**2>
```

__rdiv__(*self*, *other*)

Divide something by a unit

```
>>> a = PhysicalField("1. m")
>>> 3. / a.getUnit()
<PhysicalUnit 3.0/m>
```

Units must have zero offset to be divided

```
>>> b = PhysicalField("25. degC")
>>> 3. / b.getUnit()
Traceback (most recent call last):
    ...
TypeError: cannot divide units with non-zero offset
>>> 3. / b.inBaseUnits().getUnit()
<PhysicalUnit 3.0/K>
```

__repr__(*self*)

Return representation of a physical unit

```
>>> PhysicalUnit('m',   1.,    [1,0,0,0,0,0,0,0,0])
<PhysicalUnit m>
```

__rmul__(*self*, *other*)

Multiply units together

```
>>> a = PhysicalField("1. m")
>>> b = PhysicalField("3. ft")
>>> a.getUnit() * b.getUnit()
<PhysicalUnit ft*m>
>>> a.getUnit() * b.inBaseUnits().getUnit()
<PhysicalUnit m**2>
>>> c = PhysicalField("1. s")
>>> d = PhysicalField("3. Hz")
>>> c.getUnit() * d.getUnit()
<PhysicalUnit Hz*s>
>>> c.getUnit() * d.inBaseUnits().getUnit()
<PhysicalUnit 1>
```

or multiply units by numbers

```
>>> a.getUnit() * 3.
<PhysicalUnit m*3.0>
```

Units must have zero offset to be multiplied

```
>>> e = PhysicalField("1. kB")
>>> f = PhysicalField("25. degC")
>>> e.getUnit() * f.getUnit()
Traceback (most recent call last):
    ...
TypeError: cannot multiply units with non-zero offset
>>> e.getUnit() * f.inBaseUnits().getUnit()
<PhysicalUnit kB*K>
```

__str__(*self*)

Return representation of a physical unit

```
>>> PhysicalUnit('m',   1.,    [1,0,0,0,0,0,0,0,0])
<PhysicalUnit m>
```

conversionFactorTo(*self*, *other*)

Return the multiplication factor between two physical units

```
>>> a = PhysicalField("1. mm")
>>> b = PhysicalField("1. inch")
>>> print round(b.getUnit().conversionFactorTo(a.getUnit()), 6)
25.4
```

Units must have the same fundamental SI units

```
>>> c = PhysicalField("1. K")
>>> c.getUnit().conversionFactorTo(a.getUnit())
Traceback (most recent call last):
    ...
TypeError: Incompatible units
```

If units have different offsets, they must have the same factor

```
>>> d = PhysicalField("1. degC")
>>> c.getUnit().conversionFactorTo(d.getUnit())
1.0
>>> e = PhysicalField("1. degF")
>>> c.getUnit().conversionFactorTo(e.getUnit())
Traceback (most recent call last):
    ...
TypeError: Unit conversion (K to degF) cannot be expressed as a simple multiplicative factor
```

conversionTupleTo(*self*, *other*)

Return a `tuple` of the multiplication factor and offset between two physical units

```
>>> a = PhysicalField("1. K").getUnit()
>>> b = PhysicalField("1. degF").getUnit()
>>> print tuple([str(round(element,6)) for element in b.conversionTupleTo(a)])
('0.555556', '459.67')
```

isAngle(*self*)

Returns `True` if the unit is an angle

```
>>> PhysicalField("1. deg").getUnit().isAngle()
1
>>> PhysicalField("1. rad").getUnit().isAngle()
1
>>> PhysicalField("1. inch").getUnit().isAngle()
0
```

isCompatible(*self*, *other*)

Returns a list of which fundamental SI units are compatible between `self` and `other`

```
>>> a = PhysicalField("1. mm")
>>> b = PhysicalField("1. inch")
>>> a.getUnit().isCompatible(b.getUnit())
[1,1,1,1,1,1,1,1,1,]
>>> c = PhysicalField("1. K")
>>> a.getUnit().isCompatible(c.getUnit())
[0,1,1,1,0,1,1,1,1,]
```

isDimensionless(*self*)

Returns `True` if the unit is dimensionless

```
>>> PhysicalField("1. m/m").getUnit().isDimensionless()
1
>>> PhysicalField("1. inch").getUnit().isDimensionless()
0
```

isDimensionlessOrAngle(*self*)

Returns `True` if the unit is dimensionless or an angle

```
>>> PhysicalField("1. m/m").getUnit().isDimensionlessOrAngle()
1
>>> PhysicalField("1. deg").getUnit().isDimensionlessOrAngle()
1
>>> PhysicalField("1. rad").getUnit().isDimensionlessOrAngle()
1
>>> PhysicalField("1. inch").getUnit().isDimensionlessOrAngle()
0
```

name(*self*)

Return the name of the unit

```
>>> PhysicalField("1. m").getUnit().name()
'm'
>>> (PhysicalField("1. m")/PhysicalField("1. s")/PhysicalField("1. s")).getUnit().name()
'm/s**2'
```

setName(*self*, *name*)

Set the name of the unit to `name`

```
>>> a = PhysicalField("1. m/s").getUnit()
>>> a
<PhysicalUnit m/s>
>>> a.setName('meterpersecond')
>>> a
<PhysicalUnit meterpersecond>
```

10.5 Module fipy.tools.dump

10.5.1 Functions

read(*fileName*=`None`)

write(*data*, *fileName*)

10.6 Module fipy.tools.inline.inline

10.6.1 Functions

optionalInline(*inlineFn*, *pythonFn*, **args*)

runInline(*code*, ***args*)

runInlineLoop(*code_in*, ***args*)

runInlineLoop1(*code_in*, ***args*)

runInlineLoop2(*code_in*, ***args*)

runInlineLoop3(*code_in*, ***args*)

10.7 Module fipy.tools.memoryLeak

This python script is ripped from http://www.nightmare.com/medusa/memory-leaks.html

It outputs the top 100 number of outstanding references for each object.

10.7.1 Functions

get_refcounts()

print_top_N(n=100)

10.8 Module fipy.tools.sparseMatrix

10.8.1 Class SparseIdentityMatrix

fipy.tools.sparseMatrix.SparseMatrix
└
SparseIdentityMatrix

Represents a sparse identity matrix.

Methods

__init__(*self*, *size*)

Create a sparse matrix with '1' in the diagonal

```
>>> print SparseIdentityMatrix(size = 3)
 1.000000      ---        ---
    ---     1.000000      ---
    ---        ---     1.000000
```

Overrides: fipy.tools.sparseMatrix.SparseMatrix.__init__

Inherited from SparseMatrix: __add__, __array__, __getitem__, __mul__, __neg__, __pos__, __repr__, __rmul__, __setitem__, __str__, __sub__, addAt, addAtDiagonal, copy, getMatrix, getShape, put, putDiagonal, take, takeDiagonal, transpose

10.8.2 Class SparseMatrix

Known Subclasses: SparseIdentityMatrix

SparseMatrix class wrapper for pysparse. SparseMatrix is always NxN Allowing basic python operations __add__, __sub__ etc. Facilitate matrix populating in an easy way An example subcalls will be the identity matrix or the empty matrix.

Methods

__init__(*self*, *size*=`None`, *bandwidth*=`0`, *matrix*=`None`)

__add__(*self*, *other*)

Add two sparse matrices

```
>>> L = SparseMatrix(size = 3)
>>> L.put((3.,10.,Numeric.pi,2.5), (0,0,1,2), (2,1,1,0))
>>> print L + SparseIdentityMatrix(3)
 1.000000  10.000000   3.000000
    ---     4.141593      ---
 2.500000      ---     1.000000
```

__array__(*self*)

__getitem__(*self*, *index*)

__mul__(*self*, *other*)

Multiply a sparse matrix by another sparse matrix

```
>>> L1 = SparseMatrix(size = 3)
>>> L1.put((3.,10.,Numeric.pi,2.5), (0,0,1,2), (2,1,1,0))
>>> L2 = SparseIdentityMatrix(size = 3)
>>> L2.put((4.38,12357.2,1.1), (2,1,0), (1,0,2))
>>> print L1 * L2
 1.24e+05  23.140000   3.000000
 3.88e+04   3.141593      ---
 2.500000      ---     2.750000
```

or a sparse matrix by a vector

```
>>> print L1 * Numeric.array((1,2,3),'d')
[ 29.        ,  6.28318531,  2.5       ,]
```

or a vector by a sparse matrix

```
>>> print Numeric.array((1,2,3),'d') * L1
[  7.5       , 16.28318531,  3.        ,]
```

__neg__(*self*)

Negate a sparse matrix

```
>>> print -SparseIdentityMatrix(size = 3)
-1.000000      ---        ---
    ---    -1.000000      ---
    ---        ---    -1.000000
```

__pos__(*self*)

__repr__(*self*)

__rmul__(*self*, *other*)

__setitem__(*self*, *index*, *value*)

__str__(*self*)

__sub__(*self*, *other*)

addAt(*self*, *vector*, *id1*, *id2*)

Add elements of `vector` to the positions in the matrix corresponding to (`id1`,‘id2‘)

```
>>> L = SparseMatrix(size = 3)
>>> L.put((3.,10.,Numeric.pi,2.5), (0,0,1,2), (2,1,1,0))
>>> L.addAt((1.73,2.2,8.4,3.9,1.23), (1,2,0,0,1), (2,2,0,0,2))
>>> print L
12.300000  10.000000   3.000000
    ---     3.141593   2.960000
 2.500000      ---     2.200000
```

addAtDiagonal(*self*, *vector*)

copy(*self*)

getMatrix(*self*)

getShape(*self*)

put(*self*, *vector*, *id1*, *id2*)

Put elements of `vector` at positions of the matrix corresponding to (`id1`, `id2`)

```
>>> L = SparseMatrix(size = 3)
>>> L.put((3.,10.,Numeric.pi,2.5), (0,0,1,2), (2,1,1,0))
>>> print L
    ---    10.000000   3.000000
    ---     3.141593      ---
 2.500000      ---        ---
```

putDiagonal(*self*, *vector*)

Put elements of `vector` along diagonal of matrix

```
>>> L = SparseMatrix(size = 3)
>>> L.putDiagonal((3.,10.,Numeric.pi))
>>> print L
 3.000000      ---        ---
    ---    10.000000      ---
    ---        ---     3.141593
>>> L.putDiagonal((10.,3.))
>>> print L
10.000000      ---        ---
    ---     3.000000      ---
    ---        ---     3.141593
```

take(*self*, *id1*, *id2*)

takeDiagonal(*self*)

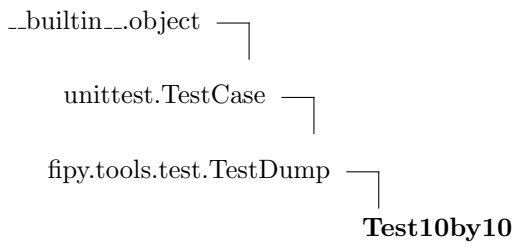
transpose(*self*)

10.9 Module fipy.tools.test

10.9.1 Functions

suite()

10.9.2 Class Test10by10

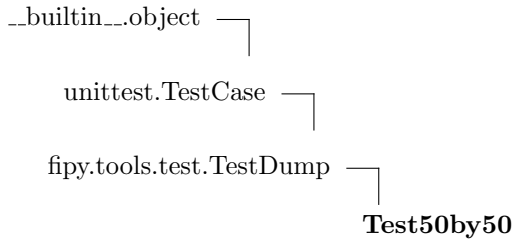


Methods

setUp(*self*)
Overrides: fipy.tools.test.TestDump.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestDump: assertEqual, testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

10.9.3 Class Test50by50

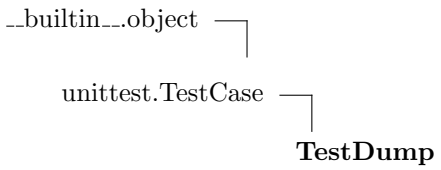


Methods

setUp(*self*)
Overrides: fipy.tools.test.TestDump.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestDump: assertEqual, testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

10.9.4 Class TestDump



Known Subclasses: Test10by10, Test50by50

Methods

assertEqual(*self*, *first*, *second*, *msg*=`None`)

Fail if the two objects are unequal as determined by the '==' operator.

Overrides: unittest.TestCase.failUnlessEqual extit(inherited documentation)

setUp(*self*, *nx*, *ny*)
Overrides: unittest.TestCase.setUp

testResult(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

10.10 Module fipy.tools.vector

Vector utility functions that are inexplicably absent from Numeric

10.10.1 Functions

crossProd(*v1*, *v2*)

Return vector cross-product of v1 and v2.

prune(*array*, *shift*, *start*=0)

removes elements with indices i = start + shift * n where n = 0, 1, 2, ...

putAdd(*vector*, *ids*, *additionVector*)

This is a temporary replacement for Numeric.put as it was not doing what we thought it was doing.

sqrtDot(*v1*, *v2*)

Return square root of vector dot-product of v1 and v2.
Usually used with v1==v2 to return magnitude of v1.

Chapter 11

Module fipy.variables

11.1 Module fipy.variables.addOverFacesVariable

11.1.1 Class AddOverFacesVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

AddOverFacesVariable

Methods

__**init**__(*self*, *faceVariable*, *mesh*=`None`)
Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.2 Module fipy.variables.arithmeticCellToFaceVariable

11.2.1 Class ArithmeticCellToFaceVariable

fipy.variables.variable.Variable ┐

fipy.variables.faceVariable.FaceVariable ┐

fipy.variables.cellToFaceVariable.CellToFaceVariable ┐

ArithmeticCellToFaceVariable

Known Subclasses: ModCellToFaceVariable

Methods

Inherited from CellToFaceVariable: __init__
Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.3 Module fipy.variables.cellGradVariable

11.3.1 Class CellGradVariable

fipy.variables.variable.Variable ┐

fipy.variables.vectorCellVariable.VectorCellVariable ┐

CellGradVariable

Known Subclasses: ModCellGradVariable

Methods

__**init**__(*self*, *var*) Overrides: fipy.variables.vectorCellVariable.VectorCellVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorCellVariable: dot, getArithmeticFaceValue, getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.4 Module fipy.variables.cellToFaceVariable

11.4.1 Class CellToFaceVariable

fipy.variables.variable.Variable

fipy.variables.faceVariable.FaceVariable

CellToFaceVariable

Known Subclasses: ArithmeticCellToFaceVariable, HarmonicCellToFaceVariable, LevelSetDiffusionVariable

Methods

__init__(*self*, *var*) Overrides: fipy.variables.faceVariable.FaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.5 Module fipy.variables.cellVariable

11.5.1 Class CellVariable

fipy.variables.variable.Variable
└
CellVariable

Known Subclasses: AddOverFacesVariable, AddOverFacesVariable, AdsorptionCoeff, AdsorptionCoeff, AnisotropyVariable, DepositionRateVariable, DistanceVariable, InterfaceSurfactantVariable, MetalIonSourceVariable, ModularVariable, MPhiVariable, NoModularVariable, PhaseVariable, RefinedMeshCellVariable, ReMeshedCellVariable, ScaledCellVariable, ScSourceVariable, SourceVariable, SpSourceVariable, SurfactantVariable, ThetaHalfAngleVariable, TransientVariable

Methods

__init__(*self*, *mesh*, *name*=`''`, *value*=`0.0`, *unit*=`None`, *hasOld*=`0`)
Overrides: fipy.variables.variable.Variable.__init__

__call__(*self*, *point*=`None`, *order*=`0`)

"Evaluate" the Variable and return its value

```
>>> a = Variable(value = 3)
>>> a()
3
>>> b = a + 4
>>> b
(Variable(value = 3) + 4)
>>> b()
7
```

Overrides: fipy.variables.variable.Variable.__call__ extit(inherited documentation)

__getstate__(*self*)

__setstate__(*self*, *dict*)

copy(*self*)

Make an duplicate of the Variable

```
>>> a = Variable(value = 3)
>>> b = a.copy()
>>> b
Variable(value = 3)
```

The duplicate will not reflect changes made to the original

```
>>> a.setValue(5)
>>> b
Variable(value = 3)
```

Overrides: fipy.variables.variable.Variable.copy extit(inherited documentation)

getArithmeticFaceValue(*self*)

getCellVolumeAverage(*self*)

Here is a test case for the volume average

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(nx = 3, ny = 1, dx = .5, dy = .1)
>>> print CellVariable(value = (1, 2, 6), mesh = mesh).getCellVolumeAverage()
3.0
```

getFaceDifference(*self*, *order*)

order is odd

getFaceGrad(*self*)

getGrad(*self*)

getGridArray(*self*)

getHarmonicFaceValue(*self*)

getLaplacian(*self*, *order*)

order is even

getOld(*self*)

getValue(*self*, *points*=(), *cells*=())

"Evaluate" the Variable and return its value (longhand)

```
>>> a = Variable(value = 3)
>>> a.getValue()
3
>>> b = a + 4
>>> b
(Variable(value = 3) + 4)
>>> b.getValue()
7
```

Overrides: fipy.variables.variable.Variable.getValue extit(inherited documentation)

getVariableClass(*self*)
Overrides: fipy.variables.variable.Variable.getVariableClass

remesh(*self*, *mesh*)

resetToOld(*self*)

setValue(*self*, *value*, *cells*=())
Overrides: fipy.variables.variable.Variable.setValue

updateOld(*self*)

Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.5.2 Class ReMeshedCellVariable

fipy.variables.variable.Variable

fipy.variables.cellVariable.CellVariable

ReMeshedCellVariable

Methods

__init__(*self*, *oldVar*, *newMesh*) Overrides: fipy.variables.cellVariable.CellVariable.__init__

Inherited from CellVariable: __call__, __getstate__, __setstate__, copy, getArithmeticFaceValue, getCellVolumeAverage, getFaceDifference, getFaceGrad, getGrad, getGridArray, getHarmonicFaceValue, getLaplacian, getOld, getValue, getVariableClass, remesh, resetToOld, setValue, updateOld
Inherited from Variable: __abs__, __add__, __array__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.6 Module fipy.variables.cellVolumeAverageVariable

Takes a `CellVariable` and evaluates its volume average over all the cells.

11.6.1 Class CellVolumeAverageVariable

fipy.variables.variable.Variable
└
CellVolumeAverageVariable

Here is a simple test case:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(nx = 2, ny = 2, dx = 2., dy = 5.)
>>> from fipy.variables.cellVariable import CellVariable
>>> var = CellVariable(value = (1, 2, 3 ,4), mesh = mesh)
>>> print CellVolumeAverageVariable(var)
2.5
```

Methods

__init__(*self*, *var*)
Overrides: fipy.variables.variable.Variable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, getVariableClass, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.7 Module fipy.variables.faceDifferenceVariable

11.7.1 Class FaceDifferenceVariable

fipy.variables.variable.Variable

fipy.variables.faceVariable.FaceVariable

FaceDifferenceVariable

Calculates the difference between the cell values adjacent to the face, weighted by the aspect ratio of the face.

Methods

__init__(*self*, *var*) Overrides: fipy.variables.faceVariable.FaceVariable.__init__

Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.8 Module fipy.variables.faceGradContributionsVariable

11.8.1 Class FaceGradContributions

fipy.variables.variable.Variable ┐

fipy.variables.vectorCellVariable.VectorCellVariable ┐

FaceGradContributions

Methods

__**init**__(*self*, *var*) Overrides: fipy.variables.vectorCellVariable.VectorCellVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorCellVariable: dot, getArithmeticFaceValue, getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.9 Module fipy.variables.faceGradVariable

11.9.1 Class FaceGradVariable

fipy.variables.variable.Variable ┐

fipy.variables.vectorFaceVariable.VectorFaceVariable ┐

FaceGradVariable

Known Subclasses: ModFaceGradVariable

Methods

__init__(*self*, *var*) Overrides: fipy.variables.vectorFaceVariable.VectorFaceVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.10 Module fipy.variables.faceVariable

11.10.1 Class FaceVariable

fipy.variables.variable.Variable
FaceVariable

Known Subclasses: Alpha, Alpha, Alpha, Alpha, Alpha, CellToFaceVariable, DiffusionVariable, FaceDifferenceVariable, FFVariable, PhaseDiffusionVariable, PhaseHalfAngleVariable

Methods

__init__(*self*, *mesh*, *name*=`''`, *value*=`0.0`, *unit*=`None`)
Overrides: fipy.variables.variable.Variable.__init__

getVariableClass(*self*)
Overrides: fipy.variables.variable.Variable.getVariableClass

transpose(*self*)
Overrides: fipy.variables.variable.Variable.transpose

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.11 Module fipy.variables.harmonicCellToFaceVariable

11.11.1 Class HarmonicCellToFaceVariable

fipy.variables.variable.Variable

fipy.variables.faceVariable.FaceVariable

fipy.variables.cellToFaceVariable.CellToFaceVariable

HarmonicCellToFaceVariable

Methods

Inherited from CellToFaceVariable: __init__
Inherited from FaceVariable: getVariableClass, transpose
Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.12 Module fipy.variables.magVariable

11.12.1 Class MagVariable

fipy.variables.variable.Variable
└
MagVariable

Methods

__**init**__(*self*, *var*)
Overrides: fipy.variables.variable.Variable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, getVariableClass, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.13 Module fipy.variables.sumVariable

11.13.1 Class SumVariable

fipy.variables.variable.Variable
└
SumVariable

Methods

__init__(*self*, *var*, *index*)
Overrides: fipy.variables.variable.Variable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, getVariableClass, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.14 Module fipy.variables.test

Test numeric implementation of the mesh

11.14.1 Functions

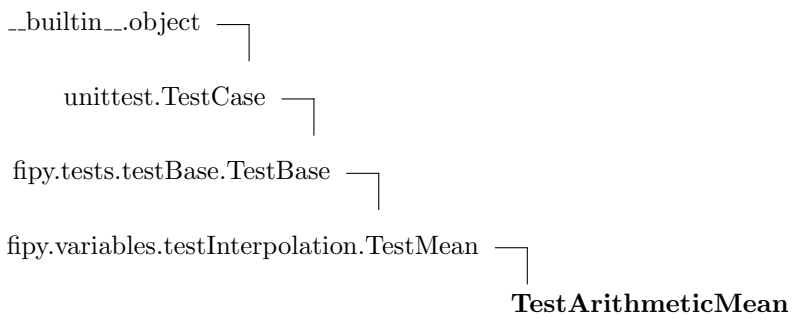
suite()

11.15 Module fipy.variables.testInterpolation

11.15.1 Functions

suite()

11.15.2 Class TestArithmeticMean



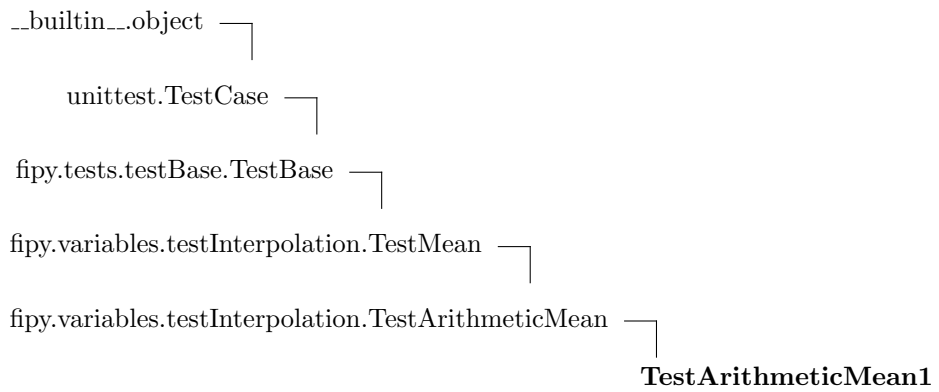
Known Subclasses: TestArithmeticMean1, TestArithmeticMean2, TestArithmeticMean3

Methods

setUp(*self*, *value*, *dx*=`1.0`, *dy*=`1.0`, *factor*=`1`)
Overrides: fipy.variables.testInterpolation.TestMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.3 Class TestArithmeticMean1

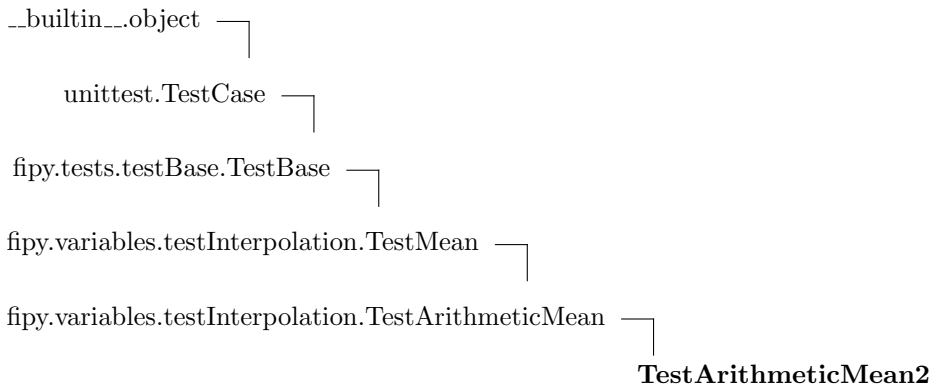


Methods

setUp(*self*)
Overrides: fipy.variables.testInterpolation.TestArithmeticMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.4 Class TestArithmeticMean2

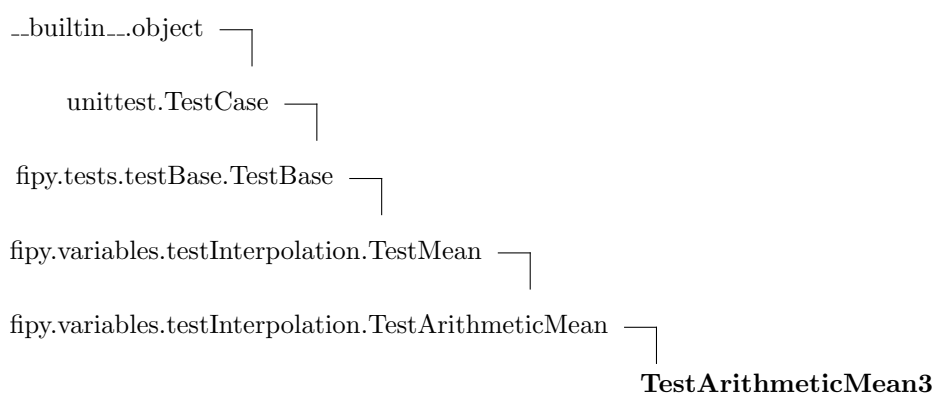


Methods

setUp(*self*)
Overrides: fipy.variables.testInterpolation.TestArithmeticMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.5 Class TestArithmeticMean3

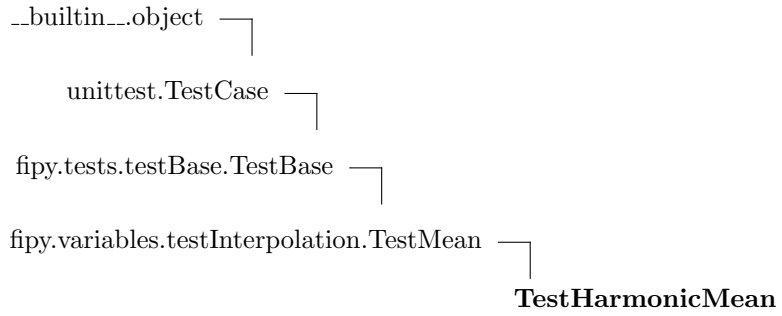


Methods

setUp(*self*)
Overrides: fipy.variables.testInterpolation.TestArithmeticMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.6 Class TestHarmonicMean



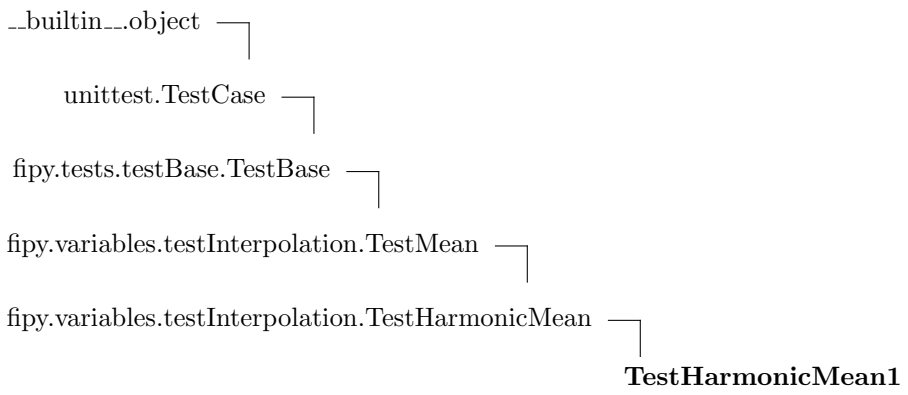
Known Subclasses: TestHarmonicMean1, TestHarmonicMean2, TestHarmonicMean3

Methods

setUp(*self*, *value*, *dx*=`1.0`, *dy*=`1.0`, *factor*=`1`)
Overrides: fipy.variables.testInterpolation.TestMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.7 Class TestHarmonicMean1



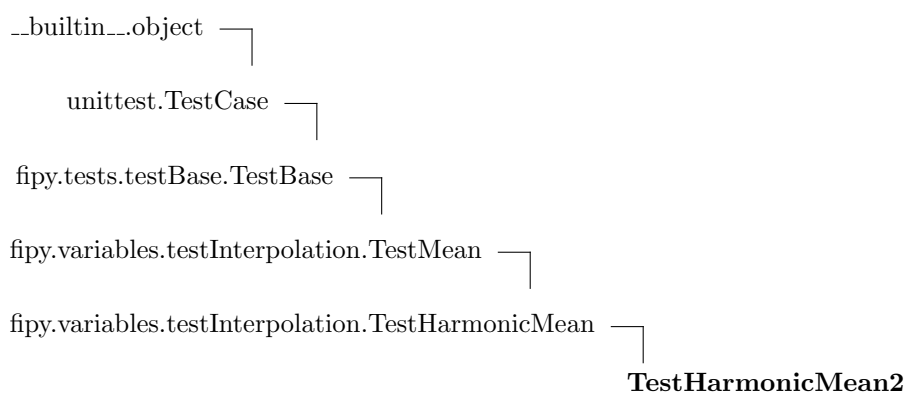
Methods

setUp(*self*)
Overrides: fipy.variables.testInterpolation.TestHarmonicMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult

Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.8 Class TestHarmonicMean2

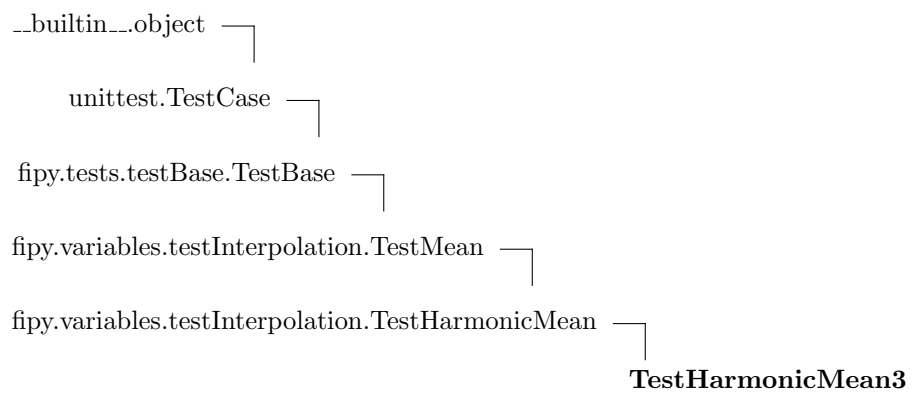


Methods

setUp(*self*)
Overrides: fipy.variables.testInterpolation.TestHarmonicMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.9 Class TestHarmonicMean3

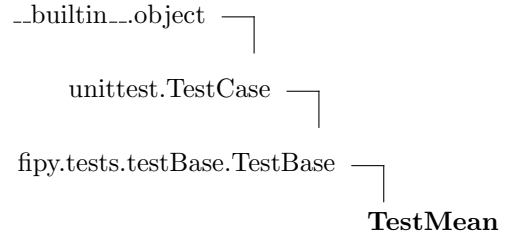


Methods

setUp(*self*)
Overrides: fipy.variables.testInterpolation.TestHarmonicMean.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestMean: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.10 Class TestMean



Known Subclasses: TestArithmeticMean, TestHarmonicMean

Methods

setUp(*self*, *value*, *dx*=`1.0`, *dy*=`1.0`, *factor*=`1`)
Overrides: unittest.TestCase.setUp

testResult(*self*)
Overrides: fipy.tests.testBase.TestBase.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.15.11 Class TestMesh

fipy.meshes.common.mesh.Mesh
  fipy.meshes.numMesh.mesh.Mesh
    fipy.meshes.numMesh.mesh2D.Mesh2D
      fipy.meshes.numMesh.grid2D.Grid2D
        TestMesh

Methods

__init__(*self*, *dx*, *dy*, *nx*, *ny*, *factor*)
Overrides: fipy.meshes.numMesh.grid2D.Grid2D.__init__

createVertices(*self*)
Overrides: fipy.meshes.numMesh.grid2D.Grid2D.createVertices

Inherited from Mesh: calcCellAreas, calcCellToCellIDsFilled, calcExteriorCellIDs, calcFaceAspectRatios, calcInteriorCellIDs, calcScaledGeometry, getAdjacentCellIDs, getAreaProjections, getCellAreaProjections, getCellAreas, getCellCenters, getCellDistances, getCellFaceIDs, getCellFaceOrientations, getCellNormals, getCells, getCellToCellDistances, getCellToCellIDs, getCellToCellIDsFilled, getCellVolumes, getDim, getExteriorCellIDs, getExteriorFaceIDs, getFaceAreas, get-

FaceAspectRatios, getFaceNormals, getFacesWithFilter, getFaceTangents1, getFaceTangents2, getFaceToCellDistanceRatio, getFaceToCellDistances, getInteriorCellIDs, getInteriorFaceIDs, getNearestCell, getNearestCellID, getNumberOfCells, getNumberOfFaces, getOrientedAreaProjections, getOrientedFaceNormals, getPointToCellDistances, setScale
Inherited from Grid2D: __getstate__, __setstate__, createCells, createFaces, getFacesBottom, getFacesLeft, getFacesRight, getFacesTop, getMeshSpacing, getPhysicalShape, getScale, getShape
Inherited from Mesh: __add__, __mul__, calcAdjacentCellIDs, calcAreaProjections, calcCellCenters, calcCellDistances, calcCellNormals, calcCellToCellDistances, calcCellToCellIDs, calcCellToFaceOrientations, calcCellVolumes, calcFaceCellIDs, calcFaceCenters, calcFaceToCellDistanceRatio, calcFaceToCellDistances, calcGeometry, calcInteriorAndExteriorCellIDs, calcInteriorAndExteriorFaceIDs, calcOrientedAreaProjections, calcOrientedFaceNormals, calcTopology, equalExceptOrder, getAddedMeshValues, getCellsByID, getExteriorFaces, getFaceCellIDs, getFaceCenters, getFaces, getInteriorFaces, getMaxFacesPerCell, getVertexCoords
Inherited from Mesh2D: calcFaceAreas, calcFaceNormals, calcFaceTangents, calcHigherOrderScalings, dilate, getNonOrthogonality, getOrderedCellVertexIDs, meshAdd, translate

11.16 Module fipy.variables.testLaplacian

11.16.1 Functions

suite()

11.16.2 Class TestLaplacian

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
TestLaplacian

Known Subclasses: TestLaplacian2, TestLaplacian4, TestLaplacian6

Methods

setUp(*self*, *nx*=`5`, *ny*=`5`, *dx*=`1.0`, *dy*=`1.0`)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

testResult(*self*)
Overrides: fipy.tests.testBase.TestBase.testResult

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.16.3 Class TestLaplacian2

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.variables.testLaplacian.TestLaplacian
TestLaplacian2

Methods

getValue(*self*)

setUp(*self*)
Overrides: fipy.variables.testLaplacian.TestLaplacian.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestLaplacian: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.16.4 Class TestLaplacian4

__builtin__.object
unittest.TestCase
fipy.tests.testBase.TestBase
fipy.variables.testLaplacian.TestLaplacian
TestLaplacian4

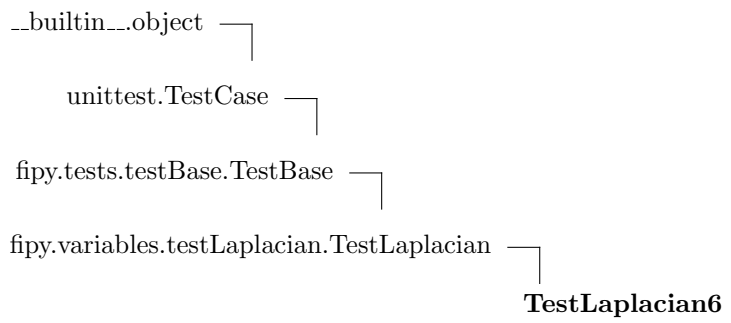
Methods

getValue(*self*)

setUp(*self*)
Overrides: fipy.variables.testLaplacian.TestLaplacian.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestLaplacian: testResult
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.16.5 Class TestLaplacian6



Methods

getValue(*self*)

setUp(*self*)
Overrides: fipy.variables.testLaplacian.TestLaplacian.setUp

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestLaplacian: testResult

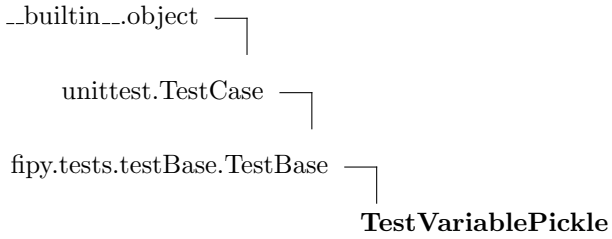
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.17 Module fipy.variables.testPickle

11.17.1 Functions

suite()

11.17.2 Class TestVariablePickle



Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

testOldValue(*self*)

testResult(*self*)
Overrides: fipy.tests.testBase.TestBase.testResult

testValue(*self*)

Inherited from object: __delattr__, __getattribute__, __hash__, __new__, __reduce__, __reduce_ex__, __setattr__
Inherited from TestBase: assertArrayEqual, assertArrayWithinTolerance, assertWithinTolerance, getTestValue, getTestValues
Inherited from TestCase: __init__, __call__, __repr__, __str__, assert_, assertAlmostEqual, assertAlmostEquals, assertEqual, assertEquals, assertNotAlmostEqual, assertNotAlmostEquals, assertNotEqual, assertNotEquals, assertRaises, countTestCases, debug, defaultTestResult, fail, failIf, failIfAlmostEqual, failIfEqual, failUnless, failUnlessAlmostEqual, failUnlessEqual, failUnlessRaises, id, run, shortDescription, tearDown

11.18 Module fipy.variables.transposeVariable

11.18.1 Class TransposeVariable

fipy.variables.variable.Variable

fipy.variables.vectorFaceVariable.VectorFaceVariable

TransposeVariable

Methods

__init__(*self*, *var*) Overrides: fipy.variables.vectorFaceVariable.VectorFaceVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.19 Module fipy.variables.variable

11.19.1 Class Variable

Known Subclasses: CellVariable, CellVolumeAverageVariable, FaceVariable, MagVariable, SumVariable, VectorCellVariable, VectorFaceVariable

Lazily evaluated quantity with units.

Using a Variable in a mathematical expression will create an automatic dependency Variable, e.g.,

```
>>> a = Variable(value = 3)
>>> b = 4 * a
>>> b
(Variable(value = 3) * 4)
>>> b()
12
```

Changes to the value of a Variable will automatically trigger changes in any dependent Variables

```
>>> a.setValue(5)
>>> b
(Variable(value = 5) * 4)
>>> b()
20
```

Methods

__init__(*self*, *value*=`0.0`, *unit*=`None`, *array*=`None`, *name*=`''`, *mesh*=`None`)

Create a Variable.

```
>>> Variable(value = 3)
Variable(value = 3)
>>> Variable(value = 3, unit = "m")
Variable(value = PhysicalField(3,'m'))
>>> Variable(value = 3, unit = "m", array = Numeric.zeros((3,2)))
Variable(value = PhysicalField([[3,3,]
 [3,3,]
 [3,3,]],'m'))
```

Parameters

- `value`: the initial value
- `unit`: the physical units of the variable
- `array`: the storage array for the variable
- `name`: the user-readable name of the variable
- `mesh`: the mesh that defines the geometry of this variable

__abs__(*self*)

__add__(*self*, *other*)

__array__(*self*, *t*=`None`)

Attempt to convert the Variable to a Numeric `array` object

```
>>> v = Variable(value = [2,3])
>>> Numeric.array(v)
[2,3,]
```

It is an error to convert a dimensional Variable to a Numeric `array`

```
>>> v = Variable(value = [2,3], unit = "m")
>>> Numeric.array(v)
Traceback (most recent call last):
    ...
TypeError: Numeric array value must be dimensionless
```

__call__(*self*)

"Evaluate" the Variable and return its value

```
>>> a = Variable(value = 3)
>>> a()
3
>>> b = a + 4
>>> b
(Variable(value = 3) + 4)
>>> b()
7
```

__div__(*self*, *other*)

__eq__(*self*, *other*)

Test if a Variable is equal to another quantity

```
>>> a = Variable(value = 3)
>>> b = (a == 4)
>>> b
(Variable(value = 3) == 4)
>>> b()
0
```

__float__(*self*)

__ge__(*self*, *other*)

Test if a Variable is greater than or equal to another quantity

```
>>> a = Variable(value = 3)
>>> b = (a >= 4)
>>> b
(Variable(value = 3) >= 4)
>>> b()
0
>>> a.setValue(4)
>>> b()
1
>>> a.setValue(5)
>>> b()
1
```

__getitem__(*self*, *index*)

"Evaluate" the variable and return the specified element

```
>>> a = Variable(value = ((3.,4.),(5.,6.)), unit = "m") + "4 m"
>>> print a[1,1]
10.0 m
```

It is an error to slice a Variable whose `value` is not sliceable

```
>>> Variable(value = 3)[2]
Traceback (most recent call last):
      ...
TypeError: unsubscriptable object
```

__gt__(*self*, *other*)

Test if a Variable is greater than another quantity

```
>>> a = Variable(value = 3)
>>> b = (a > 4)
>>> b
(Variable(value = 3) > 4)
>>> b()
0
>>> a.setValue(5)
>>> b()
1
```

__le__(*self*, *other*)

Test if a Variable is less than or equal to another quantity

```
>>> a = Variable(value = 3)
>>> b = (a <= 4)
>>> b
(Variable(value = 3) <= 4)
>>> b()
1
>>> a.setValue(4)
>>> b()
1
>>> a.setValue(5)
>>> b()
0
```

__len__(*self*)

__lt__(*self*, *other*)

Test if a Variable is less than another quantity

```
>>> a = Variable(value = 3)
>>> b = (a < 4)
>>> b
(Variable(value = 3) < 4)
>>> b()
1
>>> a.setValue(4)
>>> b()
0
```

Python automatically reverses the arguments when necessary

```
>>> 4 > Variable(value = 3)
(Variable(value = 3) < 4)
```

__mod__(*self*, *other*)

__mul__(*self*, *other*)

__ne__(*self*, *other*)

Test if a Variable is not equal to another quantity

```
>>> a = Variable(value = 3)
>>> b = (a != 4)
>>> b
(Variable(value = 3) != 4)
>>> b()
1
```

__neg__(*self*)

__pos__(*self*)

__pow__(*self*, *other*)

__radd__(*self*, *other*)

__rdiv__(*self*, *other*)

__repr__(*self*)

__rmul__(*self*, *other*)

__rpow__(*self*, *other*)

__rsub__(*self*, *other*)

__setitem__(*self*, *index*, *value*)

__str__(*self*)

__sub__(*self*, *other*)

allclose(*self*, *other*, *atol*=`1.0000000000000001e-05`, *rtol*=`1e-08`)

arctan(*self*)

copy(*self*)

Make an duplicate of the Variable

```
>>> a = Variable(value = 3)
>>> b = a.copy()
>>> b
Variable(value = 3)
```

The duplicate will not reflect changes made to the original

```
>>> a.setValue(5)
>>> b
Variable(value = 3)
```

cos(*self*)

dot(*self*, *other*)

getBinaryOperatorVariable(*self*, *op*, *other*, *parentClass*=`None`)

getMag(*self*)

getMesh(*self*)

getNumericValue(*self*)

getOperatorVariableClass(*self*, *parentClass*=`None`)

getUnaryOperatorVariable(*self*, *op*, *parentClass*=`None`)

getUnit(*self*)

Return the unit object of `self`.

```
>>> Variable(value = "1 m").getUnit()
<PhysicalUnit m>
```

getValue(*self*)

"Evaluate" the Variable and return its value (longhand)

```
>>> a = Variable(value = 3)
>>> a.getValue()
3
>>> b = a + 4
>>> b
(Variable(value = 3) + 4)
>>> b.getValue()
7
```

getVariableClass(*self*)

inBaseUnits(*self*)

Return the value of the Variable with all units reduced to their base SI elements.

```
>>> e = Variable(value = "2.7 Hartree*Nav")
>>> print e.inBaseUnits()
7088849.01085 kg*m**2/s**2/mol
```

markFresh(*self*)

markStale(*self*)

refresh(*self*)

requiredBy(*self*, *var*)

requires(*self*, *var*)

setValue(*self*, *value*, *unit*=`None`, *array*=`None`)

sin(*self*)

sqrt(*self*)

sum(*self*, *index*=`0`)

take(*self*, *ids*)

tan(*self*)

transpose(*self*)

Class Variables

Name	Description
__variable__	**Value:** `True` *(type=`bool`)*

11.20 Module fipy.variables.vectorArithmeticCellToFaceVariable

11.20.1 Class VectorArithmeticCellToFaceVariable

fipy.variables.variable.Variable

fipy.variables.vectorFaceVariable.VectorFaceVariable

fipy.variables.vectorCellToFaceVariable.VectorCellToFaceVariable

VectorArithmeticCellToFaceVariable

Methods

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorCellToFaceVariable: __init__
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.21 Module fipy.variables.vectorCellToFaceVariable

11.21.1 Class VectorCellToFaceVariable

fipy.variables.variable.Variable

fipy.variables.vectorFaceVariable.VectorFaceVariable

VectorCellToFaceVariable

Known Subclasses: VectorArithmeticCellToFaceVariable

Methods

__**init**__(*self*, *var*) Overrides: fipy.variables.vectorFaceVariable.VectorFaceVariable.__init__

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose
Inherited from VectorFaceVariable: getVariableClass

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.22 Module fipy.variables.vectorCellVariable

11.22.1 Class VectorCellVariable

fipy.variables.variable.Variable
└
VectorCellVariable

Known Subclasses: CellGradVariable, FaceGradContributions

Methods

__init__(*self*, *mesh*, *name*=`''`, *value*=`0.0`, *unit*=`None`)
Overrides: fipy.variables.variable.Variable.__init__

dot(*self*, *other*)
Overrides: fipy.variables.variable.Variable.dot

getArithmeticFaceValue(*self*)

Return a `VectorFaceVariable` with values determined by the arithmetic mean from the neighboring cells.

```
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1, nx = 2, ny = 1)
>>> var = VectorCellVariable(mesh, value = Numeric.array(((0,0),(1,1))))
>>> answer = Numeric.array(((0, 0), (1, 1), (0, 0), (1, 1), (0, 0), (.5, .5), (1, 1)))
>>> Numeric.allclose(answer, Numeric.array(var.getArithmeticFaceValue()))
1
```

getVariableClass(*self*)
Overrides: fipy.variables.variable.Variable.getVariableClass

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

11.23 Module fipy.variables.vectorFaceVariable

11.23.1 Class VectorFaceVariable

fipy.variables.variable.Variable
└
VectorFaceVariable

Known Subclasses: ConvectionCoeff, DPhiReverse, FaceGradVariable, TransposeVariable, VectorCellToFaceVariable

Methods

__init__(*self*, *mesh*, *name*=`''`, *value*=`0.0`, *unit*=`None`, *length*=`1`)
Overrides: fipy.variables.variable.Variable.__init__

getVariableClass(*self*)
Overrides: fipy.variables.variable.Variable.getVariableClass

Inherited from Variable: __abs__, __add__, __array__, __call__, __div__, __eq__, __float__, __ge__, __getitem__, __gt__, __le__, __len__, __lt__, __mod__, __mul__, __ne__, __neg__, __pos__, __pow__, __radd__, __rdiv__, __repr__, __rmul__, __rpow__, __rsub__, __setitem__, __str__, __sub__, allclose, arctan, copy, cos, dot, getBinaryOperatorVariable, getMag, getMesh, getNumericValue, getOperatorVariableClass, getUnaryOperatorVariable, getUnit, getValue, inBaseUnits, markFresh, markStale, refresh, requiredBy, requires, setValue, sin, sqrt, sum, take, tan, transpose

Class Variables

Name	Description
Inherited from Variable: __variable__ *(p. 298)*	

Chapter 12

Module fipy.viewers

12.1 Module fipy.viewers.colorbar

12.1.1 Functions

color_bar(*minz*, *maxz*, *split*=`0`, *ncol*=`None`, *ymax*=`0.84999999999999998`, *ymin*=`0.44`, *xmin0*=`0.62`, *xmax0*=`0.64000000000000001`, *zlabel*=`None`, *fontsize*=`16`, *font*=`'helvetica'`, *color*=`'black'`)

color_bar (minz, maxz) plots a color bar to the right of the plot square labelled by the z values from minz to maxz.
plf (z, y, x) color_bar (z (min, min), z (max, max))
or plf (z, y, x, cmin = MINZ, cmax = MAXZ) color_bar (MINZ, MAXZ)
are typical usage

nice_levels(*z*, *n*=`8`)

nice_levels(z, n = 8) finds approximately n "nice values" between min(z) and max(z) for axis labels. n defaults to eight.

12.2 Module fipy.viewers.gist1DViewer

12.2.1 Class Gist1DViewer

fipy.viewers.gistViewer.GistViewer
└
Gist1DViewer

Methods

__init__(*self*, *vars*=`None`, *title*=`None`, *limits*=`None`, *xlog*=`0`, *ylog*=`0`, *style*=`'work.gs'`)
Overrides: fipy.viewers.gistViewer.GistViewer.__init__

getArrays(*self*)

plot(*self*, *minVal*=`None`, *maxVal*=`None`)
Overrides: fipy.viewers.gistViewer.GistViewer.plot

plotArrays(*self*)

Inherited from GistViewer: getArray

Class Variables

Name	Description
Inherited from GistViewer: id *(p. 314)*	

12.3 Module fipy.viewers.gistVectorViewer

12.3.1 Class GistVectorViewer

fipy.viewers.gistViewer.GistViewer
└
GistVectorViewer

Methods

__init__(*self*, *var*=`None`, *title*=`''`, *grid*=`1`)
Overrides: fipy.viewers.gistViewer.GistViewer.__init__

getArray(*self*)
Overrides: fipy.viewers.gistViewer.GistViewer.getArray

plot(*self*, *fileName*=`None`)
Overrides: fipy.viewers.gistViewer.GistViewer.plot

Class Variables

Name	Description
Inherited from GistViewer: id *(p. 314)*	

12.4 Module fipy.viewers.gistViewer

12.4.1 Class GistViewer

Known Subclasses: Gist1DViewer, GistVectorViewer, Grid2DGistViewer, NumpyGistViewer, VarGistViewer

Methods

__init__(*self*, *minVal*=`None`, *maxVal*=`None`, *title*=`''`, *palette*=`'heat.gp'`, *grid*=`1`, *limits*=`None`, *dpi*=`75`)

getArray(*self*)

plot(*self*, *minVal*=`None`, *maxVal*=`None`, *fileName*=`None`)

Class Variables

Name	Description
id	**Value:** `0` *(type=`int`)*

12.5 Module fipy.viewers.gnuplotViewer

The `GnuplotViewer` plots a 2D Numeric array using a front end python wrapper available to download (Gnuplot.py).

If one would like more specific styles for a plot, it is probably best to create an inherited class and change the `gnuplotCommands` method.

Different style script demos are available at the Gnuplot site.

> **Note**
>
> `GnuplotViewer` requires Gnuplot version 4.0.

12.5.1 Class GnuplotViewer

Methods

__init__(*self*, *array*, *dx*=`1.0`, *dy*=`1.0`, *maxVal*=`None`, *minVal*=`None`, *palette*=`'color'`, *legend*=`True`, *title*=`''`)

Parameters

`array:`	A 2D Numeric array
`dx:`	The unit size is the x-direction.
`dy:`	The unit size in the y-direction.
`maxVal:`	The maximum value to appear on the legend
`palette:`	The color scheme. Other choices might be `color negative` or `gray`. This is completely configurable.
`legend:`	Whether to display the legend.

gnuplotCommands(*self*, *g*)

plot(*self*, *fileName*=`None`)

Parameters

`fileName:`	If no file name is passed a plot appears on the screen. If `*.ps` or `*.pdf` is passed, a postscript or PDF file is produced and nothing appears on the screen.

12.6 Module fipy.viewers.grid2DGistViewer

The `Grid2DGistViewer` uses the gist plotting package to display a contour plot of a variable. It assumes the mesh that the variable has is a `Grid2D` type mesh.

It takes a resolution argument when it is instantiated. This argument increases the resolution of the plot by doing interpolation on the values from the variable. Here is a test for increasing the resolution:

```
>>> from fipy.variables.cellVariable import CellVariable
>>> from fipy.meshes.grid2D import Grid2D
>>> nx = 2
>>> ny = 4
>>> dx = .5
>>> dy = 1.
>>> mesh = Grid2D(dx = dx, dy = dy, nx = nx, ny = ny)
>>> v = (mesh.getCellCenters()[:,0] + mesh.getCellCenters()[:,1])
>>> var = CellVariable(mesh = mesh, value = v)
>>> resolution = 3
>>> viewer = Grid2DGistViewer(var = var, resolution = resolution)
>>> pow = 2**(resolution-1)
>>> nx = pow * nx - (pow - 1)
>>> ny = pow * ny - (pow - 1)
>>> index = Numeric.indices((ny, nx))
>>> ndx = dx / pow
>>> ndy = dy / pow
>>> answer = (index[1] * ndx + dx / 2) + (index[0] * ndy + dy / 2)
>>> Numeric.allclose(answer, viewer.getArray())
1
```

12.6.1 Class Grid2DGistViewer

fipy.viewers.gistViewer.GistViewer

Grid2DGistViewer

Methods

__init__(*self*, *var*=`None`, *minVal*=`None`, *maxVal*=`None`, *palette*=`'heat.gp'`, *grid*=`1`, *resolution*=`1`, *limits*=`None`, *dpi*=`75`)

Parameters

`var:`	The variable that is to be plotted
`minVal:`	The minimum value to display in the plot. The default will use the minimum value from the variable values.
`maxVal:`	The maximum value to display in the plot. The default will use the maximum value from the variable values.
`palette:`	The color scheme to use for the contour plot. Default is `heat.gp`. Another choice would be `rainbow.gp`.
`grid:`	Whether to show the grid lines in the plot. Default is 1. Use 0 to switch them off.
`resolution:`	Is an integer greater than 0. It corresponds to the amount of resolution required for the plot. The default is 1. The increase in resolution is equivalent to `2**(resolution - 1)`.
`limits:`	Plot limits of interest. A tuple of four values corresponding to `(xmin, xmax, ymin, ymax)`

Overrides: fipy.viewers.gistViewer.GistViewer.__init__

getArray(*self*)
Overrides: fipy.viewers.gistViewer.GistViewer.getArray

increaseResolution(*self*, *array*)

```
>>> from fipy.variables.cellVariable import CellVariable
>>> from fipy.meshes.grid2D import Grid2D
>>> mesh = Grid2D(dx = 1., dy = 1., nx = 2, ny = 4)
>>> v = (mesh.getCellCenters()[:,0] + mesh.getCellCenters()[:,1])
>>> var = CellVariable(mesh = mesh, value = v)
>>> viewer = Grid2DGistViewer(var = var)
>>> nx,ny = mesh.getShape()
>>> array = Numeric.reshape(var.getNumericValue(),(ny,nx))
>>> print viewer.increaseResolution(array)
[[ 1. , 1.5, 2. ,]
 [ 1.5, 2. , 2.5,]
 [ 2. , 2.5, 3. ,]
 [ 2.5, 3. , 3.5,]
 [ 3. , 3.5, 4. ,]
 [ 3.5, 4. , 4.5,]
 [ 4. , 4.5, 5. ,]]
```

setVar(*self*, *var*)

Inherited from GistViewer: plot

Class Variables

Name	Description
Inherited from GistViewer: id *(p. 314)*	

12.7 Module fipy.viewers.numpyGistViewer

12.7.1 Class NumpyGistViewer

fipy.viewers.gistViewer.GistViewer
└
NumpyGistViewer

Methods

__init__(*self*, *array*, *minVal*=`0.0`, *maxVal*=`1.0`, *palette*=`'heat.gp'`, *grid*=`1`, *limits*=`None`, *dpi*=`75`)
Overrides: fipy.viewers.gistViewer.GistViewer.__init__

getArray(*self*)
Overrides: fipy.viewers.gistViewer.GistViewer.getArray

Inherited from GistViewer: plot

Class Variables

Name	Description
Inherited from GistViewer: id *(p. 314)*	

12.8 Module fipy.viewers.numpyPyxViewer

‘NumpyPyxViewer‘ - A simple PyxViewer for 2D arrays

12.8.1 Class NumpyPyxViewer

Methods

__init__(*self*, *array*, *dx*=`1.0`, *dy*=`1.0`, *plotWidth*=`10`, *viewCommand*=`'gv'`, *maxVal*=`1.0`, *minVal*=`0.0`)

plot(*self*, *fileName*=`'tmp'`)

12.9 Module fipy.viewers.pyxviewer

The PyxViewer module takes a variable as input and plots a contour map of the values of that variable. The contour map consists of a large number of small rectangles (hereafter referred to as 'plot cells') each of which is a different color corresponding to the value of the variable at the center of it. The PyxViewer has a subclass Grid2DPyxViewer. Grid2DPyxViewer should be used when the variable's mesh is a Grid2D mesh, and PyxViewer should be used otherwise.

A PyxViewer is created with one argument: the variable to be plotted. To actually plot the variable, it is necessary to call the PyxViewer's plot() method. The plot() method has the following keyword arguments:

debug - When set to 1, prints debugging information. Default value is 0.

returnlist - When set to 1, returns the displaylist. Used for debugging and test cases. Default value is 0.

minx, maxx - The minimum and maximum values of X to appear on the plot. By default, minx and maxx are set to the minimum and maximum values of the X coordinates of the mesh vertices.

miny, maxy - The minimum and maximum values of Y to appear on the plot. By default, miny and maxy are set to the minimum and maximum values of the Y coordinates of the mesh vertices.

minval, maxval - The minimum and maximum values on the color scale. A value of minval will appear completely blue and a value of maxval will appear completely red. If no values are specified, minval will default to the lowest value of the variable over the area to be plotted and maxval will default to the highest value of the variable over the area to be plotted.

resolution - The distance between the centers of adjacent plot cells. Lower values will result in higher quality images. If no value is specified, the resolution will be set such that there will be approximately 10,000 plot cells.

filename - The name of the file to store the resulting image in. An extension is not needed here - the .eps extension will automatically be added. If no filename is specified, the image will be stored in the file `temp.eps` so it can be viewed.

viewcmd - The OS command used to access your graphics viewer. If specified, the graphics viewer will be automatically opened to view your plot. Default is 'gv'.

xlabel - The label to use for the X axis. Default is 'X values'.

ylabel - The label to use for the Y axis. Default is 'Y values'.

valuelabel - The label to use for the value of the variable on the scale. Default is the variable's name.

gridcorrect - The amount, in multiples of the resolution, that adjacent plot cells overlap. This overlap eliminates the 'grid effect'. If you still see a grid of white space, increase this value. Default value is 0.3.

graphheight - The physical height of the graph. Default is 12.

graphwidth - The physical width of the graph. Default is 12.

mask - Used to 'blank out' specified areas of the graph, useful if you have a mesh whose boundaries do not form a rectangle. The format for mask is an N-by-4 array, where N is the number of different rectangles you want to blank out. Each row represents a separate blanked-out rectangle: the first two numbers being the X and Y coordinates of the lower left hand corner of said rectangle, and the last two being the X and Y coordinates of the upper right hand corner. For example, mask = [[0, 0, 3, 2], [7, 8, 10, 10]] indicates that the area inside the rectangle with lower left corner (0, 0) and upper left corner (3, 2) should be blanked out, as should the area inside the rectangle with lower left coordinate (7, 8) and upper left coordinate (10, 10).

The following are keyword arguments in the __init__() method: (Note that these work for PyxViewer only, not Grid2DPyxViewer)

showpercent - If set to 1, this displays the percentage complete at 5% intervals as the plot progresses. Default value is 1.

showtime - If set to 1, this displays the approximate time it will take to plot it. Default value is 1.

Test cases:

```
>>> from fipy.meshes.grid2D import Grid2D
>>> import fipy.variables.cellVariable
>>> mesh = Grid2D(1.0, 1.0, 2, 2)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
>>> myvar[0] = 1.0
>>> myvar[1] = 2.0
>>> myvar[2] = 3.0
>>> myvar[3] = 4.0
>>> myviewer = Grid2DPyxViewer(myvar, showpercent = 0, showtime = 0)
>>> array = myviewer.getValueMatrix(0.0, 2.0, 0.0, 2.0, 1.0)
>>> testlist = array.tolist()
>>> print testlist
[[1.0, 3.0], [2.0, 4.0]]
```

```
>>> from fipy.meshes.grid2D import Grid2D
>>> import fipy.variables.cellVariable
>>> mesh = Grid2D(1.0, 1.0, 2, 2)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
>>> myvar[0] = 1.0
>>> myvar[1] = 2.0
>>> myvar[2] = 3.0
>>> myvar[3] = 4.0
>>> myviewer = Grid2DPyxViewer(myvar, showpercent = 0, showtime = 0)
>>> testlist = myviewer.plot(minx = 0.0, maxx = 2.0, miny = 0.0, maxy = 2.0, resolution = 1
>>> print testlist
[[0.0, 1.0, 0.0, 1.0, 0.0], [0.0, 1.0, 1.0, 2.0, 0.5], [1.0, 2.0, 0.0, 1.0, 0.25], [1.0, 2.
```

```
>>> from fipy.meshes.grid2D import Grid2D
>>> import fipy.variables.cellVariable
>>> mesh = Grid2D(1.0, 1.0, 2, 2)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
```

```
>>> myvar[0] = 2.0
>>> myvar[1] = 3.0
>>> myvar[2] = 5.0
>>> myvar[3] = 6.0
>>> myviewer = Grid2DPyxViewer(myvar)
>>> testlist = myviewer.plot(minx = 0.0, maxx = 2.0, miny = 0.0, maxy = 2.0, resolution = 1.0, gri
>>> print testlist
[[0.0, 1.0, 0.0, 1.0, 0.0], [0.0, 1.0, 1.0, 2.0, 0.75], [1.0, 2.0, 0.0, 1.0, 0.25], [1.0, 2.0, 1.0
```

```
>>> from fipy.meshes.grid2D import Grid2D
>>> import fipy.variables.cellVariable
>>> mesh = Grid2D(1.0, 1.0, 2, 2)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
>>> myvar[0] = 1.0
>>> myvar[1] = 2.0
>>> myvar[2] = 3.0
>>> myvar[3] = 4.0
>>> myviewer = Grid2DPyxViewer(myvar)
>>> array = myviewer.getValueMatrix(0.0, 2.0, 0.0, 2.0, 0.5)
>>> testlist = array.tolist()
>>> print testlist
[[0.25, 1.25, 2.25, 3.25], [0.75, 1.75, 2.75, 3.75], [1.25, 2.25, 3.25, 4.25], [1.75, 2.75, 3.75,
```

```
>>> from fipy.meshes.grid2D import Grid2D
>>> import fipy.variables.cellVariable
>>> mesh = Grid2D(1.0, 1.0, 2, 2)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
>>> myvar[0] = 1.0
>>> myvar[1] = 2.0
>>> myvar[2] = 3.0
>>> myvar[3] = 4.0
>>> myviewer = PyxViewer(myvar, showpercent = 0, showtime = 0)
>>> array = myviewer.getValueMatrix(0.0, 2.0, 0.0, 2.0, 0.5)
>>> testlist = array.tolist()
>>> print testlist
[[1.0, 1.0, 3.0, 3.0], [1.0, 1.0, 3.0, 3.0], [2.0, 2.0, 4.0, 4.0], [2.0, 2.0, 4.0, 4.0]]
```

```
>>> from fipy.meshes.numMesh.grid3D import Grid3D
>>> import fipy.variables.cellVariable
>>> mesh = Grid3D(1.0, 1.0, 1.0, 2, 2, 2)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
>>> myvar[0] = 1.0
>>> myvar[1] = 2.0
>>> myvar[2] = 3.0
>>> myvar[3] = 4.0
>>> myvar[4] = 5.0
>>> myvar[5] = 6.0
>>> myvar[6] = 7.0
```

```
>>> myvar[7] = 8.0
>>> myviewer = Grid3DPyxViewer(myvar, zvalue = 1.0)
>>> array = myviewer.getValueMatrix(0.0, 2.0, 0.0, 2.0, 0.5)
>>> testlist = array.tolist()
>>> print testlist
[[2.25, 3.25, 4.25, 5.25], [2.75, 3.75, 4.75, 5.75], [3.25, 4.25, 5.25, 6.25], [3.75, 4.75,
```

```
>>> from fipy.meshes.numMesh.grid3D import Grid3D
>>> import fipy.variables.cellVariable
>>> mesh = Grid3D(1.0, 1.0, 1.0, 2, 2, 1)
>>> myvar = fipy.variables.cellVariable.CellVariable(mesh)
>>> myvar[0] = 1.0
>>> myvar[1] = 2.0
>>> myvar[2] = 3.0
>>> myvar[3] = 4.0
>>> myviewer = Grid3DPyxViewer(myvar, zvalue = 0.75)
>>> array = myviewer.getValueMatrix(0.0, 2.0, 0.0, 2.0, 0.5)
>>> testlist = array.tolist()
>>> print testlist
[[0.25, 1.25, 2.25, 3.25], [0.75, 1.75, 2.75, 3.75], [1.25, 2.25, 3.25, 4.25], [1.75, 2.75,
```

12.9.1 Variables

Name	Description
InvalidClassException	**Value:** `'Wrong class'` *(type=`str`)*
ValueOutOfRangeException	**Value:** `'Value out of range'` *(type=`str`)*

12.9.2 Class Grid2DPyxViewer

fipy.viewers.pyxviewer.PyxViewer
└
Grid2DPyxViewer

Known Subclasses: Grid3DPyxViewer

Methods

__init__(*self*, *variable*, *showpercent*=0, *showtime*=0)
Overrides: fipy.viewers.pyxviewer.PyxViewer.__init__

getValue(*self*, *x*, *y*)
Overrides: fipy.viewers.pyxviewer.PyxViewer.getValue

Inherited from PyxViewer: generateArray, getValueMatrix, plot

12.9.3 Class Grid3DPyxViewer

fipy.viewers.pyxviewer.PyxViewer

fipy.viewers.pyxviewer.Grid2DPyxViewer

Grid3DPyxViewer

Methods

__init__(*self*, *variable*, *showpercent*=`0`, *showtime*=`0`, *zvalue*=`0`)
Overrides: fipy.viewers.pyxviewer.Grid2DPyxViewer.__init__

generateArray(*self*)

generates the array of variable values

Overrides: fipy.viewers.pyxviewer.PyxViewer.generateArray extit(inherited documentation)

Inherited from Grid2DPyxViewer: getValue
Inherited from PyxViewer: getValueMatrix, plot

12.9.4 Class PyxViewer

Known Subclasses: Grid2DPyxViewer

Methods

__init__(*self*, *variable*, *showpercent*=`1`, *showtime*=`1`)

generateArray(*self*)

generates the array of variable values

getValue(*self*, *x*, *y*)

getValueMatrix(*self*, *minx*, *maxx*, *miny*, *maxy*, *resolution*)

plot(*self*, *returnlist*=`0`, *debug*=`0`, *minx*=`None`, *maxx*=`None`, *miny*=`None`, *maxy*=`None`, *minval*=`None`, *maxval*=`None`, *resolution*=`None`, *filename*=`None`, *viewcmd*=`'gv'`, *xlabel*=`'X values'`, *ylabel*=`'Y values'`, *valuelabel*=`None`, *mask*=`None`, *gridcorrect*=`0.29999999999999999`, *graphheight*=`12`, *graphwidth*=`12`)

12.10 Module fipy.viewers.test

Test numeric implementation of the mesh

12.10.1 Functions

suite()

12.11 Module fipy.viewers.varGistViewer

12.11.1 Class VarGistViewer

fipy.viewers.gistViewer.GistViewer
└
VarGistViewer

Methods

__init__(*self*, *var*=`None`, *minVal*=`0.0`, *maxVal*=`1.0`)
Overrides: fipy.viewers.gistViewer.GistViewer.__init__

getArray(*self*)
Overrides: fipy.viewers.gistViewer.GistViewer.getArray

setVar(*self*, *var*)

Inherited from GistViewer: plot

Class Variables

Name	Description
Inherited from GistViewer: id *(p. 314)*	

12.12 Module fipy.viewers.viewer

12.12.1 Class gistViewer

Methods

__init__(*self*, *var*, *minVal*=`0.0`, *maxVal*=`1.0`)

plot(*self*)

Class Variables

Name	Description
id	**Value:** `0` *(type=`int`)*

Index

fipy.boundaryConditions.boundaryCondition *(module)*, 1–2
BoundaryCondition *(class)*, 1–2
__init__ *(method)*, 1
getContribution *(method)*, 1
getDerivative *(method)*, 1
getFaces *(method)*, 2
fipy.boundaryConditions.fixedFlux *(module)*, 3
FixedFlux *(class)*, 3
__init__ *(method)*, 3
getContribution *(method)*, 3
getDerivative *(method)*, 3
fipy.boundaryConditions.fixedValue *(module)*, 4
FixedValue *(class)*, 4
getContribution *(method)*, 4
fipy.boundaryConditions.nthOrderBoundaryCondition *(module)*, 5
NthOrderBoundaryCondition *(class)*, 5
__init__ *(method)*, 5
getContribution *(method)*, 5
getDerivative *(method)*, 5
fipy.equations.advectionEquation *(module)*, 7
AdvectionEquation *(class)*, 7
__init__ *(method)*, 7
fipy.equations.diffusionEquation *(module)*, 8
DiffusionEquation *(class)*, 8
__init__ *(method)*, 8
fipy.equations.diffusionEquationWithSource *(module)*, 9
DiffusionEquationWithSource *(class)*, 9
__init__ *(method)*, 9
fipy.equations.equation *(module)*, 10
Equation *(class)*, 10
__init__ *(method)*, 10
getFigureOfMerit *(method)*, 10
getResidual *(method)*, 10
getSolutionTolerance *(method)*, 10
getVar *(method)*, 10
isConverged *(method)*, 10
solve *(method)*, 10
updateVar *(method)*, 10
fipy.equations.explicitDiffusionEquation *(module)*, 11
ExplicitDiffusionEquation *(class)*, 11
__init__ *(method)*, 11
fipy.equations.matrixEquation *(module)*, 12
MatrixEquation *(class)*, 12
buildMatrix *(method)*, 12
getResidual *(method)*, 12
getResidual2 *(method)*, 12
getVar *(method)*, 12
postSolve *(method)*, 12
solve *(method)*, 12
fipy.equations.nthOrderDiffusionEquation *(module)*, 13
NthOrderDiffusionEquation *(class)*, 13
__init__ *(method)*, 13
fipy.equations.postRelaxationEquation *(module)*, 14
PostRelaxationEquation *(class)*, 14
postSolve *(method)*, 14
fipy.equations.preRelaxationEquation *(module)*, 15
PreRelaxationEquation *(class)*, 15
buildMatrix *(method)*, 15
fipy.equations.relaxationEquation *(module)*, 16
RelaxationEquation *(class)*, 16
__init__ *(method)*, 16
getRelaxation *(method)*, 16
postSolve *(method)*, 16
fipy.equations.sourceEquation *(module)*, 17
SourceEquation *(class)*, 17
__init__ *(method)*, 17
fipy.equations.stdyConvDiffEquation *(module)*, 18
SteadyConvectionDiffusionEquation *(class)*, 18
__init__ *(method)*, 18
fipy.equations.stdyConvDiffScEquation *(module)*, 19

SteadyConvectionDiffusionScEquation *(class)*, 19
__init__ *(method)*, 19
fipy.iterators.adaptiveIterator *(module)*, 21–22
AdaptiveIterator *(class)*, 21–22
__init__ *(method)*, 21
adjustTimeStep *(method)*, 21
resetTimeStep *(method)*, 21
timestep *(method)*, 21
fipy.iterators.iterator *(module)*, 23–24
ConvergenceError *(class)*, 23
__init__ *(method)*, 23
__str__ *(method)*, 23
Iterator *(class)*, 23–24
__init__ *(method)*, 23
advanceTimeStep *(method)*, 23
elapseTime *(method)*, 23
sweep *(method)*, 24
sweeps *(method)*, 24
timestep *(method)*, 24
fipy.meshes.common.mesh *(module)*, 25–28
Mesh *(class)*, 25–28
__init__ *(method)*, 25
calcAdjacentCellIDs *(method)*, 25
calcAreaProjections *(method)*, 25
calcCellAreas *(method)*, 25
calcCellCenters *(method)*, 25
calcCellDistances *(method)*, 25
calcCellToCellDistances *(method)*, 25
calcCellToCellIDs *(method)*, 26
calcCellToCellIDsFilled *(method)*, 26
calcCellToFaceOrientations *(method)*, 26
calcCellVolumes *(method)*, 26
calcExteriorCellIDs *(method)*, 26
calcFaceAreas *(method)*, 26
calcFaceAspectRatios *(method)*, 26
calcFaceNormals *(method)*, 26
calcFaceTangents *(method)*, 26
calcFaceToCellDistanceRatio *(method)*, 26
calcFaceToCellDistances *(method)*, 26
calcGeometry *(method)*, 26
calcHigherOrderScalings *(method)*, 26
calcInteriorAndExteriorCellIDs *(method)*, 26
calcInteriorAndExteriorFaceIDs *(method)*, 26
calcInteriorCellIDs *(method)*, 26
calcOrientedAreaProjections *(method)*, 26
calcOrientedFaceNormals *(method)*, 26
calcScaledGeometry *(method)*, 26
calcTopology *(method)*, 26
getAdjacentCellIDs *(method)*, 26
getAreaProjections *(method)*, 27
getCellAreaProjections *(method)*, 27
getCellAreas *(method)*, 27
getCellCenters *(method)*, 27
getCellDistances *(method)*, 27
getCellFaceIDs *(method)*, 27
getCellFaceOrientations *(method)*, 27
getCellNormals *(method)*, 27
getCells *(method)*, 27
getCellsByID *(method)*, 27
getCellToCellDistances *(method)*, 27
getCellToCellIDs *(method)*, 27
getCellToCellIDsFilled *(method)*, 27
getCellVolumes *(method)*, 27
getDim *(method)*, 27
getExteriorCellIDs *(method)*, 27
getExteriorFaceIDs *(method)*, 27
getExteriorFaces *(method)*, 27
getFaceAreas *(method)*, 27
getFaceAspectRatios *(method)*, 27
getFaceNormals *(method)*, 28
getFaces *(method)*, 28
getFacesWithFilter *(method)*, 28
getFaceTangents1 *(method)*, 28
getFaceTangents2 *(method)*, 28
getFaceToCellDistanceRatio *(method)*, 28
getFaceToCellDistances *(method)*, 28
getInteriorCellIDs *(method)*, 28
getInteriorFaceIDs *(method)*, 28
getInteriorFaces *(method)*, 28
getMaxFacesPerCell *(method)*, 28
getNearestCell *(method)*, 28
getNearestCellID *(method)*, 28
getNumberOfCells *(method)*, 28
getNumberOfFaces *(method)*, 28
getOrientedAreaProjections *(method)*, 28

getOrientedFaceNormals *(method)*, 28
getPointToCellDistances *(method)*, 28
setScale *(method)*, 28
fipy.meshes.numMesh.adaptiveMesh *(module)*, 29–31
AdaptiveMesh2D *(class)*, 29–31
__init__ *(method)*, 30
createBackgroundMesh *(method)*, 30
finalInit *(method)*, 30
getExteriorLines *(method)*, 30
getExteriorVertexIDs *(method)*, 30
getGeometryPoints *(method)*, 30
writeBackgroundMesh *(method)*, 30
writeGeometryFile *(method)*, 30
bracedList *(function)*, 29
orderVertices *(function)*, 29
parenList *(function)*, 29
removeDuplicates *(function)*, 29
fipy.meshes.numMesh.cell *(module)*, 32
Cell *(class)*, 32
__cmp__ *(method)*, 32
__init__ *(method)*, 32
getCellToCellDistances *(method)*, 32
getCellToCellIDs *(method)*, 32
getCenter *(method)*, 32
getID *(method)*, 32
getMesh *(method)*, 32
getNormal *(method)*, 32
fipy.meshes.numMesh.face *(module)*, 33
Face *(class)*, 33
__init__ *(method)*, 33
getArea *(method)*, 33
getCellID *(method)*, 33
getCenter *(method)*, 33
getID *(method)*, 33
fipy.meshes.numMesh.gmshExport *(module)*, 34
exportAsMesh *(function)*, 34
getElementType *(function)*, 34
orderVertices *(function)*, 34
fipy.meshes.numMesh.gmshImport *(module)*, 35–41
DataGetter *(class)*, 39–40
computeBaseFaceVertexIDs *(method)*, 39
computeCellFaceIDs *(method)*, 39
computeCellVertexIDs *(method)*, 39
computeFaceVertexIDs *(method)*, 40
computeVertexCoords *(method)*, 40
getData *(method)*, 40
GmshImporter2D *(class)*, 40
__init__ *(method)*, 40
GmshImporter3D *(class)*, 40–41
__init__ *(method)*, 41
listToString *(function)*, 39
stringToList *(function)*, 39
fipy.meshes.numMesh.grid2D *(module)*, 42–43
Grid2D *(class)*, 42–43
__getstate__ *(method)*, 42
__init__ *(method)*, 42
__setstate__ *(method)*, 42
createCells *(method)*, 42
createFaces *(method)*, 42
createVertices *(method)*, 42
getFacesBottom *(method)*, 42
getFacesLeft *(method)*, 42
getFacesRight *(method)*, 43
getFacesTop *(method)*, 43
getMeshSpacing *(method)*, 43
getPhysicalShape *(method)*, 43
getScale *(method)*, 43
getShape *(method)*, 43
fipy.meshes.numMesh.grid3D *(module)*, 44–46
Grid3D *(class)*, 44–46
__getstate__ *(method)*, 44
__init__ *(method)*, 44
__setstate__ *(method)*, 44
calcFaceAreas *(method)*, 44
calcFaceNormals *(method)*, 44
calcFaceTangents *(method)*, 44
calcHigherOrderScalings *(method)*, 44
createCells *(method)*, 45
createFaces *(method)*, 45
createVertices *(method)*, 45
getFacesBack *(method)*, 45
getFacesBottom *(method)*, 45
getFacesFront *(method)*, 45
getFacesLeft *(method)*, 45
getFacesRight *(method)*, 45
getFacesTop *(method)*, 45
getMeshSpacing *(method)*, 45
getPhysicalShape *(method)*, 45
getScale *(method)*, 45

getShape *(method)*, 45
repeatWithOffset *(method)*, 46
fipy.meshes.numMesh.mesh *(module)*, 47–54
MAtake *(function)*, 50
Mesh *(class)*, 50–54
__add__ *(method)*, 51
__getstate__ *(method)*, 51
__init__ *(method)*, 51
__mul__ *(method)*, 51
__setstate__ *(method)*, 51
calcAdjacentCellIDs *(method)*, 51
calcAreaProjections *(method)*, 51
calcCellCenters *(method)*, 51
calcCellDistances *(method)*, 51
calcCellNormals *(method)*, 51
calcCellToCellDistances *(method)*, 51
calcCellToCellIDs *(method)*, 51
calcCellToFaceOrientations *(method)*, 51
calcCellVolumes *(method)*, 52
calcFaceAreas *(method)*, 52
calcFaceCellIDs *(method)*, 52
calcFaceCenters *(method)*, 52
calcFaceNormals *(method)*, 52
calcFaceTangents *(method)*, 52
calcFaceToCellDistanceRatio *(method)*, 52
calcFaceToCellDistances *(method)*, 52
calcGeometry *(method)*, 52
calcInteriorAndExteriorCellIDs *(method)*, 52
calcInteriorAndExteriorFaceIDs *(method)*, 52
calcOrientedAreaProjections *(method)*, 52
calcOrientedFaceNormals *(method)*, 52
calcTopology *(method)*, 52
dilate *(method)*, 52
equalExceptOrder *(method)*, 53
getAddedMeshValues *(method)*, 53
getCellsByID *(method)*, 53
getExteriorFaces *(method)*, 53
getFaceCellIDs *(method)*, 53
getFaceCenters *(method)*, 53
getFaces *(method)*, 53
getInteriorFaces *(method)*, 53
getMaxFacesPerCell *(method)*, 53
getVertexCoords *(method)*, 53
meshAdd *(method)*, 53
translate *(method)*, 53
fipy.meshes.numMesh.mesh2D *(module)*, 55–56
Mesh2D *(class)*, 55–56
calcFaceAreas *(method)*, 55
calcFaceNormals *(method)*, 55
calcFaceTangents *(method)*, 55
calcHigherOrderScalings *(method)*, 55
dilate *(method)*, 55
getNonOrthogonality *(method)*, 55
getOrderedCellVertexIDs *(method)*, 55
meshAdd *(method)*, 55
translate *(method)*, 56
orderVertices *(function)*, 55
fipy.meshes.numMesh.refinedMesh *(module)*, 57–59
listToString *(function)*, 58
RefinedMesh2D *(class)*, 58–59
__init__ *(method)*, 58
RefinedMeshCellVariable *(class)*, 59
__init__ *(method)*, 59
fipy.meshes.numMesh.skewedGrid2D *(module)*, 60–61
SkewedGrid2D *(class)*, 60–61
__getstate__ *(method)*, 60
__init__ *(method)*, 60
__setstate__ *(method)*, 60
createCells *(method)*, 60
createFaces *(method)*, 60
createVertices *(method)*, 60
getFacesBottom *(method)*, 60
getFacesLeft *(method)*, 61
getFacesRight *(method)*, 61
getFacesTop *(method)*, 61
getMeshSpacing *(method)*, 61
getPhysicalShape *(method)*, 61
getScale *(method)*, 61
getShape *(method)*, 61
fipy.meshes.numMesh.test *(module)*, 62
suite *(function)*, 62
fipy.meshes.numMesh.testGrid *(module)*, 63–65
suite *(function)*, 63
TestGrid *(class)*, 63–64

setUp *(method)*, 63
testCells *(method)*, 63
testFaces *(method)*, 63
testFacesBottom *(method)*, 63
testFacesLeft *(method)*, 63
testFacesRight *(method)*, 63
testFacesTop *(method)*, 63
testVertices *(method)*, 63
TestGridPickle *(class)*, 64–65
setUp *(method)*, 64
fipy.meshes.numMesh.testGrid3D *(module)*, 66–68
suite *(function)*, 66
TestGrid *(class)*, 66–67
setUp *(method)*, 66
testCells *(method)*, 66
testFaces *(method)*, 66
testFacesBack *(method)*, 66
testFacesBottom *(method)*, 66
testFacesFront *(method)*, 66
testFacesLeft *(method)*, 66
testFacesRight *(method)*, 66
testFacesTop *(method)*, 67
testVertices *(method)*, 67
TestGridPickle *(class)*, 67–68
setUp *(method)*, 67
fipy.meshes.numMesh.testMesh *(module)*, 69–70
suite *(function)*, 69
TestMesh *(class)*, 69
setUp *(method)*, 69
TestMeshPickle *(class)*, 69–70
setUp *(method)*, 70
fipy.meshes.numMesh.testMesh3D *(module)*, 71–72
suite *(function)*, 71
TestMesh3D *(class)*, 71
setUp *(method)*, 71
TestMesh3DPickle *(class)*, 71–72
setUp *(method)*, 72
fipy.meshes.numMesh.testMeshBase *(module)*, 73–74
TestMeshBase *(class)*, 73–74
testAreaProjections *(method)*, 73
testCellAreaProjections *(method)*, 73
testCellCenters *(method)*, 73
testCellDistances *(method)*, 73
testCellNormals *(method)*, 73
testCellToCellDistances *(method)*, 73
testCellToCellIDs *(method)*, 73
testCellToFaceOrientations *(method)*, 73
testCellVolumes *(method)*, 73
testExteriorCellIDs *(method)*, 73
testExteriorFaces *(method)*, 73
testFaceAreas *(method)*, 73
testFaceCellIds *(method)*, 73
testFaceCenters *(method)*, 74
testFaceNormals *(method)*, 74
testFaceToCellDistanceRatios *(method)*, 74
testFaceToCellDistances *(method)*, 74
testInteriorCellIDs *(method)*, 74
testInternalFaces *(method)*, 74
testResult *(method)*, 74
testTangents1 *(method)*, 74
testTangents2 *(method)*, 74
fipy.meshes.numMesh.testTri2D *(module)*, 75–76
suite *(function)*, 75
TestTri2D *(class)*, 75–76
setUp *(method)*, 75
testCellAreaProjections *(method)*, 75
testCellNormals *(method)*, 75
testCells *(method)*, 75
testFaces *(method)*, 75
testFacesBottom *(method)*, 75
testFacesLeft *(method)*, 75
testFacesRight *(method)*, 75
testFacesTop *(method)*, 76
testVertices *(method)*, 76
fipy.meshes.numMesh.tri2D *(module)*, 77–79
Tri2D *(class)*, 77–79
__getstate__ *(method)*, 78
__init__ *(method)*, 78
__setstate__ *(method)*, 78
createCells *(method)*, 78
createFaces *(method)*, 78
createVertices *(method)*, 78
getFacesBottom *(method)*, 78
getFacesLeft *(method)*, 78
getFacesRight *(method)*, 78
getFacesTop *(method)*, 79

getMeshSpacing *(method)*, 79
getPhysicalShape *(method)*, 79
getScale *(method)*, 79
getShape *(method)*, 79
fipy.meshes.pyMesh.cell *(module)*, 80–81
Cell *(class)*, 80–81
__init__ *(method)*, 80
__repr__ *(method)*, 80
calcCenter *(method)*, 80
calcFaceIDs *(method)*, 80
calcVolume *(method)*, 80
getBoundingCells *(method)*, 80
getCenter *(method)*, 81
getFaceIDs *(method)*, 81
getFaceOrientations *(method)*, 81
getFaces *(method)*, 81
getID *(method)*, 81
getVolume *(method)*, 81
setID *(method)*, 81
fipy.meshes.pyMesh.face *(module)*, 82–84
Face *(class)*, 82–84
__init__ *(method)*, 82
__repr__ *(method)*, 82
addBoundingCell *(method)*, 82
calcArea *(method)*, 82
calcCellDistance *(method)*, 82
calcCenter *(method)*, 82
calcFaceToCellDistance *(method)*, 82
calcNormal *(method)*, 83
calcTangent1 *(method)*, 83
calcTangent2 *(method)*, 83
getArea *(method)*, 83
getCellDistance *(method)*, 83
getCellID *(method)*, 83
getCells *(method)*, 83
getCenter *(method)*, 83
getFaceToCellDistance *(method)*, 83
getID *(method)*, 83
getNormal *(method)*, 83
getOrientation *(method)*, 83
removeBoundingCell *(method)*, 83
setCellDistance *(method)*, 84
setFaceToCellDistances *(method)*, 84
setID *(method)*, 84
setNormal *(method)*, 84
fipy.meshes.pyMesh.face2D *(module)*, 85
Face2D *(class)*, 85
calcArea *(method)*, 85
calcNormal *(method)*, 85
calcTangent1 *(method)*, 85
calcTangent2 *(method)*, 85
fipy.meshes.pyMesh.grid2D *(module)*, 86–89
Grid2D *(class)*, 86–89
__init__ *(method)*, 87
createCells *(method)*, 87
createFaces *(method)*, 87
createInteriorFaces *(method)*, 87
createVertices *(method)*, 88
getCellCenters *(method)*, 88
getCellDistances *(method)*, 88
getCellVolumes *(method)*, 88
getFaceAreas *(method)*, 88
getFacesBottom *(method)*, 88
getFacesLeft *(method)*, 88
getFacesRight *(method)*, 88
getFacesTop *(method)*, 88
getFaceToCellDistances *(method)*, 88
getMaxFacesPerCell *(method)*, 88
getMeshSpacing *(method)*, 88
getPhysicalShape *(method)*, 88
getShape *(method)*, 89
makeGridData *(method)*, 89
reorderFaces *(method)*, 89
fipy.meshes.pyMesh.mesh *(module)*, 90–92
Mesh *(class)*, 90–92
__init__ *(method)*, 90
calcAdjacentCellIDs *(method)*, 90
calcAreaProjections *(method)*, 90
calcCellCenters *(method)*, 90
calcCellDistances *(method)*, 90
calcCellFaceIDs *(method)*, 90
calcCellToCellDistances *(method)*, 90
calcCellToCellIDs *(method)*, 90
calcCellToFaceOrientations *(method)*, 90
calcCellVolumes *(method)*, 90
calcExteriorCellIDs *(method)*, 91
calcFaceAreas *(method)*, 91
calcFaceNormals *(method)*, 91
calcFaceOrientations *(method)*, 91
calcFaceTangents *(method)*, 91
calcFaceToCellDistanceRatio *(method)*, 91

calcFaceToCellDistances *(method)*, 91
calcInteriorAndExteriorFaceIDs *(method)*, 91
calcOrientedAreaProjections *(method)*, 91
calcOrientedFaceNormals *(method)*, 91
calcTopology *(method)*, 91
getCellsByID *(method)*, 91
getExteriorFaces *(method)*, 91
getFaceOrientations *(method)*, 91
getFaces *(method)*, 92
getInteriorFaces *(method)*, 92
getPhysicalShape *(method)*, 92
getScale *(method)*, 92
makeGridData *(method)*, 92
setScale *(method)*, 92
fipy.meshes.pyMesh.test *(module)*, 93
suite *(function)*, 93
fipy.meshes.pyMesh.vertex *(module)*, 94
Vertex *(class)*, 94
__init__ *(method)*, 94
__repr__ *(method)*, 94
getCoordinates *(method)*, 94
fipy.models.cahnHilliard.cahnHilliardEquation *(module)*, 95–96
CahnHilliardEquation *(class)*, 95–96
__init__ *(method)*, 96
fipy.models.elphf.componentVariable *(module)*, 97–98
ComponentVariable *(class)*, 97–98
__init__ *(method)*, 97
copy *(method)*, 97
getBarrierHeight *(method)*, 97
getDiffusivity *(method)*, 97
getStandardPotential *(method)*, 97
getValence *(method)*, 97
fipy.models.elphf.concentrationEquation *(module)*, 99
ConcentrationEquation *(class)*, 99
__init__ *(method)*, 99
getConvectionCoeff *(method)*, 99
fipy.models.elphf.elphf *(module)*, 100
addScales *(function)*, 100
makeEquations *(function)*, 100
makeFields *(function)*, 100
fipy.models.elphf.elphfIterator *(module)*, 101
ElPhFIterator *(class)*, 101
__init__ *(method)*, 101
sweep *(method)*, 101
fipy.models.elphf.interstitialEquation *(module)*, 102
InterstitialEquation *(class)*, 102
getConvectionCoeff *(method)*, 102
fipy.models.elphf.phaseEquation *(module)*, 103
PhaseEquation *(class)*, 103
__init__ *(method)*, 103
fipy.models.elphf.phaseVariable *(module)*, 104
PhaseVariable *(class)*, 104
__init__ *(method)*, 104
get_g *(method)*, 104
get_gPrime *(method)*, 104
get_p *(method)*, 104
get_pPrime *(method)*, 104
fipy.models.elphf.poissonEquation *(module)*, 105
PoissonEquation *(class)*, 105
__init__ *(method)*, 105
getConcentration *(method)*, 105
fipy.models.elphf.scaledCellVariable *(module)*, 106
ScaledCellVariable *(class)*, 106
__init__ *(method)*, 106
getScale *(method)*, 106
getScaled *(method)*, 106
setValue *(method)*, 106
fipy.models.elphf.semiImplicitPoissonEquation *(module)*, 107
SemiImplicitPoissonEquation *(class)*, 107
getConcentration *(method)*, 107
fipy.models.elphf.solventVariable *(module)*, 108
SolventVariable *(class)*, 108
__init__ *(method)*, 108
fipy.models.elphf.substitutionalEquation *(module)*, 109–110
SubstitutionalEquation *(class)*, 109–110
getConvectionCoeff *(method)*, 109
fipy.models.elphf.substitutionalVariable *(module)*, 111–112
SubstitutionalVariable *(class)*, 111–112
__init__ *(method)*, 111
copy *(method)*, 111
fipy.models.elphf.test *(module)*, 113–121
suite *(function)*, 113

TestElPhF *(class)*, 113–114
assertFieldWithinTolerance *(method)*, 113
setUp *(method)*, 113
testResult *(method)*, 113
TestElPhF1D *(class)*, 114
setUp *(method)*, 114
TestElPhF1Dphase *(class)*, 114–115
setUp *(method)*, 115
testResult *(method)*, 115
TestElPhF1DphaseBinary *(class)*, 115–116
setUp *(method)*, 116
TestElPhF1DphaseQuaternary *(class)*, 116
setUp *(method)*, 116
TestElPhF1DphaseTernaryAndElectrons *(class)*, 116–117
setUp *(method)*, 117
TestElPhF1DpoissonAllCharge *(class)*, 117–118
setUp *(method)*, 118
testResult *(method)*, 118
TestElPhF1DpoissonLeftCharge *(class)*, 118–119
setUp *(method)*, 118
testResult *(method)*, 118
TestElPhF1DpoissonRightCharge *(class)*, 119
setUp *(method)*, 119
testResult *(method)*, 119
TestElPhF2D *(class)*, 119–120
setUp *(method)*, 120
TestElPhF2Dcorner *(class)*, 120–121
setUp *(method)*, 121
fipy.models.levelSet.advection.advectionEquation *(module)*, 122
AdvectionEquation *(class)*, 122
__init__ *(method)*, 122
fipy.models.levelSet.advection.advectionTerm *(module)*, 123–124
AdvectionTerm *(class)*, 123–124
__init__ *(method)*, 124
buildMatrix *(method)*, 124
getDifferences *(method)*, 124
fipy.models.levelSet.advection.higherOrderAdvectionEquation *(module)*, 125
HigherOrderAdvectionEquation *(class)*, 125
__init__ *(method)*, 125
fipy.models.levelSet.advection.higherOrderAdvectionTerm *(module)*, 126–128
HigherOrderAdvectionTerm *(class)*, 128
getDifferences *(method)*, 128
fipy.models.levelSet.distanceFunction.distanceEquation *(module)*, 129–130
DistanceEquation *(class)*, 129–130
__init__ *(method)*, 130
solve *(method)*, 130
fipy.models.levelSet.distanceFunction.distanceVariable *(module)*, 131–135
DistanceVariable *(class)*, 131–135
getCellInterfaceAreas *(method)*, 132
getCellInterfaceFlag *(method)*, 132
getCellInterfaceNormals *(method)*, 133
getCellValueOverFaces *(method)*, 133
getDifferences *(method)*, 133
getInterfaceFlag *(method)*, 133
getInterfaceNormals *(method)*, 134
getLevelSetNormals *(method)*, 134
getUpwindMag *(method)*, 134
fipy.models.levelSet.distanceFunction.extensionEquation *(module)*, 136–137
ExtensionEquation *(class)*, 136–137
__init__ *(method)*, 137
solve *(method)*, 137
fipy.models.levelSet.distanceFunction.levelSetDiffusionEquatio *(module)*, 138
LevelSetDiffusionEquation *(class)*, 138
__init__ *(method)*, 138
fipy.models.levelSet.distanceFunction.levelSetDiffusionVariable *(module)*, 139–140
LevelSetDiffusionVariable *(class)*, 139–140
__init__ *(method)*, 139
fipy.models.levelSet.electroChem.depositionRateVariable *(module)*, 141–143
DepositionRateVariable *(class)*, 142–143
__init__ *(method)*, 142
getCurrentDensity *(method)*, 142
getExpoConstant *(method)*, 142
getTransferCoefficient *(method)*, 142
setOverpotential *(method)*, 142
fipy.models.levelSet.electroChem.metalIonDiffusionEquation *(module)*, 144–145
MetalIonDiffusionEquation *(class)*, 145
__init__ *(method)*, 145

fipy.models.levelSet.electroChem.metalIonSourceVariable *(module)*, 146–147
MetalIonSourceVariable *(class)*, 146–147
__init__ *(method)*, 146
fipy.models.levelSet.electroChem.test *(module)*, 148
suite *(function)*, 148
fipy.models.levelSet.surfactant.adsorbingSurfactantEquation *(module)*, 149–152
AdsorbingSurfactantEquation *(class)*, 149–150
__init__ *(method)*, 150
solve *(method)*, 150
AdsorptionCoeff *(class)*, 150–151
__init__ *(method)*, 150
updateDt *(method)*, 150
AdsorptionCoeffAreaOverVolume *(class)*, 151
multiplier *(method)*, 151
AdsorptionCoeffInterfaceFlag *(class)*, 151–152
multiplier *(method)*, 152
fipy.models.levelSet.surfactant.convectionCoeff *(module)*, 153–155
ConvectionCoeff *(class)*, 153–155
__init__ *(method)*, 154
fipy.models.levelSet.surfactant.surfactantBulkDiffusionEquation *(module)*, 156–158
AdsorptionCoeff *(class)*, 156–157
__init__ *(method)*, 156
ScAdsorptionCoeff *(class)*, 157–158
__init__ *(method)*, 157
SurfactantBulkDiffusionEquation *(class)*, 158
__init__ *(method)*, 158
fipy.models.levelSet.surfactant.surfactantEquation *(module)*, 159
SurfactantEquation *(class)*, 159
__init__ *(method)*, 159
solve *(method)*, 159
fipy.models.levelSet.surfactant.surfactantVariable *(module)*, 160–162
InterfaceSurfactantVariable *(class)*, 160
__init__ *(method)*, 160
SurfactantVariable *(class)*, 160–162
__init__ *(method)*, 161
getDistanceVar *(method)*, 161
getInterfaceVar *(method)*, 161
fipy.models.levelSet.test *(module)*, 163
suite *(function)*, 163
fipy.models.phase.phase.addOverFacesVariable *(module)*, 164
AddOverFacesVariable *(class)*, 164
__init__ *(method)*, 164
fipy.models.phase.phase.anisotropyVariable *(module)*, 165–167
AnisotropyVariable *(class)*, 165
__init__ *(method)*, 165
DPhiReverse *(class)*, 165–166
__init__ *(method)*, 166
FFVariable *(class)*, 166–167
__init__ *(method)*, 166
fipy.models.phase.phase.mPhiVariable *(module)*, 168
MPhiVariable *(class)*, 168
__init__ *(method)*, 168
fipy.models.phase.phase.phaseDiffusionVariable *(module)*, 169
PhaseDiffusionVariable *(class)*, 169
__init__ *(method)*, 169
fipy.models.phase.phase.phaseEquation *(module)*, 170
PhaseEquation *(class)*, 170
__init__ *(method)*, 170
fipy.models.phase.phase.phaseHalfAngleVariable *(module)*, 171
PhaseHalfAngleVariable *(class)*, 171
__init__ *(method)*, 171
fipy.models.phase.phase.scSourceVariable *(module)*, 172
ScSourceVariable *(class)*, 172
__init__ *(method)*, 172
fipy.models.phase.phase.spSourceVariable *(module)*, 173
SpSourceVariable *(class)*, 173
__init__ *(method)*, 173
fipy.models.phase.phase.type1MPhiVariable *(module)*, 174
Type1MPhiVariable *(class)*, 174
fipy.models.phase.phase.type2MPhiVariable *(module)*, 175
Type2MPhiVariable *(class)*, 175

fipy.models.phase.temperature.temperatureEquation *(module)*, 176
TemperatureEquation *(class)*, 176
__init__ *(method)*, 176
fipy.models.phase.test *(module)*, 177
suite *(function)*, 177
fipy.models.phase.theta.diffusionVariable *(module)*, 178
DiffusionVariable *(class)*, 178
__init__ *(method)*, 178
fipy.models.phase.theta.modCellGradVariable *(module)*, 179
ModCellGradVariable *(class)*, 179
__init__ *(method)*, 179
fipy.models.phase.theta.modCellToFaceVariable *(module)*, 180
ModCellToFaceVariable *(class)*, 180
__init__ *(method)*, 180
fipy.models.phase.theta.modFaceGradVariable *(module)*, 181
ModFaceGradVariable *(class)*, 181
__init__ *(method)*, 181
fipy.models.phase.theta.modPhysicalField *(module)*, 182
ModPhysicalField *(class)*, 182
__rsub__ *(method)*, 182
__sub__ *(method)*, 182
mod *(method)*, 182
fipy.models.phase.theta.modularVariable *(module)*, 183
ModularVariable *(class)*, 183
__init__ *(method)*, 183
getArithmeticFaceValue *(method)*, 183
getFaceGrad *(method)*, 183
getGrad *(method)*, 183
updateOld *(method)*, 183
fipy.models.phase.theta.noModularVariable *(module)*, 184
NoModularVariable *(class)*, 184
__init__ *(method)*, 184
fipy.models.phase.theta.sourceVariable *(module)*, 185
SourceVariable *(class)*, 185
__init__ *(method)*, 185
fipy.models.phase.theta.thetaEquation *(module)*, 186
ThetaEquation *(class)*, 186
__init__ *(method)*, 186
fipy.models.phase.theta.thetaHalfAngleVariable *(module)*, 187
ThetaHalfAngleVariable *(class)*, 187
__init__ *(method)*, 187
fipy.models.phase.theta.transientVariable *(module)*, 188
TransientVariable *(class)*, 188
__init__ *(method)*, 188
fipy.models.test *(module)*, 189
suite *(function)*, 189
fipy.solvers.linearCGSSolver *(module)*, 191
LinearCGSSolver *(class)*, 191
__init__ *(method)*, 191
solve *(method)*, 191
fipy.solvers.linearGMRESSolver *(module)*, 192
LinearGMRESSolver *(class)*, 192
__init__ *(method)*, 192
solve *(method)*, 192
fipy.solvers.linearLUSolver *(module)*, 193
LinearLUSolver *(class)*, 193
__init__ *(method)*, 193
solve *(method)*, 193
fipy.solvers.linearPCGSolver *(module)*, 194
LinearPCGSolver *(class)*, 194
__init__ *(method)*, 194
solve *(method)*, 194
fipy.solvers.linearScipyLUSolver *(module)*, 195
LinearScipyLUSolver *(class)*, 195
__init__ *(method)*, 195
solve *(method)*, 195
fipy.solvers.linearScipySolver *(module)*, 196
LinearScipySolver *(class)*, 196
__init__ *(method)*, 196
solve *(method)*, 196
fipy.solvers.solver *(module)*, 197
Solver *(class)*, 197
__init__ *(method)*, 197
solve *(method)*, 197
fipy.terms.cellTerm *(module)*, 199
CellTerm *(class)*, 199
__init__ *(method)*, 199
buildMatrix *(method)*, 199
fipy.terms.centralDiffConvectionTerm *(module)*, 200

CentralDifferenceConvectionTerm *(class)*, 200
fipy.terms.convectionTerm *(module)*, 201
ConvectionTerm *(class)*, 201
__init__ *(method)*, 201
getWeight *(method)*, 201
fipy.terms.diffusionTerm *(module)*, 202
DiffusionTerm *(class)*, 202
__init__ *(method)*, 202
getCoeff *(method)*, 202
fipy.terms.explicitDiffusionTerm *(module)*, 203
ExplicitDiffusionTerm *(class)*, 203
__init__ *(method)*, 203
fipy.terms.explicitUpwindConvectionTerm *(module)*, 204
ExplicitUpwindConvectionTerm *(class)*, 204
getWeight *(method)*, 204
fipy.terms.exponentialConvectionTerm *(module)*, 205
ExponentialConvectionTerm *(class)*, 205
fipy.terms.faceTerm *(module)*, 206
FaceTerm *(class)*, 206
__init__ *(method)*, 206
buildMatrix *(method)*, 206
explicitBuildMatrix *(method)*, 206
getOldAdjacentValues *(method)*, 206
implicitBuildMatrix *(method)*, 206
fipy.terms.hybridConvectionTerm *(module)*, 207
HybridConvectionTerm *(class)*, 207
fipy.terms.implicitDiffusionTerm *(module)*, 208
ImplicitDiffusionTerm *(class)*, 208
__init__ *(method)*, 208
fipy.terms.nthOrderDiffusionTerm *(module)*, 209–211
NthOrderDiffusionTerm *(class)*, 211
__init__ *(method)*, 211
buildMatrix *(method)*, 211
getCoefficientMatrix *(method)*, 211
fipy.terms.powerLawConvectionTerm *(module)*, 212
PowerLawConvectionTerm *(class)*, 212
fipy.terms.scSourceTerm *(module)*, 213
ScSourceTerm *(class)*, 213
__init__ *(method)*, 213
fipy.terms.sourceTerm *(module)*, 214
SourceTerm *(class)*, 214
__init__ *(method)*, 214
fipy.terms.spSourceTerm *(module)*, 215
SpSourceTerm *(class)*, 215
__init__ *(method)*, 215
fipy.terms.term *(module)*, 216
Term *(class)*, 216
__init__ *(method)*, 216
buildMatrix *(method)*, 216
calcCoeffScale *(method)*, 216
getCoeffScale *(method)*, 216
getFigureOfMerit *(method)*, 216
getMesh *(method)*, 216
fipy.terms.test *(module)*, 217
suite *(function)*, 217
fipy.terms.transientTerm *(module)*, 218
TransientTerm *(class)*, 218
__init__ *(method)*, 218
fipy.terms.upwindConvectionTerm *(module)*, 219
UpwindConvectionTerm *(class)*, 219
fipy.terms.vanLeerConvectionTerm *(module)*, 220
VanLeerConvectionTerm *(class)*, 220
getFigureOfMerit *(method)*, 220
getGradient *(method)*, 220
getOldAdjacentValues *(method)*, 220
fipy.test *(module)*, 221
suite *(function)*, 221
fipy.tests.blinky *(module)*, 223
fipy.tests.doctestPlus *(module)*, 224
getScript *(function)*, 224
fipy.tests.testBase *(module)*, 225
TestBase *(class)*, 225
assertArrayEqual *(method)*, 225
assertArrayWithinTolerance *(method)*, 225
assertWithinTolerance *(method)*, 225
getTestValue *(method)*, 225
getTestValues *(method)*, 225
testResult *(method)*, 225
fipy.tests.testProgram *(module)*, 226–227
TestProgram *(class)*, 226–227
parseArgs *(method)*, 226
fipy.tools.array *(module)*, 229–230
allclose *(function)*, 229
allequal *(function)*, 229
arctan *(function)*, 229

arctan2 *(function)*, 229
crossProd *(function)*, 229
dot *(function)*, 229
put *(function)*, 229
reshape *(function)*, 229
sqrt *(function)*, 229
sqrtDot *(function)*, 229
sum *(function)*, 230
take *(function)*, 230
tan *(function)*, 230
fipy.tools.dimensions.DictWithDefault *(module)*, 231
DictWithDefault *(class)*, 231
__delitem__ *(method)*, 231
__getitem__ *(method)*, 231
__init__ *(method)*, 231
fipy.tools.dimensions.NumberDict *(module)*, 232–233
NumberDict *(class)*, 232–233
__add__ *(method)*, 232
__coerce__ *(method)*, 232
__div__ *(method)*, 232
__init__ *(method)*, 232
__mul__ *(method)*, 232
__repr__ *(method)*, 232
__rsub__ *(method)*, 232
__str__ *(method)*, 232
__sub__ *(method)*, 232
fipy.tools.dimensions.physicalField *(module)*, 234–253
PhysicalField *(class)*, 234–245
__abs__ *(method)*, 235
__add__ *(method)*, 236, 239
__array__ *(method)*, 236
__div__ *(method)*, 236
__eq__ *(method)*, 237
__float__ *(method)*, 237
__ge__ *(method)*, 237
__getitem__ *(method)*, 237
__gt__ *(method)*, 237
__init__ *(method)*, 235
__le__ *(method)*, 238
__len__ *(method)*, 238
__lt__ *(method)*, 238
__mod__ *(method)*, 238
__mul__ *(method)*, 238, 240
__ne__ *(method)*, 239
__neg__ *(method)*, 239
__nonzero__ *(method)*, 239
__pos__ *(method)*, 239
__pow__ *(method)*, 239
__rdiv__ *(method)*, 240
__repr__ *(method)*, 240
__rpow__ *(method)*, 240
__rsub__ *(method)*, 240
__setitem__ *(method)*, 240
__str__ *(method)*, 241
__sub__ *(method)*, 241
allclose *(method)*, 241
arctan *(method)*, 241
arctan2 *(method)*, 242
convertToUnit *(method)*, 242
copy *(method)*, 242
cos *(method)*, 242
dot *(method)*, 242
getNumericValue *(method)*, 243
getUnit *(method)*, 243
inBaseUnits *(method)*, 243
inUnitsOf *(method)*, 243
isCompatible *(method)*, 243
put *(method)*, 243
reshape *(method)*, 244
sin *(method)*, 244
sqrt *(method)*, 244
sum *(method)*, 245
take *(method)*, 245
tan *(method)*, 245
PhysicalUnit *(class)*, 245–253
__cmp__ *(method)*, 246
__div__ *(method)*, 246
__init__ *(method)*, 246
__mul__ *(method)*, 247, 249
__pow__ *(method)*, 248
__rdiv__ *(method)*, 249
__repr__ *(method)*, 249, 250
conversionFactorTo *(method)*, 250
conversionTupleTo *(method)*, 251
isAngle *(method)*, 251
isCompatible *(method)*, 251
isDimensionless *(method)*, 252
isDimensionlessOrAngle *(method)*, 252
name *(method)*, 252

setName *(method)*, 252
Scale *(function)*, 234
fipy.tools.dump *(module)*, 254
read *(function)*, 254
write *(function)*, 254
fipy.tools.inline.inline *(module)*, 255
optionalInline *(function)*, 255
runInline *(function)*, 255
runInlineLoop *(function)*, 255
runInlineLoop1 *(function)*, 255
runInlineLoop2 *(function)*, 255
runInlineLoop3 *(function)*, 255
fipy.tools.memoryLeak *(module)*, 256
get_refcounts *(function)*, 256
print_top_N *(function)*, 256
fipy.tools.sparseMatrix *(module)*, 257–260
SparseIdentityMatrix *(class)*, 257
__init__ *(method)*, 257
SparseMatrix *(class)*, 257–260
__add__ *(method)*, 257
__array__ *(method)*, 258
__getitem__ *(method)*, 258
__init__ *(method)*, 257
__mul__ *(method)*, 258
__neg__ *(method)*, 258
__pos__ *(method)*, 258
__repr__ *(method)*, 258
__rmul__ *(method)*, 258
__setitem__ *(method)*, 258
__str__ *(method)*, 259
__sub__ *(method)*, 259
addAt *(method)*, 259
addAtDiagonal *(method)*, 259
copy *(method)*, 259
getMatrix *(method)*, 259
getShape *(method)*, 259
put *(method)*, 259
putDiagonal *(method)*, 259
take *(method)*, 260
takeDiagonal *(method)*, 260
transpose *(method)*, 260
fipy.tools.test *(module)*, 261–262
suite *(function)*, 261
Test10by10 *(class)*, 261
setUp *(method)*, 261
Test50by50 *(class)*, 261–262
setUp *(method)*, 262
TestDump *(class)*, 262
assertEqual *(method)*, 262
setUp *(method)*, 262
testResult *(method)*, 262
fipy.tools.vector *(module)*, 263
crossProd *(function)*, 263
prune *(function)*, 263
putAdd *(function)*, 263
sqrtDot *(function)*, 263
fipy.variables.addOverFacesVariable *(module)*, 265–266
AddOverFacesVariable *(class)*, 265–266
__init__ *(method)*, 265
fipy.variables.arithmeticCellToFaceVariable *(module)*, 267
ArithmeticCellToFaceVariable *(class)*, 267
fipy.variables.cellGradVariable *(module)*, 268
CellGradVariable *(class)*, 268
__init__ *(method)*, 268
fipy.variables.cellToFaceVariable *(module)*, 269
CellToFaceVariable *(class)*, 269
__init__ *(method)*, 269
fipy.variables.cellVariable *(module)*, 270–273
CellVariable *(class)*, 270–273
__call__ *(method)*, 270
__getstate__ *(method)*, 270
__init__ *(method)*, 270
__setstate__ *(method)*, 270
copy *(method)*, 270
getArithmeticFaceValue *(method)*, 271
getCellVolumeAverage *(method)*, 271
getFaceDifference *(method)*, 271
getFaceGrad *(method)*, 271
getGrad *(method)*, 271
getGridArray *(method)*, 271
getHarmonicFaceValue *(method)*, 271
getLaplacian *(method)*, 271
getOld *(method)*, 271
getValue *(method)*, 271
getVariableClass *(method)*, 272
remesh *(method)*, 272
resetToOld *(method)*, 272
setValue *(method)*, 272
updateOld *(method)*, 272
ReMeshedCellVariable *(class)*, 273

__init__ *(method)*, 273
fipy.variables.cellVolumeAverageVariable *(module)*, 274
CellVolumeAverageVariable *(class)*, 274
__init__ *(method)*, 274
fipy.variables.faceDifferenceVariable *(module)*, 275
FaceDifferenceVariable *(class)*, 275
__init__ *(method)*, 275
fipy.variables.faceGradContributionsVariable *(module)*, 276
FaceGradContributions *(class)*, 276
__init__ *(method)*, 276
fipy.variables.faceGradVariable *(module)*, 277
FaceGradVariable *(class)*, 277
__init__ *(method)*, 277
fipy.variables.faceVariable *(module)*, 278
FaceVariable *(class)*, 278
__init__ *(method)*, 278
getVariableClass *(method)*, 278
transpose *(method)*, 278
fipy.variables.harmonicCellToFaceVariable *(module)*, 279
HarmonicCellToFaceVariable *(class)*, 279
fipy.variables.magVariable *(module)*, 280
MagVariable *(class)*, 280
__init__ *(method)*, 280
fipy.variables.sumVariable *(module)*, 281
SumVariable *(class)*, 281
__init__ *(method)*, 281
fipy.variables.test *(module)*, 282
suite *(function)*, 282
fipy.variables.testInterpolation *(module)*, 283–291
suite *(function)*, 283
TestArithmeticMean *(class)*, 283
setUp *(method)*, 283
TestArithmeticMean1 *(class)*, 283–284
setUp *(method)*, 284
TestArithmeticMean2 *(class)*, 284–285
setUp *(method)*, 285
TestArithmeticMean3 *(class)*, 285–286
setUp *(method)*, 286
TestHarmonicMean *(class)*, 286–287
setUp *(method)*, 287
TestHarmonicMean1 *(class)*, 287–288
setUp *(method)*, 287
TestHarmonicMean2 *(class)*, 288
setUp *(method)*, 288
TestHarmonicMean3 *(class)*, 288–289
setUp *(method)*, 289
TestMean *(class)*, 289–290
setUp *(method)*, 290
testResult *(method)*, 290
TestMesh *(class)*, 290–291
__init__ *(method)*, 290
createVertices *(method)*, 290
fipy.variables.testLaplacian *(module)*, 292–295
suite *(function)*, 292
TestLaplacian *(class)*, 292
setUp *(method)*, 292
testResult *(method)*, 292
TestLaplacian2 *(class)*, 292–293
getValue *(method)*, 293
setUp *(method)*, 293
TestLaplacian4 *(class)*, 293–294
getValue *(method)*, 294
setUp *(method)*, 294
TestLaplacian6 *(class)*, 294–295
getValue *(method)*, 294
setUp *(method)*, 294
fipy.variables.testPickle *(module)*, 296
suite *(function)*, 296
TestVariablePickle *(class)*, 296
setUp *(method)*, 296
testOldValue *(method)*, 296
testResult *(method)*, 296
testValue *(method)*, 296
fipy.variables.transposeVariable *(module)*, 297
TransposeVariable *(class)*, 297
__init__ *(method)*, 297
fipy.variables.variable *(module)*, 298–305
Variable *(class)*, 298–305
__abs__ *(method)*, 298
__add__ *(method)*, 298, 302
__array__ *(method)*, 299
__call__ *(method)*, 299
__div__ *(method)*, 299
__eq__ *(method)*, 299
__float__ *(method)*, 299
__ge__ *(method)*, 299
__getitem__ *(method)*, 300

__gt__ *(method)*, 300
__init__ *(method)*, 298
__le__ *(method)*, 300
__len__ *(method)*, 301
__lt__ *(method)*, 301
__mod__ *(method)*, 301
__mul__ *(method)*, 301, 302
__ne__ *(method)*, 301
__neg__ *(method)*, 302
__pos__ *(method)*, 302
__pow__ *(method)*, 302
__rdiv__ *(method)*, 302
__repr__ *(method)*, 302
__rpow__ *(method)*, 302
__rsub__ *(method)*, 302
__setitem__ *(method)*, 302
__str__ *(method)*, 302
__sub__ *(method)*, 302
allclose *(method)*, 302
arctan *(method)*, 302
copy *(method)*, 302
cos *(method)*, 303
dot *(method)*, 303
getBinaryOperatorVariable *(method)*, 303
getMag *(method)*, 303
getMesh *(method)*, 303
getNumericValue *(method)*, 303
getOperatorVariableClass *(method)*, 303
getUnaryOperatorVariable *(method)*, 303
getUnit *(method)*, 303
getValue *(method)*, 303
getVariableClass *(method)*, 304
inBaseUnits *(method)*, 304
markFresh *(method)*, 304
markStale *(method)*, 304
refresh *(method)*, 304
requiredBy *(method)*, 304
requires *(method)*, 304
setValue *(method)*, 304
sin *(method)*, 304
sqrt *(method)*, 304
sum *(method)*, 304
take *(method)*, 304
tan *(method)*, 304
transpose *(method)*, 304
fipy.variables.vectorArithmeticCellToFaceVariable *(module)*, 306
VectorArithmeticCellToFaceVariable *(class)*, 306
fipy.variables.vectorCellToFaceVariable *(module)*, 307
VectorCellToFaceVariable *(class)*, 307
__init__ *(method)*, 307
fipy.variables.vectorCellVariable *(module)*, 308
VectorCellVariable *(class)*, 308
__init__ *(method)*, 308
dot *(method)*, 308
getArithmeticFaceValue *(method)*, 308
getVariableClass *(method)*, 308
fipy.variables.vectorFaceVariable *(module)*, 309
VectorFaceVariable *(class)*, 309
__init__ *(method)*, 309
getVariableClass *(method)*, 309
fipy.viewers.colorbar *(module)*, 311
color_bar *(function)*, 311
nice_levels *(function)*, 311
fipy.viewers.gist1DViewer *(module)*, 312
Gist1DViewer *(class)*, 312
__init__ *(method)*, 312
getArrays *(method)*, 312
plot *(method)*, 312
plotArrays *(method)*, 312
fipy.viewers.gistVectorViewer *(module)*, 313
GistVectorViewer *(class)*, 313
__init__ *(method)*, 313
getArray *(method)*, 313
plot *(method)*, 313
fipy.viewers.gistViewer *(module)*, 314
GistViewer *(class)*, 314
__init__ *(method)*, 314
getArray *(method)*, 314
plot *(method)*, 314
fipy.viewers.gnuplotViewer *(module)*, 315
GnuplotViewer *(class)*, 315
__init__ *(method)*, 315
gnuplotCommands *(method)*, 315
plot *(method)*, 315
fipy.viewers.grid2DGistViewer *(module)*, 316–318
Grid2DGistViewer *(class)*, 316–318
__init__ *(method)*, 317

getArray *(method)*, 317
increaseResolution *(method)*, 317
setVar *(method)*, 317
fipy.viewers.numpyGistViewer *(module)*, 319
NumpyGistViewer *(class)*, 319
__init__ *(method)*, 319
getArray *(method)*, 319
fipy.viewers.numpyPyxViewer *(module)*, 320
NumpyPyxViewer *(class)*, 320
__init__ *(method)*, 320
plot *(method)*, 320
fipy.viewers.pyxviewer *(module)*, 321–325
Grid2DPyxViewer *(class)*, 324
__init__ *(method)*, 324
getValue *(method)*, 324
Grid3DPyxViewer *(class)*, 324–325
__init__ *(method)*, 325
generateArray *(method)*, 325
PyxViewer *(class)*, 325
__init__ *(method)*, 325
generateArray *(method)*, 325
getValue *(method)*, 325
getValueMatrix *(method)*, 325
plot *(method)*, 325
fipy.viewers.test *(module)*, 326
suite *(function)*, 326
fipy.viewers.varGistViewer *(module)*, 327
VarGistViewer *(class)*, 327
__init__ *(method)*, 327
getArray *(method)*, 327
setVar *(method)*, 327
fipy.viewers.viewer *(module)*, 328
gistViewer *(class)*, 328
__init__ *(method)*, 328
plot *(method)*, 328