EM Casing Research Documentation

Release 0.0.4

Lindsey Heagy

Jan 05, 2018

Contents

Lindsey Heagy

Software for running simuations of electromagnetic problems in the presence of steel cased wells.

The numerical simulation engine we rely on is SimPEG

Contents:

CHAPTER 1

Base

class casingSimulations.base.**BaseCasing**(***kwargs*)

Bases: properties.base.base.HasProperties

Base class that contains working directories, code version and can be saved

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string
- **version** (String): version of the software, a unicode string, Default: 0.0.4

copy()

Make a copy of the current casing object

directory

directory (String): Working directory, a unicode string, Default: .

filename

filename (String): Filename to which the properties are serialized and written to, a unicode string

save(*filename=None*, *directory=None*)

Save the casing properties to json :param str file: filename for saving the casing properties :param str directory: working directory for saving the file

version

version (String): version of the software, a unicode string, Default: 0.0.4

class casingSimulations.base.**LoadableInstance**(*doc*, *instance_class*, ***kwargs*)

Bases: properties.base.instance.Instance

class_info = 'an instance of a class or the name of a file from which the '

validate(*instance*, *value*)

CHAPTER 2

Model

class casingSimulations.model.**BaseCasingParametersMixin**(***kwargs*)

Bases: *casingSimulations.base.BaseCasing*

Parameters used to set up a casing in a background. This class does not function on its own. It should be mixed in with the background model of your choice

Required Properties:

- **casing_d** (Float): diameter of the casing (m), a float, Default: 0.1
- **casing_l** (Float): length of the casing (m), a float, Default: 1000
- **casing_t** (Float): thickness of the casing (m), a float, Default: 0.01
- **casing_top** (Float): top of the casing (m), a float, Default: 0.0
- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string
- **mur_casing** (Float): relative permeability of the casing, a float, Default: 100.0
- **sigma_casing** (Float): conductivity of the casing (S/m), a float, Default: 5500000.0
- **sigma_inside** (Float): conductivity of the fluid inside the casing (S/m), a float, Default: 1.0
- **version** (String): version of the software, a unicode string, Default: 0.0.4

add_mur_casing(*mesh*, *mur*)

add relative magnetic permeability of the casing to the provided model :param discretize.BaseMesh mesh: a discretize mesh :param numpy.ndarray mur: relative magnetic permittivity model to modify :rtype: numpy.ndarray :return: relative magnetic permeability model with casing

add_sigma_casing(*mesh*, *sigma*)

add the conductivity of the casing to the provided conductivity model :param discretize.BaseMesh mesh: a discretize mesh :param numpy.ndarray sigma: electrical conductivity model to modify :rtype: numpy.ndarray :return: electrical conductivity model with casing

casing_a

Inner casing radius

Return type float

casing_b
Outer casing radius

Return type float

casing_d
casing_d (`Float`): diameter of the casing (m), a float, Default: 0.1

casing_l
casing_l (`Float`): length of the casing (m), a float, Default: 1000

casing_r
Casing radius

Return type float

casing_t
casing_t (`Float`): thickness of the casing (m), a float, Default: 0.01

casing_top
casing_top (`Float`): top of the casing (m), a float, Default: 0.0

casing_z
z-extent of the casing

Return type numpy.array

ind_casing(*mesh*)
indices of the cell centers of the casing

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

ind_inside(*mesh*)
indices of the cell centers of the inside portion of the casing

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

indx_casing(*mesh*)
x-indices of the casing

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

indx_inside(*mesh*)
x indicies of the inside of the casing

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

indz_casing(*mesh*)
z-indices of the casing

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

info_casing

mur_casing
mur_casing (Float): relative permeability of the casing, a float, Default: 100.0

sigma_casing
sigma_casing (Float): conductivity of the casing (S/m), a float, Default: 5500000.0

sigma_inside
sigma_inside (Float): conductivity of the fluid inside the casing (S/m), a float, Default: 1.0

class casingSimulations.model.**CasingInHalfspace**(*filename=None*, ***kwargs*)
Bases: *casingSimulations.model.Halfspace*, *casingSimulations.model.BaseCasingParametersMixin*

A model of casing in a halfspace

Required Properties:

- **casing_d** (Float): diameter of the casing (m), a float, Default: 0.1
- **casing_l** (Float): length of the casing (m), a float, Default: 1000
- **casing_t** (Float): thickness of the casing (m), a float, Default: 0.01
- **casing_top** (Float): top of the casing (m), a float, Default: 0.0
- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json
- **mur_back** (Float): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0
- **mur_casing** (Float): relative permeability of the casing, a float, Default: 100.0
- **sigma_air** (Float): conductivity of the air (S/m), a float, Default: 1e-06
- **sigma_back** (Float): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01
- **sigma_casing** (Float): conductivity of the casing (S/m), a float, Default: 5500000.0
- **sigma_inside** (Float): conductivity of the fluid inside the casing (S/m), a float, Default: 1.0
- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]
- **surface_z** (Float): elevation of the air-earth interface (m), a float, Default: 0
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

mur(*mesh*)
put the permeability model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

sigma(*mesh*)
put the conductivity model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

class casingSimulations.model.**CasingInSingleLayer**(*filename=None*, ***kwargs*)

Bases: *casingSimulations.model.SingleLayer*, *casingSimulations.model.BaseCasingParametersMixin*

A model of casing in an earth that has a single layer

Required Properties:

- **casing_d** (Float): diameter of the casing (m), a float, Default: 0.1
- **casing_l** (Float): length of the casing (m), a float, Default: 1000
- **casing_t** (Float): thickness of the casing (m), a float, Default: 0.01
- **casing_top** (Float): top of the casing (m), a float, Default: 0.0
- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json
- **layer_z** (Array): z-limits of the layer, a list or numpy array of <type 'float'>, <type 'int'> with shape (2), Default: [-1000. -900.]
- **mur_back** (Float): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0
- **mur_casing** (Float): relative permeability of the casing, a float, Default: 100.0
- **sigma_air** (Float): conductivity of the air (S/m), a float, Default: 1e-06
- **sigma_back** (Float): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01
- **sigma_casing** (Float): conductivity of the casing (S/m), a float, Default: 5500000.0
- **sigma_inside** (Float): conductivity of the fluid inside the casing (S/m), a float, Default: 1.0
- **sigma_layer** (Float): conductivity of the layer (S/m), a float, Default: 0.01
- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]
- **surface_z** (Float): elevation of the air-earth interface (m), a float, Default: 0
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

mur(*mesh*)

put the permeability model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

sigma(*mesh*)

put the conductivity model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

class casingSimulations.model.**CasingInWholespace**(*filename=None*, ***kwargs*)
Bases: *casingSimulations.model.Wholespace*, *casingSimulations.model.BaseCasingParametersMixin*

A model of casing in a wholespace

Required Properties:

- **casing_d** (Float): diameter of the casing (m), a float, Default: 0.1
- **casing_l** (Float): length of the casing (m), a float, Default: 1000
- **casing_t** (Float): thickness of the casing (m), a float, Default: 0.01
- **casing_top** (Float): top of the casing (m), a float, Default: 0.0
- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json
- **mur_back** (Float): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0
- **mur_casing** (Float): relative permeability of the casing, a float, Default: 100.0
- **sigma_back** (Float): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01
- **sigma_casing** (Float): conductivity of the casing (S/m), a float, Default: 5500000.0
- **sigma_inside** (Float): conductivity of the fluid inside the casing (S/m), a float, Default: 1.0
- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

mur(*mesh*)
put the permeability model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

sigma(*mesh*)
put the conductivity model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – a discretize mesh

Return type numpy.array

class casingSimulations.model.**Halfspace**(*filename=None*, ***kwargs*)
Bases: *casingSimulations.model.Wholespace*

Model and survey parameters for an electromagnetic survey in a halfspace

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json
- **mur_back** (Float): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0
- **sigma_air** (Float): conductivity of the air (S/m), a float, Default: 1e-06
- **sigma_back** (Float): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01
- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]
- **surface_z** (Float): elevation of the air-earth interface (m), a float, Default: 0
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

ind_air(*mesh*)
indices where the air is

Parameters **mesh** (*discretize.BaseMesh*) – mesh to find the air cells of

Return type bool

info_model

sigma(*mesh*)
put the conductivity model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – mesh to find air cells of

Return type numpy.array

sigma_air
sigma_air (Float): conductivity of the air (S/m), a float, Default: 1e-06

surface_z
surface_z (Float): elevation of the air-earth interface (m), a float, Default: 0

class casingSimulations.model.**PhysicalProperties**(*meshGenerator*, *modelParameters*)
Bases: object

Physical properties on the mesh

model
model vector [sigma, mu]

Return type numpy.array

mu
permeability

Return type numpy.array

mur
relative permeability

Return type numpy.array

plot(*ax=None, clim=[None, None], pcolorOpts=None*)
plot the electrical conductivity and relative permeability

Parameters

- **ax** (*matplotlib.axes*) – axis
- **clim** (*list*) – list of numpy arrays: colorbar limits
- **pcolorOpts** (*dict*) – dictionary of pcolor options

plot_mur(*ax=None*, *clim=None*, *pcolorOpts=None*)
plot the relative permeability

Parameters

- **ax** (*matplotlib.axes*) – axis
- **clim** (*numpy.array*) – colorbar limits
- **pcolorOpts** (*dict*) – dictionary of pcolor options

plot_prop(*prop*, *ax=None*, *clim=None*, *pcolorOpts=None*, *theta_ind=0*)
Plot a cell centered property

Parameters

- **prop** (*numpy.array*) – cell centered property to plot
- **ax** (*matplotlib.axes*) – axis
- **clim** (*numpy.array*) – colorbar limits
- **pcolorOpts** (*dict*) – dictionary of pcolor options

plot_sigma(*ax=None*, *clim=None*, *pcolorOpts=None*)
plot the electrical conductivity

Parameters

- **ax** (*matplotlib.axes*) – axis
- **clim** (*numpy.array*) – colorbar limits
- **pcolorOpts** (*dict*) – dictionary of pcolor options

sigma
electrical conductivity

Return type numpy.array

wires
wires to hook up maps to sigma, mu

Return type SimPEG.Maps.Wires

class casingSimulations.model.**SingleLayer**(*filename=None*, ***kwargs*)
Bases: *casingSimulations.model.Halfspace*

A model consisting of air, subsurface and a single subsurface layer

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .

- **filename** (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json
- **layer_z** (Array): z-limits of the layer, a list or numpy array of <type 'float'>, <type 'int'> with shape (2), Default: [-1000. -900.]
- **mur_back** (Float): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0
- **sigma_air** (Float): conductivity of the air (S/m), a float, Default: 1e-06
- **sigma_back** (Float): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01
- **sigma_layer** (Float): conductivity of the layer (S/m), a float, Default: 0.01
- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]
- **surface_z** (Float): elevation of the air-earth interface (m), a float, Default: 0
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

ind_layer(*mesh*)
Indices where the layer is

info_model

layer_z
layer_z (Array): z-limits of the layer, a list or numpy array of <type 'float'>, <type 'int'> with shape (2), Default: [-1000. -900.]

sigma(*mesh*)
Construct the conductivity model on a mesh

Parameters **mesh** (*discretize.BaseMesh*) – mesh to put conductivity model on

sigma_layer
sigma_layer (Float): conductivity of the layer (S/m), a float, Default: 0.01

class casingSimulations.model.**SurveyParametersMixin**(***kwargs*)
Bases: properties.base.base.HasProperties

A mixin that has the properties of the survey. It doesn't do anything on its own

Required Properties:

- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

freqs
freqs (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)

info_survey

src_a
src_a (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]

src_b
src_b (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]

timeSteps
timeSteps (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

class casingSimulations.model.**TimeStepArray**(*doc*, ***kwargs*)
Bases: properties.math.Array

class_info = 'an array or list of tuples specifying the mesh tensor'

validate(*instance*, *value*)

class casingSimulations.model.**Wholespace**(*filename=None*, ***kwargs*)
Bases: *casingSimulations.model.SurveyParametersMixin*, *casingSimulations.base.BaseCasing*

Model and survey parameters for an electromagnetic survey in a wholespace

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json
- **mur_back** (Float): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0
- **sigma_back** (Float): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01
- **src_a** (Array): down-hole z-location for the source, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [0. 0. -975.]
- **src_b** (Array): B electrode location, a list or numpy array of <type 'float'>, <type 'int'> with shape (*), Default: [1000. 0. 0.]
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **freqs** (Array): source frequencies, a list or numpy array of <type 'float'> with shape (*)
- **timeSteps** (*TimeStepArray*): times-steps at which to solve, an array or list of tuples specifying the mesh tensor of <type 'float'> with shape (*)

diffusion_distance(*t=None*, *sigma=None*, *mu=None*)
Difusion distance

filename
filename (String): Filename to which the properties are serialized and written to, a unicode string, Default: ModelParameters.json

info

info_model

mu(*mesh*)
Magnetic permeability on a mesh :param discretize.BaseMesh mesh: a discretize mesh :rtype: numpy.ndarray :return: magnetic permeability on the mesh

mur(*mesh*)
Relative magnetic permeability on a mesh :param discretize.BaseMesh mesh: a discretize mesh :rtype: numpy.ndarray :return: relative magnetic permeability on the mesh

mur_back
mur_back (`Float`): relative permittivity of the background, a float in range [0.0, inf], Default: 1.0

sigma(*mesh*)
Electrical conductivity on a mesh :param discretize.BaseMesh mesh: a discretize mesh :rtype: numpy.ndarray :return: electrical conductivity on the mesh

sigma_back
sigma_back (`Float`): conductivity of the background (S/m), a float in range [0.0, inf], Default: 0.01

skin_depth(*sigma=None*, *mu=None*, *f=None*)
Skin depth

$$\delta = \sqrt{(\frac{2}{\omega\mu\sigma})}$$

CHAPTER 3

Mesh Generators

class casingSimulations.mesh.**BaseCylMixin**(***kwargs*)

Bases: properties.base.base.HasProperties

Mixin class that contains properties and methods common to a Cyl Mesh Generator

Required Properties:

- **csz** (Float): cell size in the z-direction, a float, Default: 25.0
- **domain_x** (Float): domain extent in the x-direction, a float, Default: 1000.0
- **hy** (Array): cell spacings in the y direction, a list or numpy array of <type 'float'> with shape (*), Default: [6.28318531]
- **nca** (Integer): number of fine cells above the air-earth interface, an integer, Default: 5
- **ncb** (Integer): number of fine cells below the casing, an integer, Default: 5
- **npadx** (Integer): number of padding cells required to get to infinity!, an integer, Default: 23
- **npadz** (Integer): number of padding cells in z, an integer, Default: 38
- **pfz** (Float): padding factor in the z-direction, a float, Default: 1.5

create_2D_mesh()

create cylindrically symmetric mesh generator

csz

csz (Float): cell size in the z-direction, a float, Default: 25.0

domain_x

domain_x (Float): domain extent in the x-direction, a float, Default: 1000.0

domain_z

z-extent extent of the core mesh

hy

hy (Array): cell spacings in the y direction, a list or numpy array of <type 'float'> with shape (*), Default: [6.28318531]

hz
cell spacings in the z-direction

Return type numpy.array

nca
nca (Integer): number of fine cells above the air-earth interface, an integer, Default: 5

ncb
ncb (Integer): number of fine cells below the casing, an integer, Default: 5

ncy
number of core y-cells

Return type float

ncz
number of core z-cells

Return type float

npadx
npadx (Integer): number of padding cells required to get to infinity!, an integer, Default: 23

npadz
npadz (Integer): number of padding cells in z, an integer, Default: 38

pfz
pfz (Float): padding factor in the z-direction, a float, Default: 1.5

plotModels(*sigma, mu, xlim=[0.0, 1.0], zlim=[-1200.0, 100.0], ax=None*)
Plot conductivity and permeability models

x0
Origin of the mesh

class casingSimulations.mesh.**BaseMeshGenerator**(***kwargs*)
Bases: *casingSimulations.base.BaseCasing*

Base Mesh Generator Class

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: MeshParameters.json
- **modelParameters** (*Wholespace*): casing parameters instance, an instance of Wholespace
- **version** (String): version of the software, a unicode string, Default: 0.0.4

copy()
Make a copy of the object

Return type *BaseMeshGenerator*

filename
filename (String): filename to serialize properties to, a unicode string, Default: MeshParameters.json

mesh
discretize mesh

Return type discretize.BaseMesh

modelParameters
modelParameters (*Wholespace*): casing parameters instance, an instance of Wholespace

class casingSimulations.mesh.**CasingMeshGenerator**(***kwargs*)

Bases: *casingSimulations.mesh.BaseMeshGenerator*, *casingSimulations.mesh.BaseCylMixin*

Mesh that makes sense for casing examples

Required Properties:

- **csx1** (Float): finest cells in the x-direction, a float, Default: 0.0025
- **csx2** (Float): second uniform cell region in x-direction, a float, Default: 25.0
- **csz** (Float): cell size in the z-direction, a float, Default: 25.0
- **directory** (String): Working directory, a unicode string, Default: .
- **domain_x** (Float): domain extent in the x-direction, a float, Default: 1000.0
- **filename** (String): filename to serialize properties to, a unicode string, Default: MeshParameters.json
- **hy** (Array): cell spacings in the y direction, a list or numpy array of <type 'float'> with shape (*), Default: [6.28318531]
- **modelParameters** (*Wholespace*): casing parameters instance, an instance of Wholespace
- **nca** (Integer): number of fine cells above the air-earth interface, an integer, Default: 5
- **ncb** (Integer): number of fine cells below the casing, an integer, Default: 5
- **npadx** (Integer): number of padding cells required to get to infinity!, an integer, Default: 23
- **npadz** (Integer): number of padding cells in z, an integer, Default: 38
- **pfx1** (Float): padding factor to pad from csx1 to csx2, a float, Default: 1.3
- **pfx2** (Float): padding factor to pad to infinity, a float, Default: 1.5
- **pfz** (Float): padding factor in the z-direction, a float, Default: 1.5
- **version** (String): version of the software, a unicode string, Default: 0.0.4

csx1
: **csx1** (Float): finest cells in the x-direction, a float, Default: 0.0025

csx2
: **csx2** (Float): second uniform cell region in x-direction, a float, Default: 25.0

hx
: cell spacings in the x-direction

ncx1
: number of cells with size csx1

npadx1
: number of padding cells to get from csx1 to csx2

pfx1
: **pfx1** (Float): padding factor to pad from csx1 to csx2, a float, Default: 1.3

pfx2
: **pfx2** (Float): padding factor to pad to infinity, a float, Default: 1.5

class casingSimulations.mesh.**CylMeshGenerator**(***kwargs*)

Bases: *casingSimulations.mesh.BaseMeshGenerator*, *casingSimulations.mesh.BaseCylMixin*

Simple 3D cylindrical mesh

Required Properties:

- **csx** (Float): cell size in the x-direction, a float, Default: 25.0
- **csz** (Float): cell size in the z-direction, a float, Default: 25.0
- **directory** (String): Working directory, a unicode string, Default: .
- **domain_x** (Float): domain extent in the x-direction, a float, Default: 1000.0
- **filename** (String): filename to serialize properties to, a unicode string, Default: MeshParameters.json
- **hy** (Array): cell spacings in the y direction, a list or numpy array of <type 'float'> with shape (*), Default: [6.28318531]
- **modelParameters** (*Wholespace*): casing parameters instance, an instance of Wholespace
- **nca** (Integer): number of fine cells above the air-earth interface, an integer, Default: 5
- **ncb** (Integer): number of fine cells below the casing, an integer, Default: 5
- **nch** (Integer): number of cells to add on each side of the mesh horizontally, an integer, Default: 10.0
- **npadx** (Integer): number of padding cells required to get to infinity!, an integer, Default: 23
- **npadz** (Integer): number of padding cells in z, an integer, Default: 38
- **pfx** (Float): padding factor to pad to infinity, a float, Default: 1.5
- **pfz** (Float): padding factor in the z-direction, a float, Default: 1.5
- **version** (String): version of the software, a unicode string, Default: 0.0.4

csx

csx (Float): cell size in the x-direction, a float, Default: 25.0

hx

nch

nch (Integer): number of cells to add on each side of the mesh horizontally, an integer, Default: 10.0

ncx

pfx

pfx (Float): padding factor to pad to infinity, a float, Default: 1.5

class casingSimulations.mesh.**TensorMeshGenerator**(***kwargs*)

Bases: *casingSimulations.mesh.BaseMeshGenerator*

Tensor mesh designed based on the source and formulation

Required Properties:

- **csx** (Float): cell size in the x-direction, a float, Default: 25.0
- **csy** (Float): cell size in the y-direction, a float, Default: 25.0
- **csz** (Float): cell size in the z-direction, a float, Default: 25.0
- **directory** (String): Working directory, a unicode string, Default: .
- **domain_x** (Float): domain extent in the x-direction, a float, Default: 1000.0
- **domain_y** (Float): domain extent in the y-direction, a float, Default: 1000.0
- **filename** (String): filename to serialize properties to, a unicode string, Default: MeshParameters.json
- **modelParameters** (*Wholespace*): casing parameters instance, an instance of Wholespace
- **nca** (Integer): number of extra cells above the air-earth interface, an integer, Default: 5

- **ncb** (Integer): number of cells below the casing, an integer, Default: 5
- **nch** (Integer): number of cells to add on each side of the mesh horizontally, an integer, Default: 10
- **npadx** (Integer): number of x-padding cells, an integer, Default: 10
- **npady** (Integer): number of y-padding cells, an integer, Default: 10
- **npadz** (Integer): number of z-padding cells, an integer, Default: 10
- **pfx** (Float): padding factor to pad to infinity, a float, Default: 1.5
- **pfy** (Float): padding factor to pad to infinity, a float, Default: 1.5
- **pfz** (Float): padding factor to pad to infinity, a float, Default: 1.5
- **version** (String): version of the software, a unicode string, Default: 0.0.4

csx
csx (Float): cell size in the x-direction, a float, Default: 25.0

csy
csy (Float): cell size in the y-direction, a float, Default: 25.0

csz
csz (Float): cell size in the z-direction, a float, Default: 25.0

domain_x
domain_x (Float): domain extent in the x-direction, a float, Default: 1000.0

domain_y
domain_y (Float): domain extent in the y-direction, a float, Default: 1000.0

domain_z
vertical extent of the mesh

Return type float

hx
vector of cell spacings in the x-direction

Return type numpy.array

hy
vector of cell spacings in the y-direction

Return type numpy.array

hz
vector of cell spacings in the z-direction

Return type numpy.array

nca
nca (Integer): number of extra cells above the air-earth interface, an integer, Default: 5

ncb
ncb (Integer): number of cells below the casing, an integer, Default: 5

nch
nch (Integer): number of cells to add on each side of the mesh horizontally, an integer, Default: 10

ncx
number of x-cells

Return type int

ncy
number of y-cells

Return type int

ncz
number of z-cells

Return type int

npadx
npadx (Integer): number of x-padding cells, an integer, Default: 10

npady
npady (Integer): number of y-padding cells, an integer, Default: 10

npadz
npadz (Integer): number of z-padding cells, an integer, Default: 10

pfx
pfx (Float): padding factor to pad to infinity, a float, Default: 1.5

pfy
pfy (Float): padding factor to pad to infinity, a float, Default: 1.5

pfz
pfz (Float): padding factor to pad to infinity, a float, Default: 1.5

x0
Origin of the mesh

Return type numpy.array

CHAPTER 4

Sources

class casingSimulations.sources.**BaseCasingSrc**(***kwargs*)

Bases: *casingSimulations.base.BaseCasing*

The base class for sources. Inherit this to attach properties.

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: Source.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): casing parameters, an instance of Wholespace
- **version** (String): version of the software, a unicode string, Default: 0.0.4

casing_a
inner radius of the casing

filename
filename (String): filename to serialize properties to, a unicode string, Default: Source.json

freqs
frequencies to consider

mesh
discretize mesh

meshGenerator
meshGenerator (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator

modelParameters
modelParameters (*Wholespace*): casing parameters, an instance of Wholespace

srcList
Source List

src_a
location of the a-electrode

src_b
location of the b-electrode

class casingSimulations.sources.**DownHoleCasingSrc**(***kwargs*)
Bases: *casingSimulations.sources.DownHoleTerminatingSrc*

Source that is coupled to the casing down-hole and has a return electrode at the surface.

param CasingSimulations.Model.CasingProperties modelParameters a casing properties instance

param discretize.CylMesh mesh a cylindrical mesh

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: Source.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): casing parameters, an instance of Wholespace
- **version** (String): version of the software, a unicode string, Default: 0.0.4

downhole_electrode
Down-hole horizontal part of the wire, coupled to the casing

plot(*ax=None*)
Plot the source.

s_e
Source current density on faces

class casingSimulations.sources.**DownHoleTerminatingSrc**(***kwargs*)
Bases: *casingSimulations.sources.BaseCasingSrc*

A source that terminates down-hole. It is not coupled to the casing

param CasingSimulations.Model.CasingProperties modelParameters a casing properties instance

param discretize.BaseMesh mesh a discretize mesh

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: Source.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): casing parameters, an instance of Wholespace
- **version** (String): version of the software, a unicode string, Default: 0.0.4

plot(*ax=None*)
Plot the source.

s_e
Source List

src_a_closest
closest face to where we want the return current electrode

src_b_closest
closest face to where we want the return current electrode

surface_electrode
Return electrode on the surface

surface_wire
Horizontal part of the wire that runs along the surface (one cell above) from the center of the well to the return electrode

surface_wire_direction

wire_in_borehole
Indices of the verically directed wire inside of the borehole. It goes through the center of the well

class casingSimulations.sources.**HorizontalElectricDipole**(***kwargs*)
Bases: *casingSimulations.sources.BaseCasingSrc*

A horizontal electric dipole

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: Source.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): casing parameters, an instance of Wholespace
- **version** (String): version of the software, a unicode string, Default: 0.0.4

plot(*ax=None*)
Plot the source.

s_e
electric source term used to build the right hand side of the maxwell system

src_a_closest
closest face to where we want the return current electrode

src_b_closest
closest face to where we want the return current electrode

surface_wire
Horizontal part of the wire that runs along the surface (one cell above) from the center of the well to the return electrode

surface_wire_direction
direction of the source wire

class casingSimulations.sources.**TopCasingSrc**(***kwargs*)
Bases: *casingSimulations.sources.DownHoleTerminatingSrc*

Source that has one electrode coupled to the top of the casing, one return electrode and a wire in between. This source is set up to live on faces.

param discretize.CylMesh mesh the cylindrical simulation mesh

param CasingSimulations modelParameters Casing parameters object

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: Source.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): casing parameters, an instance of Wholespace

- **version** (String): version of the software, a unicode string, Default: 0.0.4

plot(*ax=None*)
plot the source on the mesh.

s_e
source list

surface_wire
indices of the wire that runs along the surface

tophole_electrode
Indices of the electrode that is grounded on the top of the casing

class casingSimulations.sources.**VerticalElectricDipole**(***kwargs*)
Bases: *casingSimulations.sources.BaseCasingSrc*

A vertical electric dipole. It is not coupled to the casing

param CasingSimulations.Model.CasingProperties modelParameters a casing properties instance

param discretize.BaseMesh mesh a discretize mesh

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **filename** (String): filename to serialize properties to, a unicode string, Default: Source.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): casing parameters, an instance of Wholespace
- **version** (String): version of the software, a unicode string, Default: 0.0.4

plot(*ax=None*)
Plot the source.

s_e
Source List

src_a_closest
closest face to where we want the return current electrode

src_b_closest
closest face to where we want the return current electrode

wire_in_borehole
Indices of the verically directed wire inside of the borehole. It goes through the center of the well

CHAPTER 5

Run

class casingSimulations.run.**BaseSimulation**(***kwargs*)

Bases: *casingSimulations.base.BaseCasing*

Base class wrapper to run an EM Forward Simulation

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **fields_filename** (String): filename for the fields, a unicode string, Default: fields.npy
- **filename** (String): filename for the simulation parameters, a unicode string, Default: simulationParameters.json
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): Model Parameters instance, an instance of Wholespace
- **num_threads** (Integer): number of threads, an integer, Default: 1
- **verbose** (Bool): run the simulation in Verbose mode?, a boolean, Default: False
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **src** (*BaseCasingSrc*): Source Parameters instance, an instance of BaseCasingSrc

fields()
fields from the forward simulation

fields_filename
fields_filename (String): filename for the fields, a unicode string, Default: fields.npy

filename
filename (String): filename for the simulation parameters, a unicode string, Default: simulationParameters.json

meshGenerator
meshGenerator (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator

modelParameters
modelParameters (*Wholespace*): Model Parameters instance, an instance of Wholespace

num_threads
num_threads (Integer): number of threads, an integer, Default: 1

physprops

prob

run()
Run the forward simulation

src
src (*BaseCasingSrc*): Source Parameters instance, an instance of BaseCasingSrc

survey

verbose
verbose (Bool): run the simulation in Verbose mode?, a boolean, Default: False

write_py(*physics=None*, *includeDC=True*, *include2D=True*)
Write a python script for running the simulation :param str physics: 'TDEM', 'FDEM' :param bool includeDC: include a DC simulation with the EM one (default is True) :param bool include2D: include a 2D simulation? (default is True)

class casingSimulations.run.**SimulationDC**(***kwargs*)
Bases: *casingSimulations.run.BaseSimulation*

A wrapper to run a DC Forward Simulation :param CasingSimulations.model.WholeSpace modelParameters: casing parameters object :param CasingSimulations.mesh.BaseMeshGenerator mesh: a CasingSimulation mesh generator object

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **fields_filename** (String): filename for the fields, a unicode string, Default: fieldsDC.npy
- **filename** (String): filename for the simulation parameters, a unicode string, Default: simulationParameters.json
- **formulation** (String): field that we are solving for, a unicode string, Default: phi
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): Model Parameters instance, an instance of Wholespace
- **num_threads** (Integer): number of threads, an integer, Default: 1
- **src_a** (Vector3): a electrode location, a 3D Vector of <type 'float'> with shape (3)
- **src_b** (Vector3): return electrode location, a 3D Vector of <type 'float'> with shape (3)
- **verbose** (Bool): run the simulation in Verbose mode?, a boolean, Default: False
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **src** (*BaseCasingSrc*): Source Parameters instance, an instance of BaseCasingSrc

fields_filename
fields_filename (String): filename for the fields, a unicode string, Default: fieldsDC.npy

formulation
formulation (String): field that we are solving for, a unicode string, Default: phi

physics = 'DC'

src_a
src_a (Vector3): a electrode location, a 3D Vector of <type 'float'> with shape (3)

src_b
src_b (Vector3): return electrode location, a 3D Vector of <type 'float'> with shape (3)

class casingSimulations.run.**SimulationFDEM**(***kwargs*)
Bases: *casingSimulations.run.BaseSimulation*

A wrapper to run an FDEM Forward Simulation :param CasingSimulations.model.WholeSpace modelParameters: casing parameters object :param CasingSimulations.mesh.BaseMeshGenerator mesh: a CasingSimulation mesh generator object

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **fields_filename** (String): filename for the fields, a unicode string, Default: fields.npy
- **filename** (String): filename for the simulation parameters, a unicode string, Default: simulationParameters.json
- **formulation** (StringChoice): Formulation of the problem to solve [e, b, h, j], any of "e", "b", "h", "j", Default: h
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator
- **modelParameters** (*Wholespace*): Model Parameters instance, an instance of Wholespace
- **num_threads** (Integer): number of threads, an integer, Default: 1
- **verbose** (Bool): run the simulation in Verbose mode?, a boolean, Default: False
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **src** (*BaseCasingSrc*): Source Parameters instance, an instance of BaseCasingSrc

formulation
formulation (StringChoice): Formulation of the problem to solve [e, b, h, j], any of "e", "b", "h", "j", Default: h

physics = 'FDEM'

class casingSimulations.run.**SimulationTDEM**(***kwargs*)
Bases: *casingSimulations.run.BaseSimulation*

A wrapper to run a TDEM Forward Simulation :param CasingSimulations.model.WholeSpace modelParameters: casing parameters object :param CasingSimulations.mesh.BaseMeshGenerator mesh: a CasingSimulation mesh generator object

Required Properties:

- **directory** (String): Working directory, a unicode string, Default: .
- **fields_filename** (String): filename for the fields, a unicode string, Default: fields.npy
- **filename** (String): filename for the simulation parameters, a unicode string, Default: simulationParameters.json
- **formulation** (StringChoice): Formulation of the problem to solve [e, b, h, j], any of "e", "b", "h", "j", Default: j
- **meshGenerator** (*BaseMeshGenerator*): mesh generator instance, an instance of BaseMeshGenerator

- **modelParameters** (*Wholespace*): Model Parameters instance, an instance of Wholespace
- **num_threads** (Integer): number of threads, an integer, Default: 1
- **verbose** (Bool): run the simulation in Verbose mode?, a boolean, Default: False
- **version** (String): version of the software, a unicode string, Default: 0.0.4

Optional Properties:

- **src** (*BaseCasingSrc*): Source Parameters instance, an instance of BaseCasingSrc

formulation
: **formulation** (StringChoice): Formulation of the problem to solve [e, b, h, j], any of "e", "b", "h", "j", Default: j

physics = 'TDEM'

CHAPTER 6

Physics

casingSimulations.physics.**CasingCurrents**(*j*, *mesh*, *survey*, *casing_a*, *casing_b*, *casing_z*)

casingSimulations.physics.**plotCurrentDensity**(*mesh*, *fields_j*, *saveFig=False*, *figsize=(4, 5)*, *fontsize=12*, *csx=5.0*, *csz=5.0*, *xmax=1000.0*, *zmin=0.0*, *zmax=-1200.0*, *real_or_imag='real'*, *mirror=False*, *ax=None*, *fig=None*, *clim=None*)

casingSimulations.physics.**plot_currents_over_freq**(*IxCasing, IzCasing, modelParameters, mesh, mur=1, subtract=None, real_or_imag='real', ax=None, xlim=[-1100.0, 0.0], logScale=True, srcinds=[0], ylim_0=None, ylim_1=None*)

casingSimulations.physics.**plot_currents_over_mu**(*IxCasing, IzCasing, modelParameters, mesh, freqind=0, real_or_imag='real', subtract=None, ax=None, fig=None, logScale=True, srcinds=[0], ylim_0=None, ylim_1=None*)

casingSimulations.physics.**plot_j_over_freq_z**(*modelParameters, fields, mesh, survey, mur=1.0, r=1.0, xlim=[-1100.0, 0.0], real_or_imag='real', subtract=None, ax=None, logScale=True, srcinds=[0], ylim_0=None, ylim_1=None, fig=None*)

casingSimulations.physics.**plot_j_over_mu_x**(*modelParameters, fields, mesh, survey, srcind=0, mur=1, z=-950.0, real_or_imag='real', subtract=None, xlim=[0.0, 2000.0], logScale=True, srcinds=[0], ylim_0=None, ylim_1=None, ax=None, fig=None*)

casingSimulations.physics.**plot_j_over_mu_z**(*modelParameters, fields, mesh, survey, freqind=0, r=1.0, xlim=[-1100.0, 0.0], real_or_imag='real', subtract=None, ax=None, logScale=True, srcinds=[0], ylim_0=None, ylim_1=None, fig=None*)

CHAPTER 7

Utils

casingSimulations.utils.**ccv3DthetaSlice**(*mesh3D*, *v3D*, *theta_ind=0*)
Grab a theta slice through a 3D field defined at cell centers

Parameters

- **mesh3D** (*discretize.CylMesh*) – 3D cyl mesh
- **v3D** (*numpy.ndarray*) – vector of fields on mesh
- **theta_ind** (*int*) – index of the theta slice that you want

casingSimulations.utils.**edge3DthetaSlice**(*mesh3D*, *h3D*, *theta_ind=0*)
Grab a theta slice through a 3D field defined on edges (y component), consistent with what would be found from a 2D simulation

Parameters

- **mesh3D** (*discretize.CylMesh*) – 3D cyl mesh
- **h3D** (*numpy.ndarray*) – vector of fields on mesh
- **theta_ind** (*int*) – index of the theta slice that you want

casingSimulations.utils.**face3DthetaSlice**(*mesh3D*, *j3D*, *theta_ind=0*)
Grab a theta slice through a 3D field defined on faces (x, z components), consistent with what would be found from a 2D simulation

Parameters

- **mesh3D** (*discretize.CylMesh*) – 3D cyl mesh
- **j3D** (*numpy.ndarray*) – vector of fluxes on mesh
- **theta_ind** (*int*) – index of the theta slice that you want

casingSimulations.utils.**loadSimulationResults**(*directory='.'*, *simulationParameters='simulationParameters.json'*, *fields='fields.npy'*, *meshGenerator=None*)

casingSimulations.utils.**load_properties**(*filename*)

Open a json file and load the properties into the target class

As long as there are no namespace conflicts, the target __class__ will be stored on the properties.HasProperties registry and may be fetched from there.

Parameters **filename** (*str*) – name of file to read in

casingSimulations.utils.**writeSimulationPy**(*modelParameters='ModelParameters.json'*, *meshGenerator='MeshParameters.json'*, *src='Source.json'*, *physics='FDEM'*, *fields_filename='fields.npy'*, *directory='.'*, *simulation_filename='simulation.py'*, *includeDC=True*, *include2D=True*)

CHAPTER 8

View

casingSimulations.view.**plotEdge2D**(*mesh2D*, *h*, *real_or_imag='real'*, *ax=None*, *range_x=None*, *range_y=None*, *sample_grid=None*, *logScale=True*, *clim=None*, *mirror=False*, *pcolorOpts=None*)

Create a pcolor plot (a slice in the theta direction) of an edge vector

Parameters

- **mesh2D** (*discretize.CylMesh*) – cylindrically symmetric mesh
- **h** (*np.ndarray*) – edge vector (y components)
- **real_or_imag** (*str*) – real or imaginary component
- **ax** (*matplotlib.axes*) – axes
- **range_x** (*numpy.ndarray*) – x-extent over which we want to plot
- **range_y** (*numpy.ndarray*) – y-extent over which we want to plot
- **sample_grid** (*numpy.ndarray*) – x, y spacings at which to re-sample the plotting grid
- **logScale** (*bool*) – use a log scale for the colorbar?

casingSimulations.view.**plotFace2D**(*mesh2D*, *j*, *real_or_imag='real'*, *ax=None*, *range_x=None*, *range_y=None*, *sample_grid=None*, *logScale=True*, *clim=None*, *mirror=False*, *pcolorOpts=None*, *cbar=True*)

Create a streamplot (a slice in the theta direction) of a face vector

Parameters

- **mesh2D** (*discretize.CylMesh*) – cylindrically symmetric mesh
- **j** (*np.ndarray*) – face vector (x, z components)
- **real_or_imag** (*str*) – real or imaginary component
- **ax** (*matplotlib.axes*) – axes
- **range_x** (*numpy.ndarray*) – x-extent over which we want to plot

- **range_y** (*numpy.ndarray*) – y-extent over which we want to plot
- **sample_grid** (*numpy.ndarray*) – x, y spacings at which to re-sample the plotting grid
- **logScale** (*bool*) – use a log scale for the colorbar?

casingSimulations.view.**plotLinesFx**(*mesh, field, pltType='semilogy', ax=None, theta_ind=0, xlim=[0.0, 2500.0], zloc=0.0, real_or_imag='real', color_ind=0, label=None*)

Python Module Index

C

Index

D

E

F

H

I

L

M

N

P

R

S

T

V

W

X