Open FDEM Post-Processing

Release 1.0

open-fdem 2021

Sep 22, 2021

CONTENTS:

CHAPTER

ONE

OPENFDEM

1.1 openfdem package

1.1.1 Submodules

1.1.2 openfdem.openfdem module

class openfdem.openfdem.**Model**(*folder=None*, *runfile=None*, *fdem_engine=None*)

Bases: `object`

Model class collects datafiles into one interface.

Each data array returns as a list ordered by timestep Collection of timesteps? handles temporal manipulations

Example

```
>>> import openfdem as fdem
>>> model = fdem.Model("../example_outputs/Irazu_UCS")
```

Eavg_mod(*ucs_data*, *upperrange*, *lowerrange*, *loc_strain='Platen Strain'*)

Average Elastic modulus between two ranges

Parameters

- **ucs_data** (*pandas.DataFrame*) – DataFrame containing the stress-strain data
- **upperrange** (*float*) – Upper range to calculate the average
- **lowerrange** (*float*) – Lower range to calculate the average
- **loc_strain** (*str*) – Column to obtain strain from. Defaults to Platen Strain

Returns Average Elastic modulus

Return type float

Raises **ZeroDivisionError** – The range over which to calculate the Eavg is too small. Consider a larger range.

Example

```
>>> data = pv.read("../example_outputs/Irazu_UCS")
>>> df_1 = data.complete_stress_strain(True)
>>> data.Eavg_mod(df_1, 0.5, 0.6)
51485.33001517835
>>> data.Eavg_mod(df_1, 0.5, 0.6, 'Gauge Displacement Y')
50976.62587224803
```

Esec_mod(*ucs_data*, *upperrange*, *loc_strain='Platen Strain'*)
Secant Modulus between 0 and upperrange. The upperrange can be a % or a fraction.

Parameters

- **ucs_data** (*pandas.DataFrame*) – DataFrame containing the stress-strain data
- **upperrange** (*float*) – Range over which to calculate the Secant Modulus
- **loc_strain** (*str*) – Column to obtain strain from. Defaults to Platen Strain

Returns Secant Elastic modulus between 0 and upperrange

Return type float

Example

```
>>> data = pv.read("../example_outputs/Irazu_UCS")
>>> df_1 = data.complete_stress_strain(True)
>>> data.Esec_mod(df_1, 0.5)
51751.010161057035
>>> data.Esec_mod(df_1, 0.5, 'Gauge Displacement Y')
51279.95421163901
```

Etan50_mod(*ucs_data*, *loc_strain='Platen Strain'*)
Tangent Elastic modulus at 50%. Calculates +/- 1 datapoint from the 50% Stress

Parameters

- **ucs_data** (*pandas.DataFrame*) – DataFrame containing the stress-strain data
- **loc_strain** (*str*) – Column to obtain strain from. Defaults to Platen Strain

Returns Tangent Elastic modulus at 50%

Return type float

Example

```
>>> data = pv.read("../example_outputs/Irazu_UCS")
>>> df_1 = data.complete_stress_strain(True)
>>> data.Etan_mod(df_1)
51539.9101160927
>>> data.Etan_mod(df_1, 'Gauge Displacement Y')
51043.327845235595
```

complete_BD_stress_strain(*st_status=False*, *gauge_width=0*, *gauge_length=0*, *progress_bar=False*)
Calculate the full stress-strain curve

Parameters

- **st_status** (*bool*) – Enable/Disable SG
- **gauge_width** (*float*) – width of the virtual strain gauge
- **gauge_length** (*float*) – length of the virtual strain gauge

Returns full stress-strain information

Return type pandas.DataFrame

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("../example_outputs/OpenFDEM_BD")
# full stress-strain without SG
>>> df_wo_SG = data.complete_BD_stress_strain(False)
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
# full stress-strain with SG and default dimensions
>>> df_Def_SG = data.complete_BD_stress_strain(True)
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
    Name: Gauge Displacement X, dtype=float64, nullable: False
    Name: Gauge Displacement Y, dtype=float64, nullable: False
# full stress-strain with SG and user-defined dimensions
>>> df_userdf_SG = data.complete_BD_stress_strain(True, 10, 10)
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
    Name: Gauge Displacement X, dtype=float64, nullable: False
    Name: Gauge Displacement Y, dtype=float64, nullable: False
```

complete_stress_strain(*platen_id=None*, *st_status=False*, *gauge_width=0*, *gauge_length=0*, *progress_bar=False*)

Calculate the full stress-strain curve

Parameters

- **platen_id** (*None or int*) – Manual override of Platen ID
- **st_status** (*bool*) – Enable/Disable SG
- **gauge_width** (*float*) – width of the virtual strain gauge
- **gauge_length** (*float*) – length of the virtual strain gauge
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns full stress-strain information

Return type pandas.DataFrame

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
# Minimal Arguments
>>> df_wo_SG = data.complete_stress_strain()
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
# full stress-strain without SG
>>> df_wo_SG = data.complete_stress_strain(None, False)
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
# full stress-strain with SG and default dimensions
>>> df_Def_SG = data.complete_stress_strain(None, True)
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
```

(continues on next page)

(continued from previous page)

```
    Name: Gauge Displacement X, dtype=float64, nullable: False
    Name: Gauge Displacement Y, dtype=float64, nullable: False
# full stress-strain with SG and user-defined dimensions
>>> df_userdf_SG = data.complete_stress_strain(None, True, 10, 10)
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
    Name: Gauge Displacement X, dtype=float64, nullable: False
    Name: Gauge Displacement Y, dtype=float64, nullable: False
```

extract_based_coord(*thres_model*, *coord_xyz*, *location*, *include_cells=False*, *adjacent_cells=False*)

Extract the vtkdata set based on the defined coord location in the x=0 y=1 z=2 location.

Parameters

- **thres_model** (*pyvista.core.pointset.UnstructuredGrid*) – threshold dataset of the material id of the rock
- **coord_xyz** (*int*) – x=0 y=1 z=2
- **location** (*float*) – Xmin/Xmax/Ymin/Ymax/Zmin/Zmax
- **include_cells** (*bool*) – If True, extract the cells that contain at least one of the extracted points. If False, extract the cells that contain exclusively points from the extracted points list.
- **adjacent_cells** (*bool*) – Specifies if the cells shall be returned or not

Returns Pointset of the data being filtered

Return type pyvista.core.pointset.UnstructuredGrid

extract_cell_info(*cell_id*, *arrays_needed*, *progress_bar=False*)

Returns the information of the cell based on the array requested. If the array is a point data, the array is suffixed with _Nx where x is the node on that cell. Also shows a quick example on how to plot the information extracted.

Parameters

- **cell_id** (*int*) – Cell ID to extract
- **arrays_needed** (*list[str]*) – list of array names to extract
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns unpacked DataFrame

Return type pandas.DataFrame

Example

```
>>> import openfdem as fdem
>>> import matplotlib.pyplot as plt
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
# Extract data platen_force', 'mineral_type' from Cell ID 1683
>>> extraction_of_cellinfo = data.extract_cell_info(1683, ['platen_
↪force', 'mineral_type'])
Columns:
    Name: platen_force_N1, dtype=object, nullable: False
    Name: platen_force_N2, dtype=object, nullable: False
    Name: platen_force_N3, dtype=object, nullable: False
```

(continues on next page)

(continued from previous page)

```
    Name: mineral_type, dtype=object, nullable: False
# For noded information => PLOTTING METHOD ONE
>>> x, y = [], []
>>> for i, row in extraction_of_cellinfo.iterrows():
>>>     x.append(i)
>>>     y.append(row['platen_force_N2'][0])
>>> plt.plot(x, y, c='red', label='platen_force_N2_x')
>>> plt.legend()
>>> plt.show()
# For noded information => PLOTTING METHOD TWO
>>> lx = extraction_of_cellinfo['platen_force_N2'].to_list()
>>> lx1 = list(zip(*lx))
>>> plt.plot(lx1[0], label='platen_force_N2_x')
>>> plt.plot(lx1[1], label='platen_force_N2_y')
>>> plt.plot(lx1[2], label='platen_force_N2_z')
>>> plt.legend()
>>> plt.show()
# For non-nonded information
>>> plt.plot(lx1[0], label='mineral_type')
>>> plt.legend()
>>> plt.show()
```

find_cell (*model_point*)
Identify the nearest cell in the model to the defined point

Parameters **model_point** (*list[float, float, float]*) – x,y,z of a point in the model which

Returns the cell nearest to the point

Return type int

Raises **IndexError** – Point outside model domain.

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> data.find_cell([0, 0, 0])
2167
>>> data.find_cell([100, 100, 0])
IndexError: Point outside model domain.
X=56.0, Y=116.0, Z=0.0
```

mat_bound_check (*mat_id*)
Checks the material ID is a valid choice.

Parameters **mat_id** (*int*) – Material ID

Returns ID of the material

Return type int

Raises **IndexError** – Material ID for platen out of range.

Example

```
>>> import openfdem as fdem
>>> model = fdem.Model("../example_outputs/Irazu_UCS")
>>> model.mat_bound_check(0)
```

(continues on next page)

(continued from previous page)

```
0
>>> model.mat_bound_check(5)
IndexError: Material ID for platen out of range.
Material Range 0-1
```

model_dimensions(*mat_id=None*)

Function to get the "INITIAL" model bounds and returns the width, height, thickness

Parameters **mat_id** (*int*) – Optional, if a threshold is specific to a material type

Returns model width, model height, model thickness

Type tuple[float, float, float]

Example

```
>>> import openfdem as fdem
>>> model = fdem.Model("../example_outputs/Irazu_UCS")
>>> # Returns the overall model dimensions
>>> model.model_dimensions()
(56.0, 116.0, 0.0)
>>> # Returns the model dimensions based on material id 1
>>> model.model_dimensions(1)
(56.0, 116.0, 0.0)
>>> # Error when material is not found
>>> model.model_dimensions(3)
IndexError: Material ID for platen out of range.
Material Range 0-1
```

model_domain()

Identifies the model domain by confirming the simulation cell vertex. 2D (3 Points - Triangle) 3D (4 Points - Tetrahedral)

Returns number of nodes to skip in analysis

Return type int

Raises **Exception** – The simulation is not currently supported.

Example

```
>>> import openfdem as fdem
>>> model = fdem.Model("../example_outputs/Irazu_UCS")
>>> model.model_domain()
2D Simulation
4
```

openfdem_att_check(*att*)

Checks that the attribute is a valid choice.

Param Attribute

Type str

Returns Attribute

Return type str

Raises **KeyError** – Attribute does not exist.

Example

```
>>> import openfdem as fdem
>>> model = fdem.Model("../example_outputs/Irazu_UCS")
>>> model.openfdem_att_check('mineral_type')
'mineral_type'
>>> model.openfdem_att_check('material_property')
KeyError: Attribute does not exist.
Available options are mineral_type, boundary, platen_force, platen_
↪displacement, gauge_displacement'
```

platen_info (*pv_cells*, *platen_boundary_id*, *var_property*)

This function thresholds cells based on boundary condition and sums them based on the defined parameter var_property

Parameters

- **pv_cells** (*pyvista.core.pointset.UnstructuredGrid*) –
- **platen_boundary_id** (*float*) – boundary id that the threshold should be based on
- **var_property** (*str*) – name of the property (array to b returned)

Returns array of the property based on the threshold

Return type ndarray

plot_stress_strain (*strain*, *stress*, *ax=None*, ***plt_kwargs*)

Simple plot of the stress-strain curve

Parameters

- **strain** (*pandas.DataFrame*) – X-axis data [Strain]
- **stress** (*pandas.DataFrame*) – Y-axis data [Stress]
- **ax** (*matplotlib*) – Matplotlib Axis
- **plt_kwargs** – *~matplotlib.Modules* submodules

Returns Matplotlib AxesSubplots

Return type Matplotlib Axis

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("../example_outputs/OpenFDEM_BD")
# Minimal Arguments
>>> df_wo_SG = data.complete_stress_strain()
Columns:
    Name: Platen Stress, dtype=float64, nullable: False
    Name: Platen Strain, dtype=float64, nullable: False
>>> data.plot_stress_strain(df_wo_SG['Platen Strain'], df_wo_SG[
↪'Platen Stress'], label='stress-strain', color='green')
<AxesSubplot:xlabel='Strain (-)', ylabel='Axial Stress (MPa)'>
```

rock_sample_dimensions (*platen_id=None*)

Lookup cell element ID on the top center and then trace points Using this information, we obtain the platen prop ID. Alternatively the user can define the material ID to exclude

Parameters **platen_id** (*None or int*) – Manual override of Platen ID

Returns sample width, sample height, sample thickness

Return type tuple[float, float, float]

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> # Let the script try to identify the platen material ID
>>> data.rock_sample_dimensions()
Script Identifying Platen
    Platen Material ID found as [1]
(52.0, 108.0, 0.0)
>>> # Explicitly defined the platen material ID
>>> data.rock_sample_dimensions(0)
User Defined Platen ID
    Platen Material ID found as [0]
(56.0, 116.0, 0.0)
>>> # Explicitly defined the platen material ID is out of range
>>> data.rock_sample_dimensions(3)
IndexError: Material ID for platen out of range.
Material Range 0-1
```

rotary_shear_calculation (*platen_id*, *array*, *progress_bar=False*)

Parameters

- **platen_id** (*int*) – Material id of the platen
- **array** (*str*) – the name of the array to be extracted
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns DataFrame containing the absolute value of the array for each identified corner. Absolute sum of the extracted array split in Top/Bottom ane Left/Rigth sub-set into Top/Bottom.

Return type pandas.DataFrame

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("/external/2D_shear_4mm_profile_normal_load_test")
>>> df = data.rotary_shear_calculation(1, 'platen_force', progress_bar=True)
User Defined Platen ID
    Platen Material ID found as 1
No. of points
    Left        158
    Left_Top    78
    Left_Bottom 80
    Right       158
    Right_Top   76
    Right_Bottom        82
    Top 35
    Bottom      38
>>> import matplotlib.pyplot as plt
>>> plt.plot(df['Left_Top'], label='Left Top')
[<matplotlib.lines.Line2D object at 0x7fe71f187320>]
>>> plt.plot(df['Left_Bottom'], label='Left Bottom')
[<matplotlib.lines.Line2D object at 0x7f65cf8975f8>]
>>> plt.plot(df['Left'], label='Left')
[<matplotlib.lines.Line2D object at 0x7fe71f187390>]
>>> plt.legend()
```

(continues on next page)

(continued from previous page)

```
<matplotlib.legend.Legend object at 0x7fe71f187668>
>>> plt.show()
```

simulation_type()

Identifies the type of simulation running. BD or UCS. Checks the top left corner of the model. If it contains material it is assumed as a rectangle.

Returns Type of simulation. BD/UCS

Return type str

Example

```
>>> import openfdem as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> data.rock_sample_dimensions()
Script Identifying Platen
    Platen Material ID found as [1]
(52.0, 108.0, 0.0)
>>> data.simulation_type()
'UCS Simulation'
```

unpack_DataFrame(*packed_cell_info*)

Unpacking of the original array produced by pyvista If the array is a point data, the array is suffixed with _Nx where x is the node on that cell.

Parameters **packed_cell_info** (*pandas.DataFrame*) –

Returns Unpacked DataFrame

Return type pandas.DataFrame

class openfdem.openfdem.**Timestep**(*file*, *runfile=None*)

Bases: `object`

A class handling the data of each timestep.

Each data array returns for only the timestep handles spatial manipulations

1.1.3 openfdem.aggregate_storage module

class openfdem.aggregate_storage.**aggregate_storage**(*file_directory*, *h5filename=None*, *overwrite=False*, *compression=None*, *verbose=True*)

Bases: `object`

Aggregator class to store VTK files in a single h5 file for faster access to data.

file_group_key(*vtkfilename*)

Produces a standard group/key based on VTK file name

Parameters **vtkfilename** (*str path*) – VTK file name to be stored/read

Returns Key described using timestep and filename

Return type str

read_file(*filename*, *verbose=False*)

Extract VTK file from HDF5 file given original filename

The VTK file is reconstructed from the data arrays stored in the HDF5 file. It will be similar but different from the original.

Parameters

- **filename** (*str path*) – File name to be extracted (unaltered since HDF5 file creation)
- **verbose** (*bool, optional*) – Print progress statements, defaults to False

Returns VTK Unstructured Grid as if read from a *.vtp or *.vtu file

Return type VTK Unstructured Grid

store_file(*vtkfilename*)

Stores VTK file into HDF5 file

Parameters **vtkfilename** (*str path*) – VTK file name

1.1.4 openfdem.complete_BD_thread_pool_generators module

openfdem.complete_BD_thread_pool_generators.**history_strain_func**(*f_name*, *model*, *cv*, *ch*)

Calculate the axial stress from platens, axial strain from platens and SG as well as lateral strain from SG

Parameters

- **f_name** (*str*) – name of vtu file being processed
- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **cv** (*list*) – list of cells at the corner of the vertical strain gauge
- **ch** (*list*) – list of cells at the corner of the horizontal strain gauge

Returns Stress, Platen Strain, SG Strain, SG Lateral Strain

Return type Generator[Tuple[list, list, list, list], Any, None]

openfdem.complete_BD_thread_pool_generators.**main**(*model*, *st_status*, *gauge_width*, *gauge_length*, *progress_bar=False*)

Main concurrent Thread Pool to calculate the full stress-strain

Parameters

- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **st_status** (*bool*) – Enable/Disable SG Calculations
- **gauge_width** (*float*) – SG width
- **gauge_length** (*float*) – SG length
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns full stress-strain information

Return type pd.DataFrame

openfdem.complete_BD_thread_pool_generators.**set_strain_gauge**(*model*, *gauge_length=None*, *gauge_width=None*)

Calculate local strain based on the dimensions of a virtual strain gauge placed at the center of teh model with x/y dimensions. By default set to 0.25 of the length/width.

Parameters

- **model** (openfdem.openfdem.Model) – FDEM Model Class

- **gauge_length** (*float*) – length of the virtual strain gauge
- **gauge_width** (*float*) – width of the virtual strain gauge

Returns Cells that cover the horizontal and vertical gauges as well as the gauge width and length

Return type [list, list, float, float]

1.1.5 openfdem.complete_UCS_thread_pool_generators module

`openfdem.complete_UCS_thread_pool_generators.`**history_strain_func**(*f_name*, *model*, *cv*, *ch*)

Calculate the axial stress from platens, axial strain from platens and SG as well as lateral strain from SG

Parameters

- **f_name** (*str*) – name of vtu file being processed
- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **cv** (*list[int]*) – list of cells at the corner of the vertical strain gauge
- **ch** (*list[int]*) – list of cells at the corner of the horizontal strain gauge

Returns Stress, Platen Strain, SG Strain, SG Lateral Strain

Return type Generator[Tuple[list, list, list, list], Any, None]

`openfdem.complete_UCS_thread_pool_generators.`**main**(*model*, *platen_id*, *st_status*, *gauge_width*, *gauge_length*, *progress_bar=False*)

Main concurrent Thread Pool to calculate the full stress-strain

Parameters

- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **platen_id** (*None or int*) – Manual override of Platen ID
- **st_status** (*bool*) – Enable/Disable SG Calculations
- **gauge_width** (*float*) – SG width
- **gauge_length** (*float*) – SG length
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns full stress-strain information

Return type pd.DataFrame

`openfdem.complete_UCS_thread_pool_generators.`**set_strain_gauge**(*model*, *gauge_length=None*, *gauge_width=None*)

Calculate local strain based on the dimensions of a virtual strain gauge placed at the center of teh model with x/y dimensions. By default set to 0.25 of the length/width.

Parameters

- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **gauge_length** (*float*) – length of the virtual strain gauge
- **gauge_width** (*float*) – width of the virtual strain gauge

Returns Cells that cover the horizontal and vertical gauges as well as the gauge width and length

Return type [list, list, float, float]

1.1.6 openfdem.extract_cell_thread_pool_generators module

`openfdem.extract_cell_thread_pool_generators.`**`history_cellinfo_func`**(*f_name*, *model*, *cell_id*, *array_needed*)

Generate a dictionary of the various array being interrogated for the said cell ID

Parameters

- **`f_name`** (*`str`*) – name of vtu file being processed
- **`model`** (`openfdem.openfdem.Model`) – FDEM Model Class
- **`cell_id`** (*`int`*) – ID of the cell from which the data needs to be extracted
- **`array_needed`** (*`list[str]`*) – Name of the property to extract

Returns The value of the property from the cell being extracted

Return type Generator[Tuple()]

`openfdem.extract_cell_thread_pool_generators.`**`main`**(*model*, *cellid*, *arrayname*, *progress_bar=False*)

Main concurrent Thread Pool to get value of the property from the cell being extracted

Parameters

- **`model`** (`openfdem.openfdem.Model`) – FDEM Model Class
- **`cellid`** (*`int`*) – ID of the cell from which the data needs to be extracted
- **`arrayname`** (*`list[str]`*) – Name of the property to extract
- **`progress_bar`** – Show/Hide progress bar

Returns DataFrame of the values of the property from the cell being extracted

Return type pandas.DataFrame

1.1.7 openfdem.formatting_codes module

`openfdem.formatting_codes.`**`bold_text`**(*val*)

Returns text as bold

Parameters **`val`** (*`str`*) – Text

Returns Text as bold

Return type str

`openfdem.formatting_codes.`**`calc_timer_values`**(*end_time*)

Function to calculate the time

Parameters **`end_time`** (*`float`*) – Time (Difference in time in seconds)

Returns Time in minutes and seconds

Return type float

`openfdem.formatting_codes.`**`docstring_creator`**`(`*df*`)`
Write the example output for a doctstring DataFrame

Parameters **`df`** (*`pandas.DataFrame`*) – DataFrame to be read

Returns prints the docstring and type for each element in the DataFrame

Return type str

`openfdem.formatting_codes.`**`green_text`**`(`*val*`)`
Returns text as bold in green font color

Parameters **`val`** (*`str`*) – Text

Returns Text as bold in green font color

Return type str

`openfdem.formatting_codes.`**`print_progress`**`(`*iteration*, *total*, *prefix=''*, *suffix=''*, *decimals=1*, *bar_length=50*`)`
Call in a loop to create terminal progress bar Adjusted bar length to 50, to display on small screen

Parameters

- **`iteration`** (*`int`*) – current iteration
- **`total`** (*`int`*) – total iteration
- **`prefix`** (*`str`*) – prefix string
- **`suffix`** (*`str`*) – suffix string
- **`decimals`** (*`int`*) – positive number of decimals in percent complete
- **`bar_length`** (*`int`*) – character length of bar

Returns system output showing progress

Return type

`openfdem.formatting_codes.`**`red_text`**`(`*val*`)`
Returns text as bold in red font color

Parameters **`val`** (*`str`*) – Text

Returns Text as bold in red font color

Return type str

1.1.8 openfdem.model_reader module

`openfdem.model_reader.`**`mp_read`**`(`**args*, ***kwargs*`)`

`openfdem.model_reader.`**`multiprocess_async`**`(`**args*, ***kwargs*`)`

`openfdem.model_reader.`**`multiprocess_lib_read`**`(`**args*, ***kwargs*`)`

`openfdem.model_reader.`**`normal_read`**`(`**args*, ***kwargs*`)`

`openfdem.model_reader.`**`pv_read`**`(`**args*, ***kwargs*`)`

`openfdem.model_reader.`**`pv_read_queue`**`(`*list_of_files*, *q*`)`

`openfdem.model_reader.`**`timed`**`(`*func*`)`

1.1.9 openfdem.rotary_ds_thread_pool_generators module

openfdem.rotary_ds_thread_pool_generators.**abs_sum_array**(*f_name*, *model*, *array*, *edge_list*)

Calculates the absolute sum of the array being interrogated in the sample.

Parameters

- **f_name** (*str*) – name of vtu file being processed
- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **array** (*str*) – the name of the array to be extracted
- **edge_list** (*Dict[str, int]*) – dictionary of location:integers that represent the nodes of the synthetic sample.

Returns list of the absolute sum of the array being interrogated.

Return type list[float]

Raises IndexError – Unknown Location

openfdem.rotary_ds_thread_pool_generators.**main**(*model*, *platen_id*, *array*, *progress_bar=True*)

Main concurrent Thread Pool to calculate the absolute sum of data along edge of sample.

Parameters

- **model** (openfdem.openfdem.Model) – FDEM Model Class
- **platen_id** (*int*) – Manual override of Platen ID
- **array** (*str*) – the name of the array to be extracted
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns Absolute sum of the extracted array split in Top/Bottom ane Left/Rigth sub-set into Top/Bottom.

Return type pd.DataFrame

openfdem.rotary_ds_thread_pool_generators.**sub_filter**(*vtk_data*, *y_middle*)

Identify the points on the top/bottom half of the sample. Assumes symmetric sample and the middle Y is the center.

Parameters

- **vtk_data** (*pyvista.core.pointset.UnstructuredGrid*) – Pointset of the data being filtered
- **y_middle** (*float*) – mid-point of Y based on sample bounds

Returns list of integers that represent the nodes on the top and bottom halves of the DS synthetic sample.

Return type list[int]

1.1.10 Module contents

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

N

O

P

R

S

T

U