
FDEM Post-Processing

Release 0.2

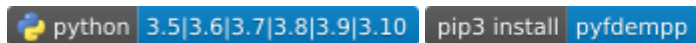
Grasselli's Geomechanics Group - University of Toronto

Feb 24, 2023

CONTENTS:

1	pyfdempp - An FDEM Visualiser Post-Processing Package	1
1.1	About	1
1.2	Functionality	1
1.3	Additional Support	6
2	pyfdempp	7
2.1	pyfdempp package	7
3	Indices and tables	33
	Python Module Index	35
	Index	37

PYFDEMPP - AN FDEM VISUALISER POST-PROCESSING PACKAGE



1.1 About

This Python package performs transformations on hybrid finite-discrete element method (FDEM) models with an unstructured grid in vtk/vtu/vtp format. It currently supports arrays of simulation files from the FDEM solvers:

- Geomechanica's Irazu software,
- Y-Geo (and its common derivatives), as well as
- OpenFDEM.

The package is heavily dependent on `pyvista` and is limited to Python ≥ 3.5 . The package is maintained by the [Grasselli's Geomechanics Group](#) at the University of Toronto, Canada, and is part of a collaborative effort by the open-source package `OpenFDEM`.

1.2 Functionality

The functionality of this script was developed with the objective of extracting common information needed when running simulations. Highlights of the script are:

- Get model information.

```
import pyfdempp as fd
model = fd.Model("abs_model_path_on_machine")
# Getting number of points in your model.
model.n_points
```

Output:

```
11904
```

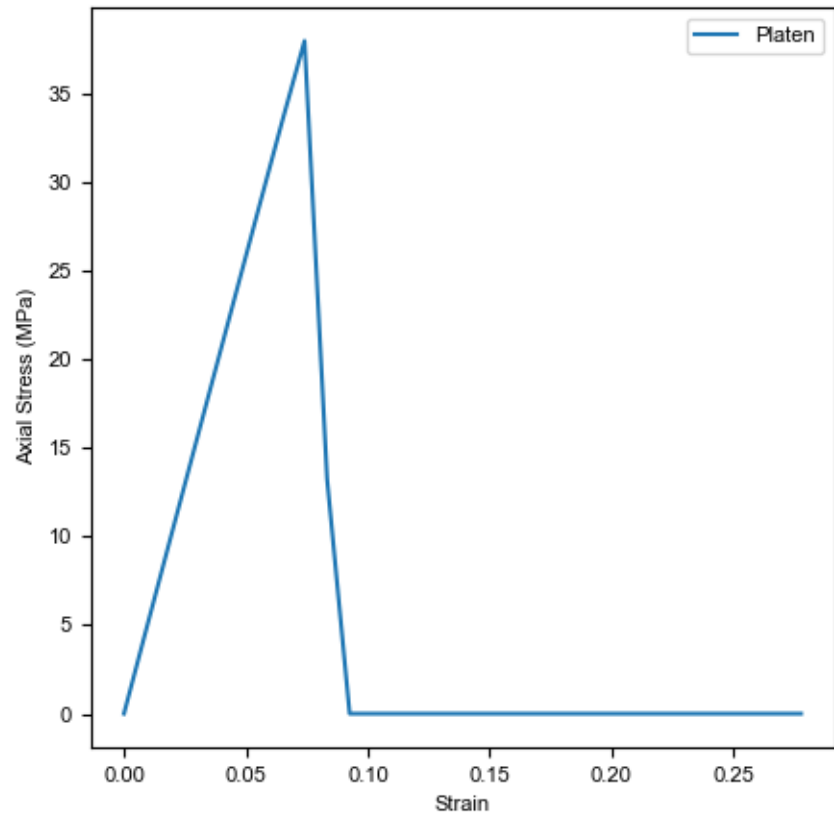
- Extract information within the FDEM Model based on the name of the array (e.g., Stress, Strain, Temperature, etc...) Works in 2D and 3D.
- Extract stress-strain information for UCS and BD Simulations (Works in 2D and 3D). Optional addition of virtual strain gauges (Limited to 2D).

```
import pyfdempp as fd
model = fd.Model("abs_model_path_on_machine")
model.complete_stress_strain(progress_bar=True)
```

Output:

```
Script Identifying Platen
Platen Material ID found as [1]
Progress: |////////////////////| 100.0% Complete
1.51 seconds
    Platen Stress   Platen Strain
0      0.000000e+00      0.000000
1      4.825237e+00      0.009259
2      9.628823e+00      0.018519
3      1.441437e+01      0.027778
4      1.919164e+01      0.037037
..      ...           ...
57     2.036137e-30      0.240741
58     2.036137e-30      0.250000
59     2.036137e-30      0.259259
60     2.036137e-30      0.268519
61     2.036137e-30      0.277778

[62 rows x 2 columns]
```



- Plotting stress vs strain curve.
- Calculate the Elastic Modulus of the dataset. Eavg, Esec and Etan can be evaluated. Works in 2D and 3D.

```
import pyfdempp as fd
model = fd.Model("abs_model_path_on_machine")
df_1 = model.complete_UCS_stress_strain(st_status=True)

# Variants of E tangent
print('Etan at 50%%: %.2fMPa' % model.Etan50_mod(df_1)[0])
print('Etan at 50%% with linear best fit disabled: %.2fMPa' % model.Etan50_mod(df_1, ↵
↵linear_bestfit=False)[0])
print('Etan at 50%% using strain gauge data: %.2fMPa' % model.Etan50_mod(df_1, loc_
↵strain='Gauge Displacement Y', plusminus_range=1)[0])
# Variants of E secant
print('Esec at 70%%: %.2fMPa' % model.Esec_mod(df_1, 70))
print('Esec at 50%%: %.2fMPa' % model.Esec_mod(df_1, 0.5))
# Variants of E average
print('Eavg between 50-60%%: %.2fMPa' % model.Eavg_mod(df_1, 0.5, 0.6)[0])
print('Eavg between 20-70%% with linear best fit disabled: %.2fMPa' % model.Eavg_mod(df_
↵1, 0.2, 0.7, linear_bestfit=False)[0])
```

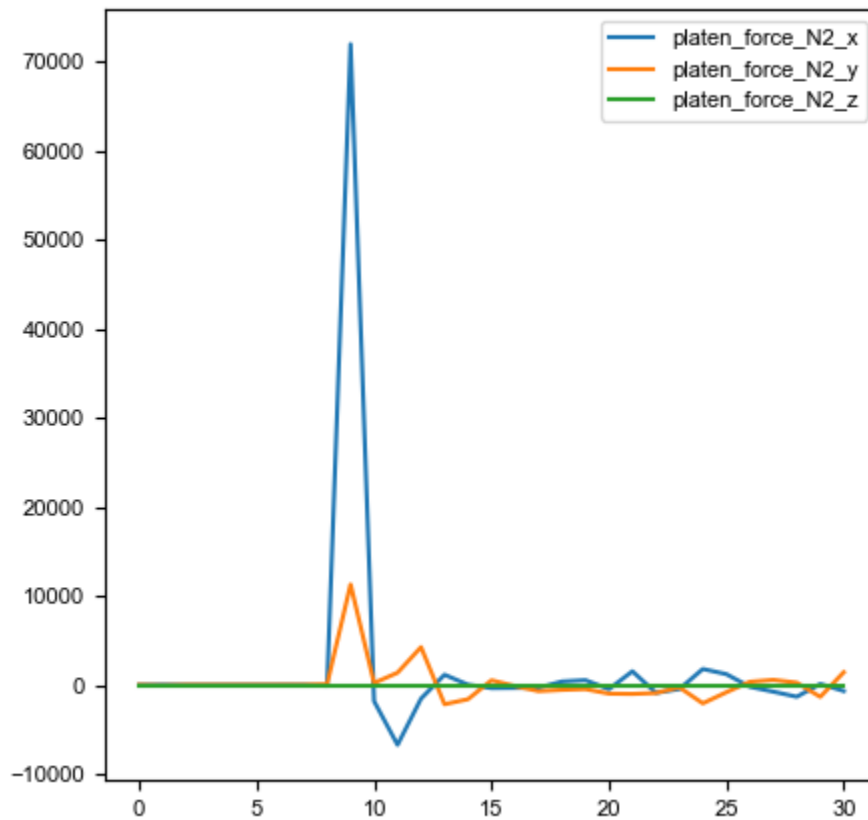
Output:

```
Etan at 50%: 51683.94MPa
Etan at 50% with linear best fit disabled: 51639.22MPa
Etan at 50% using strain gauge data: 50275.03MPa

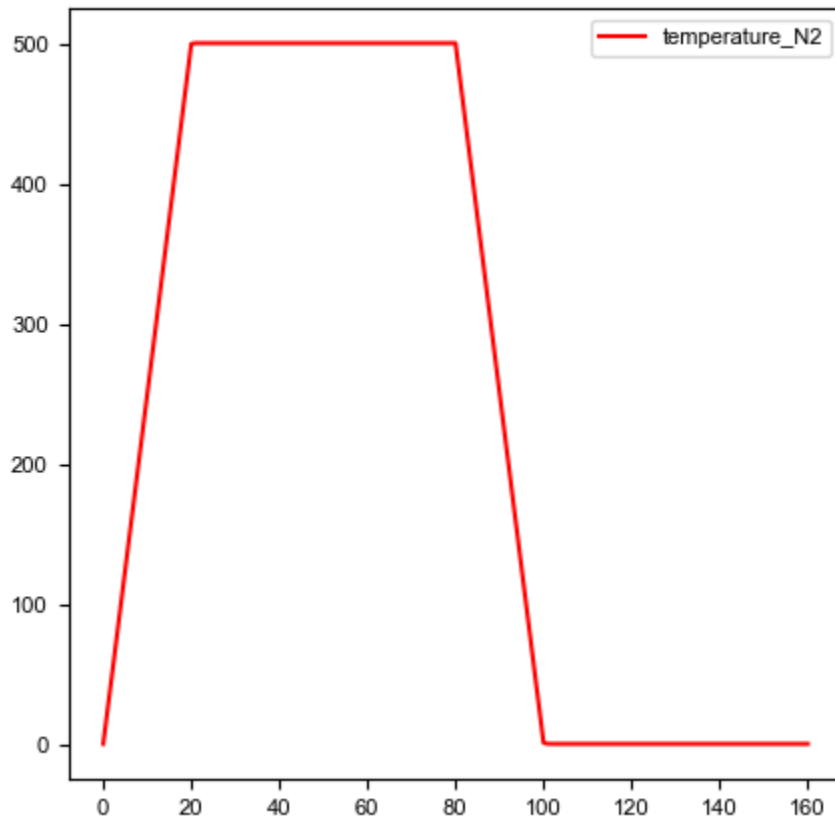
Esec at 70%: 51681.01MPa
Esec at 50%: 51817.43MPa

Eavg between 50-60%: 51594.49MPa
Eavg between 20-70% with linear best fit disabled: 51660.62MPa
```

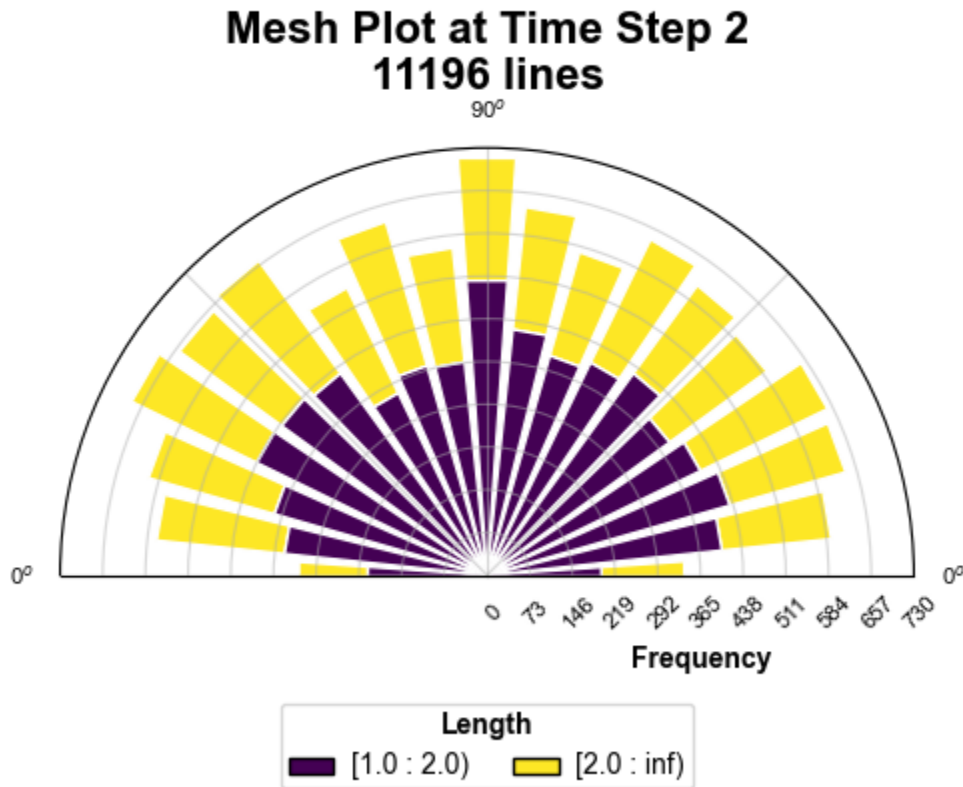
- Extract information of a particular cell based on a sequence of array names. This can be extended to extracting information along a line. Works in 2D and 3D.



- Extract information of a threshold dataset criteria based on a sequence of array names. Works in 2D and 3D.



- Extract mesh information and plot rosette/polar plots. Limited to 2D.



- Automatic detection/ User-defined assignment of loading direction when analysing mechanical simulations, namely UCS, BD, and PLT, in both 2D and 3D simulations.

```
Script Identifying Platen
  Platen Material ID found as [1]
  3D Loading direction detected as [1] is Y-direction
Values used in calculations are
  Area      3721.00
  Length    122.00
Progress: |////////////////////| 100.0% Complete
```

1.3 Additional Support

Please refer to the user manual for detailed information pertaining to the various functions and their usage/arguments. For specific script requests and bug, please report them on our [github page](#).

2.1 pyfdempp package

2.1.1 Submodules

pyfdempp.pyfdempp

class pyfdempp.pyfdempp.**Model**(*dir_path=None, runfile=None, fdem_engine=None*)

Bases: object

Collects datafiles into one Class. Returns the data array ordered by simulation timestep.

Example

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

Eavg_mod(*ucs_data, upperrange, lowerrange, linear_bestfit=True, loc_stress='Platen Stress', 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
- **linear_bestfit** (*bool*) – Calculate data based on range extents or linear best fit line.
- **loc_stress** (*str*) – Column to obtain stress from. Defaults to Platen Stress
- **loc_strain** (*str*) – Column to obtain strain from. Defaults to Platen Strain

Returns

Average Elastic modulus

Return type

list[float]

Raises

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

Example

```

>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> df_1 = data.complete_UCS_stress_strain(st_status=True)
>>> data.Eavg_mod(df_1, 0.5, 0.6)[0]
51594.490217007056
>>> data.Eavg_mod(df_1, 0.5, 0.6, loc_strain='Gauge Displacement Y
↪')[0]
51110.06292512191

```

Esec_mod(*ucs_data*, *upperrange*, *loc_stress*='Platen Stress', *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_stress** (*str*) – Column to obtain stress from. Defaults to Platen Stress
- **loc_strain** (*str*) – Column to obtain strain from. Defaults to Platen Strain

Returns

Secant Elastic modulus between 0 and upperrange

Return type

float

Example

```

>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> df_1 = data.complete_UCS_stress_strain(st_status=True)
>>> data.Esec_mod(df_1, 0.5)
51817.43019752671
>>> data.Esec_mod(df_1, 0.5, loc_strain='Gauge Displacement Y')
51355.860814069754

```

Etan50_mod(*ucs_data*, *linear_bestfit*=True, *loc_stress*='Platen Stress', *loc_strain*='Platen Strain', *plusminus_range*=1)

Tangent Elastic modulus at 50%. Calculates +/- number of datapoint from the 50% Stress. Defaults to +/- 1 datapoint.

Parameters

- **ucs_data** (*pandas.DataFrame*) – DataFrame containing the stress-strain data
- **linear_bestfit** (*bool*) – Calculate data based on range extents or linear best fit line.
- **loc_stress** (*str*) – Column to obtain stress from. Defaults to Platen Stress
- **loc_strain** (*str*) – Column to obtain strain from. Defaults to Platen Strain
- **plusminus_range** (*int*) – Range over which to calculate the Elastic modulus

Returns

Tangent Elastic modulus at 50% as a slope and Y-Intercept. Y-Intercept = 0 if *linear_bestfit* is False

Return type

list[float]

Example

```
>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> df_1 = data.complete_UCS_stress_strain()
>>> data.Etan50_mod(df_1)[0]
51683.94337878284
>>> data.Etan50_mod(df_1, linear_bestfit=False)[0]
51639.21679789497
>>> df_1 = data.complete_UCS_stress_strain(st_status=True)
>>> data.Etan50_mod(df_1, loc_strain='Gauge Displacement Y',
↳ plusminus_range=1)[0]
51216.33411269702
```

`clip_data(progress_bar=True)`

`complete_BD_stress_strain(platen_id=None, st_status=False, axis_of_loading=None, gauge_width=0, gauge_length=0, c_center=None, user_samp_A=None, user_samp_L=None, progress_bar=True)`

Calculate the full stress-strain curve for an indirect tensile simulation.

Parameters

- **platen_id** (*None* or *int*) – Manual override of Platen ID
- **st_status** (*bool*) – Enable/Disable SG
- **axis_of_loading** (*None* or *int*) – Loading Direction
- **gauge_width** (*float*) – width of the virtual strain gauge
- **gauge_length** (*float*) – length of the virtual strain gauge
- **c_center** (*None* or *list[float, float, float]*) – User-defined center of the SG
- **samp_A** (*None* or *float*) – Sample Thickness
- **samp_L** (*None* or *float*) – Sample Diameter
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

full stress-strain information

Return type

pandas.DataFrame

Example

```
>>> import pyfdempp as fdem
>>> data = fdem.Model("/external/Speed_Cal_Using_Flowstone/BD/BD_c_17_
↳ 5_ts_2_55_GII_90000_v_0_6")
# full stress-strain without SG
>>> df_wo_SG = data.complete_BD_stress_strain(st_status=False)
Columns:
  Name: Platen Stress, dtype=float64, nullable: False
  Name: Platen Strain, dtype=float64, nullable: False
Script Identifying Platen
  Platen Material ID found as [1]
Progress: |////////////////////| 100.0%
↳ Complete
```

(continues on next page)

(continued from previous page)

```

# full stress-strain with SG and default dimensions
>>> df_Def_SG = data.complete_BD_stress_strain(st_status=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
  Script Identifying Platen
Platen Material ID found as [1]
Dimensions of SG are 12.0 x 12.0
Vertical Gauges
  extends between [[6.0, -6.0, 0.0], [-6.0, -6.0, 0.0], [6.0, 6.0, 0.0], [-6.0, 6.0, 0.0]]
  cover cells ID [3644, 7481, 4635, 2872]
Horizontal Gauges
  extends between [[6.0, 6.0, 0.0], [6.0, -6.0, 0.0], [-6.0, 6.0, 0.0], [-6.0, -6.0, 0.0]]
  cover cells ID [4635, 3644, 2872, 7481]
Progress: | 100.0%
Complete
# full stress-strain with SG and user-defined dimensions
>>> df_userdf_SG = data.complete_BD_stress_strain(st_status=True, gauge_width=10, gauge_length=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
  Script Identifying Platen
Platen Material ID found as [1]
Dimensions of SG are 10 x 10
Vertical Gauges
  extends between [[5.0, -5.0, 0.0], [-5.0, -5.0, 0.0], [5.0, 5.0, 0.0], [-5.0, 5.0, 0.0]]
  cover cells ID [1898, 5999, 5249, 6806]
Horizontal Gauges
  extends between [[5.0, 5.0, 0.0], [5.0, -5.0, 0.0], [-5.0, 5.0, 0.0], [-5.0, -5.0, 0.0]]
  cover cells ID [5249, 1898, 6806, 5999]
Progress: | 100.0%
Complete

```

complete_PLT_stress_strain(*load_config*, *platen_id*=None, *axis_of_loading*=None, *De_squared*=None, *progress_bar*=True)

Calculate the full stress-strain curve for a point load simulation.

Parameters

- **load_config** (*str*) – type of PLT Test. “A” “D” “B”
- **platen_id** (*None* or *int*) – Manual override of Platen ID
- **axis_of_loading** (*None* or *int*) – Loading Direction
- **De_squared** (*None* or *float*) – equivalent core diameter (i.e., the value of *De_squared*)

- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

full stress-strain information

Return type

pandas.DataFrame

Example

```
>>> import pyfdempp as fdem
>>> data = fdem.Model("/external/Yusuf_PLT/Axial")
# Minimal Arguments
>>> df = data.complete_PLT_stress_strain(load_config="A", platen_id=1)
Columns:
      Name: Platen Stress, dtype=float64, nullable: False
```

complete_UCS_stress_strain(*platen_id=None, st_status=False, axis_of_loading=None, gauge_width=0, gauge_length=0, c_center=None, samp_A=None, samp_L=None, progress_bar=True*)

Calculate the full stress-strain curve for a uniaxial compressive strength simulation

Parameters

- **platen_id** (*None or int*) – Manual override of Platen ID
- **st_status** (*bool*) – Enable/Disable SG
- **axis_of_loading** (*None or int*) – Loading Direction
- **gauge_width** (*float*) – width of the virtual strain gauge
- **gauge_length** (*float*) – length of the virtual strain gauge
- **c_center** (*None or list[float, float, float]*) – User-defined center of the SG
- **samp_A** (*None or float*) – Sample Area
- **samp_L** (*None or float*) – Sample Length
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

full stress-strain information

Return type

pandas.DataFrame

Example

```
>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
# Minimal Arguments
>>> df_wo_SG = data.complete_UCS_stress_strain()
Columns:
      Name: Platen Stress, dtype=float64, nullable: False
      Name: Platen Strain, dtype=float64, nullable: False
Script Identifying Platen
      Platen Material ID found as [1]
      Predefined loading Axis [1] is Y-direction
Values used in calculations are
```

(continues on next page)

(continued from previous page)

```

Area      52.00
Length    108.00
Progress: |////////////////////////////////////////////////////////////////| 100.0%
↳Complete
# full stress-strain without SG
>>> df_Def_SG = data.complete_UCS_stress_strain(None, 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
Script Identifying Platen
Platen Material ID found as [1]
  Predefined loading Axis [1] is Y-direction
Values used in calculations are
  Area      52.00
  Length    108.00
  Dimensions of SG are 27.0 x 13.0
  Vertical Gauges
    extends between [[6.5, -13.5, 0.0], [-6.5, -13.5, 0.0],
↳[6.5, 13.5, 0.0], [-6.5, 13.5, 0.0]]
    cover cells ID [2744, 1377, 3466, 3789]
  Horizontal Gauges
    extends between [[13.5, 6.5, 0.0], [13.5, -6.5, 0.0], [-
↳13.5, 6.5, 0.0], [-13.5, -6.5, 0.0]]
    cover cells ID [2089, 1582, 2210, 1504]
  Progress: |////////////////////////////////////////////////////////////////|
↳100.0% Complete
# full stress-strain with SG and user-defined dimensions
>>> df_userdf_SG = data.complete_UCS_stress_strain(None, True, gauge_
↳width=10, gauge_length=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
Script Identifying Platen
Platen Material ID found as [1]
  Predefined loading Axis [1] is Y-direction
Values used in calculations are
  Area      52.00
  Length    108.00
  Dimensions of SG are 10 x 10
  Vertical Gauges
    extends between [[5.0, -5.0, 0.0], [-5.0, -5.0, 0.0], [5.0, 5.
↳0, 0.0], [-5.0, 5.0, 0.0]]
    cover cells ID [1186, 1397, 1669, 1148]
  Horizontal Gauges
    extends between [[5.0, 5.0, 0.0], [5.0, -5.0, 0.0], [-5.0, 5.
↳0, 0.0], [-5.0, -5.0, 0.0]]
    cover cells ID [1669, 1186, 1148, 1397]
Progress: |////////////////////////////////////////////////////////////////| 100.0%
↳Complete

```

(continues on next page)

(continued from previous page)

convert_to_xyz_array(*node_df*)

Convert extracted node information into summation based on X, Y and Z

Parameters**node_df** (*pandas.DataFrame*) – Extracted node information**Returns**

A DataFrame with summations along X, Y, Z axis. Column names are ["sum_X", "sum_Y", "sum_Z"]

Return type*pandas.DataFrame***Example**

```

>>> import pyfдемpp as fдем
>>> data = fдем.Model("../example_outputs/Irazu_3D_UCS")
>>> # # Extract all cells that meet the criteria and split to_
↳ nodewise data for each time step.
>>> # In this case "BOUNDARY CONDITION" is set to "1" for the_
↳ threshold with the "FORCE" being extracted at each node.
>>> df = data.extract_threshold_info(thres_id=1, thres_array='boundary
↳ ', arrays_needed=['platen_force'])
>>> # Sum the X,Y,Z of all nodes for each time step.
>>> df_sum = data.convert_to_xyz_array(df)
>>> print(df_sum)

```

	sum_X	sum_Y	sum_Z
0	0.000000e+00	0.000000e+00	0.000000e+00
1	-1.224291e+05	-2.348118e+09	4.645789e+04
2	-8.768720e+04	-4.663436e+09	7.953211e+03
3	-5.580583e+04	-6.948494e+09	-1.039933e+04
4	-1.602602e+05	-9.240063e+09	1.065935e+04
5	-1.588623e+05	-1.152608e+10	4.616695e+04
...			

crack_failure_mode(*remove_boundary=True, progress_bar=True*)

Get all failure modes in every timestep. Boundaries Optional. This will create a dataframe that has the failure mode for every crack. The time the crack first appears would be the time it initiated.

Parameters

- **remove_boundary** (*bool*) – Optional. Keep or remove boundaries in DataFrame
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

A DataFrame containing the failure mode for each crack at every time step.

Return type*pd.DataFrame***Example**

```

>>> import pyfдемpp as fдем
>>> data = fдем.Model("../example_outputs/Irazu_UCS")
>>> df_cracks = data.crack_failure_mode()

```

crack_failure_mode_clustering(*crack_LUT=None, crack_LUT_name=None, remove_boundary=True, progress_bar=True*)

Cluster the crack modes based on their failure mode values. By default, it uses the Irazu convention which is 1=pure tensile; 1-1.5= tensile dominant; 1.5-2= shear dominant, 2= pure shear; 3= mixed mode. The crack LUT can also be user defined to get a specific range within the dataset.

Parameters

- **crack_LUT** (*list[float]*) – LUT range for the crack failure (float)
- **crack_LUT_name** (*list[str]*) – LUT range for the crack failure (name)
- **remove_boundary** (*bool*) – Optional. Keep or remove boundaries in DataFrame
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

A DataFrame containing the total number of cracks in every time step.

Return type

pd.DataFrame

Example

```
>>> import pyfдемpp as fдем
>>> data = fдем.Model("../example_outputs/Irazu_UCS")
>>> df_crack_default_clustering = data.crack_failure_mode_clustering()
>>> df_crack_userdefined_clustering = data.crack_failure_mode_clustering(crack_
↳ LUT=[1,2], crack_LUT_name=['tensile_and_shear'])
```

crack_number(*progress_bar=True*)

Get the total number of cracks at every timestep. Excluding boundaries.

Parameters

- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

A DataFrame containing the total number of cracks at every time step.

Return type

pd.DataFrame

Example

```
>>> import pyfдемpp as fдем
>>> data = fдем.Model("../example_outputs/Irazu_UCS")
>>> crack_number_total = data.crack_number()
>>> max(crack_number_total['crack number'])
222.0
>>> crack_number_total
      crack number
0             0.0
1             0.0
2             0.0
3             0.0
4             0.0
5             0.0
6             0.0
7             0.0
```

(continues on next page)

(continued from previous page)

8	0.0
9	117.0
10	212.0
11	219.0
12	221.0
13	221.0
14	221.0
15	221.0
16	221.0
17	221.0
18	222.0
19	222.0
20	222.0
21	222.0
22	222.0
23	222.0
24	222.0
25	222.0
26	222.0
27	222.0
28	222.0
29	222.0
30	222.0

direct_shear_calculation(*platen_id*, *array*, *progress_bar=True*)

Analyse direct shear simulation built with a rigid platen on the outside.

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 and Left/Right sub-set into Top/Bottom.

Return type

pandas.DataFrame

Example

```
>>> import pyfдемpp as fдем
>>> data = fдем.Model("/external/2D_shear_4mm_profile_normal_load_test")
>>> df = data.direct_shear_calculation(platen_id=1, array='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
```

(continues on next page)

(continued from previous page)

```

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

```

draw_rose_diagram(*t_step*, *rose_data*=None, *thres_id*=None, *thres_array*='mineral_type',
rose_range='Length')

Draw a wind rose diagram based on the information passed.

Parameters

- **t_step** (*int*) – Time step in model. Default 0
- **rose_data** (*DataFrame*) – User can bypass requirement and pass a DataFrame with the data. Should be 2 columns with the Angle being the 2nd. Default None
- **thres_id** (*int*) – ID of item to threshold. Default None.
- **thres_array** (*str*) – Array name of item to threshold. Default “mineral_type”.
- **rose_range** (*str*) – Range to calculate the windrose bins. Default “length”

Returns

windrose figure

Return type

matplotlib.pyplot

Example

```

>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> data.draw_rose_diagram(t_step=0)
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.8/dist-packages/
↳matplotlib/pyplot.py'>
>>> data.draw_rose_diagram(t_step=0, rose_range='Length', thres_id=0, thres_
↳array='boundary')
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.8/dist-packages/
↳matplotlib/pyplot.py'>
>>> # If you want to save the figure to a pyplot format.
>>> figure_name = data.draw_rose_diagram(t_step=0, rose_range='Length', thres_
↳id=0, thres_array='boundary')

```

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

Example

```
>>> import pyfдемpp as fдем
>>> data = fдем.Model("/external/2D_shear_4mm_profile_normal_load_test
↳")
>>> data.rock_sample_dimensions()
Script Identifying Platen
Platen Material ID found as [1]
(31.713071, 30.111493, 0.0, [-0.2, 31.513071, -14.92642, 15.185073, 0.
↳0, 0.0])
>>> extracted_left = data.extract_based_coord(data.rock_model, 0,
↳data.rock_model.bounds[0])
```

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

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 pyfдемpp as fдем
>>> import matplotlib.pyplot as plt
>>> data = fдем.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:
```

(continues on next page)

(continued from previous page)

```

Name: platen_force_N1, dtype=object, nullable: False
Name: platen_force_N2, dtype=object, nullable: False
Name: platen_force_N3, dtype=object, nullable: False
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')
[<matplotlib.lines.Line2D object at 0x7f08fe98a310>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x7f08fe9854c0>
>>> 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')
[<matplotlib.lines.Line2D object at 0x7f08fe859b20>]
>>> plt.plot(lx1[1], label='platen_force_N2_y')
[<matplotlib.lines.Line2D object at 0x7f08fe859e50>]
>>> plt.plot(lx1[2], label='platen_force_N2_z')
[<matplotlib.lines.Line2D object at 0x7f08fe86a160>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x7f08fe86a340>
>>> plt.show()
# For non-noded information
>>> plt.plot(lx1[0], label='mineral_type')
[<matplotlib.lines.Line2D object at 0x7f08fe7e39a0>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x7f08fe7e39d0>
>>> plt.show()

```

extract_crack_info(arrays_needed, progress_bar=True)

Returns the information based on the array requested.

Parameters

- **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 pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> extraction_of_cracks = data.extract_crack_info(arrays_needed=[
↳ 'area', 'length'])
Progress: |////////////////////| 100.0%_
↳ Complete

```

(continues on next page)

(continued from previous page)

```
extract_threshold_info(thres_id, thres_array, arrays_needed, dataset_to_load='basic',
                        progress_bar=True)
```

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 cell ID. Also shows a quick example on how to plot the information extracted.

Parameters

- **thres_id** (*int*) – Threshold ID to extract
- **thres_array** (*str*) – Array name of item to threshold.
- **arrays_needed** (*list[str]*) – list of array names to extract
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

A DataFrame or a series of DataFrames nested in a dictionary with the key being the name of the array needed

Return type

pandas.DataFrame or dict[pandas.DataFrame]

Example

```
>>> import pyfdempp as fdem
>>> import matplotlib.pyplot as plt
>>> data = fdem.Model("../example_outputs/Irazu_3D_UCS")
>>> # Extract all cells that meet the criteria and split to nodewise_
↳data for each time step.
>>> # In this case "BOUNDARY CONDITION" is set to "1" for the_
↳threshold with the "FORCE" being extracted at each node.
>>> df = data.extract_threshold_info(thres_id=1, thres_array='boundary
↳', arrays_needed=['platen_force'])
Progress: |////////////////////| 100.0%_
↳Complete
54.74 seconds.
>>> # Sum the X,Y,Z of all nodes for each time step.
>>> df_sum = data.convert_to_xyz_array(df)
```

```
find_cell(model_point)
```

Identify the containing cell in the model to the defined point. Will return an error if the point is not within a cell.

Parameters

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

Returns

The cell that contains the point.

Return type

int

Raises

IndexError – Point outside model domain.

Example

```

>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> data.find_cell([0, 0, 0])
2167
>>> data.find_cell([2000, 2000, 0])
IndexError: Point outside model domain.
X=56.0, Y=116.0, Z=0.0

```

mesh_geometry(vertices)

Returns a unique set of vertices and calculates their length and orientation.

Parameters

vertices (*list[tuples]*) – list of vertices in the model at a given time step

Returns

DataFrame of the vertices length and orientation

Return type

pandas.DataFrame

Example

```

>>> import pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
>>> vert = data.model_vertices(t_step=0, thres_id=1, thres_array='mineral_type')
>>> data.mesh_geometry(vert)

```

	Length	Angle
0	2.236068	63.434949
1	2.000000	0.000000
2	2.363608	59.436301
3	2.000000	0.000000
4	2.244731	117.123188
..
409	2.116948	0.287685
410	2.000000	0.000000
411	2.000000	0.000000
412	1.802781	45.829911
413	2.227619	116.002627

[414 rows x 2 columns]

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 pyfdempp as fdem
>>> model = fdem.Model("../example_outputs/Irazu_UCS")

```

(continues on next page)

(continued from previous page)

```

>>> # 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**Warning** – Simulation partially supported.**Example**

```

>>> import pyfдемpp as fдем
>>> model = fдем.Model("../example_outputs/Irazu_UCS")
>>> model.model_domain()
2D Simulation
4

```

model_vertices(t_step=0, thres_id=None, thres_array='mineral_type')

Returns a list of the vertices in the form of Point1, Point 2

Parameters

- **t_step** (*int*) – Time step in model. Default 0
- **thres_id** (*int*) – ID of item to threshold. Default None.
- **thres_array** (*str*) – Array name of item to threshold. Default “mineral_type”.

Returns

list of the vertices in the model and/or the threshold of it.

Return type

list[tuples]

Example

```

>>> import pyfдемpp as fдем
>>> data = fдем.Model("../example_outputs/Irazu_UCS")
>>> len(data.model_vertices(t_step=0, thres_id=0, thres_array='mineral_type'))
11196

```

(continues on next page)

(continued from previous page)

```
>>> len(data.model_vertices(t_step=0, thres_id=1, thres_array='boundary'))
354
```

openfдем_att_check(att, dict_to_check=None)

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 pyfдемpp as fдем
>>> model = fдем.Model("../example_outputs/Irazu_UCS")
>>> model.openfдем_att_check('mineral_type')
'mineral_type'
>>> model.openfдем_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 or DataSet*) –
- **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 of a given dataframe

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 pyfdempp as fdem
>>> data = fdem.Model("../example_outputs/Irazu_UCS")
# Minimal Arguments
>>> df_wo_SG = data.complete_UCS_stress_strain()
Columns:
  Name: Platen Stress, dtype=float64, nullable: False
  Name: Platen Strain, dtype=float64, nullable: False.
  Script Identifying Platen
Platen Material ID found as [1]
  UCS Simulation
    Predefined loading Axis [1] is Y-direction
  Values used in calculations are
    Area      52.00
    Length    108.00
Progress: |////////////////////| 100.0%
↳ Complete
>>> 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 ID of thresholding

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 pyfdempp 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, [-26.0, 26.0, -54.0, 54.0, 0.0, 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, [-28.0, 28.0, -58.0, 58.0, 0.0, 0.0])
```

(continues on next page)

(continued from previous page)

```
>>> # Explicitly defined the platen material ID is out of range
>>> data.rock_sample_dimensions(3)
IndexError: Threshold ID out of range.
boundary Range -1-1
```

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 pyfdempp 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, [-26.0, 26.0, -54.0, 54.0, 0.0, 0.0])
>>> data.simulation_type()
'UCS Simulation'
>>> data_bd = fdem.Model("../example_outputs/openfдем_BD", runfile='.y
↳ ')
>>> data_bd.rock_sample_dimensions()
Script Identifying Platen
Platen Material ID found as [1]
(100.0, 99.94999694824219, 0.0, [-50.0, 50.0, -49.974998474121094, 49.
↳ 974998474121094, 0.0, 0.0])
>>> data_bd.simulation_type()
'BD Simulation'
```

slice_dataset(cell_inds, i_array_needed, progress_bar=True)

threshold_bound_check(thres_id, thres_array='boundary')

Checks the material ID is a valid choice.

Parameters

- **thres_id** (int) – ID of the item to be threshold
- **thres_array** (str) – Array name of the item to be threshold

Returns

ID of the material

Return type

int

Raises

IndexError – ID out of range.

Example

```

>>> import pyfдемpp as fдем
>>> model = fдем.Model("../example_outputs/Irazu_UCS")
>>> model.threshold_bound_check(0)
0
>>> model.threshold_bound_check(5)
IndexError: Material ID for platen out of range.
Material Range 0-1

```

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 pyfдемpp.pyfдемpp.**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

2.1.2 Supporting Modules

pyfдемpp.aggregate_storage module

class pyfдемpp.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

pyfdempp.complete_BD_thread_pool_generators module

pyfdempp.complete_BD_thread_pool_generators.**check_loading_direction**(*model, f1, f2*)

pyfdempp.complete_BD_thread_pool_generators.**history_strain_func**(*f_name, model, cv, ch, axis*)

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.pyfdempp.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
- **axis** (*list[int]*) – the axis of load application

Returns

Stress, Platen Strain, SG Strain, SG Lateral Strain

Return type

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

pyfdempp.complete_BD_thread_pool_generators.**main**(*model, platen_id, st_status, axis_of_loading, gauge_width, gauge_length, c_center, user_samp_A=None, user_samp_L=None, progress_bar=False*)

Main concurrent Thread Pool to calculate the full stress-strain

Parameters

- **model** (*openfdem.pyfdempp.Model*) – FDEM Model Class
- **platen_id** (*None or int*) – Manual override of Platen ID
- **st_status** (*bool*) – Enable/Disable SG Calculations
- **axis_of_loading** (*None or int*) – Enable/Disable SG
- **gauge_width** (*float*) – SG width
- **gauge_length** (*float*) – SG length
- **c_center** (*None or list[float, float, float]*) – User-defined center of the SG
- **user_samp_A** (*None or float*) – Sample Thickness

- **user_samp_L** (*None* or *float*) – Sample Diameter
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

full stress-strain information

Return type

pd.DataFrame

```
pyfdempp.complete_BD_thread_pool_generators.set_strain_gauge(model, gauge_length=None,
                                                             gauge_width=None,
                                                             c_center=None)
```

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

Parameters

- **model** (*openfдем.pyfdempp.Model*) – FDEM Model Class
- **gauge_length** (*float*) – length of the virtual strain gauge
- **gauge_width** (*float*) – width of the virtual strain gauge
- **c_center** (*None* or *list[float, float, float]*) – User-defined center of the SG

Returns

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

Return type

[list, list, float, float]

pyfdempp.complete_UCS_thread_pool_generators module

```
pyfdempp.complete_UCS_thread_pool_generators.check_loading_direction(model, f1, f2)
```

```
pyfdempp.complete_UCS_thread_pool_generators.history_strain_func(f_name, model, cv, ch, axis)
```

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** (*openfдем.pyfdempp.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
- **axis** (*list[int]*) – the axis of load application

Returns

Stress, Platen Strain, SG Strain, SG Lateral Strain

Return type

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

```
pyfdempp.complete_UCS_thread_pool_generators.main(model, platen_id, st_status, axis_of_loading,
                                                    gauge_width, gauge_length, c_center,
                                                    user_samp_A=None, user_samp_L=None,
                                                    progress_bar=False)
```

Main concurrent Thread Pool to calculate the full stress-strain

Parameters

- **model** (*openfдем.pyfдемpp.Model*) – FDEM Model Class
- **platen_id** (*None or int*) – Manual override of Platen ID
- **st_status** (*bool*) – Enable/Disable SG Calculations
- **axis_of_loading** (*None or int*) – Enable/Disable SG
- **gauge_width** (*float*) – SG width
- **gauge_length** (*float*) – SG length
- **c_center** (*None or list[float, float, float]*) – User-defined center of the SG
- **user_samp_A** (*None or float*) – Sample Area
- **user_samp_L** (*None or float*) – Sample Length
- **progress_bar** (*bool*) – Show/Hide progress bar

Returns

full stress-strain information

Return type

pd.DataFrame

```
pyfдемpp.complete_UCS_thread_pool_generators.set_strain_gauge(model, gauge_length=None,  
                                                             gauge_width=None,  
                                                             c_center=None)
```

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

Parameters

- **model** (*openfдем.pyfдемpp.Model*) – FDEM Model Class
- **gauge_length** (*float*) – length of the virtual strain gauge
- **gauge_width** (*float*) – width of the virtual strain gauge
- **c_center** (*None or list[float, float, float]*) – User-defined center of the SG

Returns

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

Return type

[list, list, float, float]

pyfдемpp.extract_cell_thread_pool_generators module

```
pyfдемpp.extract_cell_thread_pool_generators.history_cellinfo_func(f_name, model, cell_id,  
                                                                    array_needed,  
                                                                    thres_array=None)
```

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

Parameters

- **f_name** (*str*) – name of vtu file being processed
- **model** (*openfдем.pyfдемpp.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()]

`pyfdempp.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** (*openfдем.pyfdempp.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

pyfdempp.formatting_codes module

`pyfdempp.formatting_codes.bold_text(val)`

Returns text as bold

Parameters

val (*str*) – Text

Returns

Text as bold

Return type

str

`pyfdempp.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

`pyfdempp.formatting_codes.docstring_creator(df)`

Write the example output for a docstring DataFrame

Parameters

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

Returns

prints the docstring and type for each element in the DataFrame

Return type

str

`pyfdempp.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

`pyfdempp.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

`pyfdempp.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

pyfdempp.model_reader module

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

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

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

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

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

`pyfdempp.model_reader.pv_read_queue(list_of_files, q)`

`pyfdempp.model_reader.timed(func)`

Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- `pyfdempp`, [31](#)
- `pyfdempp.aggregate_storage`, [25](#)
- `pyfdempp.complete_BD_thread_pool_generators`,
[26](#)
- `pyfdempp.complete_UCS_thread_pool_generators`,
[27](#)
- `pyfdempp.extract_cell_thread_pool_generators`,
[28](#)
- `pyfdempp.formatting_codes`, [29](#)
- `pyfdempp.model_reader`, [30](#)
- `pyfdempp.pyfdempp`, [7](#)

INDEX

A

`aggregate_storage` (class `pyfdempp.aggregate_storage`), 25

B

`bold_text()` (in module `pyfdempp.formatting_codes`), 29

C

`calc_timer_values()` (in module `pyfdempp.formatting_codes`), 29

`check_loading_direction()` (in module `pyfdempp.complete_BD_thread_pool_generators`), 26

`check_loading_direction()` (in module `pyfdempp.complete_UCS_thread_pool_generators`), 27

`clip_data()` (`pyfdempp.pyfdempp.Model` method), 9

`complete_BD_stress_strain()` (`pyfdempp.pyfdempp.Model` method), 9

`complete_PLT_stress_strain()` (`pyfdempp.pyfdempp.Model` method), 10

`complete_UCS_stress_strain()` (`pyfdempp.pyfdempp.Model` method), 11

`convert_to_xyz_array()` (`pyfdempp.pyfdempp.Model` method), 13

`crack_failure_mode()` (`pyfdempp.pyfdempp.Model` method), 13

`crack_failure_mode_clustering()` (`pyfdempp.pyfdempp.Model` method), 13

`crack_number()` (`pyfdempp.pyfdempp.Model` method), 14

D

`direct_shear_calculation()` (`pyfdempp.pyfdempp.Model` method), 15

`docstring_creator()` (in module `pyfdempp.formatting_codes`), 29

`draw_rose_diagram()` (`pyfdempp.pyfdempp.Model` method), 16

E

`Eavg_mod()` (`pyfdempp.pyfdempp.Model` method), 7

`Esec_mod()` (`pyfdempp.pyfdempp.Model` method), 8

`Etan50_mod()` (`pyfdempp.pyfdempp.Model` method), 8

`extract_based_coord()` (`pyfdempp.pyfdempp.Model` method), 16

`extract_cell_info()` (`pyfdempp.pyfdempp.Model` method), 17

`extract_crack_info()` (`pyfdempp.pyfdempp.Model` method), 18

`extract_threshold_info()` (`pyfdempp.pyfdempp.Model` method), 19

F

`file_group_key()` (`pyfdempp.aggregate_storage.aggregate_storage` method), 25

`find_cell()` (`pyfdempp.pyfdempp.Model` method), 19

G

`green_text()` (in module `pyfdempp.formatting_codes`), 30

H

`history_cellinfo_func()` (in module `pyfdempp.extract_cell_thread_pool_generators`), 28

`history_strain_func()` (in module `pyfdempp.complete_BD_thread_pool_generators`), 26

`history_strain_func()` (in module `pyfdempp.complete_UCS_thread_pool_generators`), 27

M

`main()` (in module `pyfdempp.complete_BD_thread_pool_generators`), 26

`main()` (in module `pyfdempp.complete_UCS_thread_pool_generators`), 27

`main()` (in module `pyfdempp.extract_cell_thread_pool_generators`), 29

`mesh_geometry()` (`pyfdempp.pyfdempp.Model` method), 20

`Model` (class in `pyfdempp.pyfdempp`), 7
`model_dimensions()` (`pyfdempp.pyfdempp.Model` method), 20
`model_domain()` (`pyfdempp.pyfdempp.Model` method), 21
`model_vertices()` (`pyfdempp.pyfdempp.Model` method), 21
module
 `pyfdempp`, 31
 `pyfdempp.aggregate_storage`, 25
 `pyfdempp.complete_BD_thread_pool_generators`, 26
 `pyfdempp.complete_UCS_thread_pool_generators`, 27
 `pyfdempp.extract_cell_thread_pool_generators`, 28
 `pyfdempp.formatting_codes`, 29
 `pyfdempp.model_reader`, 30
 `pyfdempp.pyfdempp`, 7
`mp_read()` (in module `pyfdempp.model_reader`), 30
`multiprocess_async()` (in module `pyfdempp.model_reader`), 30
`multiprocess_lib_read()` (in module `pyfdempp.model_reader`), 30

N
`normal_read()` (in module `pyfdempp.model_reader`), 30

O
`openfдем_att_check()` (`pyfdempp.pyfdempp.Model` method), 22

P
`platen_info()` (`pyfdempp.pyfdempp.Model` method), 22
`plot_stress_strain()` (`pyfdempp.pyfdempp.Model` method), 22
`print_progress()` (in module `pyfdempp.formatting_codes`), 30
`pv_read()` (in module `pyfdempp.model_reader`), 30
`pv_read_queue()` (in module `pyfdempp.model_reader`), 30
`pyfdempp`
 module, 31
`pyfdempp.aggregate_storage`
 module, 25
`pyfdempp.complete_BD_thread_pool_generators`
 module, 26
`pyfdempp.complete_UCS_thread_pool_generators`
 module, 27
`pyfdempp.extract_cell_thread_pool_generators`
 module, 28
`pyfdempp.formatting_codes`
 module, 29
`pyfdempp.model_reader`
 module, 30
`pyfdempp.pyfdempp`
 module, 7

R
`read_file()` (`pyfdempp.aggregate_storage.aggregate_storage` method), 25
`red_text()` (in module `pyfdempp.formatting_codes`), 30
`rock_sample_dimensions()` (`pyfdempp.pyfdempp.Model` method), 23

S
`set_strain_gauge()` (in module `pyfdempp.complete_BD_thread_pool_generators`), 27
`set_strain_gauge()` (in module `pyfdempp.complete_UCS_thread_pool_generators`), 28
`simulation_type()` (`pyfdempp.pyfdempp.Model` method), 24
`slice_dataset()` (`pyfdempp.pyfdempp.Model` method), 24
`store_file()` (`pyfdempp.aggregate_storage.aggregate_storage` method), 26

T
`threshold_bound_check()` (`pyfdempp.pyfdempp.Model` method), 24
`timed()` (in module `pyfdempp.model_reader`), 31
`Timestep` (class in `pyfdempp.pyfdempp`), 25

U
`unpack_DataFrame()` (`pyfdempp.pyfdempp.Model` method), 25