PyDPF-Post cheat sheet

Ansys

Version: 0.5 (stable) - API is subject to change

/ Load results file

Instantiate a simulation object with the path to the results file:

On Windows, load the result file:

```
from ansys.dpf import post
from ansys.dpf.post import examples

sim = post.load_simulation(r'C:\Users\user\file.rst')
```

On Linux, load the result file:

```
sim = post.load_simulation(r'/home/user/file.rst')
```

/ Fetch result metadata

Display metadata from the simulation object:

```
example_path = examples.find_static_rst()
simulation = post.load_simulation(example_path)
print(simulation)
```

/ Access mesh

Access the mesh:

```
mesh = simulation.mesh
print(mesh)
```

/ Access results

Get the displacement result:

```
disp = simulation.displacement()
x_disp = simulation.displacement(components=["X"])
# Extract displacement on specific nodes from already
    extracted large data set
nodes_disp = disp.select(node_ids=[1, 10, 100])
# Extract displacement on specific nodes from results
    file
nodes_disp = simulation.displacement(node_ids=[1, 10,
    100])
```

Access the stress and strain results:

```
elem_nodal_stress = simulation.stress()
nodal_stress = simulation.stress_nodal()
elemental_stress = simulation.stress_elemental()
# Extract elemental stresses on specific elements from
     results file
elemental_stress = simulation.stress_elemental(
    element_ids=[5, 6, 7])
# Nodal strain
strain = simulation.elastic_strain_nodal()
```

/ Enable auto-completion for result quantities

Postprocess result quantities using a physics-oriented API, which enable auto-completion:

Return static simulation results:

```
simulation = post.StaticMechanicalSimulation(
    example_path)
print(simulation)
displacement = simulation.displacement()
```

Return modal simulation results:

```
simulation = post.ModalMechanicalSimulation(
    example_path)
# Print natural frequencies
print(simulation.time_freq_support)
```

Return transient simulation results and create an animation:

```
simulation = post.TransientMechanicalSimulation(
    example_path)
print(simulation)
# Query the displacement vectorial field for all times
displacement = simulation.displacement(all_sets=True)
# Create animation showing the norm of vectorial
    fields with several components
displacement.animate(deform=True, title="U")
```

Return harmonic simulation results:

```
simulation = post.HarmonicMechanicalSimulation(
    example_path)
print(simulation.time_freq_support)
displacement = simulation.displacement(set_ids=[1, 2])
subdisp = displacement.select(complex=0, set_ids=1)
subdisp.plot(title="U tot real")
subdisp = displacement.select(complex=1, set_ids=1)
subdisp.plot(title="U tot imaginary")
```

/ Plot Results

Plot total deformation (norm of the displacement vector field) results:

```
# Instantiate the solution object
example_path = examples.find_static_rst()
simulation = post.load_simulation(example_path)
displacement_norm = simulation.displacement(norm=True)
# Plot the data and save the image
displacement_norm.plot(screenshot="total_disp.png")
```

/ Create and manipulate a dataframe

Create a DPF dataframe by extracting a result from a simulation, which can be manipulated and viewed differently:

```
simulation = post.StaticMechanicalSimulation(
    example_path)
# Extract a result as a dataframe
displacement_dataframe = simulation.displacement(
    all_sets=True)
```

Return the dataframe's column labels:

```
print(displacement_dataframe.columns)
```

Display the results index:

```
print(displacement_dataframe.columns.results_index)
```

Display the values available in the index:

```
print(displacement_dataframe.columns[0].values)
```

Change the number of data rows or columns displayed:

```
displacement_dataframe.display_max_rows = 9
print(displacement_dataframe)
```

Select specific columns or rows, using index names as arguments for the `DataFrame.select` method and specified lists of values:

```
disp_X_1 = displacement_dataframe.select(
    set_ids=[1], node_ids=[4872, 9005], components=["X
        "]
)
print(disp_X_1)
```

Extract displacement data as an array contained in a dataframe:

```
print(disp_X_1.array)
# Plot a dataframe
displacement_dataframe.plot()
# Animate a transient dataframe
displacement_dataframe.animate()
```

References from PyDPF-Post documentation

- Getting started
- User guide
- Examples
- API reference