PyFluent cheat sheet

Ansys

Solver settings object interface

Version: 0.13 (stable)

/ Launch Fluent locally

```
import ansys.fluent.core as pyfluent

solver = pyfluent.launch_fluent(mode="solver",
    show_gui=True)
```

/ Import mesh in launched session

Read the available mesh file in the Fluent session:

```
mesh_filename = "example_file.msh.h5"
solver.file.read(file_type="mesh", file_name=
    mesh_filename)
```

Use specific methods to read case files and case data files:

```
# e.g., read_case(), read_case_data()

case_filename = "example_file.cas.h5"
solver.file.read_case(file_type="case", file_name=
    case_filename)
```

/ Enable heat transfer physics

Enable heat transfer by activating the energy equation:

```
solver.setup.models.energy.enabled = True
```

/ Access the object state using `pprint`

```
# >>> from pprint import pprint
# >>> pprint(solver.setup.models.energy())

{
    "enabled": True,
    "inlet_diffusion": True,
    "kinetic_energy": False,
    "pressure_work": False,
    "viscous_dissipation": False,
}
```

/ Define materials

Use `solver` settings objects to define materials:

```
solver.setup.materials.copy_database_material_by_name(
    type="fluid", name="water-liquid"
)
solver.setup.cell_zone_conditions.fluid[
    "elbow-fluid"
].material = "water-liquid"
```

/ Define boundary conditions

Use `solver` settings objects to define boundary conditions:

```
solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].vmag = {
    "option": "constant or expression",
    "constant": 0.4,
}

solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].ke_spec = "Intensity and Hydraulic Diameter"

solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].turb_intensity = 5

solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].turb_hydraulic_diam = "4 [in]"

solver.setup.boundary_conditions.velocity_inlet[
    "cold-inlet"
].t = {
    "option": "constant or expression",
    "constant": 293.15,
}
```

/ Modify cell zone conditions

Use `solver` settings objects to modify cell zone conditions.

```
solver.setup.cell_zone_conditions.fluid["elbow-fluid"]
      = {
    "laminar": True
}
```

/ Apply solution settings

Use `solver` settings objects to apply solution settings, initialize, and solve.

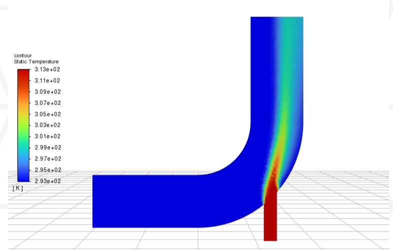
```
solver.solution.initialization.hybrid_initialize()
solver.solution.run_calculation.iterate(
    number_of_iterations=150)
```

/ Postprocessing

Postprocess data with the `results` object. For example, create and display contours on a plane:

```
solver.results.graphics.contour["contour"] = {}
solver.results.graphics.contour["contour"].print_state
    ()
solver.results.graphics.contour["contour"].field = "
    temperature"
solver.results.graphics.contour["contour"].
    surfaces_list = [
    "symmetry-xyplane"
]
```

/ Temperature contour



References from PyFluent documentation

- Getting started
- Solver settings objects
- Examples