Fluidsimfoam, a new Python framework for OpenFOAM

https://foss.heptapod.net/fluiddyn/fluidsimfoam

Pierre Augier, Pooria Danaeifar

OpenFOAM (OF) is a very popular open-source C++ CFD framework. With Fluidsimfoam, we try to design and propose a new workflow for OpenFOAM based on Python.

Principles

- Improving OpenFOAM user experience with Python
- Good quality open-source software: tested (coverage > 95%) and documented
- Description of sets of potential simulations (and not only of one “case”). A set is described in a small Python package called a “Fluidsimfoam solver”
  - Python API to describe and create parametrized input OF files
  - Potentially Jinja templates
- Python API and commands to launch, restart, reload simulations, and load/process/plot data (helped by Fluidfoam). Integrated object (`sim`) to interact with the simulation and the associated data
- All directories and files created automatically

Particularly suitable for

- automation of simulation launching (parametric studies, optimization, …)
- programmatic generation of complex and parametrized input files (for example `blockMeshDict`) and initial conditions (computed in Python)
- programmatic control of simulations at runtime

Super easy to create a Fluidsimfoam solver from a case

```
fluidsimfoam-initiate-solver dam -c $FOAM_TUTORIALS/multiphase
/interFoam/laminar/damBreak/damBreak
```

Then, few modifications to parametrize the solver. Helpers for fields (initial conditions), constant files, `controlDict` , `fvOptions` , `fvSchemes` , `decomposeParDict` , `BlockMeshDict` , …

```
_helper_transport_properties = ConstantFileHelper(
    "transportProperties",
    {
        "phases": ["water", "air"],
        "water": {
            "transportModel": "Newtonian",
            "nu": 1e-06,
            "rho": 1000,
        },
        "air": ...
    },
)
```

Programmatic generation of `BlockMeshDict`

```
bmd = BlockMeshDict()
bmd.set_scale(params.block_mesh_dict.scale)

for x_y_z_name in (
    (0, 0, 0, "left_bot"),
    (x_dam, 0, 0, "leftdam_bot"),
    (x1_dam, 0, 0, "rightdam_bot"),
    (lx, 0, 0, "right_bot"),
    ...
):
    bmd.add_vertex(*x_y_z_name)

bmd.replicate_vertices_further_z(lz)

b_bot_left = bmd.add_hexblock_from_2d(
    ["left_bot", "leftdam_bot", "leftdam_topdam",
"left_topdam"],
    [nx_left, ny_bot, nz],
    "left_bot",
)
```

Create a simulation directory and launch the simulation

```
from fluidsimfoam_dam import Simul

# creation of the `params` object
params = Simul.create_default_params()

# modification of parameters
params.output.sub_directory = "poster_fluidsimfoam/dam"
params.control_dict.end_time = 1.0

params.parallel.method = "simple"
params.parallel.nsubdoms = 4

params.constant.transport.water.nu = 2.e-6
...

# creation of the simulation directory
sim = Simul(params)

# run the simulation (i.e. all necessary OpenFOAM commands)
sim.make.exec("run")
# or for programmatic control of the simulation
sim.make.exec_async("run")
```

Reload the simulation for runtime control / data processing / plots

One can recreate the `sim` object with the command `fluidsimfoam-ipy-load` or with:

```
from fluidsimfoam import load
sim = load("/path/to/simulation/directory")
```

Read and modify files

Change a parameter affecting just one file:

```
sim.params.control_dict.end_time = 2
sim.input_files.control_dict.generate_file()
```

or (even more powerfull):

```
ctrl_dict = sim.input_files.control_dict.read()
ctrl_dict.set_child("endTime", 2)
ctrl_dict.overwrite()
```

Read output fields:

```
field = sim.output.fields.read_field("U", time_approx="last")
vx, vy, vz = field.get_components()
```

Next steps

Installation: `pip install fluidsimfoam`
Documentation: https://fluidsimfoam.readthedocs.io
Repository: https://foss.heptapod.net/fluiddyn/fluidsimfoam

Soon

- Restart utilities
- More figures and movies
- Support for `probes` , `boundaryCloud` , `singleGraph` , `surface` functions