OpenFOAM© cheat sheet

Jens Höpken

Sourceflux, www.sourceflux.de

sourceflux
Open Source Solutions

Introduction

OpenFOAM is a library written in C++, that can handle all sorts of numerical problems. Foremost it is used to solve fluiddynamical problems. In addition to the library, it provides a lot of tools (pre- and postprocessing, solvers, mesh generation), that are build upon the library.
A finite volume method is employed to discretise the computational domain and all generated meshes are handled as being arbitrarily unstructured.
This cheat sheet is by no means all-embracing and solely touches the surface of OpenFOAM.

Terminology

- **case** Project folder on the disk, containing all relevant information
- **dictionary** Any file in the case, that is read by OpenFOAM to define parameters
- **timestep** Any folder in the case, that has a numerical name (e.g. `0.3` or `122`)

Rules of Conduct

- Always read and analyse the warnings and errors that you get from OpenFOAM
- Always execute OpenFOAM commands from the case folder
- OpenFOAM does **only** what **you** tell it to do. Nothing more, nothing less.

Case Struture

```
<case>/
├─0/ ..................................... Boundary conditions
│ ├─U ..................................... Velocity boundary condition
│ └─p ..................................... Pressure boundary condition
├─constant/ ............................... Mesh, physical properties
│ └─polyMesh/ ............................. Mesh
└─system/ ................................. Solution controlling
  ├─fvSchemes ............................. Discretisation schemes
  ├─fvSolution ............................ Solver settings
  ├─controlDict ........................... Run control parameters
  └─(0.*|[1-9]*)/ ......................... Timestep folders
```

Mesh Generation

- **blockMesh** Blockstuctured mesh generation, that allows cell gradings and curved edges. Complex geometries are not easy to mesh. Definition in `constant/polyMesh/blockMeshDict` [1, 2].
- **snappyHexMesh** Generates a mesh based on one or multiple STL geomteries and a blockMesh background mesh. The result is *always* 3D and is hexahedral dominant, though other cell shapes exist. Definition in `system/snappyHexMeshDict` [1, 2].

Boundary Conditions

The boundary conditions are defined in the `0/` directory and each field has it's own definition. All files in that folder contain two sections:

1. The **internalField** specifies the initial values for all cells. This can either be uniform for all cells or defined explicitely for each cell.
2. Values on the boundaries need to be defined in **boundaryField**, by selecting an appropriate boundary condition.

- **fixedValue** represents the *Dirichlet* boundary condition and defines the solution of a property ϕ on the boundary:

$$\phi = n$$

- **zeroGradient** represents the *Neumann* boundary condition and defines the derivative of the solution of a property ϕ on the boundary:

$$\frac{\partial \phi}{\partial x} = 0$$

- **symmetryPlane** is fairly self-explanatory
- **empty** must be used to create 2D meshes. All boundaries with normal vectors that are not inside the 2D plane have to be defined as empty.

In addition to the boundary conditions mentioned above, there are much more implemented in OpenFOAM, just have a look into `$FOAM_SRC/finiteVolume/fields/fvPatchFields`.

Preprocessing

- **checkMesh** should always be used after the mesh generation to check the integrity of the mesh. Study and analyse the output!
- **setFields** can be used to initialise the fields with non uniform values (e.g. the volume fraction field).
- **mapFields** maps the volume fields from one mesh to another.

Postprocessing

- **yPlusRAS** calculates y^+ for all wall patches
- **foamCalc** can calculate any field operation, such as `div(U)`.

Aliases

run	jump to `$FOAM_RUN`	app	jump to `$FOAM_APPLICATIONS`
sol	jump to `$FOAM_SOLVERS`	util	jump to `$FOAM_UTILITIES`
foam	jump to `$WM_PROJECT_DIR`	foamfv	jump to `$FOAM_SRC/finiteVolume`
foamsrc	jump to `$FOAM_SRC/$WM_PROJECT`	tut	jump to `$FOAM_TUTORIALS`

Solvers

- **simpleFoam** is a steady state solver for an incompressible flow. Turbulence modelling can either be laminar, RAS or LES.
- **pimpleFoam** is a transient solver for an incompressible flow. Turbulence modelling can either be laminar, RAS or LES.
- **interFoam** solves the Navier-Stokes equations for two incompressible, inmiscible phases. Turbulence modelling can either be laminar, RAS or LES.
- **interDyMFoam** is similar to interFoam, but employs a dynamic mesh.

Running in Parallel

Before starting a parallel run, the computational domain needs to be decomposed. The parameters are defined in `system/decomposeParDict`. To start the decomposition, execute `decomposePar` before starting the parallel run via:

```
> mpirun -np <N> <SOLVER> -parallel
```

Where `<N>` denotes the number of subdomains and `<SOLVER>` represents the solver to run in parallel.
After the simulation is finished, you have to glue the subdomains together. For this task, `reconstructPar` is the tool of choice.

Using functionObjects

In OpenFOAM, functionObjects are code snippets that can be loaded into any solver during run-time. This enables each solver to be coupled with some additional functionality, that is independent from the flow model. All functionObjects have to be included in `system/controlDict` like the following:

```
functions
{
    <name>
    {
        type                 <type>;
        functionObjectLibs   ("<lib>");
        outputControl        timeStep;
        outputInterval       1;
        // Specific parameters for the functionObject
    }
}
```

Integrating forces

To integrate the forces over a list of patches, the functionObject **forces** can be used and must be included as shown in the previous section. The specific parameters are:

```
patches     ( "<patchNames>" ); // This may be done regexp
pName       p;                  // Name of the pressure field
UName       U;                  // Name of the velocity field
rhoName     rhoInf;             // Indicates incompressible
log         false;              // Only write to file
rhoInf      1;                  // Redundant for incompressible
CofR        (0.72 0 0);         // Ref. point for moment calc.
```

The output is written to `postProcessing/forces/<startTime>/forces.dat` and contains the pressure and viscous forces, as well as the pressure and viscous moments. The data is grouped by round brackets, that need to be removed to plot them in `gnuplot`. There are multiple ways to do that, using `sed` is just one of them:

```
?> sed -e 's/[(,)]/ /g' \
    postProcessing/forces/<startTime>/forces.dat > tmp
```

References

[1] URL: http://www.openfoam.com/docs/user.

[2] T. Marić, J. Höpken, and K. Mooney. *The OpenFOAM Technology Primer*. First Edition. Sourceflux, 2014. URL: www.sourceflux.de/book.

Disclaimer