Tutorial 3: OpenFOAM® basic usage and structure

Objective:

The objective of this tutorial is to get familiar with OpenFOAM® . You will test your installation by preparing a simulation case, running the first simulation, post-processing the results and visualizations.

The requirement of this exercise is that you are comfortable with Linux command operations. It is also assumed that you have installed OpenFOAM® and made a copy of the tutorials into your own directory, i.e., the following two commands have been executed:

```
$ mkdir -p $FOAM_RUN
$ cp -r $FOAM_TUTORIALS $FOAM_RUN
```

1 Exercises:

1.1 Initial Browsing of the OpenFOAM® structure

The whole package of OpenFOAM® is organized into a very well organized structure corresponding to a hierarchy of directories. We have seen the individual components of OpenFOAM® during the lecture. These components include solvers, libraries, utilities tools, and the underlying source codes. It also comes with tutorial cases corresponding to almost all the existing solvers that shipped with OpenFOAM® . It also contains relatively extensive documentation.

1.1.1 Aliases

To facilitate an easy get-around, OpenFOAM® has pre-defined some "aliases", which are shortcuts or "another names" for many of the common tasks. To have a look at what are the pre-defined aliases (including both from OpenFOAM® and the Linux system), type

```
$ alias
```

1.1.2 Explanations of aliases

You will see all the available aliases and we will introduce some of them for now. Type

```
$ foam
```

will bring you to the installation location of the OpenFOAM® package. For example, the default (in most cases) is under "/opt/OpenFOAM-v2306". This installation is systemwise, i.e., multiple users in this Linux system can use this OpenFOAM® installation. They don't

have to install their own copy individually. In addition to save space, it is also easy to upgrade for everybody. However, each user needs to work in his/her home directory. In the home directory, the user can modify the solvers, utility tools, and even the library as long as they are given different names from the systemwise installation. To get back to the home directory, type

```
$ run
```

This alias will bring you to `/home/user_name/OpenFOAM/user_name-v2306/run`, where your simulation cases will usually reside.

You can use the "alias" command to see what exactly these aliases are by typing for example:

```
$ alias run
```

This will display `alias run='cd $FOAM_RUN ...'`, which means the "run" command is defined to execute the "`cd $FOAM_RUN ...`" command. Here `$FOAM_RUN` is a OpenFOAM® pre-defined environment variable. Type

```
$ echo $FOAM_RUN
```

to see how it is defined.

Table 1: Major aliases defined in OpenFOAM®

Alias	Definition	Function
foam	`cd $WM_PROJECT_DIR`	Goes to the OpenFOAM® installation directory
app	`cd $FOAM_APP`	Goes to the OpenFOAM® applications directory where the source code for the generic solvers and utilities reside
lib	`cd $FOAM_LIBBIN`	Goes to the directory for the binary files of the libraries and executables
src	`cd $FOAM_SRC`	Goes to the source code directory
sol	`cd $FOAM_SOLVERS`	Goes to the solver directory under `applications`
util	`cd $FOAM_UTILITIES`	Goes to the utilities directory under `applications`
tut	`cd $FOAM_TUTORIALS`	Goes to the tutorials directory

Major aliases defined in OpenFOAM® are listed in Table 1. Now you can experiment with these aliases and move around in the OpenFOAM® system quickly. Try these aliases and use

the commands you learned from the tutorial and homework on Linux systems, such as `ls` and `cd`, to have an idea of the OpenFOAM® structure.

These aliases (as well as the environment variables) are defined in a resource file which is read every time you login. If the BASH shell is used, the corresponding definition file can be found in `$WM_PROJECT_DIR/etc`. Type

```
$ foam
$ cd etc
$ ls -l
$ more bashrc   (note: use space bar to scroll down; type ‘‘q’’ to quit)
```

In this file `bashrc`, the aliases, environment variables and compiler options are set. This `bashrc` file can be included in your login `.bashrc` file in your home directory. If you followed the official OpenFOAM® installation instruction, you can verify this by:

```
$ cd              (Note: this commands takes you back to your home directory)
$ cat .bashrc     (Note: don’t forget the ‘‘.’’)
```

You should see a line in the file as `source /path/to/OpenFOAM/etc/bashrc`. So each time you login or open a new command window, this line will be executed and all the setup (environmental variables, aliases, etc) will be updated.

In our previous tutorial for OpenFOAM installation, we used alias definition for the “source” command above. In doing so, you can have multiple installations of OpenFOAM® (different versions, official release or Extend-project, etc.). The selection can be made by defining different alias and execute the one you desire. I have also demonstrated during lecture on how to use alias to define multiple installation of OpenFOAM® with different versions.

1.2 Run your first simulation

1.2.1 Explore your simulation case

In OpenFOAM® , most time you don’t have to start from scratch. You can copy from an existing tutorial case and do modifications according to your needs (e.g., change domain shape, size, initial condition, and boundary condition). This of course will save your time and reduce the chance of making errors.

In this exercise, we will run a simple incompressible flow case.

```
$ run
$ cd tutorials/incompressible/icoFoam/cavity/cavity  (Note: use "Tab" to auto-complete).
$ ls
```

This is a copy of the `cavity` flow case from OpenFOAM® tutorial. Inside this directory,

there are three sub-directories: `0`, `constant`, and `system`.

1.2.2 The "0" directory

The "0" directory contains the initial and boundary conditions for the variables.

```
$ cd 0
$ ls
```

You will see two files called `p` and `U`, corresponding to pressure and velocity. You can examine the contents of each file by

```
$ more p
$ more U
```

The structure of the variable files (called field) is similar. For example, the pressure file contains a banner message on top and is followed by the following.

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
     movingWall
     {
         type            zeroGradient;
     }

     fixedWalls
     {
         type            zeroGradient;
     }

     frontAndBack
     {
         type            empty;
     }
 }
```

There are three sections:

- `dimensions`: It specifies the dimensions of the field, here kinematic pressure, i.e.

m^2/s^2. Now it is a good time to talk about the dimensions. For each variable (a.k.a, field), there is an associated set of dimensions and units. In OpenFOAM® , there are 7 pre-defined dimensions which are considered important for fluid dynamics. They are mass (M), length (L), time (T), temperature ($Temp$), quantity (Q), current (A), and luminous intensity (LI). The specification of dimensions is through a bracket

```
[M   L  T   Temp   Q   A  LI]
```

By default, OpenFOAM® will check the dimensional consistency of field operations. For example, if you add two variables together, these two variables should have the same dimensions. For instance, if you add a length variable to a time variable, it does not make any sense. In this case, OpenFOAM® will complain and abort the execution. This checking functionality can be switched off. However, it is strongly recommended not to do so.

OpenFOAM® will only check the dimensions, not the unit. As for the specific units, such as using meter or foot for the length dimension, it is the user's choice. The user should make sure that the units are consistent.

So how to specify the dimension of a field variable? It is in this "`dimensions`" section of the file. For example,

```
dimensions      [0 2 -2 0 0 0 0];
```

For this variable, it is the kinematic pressure (pressure/density) and the dimension is $Length^2/Time^2$.

- `internalField`: This section defines the initial conditions for this field variable. The internal field for the initial condition can be uniform or nonuniform. If it is uniform, it is described by a single value; if nonuniform, all the values of the field at each cell center must be specified as a value list.
- `boundaryField`: This section defines the boundary conditions for the field variable. The boundary is defined as a `patch` in a mesh. This boundary field data specifies boundary conditions and data for all the boundary patches. There are various boundary conditions implemented in OpenFOAM® . From the point of view of differential equations, there are three basic boundary conditions, namely Dirichlet type, Newman type, and Robin type. At certain boundary, they specify the value of the variable, the gradient of the variable, or a combination of value and gradient, respectively.

When you are done inspecting the `0` directory, go back to the case directory by

```
$ cd ..
```

```
$ ls
```

1.2.3 The "constant" directory

The `constant` directory contains the model specifications and the computational mesh.

```
$ cd   constant
$ ls
```

In this case, we have a file named `transportProperties`.

```
$ more    transportProperties
```

Have a look at the content of this file. It is probably the simplest model specification in OpenFOAM® since we only simulate incompressible laminar flows. The only fluid property matters here is the kinematic viscosity ν and its dimension is $Length^2/Time$. The value for this kinematic viscosity is specified as a constant. For more complicated models, you will see more files in this directory. For example when turbulence is considered, the specific model should be specified (such as RANS or LES; what kind of model? k-ϵ model or Spalart-Allmaras model?)

Now we can generate the mesh. There are many ways to generate meshes. OpenFOAM® comes with a simple and yet powerful mesh generation tool `blockMesh`. To use this tool, the user needs to specify inputs in the `blockMeshDict` file inside the `system` directory (more details later). The OpenFOAM® user manual has a thorough description of how to use `blockMesh` tool. For the time being, you can have a look at the file by

```
$ cd .. (note: get back to the case directory from ‘‘constant’’)
$ more system/blockMeshDict
```

After this, you can generate the mesh by typing:

```
$ blockMesh
```

It will create the mesh and store the information within the directory `constant/polyMesh`. The `ployMesh` directory contains the mesh. OpenFOAM® solve the governing equations using finite volume method which discretizes the domain into small pieces (cells).

```
$ cd constant/polyMesh
$ ls
```

The "`boundary`" file is used to specify the physical meaning of the boundaries, such as walls.

```
$ more boundary
```

Other files in this directory defines points, faces, cells and their topological relations.

Now let's go back the case directory by

```
$ cd ../..
$ ls
```

1.2.4 The "system" directory

The `system` directory contains the specifications for things like numerical discretization, linear solvers, run control, and other solution parameters.

```
$ cd system
$ ls -l
```

You will see at least four files, namely `controlDict`, `fvSchemes`, `fvSolution`, and `blockMeshDict`.

The `controlDict` file contains run control options such as starting and end time, time step size, whether dynamically adjust the time step size, output frequency, etc.

```
$ more controlDict
```

The `fvSchemes` file contains specifications for the discretization schemes, such as time derivative, gradient, divergence, interpolation, etc.

```
$ more fvSchemes
```

The `fvSolution` file contains the specifications for the linear solvers for each of the solution variable, including which linear solver to use and the convergence criteria. It also contains the specifications for the fluid dynamics solver. For this case, since we will use PISO algorithm, we need to specify things like how many correction steps.

```
$ more fvSolution
```

Now let's go back to the case directory by

```
$ cd ..
$ ls -l
```

1.2.5 Run your simulation case

Typical process of running a simulation case include pre-processing, execution, and post-processing.

- Pre-processing: In this step, you need to prepare the mesh, define boundary and initial conditions, and specify various control parameters. Since everything is pre-set already, we will demonstrate how to generate mesh by using `blockMesh`. In the case directory, type

```
$ blockMesh
```

This tool will read the `blockMeshDict` file in the `system/polyMesh` directory and generate the mesh accordingly. You can verify the output of the mesh generation by

```
$ ls system/polyMesh
```

You should see the files faces, neighbour, owner, and points. It is always a good practice to check your mesh to see if there is anything wrong by the tool `checkMesh`:

```
$ checkMesh
```

This tool will check the consistency and correctness of the mesh. It will also display the statistics of the mesh, such as number of points, faces, and cells. In addition, it will tell you the quality of the mesh, such as maximum skewness and aspect ratio which affect the accuracy of your simulation. If your simulation diverges, you should check the quality of your mesh first.

- Execution: Now we have everything set and it is time to run the simulation.

  ```
  $ icoFoam
  ```

  This will invoke the solver `icoFoam`, which stands for `incompressible fluid OpenFOAM solver`. The solver will display some information for each time step such as Courant number, iterative solver diagnostics, and execution time. This information is important for monitoring the simulation and identifying any possible problems. Our case is very small so the run should finish in very short time.

- Post-processing: Standard installation of OpenFOAM® comes with the open source (a.k.a. FREE) visualization software `ParaView`. It has a native reader to directly load the OpenFOAM® simulation results. To use `ParaView`, simply type the name of a generic script `paraFoam`,

  ```
  $ paraFoam
  ```

  Try to experiment with `ParaView` and plot the contours, vectors, etc. `ParaView` is very easy to use.

  To learn how to use `ParaView`, a good starting point is its own tutorials:

  `https://www.paraview.org/tutorials/`

  Among all its tutorials, I suggest you start with the "ParaView Self-directed Tutorial":

  `https://docs.paraview.org/en/latest/Tutorials/SelfDirectedTutorial/index.html`

  To use `paraFoam`, you need to compile the corresponding OpenFOAM® loader module

for `ParaView`. If you skip this compilation step, another way to use `ParaView` is to load the results directly by creating an empty file, say "case.foam" and open it in `ParaView`.

```
$ touch case.foam
$ paraview case.foam&
```

As an alternative, if you are familiar with other visualization tools, you can convert the OpenFOAM® results into a format that is readable by other tools. For example, you can use Tecplot.

References:

Here are two references I used to prepare for this homework exercises. Get a copy of them and they should be very helpful to you too.

1. V. Gedris, (2003). An introduction to the Linux command shell for beginners. Document version 1.2, 2003-06-25

2. D.A. Boger, (2011). Getting started with OpenFOAM, 6th OpenFOAM workshop, June 13-16, 2011, PennState University.