Introduction to fluid mechanics simulation using the OpenFOAM® technology

« Simulation in porous media from pore to large scale »

Part I: General overview and first simulations

Contact: cyprien.soulaine@stanford.edu
http://web.stanford.edu/~csoulain/

Stanford
October 27, 2014
Version 5.0
Objectives

• Have an overview of the OpenFOAM® capabilities

• Be able to find help and documentation

• Know how to start and post-treat a simulation from existing tutorials

• Start your own simulation by modifying existing tutorials

• Understand what is behind solvers to identify the most suitable for the specific problem

• Program your own solver by modifying an existing solver

• Join the OpenFOAM® adventure…
General introduction to the OpenFOAM® technology

- What is OpenFOAM®?
- Where one can find help and documentation?

First simulations with OpenFOAM®

- General structure of an OpenFOAM® case
- #1 – Heat diffusion
- #2 – Cavity
- #3 – Poiseuille flow
- #4 – Drainage experiment in a capillary tube

How to mesh complex geometries with OpenFOAM®?

- snappyHexMesh overview
- #5 – Mesh a pore-space with snappyHexMesh
- #6 – Scalar transport in porous media at the pore-scale

Programming equations with OpenFOAM®

- General structure of an application
- Basics of OpenFOAM programming

Heat transfer in porous media with OpenFOAM®

- #7 – Create a « Darcy » solver
- #8 – Temperature in porous media
- #9 – Two-equations model for heat transfer in porous media

How to solve Navier-Stokes equation with OpenFOAM®?
General introduction to the OpenFOAM® technology

- What is OpenFOAM®?
- Where one can find help and documentation?

First simulations with OpenFOAM®

- General structure of an OpenFOAM® case
- #1 – Heat diffusion
- #2 – Cavity
- #3 – Poiseuille flow
- #4 – Drainage experiment in a capillary tube

How to mesh complex geometries with OpenFOAM®?

- snappyHexMesh overview
- #5 – Mesh a pore-space with snappyHexMesh
- #6 – Scalar transport in porous media at the pore-scale

Programming equations with OpenFOAM®

- General structure of an application
- Basics of OpenFOAM programming

Heat transfer in porous media with OpenFOAM®

- #7 – Create a « Darcy » solver
- #8 – Temperature in porous media
- #9 – Two-equations model for heat transfer in porous media

How to solve Navier-Stokes equation with OpenFOAM®?
What is OpenFOAM®?

Open\(\nabla\)FOAM®

= **Open Field Operation and Manipulation**

- Solve the Partial Differential Equations using the finite volumes method
- Multiphysic simulation platform mainly devoted to fluid flow
- Manage 3D geometries by default
- Open-source software developed in C++ (object-oriented programming)
- Can be freely download at [www.openfoam.com](http://www.openfoam.com)
- Designed as a toolbox easily customisable
- Parallel computation implemented at lowest level
- Cross-platform installation (Linux preferred)

1989 : First development at Imperial College London
1996 : First release of FOAM
2004 : OpenFOAM release under GPL licence by OpenCFD Ltd.
2014 : version 2.3.0 ; acquisition of OpenCFD by ESI Group
The OpenFOAM® toolbox

OpenFOAM® = more than 200 programs (and not only 1 executable)

Pre-processing:
- Meshing (blockMesh, snappyHexMesh…)
- Mesh conversion (Ansys, Salomé, ideas, CFX, Star-CD, Gambit, Gmsh…)

Solvers:
- incompressible / compressible flow
- multiphase flow
- combustion, electro-magnetism, solid mechanics
- heat transfer
- several turbulence approach (DNS, RANS, LES)
- etc…

post-processing:
- Distributed with ParaView (and the famous paraFoam)
- Exportation toward other post-treatment softwares (Fluent, Fieldview, EnSight, Tecplot…)
- « sample » utility for 1D or 2D sampling (export to gnuplot, Grace/xmgr et jPlot)
Simulation examples

Horgue et al., 2011

Liquid Injection

Horgue et al., 2014

Time: 400 s

Time: 1200 s

Time: 2000 s

Romain Guibert, 2013
Porous media modeling with OpenFOAM

Multiphase flow simulation in porous media at Darcy-scale


Immiscible two-phase at pore-scale


Digital Rock Physics with OpenFOAM


Superfluid helium flow in porous media at pore-scale

How to program equations?

\[ \frac{\partial \rho U}{\partial t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p \]

```cpp
solve
(
    fvm::ddt(rho,U)
    + fvm::div(phi,U)
    - fvm::laplacian(mu,U)
    ==
    - fvc::grad(p)
);
```

- The considered field \((U)\) may be scalar, vector or tensor,
- Operators discretization does not need to be specified at the stage of the solver programming,
- The syntax is very closed to the mathematical formalism.
Where can one find help and documentation?

- 2 official guides provided by OpenCFD (« user guide » and « programmer guide ») (Most of the time, this documentation is not enough...)
- A tutorial per solver. Most of the time, it has a value of test-cases. (There is no public validation..)
- Direct access to source-code (however, there is very few comments in the code !!)
- Paying for support from OpenCFD Ltd, Wikki Ltd,....

An active community!

- A discussion forum (www.cfd-online.com/Forums/openfoam/)
- A community-driven wiki (openfoamwiki.net)
- An annual Workshop (9th edition in 2014) (www.openfoamworkshop.org)
- Social network related to OpenFOAM (www.extend-project.de)
- www.cocoons-project.org : A peer-reviewed documentation project
- A lot of tutorials, reports, presentations made by the community (search on Google)
Why should I use OpenFOAM?

- Completely free (No limitations due to licenses),
- Direct access to source code (not a black-box),
- An additional tool for code-to-code benchmarks,
- Regular updates,
- A lot of out-of-the-box solvers and their tutorials,
- Ease to program partial differential equations,
- A reactive and important community (forum, conference, summer schools…),
- ...

- Need some time to learn,
- Lack of documentation..
- There is no official GUI,
- Unix command lines and C++ programing,
- ...

...
## Some Unix commands

### Navigation
- **pwd**: Tells you the name of the working directory.
- **ls**: List the files in the working directory.
- **cd**: Change your working directory.

### Visualization
- **cat**: Outputs the contents of a specific file.

### Manipulation of files
- **cp**: To copy a file. Use the `-r` option to copy a directory.
- **mkdir**: Create a directory.
- **rm**: Delete a file. Use the `-r` option to remove a directory.
- **mv**: Move or rename a file/folder.

### I/O redirection
- **>**: To redirect the output of an executable toward a file.
- **|**: A pipeline to connect multiple commands together.
- **grep**: A filter to output every line that contains a specified pattern of characters.
How to use the terminal in Mitchell/A65

1. Turn on or reboot the computer and hold down the Option key until the dual-boot appears on screen.

2. Boot on Mac OS and login with your SUNetID.

3. Start Xquartz, go to preferences and tick the fullscreen option.

4. Open a terminal (Ctrl + N). Press Cmd+Option+A to activate/deactivate the fullscreen mode.
   The graphic interface (X11) only works in fullscreen mode.
How to use OpenFOAM at Stanford

OpenFOAM 2.2.2 is installed in Stanford CEES-RCF cluster
You can login to the cluster with your SUNetID
For this training, we will use the tool servers in graphic mode

```
$ ssh sunetid@cees-tool-3.stanford.edu -X
```
Or

```
$ ssh sunetid@cees-tool-4.stanford.edu -X
```

Once you are login, create your OpenFOAM home directory (this stage only needs to be done once)

```
$ cd ~
$ mkdir -p $FOAM_RUN
```

Go to your home directory

```
$ run
```
General structure of an OpenFOAM case

Simulation setup (choice of the linear solvers, of the discretization schemes, of the time steps, the output files...)

Everything regarding constant values (transport properties, thermodynamic properties, turbulence model, etc...)

All the information related to the grid

One folder per time step. Each folder includes as many files as computed fields (T, U, p, Yi, k, Omega...) Initial conditions are specified in the « 0 » directory
#1 - Heat diffusion (1/4)

- Example from tutorials delivered with OpenFOAM
- Geometry and grid generated with Ansys
  - Mesh conversion using the utility `ansysToFoam`
- Solution of the heat transfer equation
  \[ \frac{\partial T}{\partial t} = \nabla \cdot (D_T \nabla T) \]
  - Solver: `laplacianFoam`
$ run
$ cp –r $FOAM_TUTORIALS/basic/laplacianFoam/flange Exo1
$ cd Exo1
$ ls
$ gedit 0/T

Initial and boundary conditions for the temperature field $T$

Mesh: faces, points… defined after the execution of ansysToFoam

Value of the diffusion coefficient (m$^2$/s)

Simulation parameters (time steps, output time…)

Discretization of the different operators (div, laplacian, ddt, grad…)

Set up of the linear solvers

Scripts to automatically start and clean the tutorial

Grid generated with Ansys
#1 - Heat diffusion (3a/4)

**Definition of initial and boundary conditions**

- Dimensions (units) of the field $T$ 
  \[ [\text{kg} \ \text{m} \ \text{s} \ \text{K} \ \text{kgmol} \ \text{A} \ \text{cd}] \]

- Uniform initial temperature ($T=273K$) in the solid bulk

**Boundary conditions for $t=0s$**

- Zero flux

- Fixed value ($T=273K$)

- Fixed value ($T=573K$)
$ gedit constant/transportProperties

The dimensions of the diffusivity $DT$ are m$^2$/s
$ gedit system/controlDict

```c++
/*--------------------------------*- C++ -*----------------------------------*
 | Field Operation And Manipulation  |
 | OpenFOAM: The Open Source CFD Toolbox |
 | Version: 2.0.1 | Web: www.OpenFOAM.com |

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object controlDict;
}

// ************************************************************************* //

application laplacianFoam;
startFrom latestTime;
startTime 0;
stopAt endTime;
endTime 3;
deltaT 0.005;
writeControl runTime;
writeInterval 0.1;
purgeWrite 0;
writeFormat ascii;
writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable true;

// ************************************************************************* //
```
#1 – Heat diffusion (4/4)

1. Convert the mesh from Ansys to OpenFOAM
   
   ```
   $ ansysToFoam flange.ans -scale 0.001
   ```

2. Visualize the mesh with ParaView
   
   ```
   $ paraFoam
   ```

3. Start the simulation
   
   ```
   $ laplacianFoam
   ```

4. View the results with ParaView
   
   ```
   $ paraFoam
   ```

   Creation of new files in the directory `constant/polyMesh`

   Creation of a new folder for each time step

   1 – Choose the field you want to load for viewing

   2 - « apply »

   3- Choose « surface »

   4 – Choose the field to display (T)

   5 – Press play to read all the time steps

**Exo1bis:**

`DT = 1e^{-7} m^2/s` Until steady-state
#2 - Cavity (1/6)

![Diagram of a cavity with boundary conditions]

- **Design and meshing of the geometry with the utility `blockMesh`**
- **Solution of the laminar incompressible Navier-Stokes equations with the `icoFoam` solver**
  \[ \nabla \cdot \mathbf{U} = 0 \]
  \[ \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{UU}) = \nabla \cdot (\nu \nabla \mathbf{U}) - \nabla p \]
- **Post-processing with ParaView**

```
$ run
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity Exo2
cd Exo2
ls
```
#2 - Cavity (2/6)

**blockMesh** = pre-processing tool to design and mesh simple geometries

Geometry and grid are defined in the file *blockMeshDict*

```
$ gedit constant/polyMesh/blockMeshDict
```

The geometry is always defined in 3D since OpenFOAM only considers 3D geometry

The numbering is of great importance!!

We specify that the simulation will be 2D

Boundaries may be of different types:
- patch *(generic type)*
- wall *(for solid wall condition, useful for turbulence)*
- cyclic *(for cyclic simulations)*
- symmetryPlane *(for symmetry plane)*
- empty *(to specify that the simulation will be 2D or 1D)*
- wedge *(for axi-symmetric simulations)*
- processor *(for parallel computation, automatically defined during the decomposition domain process)*
#2 - Cavity (3/6)

- Generate the grid mesh: \$ blockMesh
- Check the mesh quality: \$ checkMesh
- View the mesh: \$ paraFoam

1. Choose the elements you want to visualize
2. To display the patch names on the geometry
3. « Apply »
4. « Apply »
5. « Wireframe »
6. « Edit > view Setting > use Parallel Projection »
The flow is laminar, so it fulfills the *icoFoam* assumptions.
$ gedit 0/U

```
FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    object U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions [1 1 1 0 0 0 0];
internalField uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type fixedValue;
        value uniform (1 0 0);
    }
    fixedWalls
    {
        type fixedValue;
        value uniform (0 0 0);
    }
    frontAndBack
    {
        type empty;
    }
}
```

OpenFOAM: The Open Source CFD Toolbox
Version: 1.7.1
Web: www.OpenFOAM.com
Be careful with the units! In OpenFOAM incompressible solvers, the solved pressure is

\[ p = \frac{p}{\rho} \]
#2 - Cavity (4d/6)

$ gedit system/controlDict

Management of the time discretisation (start, end, time steps...)

Management of the output files
#2 - Cavity (5/6)

Start the simulation:

\$ icoFoam

Post-processing with ParaView:

\$ paraFoam
We use the *CellCenter* filter to specify ParaView we need the value at the cells center:

```plaintext
filters>alphabetical>CellCenter>Apply
```

We then apply the Glyph filter to plot the velocity vector:

```plaintext
filters>alphabetical>Glyph>Properties
```
To view the velocity vectors

Display > color by > U
Objectives:

- Simulate a Poiseuille flow through a 2D pipe with symmetry plane condition
- Steady-state solution of laminar incompressible Navier-Stokes equations with the `simpleFoam` solver

\[ \nabla \cdot \mathbf{U} = 0 \]

\[ \nabla \cdot (\mathbf{U} \mathbf{U}) = \nabla \cdot (\nu \nabla \mathbf{U}) - \nabla p \]

```
$ run
$ cp  -r  $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily  Exo3
$ cd  Exo3
$ cp  ../Exo2/constant/polyMesh/blockMeshDict  constant/polyMesh/.
```
#3 - Poiseuille flow (2/4)

```bash
$ gedit constant/polyMesh/blockMeshDict
```

```
convertToMeters 0.1;
vertices
{
(0 0 0)
(20 0 0)
(20 1 0)
(0 1 0)
(0 0 0.1)
(20 0 0.1)
(20 1 0.1)
(0 1 0.1)
);
blocks
{
  hex (0 1 2 3 4 5 6 7) (100 20 1) 
    simpleGrading (1 5 1)
};
edges
{
};
```

```
boundary
{
  symmetryPlane
  {
    type symmetryPlane;
    faces
    {
      (3 7 6 2)
    };
  }
  inlet
  {
    type patch;
    faces
    {
      (0 4 7 3)
    };
  }
  outlet
  {
    type patch;
    faces
    {
      (2 6 5 1)
    };
  }
  bottomWall
  {
    type wall;
    faces
    {
      (1 5 4 0)
    };
  }
  frontAndBack
  {
    type empty;
    faces
    {
      (0 3 2 1)
      (4 5 6 7)
    };
  }
};
```
#3 - Poiseuille flow (3a/4)

```plaintext
$ gedit 0/p
```

```plaintext
blockMeshDict

FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object p;
}

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

internalField uniform 0;

boundaryField
{
    inlet
    {
        type zeroGradient;
    }
    outlet
    {
        type fixedValue;
        value uniform 0;
    }
    bottomWall
    {
        type zeroGradient;
    }
    symmetryPlane
    {
        type symmetryPlane;
    }
    frontAndBack
    {
        type empty;
    }
}
```

```plaintext
$ gedit 0/U
```

```plaintext
blockMeshDict

FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    object U;
}

dimensions [0 1 -1 0 0 0];

internalField uniform (0 0 0);

boundaryField
{
    inlet
    {
        type FixedValue;
        value uniform (1 0 0);
    }
    outlet
    {
        type zeroGradient;
    }
    bottomWall
    {
        type fixedValue;
        value uniform (0 0 0);
    }
    symmetryPlane
    {
        type symmetryPlane;
    }
    frontAndBack
    {
        type empty;
    }
}
Switch off the turbulence model because we run a laminar simulation.

Remove the “functions” block. It is not necessary in this exercise, and will cause problems since it’s a laminar simulation.

simpleFoam is a steady-state solver that uses an iterative algorithm called SIMPLE: the pressure field and the velocity matrix are under-relaxed to ease the convergence. Hence, in this case, the “time-step” refers to the iteration number (see `system/fvSolution` for convergence criteria).

Switch off the turbulence model because we run a laminar simulation.
#3 - Poiseuille flow (4/4)

We use the same parameters as Exo 2

To start the simulation: \$ \text{simpleFoam}

To view the results: \$ \text{paraFoam}

1 – Filters>Data Analysis>Plot Over Line

2 – We are going to plot the velocity profile in the vertical cross-section at the middle of the pipe

3 – Deselect the viewing of the pressure field p
### Objectives:

- Simulate a drainage (a non-wetting fluid pushing a wetting fluid) experiment in a simple 2D capillary tube.
- Example adapted from the damBreak tutorial detailed in the official user guide.
- Use of an interface tracking solver (`interFoam`, VoF) with the equations:
  
  \[
  \frac{\partial \rho \mathbf{U}}{\partial t} + \nabla (\rho \mathbf{U} \mathbf{U}) = -\nabla p + \nabla \cdot (\mu (\nabla \mathbf{U} + \nabla \mathbf{U}^T)) + \mathbf{F}_\alpha
  \]
  
  \[
  \frac{\partial \alpha}{\partial t} + \nabla (\mathbf{U} \alpha) = 0
  \]
  
  \[
  \mathbf{U} = \alpha \mathbf{U}_l + (1 - \alpha) \mathbf{U}_g
  \]
  
  \[
  \rho = \alpha \rho_l + (1 - \alpha) \rho_g
  \]
  
  \[
  \mu = \alpha \mu_l + (1 - \alpha) \mu_g
  \]

- Use of the `setFields` utility to initialize the phase distribution.

---

```bash
$ run
cp -r $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak Exo4
cd Exo4
cp ../Exo3/constant/polyMesh/blockMeshDict constant/polyMesh/.
```
#4 - Drainage in a capillary tube (2/5)

```plaintext
$ gedit constant/polyMesh/blockMeshDict

```
#4 - Drainage in a capillary tube (3a/5)

$ gedit constant/transportProperties

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "constant";
    object transportProperties;
}

// ***************************************************************************************************************************/

phases (water nw);

water
{
    transportModel Newtonian;
    nu
        nu [ 0 2 -1 0 0 0 0 ] 6e-05;
    rho
        rho [ 1 -3 0 0 0 0 0 ] 1000;
}

nw
{
    transportModel Newtonian;
    nu
        nu [ 0 2 -1 0 0 0 0 ] 6e-05;
    rho
        rho [ 1 -3 0 0 0 0 0 ] 10;
}

sigma
    sigma [ 1 0 -2 0 0 0 0 ] 0.097;

Wetting phase properties
Non-wetting phase properties
Surface tension

Name of the wetting (here called « water ») and non-wetting phases. In the computational domain, the two immiscible phases will be differentiated by the phase indicator function alphawater ( =1 for the wetting phase, =0 for the non-wetting phase)
We use a hydrostatic pressure.

Top and bottom have similar boundary conditions.
alphawater represents the wetting/non-wetting phase distribution in the computational domain. (alpha=0 for the non-wetting, alpha=1 for the wetting)

The phase distribution will be first initialized using the `setFields` utility. However, at this stage of the pre-processing, only the boundary conditions need to be specified.

`setFields` will overwrite 0/alphawater. It is recommended to make a backup

```
$ gedit 0/alphawater
```

Definition of the contact angle at the solid boundaries. Here theta=45 degrees

`limit gradient` to limit the wall-gradient such that alpha remains bounded on the wall.
Specify that the simulation will be without gravity in a laminar flow regime.

```plaintext
$ gedit constant/g

```
Set the `writeControl` parameter to `adjustableRunTime` when using an adjustable time step. The output files will be written every `writeInterval` seconds.

If `yes` value, then it means that the `controlDict` file can be modified on the fly.

Switch on the automatic time step management according to the Courant Numbers value (`maxCo` and `maxAlphaCo`).

`maxDeltaT` restricts the maximum value of the time step.
Before we start the simulation, we are going to specify the initial phase distribution with `setFields`.

```bash
$ cp 0/alphawater.org/ 0/alphawater
$ paraFoam
$ gedit system/setFieldsDict
$ setFields
$ paraFoam
```

```c
#ifdef __cplusplus
#include "OFA.hh"
#endif

// *-------------------------------------------------------------------

defaultFieldValues
{
    volScalarFieldValue alphawater 1
};

regions
{
    boxToCell
    {
        box (0 0 0) (0.5e-3 2e-3 .1e-3);
        fieldValues
        {
            volScalarFieldValue alphawater 0
        }
    }
};

// *-------------------------------------------------------------------
```
#4 - Drainage in a capillary tube (5/5)

Start the immiscible two-phase flow simulation:

$\text{interFoam}$

theta=45 degrees

\begin{align*}
t &= 0.05s \\
t &= 0.10s \\
t &= 0.15s \\
t &= 0.20s
\end{align*}

theta=20 degrees

\begin{align*}
t &= 0.05s \\
t &= 0.10s \\
t &= 0.15s \\
t &= 0.20s
\end{align*}

Exo4bis : Same exercise with a contact angle of 20 degrees
In the next parts...

Part II: Mesh complex geometries, application to the evaluation of permeability, transport of passive scalars at the pore-scale

Part III: First programs, development of a heat transfer solver in porous media

http://web.stanford.edu/~csoulain/