Introduction to OpenFOAM

Recep Kahraman

University of Exeter

rek209@exeter.ac.uk

18/04/2016

Overview

Brief overview of OpenFOAM

- OpenFOAM represents Open Source Field Operation and Manipulation.
- OpenFOAM is one of the first C++ library used to solve partial differential equations (PDEs) and ordinary differential equations (ODEs).
- It is licensed under the GNU General Public License (GPL). That means OpenFOAM can be freely used and distributed with the source code.
- It can be run in parallel computers and there is no need for separate licenses.
- It is completable with available commercial CFD applications and it is under continuous development.
- It is used by a wide-spread community around the world (industry, academia and research labs).

Brief overview of OpenFOAM

- **Multiphysics capabilities of OpenFOAM :**
  - Computational fluid dynamics.
  - Multiphase flows and mass transfer.
  - Heat transfer and conjugate heat transfer.
  - Combustion and chemical reactions.
  - Stress analysis and fluid-structure interaction.
  - Particle methods (DEM, DSMC, MD) and Lagrangian particles tracking.
  - Dynamic mesh handling, 6-Degrees of Freedom (6 DOF) solvers, and adaptive mesh refinement.
  - Computational aero-acoustics, computational electromagnetics, computational solid mechanics, etc.

Brief overview of OpenFOAM

- **Implementation of OpenFOAM:**
  - Finite Volume Method (FVM) based solver.
  - Uses collocated polyhedral unstructured meshes.
  - Provides many discretisation schemes including higher order and has second order accuracy in space and time.
  - SIMPLE and PISO methods are used for pressure-velocity coupling.
  - Parallel computing capability.
  - Own mesh generation tools (blockMesh,snappyHexMesh).
  - Manipulation and conversion utilities.
  - Meshes generated by any of the major mesh generators and CAD systems can be converted to OpenFOAM (ansysToFoam etc.)

Brief overview of OpenFOAM

- **Physical Models in OpenFOAM:** Physical modelling library includes:
  - Thermophysical models and physical properties for the materials (gas, liquids, etc.).
  - Newtonian and non-Newtonian viscosity models.
  - Various turbulence modelling capabilities.
    - Reynold Averaged Navier-Stokes (RANS) Models.
    - Detached eddy simulation (DES).
    - Large eddy simulation (LES).
  - SIMPLE and PISO methods are used for pressure-velocity coupling.
  - Parallel computing capability.
  - Own mesh generation tools (blockMesh, snappyHexMesh).
  - Manipulation and conversion utilities.
  - Meshes generated by any of the major mesh generators and CAD systems can be converted to OpenFOAM (ansysToFoam etc.).

Brief overview of OpenFOAM

- **More on the physical models in OpenFOAM:**
  - Two-fluid model (Euler-Euler type) for multiphase flows.
  - VOF method for multiphase flows.
  - Lagrangian particle methods (DSMC, MD).
  - Discrete particle modelling (DPM, MP-PIC).
  - Porous media, chemical reactions, combustion and optimisation.

Brief overview of OpenFOAM

- **Ready solvers to use:**
  - Basic Solvers: Laplace "laplacianFoam", potential flow "potentialFoam", etc.
  - Solvers for incompressible and compressible flows like "simpleFoam" and "pisoFoam based on segregated pressure based algorithms.
  - There are solvers for buoyancy-driven flows and conjugate heat transfer.
  - Solvers for multiphase flows: Euler-Euler, VOF for free surfaces, multiple phases, cavitation, phase change.
  - Solvers for porous media, chemical reactions, combustion and optimisation.

Brief overview of OpenFOAM

- **If this is not enough!**
- You can implement your own solver!
- Entire philosophy of the OpenFOAM is to give the users flexible and programmable environment for numerical simulations.
- Implementation of the new solvers is possible by high level of programming, modification of the existing code, equation mimicking, etc.;

Brief overview of OpenFOAM

Equation mimicking:

- The transport equation:

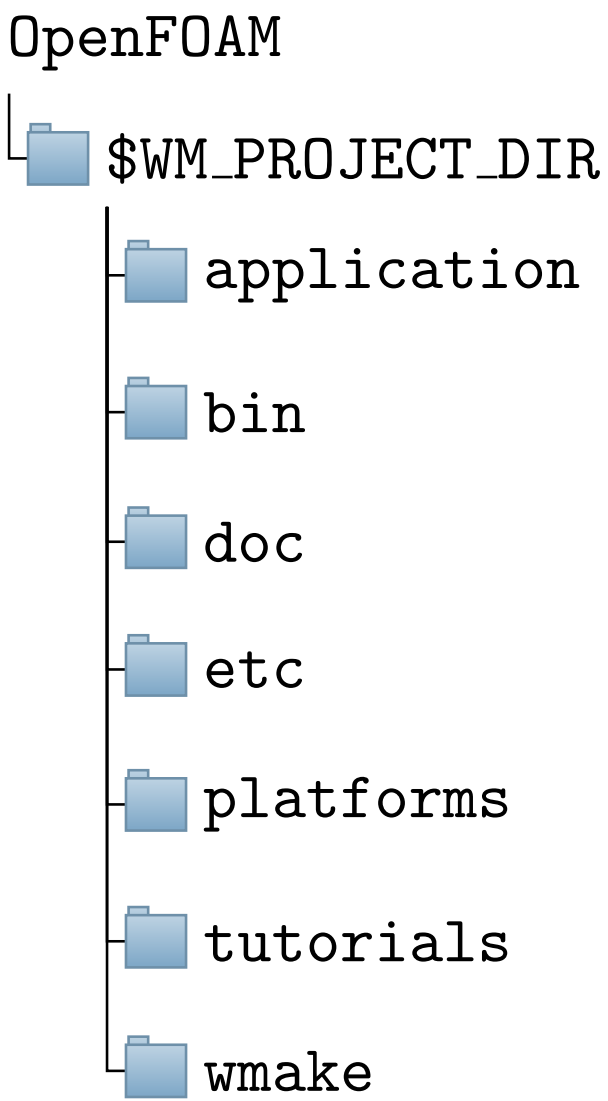
$$\frac{\partial \rho U}{\partial t} + \nabla \phi U - \nabla \mu \nabla U = -\nabla p \tag{1}$$

can be written in OpenFOAM as following

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

In OpenFOAM, identification of the terms is easy, just compare with the original equation!

OpenFOAM directories

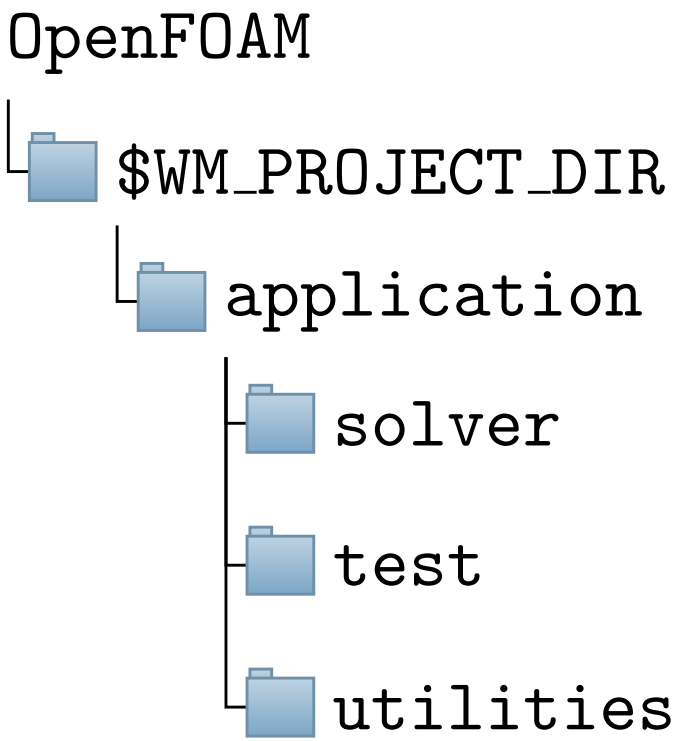


- If you install OpenFOAM in the default location, the directory $WM_PROJECT_DIR will be in:

  `$HOME/OpenFOAM/OpenFOAM-2.3.1`

- The $WM_PROJECT_DIR will contain all the directories and additional files like README.org, COPYING, etc.
- But the most important one is Allwmake file, which compiles OpenFOAM. Another is wcleanAll, which deletes all the compiled files in the directory.

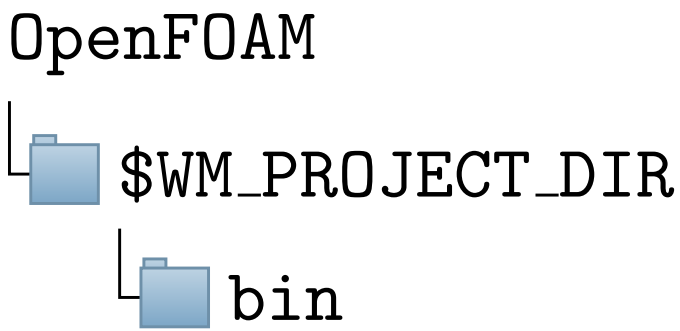
OpenFOAM directories



By typing app or cd $WM_PROJECT_DIR, you can visit the applications directory, which includes:

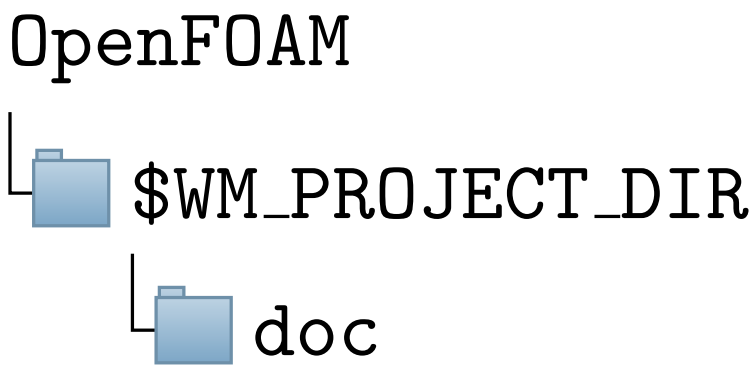
- solvers contains the source code for existing solvers in OpenFOAM.
- test contains the source code for test cases, showing the usage of OpenFOAM classes.
- utilities includes the source code for the distributed utilities (parallelProcessing, postProcessing, etc.).

OpenFOAM directories



- You can find shell scripts, such as paraFoam, foamNew, foamLog, foamJob, etc.

OpenFOAM directories

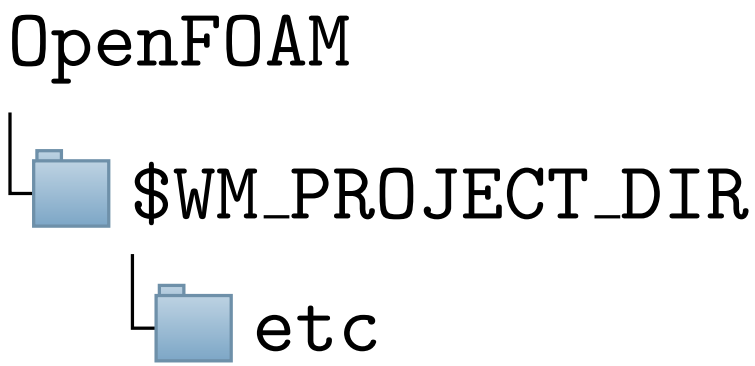


- You can find all the documentation in doc directory. And by typing:

  ```
  acroread $WM_PROJECT_DIR/doc/Guides-a4/UserGuide.pdf
  acroread $WM_PROJECT_DIR/doc/Guides-a4/ProgrammersGuide.pdf
  ```

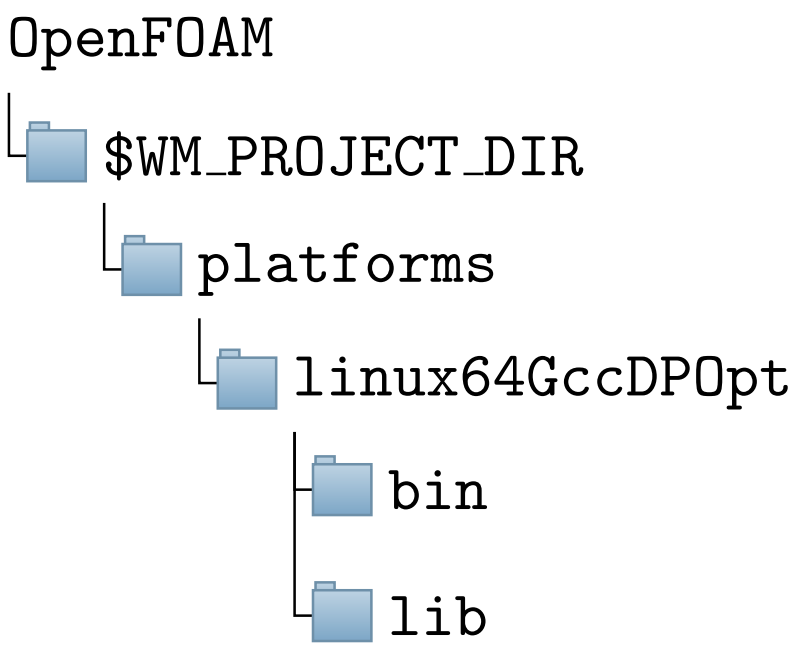
  you can read the User and Programmers Guide.

OpenFOAM directories



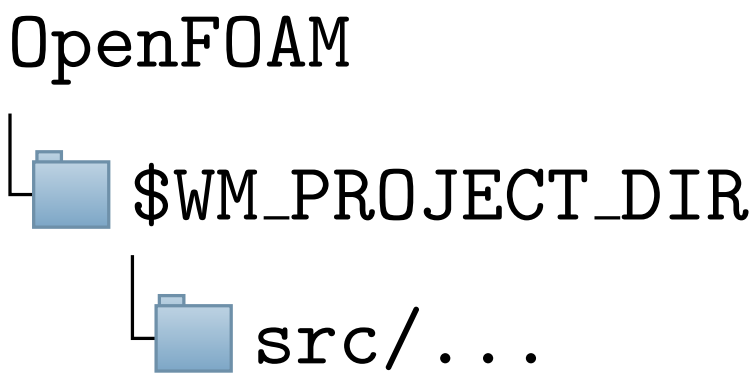
- etc contains the global OpenFOAM instructions, environment files and default thermochemical database thermoData/thermoData.
- It also contains the super dictionary controlDict, where you can set several debug flags and defaults units.

OpenFOAM directories



- The platform directory contains the binaries generated when compiling the applications. After compilation the binaries will be located in the $WM_PROJECT_DIR/platforms/linux64GccDPOpt/bin directory
- The libraries generated by compiling the source code in the src directory, after the compilation will be located in the $WM_PROJECT_DIR/platforms/linux64GccDPOpt/lib directory.

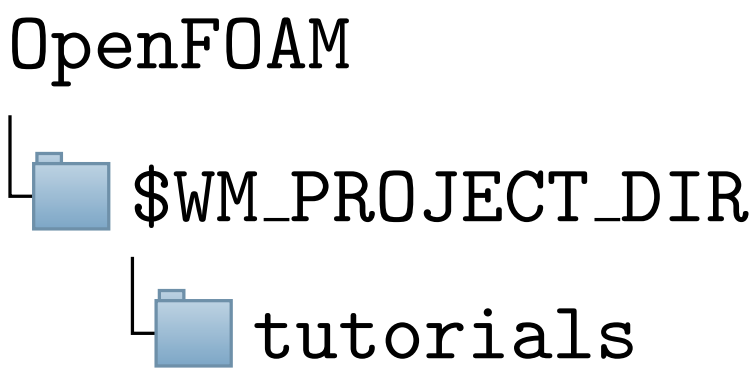
OpenFOAM directories



The src directory contains all the libraries. This is the core of OpenFOAM.

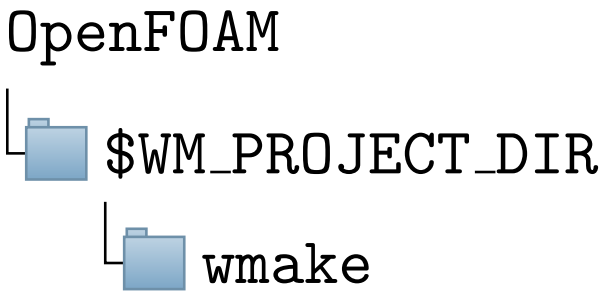
- They are divided into subdirectories; important library is finiteVolume which contains all classes for finite volume discretisation like mesh handling, operators (divergence, laplacian, gradient, etc.)
- OpenFOAM, which is the main library includes the definitions of the containers used for the operations, the field definitions, the declaration of the mesh and mesh features such as zones and sets.
- turbulenceModels contains all turbulence models.

OpenFOAM directories



The tutorials directory contains all example cases for each application.

OpenFOAM directories



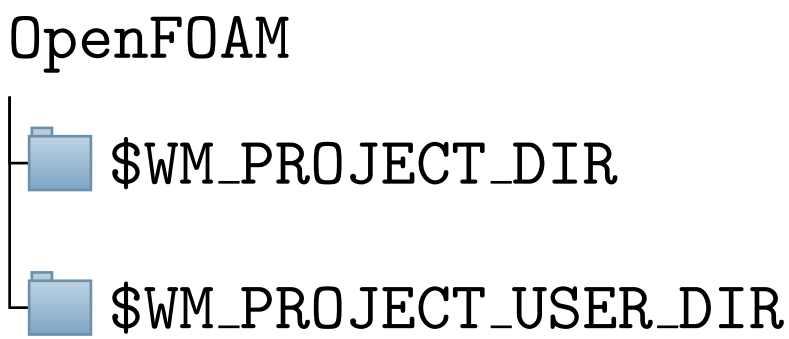
The wmake directory contains wmake and wclean files which understand the file structure in OpenFOAM. The wmake command compiles the corresponding code and wclean cleans up the output from the compilation.

Tip of Day

Until now we have seen the original installation directory of OpenFOAM.

Do not change anything in $WM_PROJECT_DIR,
if you are not 100 % sure!

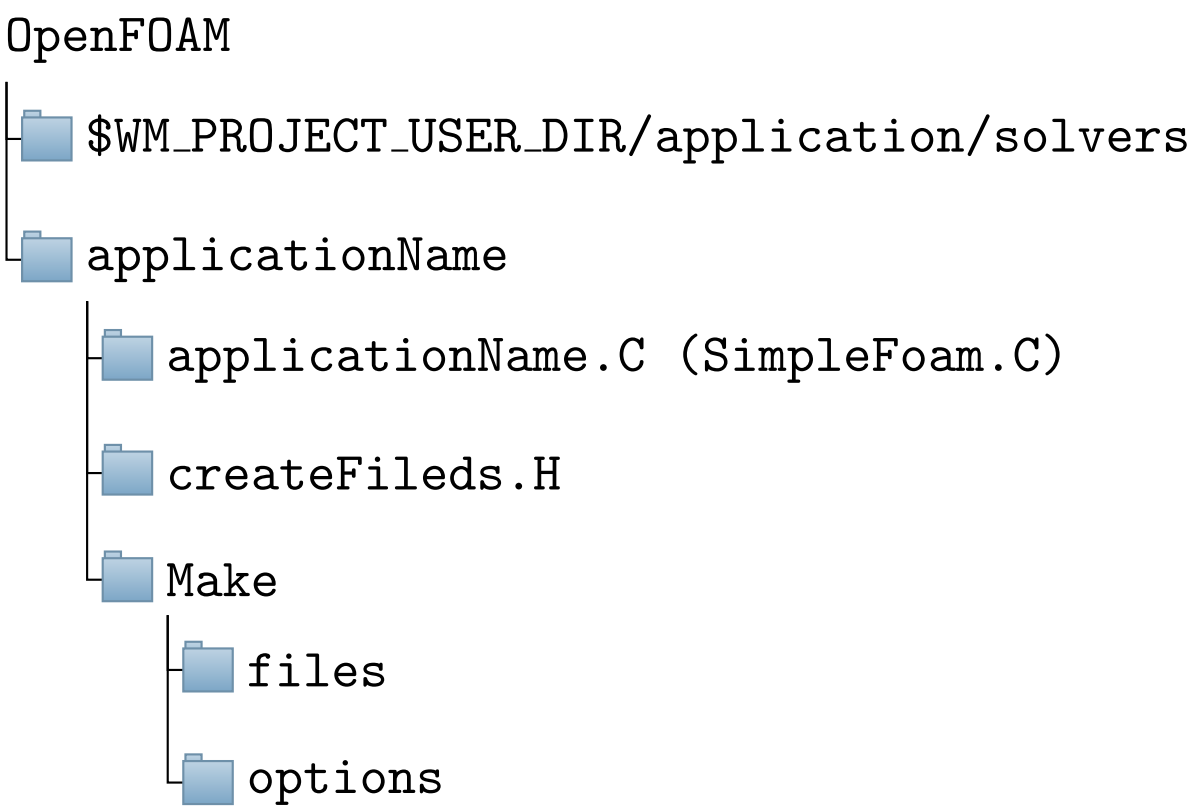
OpenFOAM directories



All user files are located in $WM_PROJECT_USER_DIR. It is highly recommended to put the source code in $WM_PROJECT_USER_DIR/applications. Also you need to modify the Make/files and Make/options to declare the new name and location of the compiled binaries and libraries. Also, it is recommended to create two directories:

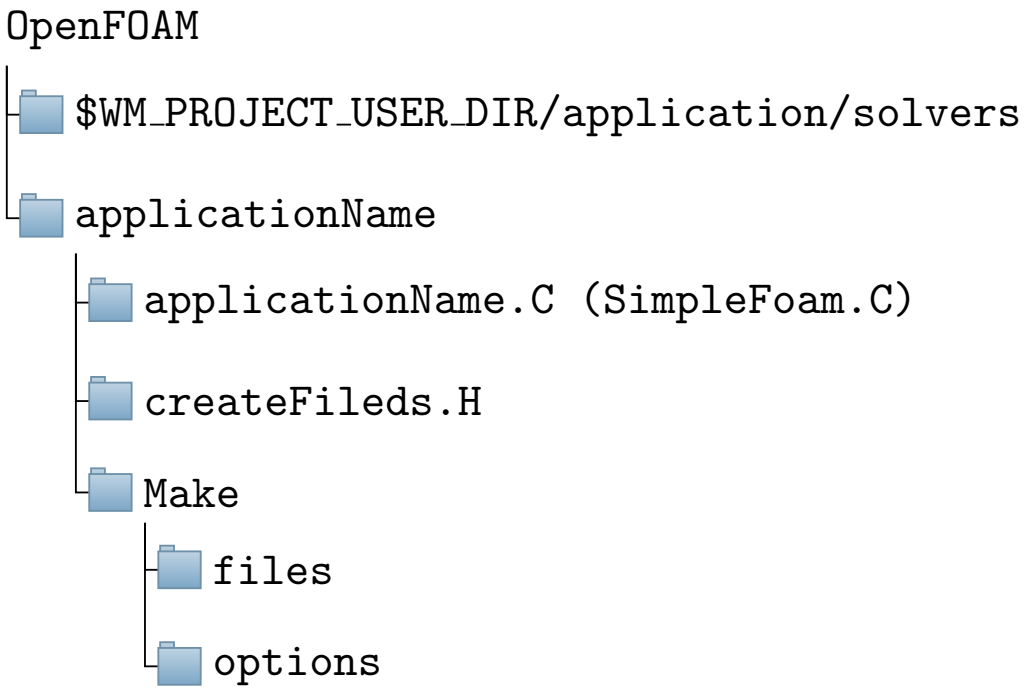
- $WM_PROJECT_USER_DIR/run
- $WM_PROJECT_USER_DIR/src

OpenFOAM directories



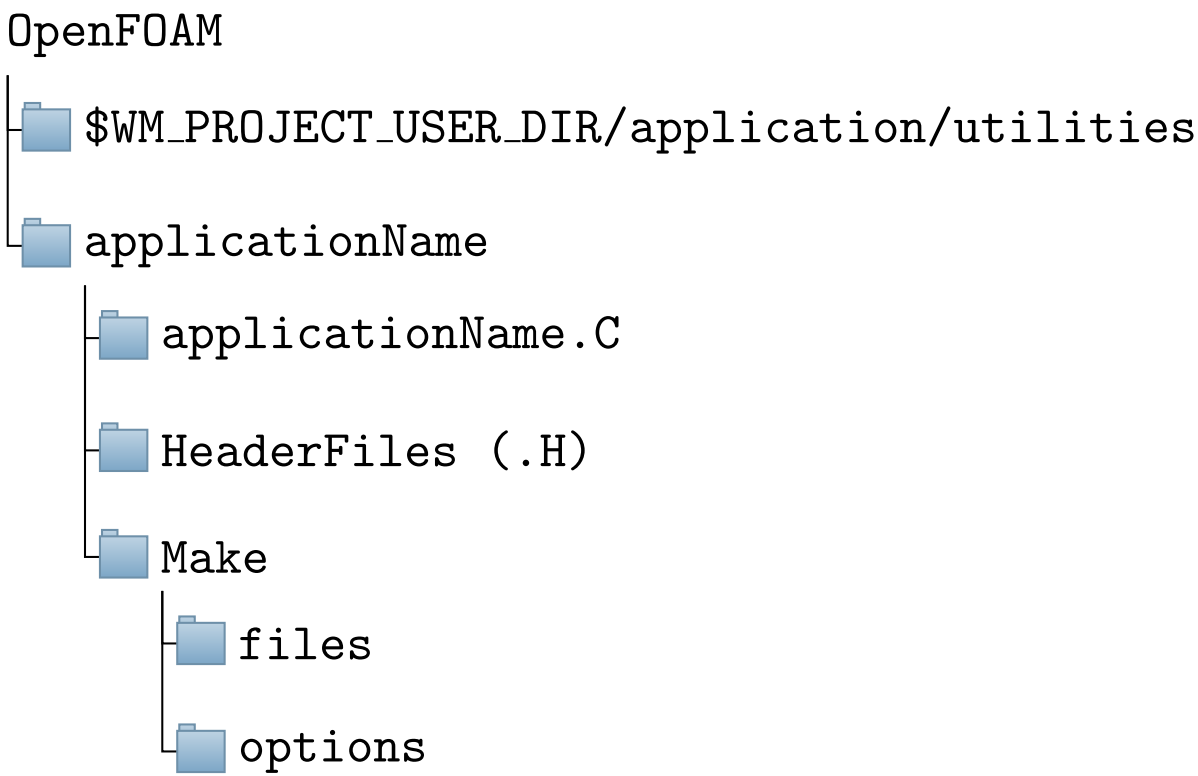
The applicationName directory contains source code, applicationName.C is the source code and createFields declares all the variables and initialise the solution.The files names the all the source files options giving the address of included files and libraries and link them with the solver.

OpenFOAM directories



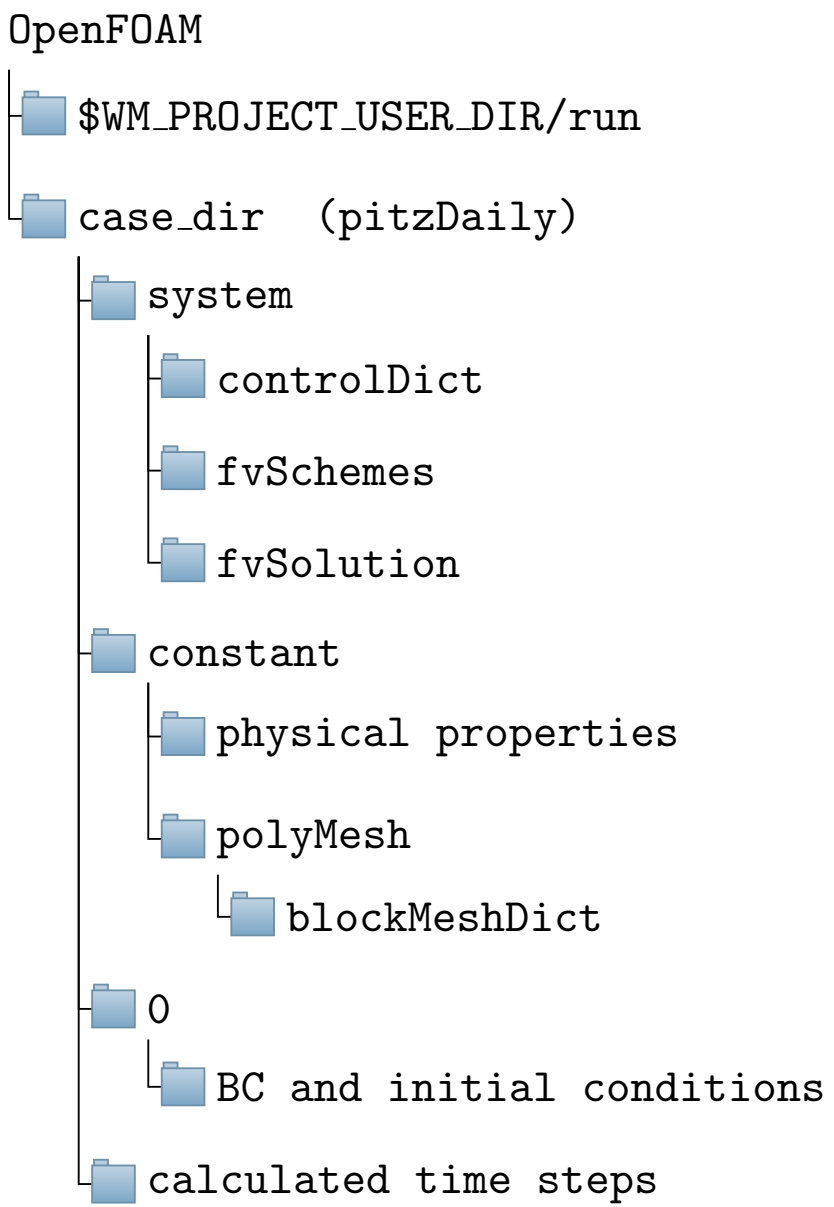
!!!For your own solver, it is highly recommended to put your source code into $WM_PROJECT_USER_DIR with the same as the original structure $WM_PROJECT_DIR/applications. Also you need to modify Make/files and Make/options to declare the new name and location of the compiled binaries and libraries!!!

OpenFOAM directories



applicationName directory contains the source code, applicationName.C is the source code and HeaderFields includes all the header files to compile the utility. The files names all source files, options gives the address of the included files and libraries and link them with the solver.

OpenFOAM directories



- system contains the run time control and numerics for your case.
- constant contains physical and turbulence properties, etc.
- constant/polyMesh contains mesh information.
- 0 contains BC and initial conditions.
- calculated time steps contains the solution for the corresponding time step.

Ready to run our first tutorial :)
Enjoy!

Your first tutorial with OpenFOAM

Tips and tricks in OpenFOAM

Tips and tricks in OpenFOAM

- As I am using OpenFOAM since 2008, I would like to share my experience and observations.
- During this period I have run different kind of simulations with different meshes (including very good and bad ones as well), and used different BC and models and so on.
- I think, CFD is quite a long journey so I suggest before starting to run a simulation, get familiar with the theory behind CFD.
- Then run the simulations and make your own conclusion from it.
- These slides are reference only, so I am not responsible for your mistakes or brilliant results :).

Tips and tricks in OpenFOAM

If you want to have several versions of OpenFOAM your .bashrc file should look like as follows:

```
### OPENFOAM ###
# source $HOME/OpenFOAM/OpenFOAM-2.0.x/etc/bashrc
# source $HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc
# source $HOME/OpenFOAM/OpenFOAM-2.2.0/etc/bashrc
# source $HOME/OpenFOAM/OpenFOAM-2.2.x/etc/bashrc
# source $HOME/OpenFOAM/OpenFOAM-2.3.0/etc/bashrc
source $HOME/OpenFOAM/OpenFOAM-2.3.1/etc/bashrc
### OPENFOAM ###
```

and you can access the file by typing gedit ~/.bashrc

Tips and tricks in OpenFOAM

Submitting job in parallel work:

```
#!/bin/sh

#PBS -N Exhelical_HX
#PBS -l nodes=2:ppn=30,walltime=240:00:00
#PBS -M rek209@ex.ac.uk
#PBS -m abe
#PBS -W group_list=group2
#PBS -q fat1278q

module load openmpi_gcc/1.10.0 gcc/4.7.3
export FOAM_INST_DIR=/home/apps/openfoam_gcc/2.3.1
source $FOAM_INST_DIR/OpenFOAM-2.3.1/etc/bashrc
#PBS -l nodes=fat001:ppn=32+fat002:ppn=32
```

Tips and tricks in OpenFOAM

Second part:

```
cd $PBS_O_WORKDIR    # run from this directory
. $WM_PROJECT_DIR/bin/tools/RunFunctions # loads RunFunctions
rm log.*
export NP=60 # number of processors
#runApplication decomposePar -force
#runParallel renumberMesh $NP -overwrite
#runParallel `getApplication` $NP
#runApplication reconstructPar
runApplication interExCondPhaseChangeFoam
```

Tips and tricks in OpenFOAM

If you want to do it in your workstation you can type:

```
decomposePar > log.decomposePar 2>&1
mpirun -np $NP renumberMesh -overwrite -parallel
reconstructPar -latestTime > log.reconstructPar 2>&1
```

- Dip note: Depending on your resources, your job request and your job priority, you may need to wait in the queue hours or even days, so before submitting your job, I would suggest to double check your script and the case!!!

Tips and tricks in OpenFOAM

- Make sure that your geometry has the correct dimensions, if not, you can use the utility transformPoints -scale '(0.001 0.001 0.001)'.
- Before starting to run your simulations, do not forget to check your mesh quality by using checkMesh.
- **Mesh quality is vital** for accuracy of your simulation!
- checkMesh will give you information about the topological errors.
- If you have topological errors, you must repair them.
- You may able to run your simulations with mesh quality errors such as skewness, aspect ratio, minimum face area, and non-orthogonality, but it will affect severely the quality of your results.
- So, if checkMesh is warning you about any kind of errors, I would **remesh** the geometry again.

Tips and tricks in OpenFOAM

- If you have an error in your mesh, the error about cells, faces, and/or points will be written in sets file in **constant/polyMesh/sets/** directory.
- If you want to visualise the errors of the mesh you need to convert it to VTK format by using foamToVTK.
- It will create a VTK file which includes the failed cells, faces or points and then you can use paraFoam or paraView to visualise failed sets.
- **And again, mesh quality is very important** for accuracy of your simulation! Probably, I am spending 60 % of my time to get good quality mesh!

Tips and tricks in OpenFOAM

If you want to check the mesh quality criteria in OpenFOAM, you can go to read the file:

```
$WM_PROJECT_DIR/src/OpenFOAM/meshes/
primitiveMesh/primitiveMeshCheck/ primitiveMeshCheck.C
```

```
Foam::scalar Foam::primitiveMesh::closedThreshold_  = 1.0e-6;
Foam::scalar Foam::primitiveMesh::aspectThreshold_  = 1000;
Foam::scalar Foam::primitiveMesh::nonOrthThreshold_ = 70;// deg
Foam::scalar Foam::primitiveMesh::skewThreshold_    = 4;
Foam::scalar Foam::primitiveMesh::planarCosAngle_   = 1.0e-6;
```

If you are not happy with it, you are free to change it! It is highly recommended to run renumberMesh. It will renumber the mesh and decrease the bandwidth of the mesh. **I always do it!**

!!!To sleep well, you need good mesh quality!!!

Tips and tricks in OpenFOAM

Boundary conditions:

- Make sure that your BCs have physical meaning.
- You need to avoid numerical diffusion near the boundaries, especially near the inlets, so minimise the grid skewness, non-orthogonality, growth rate, and aspect ratio near the boundaries.
- You need to specify BC for each variable which you are using in your simulation.
- Initial BCs are important for stability and convergence rate. Unphysical initial BCs can cause divergence.
- In some cases, you can use potentialFoam to obtain the initial solution which is computationally cheap. It will give you the sensitivity of your mesh quality as well.

!!!To sleep well, you need good mesh quality!!!

Tips and tricks in OpenFOAM

Before running your simulation

- Remember, you can change the parameters in the dictionaries controlDict, fvSchemes and fvSolution during the run. But you will need to set the keyword runTimeModifiable to yes in the controlDict directory.
- In order to start a stable simulation, you can start with first order scheme and then switch to higher order scheme.
- When the simulation starts, I usually use more correctors steps (for PIMPLE methods). As the solution advances or stabilises, I decrease the number of corrector steps, but use at least one corrector step.
- Never use first order scheme for the entire simulation, they are too diffusive. They are extremely robust but they will under predict the forces and smear the gradients.
- Start stable and finish accurate!

Discretisation schemes convective terms

Extremely stable but too diffusive scheme:

```
divSchemes
{
div(phi,U)                          Gauss upwind;
div(phi,epsilon)                      Gauss upwind;
div(phi,k)                          Gauss upwind;
div((nuEff*dev(T(grad(U)))))        Gauss linear;
}
```

Discretisation schemes convective terms

An accurate and robust scheme:

```
divSchemes
{
div(phi,U)                      Gauss linearUpwind grad(U);
div(phi,epsilon)                Gauss linearUpwind default;
div(phi,k)                      Gauss linearUpwind default;
div((nuEff*dev(T(grad(U))))) Gauss linear;
}
```

Discretisation schemes: convective terms

Even more accurate but less stable:

```
divSchemes
{
div(phi,U)                          Gauss linear;
div(phi,epsilon)                    Gauss linearUpwind default;
div(phi,k)                          Gauss linearUpwind default;
div((nuEff*dev(T(grad(U)))))        Gauss linear;
}
```

Discretisation schemes: diffusive terms

Fully orthogonal scheme:

```
laplacianSchemes
{
default Gauss linear corrected;
}
```

Blended scheme between a fully orthogonal and non-orthogonal scheme:

```
laplacianSchemes
{
default Gauss linear limited psi;
}
```

Discretisation schemes: diffusive terms

- The selection of the scheme depends on your mesh quality.
- For meshes with low non-orthogonality, the corrected or limited 1 scheme.
- For meshes with high non-orthogonality, the limited ψ scheme. Higher non-orthogonality, lower ψ.

Discretisation schemes: gradient terms

- Gradients can be calculated using the Gauss or least squares method, they are called in OpenFOAM as
  - Gauss linear;
  - leastSquares;
- leastSquares method is more accurate but it is less stable on tetrahedral meshes.
- You can use gradient limiters to overcome over- and under- shoot of the gradients.
- Available limiters are faceLimited, faceMDLimited, cellLimited and cellMDLimited; diffusivity changes from highest to lowest respectively.

Discretisation schemes: gradient terms

- If you have non-orthogonality higher than 80, I suggest you to remesh your geometry.
- If you are running LES then you must get non-orthogonality less than 60.
- I do not like to fix poor mesh quality with a numerical scheme, but if you have to use one of the following:

Discretisation schemes: choosing scheme

For non-orthogonality between 70 and 80

```
gradSchemes
{
default faceLimited leastSquares 1.0;
grad(U) faceLimited leastSquares 1.0;

}
divSchemes
{
div(phi,U)  Gauss linearUpwind grad(U);
div(phi,epsilon} upwind;
div(phi,k)     Gauss upwind;
div((nuEff*dev(T(grad(U))))) Gauss linear;
}
```

```
laplacianSchemes
{
default Gauss linear limited 0.5;
}
snGradSchemes
{
default limited 0.5;
}
```

Discretisation schemes: choosing scheme

- Change the number of non-orthogonal corrections.
- For non-orthogonal mesh between 70 and 80:
  nNonOrthogonalCorrectors 3;
- For non-orthogonal mesh between 50 and 70:
  nNonOrthogonalCorrectors 2;
- For non-orthogonal mesh less than 50:
  nNonOrthogonalCorrectors 1;

It is safe to use at least one correction!

Under-relaxation factors

Commonly used under-relaxation factors:

```
relaxationFactors
{
p               0.3;
U               0.7;
k               0.7;
epsilon         0.7;
}
```

- The target is the non-linearity of the equation attempted with under-relaxation factors.
- If you increase the under-relaxation factors, the computation will take longer but it will be more stable.

Linear solvers

- The optimal solution for the pressure equation is often the GAMG solver (generalized geometric-algebraic multigrid solver). However, if you use high number of processors, it is good to consider Newton-Krylov type solvers.
- The GAMG solver can give problems in parallel computing. Usually, the problem is related to the keyword nCoarsestCells, so you need to set high number of cells (order of 1000).
- renumberMesh can help to stabilise and increase the speed of the solver.
- The alternative solver is PCG solver. For the velocity, you can use coupled and PBiCG solver.

CFL condition

- It is possible to go up to CFL=5.0 and to have the same accuracy by increasing corrector step in PIMPLE solver and decrease under-relaxation factors and tightening the conveyance criteria. But this means higher computational cost!
- If you increase the CFL more than 5.0, it is possible to get convergence but results will be poor!
- I prefer to have CFL=1.0. For LES, you need to have CFL less than 0.6, ideally less than 0.3!

Temporal discretisation schemes

- dttSchemes you have several options as well
- Euler is a first order implicit scheme (bounded).
- Backward is a second order implicit scheme (unbounded). Similar to the linear multistep Adams-Moulton scheme.
- Crank-Nicolson is a second order implicit scheme (bounded).
- If you are not sure which one to use, go for Crank-Nicolson. You have control on stability and accuracy. If you set ϕ to 0, it means first order stable scheme. If you set ϕ to 1 than it is second order and more accurate.

Note on incompressible solvers

- In incompressible solvers pressure is modified:

$$P = \frac{p}{\rho} \tag{2}$$

Thank You