OpenFOAM and EnSight Introductory Training Course

Course notes prepared by Nainesh Patel, Civil Engineering

Tuesday 27^{th} *November to Wednesday* 28^{th} *November*

Acknowledgements to Dr Hassan Hemida and David Ryan.

UNIVERSITY OF BIRMINGHAM

1 Introduction

The OpenFOAM (Open Field Operation and Manipulation) CFD Toolbox is a free, open source CFD software package. It has a large user base across most areas of engineering and sciences. The software has an extensive range of features to solve anything from complex fluid flows involving turbulence, chemical reactions and heat transfer, to solid dynamics and mechanics.

The focus of this course is to explain to new users of OpenFOAM how to use existing tutorials to solve laminar and turbulent fluid flows found in the release of OpenFOAM and modify it to suit the user. This course will cover the following:

1. Pre-processing
    - 1.1 Generating geometry using blockMesh.
    - 1.2 Choosing boundary conditions.
    - 1.3 Setting environment variables.
2. Solving CFD problems
    - 2.1 Choosing the turbulence model
    - 2.2 Monitoring a simulation
3. Post-processing
    - 3.1 Visualising results using velocity vectors, streamlines, contour plots

Associated files for this course should be placed on the desktop of your computer, if you do not have these files please feel free to email me at nxp049@bham.ac.uk .

Contents

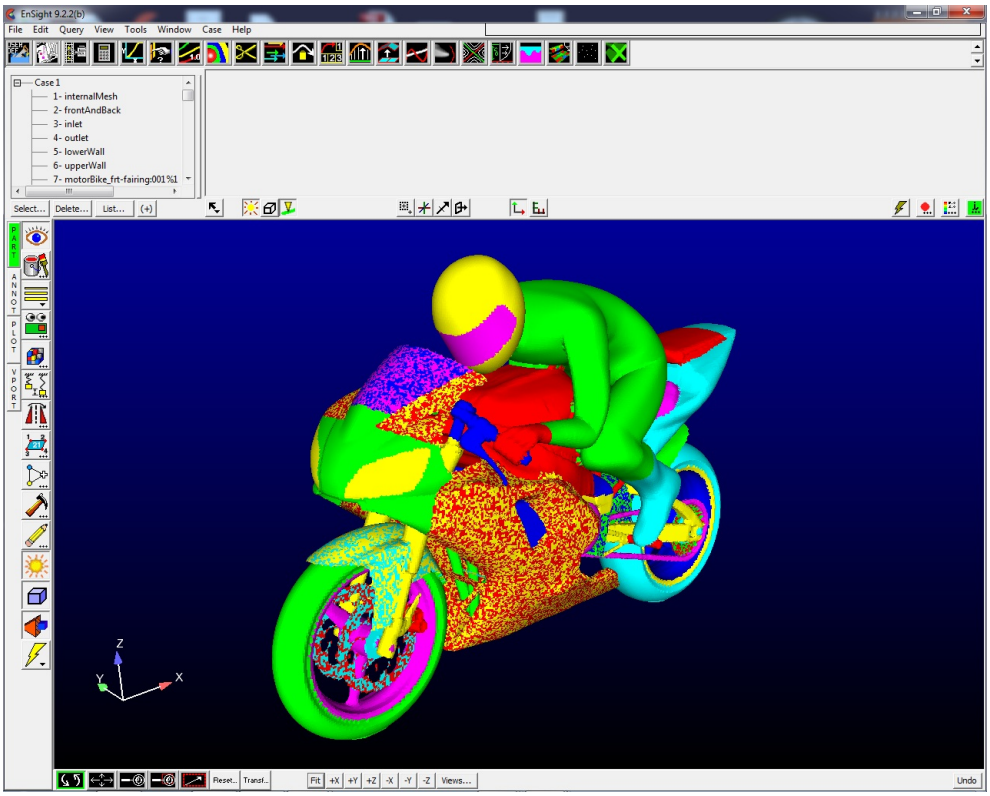
2 Basic viewing controls in EnSight 9.2

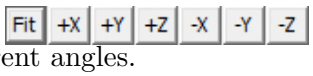
Purpose: The program EnSight 9.2 allows you to view a geometry or a mesh generated by OpenFOAM. It is also the software we will be using to post-process results.

1. Launch EnSight 9.2. 

.
2. Close the welcome message, click on file > Open...
3. Navigate to the courseFiles folder which should be located on your desktop. Open the motorbike directory, then the Ensight_Mesh directory, double click on the **motorbike.case** file.

4. Use the visibility icon  to turn patches on and off. A list of all the patches are given in the top left hand corner underneath the Case 1, drop down menu.
   (a) Turn off the frontAndBack, inlet, outlet, lowerWall and upperWall patches,
5. Mouse controls
   (a) Clicking the left mouse button and dragging allows you to rotate the viewing angle, with respect to the axis shown on the screen.
   (b) Clicking the right mouse button and dragging or scrolling allows you to zoom into and out of the geometry.
   (c) Clicking the middle button and dragging allows you to translate the geometry.
   (d) Holding down the ctrl key whilst left clicking and dragging rotates the geometry in an axis perpendicular to the screen.
6. The screen should look something like this:



7. Use the following icons at the bottom of the screen  to fit the geometry into the screen or to view from different angles.
8. A surface mesh of the motorbike can be viewed by clicking on the display shaded surfaces icon  .
9. The opacity of the surface mesh can be toggled by clicking on the overlay hidden lines icon  .

10. Parts of the motorbike can be recoloured by selecting 1 or more patches in the Case 1 drop down menu and clicking on the colour/transparency icon  , then selecting any colour of your choice.
11. **Extension task:** Try turning particular parts of the motorbike on and off and zoom into an area of interest.
12. **Extention task:** View a cross section of the mesh.
    (a) Select the internalMesh.
    (b) Click on the Clip icon  .
    (c) Change mesh slice to y set value to zero.
    (d) Click on create.
    (e) Turn the visibility of all the motorcycle patches off
    (f) Display shaded surfaces.
13. Close EnSight by clicking on File > Quit > Yes.

3 BlueBear startup

3.1 Accessing the cluster

Purpose: OpenFOAM is used through the command line. For those who have not used the BlueBear cluster, access to the software is given through a secure shell (SSH) login client.

1. Launch Exceed, found under Hummingbird connectivity, 

. This is an X-Window server for access to Linux applications.
2. Launch any SSH client. Here we will be using Putty, 

.
    (a) In the host name field type <userName>@bluebear.bham.ac.uk
    (b) Click Open.
    (c) Enter password when prompted
3. Launch any file transfer program. Here we will be using WinSCP, 

.
    (a) In the host name field type bluebear.bham.ac.uk
    (b) Enter your username and password.
    (c) Click login, then continue.

3.2 Basic setup

1. Before we start we need to load the OpenFOAM modules. This is done by typing
   `module load apps/openfoam/v2.0.1`
   in the command line, after
   `[<username>@bluebear1~]$`
   Now source the initialization script as requested with
   `$FOAM_CONFIG`

2. Files are required to be placed in specific directories. The following commands should be executed in the home directory

```
mkdir OpenFOAM
cd OpenFOAM
mkdir <userName>-2.0.1
cd <userName>-2.0.1
mkdir -p $FOAM_RUN
```

Note: Every time you type `run` into the command line you will be moved into this directory. We will now go on to solve our first CFD problem.

4 Lid-driven cavity flow

4.1 Description of the problem

This tutorial will describe how to pre-process, run and post-process a case involving isothermal, incompressible fluid flow in a two-dimensional square domain. The geometry is shown in Fig. 1 in which all the boundaries of the square are walls. The top wall moves in the x-direction at a speed of 1 m/s while the other 3 are stationary. Initially, the flow will be assumed laminar and will be solved on a uniform mesh using the icoFoam solver which has been written for laminar, isothermal, incompressible flow. During the course of the tutorial, the effect of increased mesh resolution and mesh grading towards the walls will be investigated. Finally, the flow Reynolds number will be increased and the pisoFoam solver will be used for turbulent, isothermal, incompressible flow.

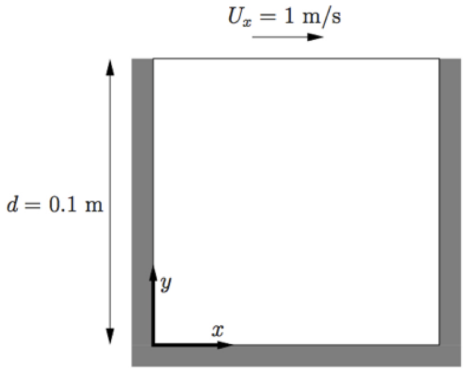


Figure 1: Geometry of a lid-driven cavity.

4.2 Pre-processing

Cases are setup in OpenFOAM by editing case files. A case being simulated involves data for mesh, fields, properties, control parameters, amongst other files required to run a simulation.

Before we go any further, the case files associated with lid-driven cavity flow can be found in the OpenFOAM tutorials directory. Here, we will copy the **cavity** case to our

home directory, modify the case and run the simulation. To obtain the files type the following into the command line.

```
run
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity .
ls
cd cavity
ls
```

The file structure of the case is shown in Fig. 2 The <case> directory is usually given any suitably descriptive name, here it is called **cavity**.

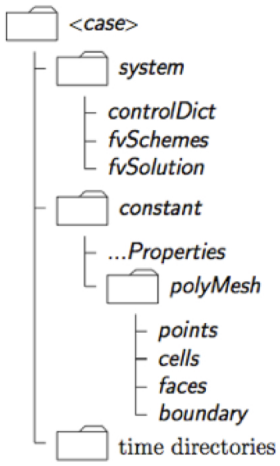


Figure 2: Case directory sructure.

The **system** directory is for setting parameters associated with the solution procedure itself. It must contain at least 3 files: the *controlDict* where run control parameters are set, for example the start time, end time, time step and parameters for data output; *fvSchemes* are where discretisation schemes are chosen; *fvSolution* is where the equation solvers numerical schemes are chosen and where tolerances for the simulation are set.

The **constant** directory contains a description of the mesh in a subdirectory **polyMesh** and files specifying the physical properties for the application concerned, such as the *transportProperties*. where fluid properties are specified.

The *time directories* contain individual files of data for particular fields. The name of each time directory is based on the simulated time at which the data is written. It is sufficient to say that since we usually start our simulations at time t = 0, the initial conditions are usually stored in a directory named **0**.

4.3 Generating the mesh for a cavity/container

OpenFOAM operates in a 3D Cartesian coordinate system and all geometries are generated in 3D. OpenFOAM solves any case in 3D by default but can be instructed to

solve in 2D by specifying a 'empty' boundary condition on boundaries normal to the (3rd) dimension for which no solution is required.

The cavity domain consists of a square of side length d = 0.1 m in the x-y plane. A uniform mesh of 40 by 40 cells will be used initially. The block structure is shown in Fig. 3. The mesh generator supplied with OpenFOAM is an application called `blockMesh`. This application generates meshes from a description specified in an input dictionary file called, *blockMeshDict* which is located in the **constant/polyMesh** directory.

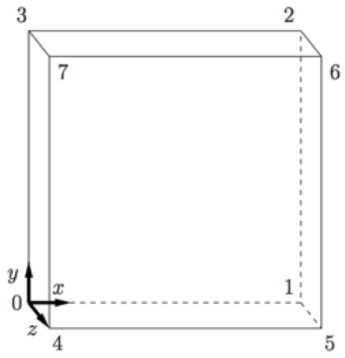


Figure 3: Block structure of the mesh for the cavity.

1. Generating a mesh.
   (a) Navigate to the *blockMeshDict* file.
   ```
   cd constant/polyMesh
   ls
   ```
   (b) The file can be viewed by typing
   ```
   more blockMeshDict
   ```
   Note: pressing the enter key or spacebar enables you to scroll through the file.
   (c) The file can be edited using any text editor of your choice, here we will be using gedit
   ```
   gedit blockMeshDict
   ```
   (d) The file starts with header information in the form of a banner. Following this is a `FoamFile` sub-dictionary which is indicated by the text in the curly braces. This contains information about the file. **Note:** Every file contains a banner and a `FoamFile` sub-dictionary. For simplicity this information will not be mentioned when looking at all other files.
   (e) `convertToMeters 0.1` implies that the units of the dimensions in the rest of the file are set in 10 cm units. If you wish to use mm throughout the file replace 0.1 with 0.001.
   (f) `vertices` contains a list of all the points that are in the geometry. All vertices contain an x, y and z coordinate. The vertices are numbered in the order they appear starting from zero. If you changed the dimensions to mm

then multiply the length of all the vertices by 100.

(g) `blocks` The domain can be divided into a number of blocks. In this case we have 1 block.

i. It starts with the word `hex` indicating a hexahedral mesh is specified.

ii. Each block contains 8 vertices and these have been labelled from 0 to 7.

iii. `(20 20 1)` indicates that there are 20 cells in the x and y direction and 1 cell in the z direction. Increase the mesh resolution to 40 cells in the x and y direction.

iv. The grading is set to `simpleGrading (1 1 1)` indicating that the mesh is uniform and no expansion ratios exists.

(h) `edges` has been left blank, implying all lines between vertices are straight. The line between 2 points can be set to be curved and is specified in this sub dictionary if required.

(i) `boundary` specifies patches; that is regions in the mesh. The patches are a number of faces grouped together. The patch is given any appropriate name for example `movingWall` the names are used as identifiers for defining boundary conditions.

i. `type` refers to the patch type here we have walls and since we are working in 2d the faces which are normal to the 3rd dimension, the `frontAndBack` patch is labelled `empty`.

ii. `faces` are the list of block faces which make up the patch. Each block face is defined by a list of 4 vertex numbers. The order in which the vertices are given must be such that, looking from inside the block and starting with any vertex, the vertices which make the face must be traversed in a clockwise direction to define the other vertices.

(j) `mergePatchPairs` A mesh can be created with more than one block. To connect the mesh together mergePatchPairs can be used. In this case we only have one block.

2. The mesh is generated by executing the *blockMeshDict* file. This is done by running the `blocMesh` application. All application in OpenFOAM are required to be executed whilst in the case directory. The application can be run by typing in the command line.

```
blockMesh
```

3. Read the output generated by running the application looking for any errors which may have come up. Navigate back to the **constant/polyMesh** directory and 5 more files should have been generated *points, owner, neighbour, faces* and *boundary*. Collectively these files define the mesh.

4. The mesh should be checked before any solvers are run. To view the mesh the application

```
foamToEnsight
```

should be executed from the case directory. A directory named **EnSight** will be created, rename this directory using

```
mv EnSight EnSight_Mesh
```

5. **Extention task:** Obtain information and statistics on the mesh by running a mesh checking utility. Any major problems with the mesh will also be highlighted here.

   ```
   checkMesh > logCheck
   ```

   Note: the output which would normally be displayed in the command window will be saved to a file called *logCheck*.

4.4 Viewing the geometry and mesh

1. In WinSCP copy the **EnSight_Mesh** directory from the cluster to your computer.
2. Launch EnSight 9.2.
3. Click on file > Open...
4. Navigate to the **EnSight_Mesh** directory, double click on the *cavity.case* file.
5. You will notice that clicking on the display shaded surfaces will not show you the mesh. To view the mesh a cut plane is required through the domain.
   (a) Select the internalMesh patch
   (b) Then click on the Clip icon  .
   (c) Set the mesh slice in the z direction, then click create.
   (d) The mesh can now be viewed by clicking on the display shaded surfaces icon  .
6. What is your opinion of the mesh?
7. Close EnSight.

4.5 Boundary and initial conditions

1. From the case directory, enter the **0** time directory. The zero time directory represents the initial fields set up for the case. The directory contains 2 fields the pressure field (p) and the velocity field (U). View the contents of the velocity file.
   (a) `dimensions` contains the dimensions of the field. Here `[0 1 -1 0 0 0 0 ]` represents ms^{-1}. Each of the 7 scalars represents a power of the SI base unit. in order the numbers represent mass (kilograms), length (meters), time (seconds), temperature (Kelvin), quantity (kilogram-mole), current (amperes) and luminosity intensity (candela).
   (b) `internalField` The internal velocity field can either be `uniform` or `nonuniform`. In this case initially the whole field is at rest, hence the field is `uniform (0 0 0)`.
   (c) `boundaryField` contains information about the boundary condition of the patches. Three patches were defined in the *blockMeshDict* file and each one requires defining. The lid of the container, `movingWall`, is travelling at 1 ms^{-1} in the x-direction hence a `fixedValue` is assigned to the patch and the value assigned to it is `uniform (1 0 0)`. The `fixedWalls` do not move in the simulation, hence they are assigned a `fixedValue` of `uniform (0 0 0)`. The `frontAndBack` patch type has been defined as `empty`.
2. Now view the pressure file.

(a) The kinematic pressure has been defined in the dimensions of m^2s^{-2}.

(b) `internalField` internal pressure field is set to zero, as we are only interested in the relative pressure differences.

(c) `boundaryField` The wall types are defined as `zeroGradient` implying the normal gradient of pressure is zero.

4.6 Physical properties

1. The physical properties for the case are defined in the **constants** directory. For an icoFoam solver, the only property that must be specified is the kinematic viscosity which is defined in the *transportProperties* file.
2. Check that the kinematic viscosity is set correctly. The keyword for kinematic viscosity is nu. It is represented in equations by the Greek symbol ν. Initially this case will be run with a Reynolds number of 10, where the Reynolds number is defined as:

$$Re = \frac{LU}{\nu}, \tag{4.1}$$

where L and U are the characteristic length and velocity respectively and ν is the kinematic viscosity. Here $L = 0.1$ m, $U = 1$ ms^{-1}. So for $Re = 10$ we choose $\nu = 0.01$ m^2s^{-1}.

4.7 System directory

1. The **controlDict** contains input data relating to the control of time, and reading and writing of the solution data are read in from this file.

   (a) The control directory contains information regarding the start time and end times. which have been defined by `startTime` and `endTime` respectively.

   (b) To ensure temporal and numerical accuracy, the Courant number of the simulation must remain below 1. The Courant number is defined as:

$$Co = \frac{U\Delta t}{\Delta x}, \tag{4.2}$$

   where Δt is the time step and Δx is the size of the smallest cell in the direction of the velocity. The flow varies everywhere so to ensure $Co < 1$ we choose the worst possible case for Δx. Here, $U = 1ms^{-1}$ and $\Delta x = 0.0025m$ hence Δt must be smaller than or equal to 0.0025.

   (c) The `writeControl` is specified as `timeStep` this implies that the number `40` written in `writeInterval` refers to pressure and velocity data fields written into new time directory after every 40 iterations so this will be at times 0.1, 0.2, 0.3, etc seconds.

2. The *fvSchemes* file sets the numerical schemes for terms, such as derivatives in equations, that appear in applications being run. We will not be discussing the discretisation schemes in detail here but it is worth looking at the file to see how the finite volume methods are applied.

(a) All the discretisation schemes are set in the dictionary *fvSchemes*. For this example the Gaussian finite volume integration is used for the discretisation of the divergence terms in the U-equations. Gaussian integration is based on summing values on cell faces, which must be interpolated from cell centres.

(b) The laplacian terms are discretised using the second order central difference schemes.

3. The specification of the linear equation solvers and tolerances and other algorithm controls is made in the *fvSolution* file.

   (a) The first sub-dictionary in our example is `solvers`. It specifies each linear-solver.

   (b) The syntax for each entry within `solvers` uses a keyword that is the word relating to the variable being solved in the particular equation. Here `p` and `U` are being solved for.

   (c) The solvers are iterative; they are based on reducing the equation residual over a succession of solutions. The residual is a measure of the error in the solution so that the smaller it is, the more accurate the solution.

      i. Before solving an equation for a particular field, the initial residual is evaluated based on the current values of the field. After each solver iteration the residual is re-evaluated. The solver stops if the residual falls below the solver `tolerance` or the ratio of current to initial residuals falls below the solver relative tolerance, `relTol`.

      ii. The tolerance should represents the level at which the residual is small enough that the solution can be regarded as sufficiently accurate.

      iii. The solver relative tolerance limits the relative improvement from initial to final solution. In transient simulations, it is usual to set the solver relative tolerance to 0 to force the solution to converge to the solver tolerance in each time step.

   (d) The second sub-dictionary is `PISO`.

      i. Most transient fluid dynamics solver applications in OpenFOAM use the pressure-implicit split-operator (PISO). Algorithms are based on evaluating some initial solutions and then correcting them. The user must therefore specify the number of correctors in the PISO dictionary by the `nCorrectors` keyword.

      ii. In a closed incompressible system such as the cavity, pressure is relative: it is the pressure difference that matters, not the absolute values. In cases such as this, the solver sets a reference level by `pRefValue` in cell `pRefCell`. In this example, both are set to 0. Changing either of these values will change the absolute pressure field, but not the relative pressures or the velocity field.

4.8 Running the icoFoam solver

1. Running the simulation:

(a) The `icoFoam` solver is going to be used. This solver is suitable for transient, incompressible, laminar flow for Newtonian fluids.

(b) To run this simulation and save the output to a file called *logRun* the command

```
icoFoam > logRun
```

should be entered into the command line.

(c) The output that is generated is saved to the file and tells the user the time, maximum Courant number, and the initial and final residuals of the fields being calculated. Now view the file that has been generated.

2. Convert the OpenFOAM results to an EnSight readable directory and rename to Ensight_Run.

4.9 Post-Processing

1. Copy the **EnSight_Run** directory to the **cavity** directory on your desktop computer, and load the results.
2. Visualise the pressure and velocity field.
   (a) Insert a clip plane in the Z direction.
   (b) Ensure the Clip_plane in the patch list is selected then click on the change colour icon 
   (c) Change colour by in the drop down menu to U.
   (d) Now change to p.
   (e) Smooth out fields:
      i. Select all patches by clicking on Case1. Ensure they are all highlighted.
      ii. Click on the calculator icon  .
      iii. Scroll down to the predefined functions area and select ElemToNode then Next >>>
      iv. Select p, scroll up and change the variable name from ElemToNode to Pressure, Click Evaluate.
      v. Close the window.
      vi. Change the colour of the Clip_plane to Pressure.
      vii. Now smooth out the velocity field and colour the Clip_plane with velocity.
   (f) Change colour to any constant constant.
   (g) Close the colour menu window.
3. Streamlines
   (a) Click on the streamlines icon 
   (b) Select U in the variable list.
   (c) Place a tick in Surface restrict.
   (d) Click on Emit...
   (e) Change direction to +/-

(f) Click Create and click on the clip plane at random points

(g) Change the colour of the streamlines to U.

(h) Click on the line thickness icon  and change it to 3 pixels.

4. Velocity vectors

(a) Highlight the Clip_plane patch and select the vector arrow icon 

(b) Select U in the variable list.

(c) In type choose Rect. fixed in the drop down menu.

(d) Change location to Element center

(e) Click on Advanced ...

(f) Change Scale Factor to 0.003

(g) Change projection to Tangential.

(h) Change arrow tip size to Fixed 0.003.

(i) Click Create, then Close.

(j) Select the Vector Arrow part and change its colour to colour by velocity.

(k) Turn the visibility of the vector arrows off.

5. Close EnSight.

5 Increasing mesh resolution

5.1 Pre-processing

1. Open the command line window and in the same directory as the cavity directory make a new directory and label it **cavityFine** using the following command

```
run
mkdir cavityFine
```

2. The **cavity** case will be very similar to the **cavityFine** case so copy the **constant** and **system** directories into the cavityFine directory using

```
cp -r cavity/constant cavityFine
cp -r cavity/system cavityFine
cd cavityFine
```

3. To increase the mesh size, edit the *blockMeshDict* file changing the number of cells from `(40 40 1)` to `(80 80 1)` Save and close the file.

4. Inside the **polyMesh** directory the old mesh still exist. To rewrite the mesh, run the `blockMesh` application again and the files which make the mesh will be rewritten.

5. It is possible to copy across the **0** directory from the cavity case and run the simulation again however the simulation runs much quicker if the results from the course mesh are mapped to the fine mesh. This is done by using the `mapFields` utility.

(a) The field data that `mapFields` maps is read from the time directory specified by `startFrom` and `startTime` in the *controlDict* of the target case. Here we wish to map the final results of the coarser mesh from case cavity onto the

finer mesh of case **cavityFine**. Therefore, since these results are stored in the **0.5** directory of cavity, the `startTime` should be set to `0.5` in the *controlDict* dictionary and `startFrom` should be set to `startTime`.

(b) A description of how to use the mapFields command is given by typing in

```
mapFields -help
```

Note: the `-help` option can be used with any utility and information detailing its application will be provided.

(c) Since the source and target geometry and boundary conditions of the case are identical the `-consistent` option should be included. Run the `mapFields` utility using:

```
mapFields -consistent ../cavity
```

(d) In the **cavityFine** directory there should now be a new time directory named **0.5**

6. Edit the *controlDict* file

(a) The `endTime` should be set to `1.0`.

(b) To ensure that the Courant number remains below 1, the time step should be reduced. Since the cell size has been halved, the new time step `deltaT` should be set to `0.00125` seconds.

(c) `writeControl` can be changed to `runTime` to output data at every specified time interval. In this case, change the `writeInterval` to `0.1`

(d) Save and close the file

5.2 Running the case

1. Run the icoFoam application and save the output to a *logRun* file. This is done by typing:

```
icoFoam > logRun
```

where logRun can be any file name you wish.

2. View the contents of the *logRun* file.
3. The residuals can be plotted in a native Linux plotting program, gnuplot:

(a) Copy the file named *Residuals* from the **cavityFine** folder located on your desktop and place it in the **cavityFine** directory

(b) In the command line type the following

```
gnuplot Residuals -
```

(c) To close the graphic window type

```
exit
```

4. Generate an **EnSight** directory and copy it to your desktop, you can rename it if you wish.

5.3 Post-processing results

1. Create a clip plane
2. Use the Calculator to smooth out the pressure and velocity fields.
3. Colour the Clip_plane by the smoother pressure.

4. Colour the Clip_plane by the smoothed velocity.
5. Create streamlines and velocity vector on the clip plane. What do you notice?
6. Probe the simulation results to investigate the velocity of the fluid along a line through the vertical centre of the domain.
   (a) Select Internal mesh.
   (b) Click on Clip.
   (c) In the drop down menu choose Line.
   (d) Click on Tool location ... and rotate the line about the X axis by -90 degrees. Close the tool location editor.
   (e) Click on Create.
   (f) Select the Clip_mesh_line and click on the Query/Plot icon. 
   (g) Choose At 1d part over distance in the sample drop down menu, choose velocity and click create.
   (h) Click on plot, then on New Plotter.
   (i) Click back to Query and plot the pressure along the same line.
   Note: By clicking on save... in the options for the plot and query toolbar, the data can be exported to a file that can be read in Excel or Matlab.

6 Introducing mesh grading

The error in any solution will be more pronounced in regions where the form of the true solution differs widely from the form assumed in the chosen numerical schemes. For example, a numerical scheme based on linear variations of variables over cells can only generate an exact solution if the true solution is itself linear in form. The error is largest in regions where the true solution deviates greatest from linear form, i.e. where the change in gradient is largest. Error decreases with cell size.

It is useful to have an intuitive appreciation of the form of the solution before setting up any problem. It is then possible to anticipate where the errors will be largest and to grade the mesh so that the smallest cells are in these regions. In the cavity case the largest variations in velocity can be expected near a wall and so in this part of the tutorial, the mesh will be graded to be smaller in this region. By using the same number of cells, greater accuracy can be achieved without a significant increase in computational cost.

A mesh of 40 x 40 cells with grading towards the walls will be created for the lid- driven cavity problem and the results from the finer mesh of section 5 will then be mapped onto the graded mesh to use as an initial condition. The results from the graded mesh will be compared with those from the previous meshes.

6.1 Creating a graded mesh

1. In the same directory as the **cavity** case create a new directory and name it **cavityGrade** using the following commands.
   ```
   run
   mkdir cavityGrade
   ```

2. Copy across the **constant** and **system** directories from the **cavity** case to the new directory.
3. Map fields from the **cavityFine** case to the **cavityGrade** case.
4. In a graded mesh 4 block are required as shown in Fig. 4

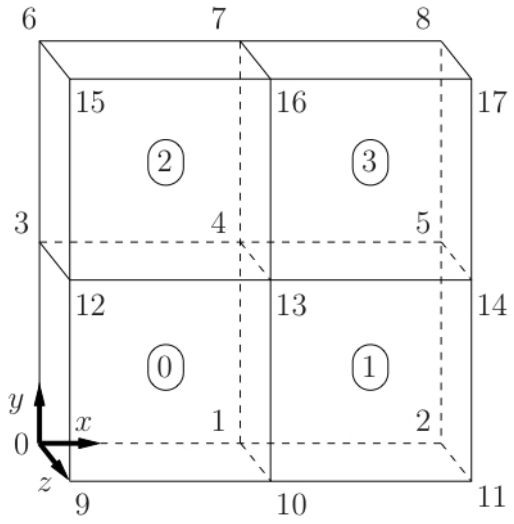


Figure 4: Geometry of lid-driven cavity.

5. To generate the graded mesh *blockMeshDict* requires editing.
   (a) The vertices sub dictionary requires editing

```
vertices
(
   (0 0 0)
   (50 0 0)
   (100 0 0)
   (0 50 0)
   (50 50 0)
   (100 50 0)
   (0 100 0)
   (50 100 0)
   (100 100 0)
   (0 0 10)
   (50 0 10)
   (100 0 10)
   (0 50 10)
   (50 50 10)
   (100 50 0)
   (0 100 10)
   (50 100 10)
   (100 100 10)

);
```

   (b) The blocks sub-dictionary requires editing. The 4 blocks that have been generated have simple grading. The numbers in the vector immediately after `simpleGrading` corresponds to the expansion ratio in their respective directions. The expansion ratio has been defined in Fig. 5.

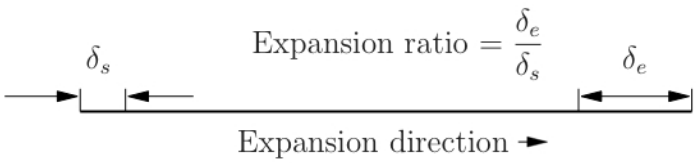


Figure 5: Mesh grading along a block edge.

The blocks sub-dictionary should be changed as follows.

```
blocks
(
  hex (0 1 4 3 9 10 13 12) (20 20 1) simpleGrading (2 2 1)
  hex (1 2 5 4 10 11 14 13) (20 20 1) simpleGrading (1 2 1)
  hex (3 4 7 6 12 13 16 15) (20 20 1) simpleGrading (2 1 1)
  hex (4 5 8 7 13 14 17 16) (20 20 1) simpleGrading (1 1 1)
);
```

(c) The boundary sub-dictionary should be edited as follows

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (6 15 16 7)
            (7 16 17 8)
        );
    }

    fixedWalls
    {
        type wall;
        faces
        (
            (3 12 15 6)
            (0 9 12 3)
            (0 1 10 9)
            (1 2 11 10)
            (2 5 14 11)
            (5 8 17 14)
        );
    }

    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 4 1)
            (1 4 5 2)
            (3 6 7 4)
            (4 7 8 5)
            (9 10 13 12)
            (10 11 14 13)
            (12 13 16 15)
            (13 14 17 16)
        );
    }
);
```

6. Run the *blockMeshDict* file and generate an **EnSight** directory to inspect the mesh.

6.2 Setting controls and running the solver

1. The maximum Courant number will be found close to the moving lid in the smallest cells. Set the `deltaT` time step to be 0.0025 to ensure temporal and numerical stability.
2. Adjust the write interval so that data is written every 0.1 seconds.
3. Run the icoFoam solver saving the output from the terminal into a log file
4. View the residuals using gnuplot.
5. View the results in EnSight.

7 Increasing the Reynolds number

The cases solved so far have had a Reynolds number of 10. This is very low and leads to a stable solution quickly with only small secondary vortices at the bottom corners of the cavity. We will now increase the Reynolds number to 100, which will take a slightly longer time to converge. The coarsest mesh in case cavity will be used initially.

1. Make a new directory in the run directory and label it **cavityHighRe**
2. Copy across the **constant** and **system** directories from the **cavity** case to the new directory.
3. Enter the **cavityHighRe** case and edit the *transportProperties* file. Since we want a Reynolds number of 100, change the kinematic viscosity to $1\text{x}10^{-3}$ m^2s^{-1}.
4. Now run this case by restarting from the solution at the end of the cavity case run. To do this we can use the option of setting the `startFrom` keyword to `latestTime` so that icoFoam takes as its initial data the values stored in the directory corresponding to the most recent time, i.e. 0.5. The `endTime` should be set to `2`. Ensure the output is saved in a *logRun* file.
5. View the log file and find out at approximately what time the initial residuals are close to $1\text{x}10^{-6}$.
6. Copy the *Residuals* file across and plot residuals using gnuplot.
7. Save the results into an EnSight file and load them in EnSight 9.2.
8. Colour the Clip_plane by velocity and project streamlines onto it. What do you notice?
9. Make a new directory and label it **cavityFineHighRe**.
10. Copy the constant and system directory from the **cavityHighRe** case into the new directory
11. Increase the mesh resolution in the new case to contain 80 cells in the x and y directions. Execute the `blockMesh` application.
12. mapFields from the **cavityFine** case to the current directory.
13. Run `icoFoam` saving output to a log file, and view its contents. What do you notice? If you are happy with the results move onto the next section.

Note: you can clean the case by removing time directories and log files with the following command

```
rm -r 0.* 1* 2* log*
```

14. Make any adjustments you see necessary in the control dictionary to ensure numerical and temporal accuracy.
15. **Extension Task:** Try setting up a case with a Reynolds number of 1000 or any number you wish. You may need to increase the mesh resolution significantly and decrease the time step to ensure temporal and numerical stability.
16. **Extension Task:** What time step and cell size do you think would be required to solve fluid flow with a Reynolds number of 10,000. Is this even possible?

7.1 Turbulent Flow

As we increase the Reynolds number, the flow eventually becomes more turbulent and the laminar fluid flow solver is not able to capture the fluid flow behaviour. The `pisoFoam` solver is better suited to solving transient incompressible flows, as turbulence models are used to predict the flow. In this section, we will solve the lid driven cavity flow using the k-epsilon model for a Reynolds number of 10,000.

7.2 Pre-processing

1. Copy the cavity case which is stored on the cluster and place it in your **run** directory using the following command.

   ```
   cp -r $FOAM_TUTORIALS/incompressible/pisoFoam/ras/cavity $FOAM_RUN
   ```

2. Double the size of the mesh resolution and generate the mesh using the standard method.
3. The mesh has been generated using uniform grading. This is because wall functions have been used when resolving k, the turbulent kinetic energy, and ϵ, the turbulent disserpation rate. The flow in the near wall behaviour has been modelled, rather than being resolved.
4. View the contents of the *0/nut* file. Note that the internal field has been uniformly set to zero and that a *nutWallFunction* has been specified in the turbulent viscosity field ν_t.
5. k and ϵ have been set up similarly. The initial values for k and ϵ have been set using the velocity U and a turbulent length scale L. k and ϵ are defined in terms of the following parameters.

$$k = \frac{3}{2}(UI)^2, \qquad \epsilon = \frac{C_\mu^{0.75} k^{1.5}}{l}. \tag{7.1}$$

   where $I = 0.16Re^{-1/8}$ the turbulence intensity.
6. In addition, we have a field R. This is the Reynolds stress terms and a generic wall function *kqRWallFunction* has been used to model R.
7. Turbulence modelling includes a range of methods: laminar; RAS or LES are provided in OpenFOAM. The choice of turbulence modelling method is through the `simulationType` keyword in *turbulenceProperties* dictionary. View this file.

With `RASModel` selected, a *RASProperties* file is required in the same directory.

8. View the *RASProperties* file. It can be seen that the `kEpsilon` model has been chosen in the `RASModel` sub-dictionary. This implies that default parameters associated to the k-epsilon model have been chosen. The `turbulence` has been switched on, and `printCoeffs` is also on so that these parameters are written to the output when the `pisoFoam` solver is run.
9. It is of interest to see the fluid flow behaviour for large Reynolds numbers = $10,000$. For this we require a kinematic viscosity of 10^{-5}.
10. Set the start time from 0, the end time at 20 and the time step to 0.005.

7.3 Running the case

1. Execute pisoFoam by entering the case directory and typing
   ```
   pisoFoam > logRun
   ```
   into the terminal. Monitor the convergence of the solution. View the results at consecutive time steps as the solution progresses to see if the solution converges to a steady-state or perhaps reaches some periodically oscillating state. In the latter case, convergence may never occur but this does not mean the results are inaccurate.
2. Visualise the output. What would you do to improve the quality of the simulation results?

8 Multiphase flow by Dr Hassan Hemida

8.1 Description of the problem

The problem considers a bottle initially filled with air being filled with water as shown in Fig. 6. For simplicity we will consider a two dimensional case by assuming that the geometry has an infinite length in the z-direction. The domain consists of two regions: an inlet chamber and a bottle. The dimensions are also in Fig. 6

Initially, the bottle is filled with air and the inlet chamber is filled with water. At time zero, the water will move to fill the bottle. The inlet velocity of the water at the inlet section to the inlet chamber is 0.1 ms^{-1}. The time needed for filling the bottle with water is about 8 seconds.

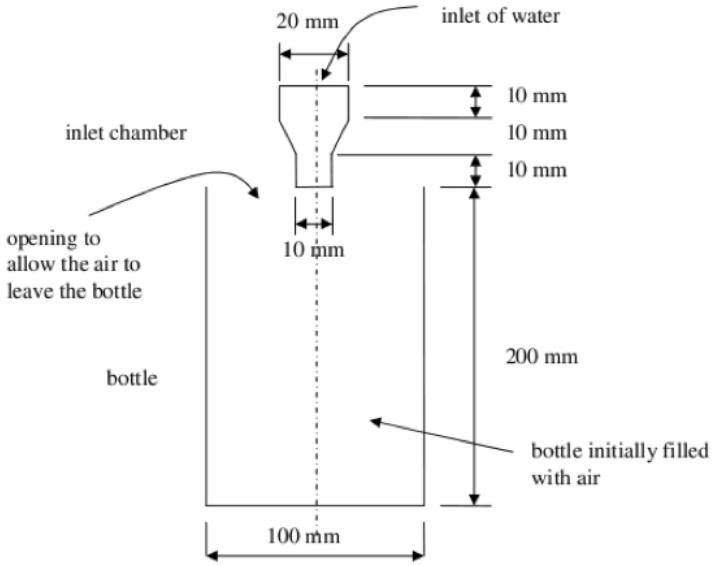


Figure 6: Schematic of the problem.

8.2 interFoam solver

The case is a free surface problem, where a sharp interface between the two fluids exists. The solver that will be used is called *interFoam*. It is a solver for 2 incompressible fluids capturing the interface using a Volume Of Fluid (VOF) method. If no turbulence model is specified. Solution may be viewed as a Direct Numerical Simulation (DNS) if the mesh is sufficiently fine enough, otherwise it is a laminar solver. Alternatively a Reynolds-Averaged Navier-Stokes (RANS) turbulence model or a Large-Eddy Simulation (LES) sub-grid scale turbulence model can be specified.

8.3 Pre-processing

1. The problem described above is similar to the existing tutorial supplied with OpenFOAM distributers and is called damBreak A copy of this tutorial should be copied to the run directory using the following commands.

   ```
   run
   mkdir bottle
   cd bottle
   cp -r $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak .
   ```

2. Rename the case directory to be called laminarFilling.
3. This case has a similar setup to the cavity case. However, they contain some additional files which will be discussed below. In the **constant** directory.
   (a) As in previous cases, a description of the fluid properties are provided in the *transportProperties* file. Since this is a multiphase problem, a description of the two different fluids is given; namely water and air.

i. The `transportModel` sub-dictionary for the two fluids are set to Newtonian

ii. Information relating to the viscosity and density is provided by `nu` an `rho` respectively.

iii. The `CrossPowerLawCoeffs` and `BirdCarreauCoeffs` are used when fluids are non-newtonian.

iv. `sigma` relates to the surface tension coefficient between the two fluids.

(b) The file named *g* sets the value and direction of the gravitational acceleration in the field. Here the gravitational acceleration has a value of 9.81 ms^{-2} in the negative direction of y-axis.

(c) *dynamicMeshDict* specifies if the mesh is dynamic or static. Here the mesh is static.

(d) *turbulenceProperties* As previously mentioned provides information about the turbulence models used. In this example, we wish to run without turbulence modelling, so we set it to laminar.

4. The Mesh is currently set up for modelling a dam break problem, we will now edit the *blockMeshDict* file to suit our case. The geometry is divided into 6 blocks as shown in Fig. 7

5. The *blockMeshDict* file should be changed as follows:

```
// * * * * * * * * * * blockMeshDict * * * * * * * * * * * //

convertToMeters     0.001

vertices
(
    (-10 230 0)
    (10 230 0)
    (10 220 0)
    (-10 220 0)
    (5 210 0)
    (-5 210 0)
    (5 200 0)
    (-5 200 0)
    (5 0 0)
    (-5 0 0)
    (50 200 0)
    (50 0 0)
    (-50 200 0)
    (-50 0 0)
    (-10 230 5)
    (10 230 5)
    (10 220 5)
    (-10 220 5)
    (5 210 5)
    (-5 210 5)
    (5 200 5)
    (-5 200 5)
    (5 0 5)
    (-5 0 5)
    (50 200 5)
    (50 0 5)
    (-50 200 5)
    (-50 0 5)
```

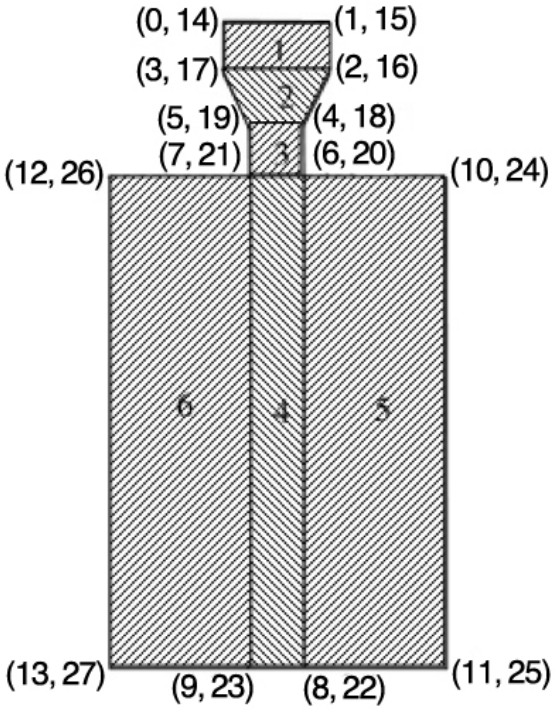


Figure 7: Block structure.

```
);

blocks
(
    hex (3 2 1 0 17 16 15 14) (10 5 1) simpleGrading (1 1 1)
    hex (5 4 2 3 19 18 16 17) (10 5 1) simpleGrading (1 1 1)
    hex (7 6 4 5 21 20 18 19) (10 5 1) simpleGrading (1 1 1)
    hex (9 8 6 7 23 22 20 21) (10 100 1) simpleGrading (1 1 1)
    hex (8 11 10 6 22 25 24 20) (40 100 1) simpleGrading (1 1 1)
    hex (13 9 7 12 27 23 21 26) (40 100 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 14 15 1)
        );
    }

    inletWall
    {
```

```
        type wall
        faces
        (
            (0 3 17 14)
            (3 5 19 17)
            (5 7 21 19)
            (1 15 16 2)
            (2 16 18 4)
            (4 18 20 6)
        );
    }

    bottleWall
    {

        type wall;
        (
            (10 24 25 11)
            (11 25 22 8)
            (8 22 23 9)
            (9 23 27 13)
            (13 27 26 12)
        );
    }

    atmosphere
    {

        type patch;
        (
            (6 20 24 10)
            (12 26 21 7)
        );
    }

    frontAndBack
    {
        type empty;
        (
            (0 1 2 3)
            (3 2 4 5)
            (5 4 6 7)
            (6 10 11 8)
            (6 8 9 7)
            (7 9 13 12)
            (14 17 16 15)
            (17 19 18 16)
            (19 21 20 18)
            (20 22 25 24)
            (20 21 23 22)
            (21 26 27 23)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************* //
```

6. We have five patches, `inlet, inletWall, bottleWall, atmosphere` and `frontAndBack` specifying the inlet of the water, the wall of the inlet pipe, the

wall of the bottle, the opening at the top of the bottle into the atmosphere and the front and back faces, respectively. Note the rotation between the different vertices to specify the corresponding block. It should be consistent for all the blocks.

7. In the system directory we have 2 additional files the *setFieldsDict* and the *decomposeParDict*.
   (a) The *setFieldsDict* specifies a non-uniform initial condition for the phase fraction alpha1 where

$$\text{alpha1} = \begin{cases} 0, & \text{for the gas phase} \\ 1, & \text{for the liquid phase} \end{cases} \tag{8.1}$$

      i. Initially the bottle is filled with air so the `defaultFieldValues` for `alpha1` is set to `0`.
      ii. The `defaultFieldValues` sets the default value of the fields, i.e. the value the field takes unless specified otherwise in the regions sub-dictionary. That sub-dictionary contains a list of sub-dictionaries containing fieldValues that override the defaults in a specified region. Here, `boxToCell` creates a bounding box within a vector minimum and maximum to define the set of cells of the liquid region. The phase fraction alpha1 is defined as 1 in this region. The `boxToCell` location should be modified to allow this. Change the location to

```
box (-0.01 0.2 -1) (0.01 0.23 1);
```

      Note that the bounding box might be large and extended outside the domain.

   (b) The *decomposeParDict* is a file that sets the case up so that it can be run on multiple processors. Open this file.
      i. `numberOfSubdomains` indicates the number of processors we wish to run on,
      ii. `method` indicates the method of domain decomposition, `simple` is selected and so the domain will be split into an equal number of sections as described in the `simpleCoeffs` subdirectory.
      iii. `simpleCoeffs` specifies that the x-coordinate will be split into 2 parts and the y-coordinate will be split in 2 part, giving a total of 4 parts altogether.

   (c) In the *controlDict* file, change the `startFrom` subdirectory to `latestTime` and change `writeInterval` to 0.1.

8. We also need to change the `boundaryField` patches in the **0** directory.
   (a) Open the *alpha1.org* file and make the following changes:

```
// * * * * * * * * * * alhpa1.org * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];
internalField     uniform 0;
```

```
boundaryField
{
    inlet
    {
        type     fixedValue;
        value     uniform 1;
    }

    inletWall
    {
        type     zeroGradient;
    }

    bottleWall
    {
        type     zeroGradient;
    }

    atmosphere
    {
        type     inletOutlet;
        inletValue     uniform 0;
        value     uniform 0;
    }

    frontAndBack
    {
        type     empty;
    }

};

// ************************************************************* //
```

(b) Copy the *alpha1.org* file into another file named *alpha1*. It will become clear why we have done this later.

(c) In a similar way change the velocity and dynamic pressure dictionaries. It is worth thinking about the input velocity and direction that the fluid is traveling in.

8.4 Running the case

1. Generate the mesh in the standard way saving the output into a file labelled *logBlock*
2. View the *logBlock* file to check for any errors. You may wish to view the mesh at this point.
3. If we start the simulation with the existing alpha1 field, the initial field will be filled with air only. To initialise the solution with water in the inlet pipe and air in the bottle we need to run the `setFields` utility on our case.

   ```
   setFields > logFields
   ```

   View the *logFields* file and the *alpha1* file in the **0** directory to see what the `setFields` utility has done.
4. Before running the simulation, if you haven't already done so view the mesh. Next, create a clip plane and colour the field by alpha1, how would you describe the field?

5. This case will be run in parallel.
   (a) Run the *decomposeParDict* file using
   ```
   decomposePar > logDecom
   ```
   (b) By decomposing the case the geometry has been split into a number of directories labelled processor0, 1, 2, 3. Note that when the simulation is run time directories will be placed inside the processor directories.
6. Copy the *runInter* file into your case directory and open the file. The file is a bash script that can be submitted as a job into the Blue Bear cluster.
   (a) The first few lines tell the front node what type of job is being requested.
   (b) `cd "PBS_O_WORKDIR"` changes the directory into the working directory.
   (c) `np = $(wc -l < $PBS_NODEFILE)` calculates the number of processors requested.
   (d) `mpirun --hostfile $PBS_NODEFILE np $np`
   `interFoam parallel > logRun` executes the interFoam solver in parallel and saves the output in a logRun file.
7. Submit the job into the cluster by typing
   ```
   qsub runInter
   ```
   from the case directory. Wait until the job is running. This can be checked by typing
   ```
   qstat
   ```
   This is indicated by a R instead of a Q in the S (status) column.
8. View the output of the data as it is being generated using
   ```
   tail -f logRun
   ```
9. Monitor the residuals.
10. Since the case is currently in a decomposed state, time directories require recomposing using the following command:
    ```
    reconstructPar
    ```
11. An **EnSight** directory can now be created to post-process results.

9 Multiphase flow using turbulence modelling

In this section we will be using a similar setup to the previous case, but instead of solving laminar flow, a RANS turbulence model will be used.

9.1 Pre-processing

1. A few modification are required. To begin with copy the **damBreak** case from the **rasInterFoam** tutorial directory. using the following command.
   ```
   run
   cd bottle
   cp -r $FOAM_TUTORIALS/multiphase/interFoam/ras/damBreak .
   mv damBreak RANSFilling
   ```
2. In the **constant** directory.
   (a) Set the mesh up as in the previous case this can be done by editing the

blockMeshDict file or you can delete the current **polyMesh** directory and copy the **polyMesh** directory from the laminar case.

(b) Open the *turbulenceProperties* file and note that the simulation type has changed from `laminar` to `RASModel`

(c) An additional file labelled *RASProperties* exist. Inside this file the `kEpsilon` turbulence model has been selected with its default coefficients.

3. In the **system** directory change the control, setFields and decomposePar dictionaries as in the previous case.
4. In the **0** directory:
   (a) Set the alpha1, velocity and the dynamic pressure field as in the previous case.
   (b) Edit the patch names in the turbulent kinetic energy (k), turbulent dissipation rate (*epsilon*), turbulent viscosity (*nut*) and kinematic turbulent viscosity (*nuTilda*) files.
   (c) Based on the theory for k and *epsilon* presented earlier, set the turbulent kinetic energy fields to $3.75x10^{-5}$ and the `inlet` patch to `fixedValue`. Change turbulent dissipation rate to $9.43x10^{-7}$ and the `inlet` patch to `fixedValue`.
   (d) Change the turbulent viscosity inlet patch to type `calculated`.

9.2 Running the case

1. Generate the mesh if it has not already been created.
2. Set the fields for alpha1, the interface between fluid and water.
3. Decompose the case so that it can be run on 4 processors.
4. Run the case in parallel using the interFoam solver.
5. View and monitor the residuals.
6. Reconstruct the time directories.
7. Generate an Ensight Directory.
8. Visualise the flow fields using methods you have learnt. Compare results to the laminar case.