Exercise: OpenFOAM I

0. Warnings and advice

Download might take some time, and should be done before guidance. Running simulations in OpenFOAM can take many hours, and might require all of the capacity bestowed by a regular computer. It is a good idea to run the simulations overnight to avoid conflict with use of other programs, and also to close all other programs so the computer won't crash. For those who worry their computer will run hot, change settings on processor use (adjust percentage of maximal use to e.g. 80%). The simulation will then run a little slower.
Read the assignments thoroughly and all the way through to not miss important information about execution.

1.Download OpenFOAM for Windows

OpenFOAM is an open source program, originally developed for Linux. It is the most used CFD program today, and can save companies large sums of money because it is free. The OpenFOAM version for Windows was launched in spring 2015, by developers in Prague, CFD support, Ltd. (http://www.cfdsupport.com/about-us.html) This makes OpenFOAM even more accessible for the common user. The happy owners of a Linux PC are of course welcome to use the real deal and follow the instructions given for the dambreak tutorial in OpenFOAM's user guide.

Link for download:
http://www.cfdsupport.com/download-openfoam-for-windows.html

2.Download OpenFOAM for linux/macOS

If you have a mac or a linux pc you can download OpenFOAM from this link:
https://openfoam.org/download/
Follow the instructions given on these pages to install the program. This involves running some commands from the command window.

It is possible to borrow a windows computer from the department for the exercise. Nevertheless, you will also have to install the program on these computers. Also, there are ways to make Windows software work on Mac, the Mac users will probably have experience in this already.

3. Navigation and commands in the command window

OpenFOAM does not have a graphical user interface. This means that you have to control the program and its features from a command window and by editing files that define the different settings for your simulation. This differ from giving the settings by buttons and drag down windows in a graphical user interface. Nevertheless, it is the same settings that need to be fed to a CFD program (initial conditions, turbulence models and so on) regardless of whether the program uses a graphical user interface, or if this information have to be given directly in text files.

Any experience using command windows cmd (Windows) or Terminal (Linux) will come in handy, but is not a necessity for these exercises. All commands will be given.

Since OpenFOAM was made for Linux, the Windows version also uses a command window that answers to Linux commands, called Cygwin. Cygwin is part of the package when you download OpenFOAM from the link given above. Some of the OpenFOAM specific commands will vary, though, and typing them correctly is crucial for the commands to work. During the work with these exercises, you should learn that the tab-button is an efficient tool when working and navigating in the command window.

There is nothing you can do on your computer that cannot be done through the command window. One important thing is to navigate between folders/directories. The starting point for cmd and Terminal allows you to go wherever you want to go on your computer.

When you start up, there should be an address and a dollar sign in the window before any commands are written. If it is not there, wait for it, the computer is working on it. To view all subdirectories, simply type `"ls –al"` and press enter. To enter or study a subdirectory, type `"cd nameOfSubdirectory"`. The student may observe that every directory lists a "." and a ".." subdirectory. These are not really directories. Typing `"cd .."` navigates out of that subdirectory, into the directory where that subdirectory is listed. Typing `"cd ."`, nothing happens. Typing just `"cd"` will take the user back to the users home folder.

Simply typing "run" in the Cygwin-window will lead the user to the run directory. This is a good place to keep case-directories. The OpenFOAM exercises (homework 9 and 10) will focus on three cases: one tutorial called damBreak, which is intended as an introduction to the execution commands, one called weir overflow, where you will have a peak into basic grid generation and post-processing, and a last one, also on grid generation, but using more complex tools.

4. Running a tutorial

There are many tutorials made for OpenFOAM. It is possible to copy these into the run directory, and then alter them to make them fit the user's needs.

An OpenFOAM case always contains a 0, constant and system directory. The 0 files gives initial conditions, while the constant files contain information about constant values/physical properties, turbulence models and mesh. The system files control the discretization schemes, time step and more. It is always a good idea to make a copy of the 0 directory and call it 0.org, since some of the files in that directory will be overwritten during calculations. If anything goes wrong, the original files can be copied into 0 and the calculations started anew. OpenFOAM does not read or alter .org folders.

Breaking of a dam

User guide

There is an OpenFOAM user guide online that can be very useful for the student. The tutorial we are going to solve is taken from the user guide, chapter 2.3. Copy the link below, if there is anything in this exercise that is unclear.

cfd.direct/openfoam/user-guide/dambreak

Keep in mind that this tutorial is made for the Linux version of OpenFOAM, and that some of the commands might be different from the ones used in the Windows version. This exercise gives the right commands.

Files

The files copied from the tutorial directory in OpenFOAM by typing in the Cygwin command window:

```
run   // to enter the run directory where you place your case files
cp -r $FOAM_TUTORIALS/multiphase/interFoam/RAS/damBreak/damBreak .
```

To view and modify the files, enter the case directory use the 'write' command to open them with wordpad:

```
cd damBreak // enter the case directory
write system/blockMeshDict
```

Notepad works as well, but it looks very messy and therefore gets much more difficult to deal with. Test and see for yourselves.

Grid

The first thing we do is create a grid. The geometry is defined by blockMeshDict, the dictionary for the blocks and meshes. blockMeshDict is located in the system directory. Open and study this file, but do not alter anything. Changes in blockMeshDict are for the next case. The command `blockMesh` creates the mesh. This is a background mesh. If there were a spillway or a boat in the picture, it would need another mesh made using other commands and files, namely `snappyHexMesh` and an stl-file. More about that later.

0 files

One of the files in this folder is called alpha.water. Before the case is run, it only holds information about boundary fields. When the simulation is run, however, each cell in the mesh is given a value of alpha (content of water/air described by a number between 0 and 1, where 0 is all air, 1 is all water). That means that the alpha.water file changes for each time step.
Open all the files, study them and try to understand what each of them defines.

Constant files

Take a look at the constant files, and make sure that the value of g is in the right direction. (x y z).

System files

controlDict controls time step, write interval and so on. To save time on simulations, increase time step. To increase precision, decrease it. fvSchemes gives the discretization scheme, open it and see the chosen schemes.

Execution

The most important thing to know might be how to start over again if some mistake has been made. There are several commands for this, but use the following two:

```
foamCleanPolyMesh
```

This one does what it says – clears your mesh.

```
foamCleanTutorials
```

This removes files that have been written by other commands, such as your time step files. After using these two commands, check that the 0-files are as they should be. If not, copy the original files from the 0.org-folder.

To create the mesh type the command:

```
blockMesh
```

To fill the volume behind the dam with water, we need to set initial fields of fluids. There is no inflow of water, so the initially set field of water makes up the total water volume in this case. For more information, see the OpenFOAM User Guide 2.3 Dambreak tutorial. To decide which cells (or which field) are filled with water, we use the dictionary setFieldsDict, which is placed in the system directory. In this case setFieldsDict make setFields to change the file alpha.water. Make sure that you have a file called alpha.water and a file called alpha.water.orig in the 0 directory. If not copy one of them to the other by the cp command. Then you are ready to run the command to set the water height behind the dam by typing

```
setFields
```

Now the case is ready for running calculations. We use a solver called interFoam for a two-phase flow, and the command is conveniently named after the solver. It can be an advantage to run the calculations in the background sometimes and store the output to a log file:

```
interFoam | tee -a log.interFoam
```

The command `tee -a` reads from the standard input and write to standard output and the specified log file.

In this case, the calculation is fast enough that it is OK to run it normally:

```
interFoam
```

Calculations in OpenFOAM may take quite some time, depending on the problem and the capacity of the computer. For large problems, it is a good idea to divide the problem by the number of cores available on the computer to shorten calculation time.

```
mpiexec -np 4 interFoam -parallel > log.interFoam &
```

Here the calculation is divided on 4 cores. Some manipulation to the decomposeParDict file is needed to run this command (see the online tutorial guide). It is not expected of the student to do this now, especially since many probably only have one or two cores.

When the calculations are finished, view the results in paraView using the command

```
paraFoam
```

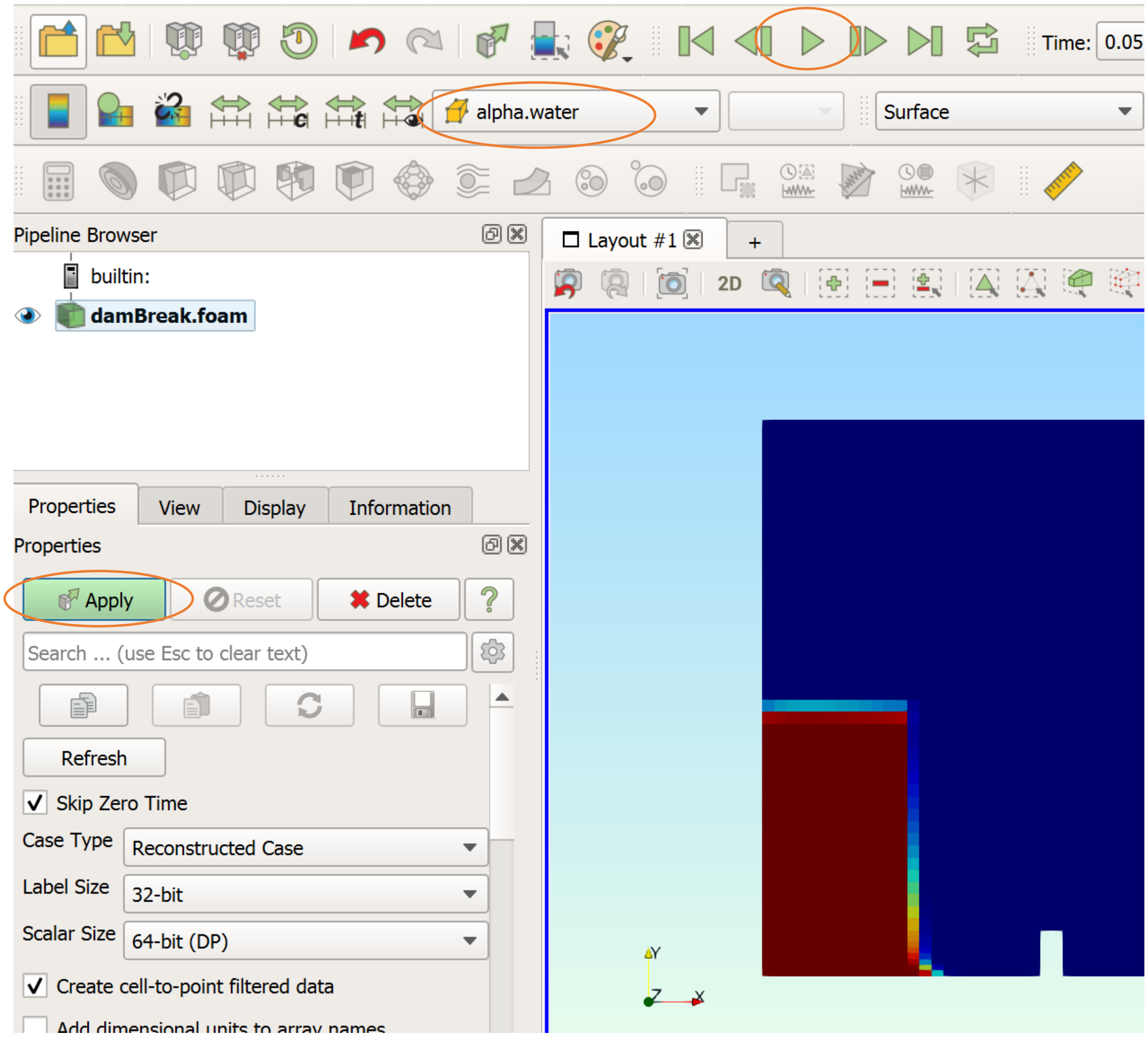
or

```
touch myfile.foam
paraview
```

Paraview will pop up by itself, looking something like the picture below. On the left, there is a menu where the desired mesh parts and volume fields can be chosen. Unless automatic application is on, the student needs to click on the Apply-button (see figure below). To the left on the top there is a menu for what field and parameter you wish to see in the graphics window.

Experiment with different combinations, but in the end choose alpha.water and surface. On top of the window there are many different buttons for perspective of view/coordinate systems. Try them all and get to know the software.

Now, press play and watch the wave. You can make it repeat itself by pressing the rightmost button on the play/stop/forward-menu.



✓ Skip Zero Time

Case Type: Reconstructed Case

Label Size: 32-bit

Scalar Size: 64-bit (DP)

✓ Create cell-to-point filtered data

Add dimensional units to array names

Reporting

Deliver paraView images from different timesteps, enough to show an outline of how the water moves.

5. A simple weir overflow

No when you are familiar with the basic concepts, we try a case where you have to implement some changes in the files to make the case run. A template for the case is available at blackboard, as the zipped folder “weiroverflow”. This is a spillway, and we are going to simulate an overflow. Download it and place the unzipped version in your run-folder.

Aim of exercise

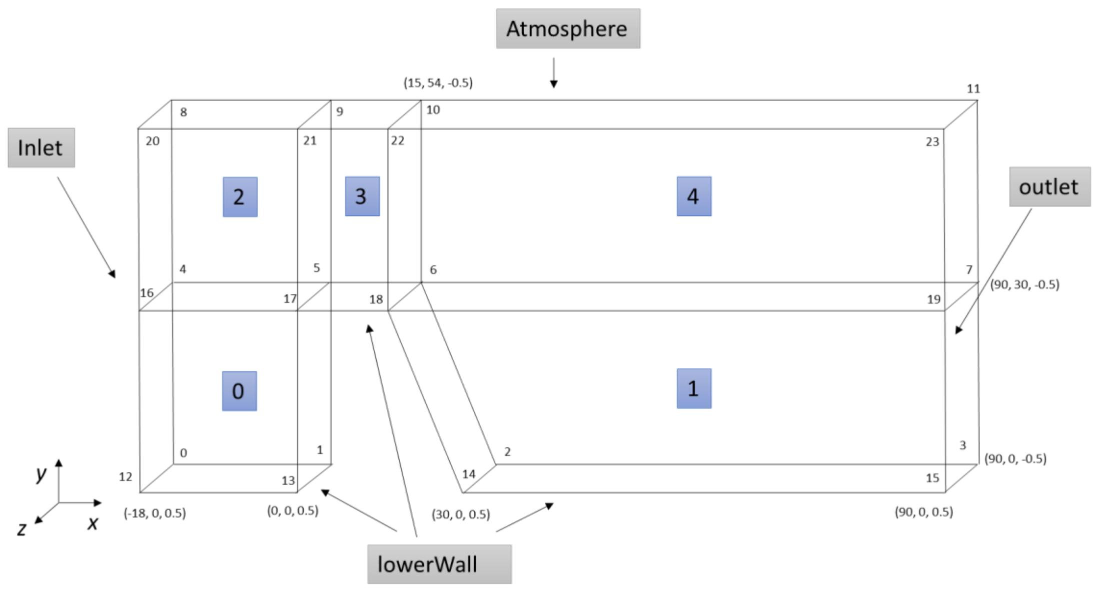
- Get familiar with grid creation with blockMesh
- Try some postprocessing utilities

Creating a mesh

A case is built in blocks. The user can choose different kinds of blocks, different numbers of blocks, different ways each vertex in a block is connected to another vertex and so forth. For deeper digging into this, check out the OpenFOAM user guide: http://cfd.direct/openfoam/user-guide/cavity/#x5-50002.1.1 .

In this case we are using hexahedral blocks, which means 8 vertices and 6 faces for every block.

The geometry we are going to create is illustrated in Figur 1.



Figur 1: Geometry to be defined in blockMeshDict

As described above the mesh is defined in *blockMeshDict,* which is located in the *system* folder. You are now going to define the geometry in this file. The file can be opened in wordpad from the Cygwin-terminal by typing

```
write system/blockMeshDict
```

Here you are supposed to define the coordinates for the vertices under the *vertices* subdirectory:

```
vertices
(
    (x1 y1 z1) //0
);
```

Each vertex gets its number based on the order in the vertices subdirectory. In Figur 1 both the vertex numbers and the coordinates are given. Use this figure to implement the vertices into your *blockMeshDict*-file.

Connect the vertices in hexahedral blocks under the *blocks* subdirectory:

```
blocks
(
    hex (0 1 5 4  12 13 17 16) (20 20 1) simpleGrading (1 1 1)
    hex (/*connecting order of vertices */) (/*no of cells in each
direction*/) simpleGrading (…)
…
);
```

In the brackets following the command `hex` you decide on the order of your coordinate system. The first two numbers give your x-direction. In Figur 1 the x-direction is defined to be in the streamwise direction, this corresponds to the direction given from vertex 0 to 1. The second and the third number inside the `hex` command will give your y-direction. In Figur 1 the y-diection is defined to be in the vertical direction. Then the next vertex in your 0-block should be vertex 5. Then you close the face by adding the last vertex, here number 4. The last direction will be your z-direction. Then you close your 0-block by adding the vertices of the other face in the same order – 12, 13, 17, 16. It is important that the face with the lowest z-value is given first.

As illustrated in Figur 1, we here choose to divide our mesh into 5 blocks. If you have used the vertex order given in Figur 1, your first block is given in the code snip above.

Create a single block at the time using the `blockMesh` command and run `paraFoam` to see whether the block is created as intended. If you type type paraFoam with the block option (does not seem to work for the windows installation, so for the windows users, type only paraFoam)

```
paraFoam -block
```

the blocks will be shown in different colors. Nevertheless, after running paraFoam with the block option you will have to run blockMesh over again to proceed. The –block command might not work with the windos installation, then just run paraFoam without the block argument to view you geometry.

The second bracket gives the partition in the different directions. If you have used the same block order as given in Figur 1, you can use the values already given in the *blockMeshDict*-file. If else you probably have to change them. Choose to show the geometry as 'Surface with edges' in paraFoam.

When you create the blocks it is important to have a careful look at the outprint given by blockMesh. The only warning it should contain at this point is a warning about patches. If there is any other warnings, have a look at the warning output and try to fix the problem. If the warning text includes something about negative cell volume, there is probably a problem with the ordering of the vertices within the block subdirectory. The order defined within hex should be according to the right and rule, going from one face to the other in the positive z-direction.

When you have created all blocks and your mesh seems to be ok, uncomment the patch subdirectory (that has been commented out). This means remove /* and */.

The last thing you have to care about regarding the mesh at this point is to prepare the boundaries. The boundary is broken into patches (regions), where each patch in the list has its name as the keyword, which is the choice of the user, although we recommend something that conveniently identifies the patch (e.g. inlet). The name is used as an identifier for setting boundary conditions.

In our case, the names are already given, and the patches should be as shown in Figur 1. What you are to do is to give in the correct order of vertices to span the desired regions.

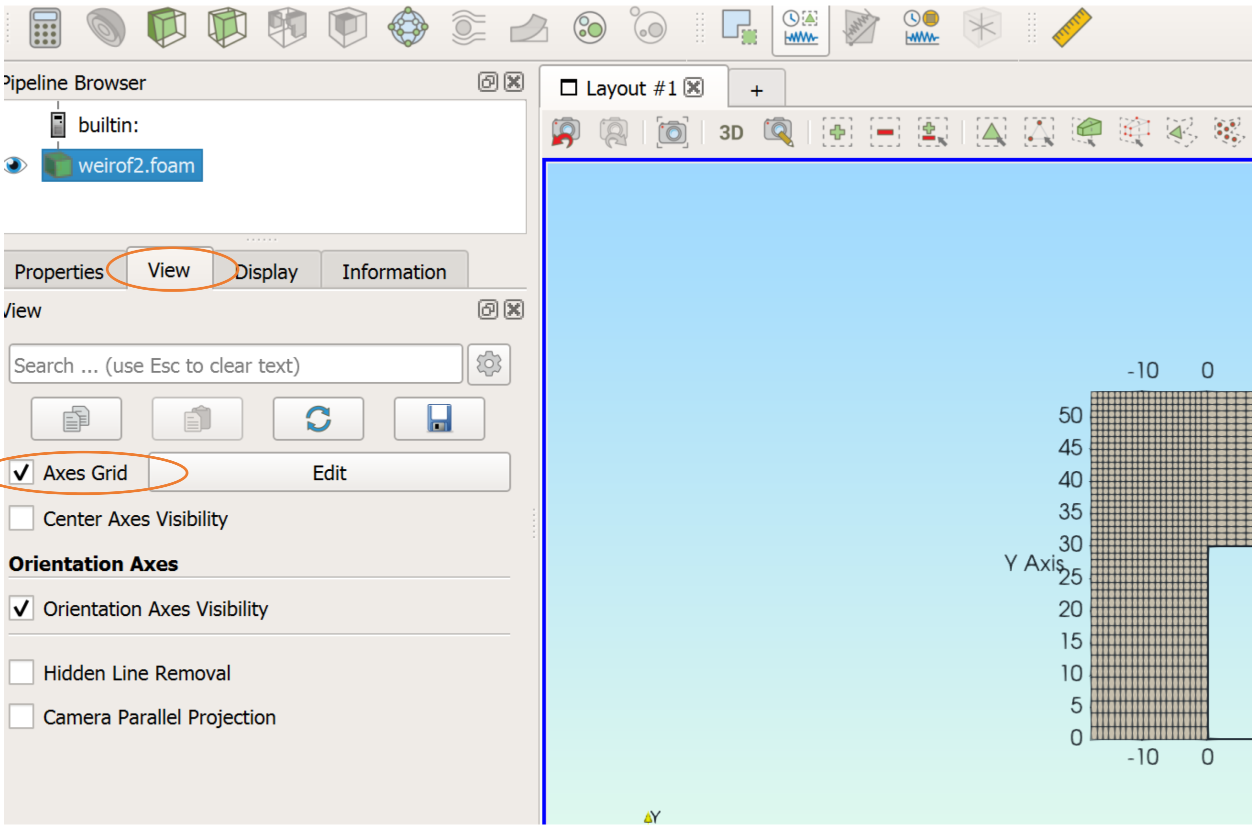
```
patches
(
    patch inlet
     (
         ()
         ()
     )
);
```

The number of faces in each patch is indicated in the *blockMeshDict* file you are given. The order in which the vertices are given must be such that, looking from inside the block and starting with any vertex, the face must be traversed in a clockwise direction to define the other vertices.

Run the `blockMesh` command and check that everything is ok. Inspect your grid in paraFoam. Check that your patches are at the right places by ticking them off and on in paraFoam. To see the patches in paraFoam you must view the mesh in 'vtkBlockColors'. If everything is ok, you are ready to proceed in the same manner as in the previous exercise.

Execution

Check whether the values given in *setFieldsDict* are reasonable for your case. Adjust if necessary. For this it might be helpful to turn on axes in paraView. This is done by checking of the box for the *Edit Axes Grid* as shown in the figure below (the geometry in the figure is not from this exercise).



Before you run the setFields command, create the *alpha.water*-file by typing

```
cp 0/alpha.water.org 0/alpha.water
```

Then run

```
setFields
```

Open paraFoam and check the alpha.water values. For the windows installation it seems to be a problem by showing the alpha.water field before the case starts running. Then just open the alpha.water file to see that it has been change by the setFields command

```
write 0/alpha.water
```

At this point this file should contain zeroes and ones.

Run the case as a background process, saving the output to a log file, by typing:

```
interFoam > log.interFoam &
```

You can then watch the progress of your simulation by watching the log file by typing

```
cat log.interFoam
```

or watch the 30 last lines in your log file by typing

```
tail –n 30 log.interFoam
```

or by looking for result files in the case directory by typing

```
ls
```

you can also watch your results in paraFoam runtime by simply launching paraFoam

```
paraFoam &
```

If you add '&' paraFoam will run as a background process.

Postprocessing – finding the free surface elevation at any point

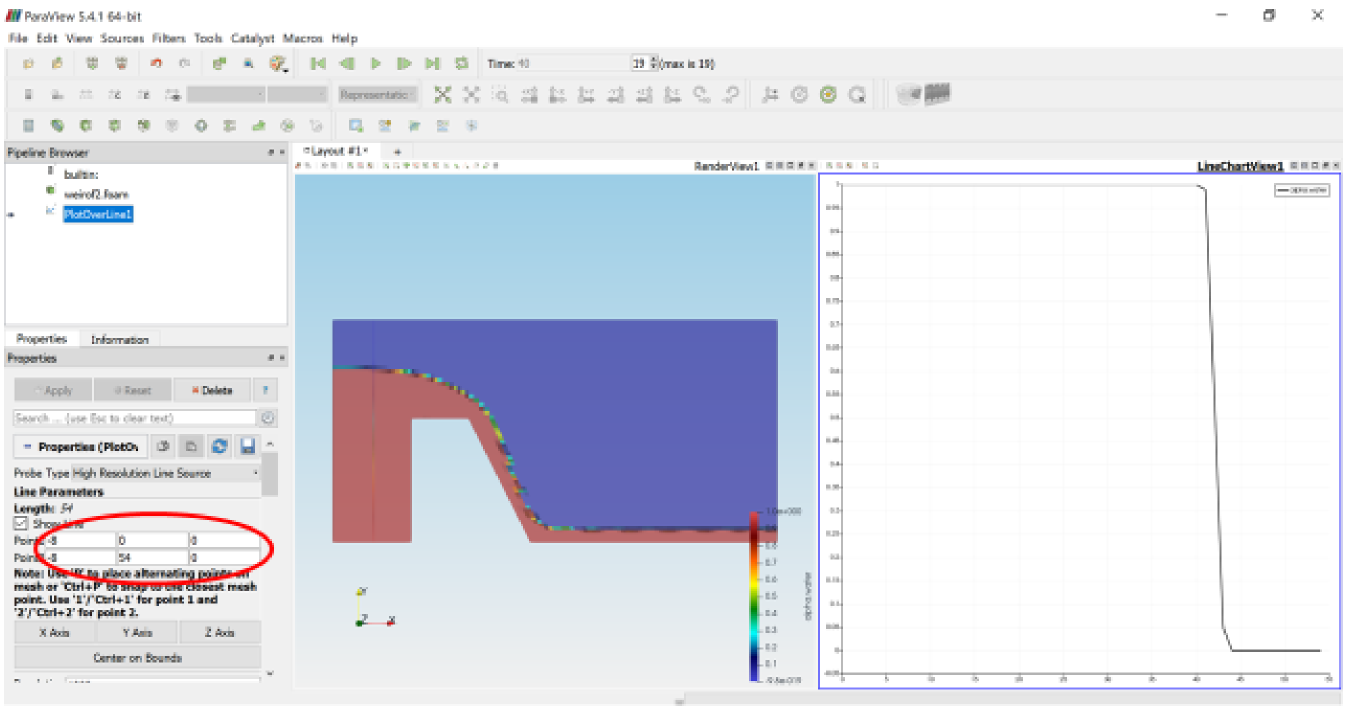
As was the case in the laboratory assignment, we are interested in finding the free surface elevation upstream the weir. There are (at least) two common ways to find the surface elevation from the volume fraction.

- Sample the volume fraction along a vertical line parallel to the y-axis (in this case) crossing the free surface, and use interpolation to find where α = 0.5.
- Integrate the volume fraction from the bottom of your domain to the atmosphere along a line parallel to the y-axis:

$$\gamma = \int_0^{y\_atm} \alpha_1 dy$$

The latter method will be used here, as it is easy implementable in Paraview using standard tools.

The first thing we need to do is to sample the volume fraction along a line. This is done by using the *Plot Over Line* utility, found under *Filters -> Data Analysis*. Create a line from the bottom of your domain to the top (see the figure below). Under the "Display" tab, deselect all other variables than alpha.water. Remember to step to the last timestep of your simulation.



Probe Type High Resolution Line Source

Line Parameters

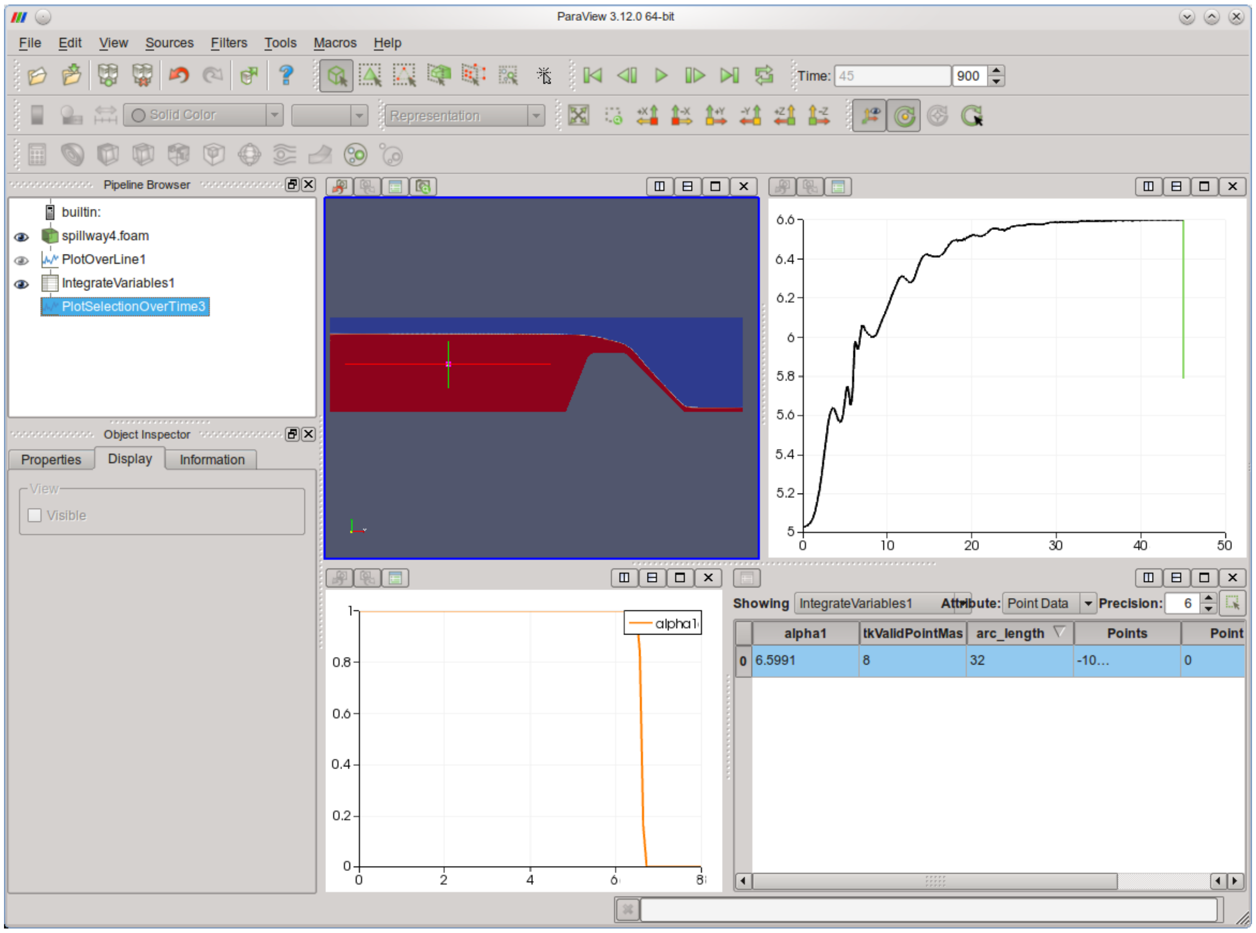
Length: 54

Show Line

Point1	-8	0	0
Point2	-8	54	0

Note: Use 'P' to place alternating points on mesh or 'Ctrl+P' to snap to the closest mesh point. Use '1'/'Ctrl+1' for point 1 and '2'/'Ctrl+2' for point 2.

Now we want to integrate the volume fraction along this line. This is done by the *Integrate Variables* tool, also found under *Filters -> Data Analysis*. Press "Apply", and you see the integrated variables. Then we want to plot this integrated variable over time, so that we can see how the free surface elevation changes during the simulation. This is done with the *Plot Selection Over Time* tool, found under *Filters -> Data Analysis*. Select the row from the table with the integrated variables, and choose "Copy active selection" found under *Filters -> Data Analysis*. Then press "Apply" and wait. This might take some time, as it will sample the volume fraction and integrate it at every timestep stored. After the process has finished, you can deselect those variables not needed in the plot. The resulting Paraview workspace now looks something like this:



Showing IntegrateVariables1 Attribute: Point Data Precision: 6

	alpha1	tkValidPointMas	arc_length	Points	Point
0	6.5991	8	32	-10...	0

From this last plot you can identify whether the surface elevation is converged. Report on your calculated surface calculation. Calculate your discharge coefficient.

Reporting

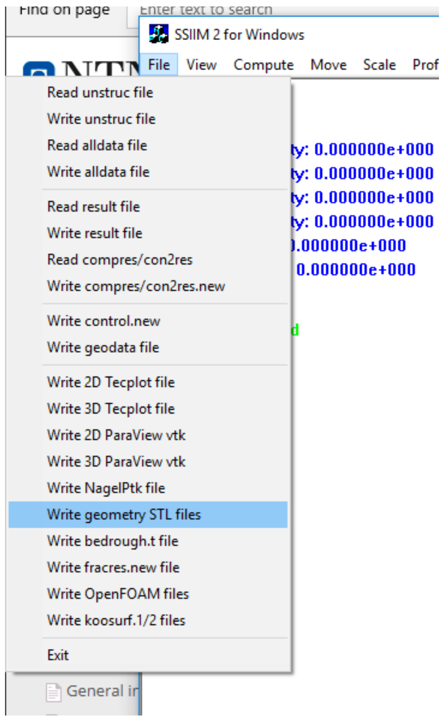
Hand in the surface elevation graph and the discharge coefficient calculations.

5. A more complicated case: creating stl file (optional)

This time we will create a model of the ogee weir used in the laboratory work. The easiest way to do this will be to create a stl-file representing the weir, and then making the geometry by the snappyHexMesh program in OpenFOAM.

OpenFOAM needs a stl-file that describes the surface of the spillway. There are several programs that can make this kind of files. This exercise explains how to make them using SSIIM, since the students are already familiar with this program.

SSIIM creates stl files from another file that must be named weir.txt. This file can be made by copying coordinates from an Excel sheet, and then pasting the data to a .txt-file. To make the 3D geometry, two planes must be defined by the coordinates in the Excel sheet. The weir will be spanned out between these planes, with the x-coordinates representing points on a line from upstream to downstream, y-coordinates representing the horizontal axis perpendicular to x, and z representing heights. They are placed in separate columns in sequence x,y,z,x,y,z, where red and blue naturally represent different planes. The coordinates are given in the Excel sheet attached to this assignment.



Note that only the coordinates in numbers, not their headers in letters, are needed in the weir.txt file. Note also that the coordinates do not only represent the surface shape of the weir, but include an extra coordinate at the end (the lower left corner of the weir) to connect the coordinates and complete the 3D shape. Do also consult the document SSIIM_intro.pdf for further assistance on this.
Once the coordinates are copied to the weir.txt file, it is time to place the weir.txt file in the same directory as SSIIM. Enter this folder by typing in the command window:

```
cd mySsiimFolder
```

Open SSIIM2 by typing in the command window:

```
cygstart -o ssiim2w64_186.exe &
```

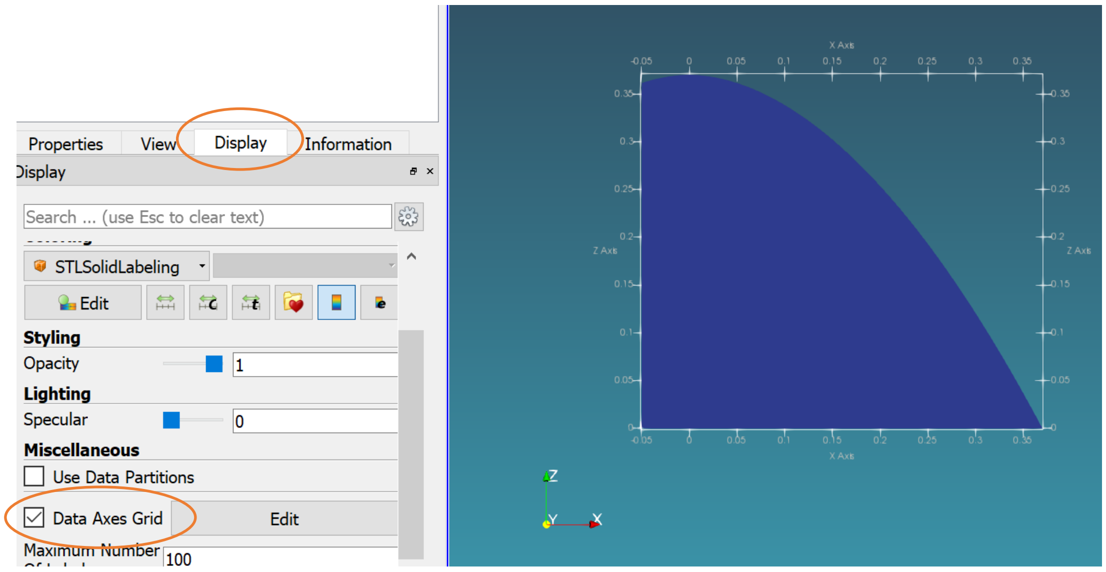
Run SSIIM, and press File -> Write geometry STL files. SSIIM will generate an STL file named weir.stl.

Open paraView to view your geometry.

```
paraview &
```

Open your .stl file by File->Open->*myStlFile.stl* and press *Apply*. View the axes by checking of the 'Edit Axes Grid' box (see figure).

In homework 10 we will use this stl-file in OpenFOAM to generate the geometry of the weir within a simple background grid.



Styling

Opacity 1

Lighting

Specular 0

Miscellaneous

☐ Use Data Partitions

☑ Data Axes Grid Edit

Maximum Number 100

Reporting

Hand in the text file weir.txt and an image of the weir.stl viewed in paraView.