CFD theory Sessions

Víctor Martínez Viol

October 7, 2016

Abstract

This paper has been written with the effort on getting comfortable when using LaTeXso please don't hate. It includes some notes taken in *Application of CFD to engineering problems.* NANDO NANDO

1 Session 4

1.1 Using Salome Meca

The first hour consisted on building the geometry of an Y-pipe with a software called *Salome Meca*. A complete guide on video (with an older version of the program) can be found in:

`http://caelinux.org/wiki/downloads/docs/Pipe2007/PipeGeom.htm`

We open Salome by going to the folder that contains it, then we right-click on the `runAppli` file, select copy, and then select 'paste filename' in a terminal window. It should append the next command to the window
'/opt/salome_meca/appli_V2016/runAppli'

When we have our geometry constructed we have to export the files produced as `.stl` files. This files are based in a bunch of coordinates refering to the points of the piece.

1.2 Constructing the mesh

We want to make the mesh using **ParaView**. We can also use directly *Salome-Meca* to mesh the pipe but its mesher is not as good as the one we're going to use. First we have to create a case. A good option is to look for a similar tutorial case as a basis for our mesh (and of course similar to the case we want to run). For the Y-pipe problem we can start with the motorbike case. To do so, create a folder and name it as Ypipe, then copy inside it the folders contained in the motorbike case. It can be done in a terminal by typing:

```
mkdir -p Ypipe
cp -r /opt/openfoam4/tutorials/incompressible/simpleFoam/motorBike/* ./Ypipe
```

Once we have the folder created the first we are going to do is to create a folder inside the constant folder containing the `.stl` files file. (Copy the .stl an paste it in /constant/trisurface and rename it as Ypipe). The next step is go to the `system` folder, inside there is a file named `BlockMeshDict`. We must modify the block geometry to put the pipe inside it. The new values for the vertices of the block are:

```
vertices
(
    (-1  -2.5  -0.5)
    (3  -2.5  -0.5)
    (3   3  -0.5)
    (-1   3  -0.5)
    (-1  -2.5  0.5)
    (3  -2.5  0.5)
    (3   3  0.5)
    (-1   3  0.5)
);
```

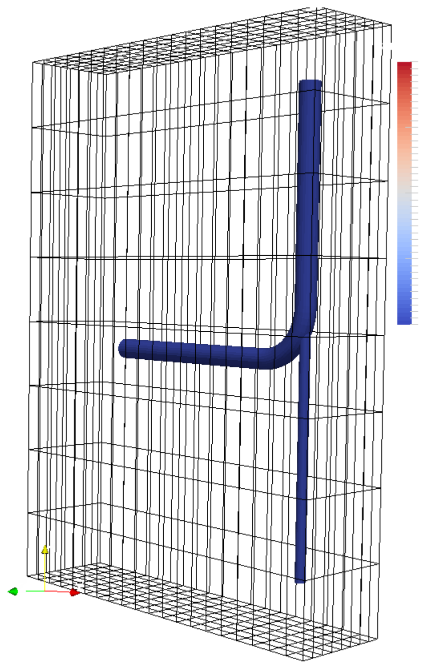

Figure 1: First mesh. Toooooo coarse

1.3 Refining the mesh

The next steps involve refining the mesh. To do so we are going to use **snappyHexMesh**(See reference). We must define the level 0 of the mesh. Open `BlockMeshDict` an edit the following lines:

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (32 64 4) simpleGrading (1 1 1)
);
```

The next step is to take `SnappyHexMesh` dictionary. Rather than create it is better to take it from `openfoam4/applications/uilities/mesh/generation/snappyhexmesh` and paste it to the `Ypipe/system` folder. Inside the file there are a lot of comments and explanations about the mesh configuration and parameters.

It is possible to define an upper level of discretization inside our blockMesh using searchablebox

SnappyHexMesh is one of the only meshers that can perform a parallel mesh of the domain (using multiple cores). The parameter `maxLocalCells` and `maxGlobalCells` define the maximum number of cells per core and the maximum number of cells on the global mesh respectively.

....

the parameter `LocationInMesh` let us decide wether we want to mesh inside or outside our geometry. the input has to be a point inside of the object

inside the folder `Ypipe/trisurface` we must have the four surfaces .stl files.

2 Session 5

2.1 Constructing the mesh II

In the last session we worked with three different `.stl` files but we can also put all of these surfaces in a single file. To do so, we must modify the one of the files in order to append all he geometries. The Pdf `pipe.pdf` which can be found in atenea explains all the process.

1. Put all the solids into a single file
2. Modify SnappyHexMesh dictionary to specify these solids into the regions section.

The process is nearly the same as in the session 4. When executing a command we can print the terminal output of the command to a log (e.g `snappyHexMesh > snappy.log` or `foamJob snappyHexMesh, this one prints also the output to the terminal`. we can clean the polymesh folder directy by tipping `foamCleanPolyMesh`. If we want to only have a single mesh stored in the constant folder we can do it by tipping `snappyHexMesh -owerwrite`

2.2 Simulating the case

A good advice before simulating a case is to make a copy of the 0 folder in order to be able to reset the case.

Now we are going to edit the `0` inside the 0.orig folder to put our boundary conditions. Inside the boundary section of this file we have to modify the code to appear as following. inlet1 0 -1 0 inlet2 -10 0 0

```
internalField uniform(0 0 0);

.... %delete all the include lines under initialConditions

patchName
{
        type            fixedValue;
        value           uniform(Ux Uy Uz);
}
```

the `inletOutlet;` option in the type field means that in the oultet when the flow is going out of the domain the boundary condition is Zero Gradient avoiding the flow to returning inside of the domain.

fixedValue change it to uniform(0 0 0)

2.3 Constructing the mesh

We want to make the mesh using **ParaView**. We can also use directly *Salome-Meca* to mesh the pipe but its mesher is not as good as the one we're going to use. First we have to create a case. A good option is to look for a similar tutorial case as a basis for our mesh (and of course similar to the case we want to run). For the Y-pipe problem we can start with the motorbike case. To do so, create a folder and name it as Ypipe, then copy inside it the folders contained in the motorbike case. It can be done in a terminal by typing:

```
mkdir -p Ypipe
cp -r /opt/openfoam4/tutorials/incompressible/simpleFoam/motorBike/* ./Ypipe
```

Once we have the folder created the first we are going to do is to create a folder inside the constant folder containing the `.stl` files file. (Copy the .stl an paste it in /constant/trisurface and rename it as Ypipe). The next step is go to the `system` folder, inside there is a file named `BlockMeshDict`. We must modify the block geometry to put the pipe inside it. The new values for the vertices of the block are:

```
vertices
(
    (-1  -2.5  -0.5)
    (3  -2.5  -0.5)
    (3   3  -0.5)
    (-1   3  -0.5)
    (-1  -2.5  0.5)
    (3  -2.5  0.5)
    (3   3  0.5)
    (-1   3  0.5)
);
```

We have to achieve a cells as regular as posible.

2.4 Refining the mesh

The next steps involve refining the mesh. To do so we are going to use **snappyHexMesh**(See reference). We must define the level 0 of the mesh. Open `BlockMeshDict` an edit the following lines:

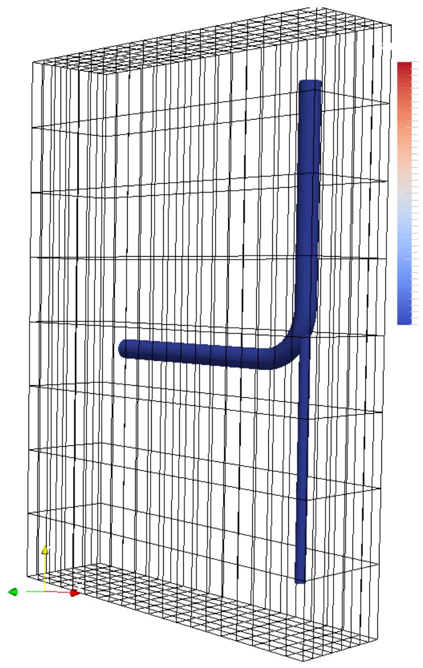

Figure 2: First mesh. Toooooo coarse

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (32 64 4) simpleGrading (1 1 1)
);
```

The next step is to take `SnappyHexMesh` dictionary. Rather than create it is better to take it from `openfoam4/applications/uilities/mesh/generation/snappyhexmesh` and paste it to the `Ypipe/system` folder. Inside the file there are a lot of comments and explanations about the mesh configuration and parameters.

It is possible to define an upper level of discretization inside our blockMesh using searchablebox

SnappyHexMesh is one of the only meshers that can perform a parallel mesh of the domain (using multiple cores). The parameter `maxLocalCells` and `maxGlobalCells` define the maximum number of cells per core and the maximum number of cells on the global mesh respectively.

....

the parameter `LocationInMesh` let us decide wether we want to mesh inside or outside our geometry. the input has to be a point inside of the object

inside the folder `Ypipe/trisurface` we must have the four surfaces .stl files.

References