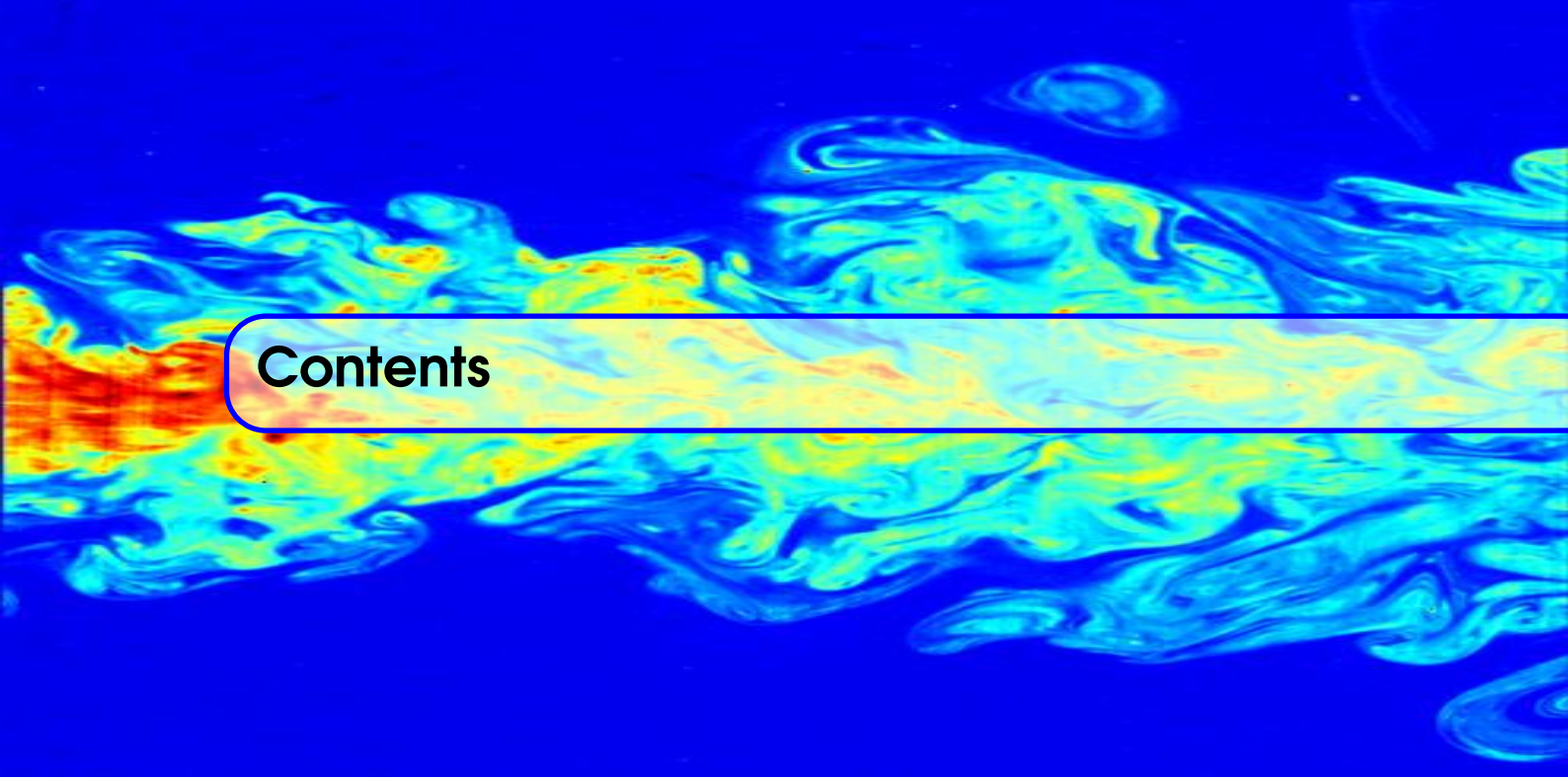
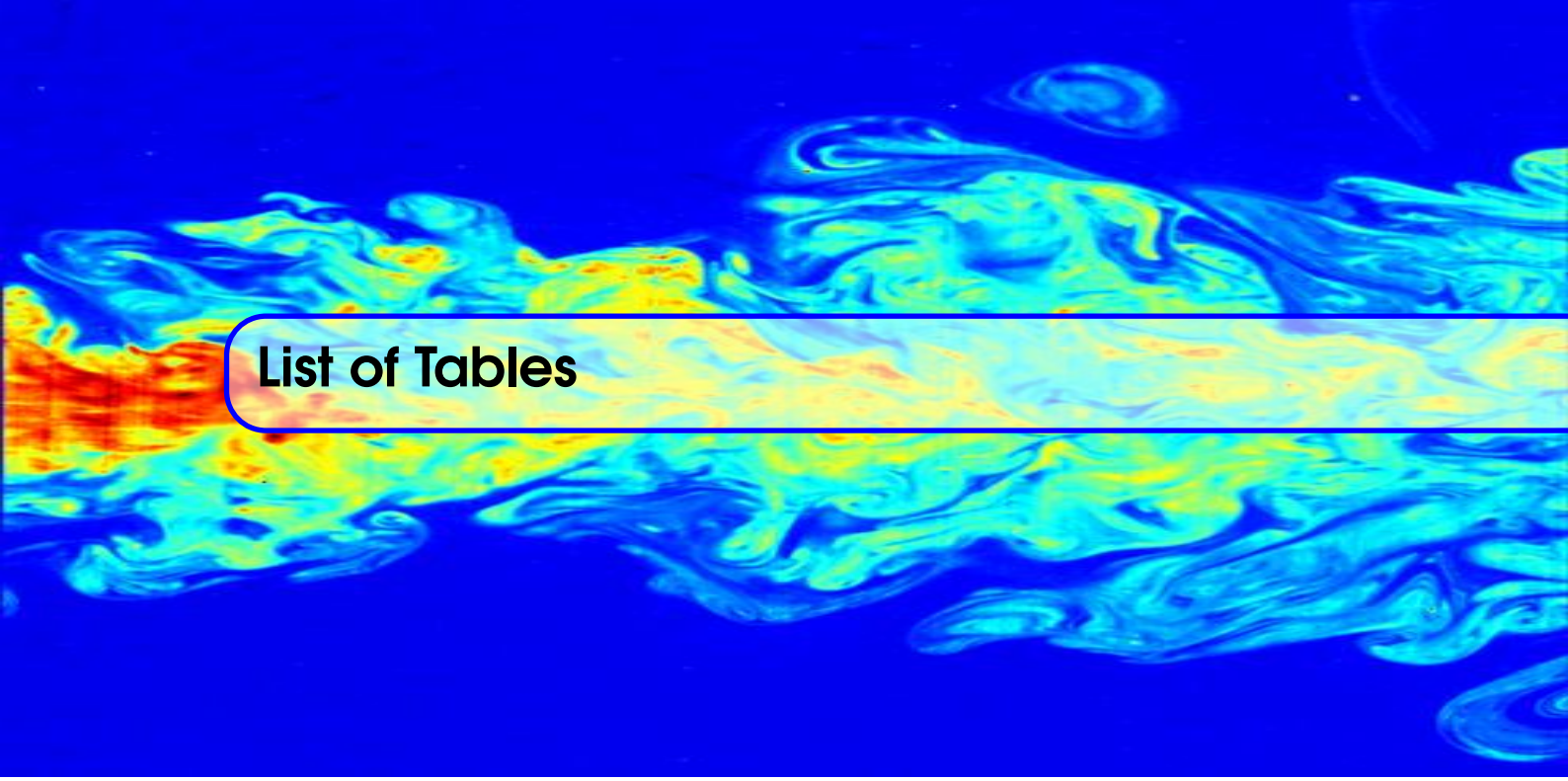
Getting started with CFD packages

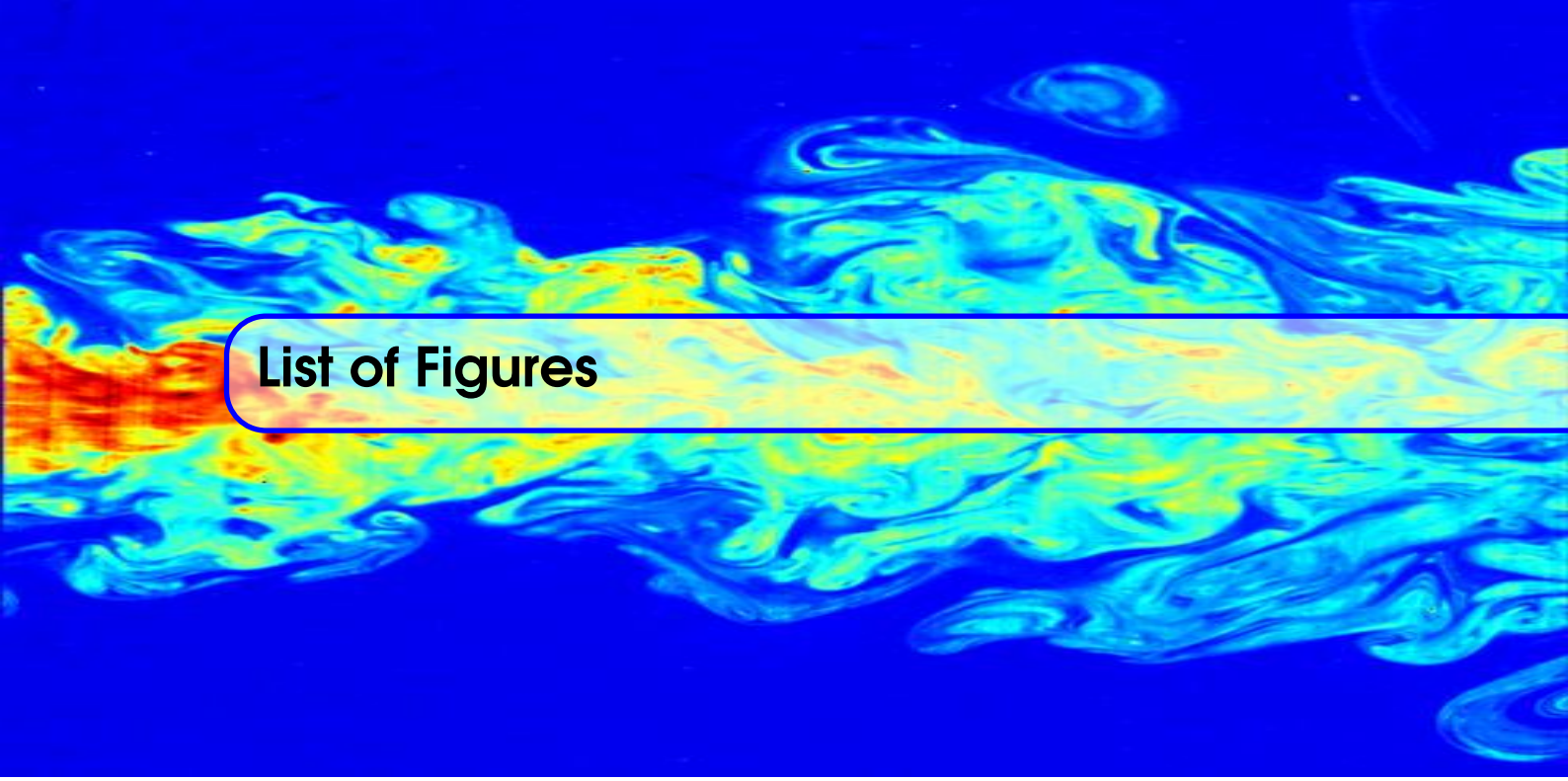
OpenFOAM® and FLUENT®



Contents

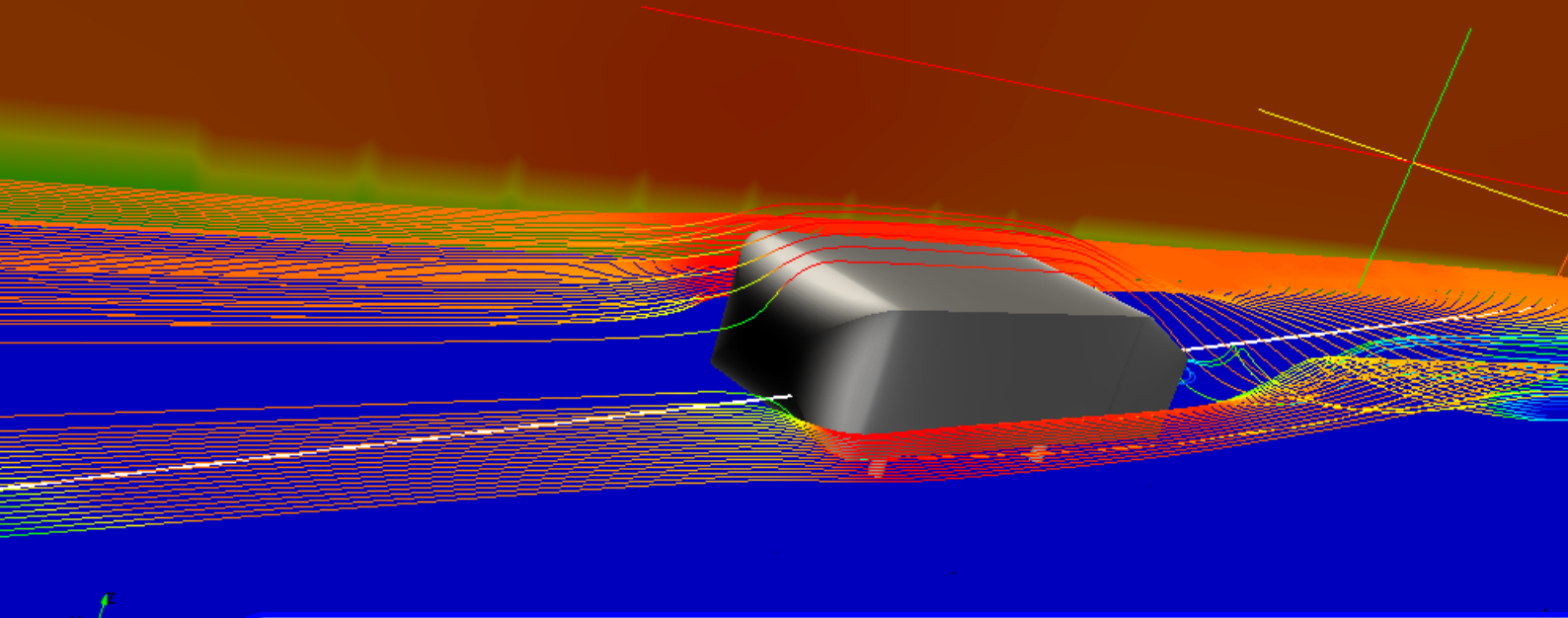


List of Tables



List of Figures

I Automotive



1. Ahmed Body

1.1 CREO®

Presented in this section a guidance through exporting the so-called **Ahmed model** geometry from CREO®package into separate patches and in `.stl` format. Assuming that most of the student here at QMUL are well trained in using CREO®in sketching complex shapes. There might be an alternative way to carry out any of the following tasks in CREO®. Nevertheless, this can give you an overview of how I managed to prepare the geometry for my project. The following steps were carried out in CREO®2.0. The following steps are compatible with other CREO®versions (i.e. Student version and Pro E), the only difference is the GUI (graphical user interface).

1.1.1 Geometry

1. Launch CREO®parametric. ☐
2. File → New → Part→ OK. ☐
3. Sketch the part with the correct dimensions as in figure. ☐
4. Extrude the part as in figure 1.3. ☐
5. Round the part as in figure1.4. ☐
6. Save as .prt. ☐
7. Repeat steps 2, 3, 4 and 6 for the middle, rear-end and stilts as shown in figure 1.7. ☐
8. File → New → Assembly → Name. as shown in figure 1.8. ☐
9. Click on assemble → front part and repeat this step for the rest as in figures[1] ☐
10. File → save as →type stereolithography (*.stl), as shown in figure 1.11. ☐
11. Export one patch at a time; Click on include → the stilts → ASCII → Reduce the step size and the chord height as shown in figure 1.12. ☐
12. Repeat the previous step for the other parts. ☐
13. Check in every .stl file that the 1st line starts with "`solid partName`" and the last line ends with "`endsolid partName`". ☐
14. Merge all the .stl files into one; by writing the following command in a terminal: ☐

[1]Importing the stilts is challenging, I did it this way because I could not find an alternative way to export the stl as multi-patches. The importance of having the stilts as a separate patch will be seen later.

```
cat front.stl middle.stl rearEnd.stl stilts.stl > ahmedBody25.stl
```

15. Copy `ahmedBody25.stl` into an OpenFOAM® case folder *See table 1.1.* □
16. Convert the dimensions of the geometry file from (**mm**) to (**m**) using the following command: □

```
surfaceTransformPoints -scale '(0.001 0.001 0.001)' ahmedBody25.stl
    ahmed-meters.stl
```

17. Check the geometry file for any illegal triangles and if there is a hole, by typing the following command in a terminal: □

```
surfaceCheck constant/triSurface/ahmed-meters.stl
```

In case of any failure in the geometry file; you can fix and repair it using Meshlab®. This open-source software can be downloaded from here `https://sourceforge.net/projects/meshlab/` and the tutorials for treating the failures in the geometry can be found here `http://tinyurl.com/ou37p5w`, `http://tinyurl.com/nmz7tfc` and `http://tinyurl.com/pnrssrw`.

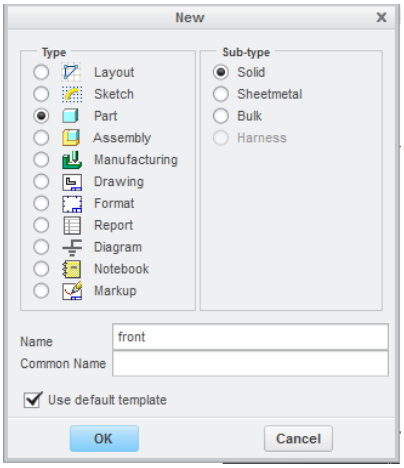


Figure 1.1: Step 2, defining a new part.

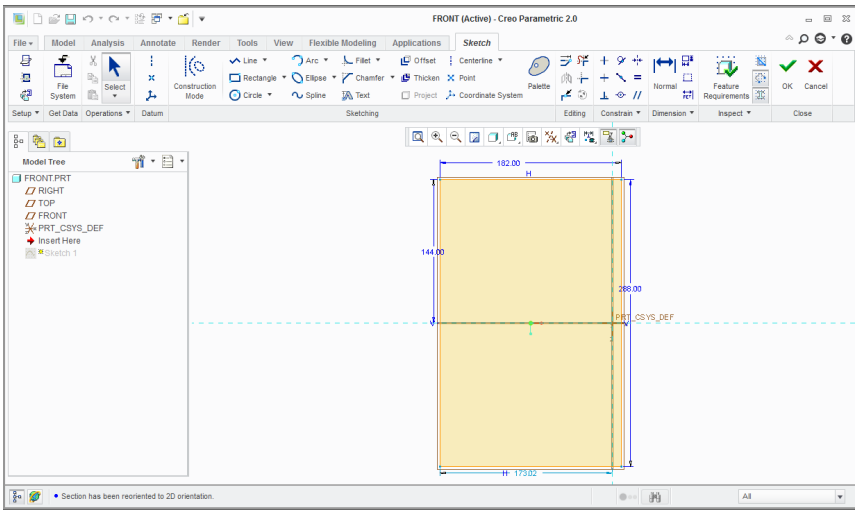


Figure 1.2: Step 3, sketching the front part,.

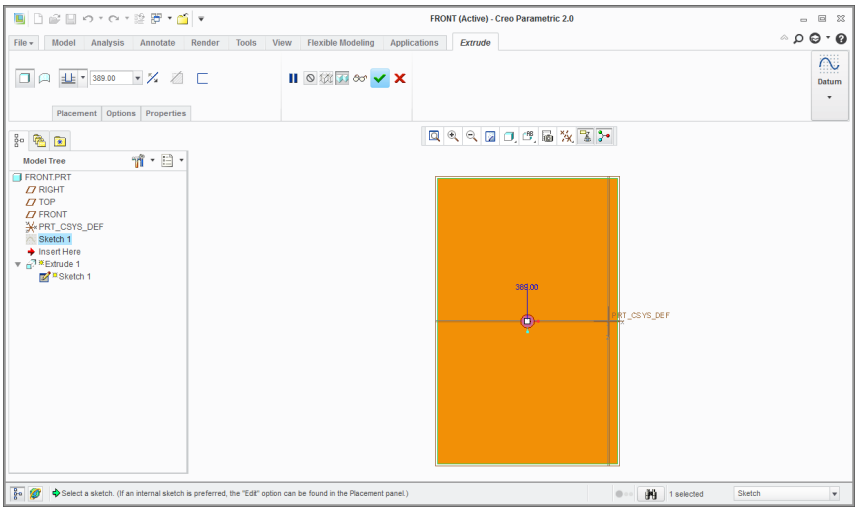


Figure 1.3: Step 4, extruding the front part.

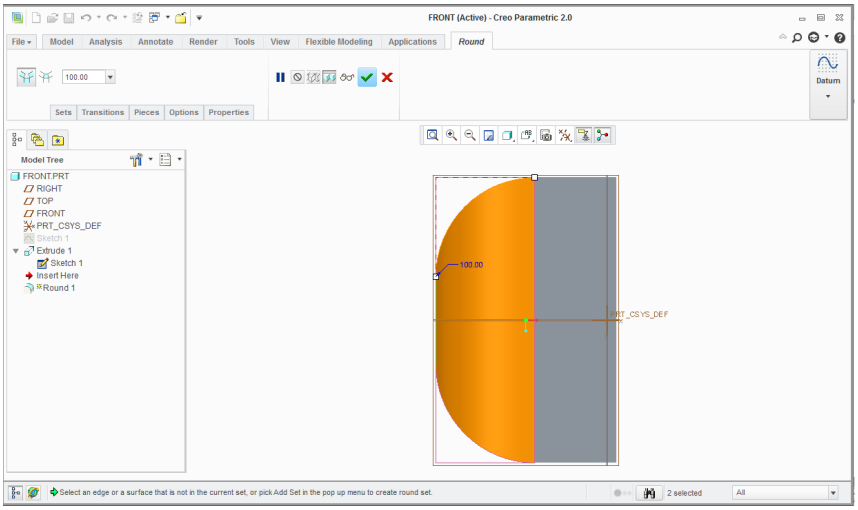


Figure 1.4: Step 5, rounding the nose.

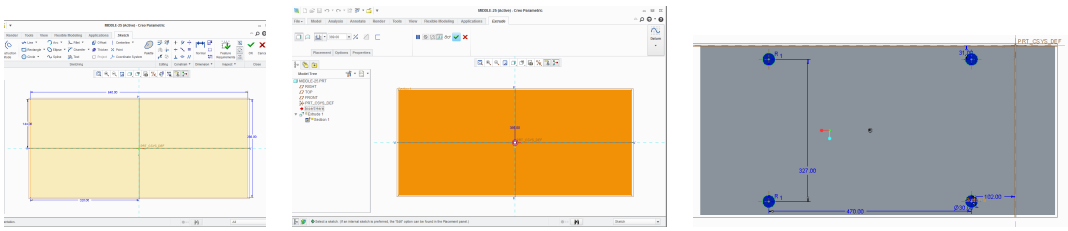


Figure 1.5: Step 7, sketching (left) and extruding the middle part (middle). Then, drawing the zones to accommodate the stilts and colouring it as blue.

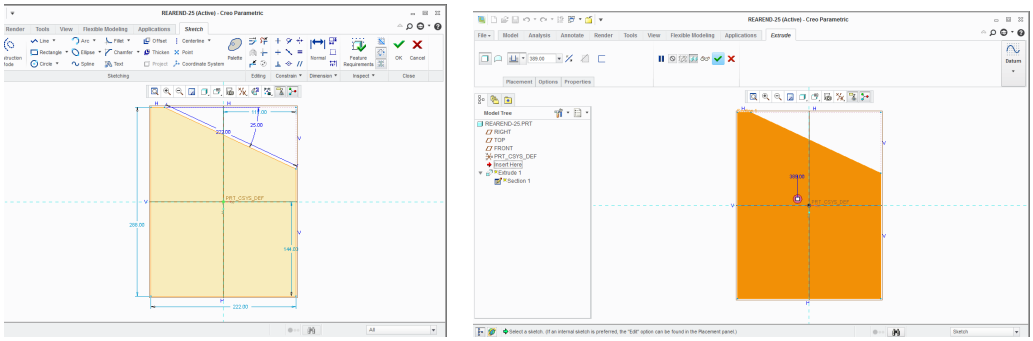


Figure 1.6: Step 7, sketching and extruding the rear part.

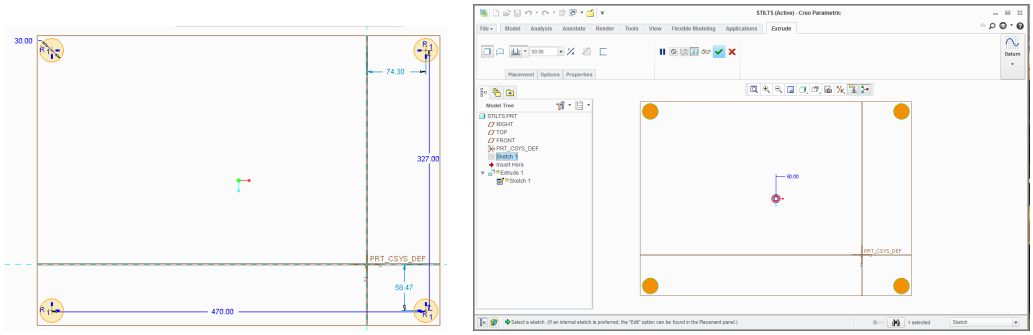


Figure 1.7: Step 7, sketching and extruding the stilts.

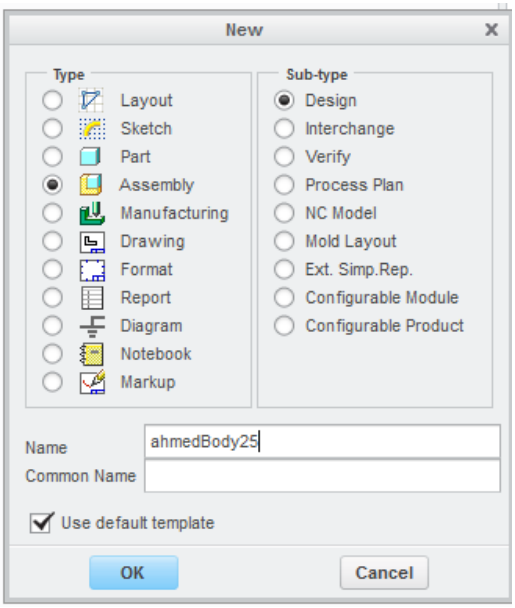


Figure 1.8: Step 8, starting an assembly mode

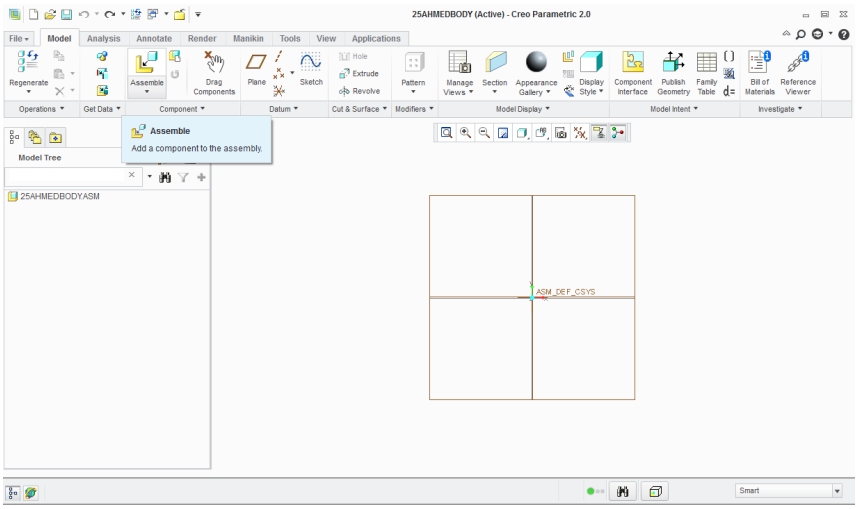


Figure 1.9: Step 9, Importing the .prt files by clicking on the assemble icon.

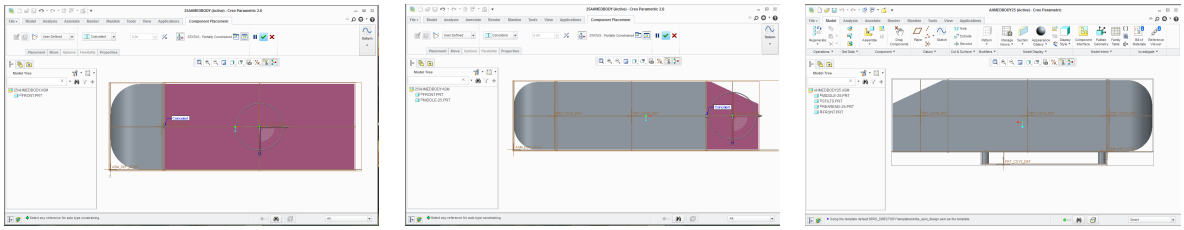


Figure 1.10: Step 9, Combining all the parts together.

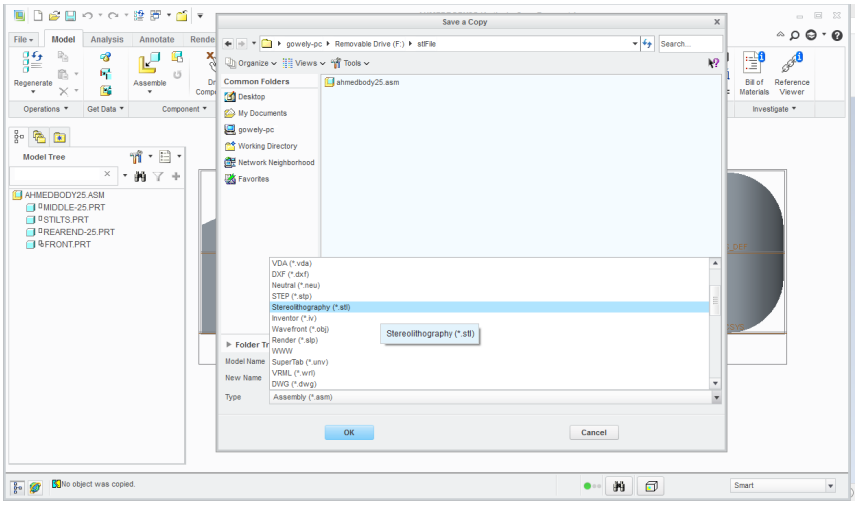


Figure 1.11: Step 10, Saving as .stl

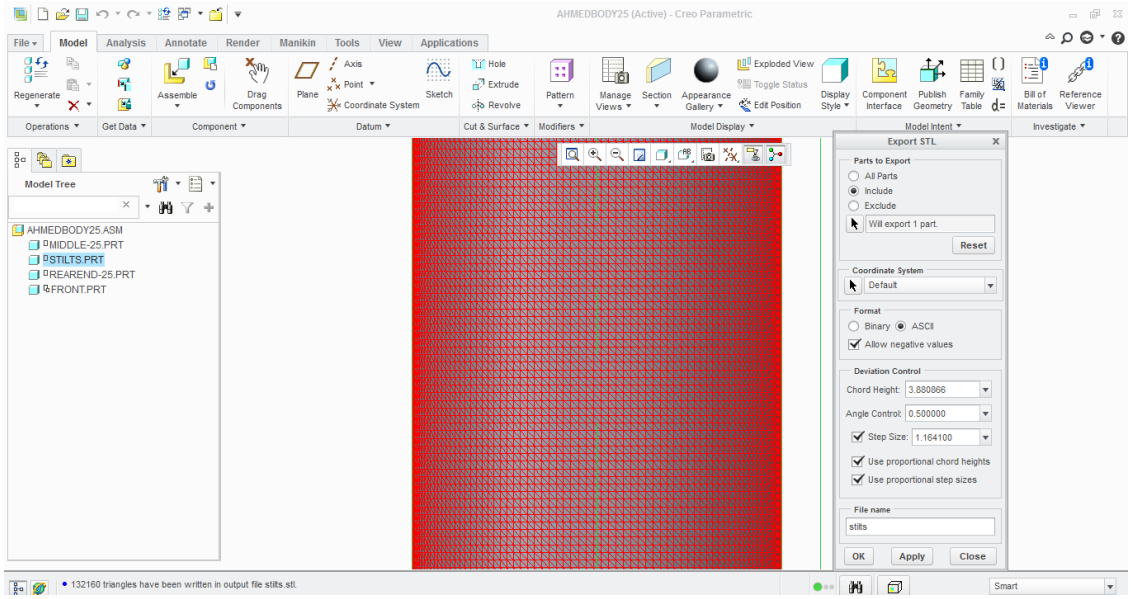


Figure 1.12: Step 11, specifying the number of vertices and the encoding type.

1.2 OpenFOAM®

Open-source **F**ield **O**peration **A**nd **M**anipulation is a free software for numerical analysis of fluids (i.e. CFD) which is written in C++. More info can be found by visiting this website `http://openfoam.com/`.

1.2.1 Structure Case File

An OpenFOAM folder should have the files shown in table 1.1.

Files	Function
`0:`	Time directory ~ *initial conditions of flow parameters.*
`U` and `p`	Should ALWAYS be there.
$\tilde{\nu}$ and ν_t	Should be there in case of Spalart-Allmaras.
$\tilde{\nu}$, ν_t and ν_{sgs}	Should be there in case of DES.
`constant:`	Mesh files and solver settings directory.
`\polyMesh`	Computational domain files.
`blockMeshDict`	User-defined dimensions and patches.
`\triSurface`	Geometry files.
`CAD_FIle.stl`	Written in ASCII and exported to .stl format.
`turbulenceProperties`	Switch the turbulence on/off.
`transportProperties`	Fluid properties.
`RASProperties` or `LESProperties`	Turbulence modelling approach.
`system:`	Integration, discretisation and iteration directory.
`controlDict`	run time, CFL number, Δt.
`fvSchemes`	Discretisation schemes.
`fvSolution`	Solver type.
`snappyHexMeshDict`	mesh features parameters.
`decomposeParDict`	Cut the mesh for parallel running.

Table 1.1: General OpenFOAM®case folder.

1.2.2 Mesh Generation

Meshing in OpenFOAM®can be carried out through HELYX-OS by making use of the notes found here `http://tinyurl.com/q2dteaw` or through the terminal by using the utilities `SnappyHexMesh` and `blockMesh`. In order to apply `SnappyHexMesh` to the case folder; there should be a geometry file written in ASCII and exported to .stl format and a computational domain that bounds the geometry which is created using `blockMesh` utility. This can be represented in figure 1.13.

Computational Domain

In order to create a correct computational domain; you need to make use of figures 1.16, 1.17 and 1.18. Then, check the following procedure:

1. Take a note of the dimensions of the boundary stack and the position of the geometry from the inlet. □
2. Take a note of the bounding box of your geometry; by typing the following command line in a terminal and the output will be similar to figure 1.14. □

```
surfaceCheck constant/triSurface/File_name.stl
```

3. Determine the bounding box of the computational domain relative to the position of your geometry. □

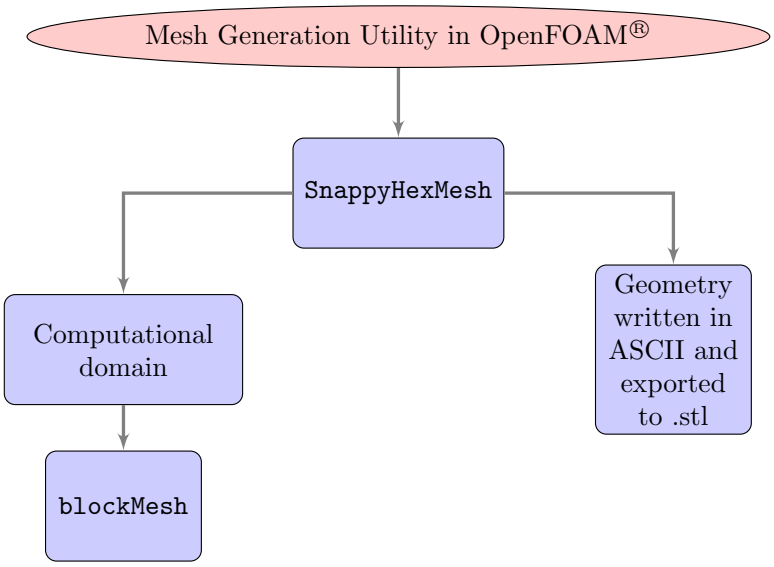


Figure 1.13: Mesh generation composition in OpenFOAM®

```
Statistics:
Triangles    : 323676
Vertices     : 161828
Bounding Box : (3.75 0.74 -3.35276e-08) (4.794 1.129 0.338)
```

Figure 1.14: Output of the `surfaceCheck` command in the terminal.

4. Fill in the `blockMeshDict`[2] by making use of figures 1.15 and 1.18 and table 1.2. ☐

```
mogo@m:~/Desktop/OpenFOAM-Case-folder$ ls
0  constant  system
mogo@m:~/Desktop/OpenFOAM-Case-folder$ gedit constant/polyMesh/blockMeshDict &
```

Figure 1.15: Shows how to open the `blockMeshDict` in a text editor from the terminal.

5. Run the following command in the terminal that orders OpenFOAM®to generate your computational domain: ☐

```
blockMesh
```

6. Visualise your computational domain and import the geometry file in ParaView®.[3]. ☐
7. Does the geometry lies in the correct position in the computational domain? If yes then tick this box. Otherwise, follow figure 1.17. ☐

[2]If you are not sure of where it is. Then, please check table 1.1

[3]If you are using your own linux machine, type the following command line `$> paraFoam &` . However, if you are on apocrita then you need to type this first `$> foamToVTK/` Then, download the VTK folder and the .stl file→ Directory on your machine → Import both files to ParaView®.

[4]Try to make the cell aspect ratio constant in all directions. For more info, the reader is encouraged to read the consecutive part about meshing the Ahmed body.

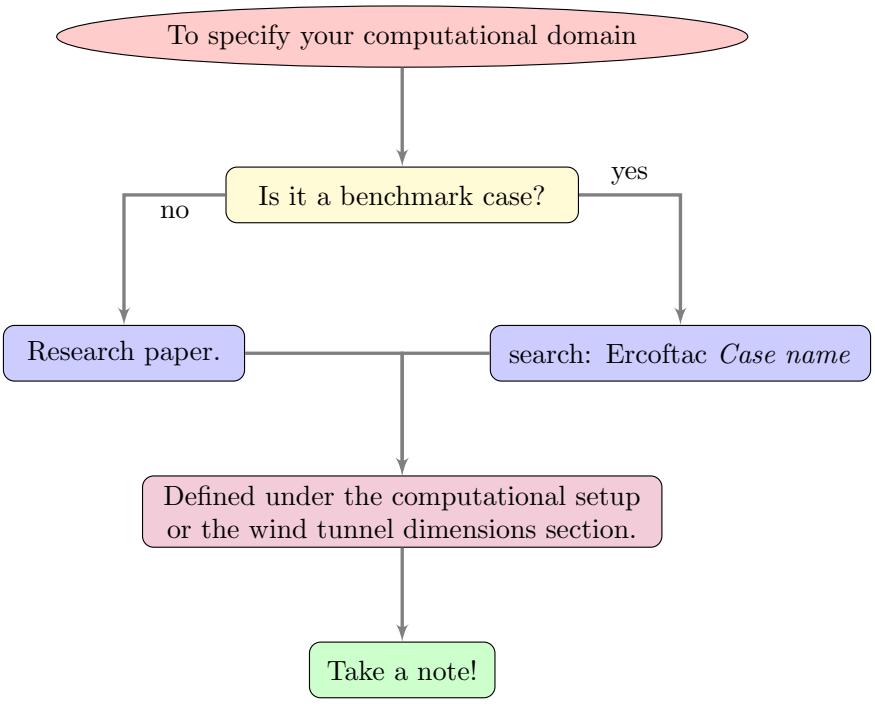


Figure 1.16: Ways to find the dimensions of the computational domain.

Problem 1.1 **The object is contained inside the computational domain. But, not placed in the right orientation as shown in figure 1.19.**
Solution:

1. Visualise the boundary stack in ParaView® by typing `blockMesh` in the terminal. ☑
2. Import the geometry file into ParaView® (*File* → *open* → *geometry.stl*). ☑
3. See what's wrong; as in figure 1.19. Then, plan a correction method. ☑
4. Rotate the geometry by 90° in the clockwise direction by typing the following line[5]. Then, see figure 1.20.: ☑

```
surfaceTransformPoints -rollPitchYaw '(90 0 0)'  geometry.stl
    rotatedGeometry.stl
```

5. Find the current position of the body by typing the following command in a terminal. ☑

```
surfaceCheck constant/triSurface/geometry.stl
```

6. Translating vector = final position (To) - the current position (From). ☑
7. Type the following line in the terminal: ☑

```
surfaceTransformPoints -translate '(translated vector)'
    rotatedGeometry.stl translatedGeometry.stl.
```

8. Repeat steps 1 and 2 (*and the output is shown in figure 1.20*). ☑

Exercise 1.1 Try an alternative correction procedure to problem 1.1.
hint: Don't change the orientation of the geometry but rather modify the blockMeshDict. ■

[5] Where geometry.stl is the current name of the geometry file while rotatedGeometry.stl is a user defined file name.

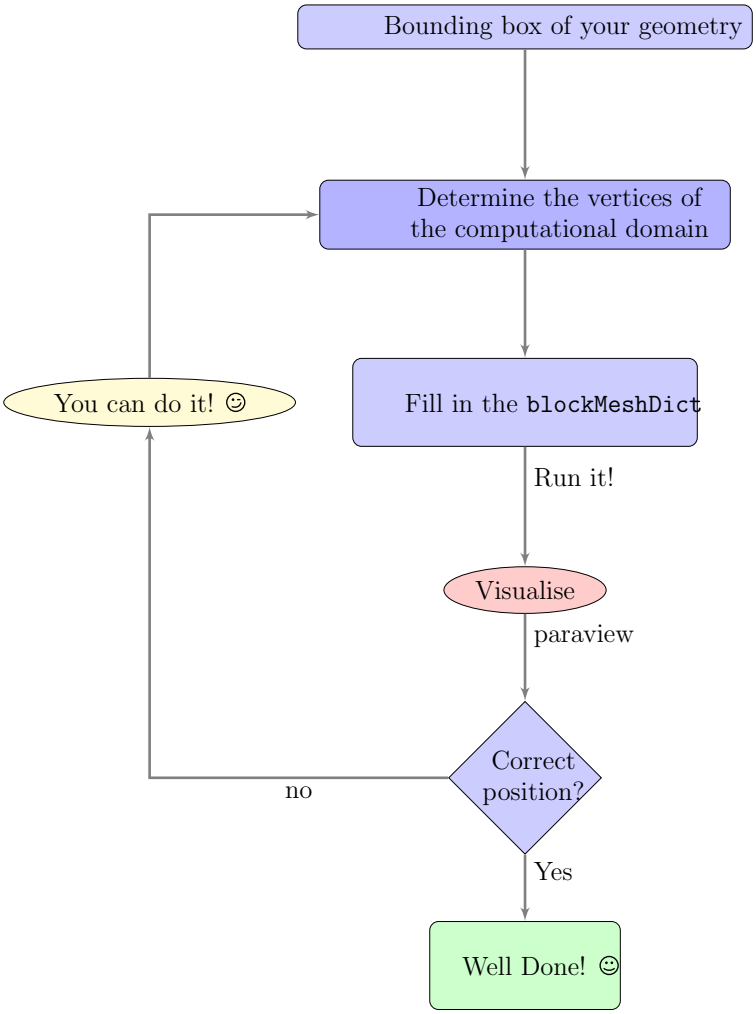


Figure 1.17: Ways to create the computational domain.

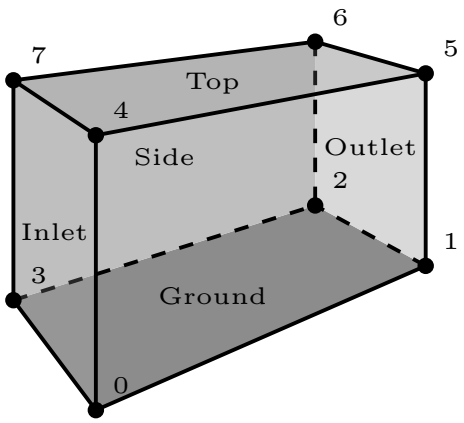


Figure 1.18: General block topology of the `blockMesh` utility.

blockMeshDict	**Function**
`convertToMeters 1;`	Assuming that you converted the cad.stl file to **m**.
`vertices`	Vertices for a single block as shown in figure 1.18.
(	
(1.662 -0.0375 0)	coordinates of vertex 0.
(10.014 -0.0375 0)	coordinates of vertex 1.
(10.014 1.9075 0)	coordinates of vertex 2.
(1.662 1.9075 0)	coordinates of vertex 3.
(1.662 -0.0375 1.44)	coordinates of vertex 4.
(10.014 -0.0375 1.44)	coordinates of vertex 5.
(10.014 1.9075 1.44)	coordinates of vertex 6.
(1.662 1.9075 1.44)	coordinates of vertex 7.
);	
`blocks`	Specify the number of cells for the defined block(s).
(	
`hex (0 1 2 3 4 5 6 7)`	Hexahedral mesh block with correctly ordered vertices as in figure 1.18.
`(24 8 6)`	Number of cells in x, y and z directions respectively[4].
`simpleGrading (1 1 1)`	Difference in the refinement between the beginning and end cells for all directions. 1 means uniform spacing.
);	
`edges ( );`	Leave it as it is; for a single block as shown in figure 1.18.
`patches`	Define 6 patches as shown in figure. 1.18
`( patch inlet ( (0 4 7 3) )`	
`patch outlet ( (1 2 6 5) )`	
`wall ground ( (0 3 2 1) )`	
`symmetry sides ( (3 7 6 2) (0 1 5 4) )`	Symmetry means that half the domain is modelled.
`symmetry top ( (4 5 6 7) )`	
);	

Table 1.2: General `blockMeshDict` file.

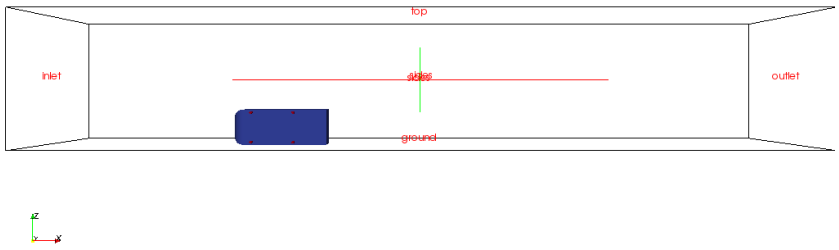


Figure 1.19: The Ahmed body is placed in the correct spot in the boundary stack. But, rotated 90° counter-clockwise.

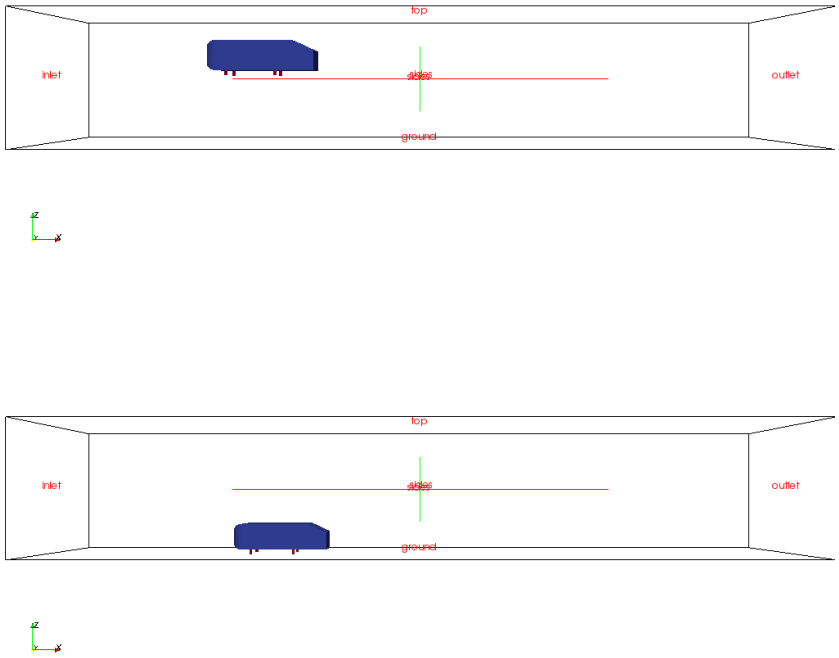


Figure 1.20: Correction procedure; rotation step (top) and translation step (bottom).

Ahmed Body

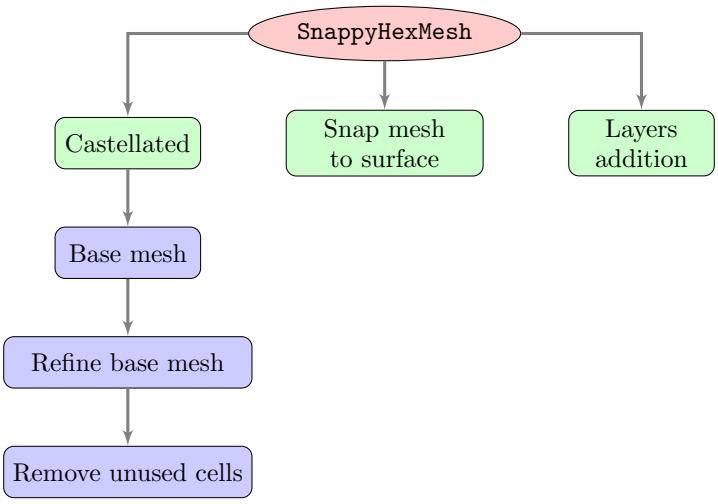


Figure 1.21: Automatic mesh generation functionality in OpenFOAM®

Presented in this part a general overview of the different processes undertaken by the `snappyHexMesh` utility. The mesh generation is decomposed into three main stages as shown in figure 1.21. Followed by a prepared copy of a `snappyHexMeshDict` which are shown in figures 11, 12, 13, 14, 15 and 16.

Castellated Mesh: is the first process undertaken by the mesher; where the user has to define the refinement regions and the level of refinement depends on the type of simulations. In this case, two boxes of very high gradient were defined at the rear end to cover the wake. Then, two boxes of varying densities covered the whole car.

Snapping stage: resolves the sharp edges of the geometry according to the skewness level and the max. orthogonality scales set by the user.

Layers addition: The last process undertaken by the mesher; where the user has to specify the number of boundary layers, the minimum thickness of the first cell away from the wall (**y**), the stretching ratio (`expansionRatio`) and the ratio of the height of the final layer cell to its width (`finalLayerThickness`).

Strategies for choosing a reasonable y for simulating a viscous flow:

1. Interested in the linear sublayer where $0 \leq y^+ \leq 5$?
   - This choice is very expensive.
   - Deactivate the "layer addition" process in the `snappyHexMeshDict`.
   - Specify a box near the wall with high level of refinement.
2. Interested in the log-layer, where $30 \leq y^+ \leq 60$?
   - OpenFOAM®contains a variety of wall functions; that are compatible with specific turbulence modelling approaches.
   - Activate the "layer addition" process.
   - Calculate the minimum thickness of the first cell near the wall using any yPlus calculator found online.

1. Resolve the sharp edges of the geometry. By typing the following command: □

```
surfaceFeatureExtract -includedAngle 150 -writeObj constant/triSurface
    /geometryfile.stl geometry
```

2. Run the automatic mesher; by typing the following command: □

```
snappyHexMesh -overwrite
```

Meshing strategy in OpenFOAM®can be summarised in figure 1.22:

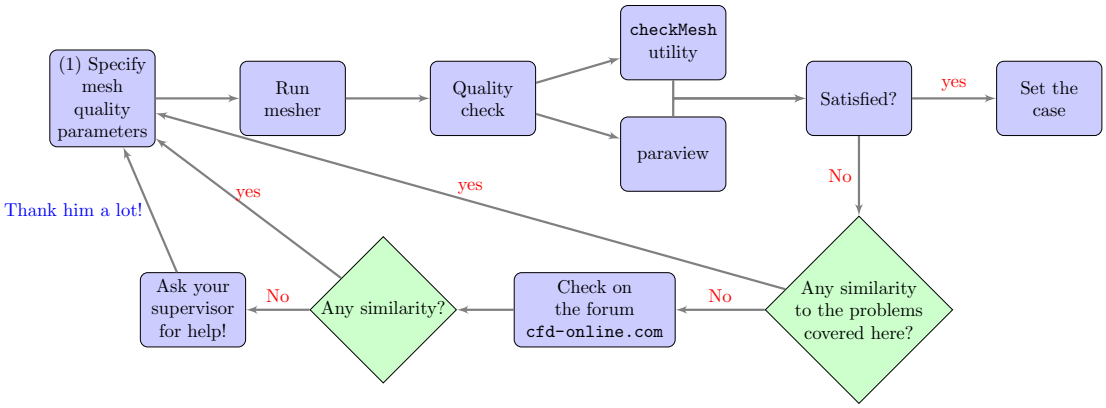


Figure 1.22: Meshing strategy in OpenFOAM®

Preceding Work

Before attempting the meshing stage; one should know the flow physics over the Ahmed body. You are encouraged to read the paper by Gillieron *et al* which can be downloaded from here `http://goo.gl/OzyN2w`.

As the flow hits the car, a high pressure gradient was captured at the nose. A separation line (Kelvin Helmholtz) was observed by Gillieron *et al.* at the top side of the nose. Then, the fluid travels with constant momentum alongside the middle part. As the flow at the top side reaches the rear-end, the momentum suddenly drops, and fluid re-attached alongside the hatch part generating swirling vortices. At the same time, the flow at the bottom rear end will generate a smaller vortices which is opposite in direction to the vortices generated from the flow at the top side. As a consequence, the vortices will combine together resulting in an unsteady flow. .

Own mesh: In order to capture the flow separation; 3 boxes of refinement were specified in the `snappyHexMeshDict` one at the frontal nose, hatch-side and the wake. In addition, a very small refinement box was added near the bottom right rear-end of the car to capture the small vortices coming from the bottom side. As shown in figure 1.23.

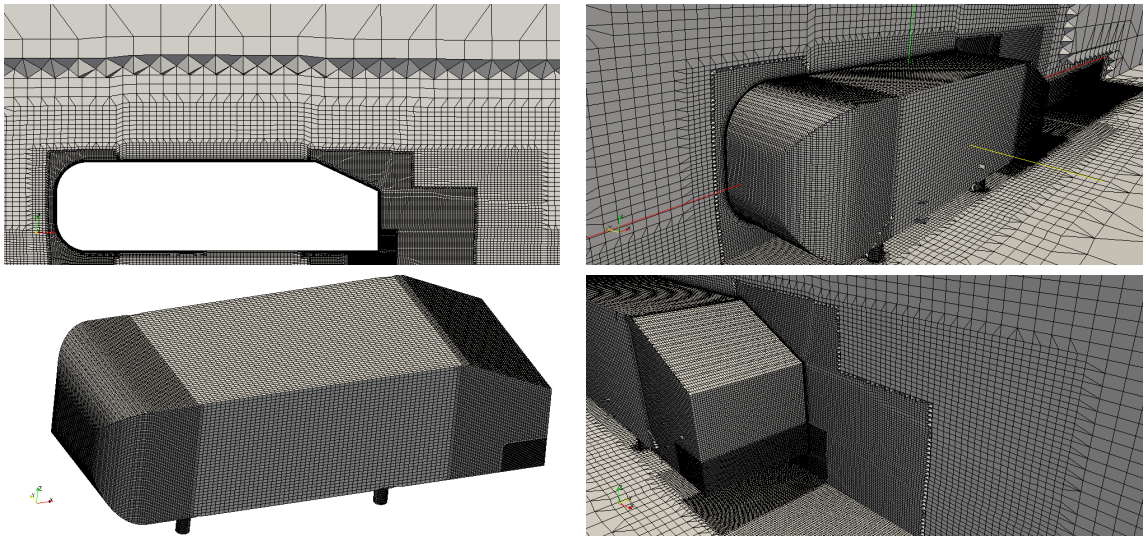


Figure 1.23: My final mesh of the Ahmed body, it consist of 2,766,483 cells and 2,280,483 points.

Problem 1.2 The cells on the body are not balanced and separated by regions of high density cells. As a consequence, the layers will grow badly as shown in figure 1.24.

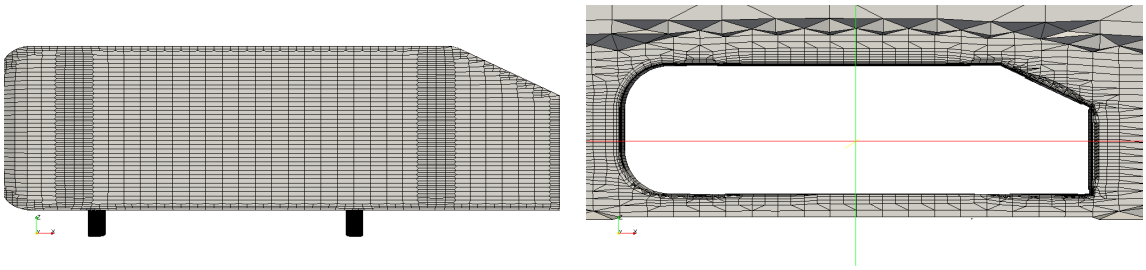


Figure 1.24: Regions of varying cells densities; Ahmed body (left), xz-symmetry plane (right).

Solution: The problem is due to the unbalance of the aspect ratio of cells in the x,y and z directions as shown in figure 1.25. Balancing the number of cells in the x, y and z directions will solve this problem as shown in figure 1.26.

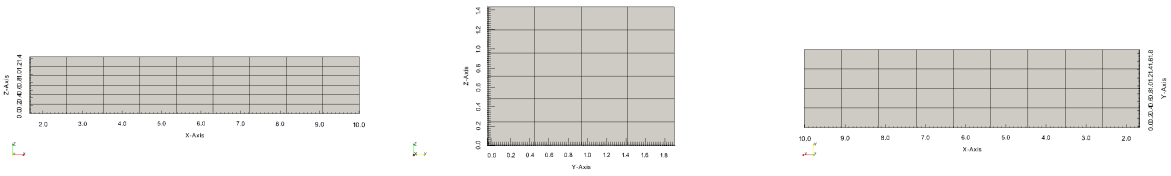


Figure 1.25: unbalanced computational domain from different orientations; xz-axis (left), yz-axis (middle) and xy-axis (right).

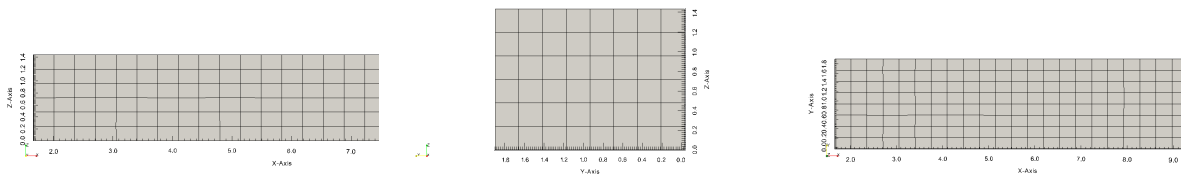


Figure 1.26: Balanced computational domain from different orientations; xz-axis (left), yz-axis (middle) and xy-axis (right).

Problem 1.3 The layers did not grow at all as in figure 1.27.

Solution: This might be due to two reasons; either the `minimumThickness` was chosen greater than 1.0 or chosen smaller than 0.1. By choosing a value between this range (i.e. `minimumThickness`=0.4) the layers will grow as in figure

Problem 1.4 There is a very bad cell at the bottom rear end of the car as in figure 1.30. Those bad cells can influence the solution and result in non-physical results.

Solution: By changing the number of cells in the computational domain so that the cell aspect ratio is **1** : **1** : **1**. The overall cells will be stretched or compressed accordingly.

Problem 1.5 Y+ is in the wrong range, less than 10 boundary layers, underestimated stretching ratio. This might result in non-physical results.

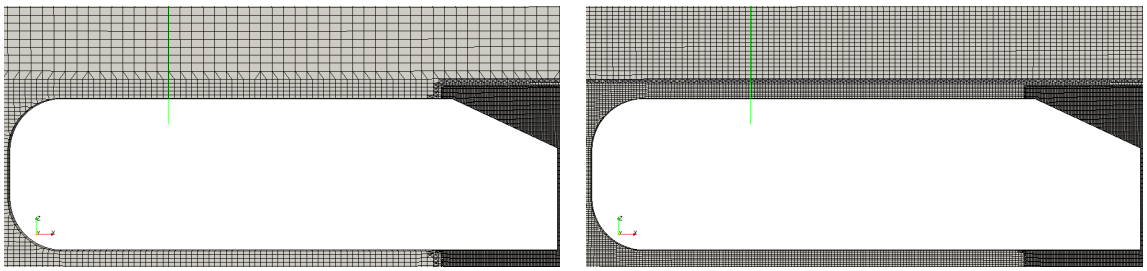


Figure 1.27: xz-symmetry plane showing no layer growth; `minimumThickness`=3 (left), =0.08 (right).

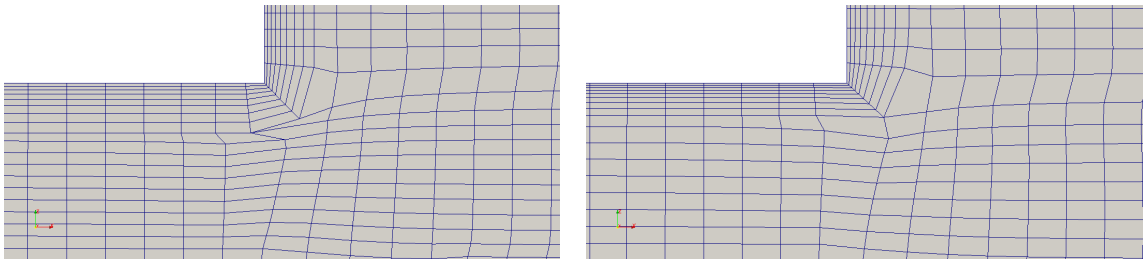


Figure 1.28: Cells at the bottom rear end of the car; bad cell (left), good cell (right).

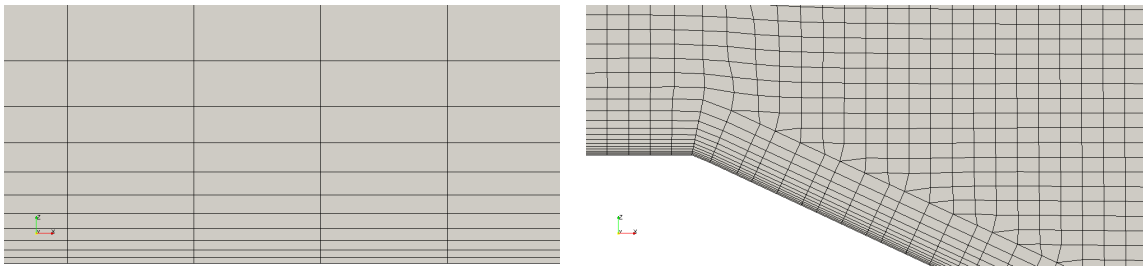


Figure 1.29: Boundary layers within y+ in the range of 40<y+<70.

Meshing journey

Presented in this part some of the attempts I went through before arriving at my final mesh

■ **Attempt 1.1**

■ **Attempt 1.2** The second mesh generated using `snappyHexMesh`. The mesh consist of 1,562,813 cells and 1,307,831 nodes.

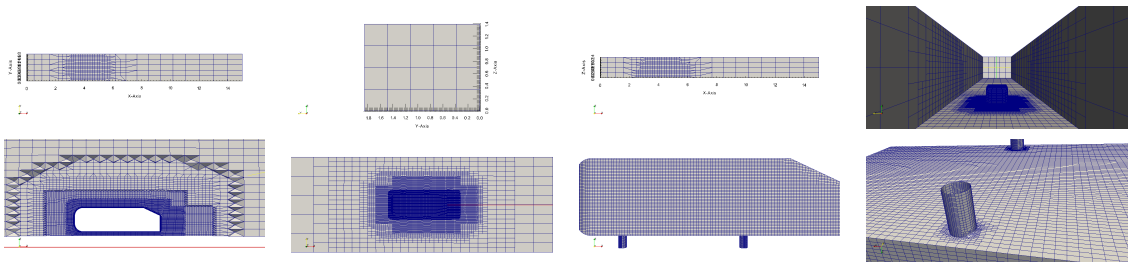


Figure 1.30: Second mesh generated in OpenFOAM®.

1.2.3 Case Setup

In order to proceed further in this section; you should check the following procedures first:

1. Determine the turbulence at the inflow boundaries. ☐
   - Specify the molecular viscosity ν from the definition of the **Reynolds number** ☐

     $$\mathrm{Re} = \frac{\mathbf{U} \cdot L}{\nu} \tag{1.1}$$

     Where $\mathbf{U}$ *is the mean velocity of the object with respect to the fluid flow.* L *is the characteristic length.*
   - Specify the turbulent viscosity using the following relation: ☐

     $$\nu_t = \beta \cdot \nu \tag{1.2}$$

     Where $\beta = 100$ *and is constant for* $Re \geq 100,000$.
   - Specify the modified kinematic viscosity $\tilde{\nu}$ at the free-stream, using the following relation: ☐

     $$\tilde{\nu} = 5\nu \tag{1.3}$$

   - Use $\tilde{\nu}$=0 at the wall. ☐
2. Determine the time step: ☐
   - Recall that the CFL number is a dimensionless ratio (briefly how much of the transport solution can travel one cell width in one second); which can be expressed mathematically as: ☐

     $$\mathrm{CFL} = \frac{u \cdot \Delta t}{\Delta x} \tag{1.4}$$

     Where Δt *is the time step,* Δx *is the cell width and is a grid characteristic*[6]
   - How many iterations? ☐

[6]For a transient solution; you need to resolve the eddy with 4 time-step. More info is available online ;-)

3. Scenarios for solution convergence? ☐
   - Request lift and drag coefficients. ☐
   - Specify the spots to place pressure probes. ☐
4. Specify the finite volume schemes? ☐
   - Steady state or transient solution? ☐
   - 1st or 2nd order solution? ☐
5. Specify the solver that you will be using? ☐
   - SIMPLE or PISO or PIMPLE? ☐

Setting the case using GUI HELYX-OS

1. Launch HELYX from a terminal. ☐

```
./HELYX-OS.sh &
```

2. Import your case folder: Open → Path to your case folder → Open. ☐
3. Make sure there is no errors. ☐
4. Switch to the "Case setup" tab. ☐
5. Set the solution modelling: ☐
   (a) Time → steady. ☐
   (b) Flow → incompressible. ☐
   (c) Turbulence model → Spalart-Allmaras ☐
6. Set the materials: ☐
   (a) $\rho = 1.225$. ☐
   (b) Dynamic viscosity = 0.000018375. ☐
   (c) Kinematic viscosity = 0.000015. ☐
7. Set the Boundary Conditions: ☐
   (a) **Inlet:**
      i. Velocity: Type → fixed value and $U_x = 40$. ☐
      ii. Pressure: zero gradient. ☐
   (b) **Outlet:**
      i. Velocity: Type → inletOutlet. Inlet value → (0.0 0.0 0.0). ☐
      ii. Pressure: Type → fixed value. Value → 0.0 ☐
   (c) **Ground & Ahmed body patches:**
      i. Patch type → wall. Momentum Type → Fixed. Wall Type → No-slip. ☐
   (d) **Sides and Top**
      i. Patch type → symmetry. ☐
8. Set the numerical schemes: ☐
   (a) **Advection:**
      i. **U**: first run → Upwind-1st order. Then, switch to bounded linear upwind 2nd O. ☐
      ii. **nuTilda**: first run → Upwind-1st order. Then, switch to bounded linear upwind 2nd O. ☐
   (b) **Laplacian:**
      i. Non-orthogonal corrections: 0.333 ☐
9. Set the Solver Settings: ☐
   (a) Non-orthogonal correctors = 0. ☐
   (b) Residual control = 0.0001. ☐
   (c) Relaxation factor[7]: ☐
      i. **U** = 0.7 ☐
      ii. **p**=0.3. ☐
      iii. $\tilde{\nu}$=0.7 ☐

[7] The sum of the relaxation factor values of the velocity and pressure should equal to 1.

10. Set the Run-time controls: ☐
    (a) **Start from**[8]**:** start time → 0.0 ☐
    (b) **End-time**[9]**:** 5000 ☐
    (c) Data Writing: ☐
        i. Write Control[10]
            A. Time step → every 500 iteration. ☐
11. Set the Field Initialisation: ☐
    (a) **U**: Fixed value → (40 0 0). ☐
    (b) **p**: Fixed value → 0.0 ☐
    (c) $\tilde{\nu}$: Fixed value → 0.0015 ☐
    (d) Click initialise[11] ☐
12. Run the simulations in series: ☐
    (a) Switch to **Solver** tab → Run. ☐

Setting the case manually in the terminal

Presented in this part a general view of the case setup. Starts by going through the dictionary files, setting the boundary and initial conditions. A quick introduction to discretization schemes and solver control. Finally, how to start the simulation in serial and parallel.
(*Setting the case in OpenFOAM® flow chart to be placed here...*)

RANS

The initial and boundary conditions dictionary files can be copied from the tutorials folders that are pre-installed in OpenFOAM®, a suitable case is the motorbike one. The initial and boundary conditions for this case can be copied using the following command:

```
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/motorBike/0.org
    path_to_your_case_folder/        // This command copies the time
    directory folder to your own case folder.
cp -r 0.org  0  // This command renames the 0.org to 0
```

The solver settings, fvSchemes and the RANS-settings files that are vital for running this simulation are available in some of the tutorials folders that are pre-installed in OpenFOAM®, a suitable case that runs Spalart-Allmaras can be found if you typed the following command:

```
cd $FOAM_TUTORIALS/incompressible/simpleFoam/airFoil2D/
```

Now, you need to copy the finite volume files (i.e. discretisation schemes and the solver type), run time file, RANS-settings files to your own case folder by running the following commands on at a time in a terminal respectively:

```
cp system/fvSchemes /your_case_folder_directory/system
cp system/fvSolutions /your_case_folder_directory/system
cp system/controlDict /your_case_folder_directory/system
cp constant/RASProperties /your_case_folder_directory/constant
cp constant/transportProperties /your_case_folder_directory/constant
```

The molecular viscosity value can be changed in the `transportProperties` file as shown in figure 1.31.

[8] Click on the drop-down arrow to see other options. Choose "Start time" if you haven't run the simulations before. Or "latestTime" if you already ran it and haven't converged and would like to run few more iterations.

[9] The number of iterations to run.

[10] After how many iterations would you like OpenFOAM® to store the data.

[11] In case, you already ran few iterations and would like to start from initial conditions rather than starting from the latest calculated conditions.

```
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

rho             rho [ 1 -3 0 0 0 0 0 ] 1.205;

nu              nu [ 0 2 -1 0 0 0 0 ] 1.5e-05;
```

Figure 1.31: The molecular viscosity value as seen in the `transportProperties` file.

DDES

Setting the velocity (U), pressure (p) and turbulent viscosity (ν_t) is the same as in figures 20 and 21 respectively. Now, we are left with two unset viscosity variables ($\tilde{\nu}$ and ν_{Sgs}) which can be set by copying the `nut` file twice and renaming one as `nuSgs` and the second as `nuTilda` as follows:

```
cp 0/nut 0/nuSgs
cp 0/nut 0/nuTilda
```

1.3 High Performance Computing

Multiprocessing computing can be classified into **MPI** and **openMP**.

MPI	OpenMP
- Parallel computing. - Communications between processors - One process runs on one core - Point-to-point operation where each processor runs one process (i.e. calculation of one cell) then passes the solution to another processor in an iterative method.	

Table 1.3: Comparison between MPI and openMP

OpenFOAM® just support MPI jobs. Thus, if you tried to run your job using openMP it will crash.
Submitting a parallel job onto Apocrita, can be as follows:

1. Connect to the apocrita server using the command: ☐
   `ssh -X username@login.hpc.qmul.ac.uk`
2. Copy and paste your password from the email sent to you by the cluster admin[12]. ☐
   Now, you arrived at the `front-end` which is the receptionist. "From here you will request the number of nodes, cores and the period you need to accommodate your job for. Followed by the type of work (simulation type, decomposing your mesh etc...) that you want the supercomputer to run all written in commands as in figure 27. Then, this request will be passed on to the `back-end` which will place your job in a queue (batch) till there is an empty slot to accommodate your job" as described by Dr. J-D. Müller in one of his lectures.
3. Generate your case folder as described in section 1.2.3. ☐
   If you have OpenFOAM® installed on your box. Then, you can prepare the case on your laptop, hence, upload it onto Apocrita using:
   `scp -r -p directory_caseFolder/folder_name QMuserID@login.hpc.qmul.ac.uk:folderName`
4. Change your directory from the current one to the case folder ☐

```
cd case_folder_name
```

5. Write a file script specifying the amount of virtual memory, the run time etc... ☐
   Or Copy the following lines if you want to mesh your geometry:

```
#!/bin/sh
#$ -cwd -V //Set the working directory for the job to the current
    directory
#$ -pe openmpi 1 //Ordering a whole node.
#$ -l h_rt=10:0:0 //Do you want your simulation to run for 10 hrs?
#$ -l h_vmem=24G //Requesting 24Gb of RAM (as 1 node=12cores=24Gb)
#$ -m base //Get an email at the Beginning of the job, Alert,
    Something that I might have forgotten and as it Ends.
#$ -M email_address //Your email address where the ''base'' can be
    sent to.
#$ -N Job_name // User-defined job name.
blockMesh // Generate the virtual wind tunnel.
surfaceFeatureExtract -includedAngle 150 -writeObj constant/triSurface
    /geometryfile.stl geometry // resolve the sharp edges
```

[12] To paste any thing in the terminal; use `ctrl+shift+v` instead of the recognised `ctrl+v`. You are advised to change your password by typing `passwd` in the terminal

```
decomposePar //Order OpenFOAM to generate your mesh on a number of
    processors.
foamJob -parallel -screen snappyHexMesh -overwrite //Terminate the
    parallel meshing process
reconstructParMesh -mergeTol 1e-6 -constant //reconstructing your mesh
    .
ls -d processor* | xargs -i rm -rf ./{} $1//delete the processor files
    .
checkMesh // check if your mesh has failed certain criteria.
foamToVTK // convert your mesh into a Paraview friendly format.
```

(a) Now that you meshed your geometry, you should visualise it in ParaView® as discussed in section 6.

(b) If you want to run the solver on apocrita, then copy the following lines:

```
#!/bin/sh
#$ -cwd -V //Set the working directory for the job to the current
    directory
#$ -pe openmpi 1 //Ordering a whole node.
#$ -l h_rt=10:0:0 //Do you want your simulation to run for 10 hrs?
#$ -l h_vmem=24G //Requesting 24Gb of RAM (as 1 node=12cores=24Gb)
#$ -m base //Get an email at the Beginning of the job, Alert,
    Something that I might have forgotten and as it Ends.
#$ -M email_address //Your email address where the ''base'' can be
     sent to.
#$ -N Job_name // User-defined job nam
decomposePar //Distribute your mesh and the time fields among n
    processors.
mpirun -np 12 simpleFoam -parallel //Run your simulation on 12
    cores using the simpleFOAM solver.
reconstructPar//reconstruct your case.
ls -d processor* | xargs -i rm -rf ./{} $1//delete the processor
    files.
yPlusRAS // Calculate the y+ incase of using any RANS approach.
foamToVTK //This format is easily imported to Paraview.
```

Try to avoid the following mistakes when writing a file script:[13]

1. Ending any of your commands with '&'. This means that you ordered your job to run in the background of the node so the Sun Grid Engine would probably kill your job.
2. Redirecting your job to a 'log' file is un-necessary. By default the Sun Grid Engine (queueing system) will create two files; an output file 'job_name.o' and an error file 'job_name.e'.
3. Following your 'mpirun' command. Write the number of cores that you are getting from one node (i.e. in case of small nodes it is 12 cores). If you wrote a number >12 (i.e. 16). Still you will have your job running. But, your job will be slowed down as a result of 2 processes running on 4 cores.

For more information and alternative ways on how to write a file-script you can visit this website: `https://www.hpc.qmul.ac.uk/twiki/bin/view/HPC/SubmittingJobs`

1.3.1 Grid Decomposition

Presented in this sub-section an overview of the different mesh decomposition methods in OpenFOAM® as shown in figure 1.32. In order to decompose your mesh, please complete the following steps:

1. Copy the `decomposeParDict` file to your `case_folder/system`. The data inside this file can be found in this report's appendix or from the motorbike tutorial case folder which can be found in the following directory: □

[13] Thanks to the IT-research and support team for their clarification in writing a correct file script.

```
$FOAM_TUTORIALS/incompressible/simpleFoam/motorBike/system/
    decomposeParDict
```

2. Edit the `decomposeParDict` by choosing a suitable mesh decomposition method and specifying the number or the order of cuts as illustrated in figure 26. ☐
3. Type the following command to start the mesh cutting process: ☐

```
decomposePar
```

After running your parallel simulation; you may want to reconstruct your mesh for post-processing. This is a one step which can be done by typing the following command:

```
reconstructPar
```

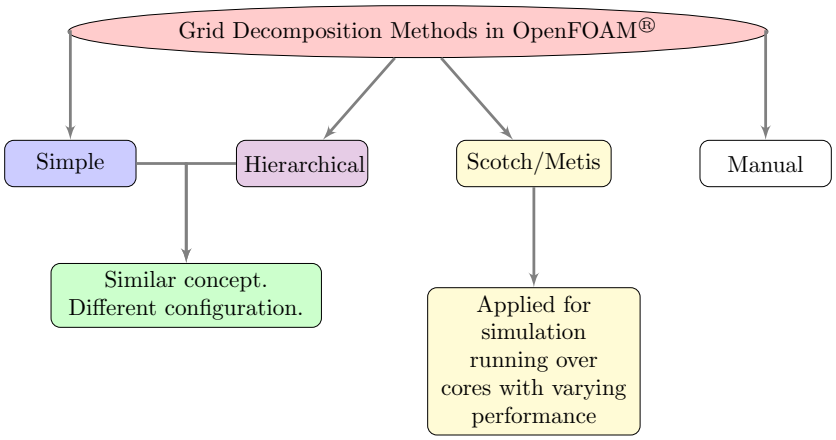


Figure 1.32: Mesh decomposition in OpenFOAM®

A good source worth considering is this link `http://tinyurl.com/nqg8k2v` and if you got time on your hands (i.e. **68 minutes and 36 seconds**) then why not considering watching these tutorials `http://tinyurl.com/pm7pc8r`, `http://tinyurl.com/qfyddvr` and `http://tinyurl.com/ozsydkm`.

1.3.2 Potential Run

- Why potential run?

1.4 Monitoring your Job

1.4.1 Convergence check techniques:

Force Coefficients

The force coefficients are useful in checking the degree of convergence of the solution. But, this function is not calculated automatically with the simulation, you need to order OpenFOAM® to do so; which can be done adding few lines at the end of the `controlDict` file. Those lines are as follows:

```
functions
{
        forceCoeffs
        {
    type         forceCoeffs;

    functionObjectLibs ( "libforces.so" );

    outputControl   timeStep;           // instead of timeStep, you can
        try runTime.
    timeInterval    1;

    log          yes;             // The calculations will be produced in
        a separate .dat file.
    patches      ( "ahmed_.*" ); // name of the patches where you want to
         calculate the forceCoeffs.
    pName        p;
    UName        U;
    rhoName      rhoInf;       // Indicates incompressible
    log          true;
    rhoInf       1.225;            // Redundant for incompressible
    liftDir      (0 0 1);                  // z-component
    dragDir      (1 0 0);                  // x-component
    CofR         (4.272 0.9345 0);  // Axle midpoint on ground
    pitchAxis    (0 1 0);
    magUInf      40;    // free stream velocity
    lRef         1.044;         // Wheelbase length
    Aref         0.112;         // Estimated
        }
}
```

The lift and drag coefficients can be found in the postProcessing folder in the main case directory (i.e. `case_directory/postProcessing/forceCoeffs/start_times/forceCoeffs.dat`, where start_times are the numbers when you started the simulation).
During the simulation you can view the force coefficients to check the degree of unsteadiness in the solution using either two ways:

1. Type the following command to view the last few lines of the force coefficients:

   ```
   tail -n 500 postProcessing/forceCoeffs/0/forceCoeffs.dat
   ```

2. Plot the force coefficients on the go using gnuplots (an opensource package that is pre-installed in most linux packages; and is the most efficient way to plot any graph from the command-line):
   (a) Order gnuplots to extract the force coefficients from its directory relative to your current directory (i.e. if you are currently in the main case folder then the force coefficients are there → `case_directory/postProcessing/forceCoeffs/start_times/forceCoeffs.dat`):

- Open a text editor, by typing the name of the editor (in this case its `nano` which is a very simple and light text editor), followed by the file name:

```
nano forcecoeffs
```

- Copy the few lines into the file you just created:

```
set logscale y
set key bottom right
set xlabel "Simulationtime [s]"
set ylabel "forceCoeff [-]"
set title "Plot of forceCoeffs over simulationtime"
set grid

plot    "postProcessing/forceCoeffsCyl/0/forceCoeffs.dat"
    using ($1):($4) with lines title "lift_coeff",\
         "postProcessing/forceCoeffsCyl/0/forceCoeffs.dat"
            using ($1):($3) with lines title "drag_coeff"
pause 1
reread
```

If you became familiar with using gnuplots, you can replace the first line in the previous script to define the x and y range so instead of "`set logscale y`" you can write the following lines :

```
set  yr [0:1]
set  xr [1:100]
```

where $0 \leq y \leq 1$ and $1 \leq x \leq 100$. You can specify any range you want!

(b) Type the command that will plot your graph instantly:

```
gnuplot forceCoeffs
```

(c) To close this window; press `ctrl+c` in the terminal.

An example of the force coefficients plots is illustrated in figure 28 for the iterations (50, 100, 500 and 4000) of the own RANS simulation.

Residuals

Plotting the residuals using gnuplots will follow the same main steps as plotting the force coefficients. However, the only difference is the file name and the lines that you will need to copy into that file: i.e.

```
set logscale y
set title "Residuals"
set ylabel 'Residual'
set xlabel 'Iteration'
plot "< cat solver_output_file | grep 'Solving for Ux' | cut -d' ' -f9 |
     tr -d ','" title 'Ux' with lines,\
     "< cat solver_output_file | grep 'Solving for Uy' | cut -d' ' -f9 |
          tr -d ','" title 'Uy' with lines,\
     "< cat solver_output_file | grep 'Solving for Uz' | cut -d' ' -f9 |
          tr -d ','" title 'Uz' with lines,\
     "< cat solver_output_file | grep 'Solving for omega' | cut -d' ' -
        f9 | tr -d ','" title 'omega' with lines,\
     "< cat solver_output_file | grep 'Solving for k' | cut -d' ' -f9 |
        tr -d ','" title 'k' with lines,\
     "< cat solver_output_file | grep 'Solving for p' | cut -d' ' -f9 |
        tr -d ','" title 'p' with lines,\
     "< cat solver_output_file | grep 'Solving for nuTilda' | cut -d' '
        -f9 | tr -d ','" title 'nuTilda' with lines

pause 1
reread
```

If you are using your own linux box to run the simulations. Then, the `solver_output_file` can be requested before running the simulation by typing:

```
simpleFoam > log &
```

On the other hand, if you are running your simulation on Apocrita. Then, you do not have to request a log file. Because, the sub grid engine will output four files. One of them is the log file. In both ways, you will have to replace `solver_output_file` with `log` (in case you are using your linux box) or with the output file name from apocrita (i.e. in most cases it will look something like this `Job_name.o11225577`).
An example of the residual plots is illustrated in figure 31 for the iterations (50, 100, 500 and 4000) of the own RANS simulation.

Probes

A useful way to check for convergence, is by placing pressure probes in the regions where separation occurs. In the separation region (i.e. wake of the car) the solution is unsteady and fluctuates rigorously. You will need to copy the few lines to the end of the `controlDict` file before the last curly bracket that belongs to the `functions` and not the `forceCoeffs`. I assume you previously copied the `forceCoeffs` function.

```
    probes
    {
        type            probes;
        functionObjectLibs ("libsampling.so");
        enabled         true;
        outputControl   timeStep;
        outputInterval  1;
        fields
        (
            p
        );

        probeLocations
        (
            ( 4.8 0.7725 0.200 ) // define your own probe locations
            ( 4.9 0.7725 0.300 )
            ( 4.9 0.7725 0.250 )
            ( 4.85 0.7725 0.250 )
            ( 4.85 0.9725 0.160 )
            ( 4.80 0.7725 0.180 )
        );
    }
```

Viewing the calculated pressure probes is a similar process to viewing the force coefficients as previously discussed. To plot the pressure probes; you can copy the following lines into a file then type the command `gnuplot file_name`:

```
set logscale y
set key bottom right
set xlabel "Simulationtime [s]"
set ylabel "Probes [-]"
set title "Plot of Probes over simulationtime"
set grid

plot    "postProcessing/probes/0/p" using ($1):($2) with lines title "
    (4.8 0.7725 0.2)",\
        "postProcessing/probes/0/p" using ($1):($3) with lines title "
            (4.9 0.7725 0.3)",\
        "postProcessing/probes/0/p" using ($1):($4) with lines title "
            (4.9 0.7725 0.25)",\
```

```
        "postProcessing/probes/0/p" using ($1):($5) with lines title "
            (4.85 0.7725 0.25)",\
        "postProcessing/probes/0/p" using ($1):($6) with lines title "
            (4.85 0.9725 0.16)",\
        "postProcessing/probes/0/p" using ($1):($7) with lines title "
            (4.8 0.7725 0.18)"

pause 1
reread
```

An example of the pressure probes plots is illustrated in figure 30 for the iterations (50, 100, 500 and 4000) of the own RANS simulation.

1.5 ParaView®

1.5.1 Pre-processing

y+ check

The y^+ can be checked manually before running the simulation. By doing the following steps:

1. Import your mesh into paraview. ☐
2. Extract one cell and view its axis, then take a note of the height of this cell (y). ☐
3. Calculate the y+ using the following equation: ☐

$$y^+ = y \cdot \frac{\nu}{u_\tau} \tag{1.5}$$

where ν is the molecular viscosity (i.e. $\nu = 1.5 \cdot 10^{-5} m^2/s$). u_τ is the friction velocity and is defined as:

$$u_\tau = U_{Ref} \sqrt{\frac{c_f}{2}} \tag{1.6}$$

Where U_{Ref} is the velocity of the air relative to the car, and c_f is the skin friction coefficient; which can approximated for a flat plate as follows:

$$c_f \approx 0.074 Re_L^{-0.2} \tag{1.7}$$

A constraint for the previous equation is that $5 \cdot 10^5 < Re_L < n \cdot 10^7$.

1.5.2 Post Processing

Presented in this sub-section the technique in carrying out the post-processing of the simulation results in ParaView®. There are few points that should be taken into account when comparing plots of the same quantities, which are:

1. Use the same legend range, which is done by:
   (a) Click **toggle color legend visibility** icon; to view the default range set for the chosen property.
   (b) Click on **edit color map** icon → **Rescale to custom range** icon (*set the min. and max.*).
2. Use the same camera view, which can be done by:
3. Click **Adjust camera** → **Configure** → Name first view → click **current view** → OK → **Close**. *as shown in figure 1.33.*
4. Save the screenshots from ParaView®, which can be done by:
   (a) **File** → **Save screenshot**.
   (b) **Override Color Palette**, drop down menu → *choose* **Print** → Print.

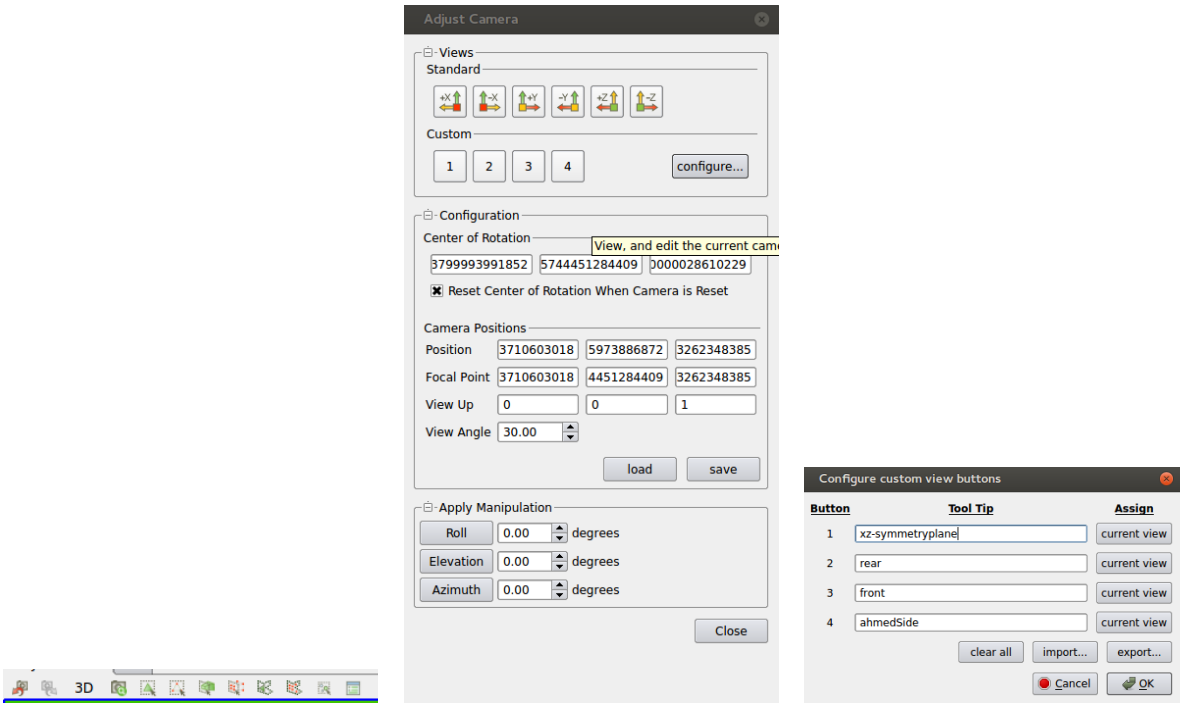


Figure 1.33: Adjusting the camera view.

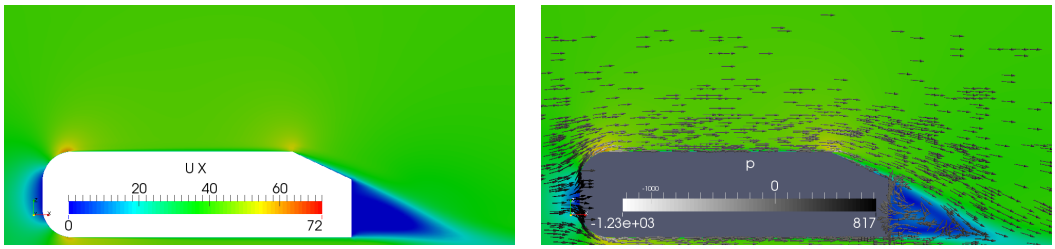


Figure 1.34: xz-symmetry plane showing the U-magnitude plots for the steady RANS simulation of mesh2.

Velocity Profiles

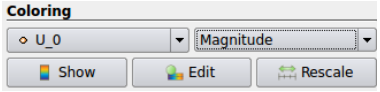
Mesh 2 results:

xz-symmetry plane:

xz-symmetry plane:

Velocity magnitude:

Velocity components: The velocity vector components can be plotted by changing the default settings from the drop down menu:



The surface vectors in figure 1.39 were plotted by following the tutorial in this website `http://goo.gl/57tLWj`.

yz-symmetry plane plots

Presented in this section the technique of plotting surface vectors along the yx-symmetry plane:

1. Click on the case file in the pipeline browser (*make it visible*). □

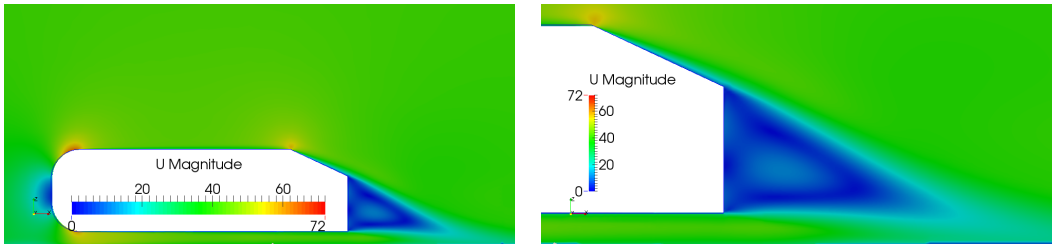


Figure 1.35: xz-symmetry plane showing the U-magnitude plots for the steady RANS simulation.

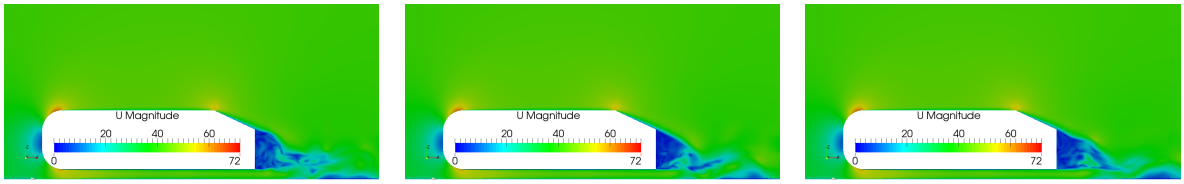


Figure 1.36: Transient DDES results of the velocity flow fields at: t_1= 4000.2s (left), t_2=4000.3s (middle), t_3=4000.4s (right)

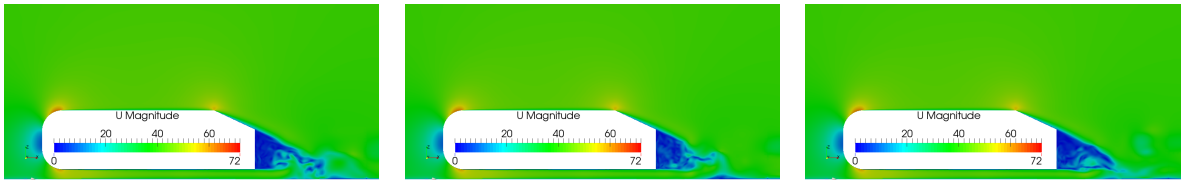


Figure 1.37: Transient DDES results of the velocity flow fields at: t_4= 4000.5s (left), t_5=4000.6s (middle), t_6=4000.7s (right)

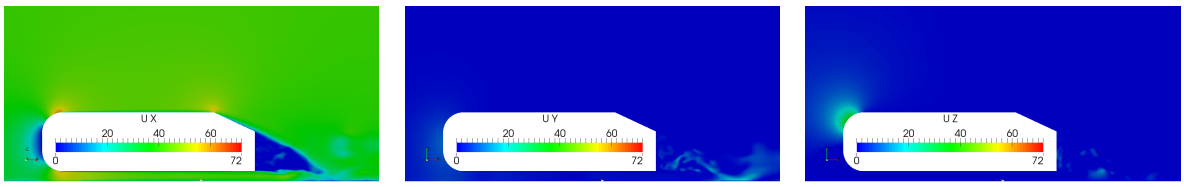


Figure 1.38: Transient DDES results of the velocity components at $t_6 = 0.7s$

2. Filters menu → Common → Slice. ☐
3. Set the origin so that you specify the x-coordinate. While, the y- and z-coordinates are set to zero → Apply. ☐
4. Filters → Alphabetical → Surface Vectors. ☐
   - Select input → U. ☐
   - Constraint Mode → Parallel. ☐
   - Apply. ☐
5. Filters → Common → Glyph. ☐
   - Vectors → U. ☐
   - Glyph type → Arrow. ☐
   - Scale Mode → off. ☐

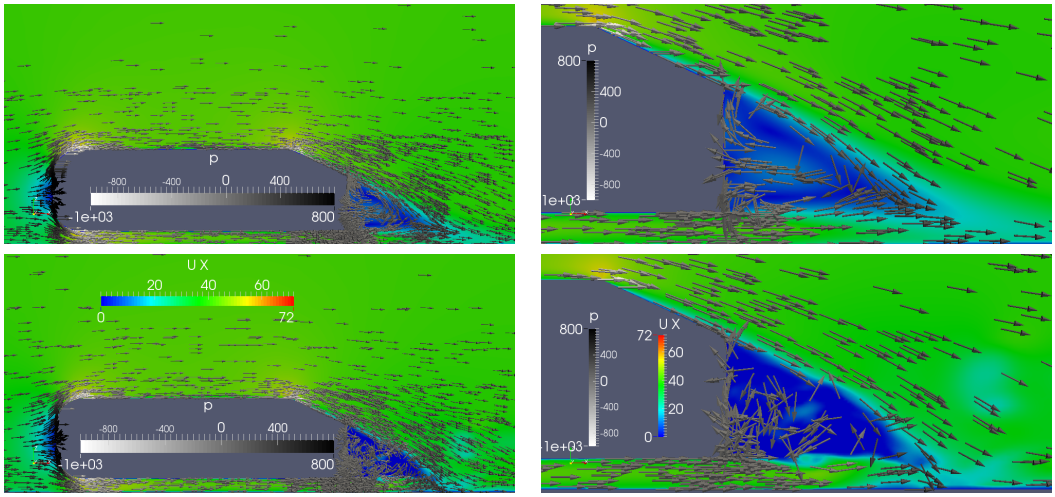


Figure 1.39: xz-symmetry plane showing $\frac{Pressure}{Density}$ plots for the steady RANS (top) simulation and transient DDES (bottom).

- Set Scale Factor → check Edit box → 0.05. ☐
- Maximum number of points (Do your judgement). ☐
- Coloring → pressure (point). ☐

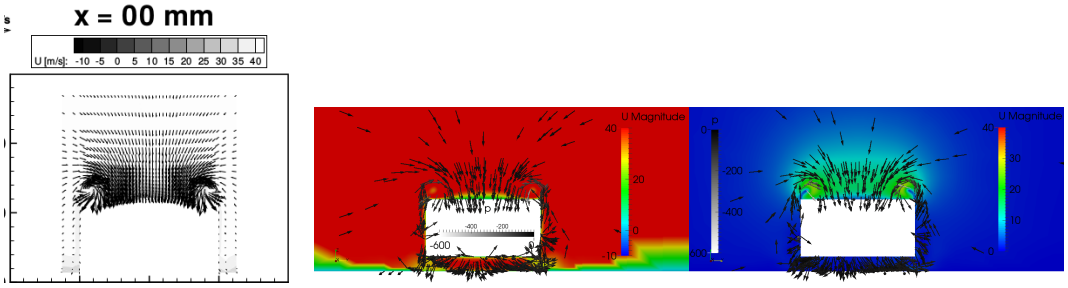


Figure 1.40: Velocity plots showing the vortex at (00mm) from the rear end, Lienhart results (left)[**Lienhart-paper**], steady RANS (middle), transient DDES at $t = 7s$ (right)

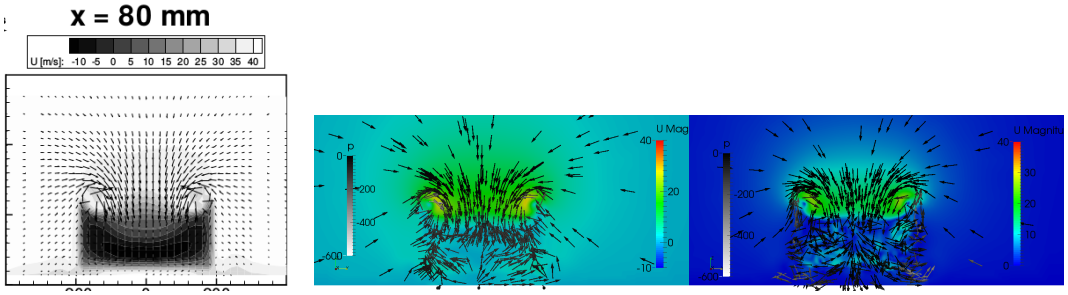


Figure 1.41: Velocity plots showing the vortex at (80mm) from the rear end, Lienhart results (left)[**Lienhart-paper**], steady RANS (middle), transient DDES at $t = 7s$ (right)

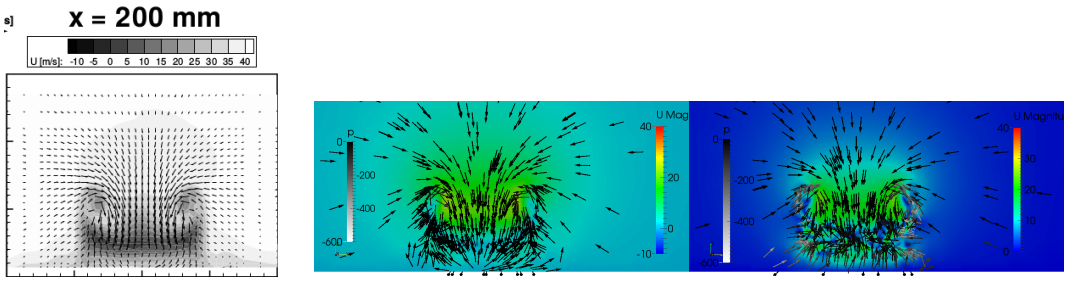


Figure 1.42: Velocity plots showing the vortex at (200mm) from the rear end, Lienhart results (left)[**Lienhart-paper**], steady RANS (middle), transient DDES at $t = 7s$ (right)

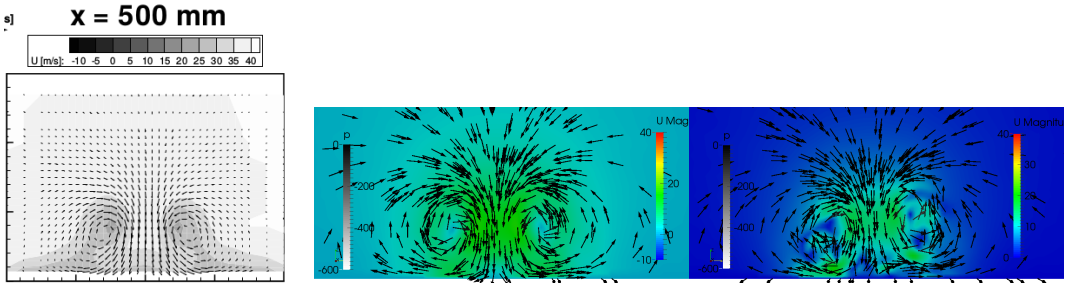


Figure 1.43: Velocity plots showing the vortex system at (500mm) from the rear end, Lienhart results (left)[**Lienhart-paper**], steady RANS (middle), transient DDES at $t = 7s$ (right)

Stream lines

The velocity stream lines were plotted as follows:

1. In the pipeline browser → click on the eye next to the case file (make it visible).
2. Filter menu → Stream tracer.
   - Vectors → U.
   - Integration direction → Both (forward and backward).
   - Integration type → Runge-Kutta 4-5.
   - Set the maximum streamline length to the highest value in the fixed range given.
   - Seed type → High resolution line source.
   - Tick Show line.
   - Set point 1 to the minimum bounding box of the car, while point 2 to the maximum bounding box of the car.
   - resolution → 100.
   - Representation → Surface.

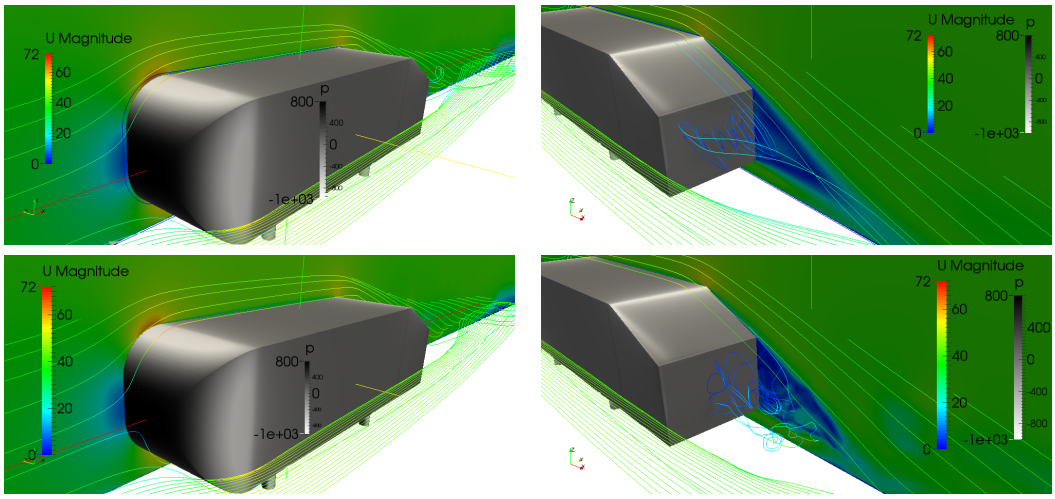


Figure 1.44: Velocity plots on the symmetry wall and the stream lines, and $\frac{Pressure}{Density}$ plots on the surface of the Ahmed body for the steady RANS (upper) and the transient DDES (lower).

Computational Time:

Grid	**Simulation**	**Wall-clock time; hh:mm:ss**	Max Vmem[14]	No. of iterations
coarse	S-A	3:57:35	5.587 GB	4,000
	DDES	45:00:00	7.50	7,000

Table 1.4: turnaround time for the simulations.

[14]Maximum amount of virtual memory

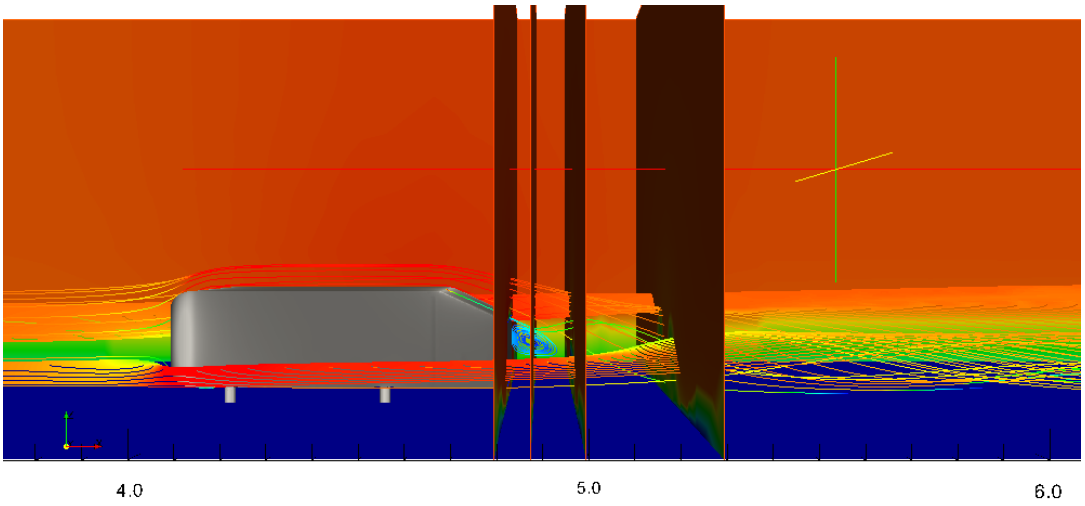


Figure 1.45: Velocity plots on the stream lines, yz-slices, ground and far-side.

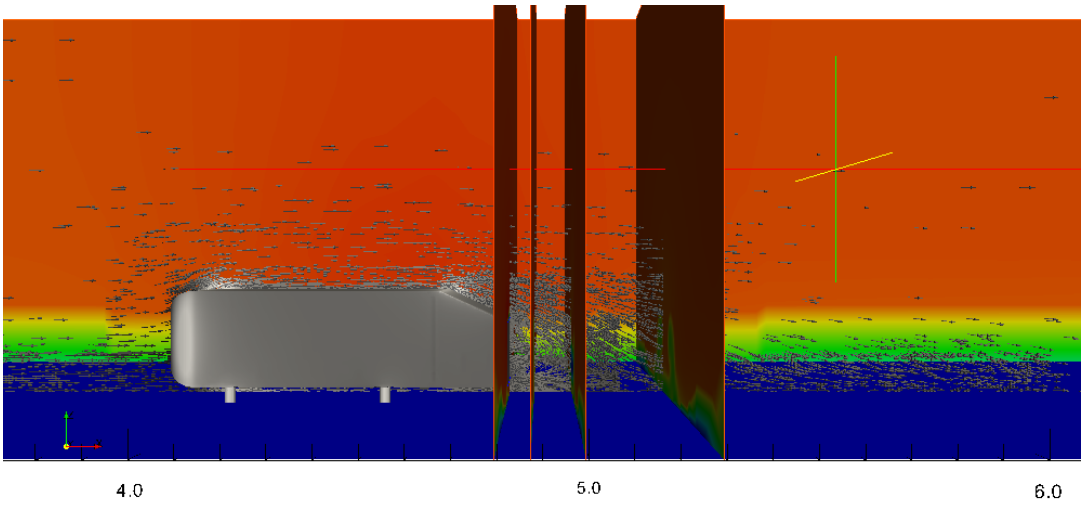


Figure 1.46: Pressure plots on the glyph vectors, velocity plots on the yz-slices, ground and far-side.

1.5.3 Mesh Refinement Analysis

Presented in this subsection a "guess" method used in the mesh refinement analysis which is based on plotting the velocity contours as shown in figures 1.47.

You need to refine in the regions of:

1. High gradient; where the contours are perpendicular to the surface:
   - Front nose.
   - Rear end.
2. Flow separation.

You Do not need to refine in the regions of:

1. Low gradient, where the contours are tangent to the surface:
   - The middle part of the Ahmed model (cuboid), where the flow travels with constant momentum. This is why the mesh is coarse compared to the nose and the rear end.
   - Far-fields (i.e. inlet, outlet, far-sides and top.)

y^+ plots

PLotting the y^+ over the mesh is useful in checking the regions where the y^+ falls below the range at which the wall function in OpenFOAM®is activated (i.e. $30 < y^+ < 300$) which in turn could be used for mesh refinement analysis. But, this utility is not calculated automatically with the simulation, you need to order OpenFOAM®to do so; which is usually done by the end of any RANS simulation with activated wall functions, by typing:

```
yPlusRAS
```

In case of running any LES approach, then type:

```
yPlusLES
```

In figure 1.48, the y^+ falls below 30 at the regions of low velocity, where separation takes place so the mesh was highly refined there (centre of nose and rear end). Apart from this, the average satisfy the range at which the wall function is activated which is shown in table 1.5.

Grid	minimum y+	maximum y+	average y+
Coarse	1.04445	1441.69	55.3169

Table 1.5: Calculated y+ values for the coarse grid.

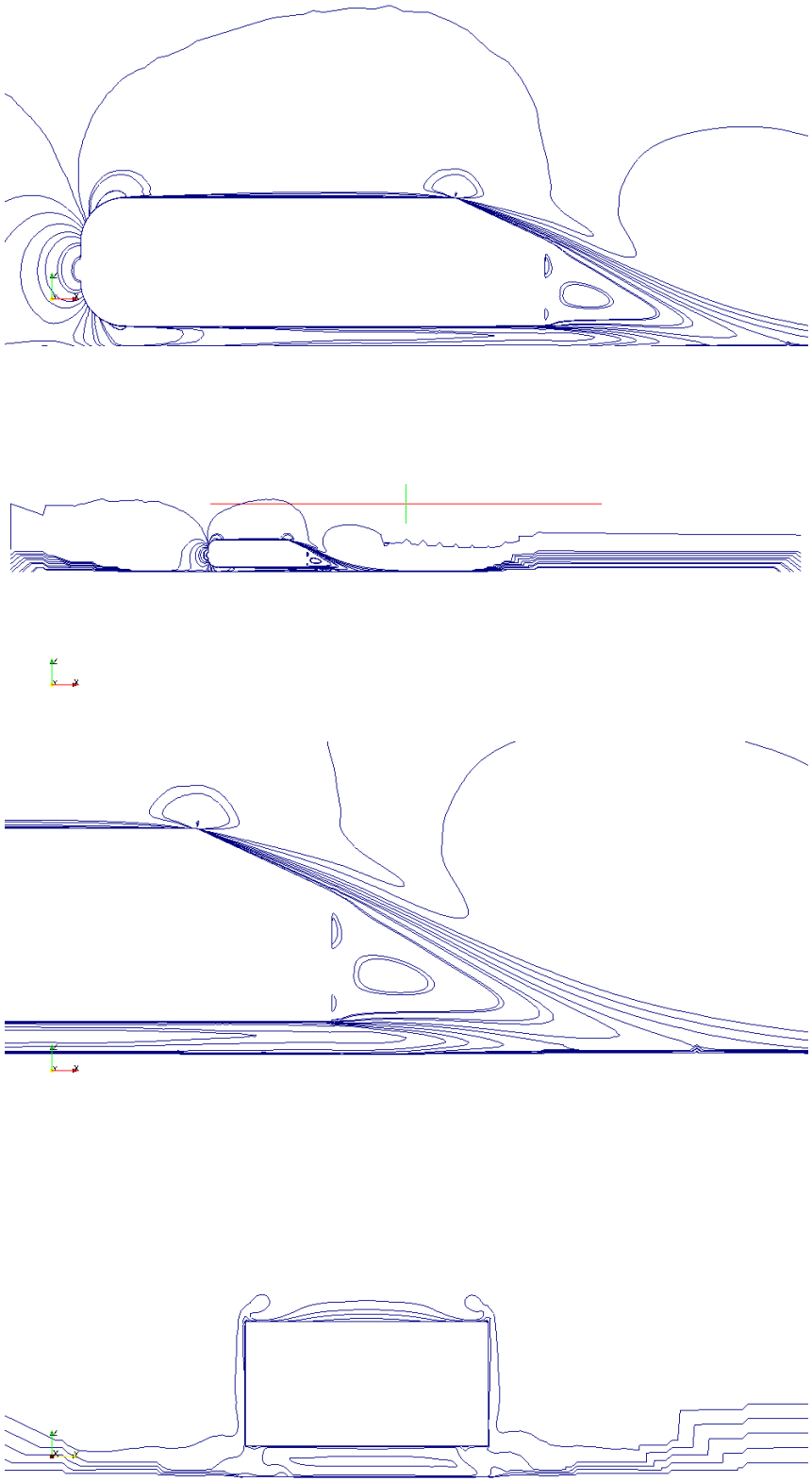


Figure 1.47: Velocity contours over the range ($0 \leq \mathbf{U} \leq 72$).

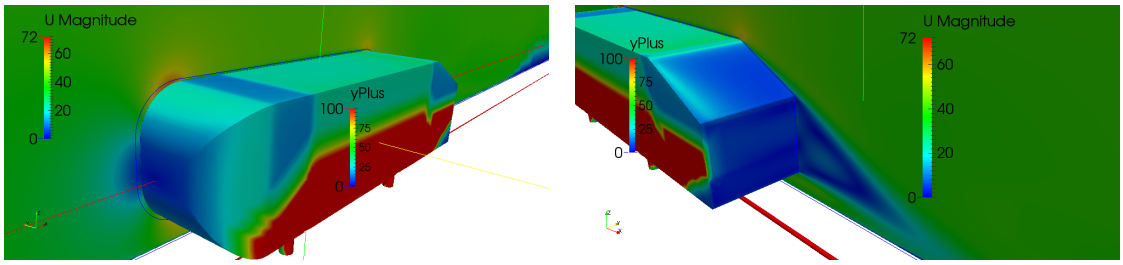


Figure 1.48: Velocity plots on the symmetry wall and y+ plots on the surface of the Ahmed body for RANS simulation.

Plotting the y^+ as shown figure 1.48 was done as follows:

1. From the terminal, in the case directory type `foamToVTK` this command will convert the calculated flow quantities into a VTK file.
2. Type `paraFoam &`, this will open ParaView®in a new window.
3. View all mesh parts → Apply.
4. Filter menu → Common → Slice. To be able to cut the plane in the xz-symmetry, you have to enter the coordinates. Here I used the origin as (0 0.9725 0) which lies at the centre of the computational domain, then, I specified the normal to be (0 1 0) which will apply a longitudinal cut perpendicular to the y axis.
5. File → Open → VTK → ahmed_body_endTime.vtk. This will import the car into the symmetry plane.

Mesh 2 refinement analysis

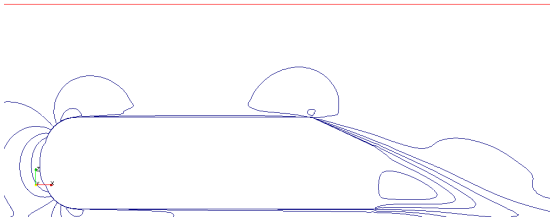


Figure 1.49: Velocity contours of mesh 2.

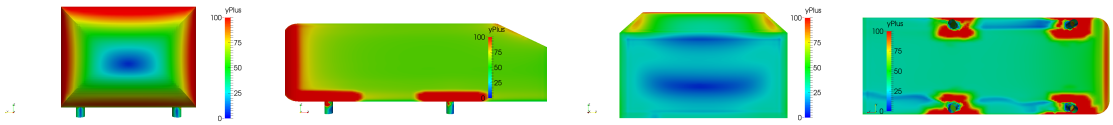


Figure 1.50: y+ plots of mesh 2.

Mesh 3 refinement analysis

1.6 Validation and Verification

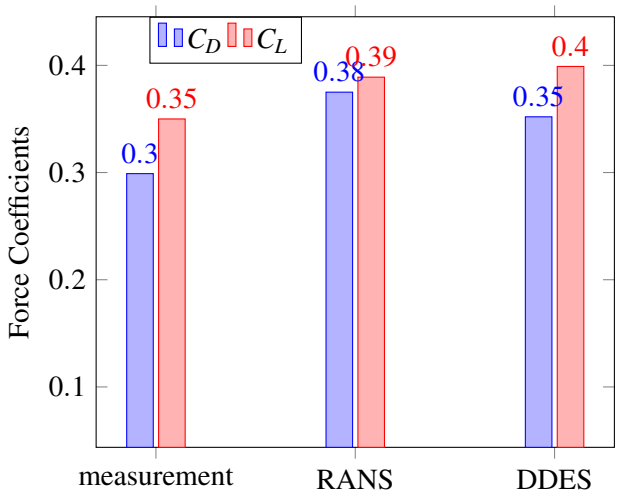


Figure 1.51: Force coefficients for the RANS and DDES simulations, measurement results obtained from [**openFOAM-ws**]

- Plot velocity over line using GIMP.

- Comparison between computational and experimental.

1.7 Useful links

Presented in this section a number of very useful URLs for learning and using OpenFOAM® and ParaView®.

1. Tommaso Luchhini, *Running OpenFOAM® in parallel*, Politecnico Di Milano, `http://tinyurl.com/qzv5x3b`.
2. Artur Lidtke, *Meshing in OpenFOAM®*, University of Southampton, `http://tinyurl.com/odavyc3`.
3. Nilsson H., Petit O., *Pre-processing in OpenFOAM®*, mesh generation, Chalmers University of Technology, `http://tinyurl.com/pfyzrzm`.
4. Hrvoje Jasak, *OpenFOAM®: Open Platform for CFD and Complex Physics Simulations*,Wikki Ltd, United Kingdom, `http://tinyurl.com/plqkxw2`.
5. Silva G., Arima G., Sousa F., *Mesh Generation in OpenFOAM® with SnappyHexMesh*, `http://tinyurl.com/q25n9ja`.
6. Nagy J., Introductory videos to CFD and OpenFOAM®,`http://tinyurl.com/q7bkb7c`.
7. Mesh Generation with the blockMesh utility `http://tinyurl.com/nlv6qe5`.
8. Brief introduction to OpenFOAM® `http://www.cfdyna.com/Home/OpenFOAM.html`.
9. All OpenFOAM® utilities from the user guide, `http://tinyurl.com/psq6jwm`.
10. Meshing through HELYX-OS, `http://tinyurl.com/oe4th5y`.
11. Comments of the senior members, Mesh decomposition, `http://tinyurl.com/pmxcx4v`.
12. Johnson G., *Visualization with ParaView®*, The university of Texas At Austin, Texas Advanced Computing Center, `https://goo.gl/SffXKF`.

II Aerospace

2. Onera M6 Wing

2.1 Geometry

The geometry was prepared by Mateusz Gugala[1] in Unigraphics NX®. Thanks to him for sharing it. In addition, Mateusz Gugala has shared how he created the geometry in the aforementioned software, such as:

1. Download the airfoil data points of the tip and root of the Onera M6. ☑
2. Import the root airfoil points into the CAD system. ☑
3. Remove the trailing edge point, so that you end up having a blunt trailing edge as can be seen in this picture `http://tinyurl.com/nbobzoz`. ☑
   See problem () for further explanation as why doing this step is worth considering for a better mesh quality.
4. Do not close the blunt trailing edge now. ☑
5. Repeat the second step for the tip by scaling the root. ☑
6. Blend the tip with the root. ☑
7. Close the blunt trailing edge. ☑
8. Export the geometry file to a STEP format. ☑

2.2 ANSYS ICEM CFD

The geometry was imported into ICEM CFD as illustrated in figure 2.1. Then, I would suggest that you follow some steps (Starting the project steps and step 1 all) from the ANSYS ICEM CFD Tutorial Manual (15.0, November 2013) for the “tetra/prism mesh in a fin configuration”. You will find “LMB” and “RMB” these are the short hand for left and right mouse button respectively.

2.2.1 Mesh Generation

Presented in this subsection the process of setting up the mesh parameters for the M6 case, as follows:

[1] PhD student, School of Engineering and Material Science

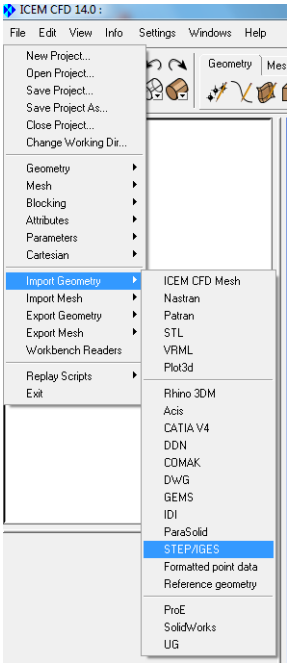


Figure 2.1: Importing the M6 geometry into ICEM CFD.

1. Assuming that you did Step 1 from the **ANSYS ICEM CFD Tutorial Manual** arriving at a geometry as shown in figure 2.2, which did not have any tolerance problem (if there was any tolerance problem, then the edges would be coloured in blue or yellow, and the tolerance is kept to the minimum when repairing the geometry). ☐
2. Click on the **Mesh** tab → Global Mesh Setup. ☐
3. Set the scale factor to 1 and the seed size to 4096. ☐
   The global element seed size is large when compared to the Ansys tutorials. Because, this geometry was not scaled to meters but, rather left as mm which is the default scaling in most CAD packages.
4. Mesh type → All Tri, mesh method → Patch Dependent. ☐
5. Volume meshing parameters → tetra/mixed, tetra/mixed meshing → Delaunay. ☐
6. Under the **Mesh** tab, click on **Part Mesh Setup** ☐.
   - **Maximum size** specifies the maximum size of each cell in the selected area.
     - For the **far-field patches** (i.e. top, ground, far-side, inlet and outlet); choose a high value which will result in a coarse, since there is not much information being transferred there other than the free-stream value and we are not interested in this,.
     - For the **Symmetry** patch (i.e. patch where the wing is fixed); choose a value smaller than for the farfield by a factor of ≈ 1.2.
     - For the **Wing parts:**
       * Leading edge; choose a very small value (this should be $\approx$ 1.5k less than the value specified to the far-field).
       * Trailing edge; choose a slightly higher value ($\approx$ 10 times higher than for the leading edge).
       * Tip; choose a slightly higher value than the leading edge ($\approx$ 5 times greater).
       * Upper and lower surfaces; $\approx$ double the value specified for the trailing edge.
       * Roots; make it $\approx$ 50 times greater than the value specified for the leading edge.
   - **Tetra size ratio** specifies to how much each cell can grow with respect to its neighbouring ones.

  - A value of 1.2 was only specified to few patches, such as:
    * Upper and lower wing surfaces (i.e. span).
    * Symmetry plane.
    * Tip.
- **Height** is the minimum thickness of the first prism layer.
  - This can be specified using a y+ calculator or manually (as described in the case of the Ahmed model, see section 1.5.1 or any y+ calculator found online, here is one `http://goo.gl/GVxlcP`).
- **Height ratio** is the stretching ratio; which is the ratio of two adjacent boundary layers (further/nearer).
  This can be calculated automatically in ICEM CFD, as follows:
  Mesh setup → Prism meshing parameters:
  Growth law → exponential.
  Specify the rest. Then, click on compute params.
- **Num layers** specifies the number of boundary layers (≈10-12).
- Just for the wing parts, make sure to tick their prism boxes in the **part mesh setup**.

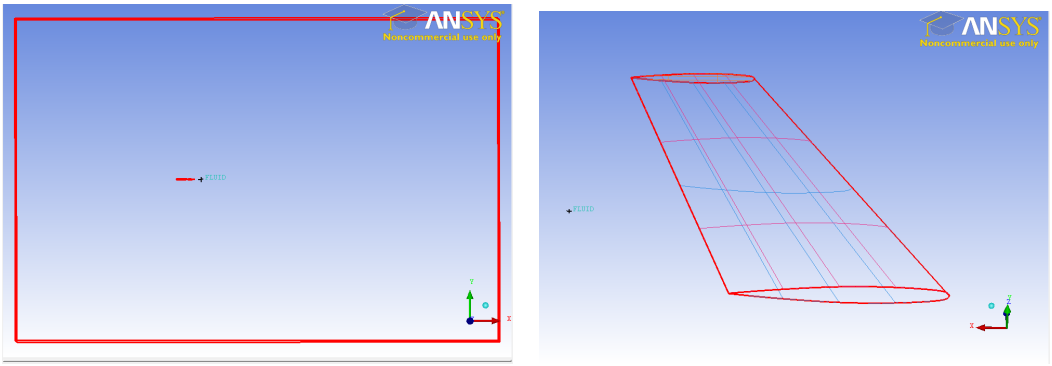


Figure 2.2: The M6 geometry file passed all tolerance checks.

Exporting the mesh from ICEM to .msh file to be able to import it in FLUENT®.

1. Click on output tab.
   - Select solver → Choose ANSYS FLUENT® from the drop down menu → apply.
   - Click on write input icon (from the output tab) and save.
   - Exit ICEM CFD.

2.2.2 Meshing journey

■ **Attempt 2.1** Presented here, the very first mesh I generated in ICEM CFD for the m6. By looking at figure 2.3, the mistakes I did were:

- Fine computational domain. (It should be much coarser.)
- Created prism layers before having a smooth mesh. (Adding boundary layers should be the last step.)
- Coarse leading and trailing edge. (I should have clustered a bit more into the leading and trailing edges.)
- Poor prism layers at trailing edge. (the wing geometry has a sharp edge. For a good quality there should be a blunt edge. For a better quality, a rounded trailing edge instead.)

■ **Attempt 2.2** Presented here, the second mesh I created in ICEM CFD after receiving the feedback

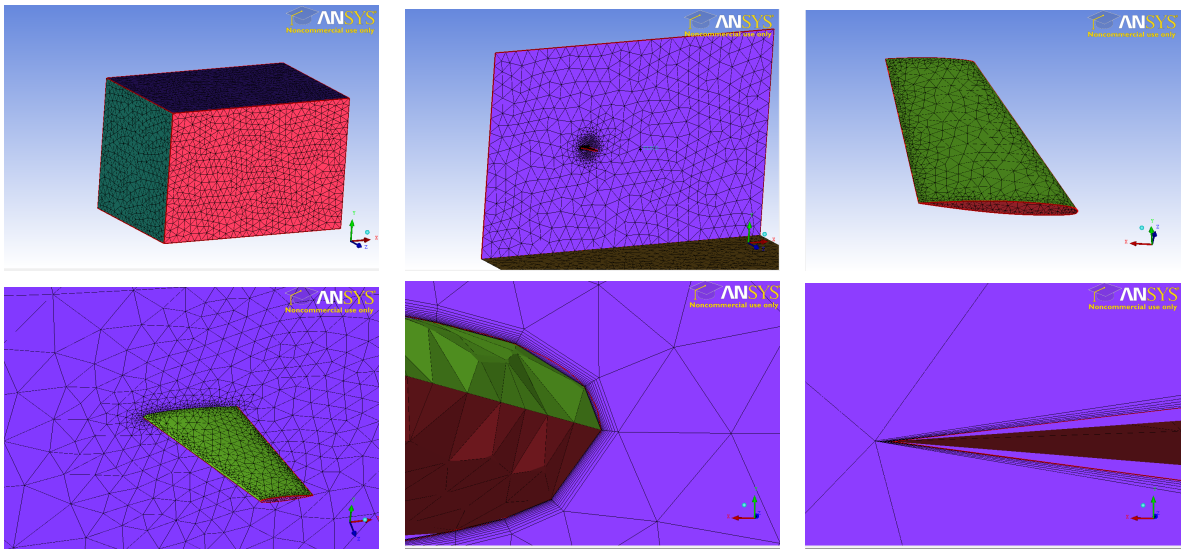


Figure 2.3: First mesh generated in ICEM CFD; enclosure (top left), symmetrical+wing (top middle), wing (top right), wing+symm (bottom left), prism layers at leading and trailing edge (bottom middle and right respectively).

from Dr. J-D. Müller and Mr. Mateusz Gugala. By looking at figure 2.5. The mistakes in this mesh are:

- Leading edge still coarse. (The number of points should be doubled there.)
- Farfield still fine. (I should have increased the maximum size of the cells there.)
- The area where the wing meets the symmetry should not be refined. (I should've avoided drawing density lines in this region.)
- The centre of the wing should be more coarser.
- The rate to which the mesh is coarsened from the wing is high. (It should be lower.)

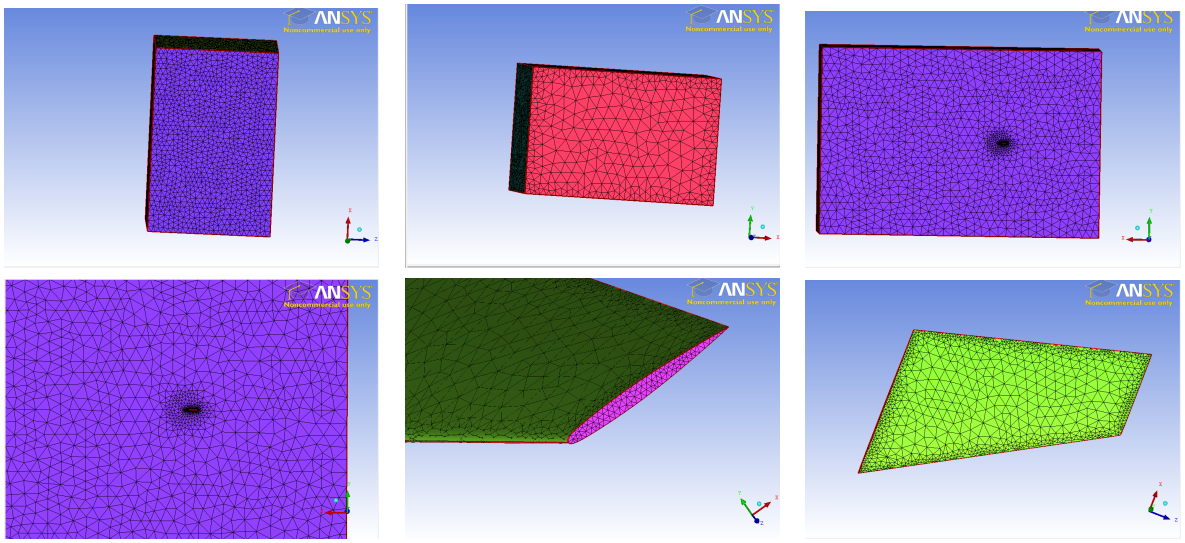


Figure 2.4: Second mesh generated in ICEM CFD; enclosure (all top), wing+symm (bottom left), wing tip (bottom middle) and wing (bottom right).

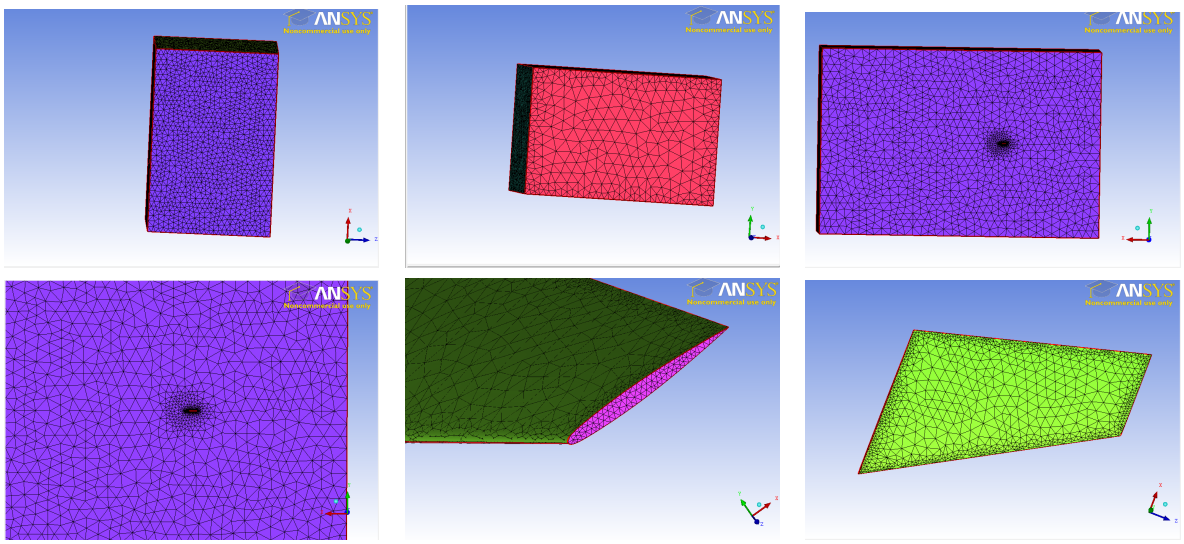


Figure 2.5: Second mesh generated in ICEM CFD; enclosure (all top), wing+symm (bottom left), wing tip (bottom middle) and wing (bottom right).

■ **Attempt 2.3** Presented here, the third mesh generated in ICEM CFD. Infer from figure 2.6, the mistakes are:

- Symmetry plane is slightly fine. (It should be more coarser though.)
- The growth rate of the cells near the aerofoil (at the symmetry plane) is still high. (The tetra size ratio should be in the range of $1.1 \approx 1.2$.)

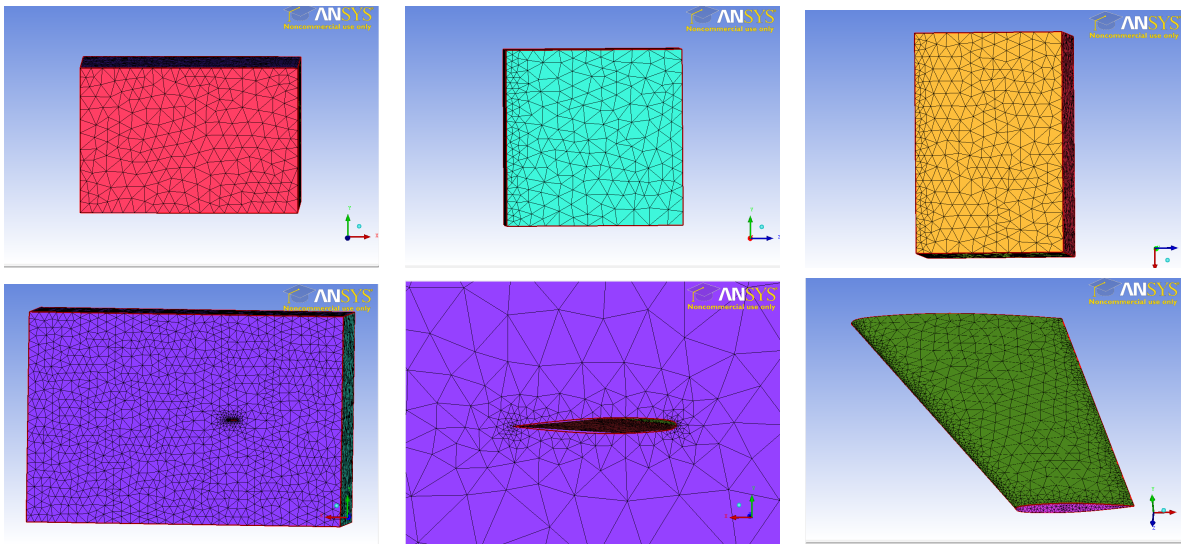


Figure 2.6: Third mesh generated in ICEM CFD; far-side (top left), inlet (top middle), ground (top right), symmetry plane (bottom left), symm. near aerofoil (bottom middle) and wing (bottom right).

■ **Attempt 2.4** Presented here, the fourth mesh generated in ICEM CFD. Infer from figure 2.9, the mistakes are:

- High mesh growth rate at the wing. (It should be reduced so that the cells coarsens slowly towards the outer boundaries.)

- The nibbled edge at the trailing edge of the wing. (The basic spacing of the nodes should be improved.)
- The leading edge is below that standard threshold in terms of the number of nodes. (The number of nodes should be doubled.)
- The size of elements at the wing surface are a bit higher than the standard threshold of the element size. (Decrease the size of elements by a factor of $\approx$ 1.3-1.5.)
- The growing ratio is still high at the wing surface patch. (Reduce the tetra size ratio to 1.2.)
- Trailing edge is coarse. (It should be refined by a factor of $\approx$ 1.5-2 than the current scale.)

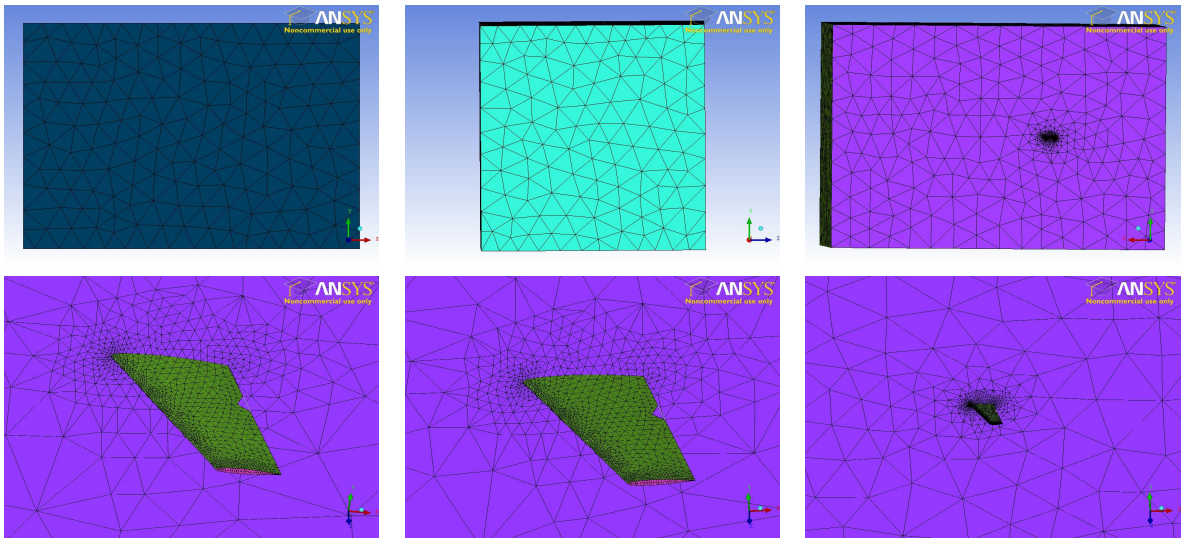


Figure 2.7: Fourth mesh generated in ICEM CFD; far-field (top left), inlet (top middle), symmetry plane (top right), symm. near aerofoil (bottom left and middle) and wing (bottom right).

■ **Attempt 2.5** Presented here the fifth mesh generated in ICEM CFD; the feedback from Dr. J-D. Müller and Mr. Mateusz Gugala are as follows:

- The wing tip is still coarse. (It should be refined.)
- The growth rate should be adjusted. (Change the tetra size ratio to 1.2)

■ **Attempt 2.6** Presented here the sixth mesh generated and the final one which is shown in figure

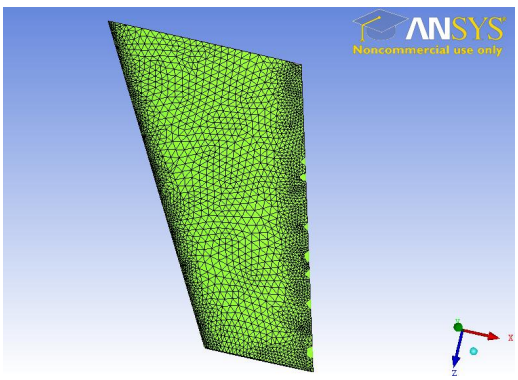


Figure 2.8: 5th mesh generated in ICEM.

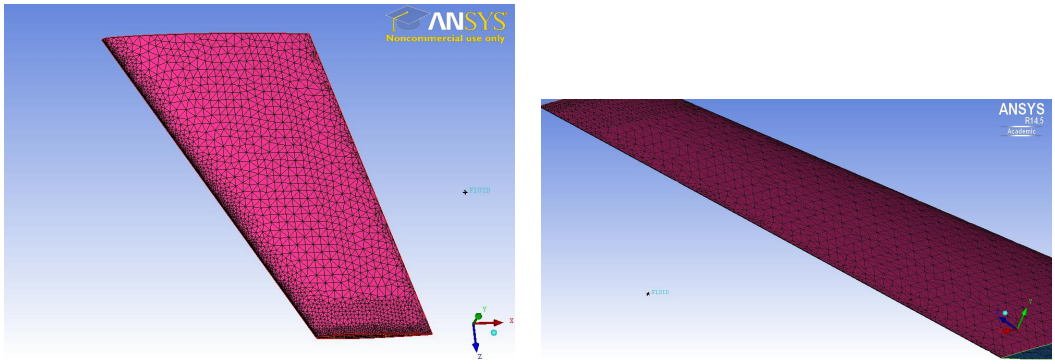


Figure 2.9: 6th mesh generated in ICEM; view from upper surface (left), view from trailing edge (right).

2.9.

■ **Attempt 2.7** The final attempt is just the refined version of mesh 6 and is shown in figure 2.10. It contains 776,741 cells and 151122 nodes.

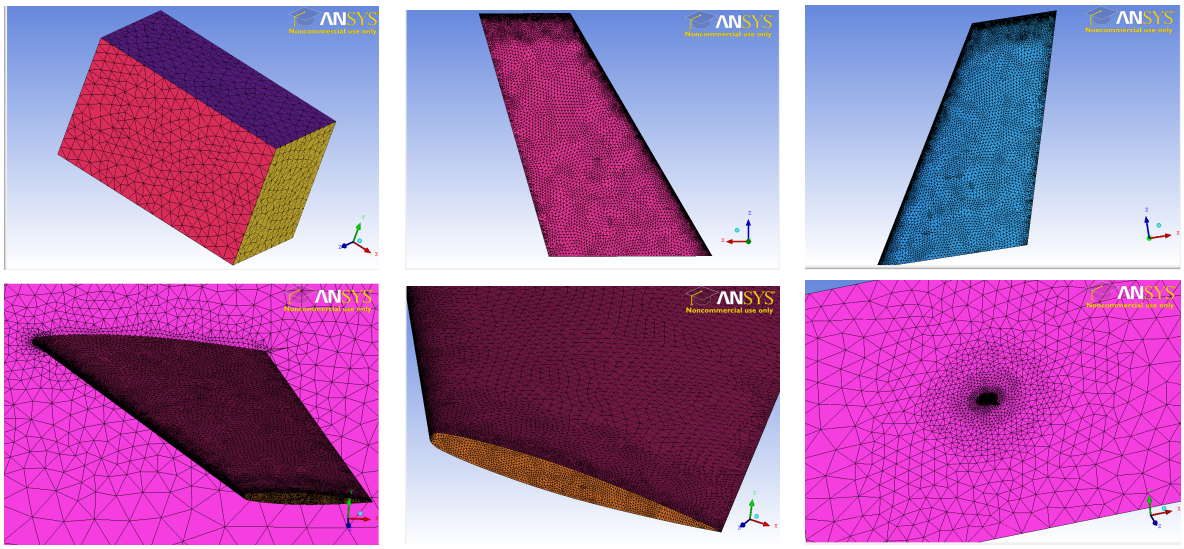


Figure 2.10: A fine mesh

2.3 FLUENT®

Presented in this section, the procedure of setting the case for an Euler run.

2.3.1 Case Setup

1. Import the mesh generated in ICEM CFD into FLUENT®. ☐
   File → **Read mesh**.
2. Check your mesh quality. ☐
   Mesh → **Check**.

Mesh → **Report Quality**.

3. Choose the flow type. ☐
   Models → **Viscous** → Edit → Inviscid.
4. Check the materials used for the fluid and the wing. ☐
   - **Fluid** → air *Create*/*Edit* check the quantities.
   - **Solid** → Aluminium.
5. Define the boundary conditions: ☐
   - Inlet, outlet, top, ground and far-side
     Type → pressure far side.
   - Symmetrical side: ☐
     Type → **symmetry**.
   - Wing parts: ☐
     Type → Wall.
6. Define the reference values to form coefficients: ☐
   Compute from → inlet.
   Area → 0.7532 m^2.
   Length → 0.64607m.
   Velocity → 272.1002.
7. Define the **Solution Methods**: ☐
   Pressure-Velocity Coupling → SIMPLE.
   Gradient → Least Squares Cell Based.
   Pressure → Linear.
   Momentum → First Order Upwind.
8. Specify the solution controls (the sum of the pressure and the momentum should be equal to 1): ☐
   Under-Relaxation factors:
   - Pressure → 0.3.
   - Density → 1.
   - Body Forces → 1.
   - Momentum → 0.7.
9. Create the Force coefficient plots: ☐
   Create drop down menu → Drag.
   Check the following boxes:
   - Print to Console.
   - Plot.
   - Write.
   - Highlight Zones

   Specify the force vectors for an angle of attack α = 3.06° such as:
   - X: cos3.06
   - Y: sin3.06
   - Z: 0

   Define the wall zones:
   Highlight all wing profiles (upper, lower etc..).
   click OK.
   Similarly, most of the steps are same for the lift coefficient. The only difference is in the force vectors, for C_L:
   - X: -sinα
   - Y: cosα

- Z: 0

10. Initialise the solution: □
    Initialization Methods → Standard Initialization.
    Compute from → Inlet.
    Reference frame → Relative to Cell Zone.
    Initial Values:
    - Gauge pressure → 0.
    - X-Velocity (m/s) = 271.7188.
    - Y-Velocity (m/s) = 14.53063.
    - Z-Velocity (m/s) = 0.

    Click Initialise.
11. Specify the write interval of the calculated quantities: □
    Autosave every (iterations) → *numberofiterations*.
12. Control the run time: □
    Before attempting to run the case, check your case from any errors by clicking on **Check Case...**. **Number of iterations** → 300.

2.3.2 Monitoring your job

2.4 Post Processing

2.4.1 Mesh Refinement Analysis

2.5 Validation and Verification

2.6 Useful links

1. More information related to the Onera M6 Wing can be found here: `http://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing.html`.
2. Tetra/Prism meshing in ICEM CFD, `https://www.youtube.com/watch?v=C1Yw_pYaPGE` and `https://www.youtube.com/watch?v=SdUjpjwUnew`.
3. Setting Ansys Fluent launcher `http://www.arc.vt.edu/ansys_help/flu_gs/flu_ug_sec_mini_launcher.html`.
4. Setting up the CFD Simulation in ANSYS FLUENT®`http://www.arc.vt.edu/ansys_help/flu_tg/param_fluent.html`.
5. Setting up the FLUENT®launcher when running on Linux clusters (i.e. Apocrita) `http://www.arc.vt.edu/ansys_help/flu_ug/flu_ug_sec_launcher_options_remote_parallel.html`
6. How to control the growth rate in ICEM CFD, `http://www.cfd-online.com/Forums/ansys-meshing/85951-growth-rate-mesh-icem.html`.

Appendices

SnappyHexMeshDict

```
{
    ahmed-metres.stl
    {
        type triSurfaceMesh;
        name ahmed;
                regions
                {
                     stilts              // Named region in the STL file
                     {
                            name ahmed_stilts; //user-defined patch name
                      }
                }
     }
refinementBox1 { type searchableBox;
min (4.5755 0.73 0.0); max (4.8875 1.14 0.36075);
}
refinementBox2 { type searchableBox;
min (3.6915 0.73 0); max (5.31 1.14 0.3705);
}
refinementBox3 { type searchableBox;
min (3.6525 0.73 0); max (5.349 1.14 0.4095);
}
refinementBox4 { type searchableBox;
min (4.8875 0.73 0.0); max (5.1 1.14 0.24418);
}
};
```

Figure 11: Defining names for the different patches in the .stl file and the determining the bounding refinement boxes.

```
// Which of the steps to run on the geometry
castellatedMesh true;
castellatedMeshControls
{
        // Refinement parameters
maxLocalCells 6000000;
maxGlobalCells 8000000;
minRefinementCells 2;
maxLoadUnbalance 0.01;//0=balance always. (Perfect balance= no. of cells/no
    . of refinement processes)
nCellsBetweenLevels 1; //A bigger number means more iterations for the
    buffer layers to grow between different levels of refinement. 1 means
    normal
        // Explicit feature edge refinement
         features
        (
                        {
                file "ahmed-metres.eMesh";
                level 2;
                        }
          );
         // Surface based refinement
           refinementSurfaces
         {ahmed
         {
                level (4 6);
                regions
                {
                 stilts
                    {
                        level (6 8);

                        patchInfo
                        {
                                type wall;
                        }
                    }
                 }
        }
}
    resolveFeatureAngle 30;
```

Figure 12: Castellated mesh generation parameters part 1 of 2.

```
// Region-wise refinement
    refinementRegions
    {
        refinementBox1
        {
            mode inside;
            levels ((0.1 6));
        }
        refinementBox2
        {
            mode inside;
            levels ((0.1 5));
        }
        refinementBox3
        {
            mode distance;
            levels ((0.1 4) (0.3 3) (0.4 2));
        }
        refinementBox4
        {
                mode inside;
                levels ((0.1 6));
        }
     }
      // Mesh selection
     locationInMesh (4.79 1.13 0.338); //Chosen to be very slightly away
         from the max. bounding box of the geometry.
    allowFreeStandingZoneFaces true;
    }
```

Figure 13: Castellated mesh generation parameters part 2 of 2.

```
// Which of the steps to run on the geometry
snap            true;
// Settings for the snapping.
snapControls
{
nSmoothPatch 5;
tolerance 5;
nSolveIter 100;
nFeatureSnapIter 15;
implicitFeatureSnap false;
explicitFeatureSnap true;
multiRegionFeatureSnap true;
}
```

Figure 14: Surface-recovering “snapping” parameters.

```
// Which of the steps to run on the geometry
addlayers             true;
//Setting for the layer addition.
addLayerControls
{
relativeSizes true; //True if the boundary layer thickness and the mesh
    size are relative to the cells spacing of the boundary stack. Otherwise
    , false.
    layers
    {
        "(ahmed|stilts).*" //State the patches where you want the layers to
             grow on.
        {
            nSurfaceLayers 8;
        }
    }
    expansionRatio 1.5; //Ratio of the layers growth.
    finalLayerThickness 0.8; //Ratio of the layer height to it width.
    minThickness 0.2; // Minimum layer thickness. If you've chosen a v.
        small no. the layers won't grow.
    nGrow 6; // Facilitates the convergence of this stage.
    featureAngle 60; // flat surface=0, perpendicular faces=90.
    nRelaxIter 3; //try it with 1 and see the difference.
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5; //This will order the process to stop the
        layers growth on highly curved surfaces.
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 300; //The number of error reduction iterations will stop if
         this number is reached. Even if there is still errors in the layer
         growth process.
}
```

Figure 15: Boundary-layers addition parameters.

```
meshQualityControls
{
maxNonOrtho 180; //set to 180 to disable.
maxBoundarySkewness 20; //Set to <0 to disable.
maxInternalSkewness 4; //set to <0 to disable.
maxConcave 80; // 0 is straight face, <0 is convex face. 180 to disable.
minFlatness 0.5; // Set to -1 to disable.
minVol 1e-13; // Set to -1E30 to disable.
minTetQuality -1; //Set to 1-30 to disable.
minArea -1; //Set to <0 to disable.
minTwist 0.02; //set to <-1 to disable.
minDeterminant 0.001; //Set to  -1 for hex cells,  <=0 for folded or
    flattened illegal cell.
minFaceWeight 0.02; // 0 < Setting range < 0.5.
minVolRatio 0.01; //  0< Setting range < 1.
minTriangleTwist -1; // V.Imp. for Fluent compatibility, this must be > 0.
nSmoothScale 4;
errorReduction 0.75;
}
debug 0; // 0 will only write the final mesh.
mergeTolerance 1E-6;
```

Figure 16: Mesh quality controls parameters.

Time directory files

```
flowVelocity                    (40.0 0.0 0.0);
pressure                        0;
nut                             0.0015;
nuTilde                         0.000075;
#inputMode                      merge
```

Figure 17: This file lies in `/case_folder/0/initialConditions`.

```
inlet
{
    type  fixedValue;
    value $internalField;
}
```

Figure 18: This file lies in: `/case_folder/0/include/fixedInlet`.

```
sides
{
        type            symmetry;
}

top
{
        type    symmetry;
}
```

Figure 19: This file lies in: `/case_folder/0/include/sidesTopPatches`

```
FoamFile //It is worth checking the headings (i.e. location and object) of
    the foam files.
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#include                ``include/initialConditions''
dimensions              [0 1 -1 0 0 0 0] // Check this [kg m s K kgmol A cd
    ]. Hence it is m/s.
internalField           uniform $flowVelocity
boundaryField
{
        #include ``include/fixedInlet''
        outlet
        {
                type            inletOutlet;
                inletValue      uniform (0.0 0.0 0.0);
                value           $internalField;
        }
        ahmed_body
        {
                type            fixedValue;
                value           uniform (0.0 0.0 0.0);
        }
        ahmed_stilts
        {
                type            fixedValue;
                value           uniform (0.0 0.0 0.0);
        }
        ground
        {
                type            fixedValue;
                value           uniform (0.0 0.0 0.0);
        }
        #include        ``include/sidesTopPatches''
}
```

Figure 20: Velocity foam file

```
FoamFile //It is worth checking the heading of the foam files.
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#include                ``include/initial Conditions''
dimensions              [0 2 -2 0 0 0 0]; //Check this [kg m s K kgmol A cd
    ]. Hence it is m^2.s^-2 instead of N.m^2.s^-2.
internalField           uniform $pressure;
boundaryField
{
        inlet
        {
                type            zeroGradient;
        }
        outlet
        {
                type            fixedValue;
                value           $internalField;
        }
        ground
        {
                type            zeroGradient;
        }
        ahmed_body
        {
                type            zeroGradient;
        }
        ahmed_stilts
        {
                type            zeroGradient;
        }
        #include        ``include/sidesTopPatches''
}
```

Figure 21: Pressure foam file.

```
FoamFile //It is worth checking the heading of the foam files.
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#include                ''include/initial Conditions''
dimensions              [0 2 -1 0 0 0 0]; // Check this [kg m s K kgmol A
    cd]. Hence, it is m^2s^-1.
internalField           uniform $nut;
boundaryField
{
        inlet
        {
                type                    freestream;
                freestreamValue         uniform $nut;
        }
        outlet
        {
                type                    freestream;
                freestreamValue         uniform $nut;
        }
        ground
        {
                type            nutUSpaldingWallFunction; //Type of the
                    wall function used differs for each turbulence
                    modelling approach.
                value   uniform 0.0;
        }
        ahmed_body
        {
                type            nutUSpaldingWallFunction;
                value           uniform 0.0;
        }
        ahmed_stilts
        {
                type            nutUSpaldingWallFunction;
                value           uniform 0.0;
        }
        #include        ''include/sidesTopPatches''
}
```

Figure 22: ν_t foam file.

```
FoamFile ////It is worth checking the heading of the foam files.
{
    version 2.0;
    format ascii;
    class volScalarField;
    location "0";
    object nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#include                ''include/initial Conditions''
dimensions              [0 2 -1 0 0 0 0]; //Check this [kg m s K kgmol A cd
    ]. Hence, it is m^2s^-1.
internalField           uniform $nuTilda
boundaryField
{
        inlet
        {
                type                    freestream;
                freestreamValue         uniform $nuTilda;
        }
        outlet
        {
                type                    freestream;
                freestreamValue         uniform $nuTilda;
        }
        ground
        {
                type            fixedValue;
                value           uniform 0;
        }
        ahmed_body
        {
                type            fixedValue;
                value           uniform 0;
        }
        ahmed_stilts
        {
                type            fixedValue;
                value           uniform 0;
        }
        #include        ''include/sidesTopPatches''
}
```

Figure 23: $\tilde{\nu}$ foam file.

```
FoamFile //Don't forget to change the ``object'' from ``nut'' to ``nuTilda
    ''
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
``include/initial Conditions''
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform $nuTilda;

boundaryField
{
        inlet
        {
                type            fixedValue;
                value   uniform $nuTilda;
        }
        outlet
        {
                type            inletOutlet;
                        inletValue              uniform $nuTilda;
                        value           uniform $nuTilda;
        }
        ground
        {
                type            fixedValue;
                value           uniform 0;
        }
        ahmed_body
        {
                type            fixedValue;
                value           uniform 0;
        }
        #include        ``include/sidesTopPatches''
}
```

Figure 24: $\tilde{\nu}$ file for DDES simulation

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuSgs;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
``include/initial Conditions''
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform $nuSgs;
boundaryField
{
        inlet
        {
                type            zeroGradient;
        }
        outlet
        {
        type            zeroGradient;
        }
        ground
        {
                type            nutUSpaldingWallFunction;
                value           uniform $nuSgs;
        }
        ahmed_body
        {
                type            nutUSpaldingWallFunction;
                value           uniform $nuSgs;
        }
        ahmed_stilts
        {
                type            nutUSpaldingWallFunction;
                value           uniform $nuSgs;
        }
        #include        ``include/sidesTopPatches''
}
```

Figure 25: $\nu_{Sgs} file for DDES$.

Mesh Decomposition

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 12; // number of CPU cores.
method          simple; // Here I activated simple and deactivated the rest.
simpleCoeffs
{
        n               (3 2 2); // Multiply the no. of cuts in the x,y and
             z = no. of CPU cores.
        delta           (0.001);
}
hierarchicalCoeffs
{
        n               (1 1 1); // Here I specified equal order of cuts in
             the x,y and z.
        delta           (0.001);
        order           xyz;
}
scotchCoeffs
{
processorWeights // Here I specified equal weights for every processor.
(
1
1
1
1
1
1
1
1
1
1
1
1
);
}

manualCoeffs
{
    dataFile        "cellDecomposition";
}
distributed     yes; //This means that you want to distribute your case on
    different directories.
roots
12Roots //How many directories you want to save your data on?
(
        ''Directory for node 0'' //This node is the master.
        ''Directory for node 1''
        etc...
);
```

Figure 26: Descriptive dictionary showing the settings of different mesh-cutting scenarios.

.0.1 File scripts

```
#!/bin/sh
#$ -cwd -V //Set the working directory for the job to the current directory
#$ -pe openmpi 1 //Ordering a whole node.
#$ -l h_rt=10:0:0 //Do you want your simulation to run for 10 hrs?
#$ -l h_vmem=24G //Requesting 24Gb of RAM (as 1 node=12cores=24Gb)
#$ -m base //Get an email at the Beginning of the job, Alert, Something
    that I might have forgotten and as it Ends.
#$ -M email_address //Your email address where the ''base'' can be sent to.
#$ -N Job_name // User-defined job name.
decomposePar //Order OpenFOAM to decompose your mesh.
foamJob -parallel -screen snappyHexMesh -overwrite //parallel meshing
reconstructParMesh -mergeTol 1e-6 -constant //reconstructing your mesh.
checkMesh //check if the mesh failed as mesh failure will kill your job.
ls -d processor* | xargs -i rm -rf ./{} $1//delete the processor files.
rm -r 0 //remove the current 0 directory which is always empty.
cp -r 0.org 0 //copy the 0.org and past it with its new name 0.
decomposePar //Distribute your mesh and the time fields among n processors.
mpirun -np 12 simpleFoam -parallel //Run your simulation on 12 cores using
    the simpleFOAM solver.
reconstructPar//reconstruct your case.
yPlusRAS // Calculate the y+ incase of using any RANS approach.
foamToVTK //This format is easily imported to Paraview.
```

Figure 27: A sample of a file script used for submitting jobs on Apocrita.

.0.2 Force Coefficients Graphs:

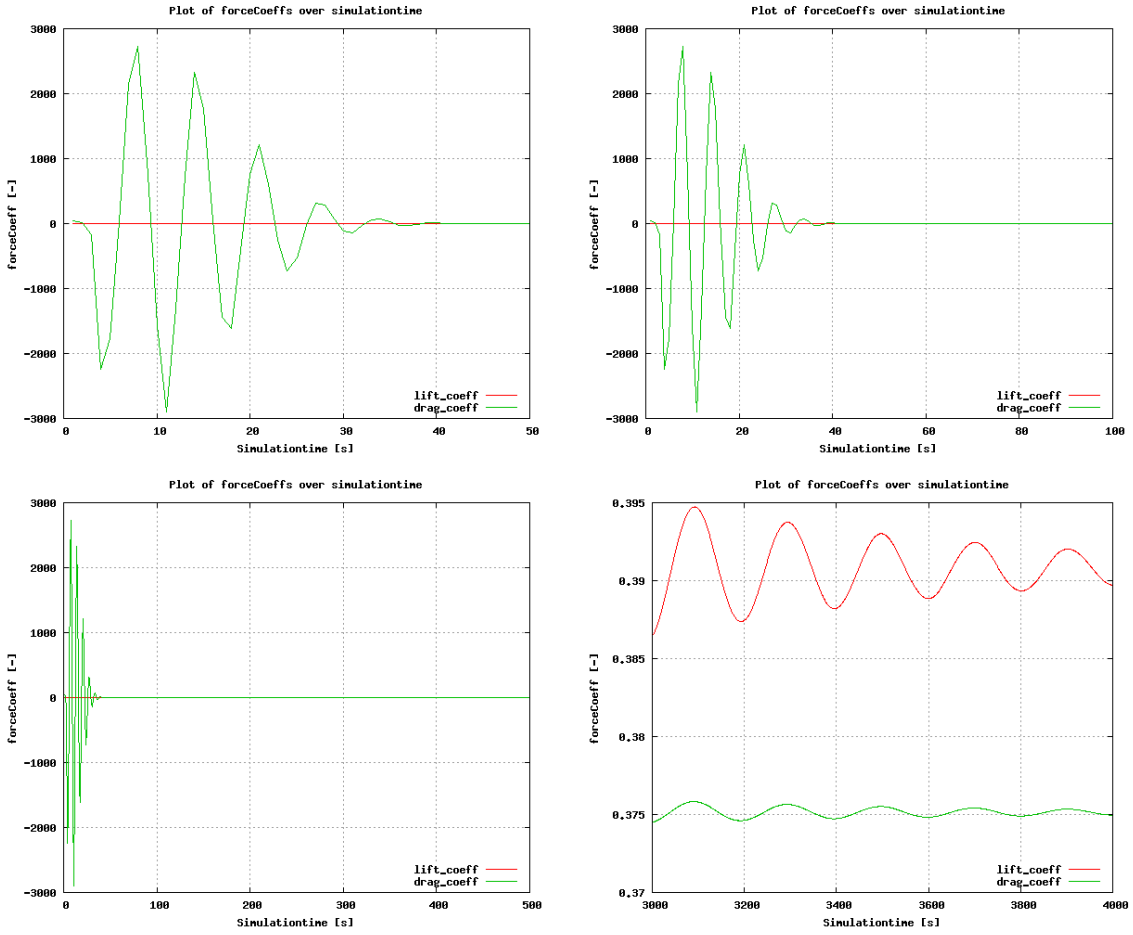


Figure 28: Force Coefficients graphs for the Spalart-Allmaras simulation.

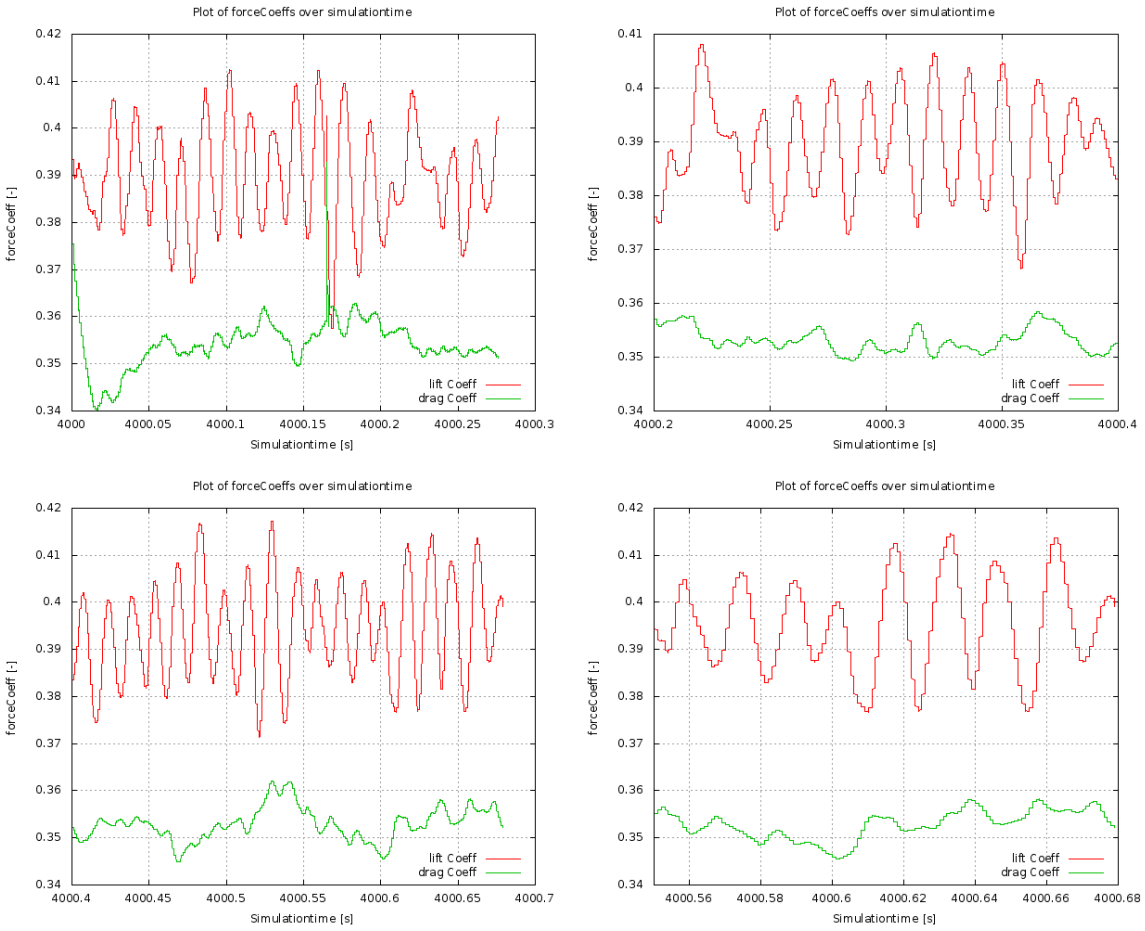


Figure 29: Force Coefficients graphs for the Delayed-Detached Eddy simulation.

force Coefficients graphs of mesh 2

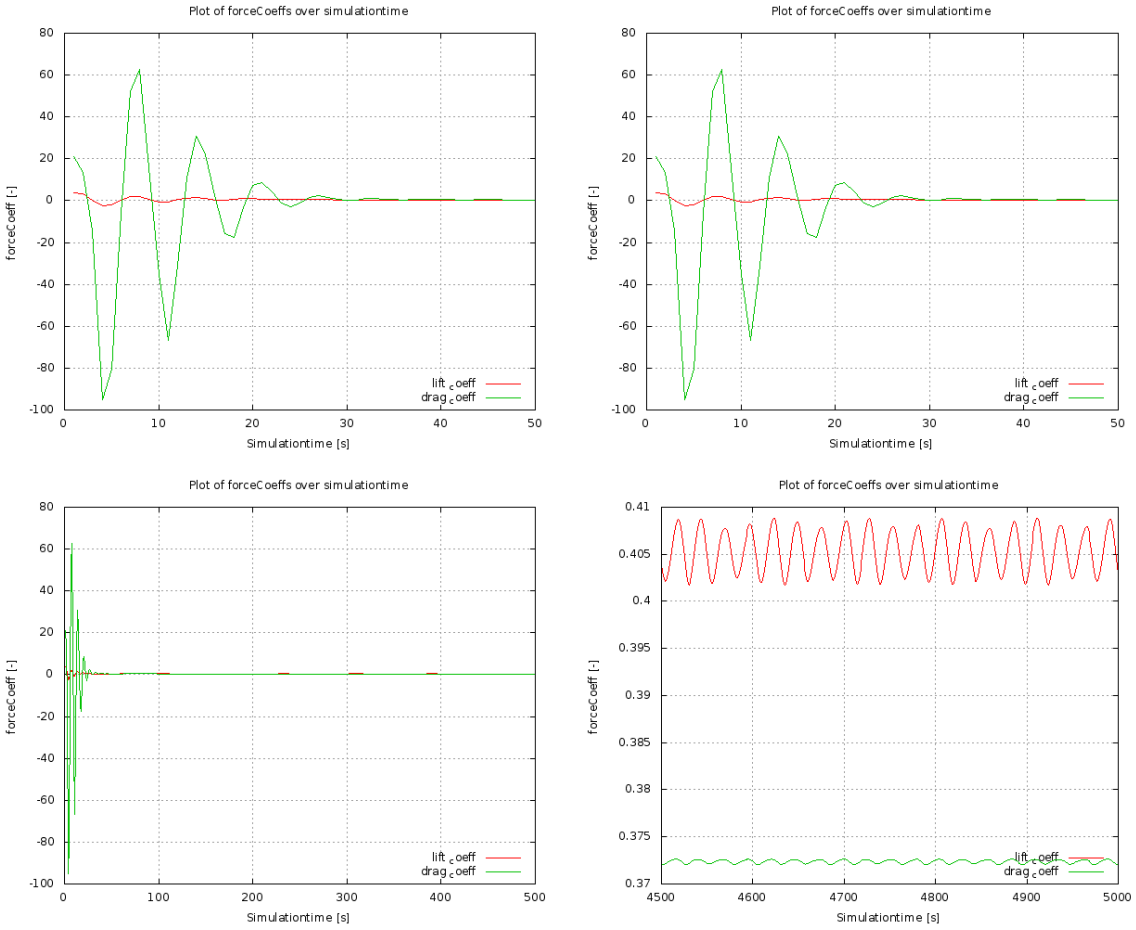


Figure 30: Force Coefficients graphs for the Spalart-Allmaras simulation of mesh 2.

.0.3 Pressure Probes Graphs

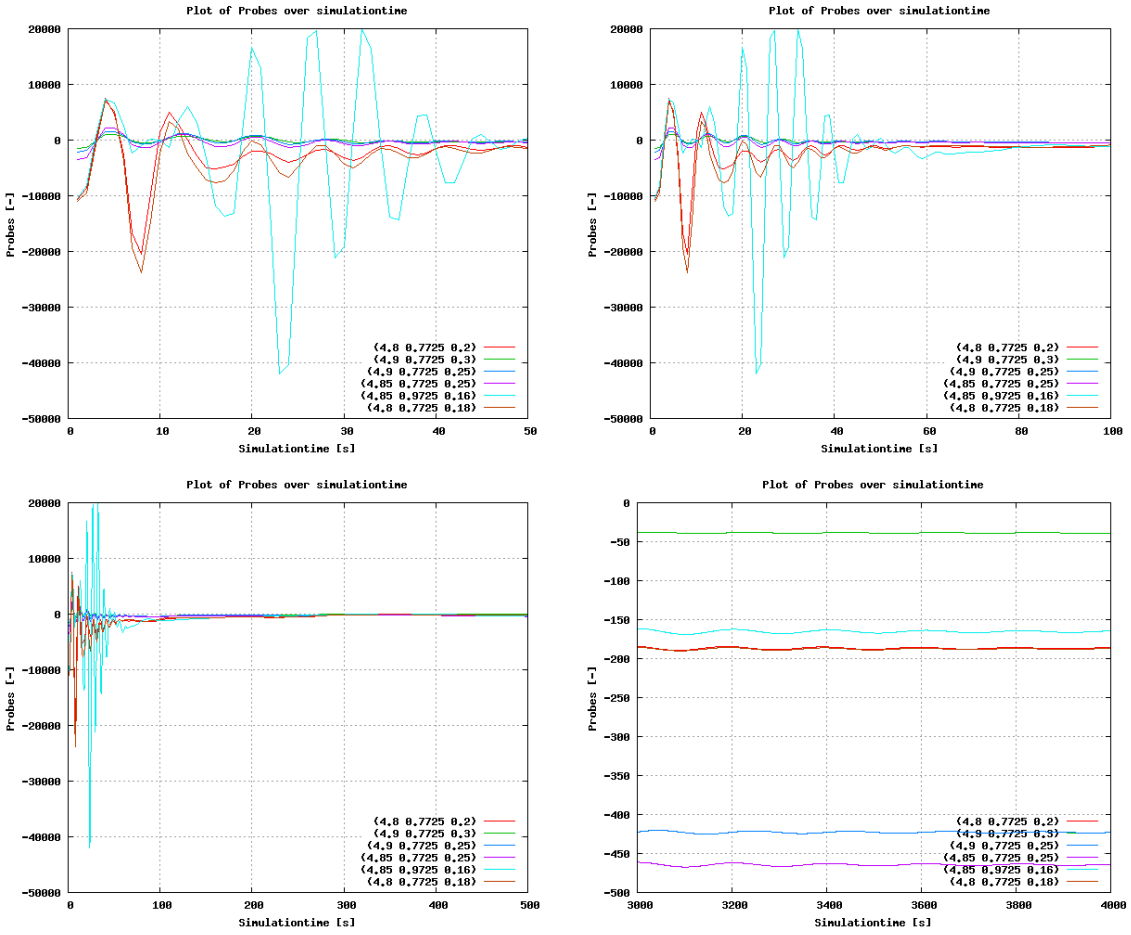


Figure 31: Pressure probes graphs for the Spalart-Allmaras simulation.

.0.4 Residuals Graphs

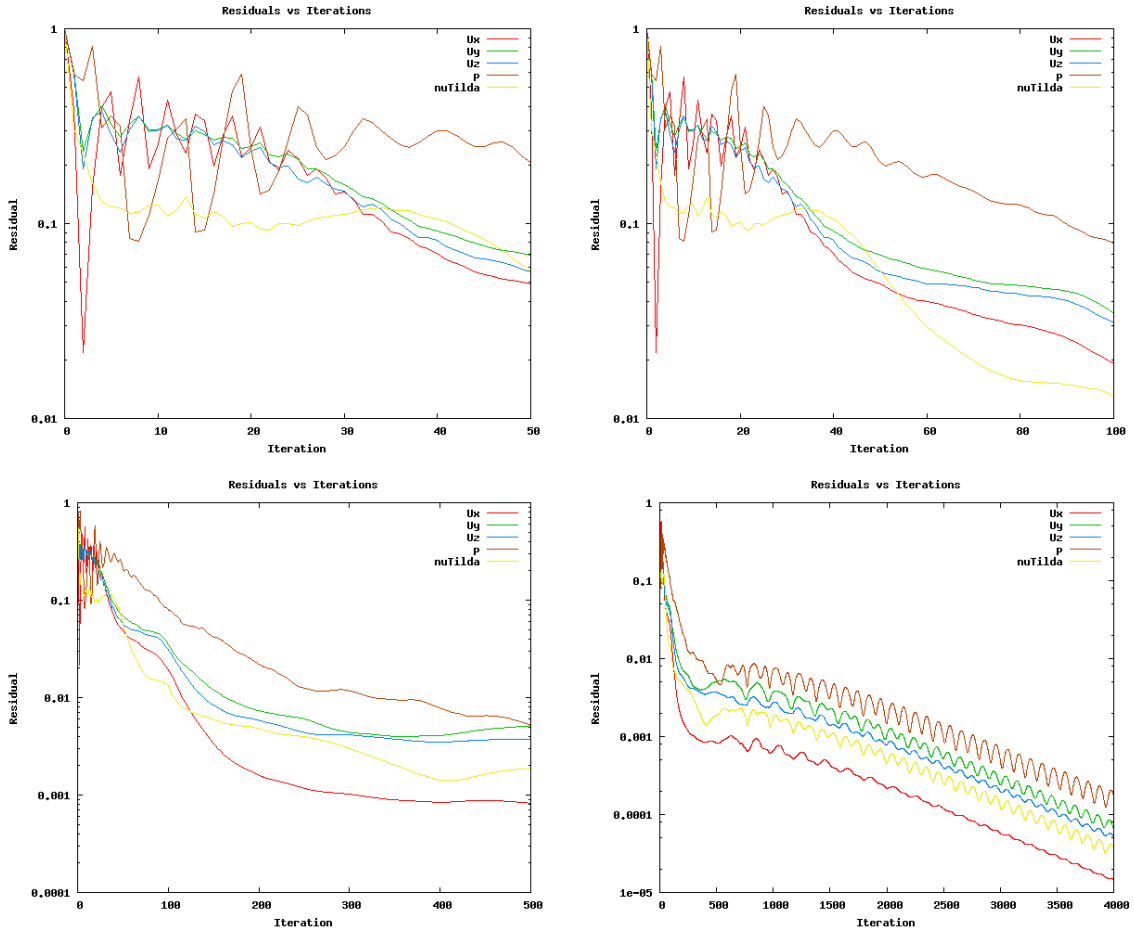


Figure 32: Residual graphs for the Spalart-Allmaras simulation.

Attempt 2

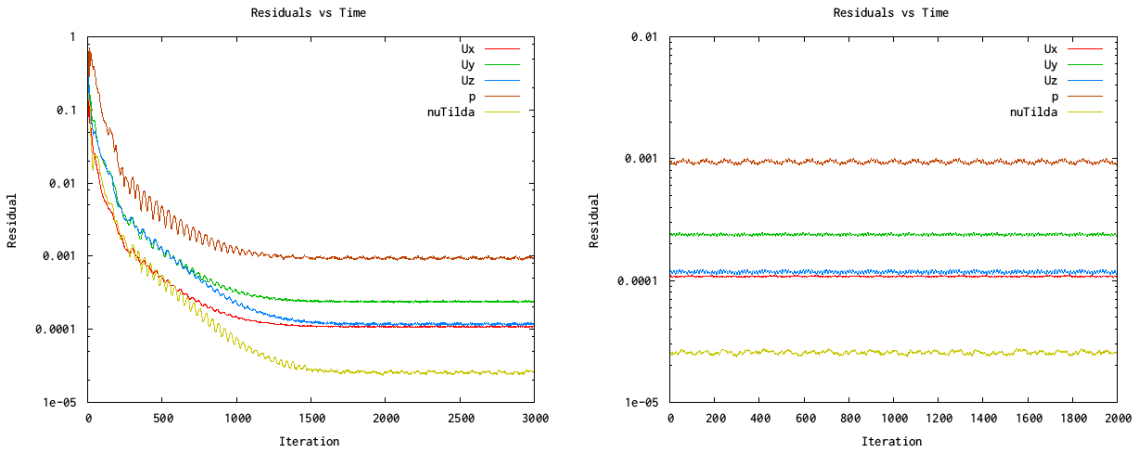


Figure 33: Residual graphs for the Spalart-Allmaras simulation of mesh 2. first run 3000 iterations (left), further 2000 iterations (right). Total iterations=5000