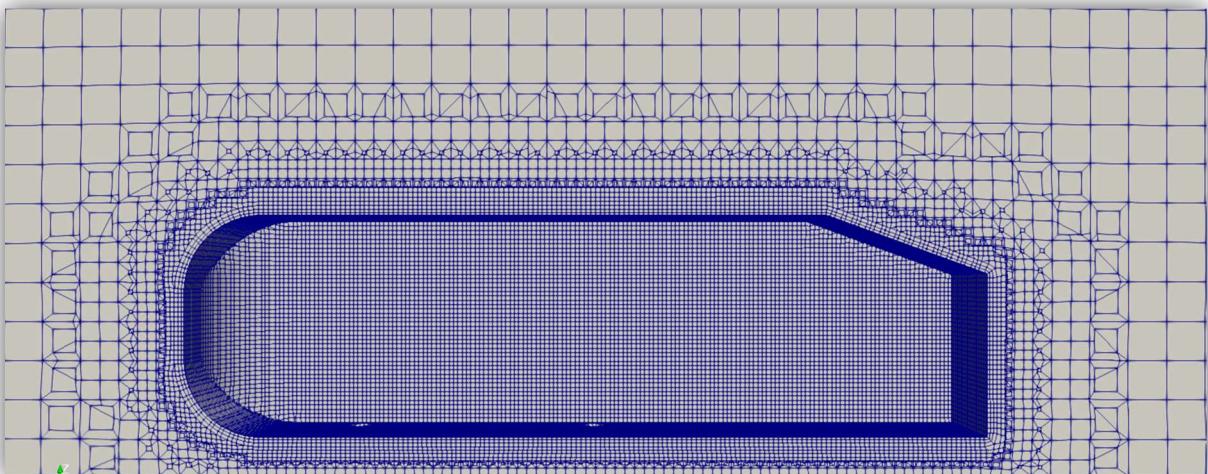




**Northumbria  
University**  
NEWCASTLE

## **KB6003 Vehicle Aerodynamics**

### **Tutorial 2**



## **Ahmed Body Meshing**

**Date: 8<sup>th</sup> October 2018**

## Contents

TABLE OF FIGURES .....	3
TABLE OF TABLES .....	3
<b>1 INTRODUCTION .....</b>	<b>4</b>
1.1.1 Creating Reference Extrudes for Outer Walls and Coarse Mesh.....	4
1.1.2 Creating Reference Extrudes for Fine Mesh .....	5
1.1 CREATING STL FILES FOR MESHING .....	6
1.2 REFINEMENT USING PARAVIEW (PARAFOAM) .....	7
1.2.1 Making the VTK File Compatible with “surfaceFeatureConvert” .....	11
<b>2 OPENFOAM INITIAL CASE SETUP .....</b>	<b>12</b>
2.1 MODIFYING DICTIONARIES.....	13
2.1.1 surfaceFeaturesDict / surfaceFeatureExtract (surfaceFeaturesExtractDict) .....	13
2.1.2 blockMesh (blockMeshDict) .....	14
2.1.3 snappyHexMeshDict .....	17
2.1.4 meshQuality (meshQualityDict).....	21
2.1.5 decomposePar (decomposeParDict) .....	22
<b>3 MESH GENERATION .....</b>	<b>22</b>
3.1 GENERIC MESH GENERATION CLI.....	22
3.2 CHECKING THE GENERATED MESH.....	23
3.2.1 Using “checkMesh” Command .....	23
3.2.2 Using ParaView .....	24
3.3 RECOMPILING THE MESH.....	26
3.4 EXAMPLE 1: SINGLE AHMED BODY WITH ONE REFINEMENT ZONE.....	26
3.5 EXAMPLE 2: SPLIT AHMED BODY WITH 2 STEP MESHING.....	29
3.5.1 Splitting the Ahmed Body .....	29
3.5.2 Generating the Mesh in 2 Steps .....	30
3.6 OTHER IMPROVEMENTS .....	35
<b>APPENDIX A: CO-ORDINATES FROM ONSHAPE .....</b>	<b>36</b>
<b>APPENDIX B: STL EXPORTS.....</b>	<b>36</b>
<b>APPENDIX C: COPYING TUTORIAL CASES AND PREPARING .....</b>	<b>37</b>
<b>APPENDIX D: VERTICES OF A THE GEOMETRY .....</b>	<b>38</b>
<b>APPENDIX E: SINGLE RUN VS 2 RUNS (SNAPPYHEXMESH) .....</b>	<b>39</b>

## Table of Figures

Figure 1.1: Using variables to dimension the rectangle.....	4
Figure 1.2: Editing appearances of parts.....	5
Figure 1.3: Enclosed Ahmed body.....	5
Figure 1.4: Dimensions for the fine mesh reference.....	6
Figure 1.5: Ahmed body with reference extrudes.....	6
Figure 1.6: ParaView .....	7
Figure 1.7: Import the ahmed_body.stl file to ParaView.....	8
Figure 1.8: Camera controls toolbar. ....	8
Figure 1.9: Extract feature edges from the Ahmed body.....	9
Figure 1.10: Change "Feature Angle" to show edge.....	9
Figure 1.11: Selecting the edge for extraction. ....	10
Figure 1.12: Extracting the edge using the "Extract Selection" tool. ....	10
Figure 1.13: Save edge as a .vtk file. ....	11
Figure 1.14: Removing additional information from the VTK file. ....	12
Figure 2.1: OpenFOAM dictionary header.....	13
Figure 2.2: Example “surfaceFeatureExtractDict”.....	14
Figure 2.3: Castellated mesh.....	18
Figure 2.4: Snapping phase.....	18
Figure 2.5: Adding layers to mesh.....	19
Figure 3.1: Output from "checkMesh".....	24
Figure 3.2: ParaView with mesh.....	25
Figure 3.3: "Clip" in "Common" toolbar. ....	25
Figure 3.4: Representations toolbar. ....	25
Figure 3.5: "VCR Controls" and "Current Time Controls". ....	26
Figure 3.6: Creating a new plane by offsetting "Top" plane. ....	30
Figure 3.7: Two step meshing process.....	31

## Table of Tables

Table 1.1: Variables for reference mesh wall dimensions.....	4
Table 1.2: Parts to export.....	6
Table 2.1: OpenFOAM case folder structure.....	12
Table 2.2: Patch types used in this tutorial (CFD Direct, 2018). ....	17
Table 2.3: Changes to "snappyHexMeshDict" dictionary. ....	19
Table 2.4: Changes to "meshQualityDict" dictionary.....	21
Table 2.5: Changes to "decomposeParDict" dictionary.....	22
Table 3.1: Visual inspection of mesh.....	27
Table 3.2: Export parameters for Ahmed body splits.....	30

# 1 Introduction

This tutorial will look into generating a refined mesh in OpenFOAM using the “snappyHexMesh” tool. This is an introductory guide and more information can be found at <https://cf.direct/openfoam/user-guide/v6-snappyhexmesh/>.

It is advisable to make “reference extrudes” (models/parts) for walls of different mesh areas (such as coarse and fine meshes). These allow easy differentiation of mesh zones using OpenFOAM and obtain co-ordinates (Appendix A) from geometry as well.

## 1.1.1 Creating Reference Extrudes for Outer Walls and Coarse Mesh.

- 1) On the model, create 5 new variables with the parameters shown in table 1.1.

Table 1.1: Variables for reference mesh wall dimensions.

Name	Value	Name	Value	Name	Value
box_x	1000	box_z	2000	back	5
box_y	4000	front	3		

**Note:** Make sure these variables are above the features that they are to be used on in the “Features” tree.

- 2) Create a sketch on the “Top” plane.
- 3) Sketch a rectangle to enclose the Ahmed body using the “Corner Rectangle” tool.
- 4) You can dimension the square using the variables created in step 1 of this section. The “box\_x” variable can be multiplied by the “front” and “back” variable to get dimensions of front and back of the enclosure box from the “Origin”. The figure 1.1 below shows the completed sketch.

**Note:** All dimensions are from the “Origin”.

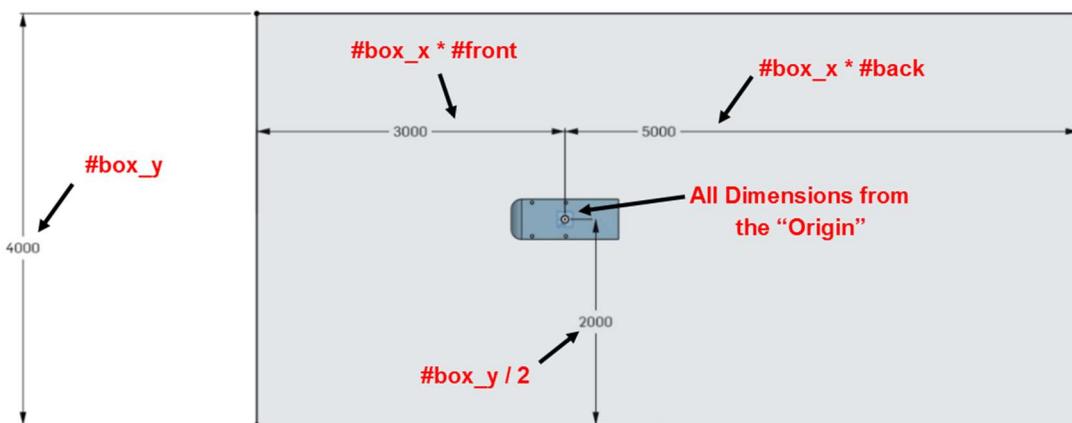


Figure 1.1: Using variables to dimension the rectangle.

- 5) Extrude the sketch with “Blind” selected from the drop down menu. Input “#box\_z” (without quotations) for the depth.

**Note:** Make sure the “New” tab is selected instead of the “Add” as the extrude should be a new part.

- 6) The new part (“Part 2”) will be opaque but can be made transparent by right clicking on the “Parts” list below “Features” list and clicking “Edit appearance”. Move the transparency slider to your liking (as shown in figure 1.2) and click ok.

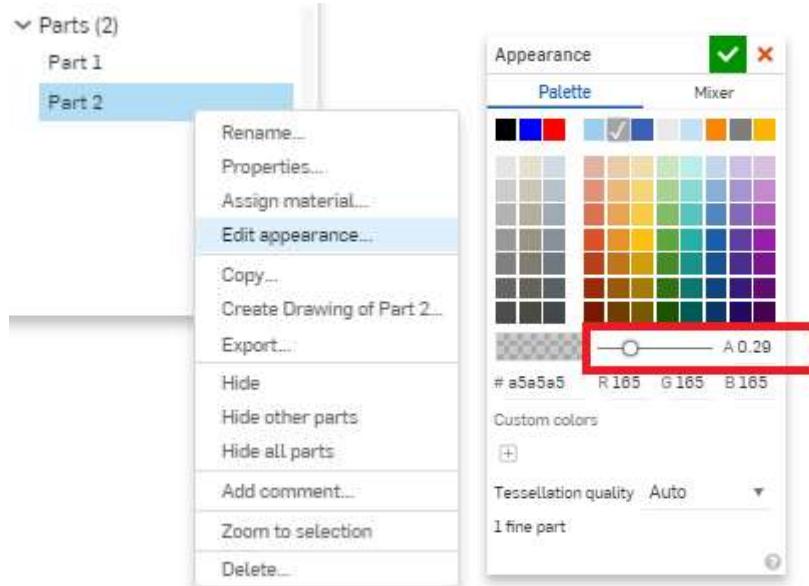


Figure 1.2: Editing appearances of parts.

- 7) The model should now look similar to figure 1.3 shown below.

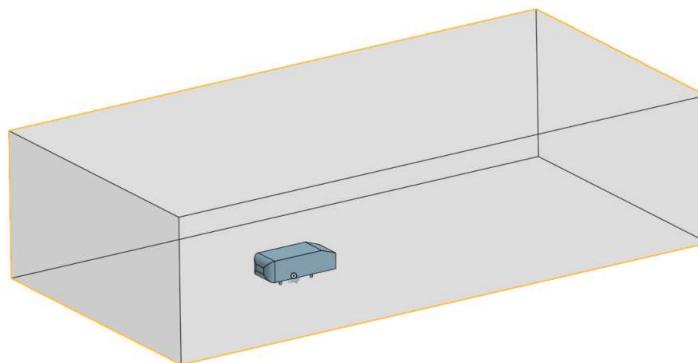


Figure 1.3: Enclosed Ahmed body.

### 1.1.2 Creating Reference Extrudes for Fine Mesh

Follow from step 2 in section 2.3.1 but with the following fixed dimensions shown in figure 1.4 extruded to a height (z-axis) of 1000 mm. Alternatively, you can create/use variables and assign them to the sketch as well. The finished model should look similar to figure 1.5 shown below.

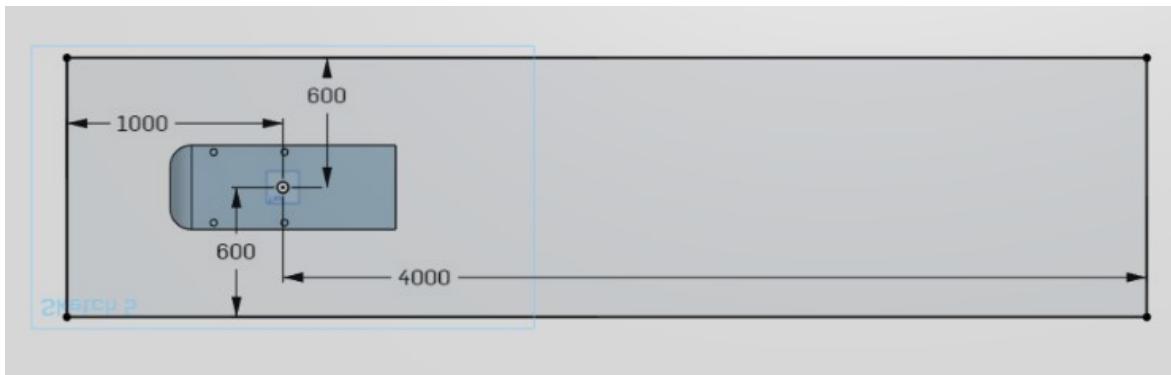


Figure 1.4: Dimensions for the fine mesh reference.

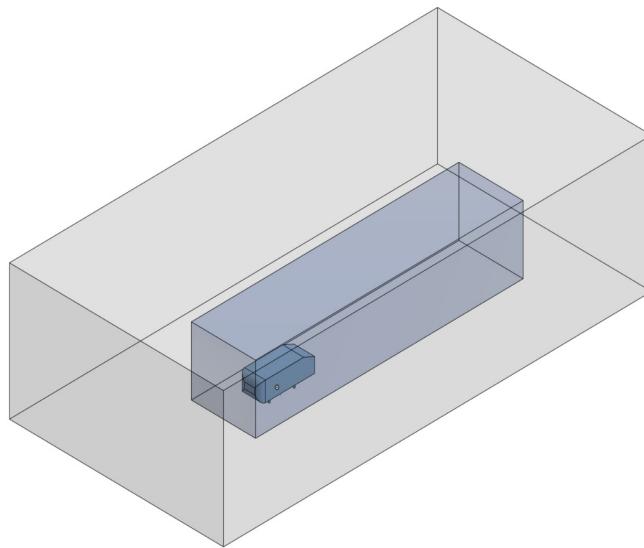


Figure 1.5: Ahmed body with reference extrudes.

## 1.1 Creating STL files for Meshing

Please export the STL files according to the table 1.2 below. Appendix B shows figures along with filenames for further clarification.

Table 1.2: Parts to export

File Name	Part
<b>ahmed_body.stl</b>	Combined Ahmed body (not needed if exported previously)
<b>ref_zone1.stl</b>	The reference extrude made in section 1.1.2.
<b>Format:</b> STL	
<b>STL Format:</b>	Binary
<b>Units:</b>	Metre
<b>Resolution:</b>	Fine
<b>Options:</b>	Download

## 1.2 Refinement Using ParaView (paraFoam)

This is an alternative method to selecting/generating refinement zones/features. It can be useful when you would want to manually insert uncaptured features. In this particular case, the starting edge of the slanted back is not picked up by the “`includeAngle`” set in section 3.2.1. This can be manually added as shown below and the mesh could later be refined in this region.

- 1) Open ParaView by typing “`paraFoam`” in a terminal window. Since you are not opening it from a specific OpenFOAM directory, you will be presented with several warnings and a prompt “Would you like to open ParaView anyway <Y | n>:”

Please type “y” (without quotations) and “Enter” (return) to launch ParaView. You will be presented with a screen as shown in figure 1.6 below.

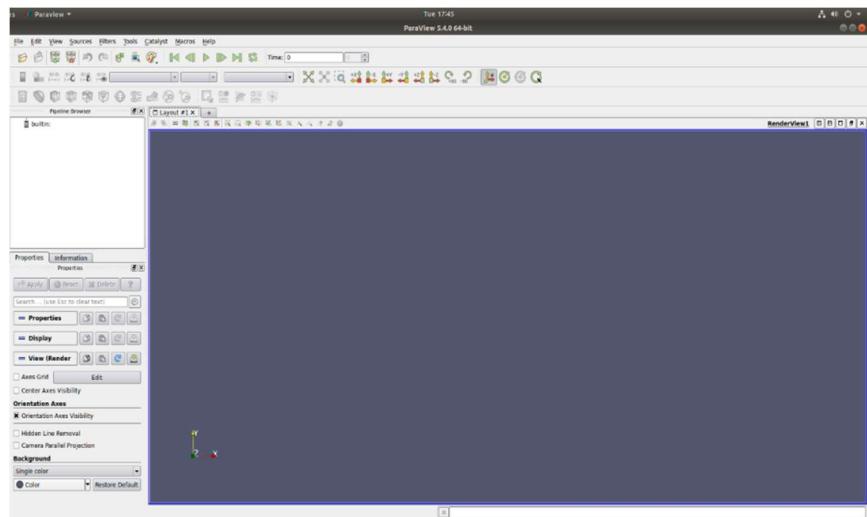


Figure 1.6: ParaView

- 2) Click “Open” (📁) or go to “File>Open ...” and open the `ahmed_body.stl` file. Please select “Apply” from the “Properties” tab to display the STL file.



Figure 1.7: Import the ahmed\_body.stl file to ParaView.

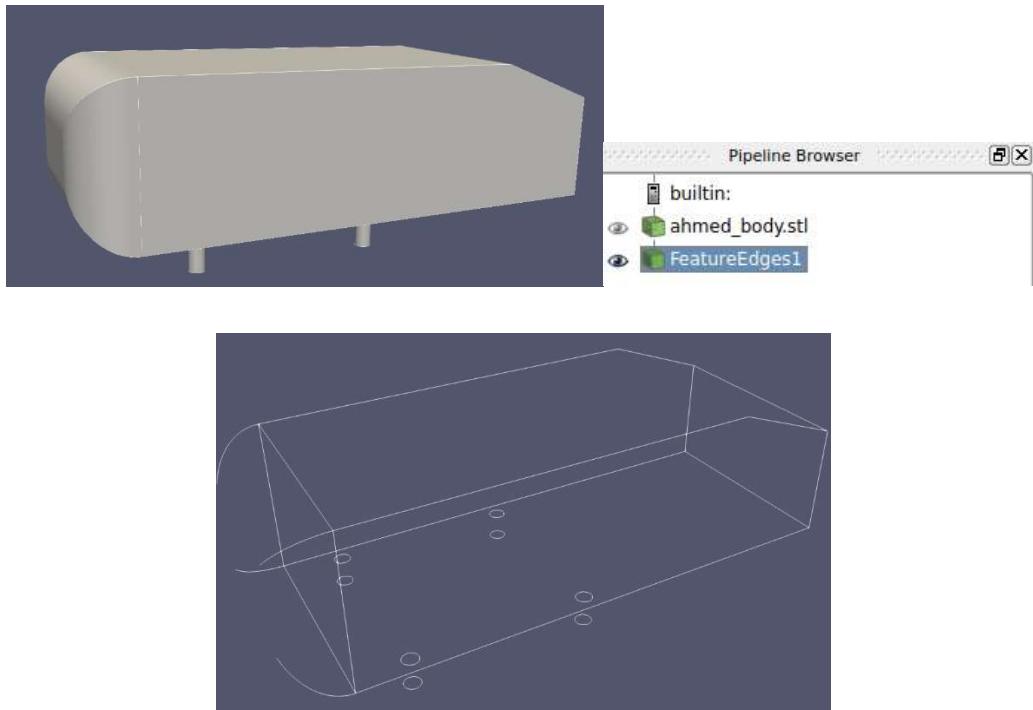
- 3) The view can be manipulated with using the mouse and the view controls in the “Camera Controls” toolbar shown in figure 1.8.



Figure 1.8: Camera controls toolbar.

- 4) To extract edges, click on the “ahmed\_body.stl” in the “Pipeline Browser”. Then on the menu bar, go to “Filters>Alphabetical>Feature Edges”. Click “Apply” on the properties window.

This should show some prominent edges of the Ahmed body outlined. The eye next to “ahmed\_body.stl” can be used to hide the solid to view the lines only as shown in figure 1.9 below.

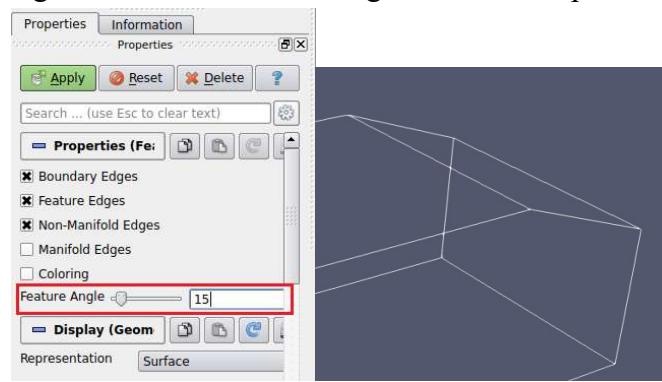


*Figure 1.9: Extract feature edges from the Ahmed body.*

- 5) As mentioned in the beginning of the section, the edge at the start of the slanted back is missing. This is due to the “Feature Angle” being set to 30 deg. by default. The feature edges are defined as “edges that are used by two polygons whose dihedral angle is greater than the feature angle.” ([Since the angle between the two planes around this edge is 20 deg., the “Feature Angle” needs to be less than 20 deg.](https://www.paraview.org/Wiki/ParaView/Users_Guide>List_of_filters</a>)</li>
</ol>
</div>
<div data-bbox=)

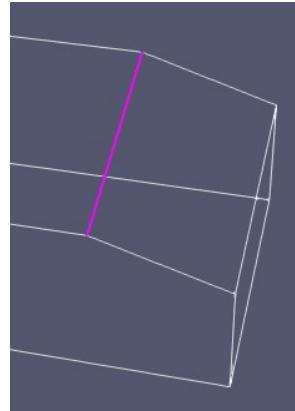
- 6) Replace the default “Feature Angle” by a value lower than 20 (19, 18, 17, ....,2) and click apply to show the missing edge as shown in figure 1.10.

**Note:** Lower the angle value, more feature edges would be captured.



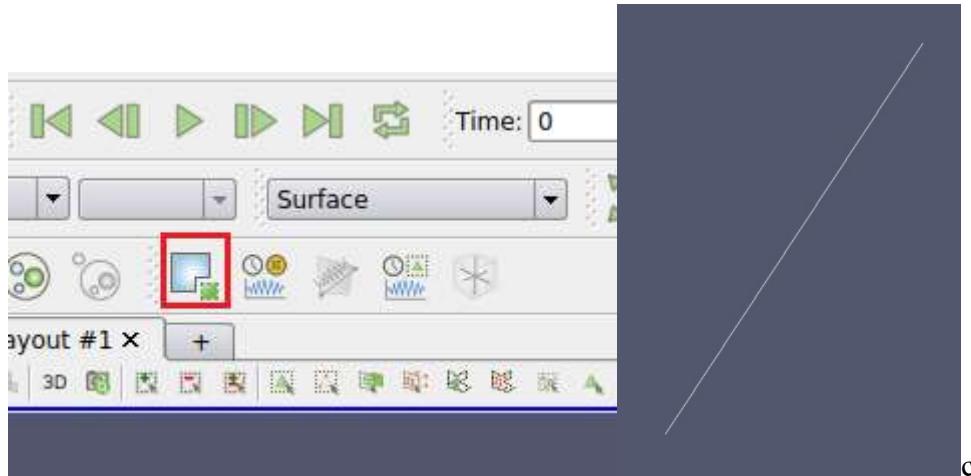
*Figure 1.10: Change “Feature Angle” to show edge.*

- 7) Select “Select Cells on (s)” tool ( ) under the “Layout” tab and click on the edge to be extracted.



*Figure 1.11: Selecting the edge for extraction.*

- 8) Now click the “Extract Selection” under “Data Analysis” toolbar as shown in [figure 2.15](#) and click “Apply” on the “Properties” tab.



*Figure 1.12: Extracting the edge using the "Extract Selection" tool.*

- 9) Make sure only the extract selection (“ExtractSelection1” in this case) is selected and visible on the “Pipeline Browser”. Click on “Save” ( ) or from menu, select “File>Save Data ...”.
- 10) Select the “Legacy VTK Files(\*.vtk)” from the “Files of type” dropdown and name it “edge1”. Save it with rest of the STL files of this tutorial.

**Note:** Please make sure “**Ascii**” is selected as the “File Type” in the following window.

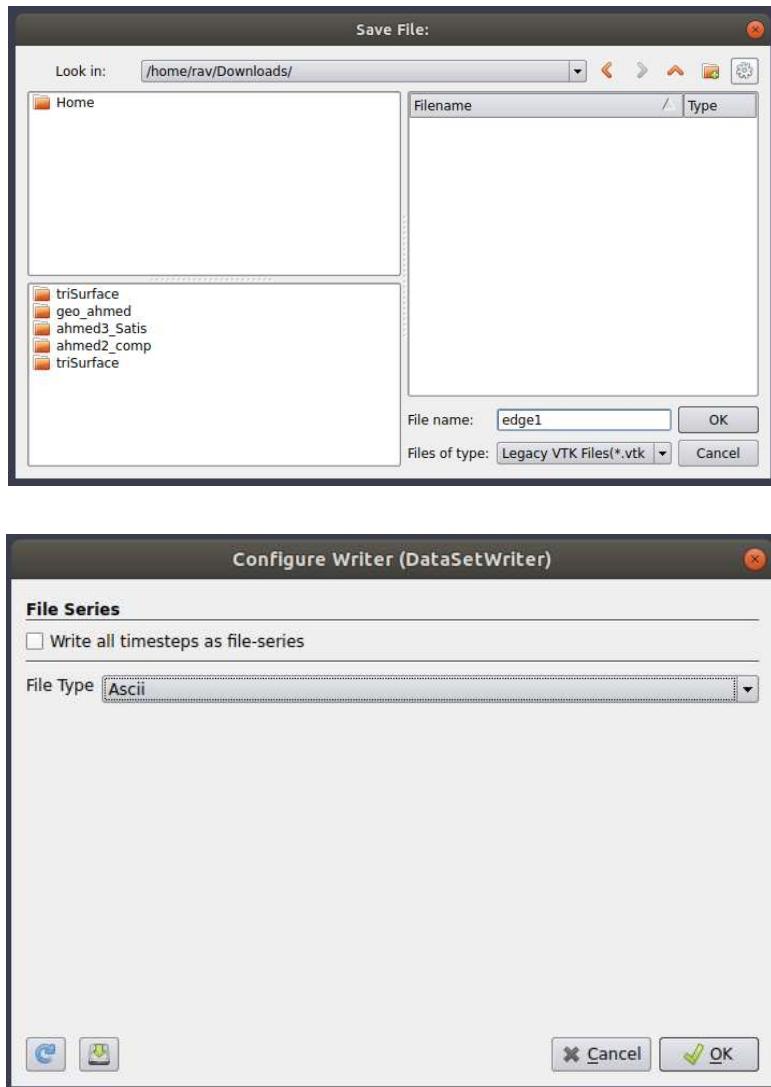


Figure 1.13: Save edge as a .vtk file.

### 1.2.1 Making the VTK File Compatible with “surfaceFeatureConvert”

The “surfaceFeatureConvert” command will be used to convert the VTK file to OpenFOAM’s eMesh format. The VTK file generated in section 1.2 may contain additional data (such as Metadata) which will output errors when converting. The following modifications to the VTK file would enable smooth conversion;

- 1) Open the VTK file generated in section 1.2 in the text editor.
- 2) Remove the line containing “METADATA” to “DATA” (highlighted in figure 1.14 below) and save it.

**Note:** Do NOT change any formats or other fields such as co-ordinates, etc.

```

# vtk DataFile Version 4.2
vtk output
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 2 float
0.313388 -0.1945 0.338 0.313388 0.1945 0.338
METADATA
INFORMATION 1
NAME L2_NORM RANGE LOCATION vtkDataArray
DATA 2 0.500286 0.500286

CELLS 1 3
2 0 1

CELL_TYPES 1
3

CELL_DATA 1
FIELD FieldData 1
vtkOriginalCellIds 1 1 vtkIdType
1349
POINT_DATA 2
FIELD FieldData 1
vtkOriginalPointIds 1 2 vtkIdType
1348 1349

```

Figure 1.14: Removing additional information from the VTK file.

## 2 OpenFOAM Initial Case Setup

OpenFOAM cases follow a certain file structure and a set of instruction files (dictionaries) to run cases. A standard OpenFOAM case consist of the file structure shown in table 2.1 below.

Table 2.1: OpenFOAM case folder structure.

	<b>Folder</b>	<b>Description</b>
	Case	Main case directory
	0_orgs	Initial data of fields such as velocity, pressure and boundary conditions used to define the problem.
	constant	Contains the physical properties and geometries for the case. Also contains the mesh when generated.
	polyMesh	Contains the mesh after it is generated.
	triSurface	Contains the geometries (STLs, VTks, eMesh, etc.)
	system	Contains dictionaries used to control and run the case (such as blockMesh, controlDict, snappyHexMeshDict, surfaceFeatureExtractDict, meshQualityDict, etc.)

As the case progresses, additional files (such as time directories, decomposed directories for parallel processing, etc. will be added).

When preparing a case for OpenFOAM, it is advisable to copy a similar tutorial case and edit the dictionaries to suite your case. For example, the “motorBike” tutorial can be used for this case as it is an incompressible, steady state case using the “simpleFoam” solver. Instructions on copying a case can be seen in **Appendix C**. However, for this tutorial session, a case has been adapted and can be downloaded from GitHub.

<https://github.com/NU-Aero-Lab/OpenFOAM-Cases/tree/master/KB6003/Tutorial2>

Download the “Start.zip” and unzip it to your working directory.

## 2.1 Modifying Dictionaries

OpenFOAM dictionaries contain instructions to run cases. The following sections would show how to manipulate the extraction of features and generation of a mesh. This section would only concentrate on “surfaceFeatureExtractDict”, “blockMesh”, “snappyHexMeshDict” and “meshQuality” dictionaries as these govern the mesh generation.

These dictionaries follow a general C++ format with “//” marking comment lines and “/\*” and “\*/” marking start and end of multiple line comments. The files contain headers, information sections and indentations, which are not mandatory but good programming practices. Furthermore, it would make it easier to debug and for others to understand your code. **Figure 2.1** below shows a typical dictionary header. More about dictionaries can be found on <https://cfd.direct/openfoam/user-guide/v6-basic-file-format/>.

```
/*-----* C++ -----*/
=====
 / F ield     | OpenFOAM: The Open Source CFD Toolbox
  \ O peration | Website: https://openfoam.org
   \ A nd       | Version: dev
    \ M anipulation
\*-----*|-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       surfaceFeatureExtractDict;
}
// * * * * * //
```

Figure 2.1: OpenFOAM dictionary header.

### 2.1.1 surfaceFeaturesDict / surfaceFeatureExtract (surfaceFeatureExtractDict)

This dictionary provides parameters for the surface features and their extraction. It is located under “<case>/system” directory. The **figure 2.2** shows a typical “surfaceFeatureExtractDict” with comments (in blue followed by “//”) describing the code.

```

surfaceFeatureExtractDict
~/OpenFOAM/rav-dev/run/Start/ahmed_0/system
Save ☰ ⌂ ✎
/*-----* C++ -*-*/
=====
\\ / Field | OpenFOAM: The Open Source CFD Toolbox
\\ / Operation | Website: https://openfoam.org
\\ / And | Version: dev
\\ / Manipulation |
\*-----*/
FoamFile
{
    version    2.0;
    format     ascii;
    class      dictionary;
    object     surfaceFeatureExtractDict;
}
// ****
geometry.stl
{
    // How to obtain raw features (extractFromFile || extractFromSurface)
    extractionMethod    extractFromSurface;

    extractFromSurfaceCoeffs
    {
        // Identify a feature when angle between faces < includedAngle
        // - 0 : selects no edges
        // - 180: selects all edges
        includedAngle    150;
    }

    // Write options
    // Write features to obj format for postprocessing
    writeObj          yes;
}
// ****

```

Saving file "/home/rav/OpenFOAM/rav-dev/run/Start/ahmed\_0/syste... C Tab Width: 8 Ln 17, Col 9 INS

Figure 2.2: Example “surfaceFeatureExtractDict”.

- 1) Add the following lines to the body which instruct to extract using the “extractFromSurface” method with “includeAngle” and to write the extraction to “.emesh” and “.obj” files. These can be found in “triSurface” and “extendedFeatureEdgeMesh” folders in “Constant” directory. A copy of the changed dictionary can be found in GitHub under “end.zip/example1/ahmed1”.

```

ahmed_body.stl
{
    extractionMethod    extractFromSurface;

    extractFromSurfaceCoeffs
    {
        includedAngle    150;
    }

    writeObj          yes;
}

```

Repeat the above code for any other STL files in lines below.

- 2) Save file (“Ctrl+s” or click save) and exit.

## 2.1.2 blockMesh (blockMeshDict)

This dictionary governs the parameters for blockMesh tool supplied with OpenFOAM which generates (as the name suggests) a mesh with blocks. This section will look at the generation of a block mesh for the background mesh of the Ahmed body case. More information about blockMesh can be found in <https://cfd.direct/openfoam/user-guide/v6-blockmesh/>.

Please insert the following code to your “blockMeshDict” dictionary below the header. The lines marked with “//” are comments and do not serve any other functions. A copy of the changed dictionary can be found in GitHub under “end.zip/example1/ahmed1”.

```
// Factor for scaling
convertToMeters 1;

// Declaring variables using macro syntax notation which can later
// be used with $variablename
xmin -3;
xmax 5;
// Try to create ymin, ymax, zmin, zmax
// Hint: vertices from geometry generated in section 1.1.1

// Cell spacing
deltax 0.1;
deltay 0.1;
deltaz 0.1;

// Calculate the length by using the above variables and assign to
// new variables
lx #calc "($xmax) - ($xmin)";
// Try to calculate ly and lz

// Calculate the number of cells with above criteria and round
xcells #calc "round($lx/$deltax)";
// Try to calculate ycells and zcells

// The 8 vertex co-ordinates of the 3D block (geometry) starting
// from 0 vertex number
// Check Appendix D in Tutorial 2 for a visual representation of
// vertices
vertices
(
    ($xmin $ymin $zmin) // 0 vertex
    ($xmax $ymin $zmin) // 1st vertex
    ($xmax $ymax $zmin) // 2nd vertex
    ($xmin $ymax $zmin) // 3rd vertex
    ($xmin $ymin $zmax) // 4th vertex
    ($xmax $ymin $zmax) // 5th vertex
    ($xmax $ymax $zmax) // 6th vertex
    ($xmin $ymax $zmax) // 7th vertex (8th counting 0)
);
// Defines the block topology
blocks
(
    // Hexahedral (hex) block with vertices in sequential order, number
    // of cells in each direction
```

```

// ($xcells, etc.) and stretching (simpleGrading) of the mesh (in
// this case uniform)
hex (0 1 2 3 4 5 6 7) ($xcells $ycells $zcells) simpleGrading (1 1 1)
);

// To be used to define edges joining vertices that are not
// straight. It is assumed straight by default (leave blank)
edges
(
);

// Define the patches for boundary conditions
boundary
(
    side2 // patch name (user-defined)
    {
        type symmetry; // patch type (refer to Tutorial 2
                        //blockMesh sections for details
        faces
        (
            (3 7 6 2) // list of vertices of the surface patch (face)
        );
    }
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    name1 // check and re-name the patch appropriately
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    side1
    {
        type symmetry;
        faces
        (
            (0 1 5 4)
        );
    }
    top
    {
        type symmetry;
        faces
        (
            (4 5 6 7)
        );
    }
    ground

```

```

{
    type patch;
    faces
    (
        (0 3 2 1)
    );
}

// Incase of multiple blocks() (defined earlier), patches to merge
mergePatchPairs
(
);

```

The patches in the “blockMeshDict” files refers areas of the boundary surface. Some available patches and their descriptions can be found in <https://cfddirect/openfoam/user-guide/v6-boundaries/#x24-1740005.2.1>. The “blockMesh” and patch names can be checked in ParaView and renamed if needed. This can be done by launching using “paraFoam” and ticking and unticking “Mesh Regions” as seen in figure 3.2. The table 2.2 below gives the patch descriptions used in this tutorial.

*Table 2.2: Patch types used in this tutorial (CFD Direct, 2018).*

Patch	Description
<b>patch</b>	Generic type containing no geometric or topological information about the mesh, e.g. used for an inlet or an outlet.
<b>wall</b>	For patch that coincides with a solid wall, required for some physical modelling.
<b>symmetry</b>	For any (non-planar) patch that uses the symmetry plane (slip) condition.

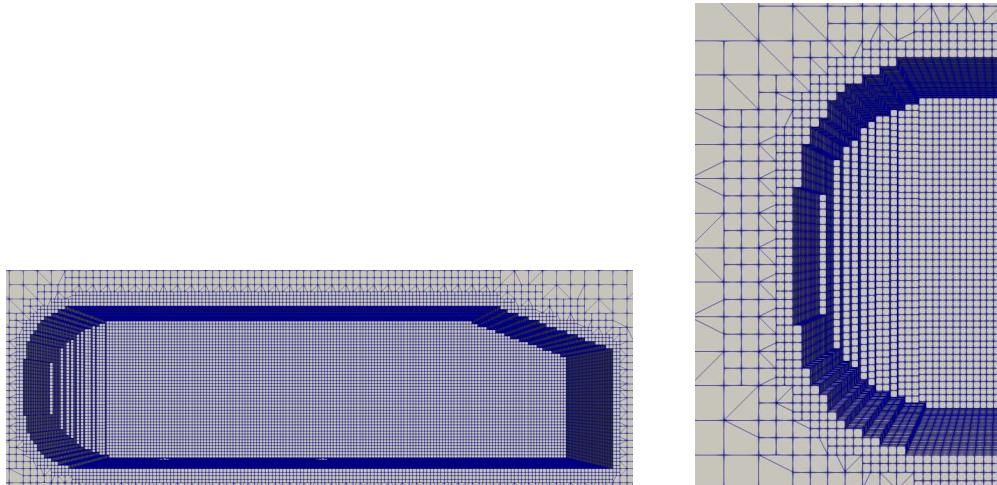
### 2.1.3 snappyHexMeshDict

This dictionary controls the generation of 3D meshes consisting of hexahedral and split-hexahedral cells from tri-surfaces, STLs or Wavefront Object (.obj) format. Due to the length of this dictionary, it has not been included in the tutorial handout. A copy of the changed dictionary can be found in GitHub under “end.zip/example1/ahmed1”. Comments (following “//”) have been added to explain each parameter.

#### 2.1.3.1 The Four Parts of snappyHexMeshDict

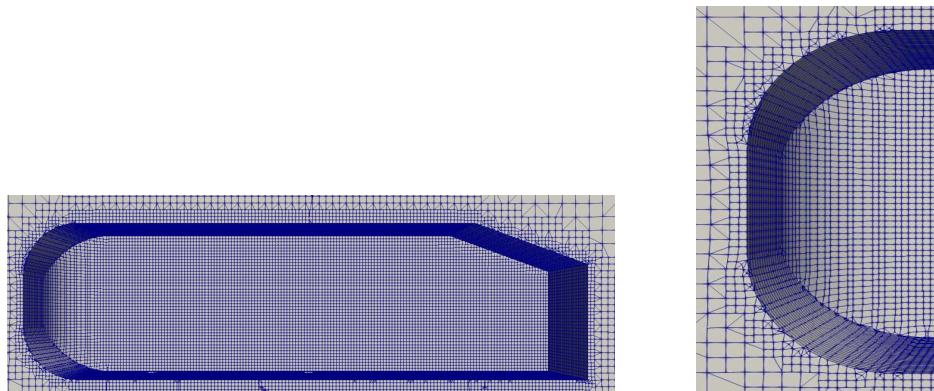
A typical snappyHexMeshDict file consist of 4 main parts, namely “castellatedMesh”, “snapControls” (“snap”), “addLayers” and “Advanced” parameters (such as meshQualityControls and writeFlags).

The “castellatedMesh” refines the background mesh close to the features and surfaces which enables easy snapping (figure 2.3). The sizes defined in here are relative to the background mesh (in this case, blockMesh created in section 2.1.2).



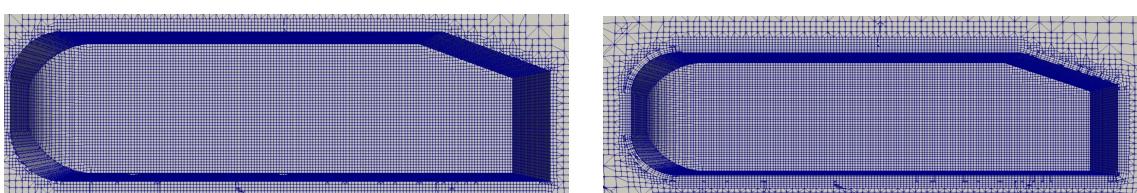
*Figure 2.3: Castellated mesh*

The “snap” (snapControls/snapping) attempts fit the mesh onto the input geometry (figure 2.4). This is an iterative process and if attempts fail to meet defined mesh quality, it would be re-attempted using reformed parameters.



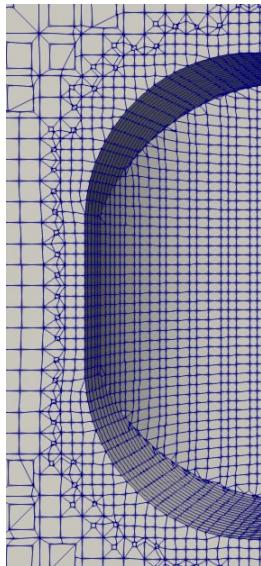
*Figure 2.4: Snapping phase.*

The “addLayers” (addLayersControls), as the name suggests, adds layers to the defined surfaces (figure 2.5). These fill any voids and irregularities created by shrinking the mesh near surfaces.

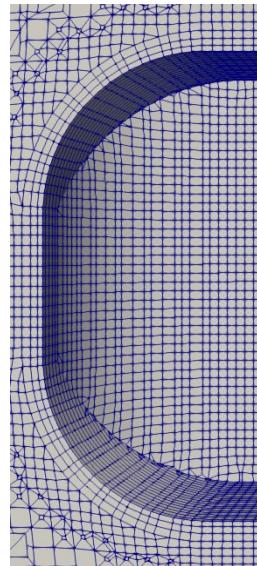


*Figure 2.5a: Side view before layers*

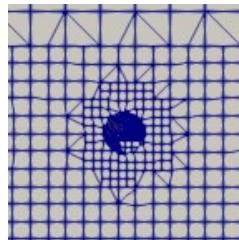
*Figure 2.5b: Side view with layers*



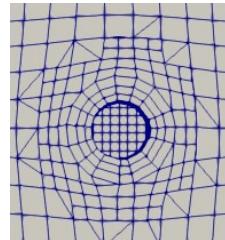
*Figure 2.5c: Side view before layers (zoomed in)*



*Figure 2.5d: Side view with layers (zoomed in)*



*Figure 2.5e: Support (bottom view) before layers*



*Figure 2.5f: Support (bottom view) with layers*

*Figure 2.5: Adding layers to mesh*

The first few lines of the body of “snappyHexMeshDict” indicates which phase of meshing to run. These can be enabled or disabled by setting them to “true” or “false” respectively. These phases can be run together, individually or re-run (together or individually) after corrections. The output can be viewed in stages in ParaView as shown in [figure 2.3-2.5](#) by following instructions in section 3.3.

The Advances parameters contain parameters such as “meshQualityControls” (see [section 2.1.4](#)), “writeFlags” (information for post processing), debug flags amongst others.

The [table 2.3](#) below will outline some important fields (only ones changed from the “motoBike” case) “snappyHexMeshDict” file.

*Table 2.3: Changes to “snappyHexMeshDict” dictionary.*

Parameter	Value	Description
<b>geometry</b>	Custom STLs	STL Files for meshing.

<b>Primitive geometry</b>	Can be created within by using co-ordinates	Simple geometry created within OpenFOAM to assign refinement zones.
<b>castellatedMesh</b>		
<b>nCellsBetweenLevels</b>	2 (default is 3)	Number of buffer layers of cells when transitioning through levels of refinement
<b>features</b>	eMesh files generated by “surfaceFeatureExtract”	Feature edge refinement
<b>level</b>	0, 4	Level of refinement (see Appendix E for levels of refinement)
<b>refinementSurfaces</b>		Surface refinement of custom STLs name in “geometry” section.
<b>level</b>	(2 2), (4 4)	Level of refinement, min and max for cells intersecting the surface.
<b>resolveFeatureAngles</b>	30	Angle above which max refinement is added.
<b>planarAngle</b>	30	Angle used to detect opposite surfaces.
<b>Region-wise Refinement</b>	Regions from Custom STLs (names assigned in “geometry” used)	Regions for refinement with level (x y) where x is distance, y is level of refinement from surface to distance x.
<b>LocationinMesh</b>	(2 0 1)	Location (co-ordinates) inside the enclosure to be meshed. This must not lie/intersect the body.
<b>snapControls</b>		
<b>nSolverIter</b>	300	Number of mesh displacement relaxation iterations. Higher values result in better fitted mesh but will take more time.
<b>nFeatureSnapIter</b>	20	Iterations for feature edge snapping. Increase for better feature edge quality.
<b>addLayersControls</b>		
<b>nSurfaceLayers</b>	3	Number of layers
<b>expansionRatio</b>	1.2	Factor to increasing from layer to layer.
<b>finalLayerThickness</b>	0.8	Thickness of the layer furthest from the wall.

<b>minThickness</b>	0.001	Minimum thickness of layers (relative or absolute)
<b>featureAngle</b>	330	Angles above which mesh is collapsed.
<b>nRelaxIter (slipfeature)</b>	10	Max. number of snapping relaxation iterations. Improves the quality of the body fitted mesh.
<b>nSmoothNormals</b>	1	Number of smoothing iterations of interior mesh movement direction.
<b>//Advanced</b>		
<b>meshQualityControls</b>	Quality parameters for the mesh	Can be in a different file with #include notation
<b>relaxed</b>	maxNonOrtho 75;	Max non-orthogonality allowed.
<b>nsmoothscale</b>	4	Number of error distribution iterations.
<b>errorReduction</b>	0.75	Scale back mesh displacement at error points.

The mesh quality file can be put in a separate meshQualityDict file and the “#include “meshQualityDict”” can be used.

#### 2.1.4 meshQuality (meshQualityDict)

This dictionary defines parameters to ensure the quality of the mesh generated is satisfactory. If the mesh does not comply with the requirements of this dictionary, it would be re-iterated with additional (defined) settings.

In this particular case, the following parameters in table 3.4 have been changed. The rest are included from the default “meshQualityDict” supplied with OpenFOAM by the line “#includeEtc “caseDicts/mesh/generation/meshQualityDict”. A copy of the changed dictionary can be found in GitHub under “end.zip/example1/ahmed1”.

Table 2.4: Changes to "meshQualityDict" dictionary.

Parameter	Value	Description
<b>maxNonOrtho</b>	75	Max non-orthogonality allowed.
<b>maxBoundarySkewness</b>	4	Maximum boundary face skewness allowed
<b>minFaceWeight</b>	0.02	Minimum face interpolation weight

### 2.1.5 decomposePar (decomposeParDict)

OpenFOAM cases can be processor intensive. These are by default, run on one core of the processor. The “decomposeParDict” allows to specify the number of cores to use and method to split (decompose) the case. More on “decomposePar” can be read on <https://cfddirect/openfoam/user-guide/v6-running-applications-parallel/>. The table 2.5 below shows the parameters needed for this case, rest (in dictionary body) can be deleted. A copy of the changed dictionary can be found in GitHub under “end.zip/example1/ahmed1”.

Table 2.5: Changes to “decomposeParDict” dictionary.

Parameter	Value	Description
<b>numberOfSubdomains</b>	4	Number of processors (cores) to use.
<b>method</b>	scotch	Method used to decompose case.

## 3 Mesh Generation

### 3.1 Generic Mesh Generation CLI

This section will look at generating a mesh that conforms to the parameters set out in the above sections. This would be done via CLI in a terminal window. Follow the below commands (also located in the “ReadMe” file in the “case” folder) to generate the mesh. You can highlight the command(s) and click the middle mouse button to copy. It can then be pasted on the terminal by clicking the middle mouse button again.

Note: The lines following “#” are not executed. Similar to “//” in OpenFOAM, these are excluded as comments.

```
# Navigate to the case directory (ahmed=name of directory)
cd $FOAM_RUN/ahmed

# Clean the case directory
foamCleanTutorials
foamCleanPolyMesh

# Convert edge1.vtk to eMesh format
surfaceFeatureConvert constant/triSurface/edge1.vtk constant/triSurface/edge1.eMesh

# Extract surface features
surfaceFeatureExtract -dict system/surfaceFeatureExtractDict

# Generating the background mesh
blockMesh

# Decompose case for parallel run
decomposePar
```

```
# Mesh using snappyHexMesh in parallel
mpirun -np 4 snappyHexMesh -parallel -dict system/snappyHexMeshDict
```

Tip: The parallel commands can be broken down as follows;

OpenMPI command	No. of processors	Mesh command	Run parallel	Use dictionary	Dictionary path
mpirun	-np	4	snappyHexMesh	-parallel	-dict System/snappyHex MeshDict

## 3.2 Checking the Generated Mesh

### 3.2.1 Using “checkMesh” Command

To check the quality of the mesh and any errors (skewness, non-orthagonality , etc.), the following command can be run.

```
# Check mesh in parallel
mpirun -np 4 checkMesh -parallel
```

It would give an output similar to **figure 3.1** below. Any deformations (errors, warnings, etc.) would be highlighted by “\*\*” in the beginning of the line. Higher the numbers of “\*” in the beginning of the line indicates the severity.

```
rav@rav-vm: ~/OpenFOAM/rav-dev/run/ahmedb
File Edit View Search Terminal Help

Checking basic patch addressing...
    Patch   Faces   Points
    side2    1600    1701
    inlet     800     861
    outlet    800     861
    side1    1600    1701
    top      3200    3321
    ground   8502    9024
    ahmed1   34452   35092
    ahmed2   4318    4463
    ahmed3    496     558

Checking geometry...
Overall domain bounding box (-3 -2 0) (5 2 2)
Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
Mesh has 3 solution (non-empty) directions (1 1 1)
Boundary openness (-1.9536797e-17 6.0701552e-18 3.344324e-16) OK.
Max cell openness = 3.3226644e-16 OK.
Max aspect ratio = 6.8804802 OK.
Minimum face area = 5.6075054e-06. Maximum face area = 0.010691097. Face area magnitudes OK.
Min volume = 1.6248931e-08. Max volume = 0.0010508195. Total volume = 63.888514. Cell volumes OK.
Mesh non-orthogonality Max: 74.751872 average: 8.0008849
*Number of severely non-orthogonal (> 70 degrees) faces: 24.
Non-orthogonality check OK.
<<Writing 24 non-orthogonal faces to set nonOrthoFaces
Face pyramids OK.
Max skewness = 1.9315824 OK.
Coupled point location match (average 0) OK.

Mesh OK.

End

Finalising parallel run
```

Figure 3.1: Output from "checkMesh".

### 3.2.2 Using ParaView

ParaView can be used to check the mesh visually once finished by typing the following code;

```
ParaFoam -builtin
```

This will launch ParaView with the case ready to be imported as shown in figure 3.2. Please follow the steps below.



Figure 3.2: ParaView with mesh.



Figure 3.3: "Clip" in "Common" toolbar.

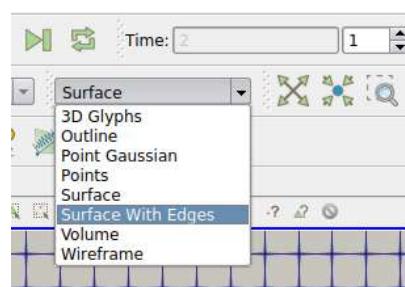


Figure 3.4: Representations toolbar.

You should now see the first step of the meshing process (castellatedMesh). You can check the rest of the mesh by navigating the steps in “VCR Controls” and “Current Time Controls”

(figurer 3.5). You can obtain different views and sections of you mesh and check if it is similar to figure 2.5 as well.



Figure 3.5: "VCR Controls" and "Current Time Controls".

To check the thickness of the layers on the Ahmed body as shown in the [Appendix F](#) comparison;

- 1) Hide the different view/sections created from the pipeline browser by clicking the “eye” next to them.
- 2) Make the “<case>.foam” file visible and select the main geometry (“ahmed1”, etc. as named in [2.1.3](#)) and unselect the “internalMesh” then click “Apply”. You should now be able to see the Ahmed body.
- 3) Select the “thickness” from “Active Variables Toolbar” and this should colour the surface of the Ahmed body with the thickness (the result is shown in [Appendix F](#)).

**Tip:** You can enable the internal mesh, and create different views/sections to see the layer sizing with the colours.

**Tip:** You can switch back to “Surface” from “Surface with Edges” in the “Representations Toolbar”.

**Note:** You will have to go through different time steps as this mesh is in its decomposed stage. Eg: In some cases (example 2), last step may only shows the layers of the supports, not the rest of the body as this was the last “addLayers” step.

### 3.3 Recompiling the Mesh

As mentioned in [section 3.2](#), the mesh is still in its decomposed stage. To reconstruct the completed mesh, use the following command;

```
reconstructParMesh
```

This will compile the mesh with numerical (1, 2, 3, 4, etc.) folders appearing on the case file directory. Each folder corresponds to the different stages (as seen in ParaView in [section 3.2](#)) but with the complete mesh up to that step. Therefore, the last folder contains the completed mesh.

### 3.4 Example 1: Single Ahmed Body with One Refinement Zone

This example will look into meshing an Ahmed body (as a whole) with one refinement zone. The refinement zones allow better detail on areas of aerodynamic interest and saves computing power by localising the refinement.

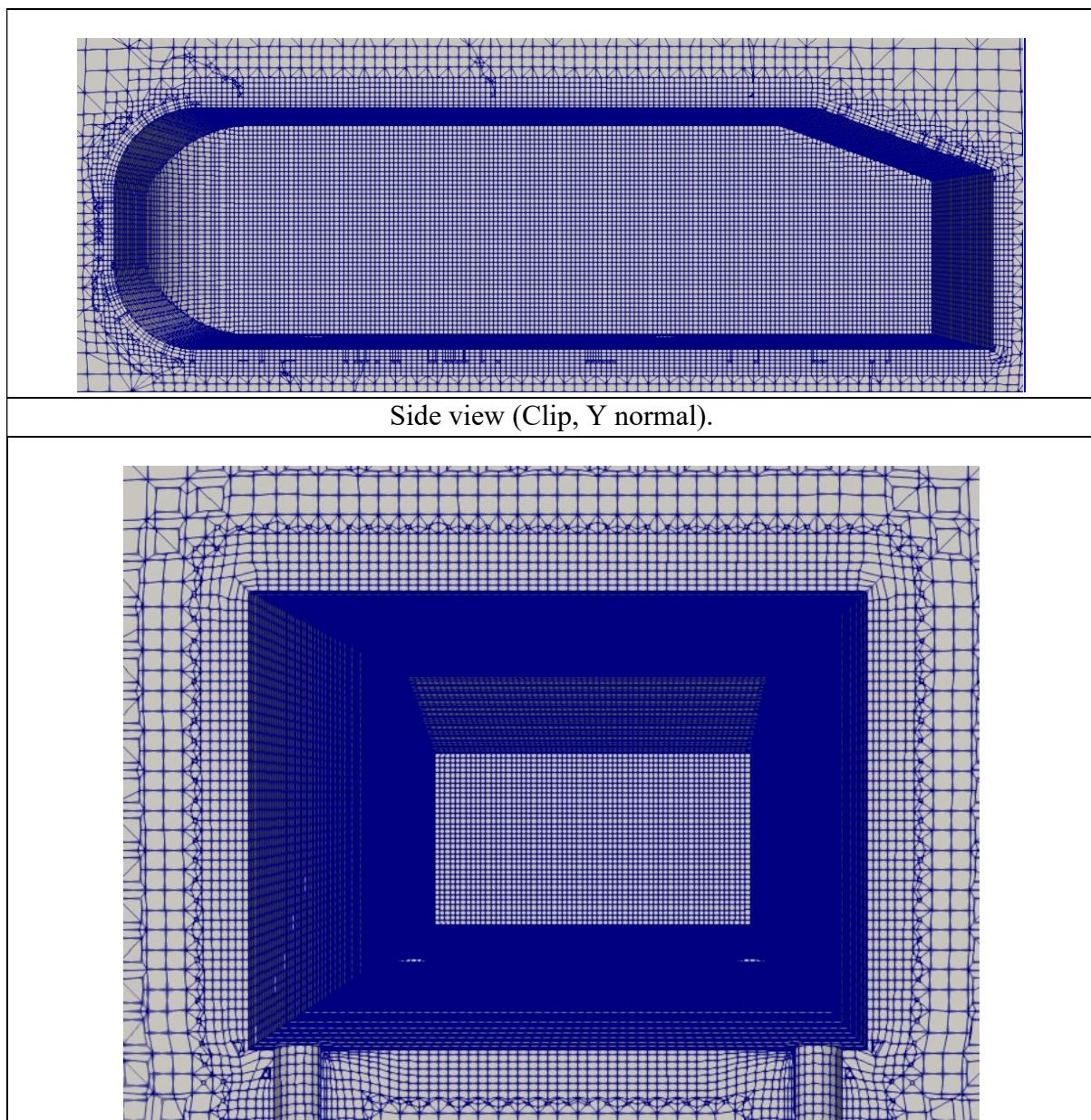
Please follow the steps below to generate and check the above mentioned mesh;

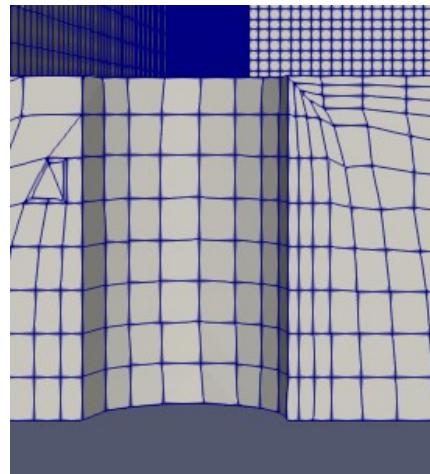
- 1) Move the Ahmed body (only) and reference zone STLs to the “<case>/constant/triSurface” directory.

- 2) Name these “ahmed\_body.stl” and “ref\_zone1.stl”.
- 3) Add the “ahmed\_body.stl” ONLY to “surfaceFeatureExtractDict”.
- 4) Complete the “blockMeshDict” and check the “snappyHexMeshDict”, “meshQuality” and “decomposePar” are set as **show in section 2**.
- 5) Make sure “castellatedMesh”, “snap” and “addLayers” are set to “true” with “nSurfaceLayers” set to 3.
- 6) Generate the mesh by running the commands in **section 3.1**.
- 7) Check the mesh by running the commands in **section 3.2.1**.
- 8) Check the mesh visually by **section 3.2.2**.

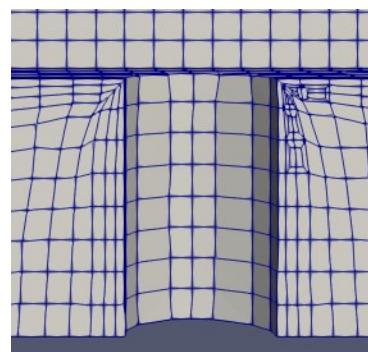
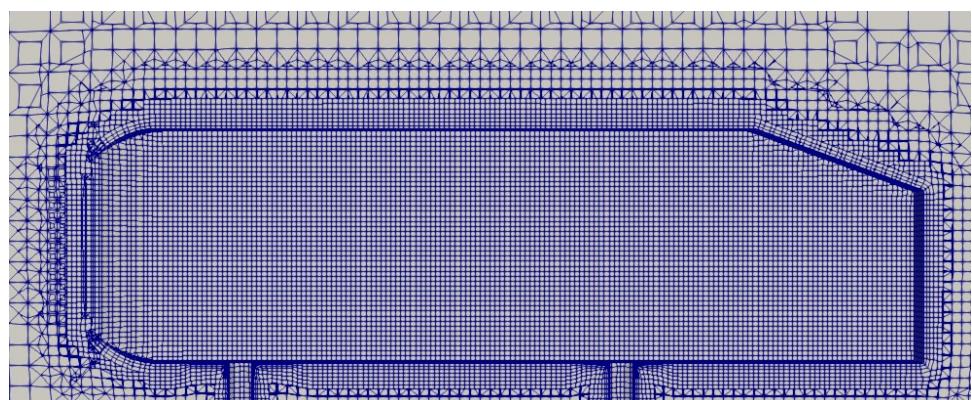
The **table 3.1** below shows the generated mesh and some of its imperfections.

*Table 3.1: Visual inspection of example 1 mesh.*

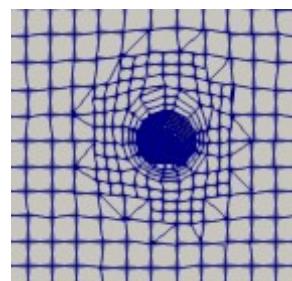




Front view (Clip, X normal).

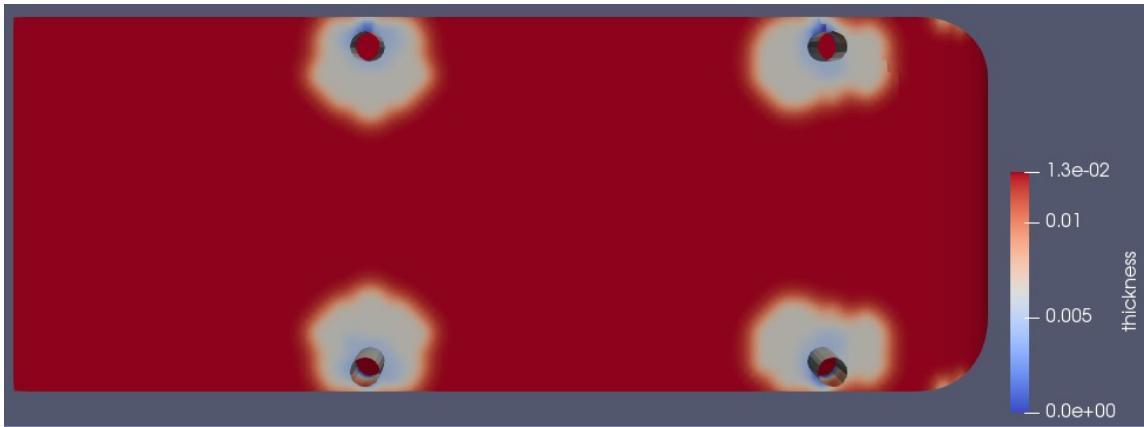


Side view (Clip, Y normal, 0.1635 offset in Y).

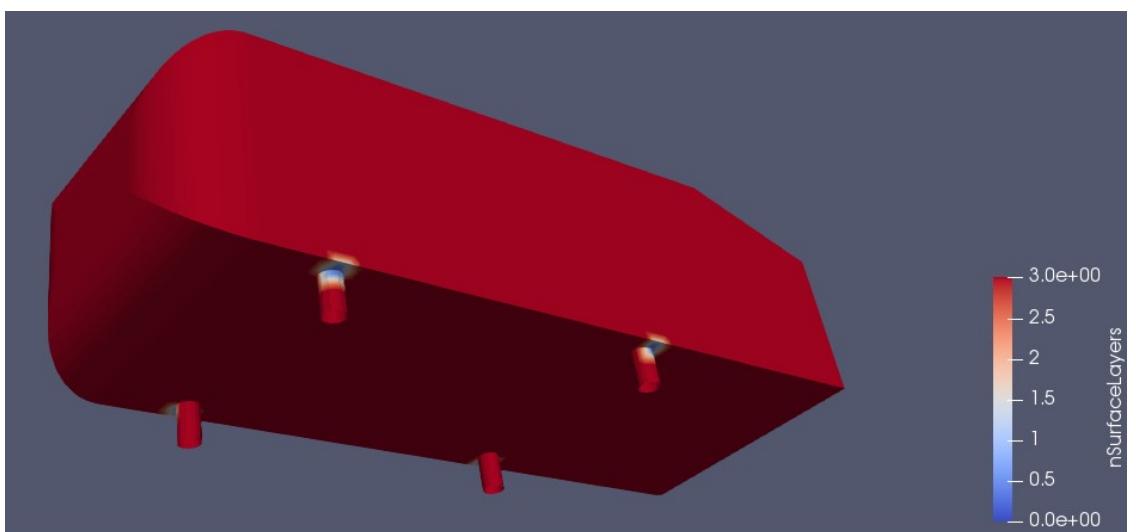


Bottom view.

Inflation layers on the main body look ok from the side but the mesh seems to be collapsing near the supports.



Thickness of the layers.



Number of layers.

Checking for thickness and number of layers confirms the layers have collapsed towards the supports.

### 3.5 Example 2: Split Ahmed Body with 2 Step Meshing.

#### 3.5.1 Splitting the Ahmed Body

This section will look at splitting the Ahmed body into different parts; the front, back and the supports. This will be done on Onshape as shown in steps below. These steps will assume that you have completed the first tutorial and hence skips detailed guidance.

- 1) Open the Ahmed body model on Onshape.
- 2) Use the “Parts” tree to make only the Ahmed body visible.

- 3) Use the “Plane” tool ( ) and select “Top” plane as the “Entities” with “Offset” selected from the dropdown and value as 50 mm (the distance from top plane to the main Ahmed body). Make sure the direction of the offset is as needed (figure 3.6). Click ok ( ) to create a plane between the main Ahmed body and the supports.



Figure 3.6: Creating a new plane by offsetting "Top" plane.

- 4) Use the “Split” tool ( ) with the main Ahmed body selected as the “Parts or surfaces to split” and the “Top” plane as the “Entity to split with”. Click ok ( ) to split the legs from the main body.
- 5) Save the main body (without supports) and the supports (all 4 in 1 file) according to the table 3.2 below. Appendix B shows the exported STLs.

Table 3.2: Export parameters for Ahmed body splits.

File Name	Part
<b>ab1.stl</b>	Main Ahmed body
<b>support.stl</b>	Legs of the Ahmed body
<hr/>	
<b>Format:</b>	STL
<b>STL Format:</b>	Binary
<b>Units:</b>	Metre
<b>Resolution:</b>	Fine
<b>Options:</b>	Download

### 3.5.2 Generating the Mesh in 2 Steps

This example will use the split body created in section 3.5.1 with hopes of refining the mesh further. The process can be summarised in figure 3.7.

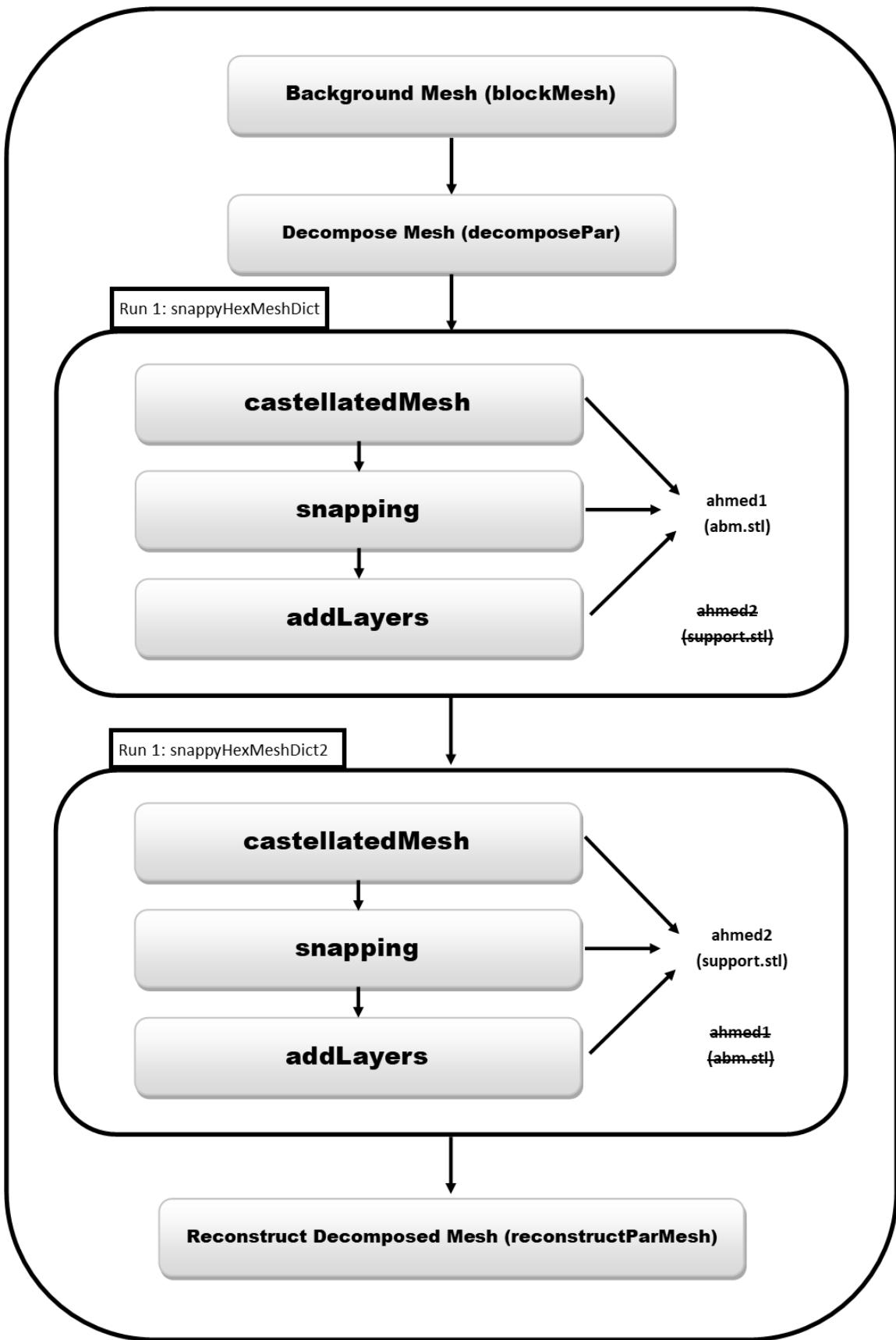


Figure 3.7: Two step meshing process.

Please follow the steps below to generate and check the above mentioned mesh;

- 1) Move the 2 STLs created in [section 3.5.1](#) to the “<case>/constant/triSurface” directory.
- 2) Name these “abm.stl” and “supports.stl”. Make sure the “ref\_zone1.stl” is in the folder as well.
- 3) Add the “abm.stl” and “supports.stl” to “surfaceFeatureExtractDict”.
- 4) Complete the “blockMeshDict” and check the “meshQuality” and “decomposePar” are set as [show in section 2](#).
- 5) Open the “snappyHexMeshDict” and include the “abm.stl” and “support.stl” to the dictionary. This can be done by modifying the already existing “ahmed\_body.stl” lines to “abm.stl”. They can then be copied with “abm.stl” replaced by “supports.stl” in the copied lines.
- 6) Similar to the above step, add the 2 new geometries and remove the old one from;
  - a. “Explicit feature edge refinement”
  - b. “Surface based refinement”
  - c. “Region-wise refinement”
  - d. “Settings for the layer addition”
- 7) Make sure “castellatedMesh”, “snap” and “addLayers” are set to “true” with “nSurfaceLayers” set to 3.
- 8) For “Run 1” comment out sections related to “supports.stl” (also called “ahmed2”). Save this dictionary.
- 9) For “Run 2” comment out sections related to “abm.stl” (also called “ahmed1”). DO NOT use “Save” and overwrite the dictionary in step 8. Instead use “Save As” and name it “snappyHexMeshDict2”.
 

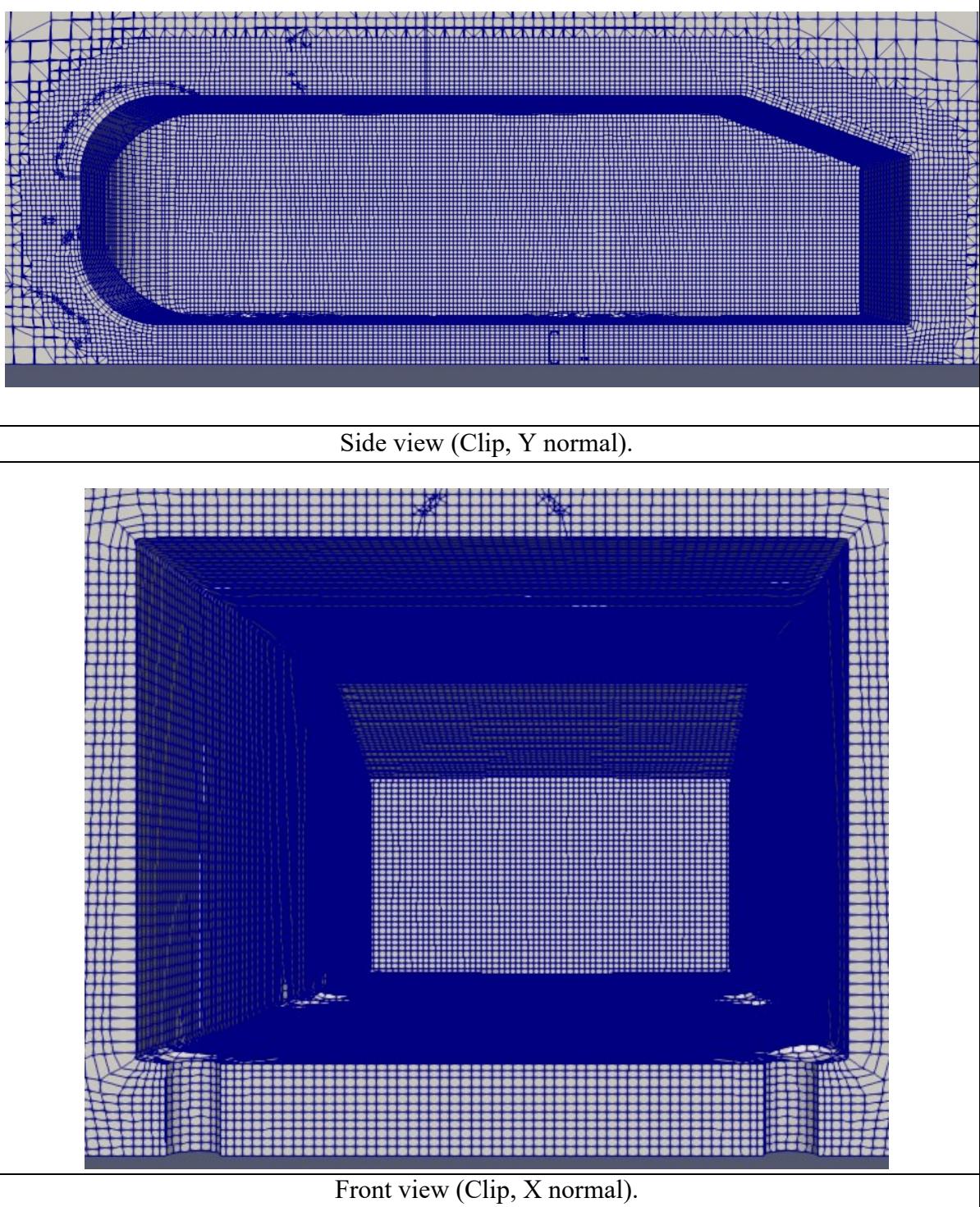
**Note:** Copies of the changed dictionaries can be found in GitHub under “end.zip/example2/ahmed3”.
- 10) Generate the mesh by running the commands in [section 3.1](#) until the first “snappyHexMesh” run.
- 11) Repeat the previous step using “snappyHexMeshDict2” for the second run.

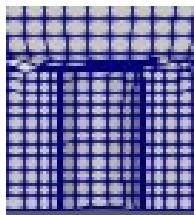
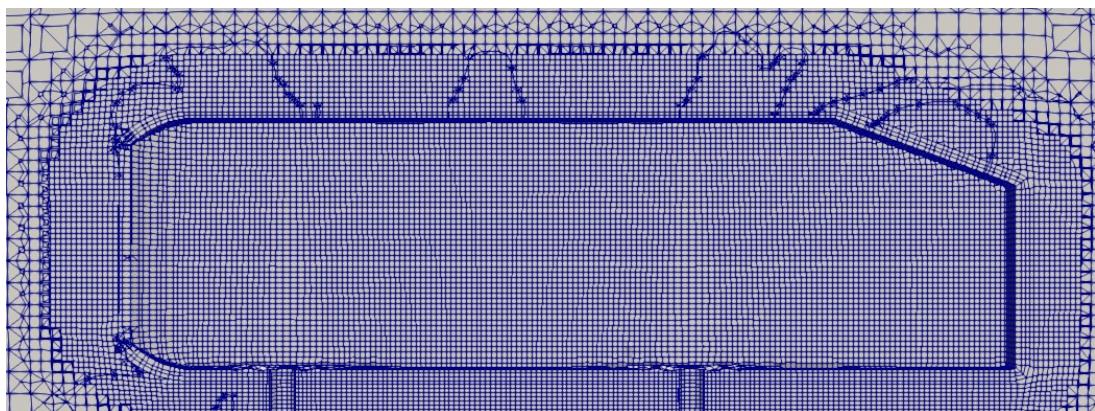
```
# Run 2: Mesh using snappyHexMesh in parallel for supports
# using snappyHexMeshDict2
mpirun -np 3 snappyHexMesh -parallel -dict system/snappyHexMeshDict2
```

- 12) Check the mesh by running the commands in [section 3.2.1](#).
- 13) Check the mesh visually by [section 3.2.2](#).

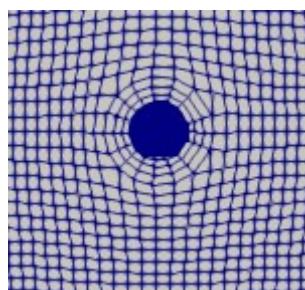
The [table 3.3](#) below shows the generated mesh and some of its imperfections.

*Table 3.3: Visual inspection of example 2 mesh.*



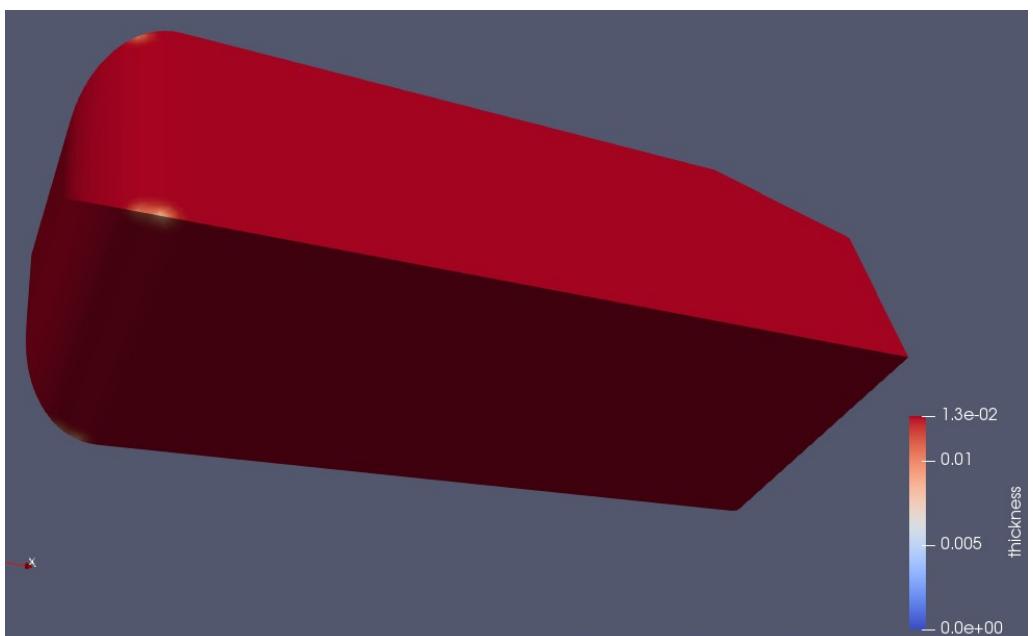


Side view (Clip, Y normal, 0.1635 offset in Y).

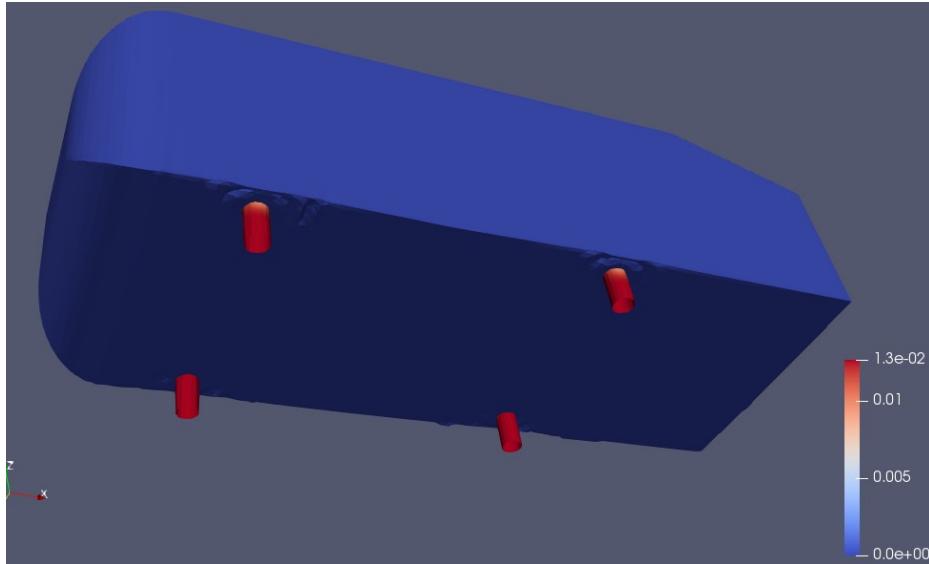


Bottom view.

The inflation layers on the legs are better, some distortion in the geometry, needs to be fixed.



“ahmed1” (“ab1.stl”) layer thickness.



Number of layers on “ahmed1” (“ab1.stl”).

When checking for thickness, the layers seem to be thin but not collapsing. Better than in section 3.3 but the surface at the bottom has some irregularities. Further improvement possible.

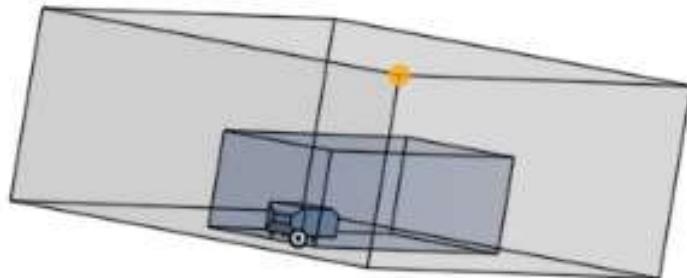
### 3.6 Other Improvements

Further improvements can be made to the mesh, such as the addition of refinement zones (“ref\_zone1.stl”) to various interest areas of the body, by splitting the body into several parts and refining these individual parts with different refinement settings (such as levels, etc.).

In your own time, try to improve the above mesh. If you need support, please do not hesitate to e-mail me ([r.t.b.ranaweera@northumbria.ac.uk](mailto:r.t.b.ranaweera@northumbria.ac.uk)) and CC in Dr. Martin ([nick.d.martin@northumbria.ac.uk](mailto:nick.d.martin@northumbria.ac.uk)).

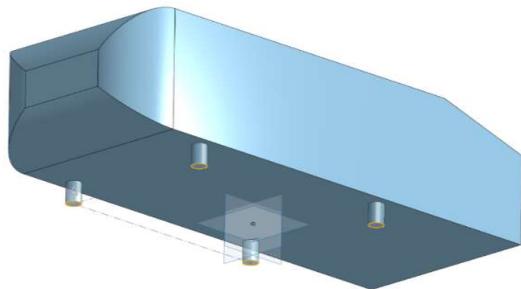
## Appendix A: Co-ordinates from Onshape

Co-ordinates of points (vertices) can be easily obtained from Onshape by clicking on the point/vertex and looking at the bottom left corner of the screen.

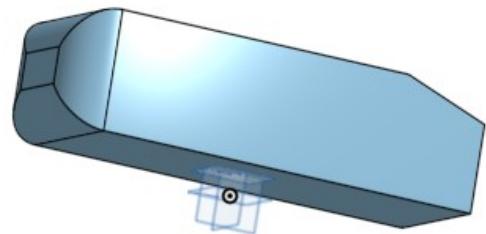


Point: X: -3000.000 Y: -2000.000 Z: 2000.000 r

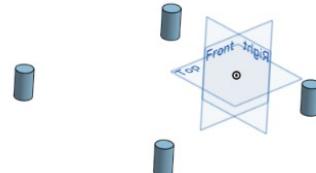
## Appendix B: STL Exports



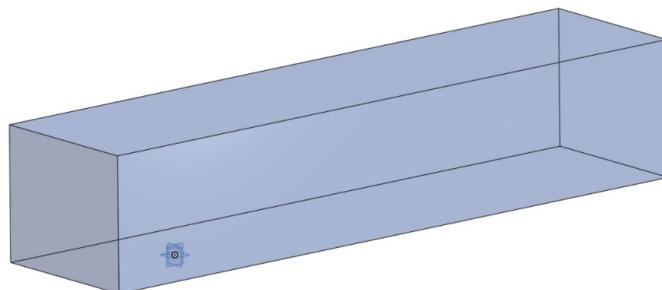
ahmed\_body.stl



abm.stl



support.stl



ref\_zone1.stl

## Appendix C: Copying Tutorial Cases and Preparing

In your PC, the “run” directory which OpenFOAM cases are run from can be accessed via terminal from “\$FOAM\_RUN” from anywhere. You do not need to type the full path. Similarly, “FOAM\_TUTORIAS” points to original tutorials storage directory.

- 1) Check if the \$FOAM\_RUN directory exist;

```
ls $FOAM_RUN
```

Tip: Use “ls” command to list files and directories within and folder.

If the directory does not exist, create one by;

```
mkdir -p $FOAM_RUN
```

- 2) Change to the \$FOAM\_RUN directory;

```
cd $FOAM_RUN
```

Tip: Typing “\$HOME” would navigate you back to the home folder and “cd ..” would navigate you back one folder.

- 3) Copy the “motorBike” tutorial to your working (run) directory;

```
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/motorBike .
```

**Note:** The commands, files and folder names are case sensitive.

Tip: Pressing the TAB key after partially typing enough letters of a command/filename would auto-complete it.

Tip: The “-r” ensures all directories and their subdirectories are copied and the “.” after refers it to copy to the current directory.

- 4) Rename the copied case by;

```
mv motorBike $FOAM_RUN/ahmed
```

- 5) Navigate to the copied case directory by;

```
cd ahmed
```

- 6) Clean the directory and delete the unnecessary files/folders;

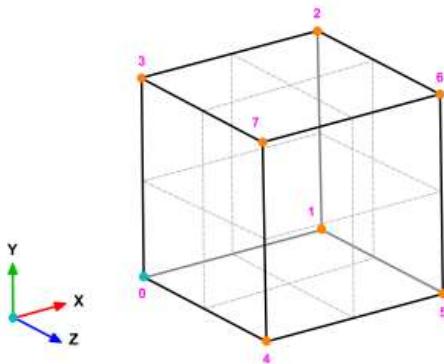
```
foamCleanTutorials  
foamCleanPolyMesh  
rm Allclean Allrun  
rm -r 0/include  
rm system/streamLines  
rm system/cuttingPlane  
rm system/forceCoeff
```

- 7) Rename the U.orig file to U;

```
mv 0/U.orig 0/U
```

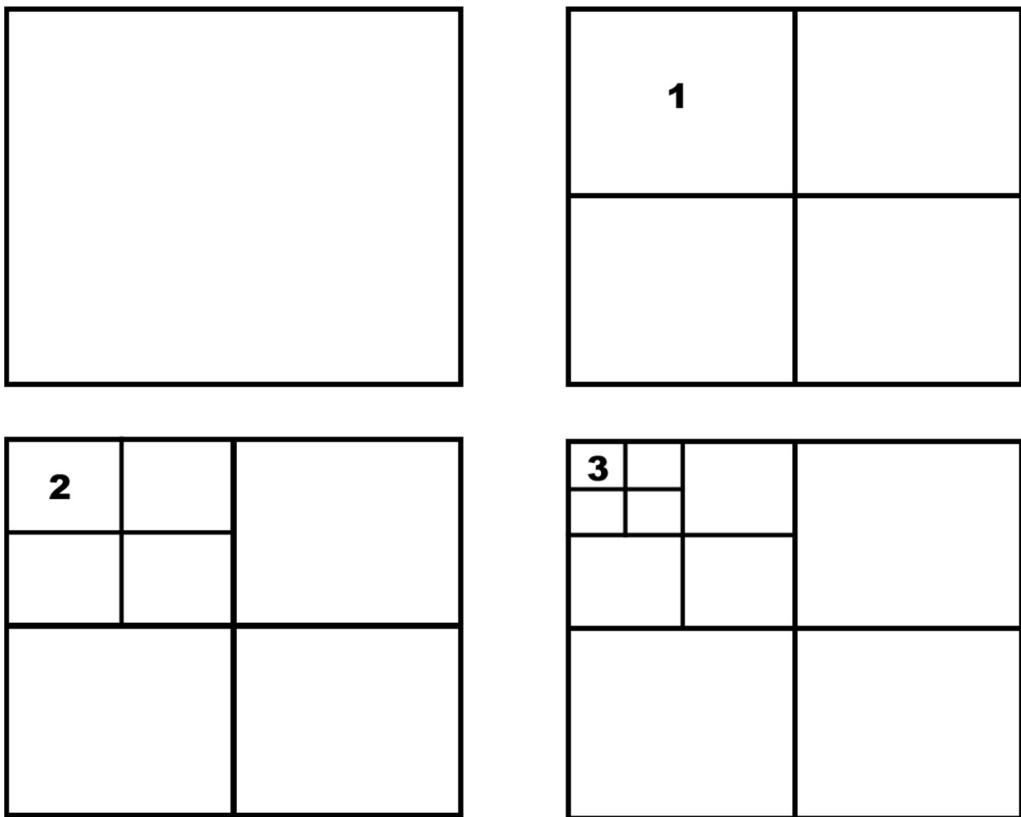
- 8) Delete files in “constant/triSurface” folder and Copy the geometries (STL files and VTK file) generated in previous steps via GUI (drag and drop) or CLI (using “cp” command). A copy of the prepared case can be found in GitHub under “Start.zip”.

## Appendix D: Vertices of a the Geometry



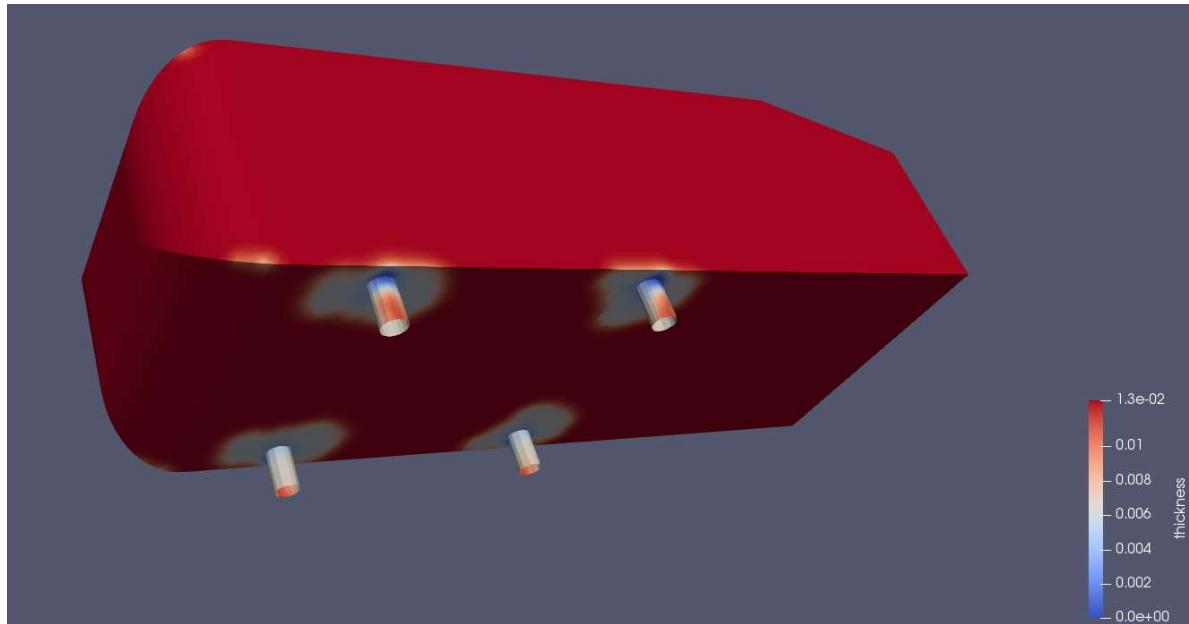
## Appendix E: Refinement Levels

Refinement level is the splitting of a hexahedron into different smaller parts. This is illustrated in the figure below.

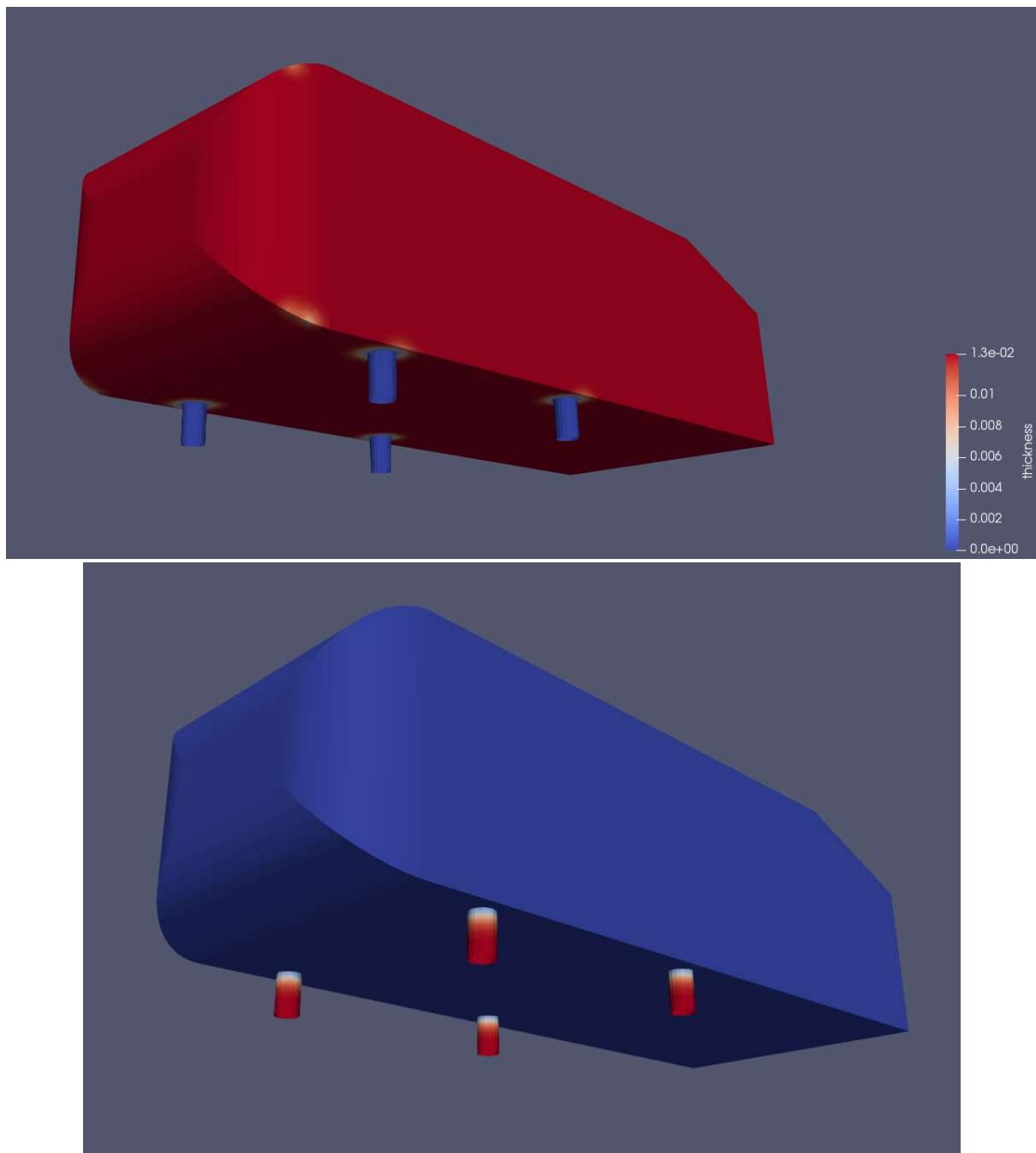


## Appendix F: Single Run vs 2 Runs (snappyHexMesh)

Single run:



Two runs (separate “addLayers” for supports):



Note: The images are in 2 different steps (stages) due to layers for supports being applied after meshing of the main body is complete.

## Recommended Learning Content

- <https://cfd.direct/openfoam/user-guide/v6-mesh/>
- <https://www.openfoam.com/documentation/user-guide/mesh.php>
- [http://www.wolfdynamics.com/wiki/meshing\\_OF\\_blockmesh.pdf](http://www.wolfdynamics.com/wiki/meshing_OF_blockmesh.pdf)
- [http://www.wolfdynamics.com/wiki/meshing\\_OF\\_SHM.pdf](http://www.wolfdynamics.com/wiki/meshing_OF_SHM.pdf)
- <https://www.youtube.com/watch?v=bRS0n8FuFVY&list=PLoI86R1JVvv-EN7BsoyomcWJIPaVPXaHO>