# Combustor case: mesh generation

Federico Piscaglia *
Dept. of Aerospace Science and Technology (DAER), Politecnico di Milano, Italy

**Abstract.** Lab handout for the use of `snappyHexMesh` to generate the computational grid of the "combustor" case in OpenFOAM®-8.

## 1 Learning outcome

In this section you will learn how to:

- Prepare an STL file to be used with `snappyHexMesh`
- Generate the background mesh using `blockMesh`
- Create the mesh using `snappyHexMesh`
- Check the results

## 2 Prepare the run

### Copy the tutorial case

1. `student@docker-PoliMi:run$ cp -r ahmedBody combustorMesh`
2. `student@docker-PoliMi:run$ cd combustorMesh`
3. `student@docker-PoliMi:combustorMesh$ ./Allclean`

### 2.1 Prepare the geometry

1. Download the file `combustor_mm.stl.gz` from the course website
2. Save it in the folder `$FOAM_RUN/combustorMesh/constant/triSurface`
3. Uncompress it:
   `student@docker-PoliMi:triSurface$ gunzip combustor\_mm.stl.gz`
4. In some configuration, it appears as the download file had been compressed twice. To test the decompression type: `file combustor.stl`
   - if the output is:
     `combustor_mm.stl: ASCII text`
     then the decompression has been successful. Proceed to point **??**.

*Tel. (+39) 02 2399 8620, E-mail: federico.piscaglia@polimi.it

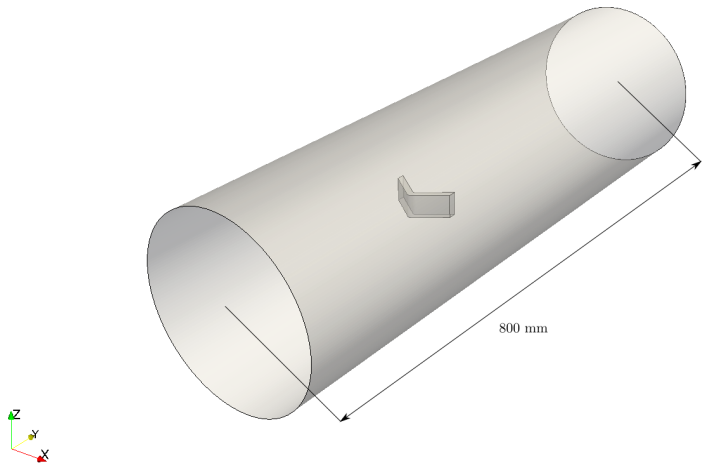


Figure 1: geometry

- if the output is:
  `combustor_mm.stl: gzip compressed data, was "combustor_mm.stl" [...]`
  then the decompression did not work. To fix:
  i. `mv combustor_mm.stl combustor_mm.stl.gz`
  ii. `gunzip combustor_mm.stl.gz`
  iii. `file combustor_mm.stl`

5. Check the geometry:
   `surfaceCheck combustor.stl`
   Optionally, you can open the file with paraview:
   `paraview combustor_mm.stl`
6. Look at the bounding box. Does it look correct? Recall that OpenFOAM considers lengths as expressed in meters
7. If necessary, scale down the surface:
   `surfaceTransformPoints -scale "(1e-3 1e-3 1e-3)" combustor.stl combustor_mm.stl`

## 2.2 Generate the background mesh

The background mesh will be generated using blockMesh. In the tutorial case a sample blockMesh is already there. We need to adapt it to the new geometry.

1. Re-run surfaceCheck on the new file with the option `-blockMesh` and save the log
   `surfaceCheck -blockMesh combustor_mm.stl > log.surfaceCheck`
2. Open blockMeshDict and paste the lines in log.surfaceCheck enclosed between `// blockMeshDict info` and `// end blockMeshDict info`
   (actually discard "edges" and "patches" entries)
   `cd ../../`
   `student@docker-PoliMi:combustorMesh$ vi system/blockMeshDict`
   `student@docker-PoliMi:combustorMesh$ vi constant/triSurface/log.surfaceCheck`

3. make the bounding box smaller in the y-direction: reduce the y-coordinates by 5e-4 on both sides
4. run blockMesh

   `student@docker-PoliMi:combustorMesh$ blockMesh`
5. Open the mesh with ParaView and compare it with the STL. Adjust the number of cells along each direction to have an aspect ratio as close as possible to 1.

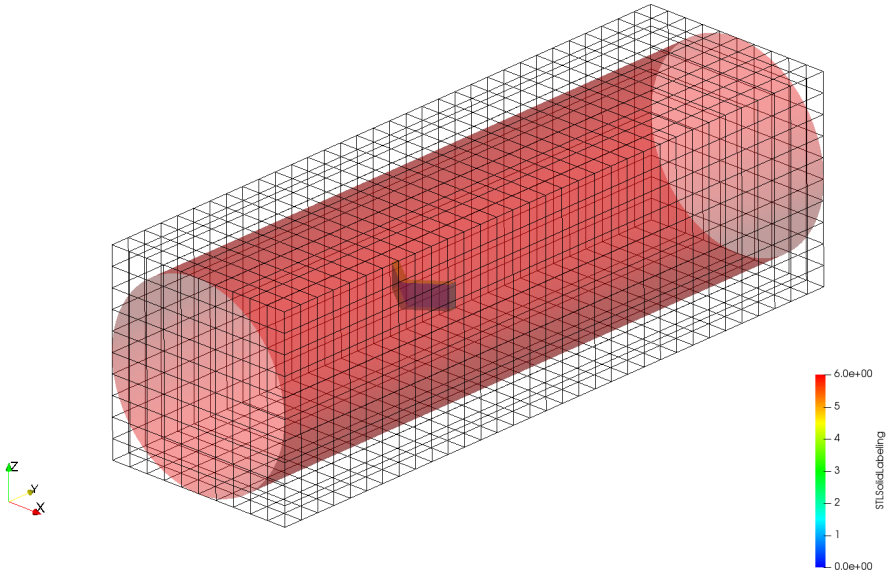


Figure 2: Background mesh and STL

## 2.3 Extract feature edges

1. Open `surfaceFeaturesDict` and substitute `combustor_mm.stl` to the old filename ('ahmedBody.stl')
2. Check all parameters
3. Run

   `surfaceFeatures`
4. Check the results by opening with PV the file:

   `constant/extendedFeatureEdgeMesh/combustor_mm_edgeMesh.obj`
5. If necessary, modify parameters and loop back to point **??**

# 3 Snap the mesh

Open the file `system/snappyHexMeshDict`

## 3.1 Geometry

1. Change the main geometry filename with `combustor_mm.stl`, and change the 'internal' name as well
2. Check region names in the STL:

   `user@host:combustorMesh$ grep solid constant/triSurface/combustor_mm.stl`

3. write an entry for each region of the STL

   **Hint:** look for the keyword 'solid' in the STL. Use:

   `grep solid constant/triSurface/combustor_mm.stl`

4. Define a refinement box centered on `(0 0.1 0)` and with $Lx = 0.1$, $L_y = 0.3$, $L_z = 0.06$

## 3.2 Castellated mesh

1. In the subdictionary 'features' change "ahmedBody.eMesh" with "combustor_mm.eMesh". Set an appropriate level of refinement (TIP: 4)
2. In subdictionary 'refinementSurfaces' change part names with the new ones. (TIP: you can use wildcards: `"splitter.*"`)
3. Set levels of refinement (TIP: 0 for overall, 3/4 on outer walls, 4/5 on splitter)
4. Set refinement level for refinement box
5. Set "locationInMesh" (TIP: `(0 -0.1 0)`)
6. Activate the 'castellatedMesh' flag at the beginning of the file:

```
// Which of the steps to run
castellatedMesh 1;
snap            0;
addLayers       0;
```

7. Run snappyHexMesh and check results with ParaView
8. If results are not satisfactory, delete folder '1' and loop from point **??**

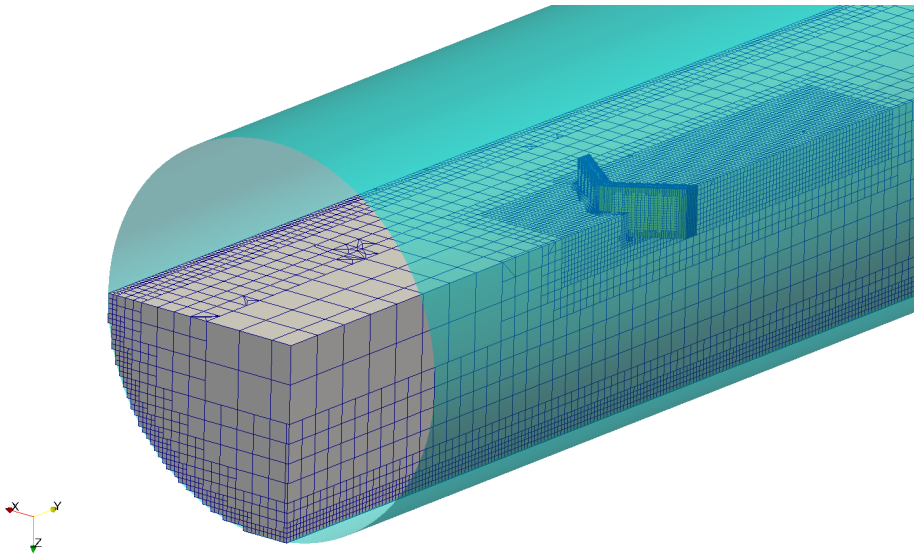

Figure 3: Castellated mesh

## 3.3 Snapped mesh

1. Keep default values in the 'snapControls'
2. Activate the second switch:

```
// Which of the steps to run
castellatedMesh 0;
snap            1;
addLayers       0;
```

3. run snappyHexMesh
4. check the results (remember to move PV to the last timestep) and iterate if necessary after deleting subfolder '2'

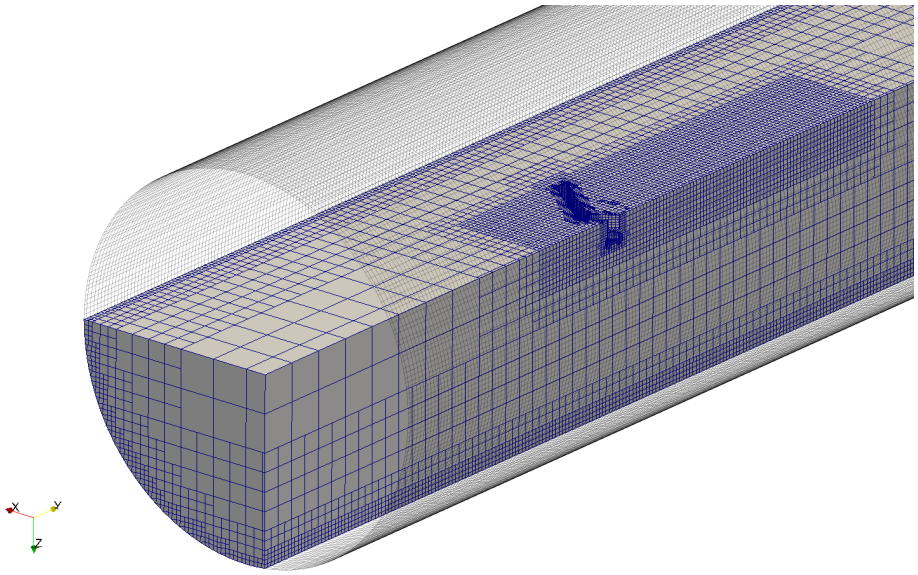

Figure 4: Snapped mesh

## 3.4 Wall layer extrusion

1. Set the `addLayerControls`:
   - Use absolute sizes;
   - add 5 layers on the outer walls and 3 on the splitter (use wildcards!)
   - first layer thickness: 0.1 mm
   - Expansion ratio
2. minThickness = 0.05 mm
3. activate the third switch

```
// Which of the steps to run
castellatedMesh 0;
snap            0;
addLayers       1;
```

4. run snappyHexMesh
5. Check the results. To debug the layer addition process show only the patches and activate the layers variables (nSurfaceLayers, thickness, thicknessFraction)

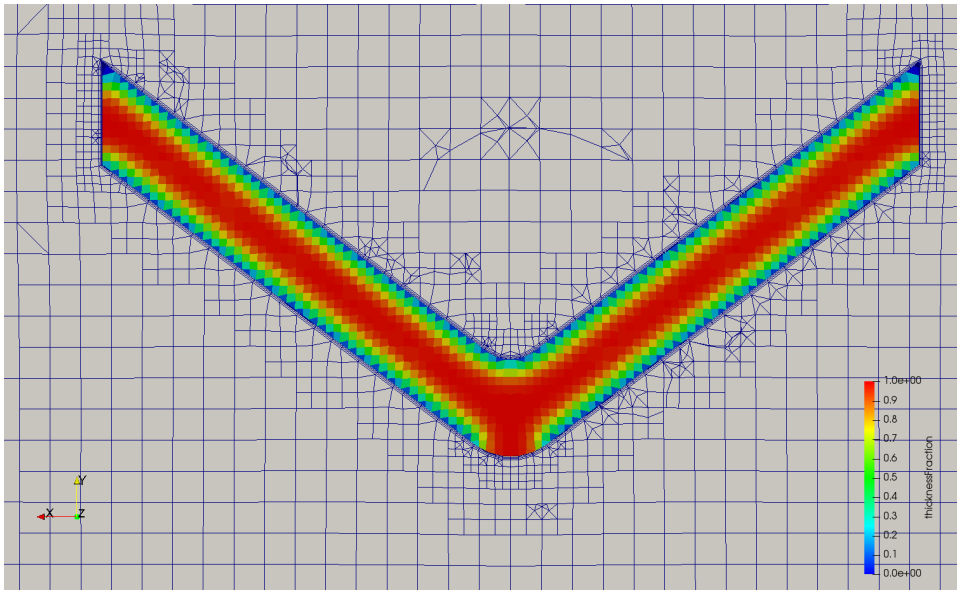


Figure 5: Mesh with extruded layers