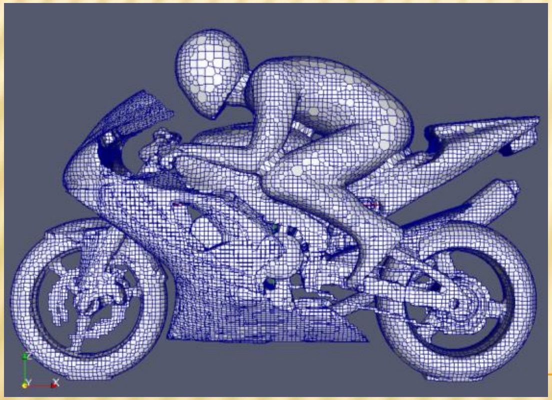
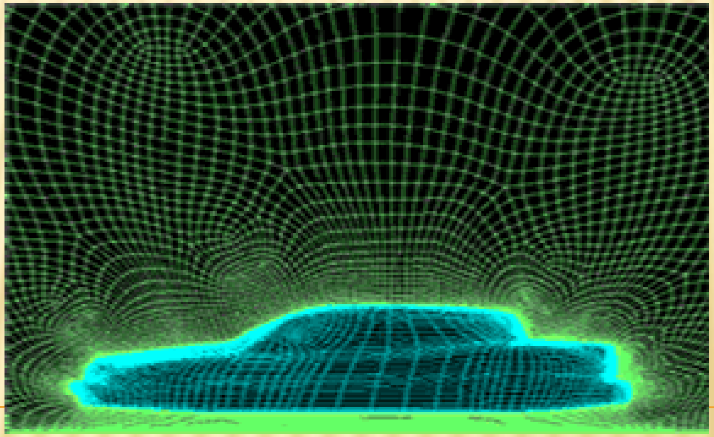
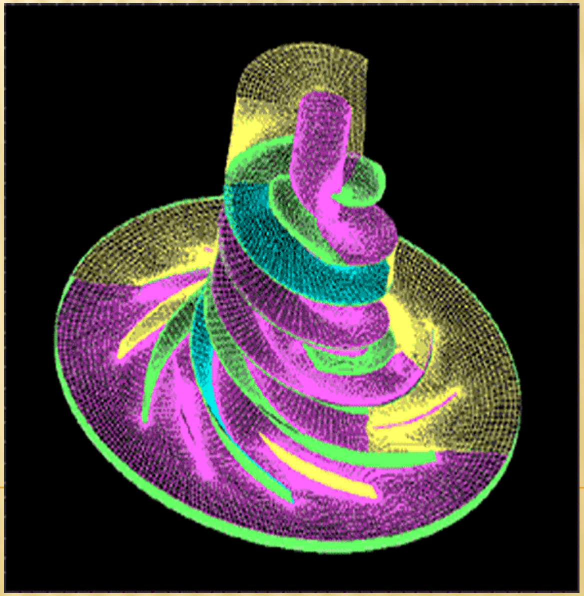
Pre-processing in openfoam, mesh generation.







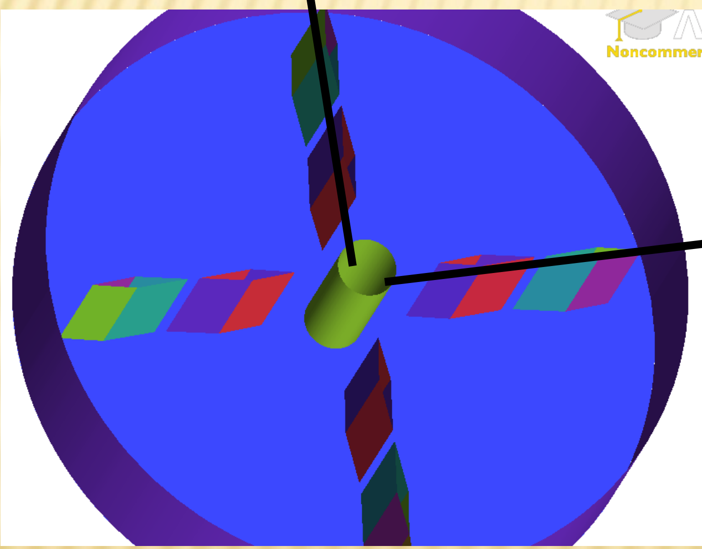
OpenFOAM kurs 2013 Håkan Nilsson Olivier Petit

Different ways of creating the mesh. Outline

- Using **SnappyHexMesh**, an OpenFOAM mesh generation tool.
- Using **blockMesh, m4** and **python**.
- Importing the mesh from **external software**.

A tutorial for snappyHexMesh.

- SnappyHexMesh generates a 3D mesh from a .stl file. (triangulated surface geometry)
- For this tutorial, a simplified pump geometry is chosen.



For more simplicity in computations, the symmetry of the geometry is used, and only one quarter of the pump is meshed.

A tutorial for snappyHexMesh.

- **Download** the tutorial from Håkan’s webpage.
- **Source** OpenFOAM 2.2.x with alias OF22x or

  export FOAM_INST_DIR=/chalmers/sw/unsup64/OpenFOAM;

  . $FOAM_INST_DIR/OpenFOAM-2.2.x/etc/bashrcHani
- To check if the right OpenFOAM was called:
  which SimpleFoam
  It should point to simpleFoam in OpenFOAM-2.2.x
- In the tutorial case, you should find:

  1. The .stl file located in constant/triSurface.
  2. A dictionnary called snappyHexMeshDict in system/.

A tutorial for snappyHexMesh.

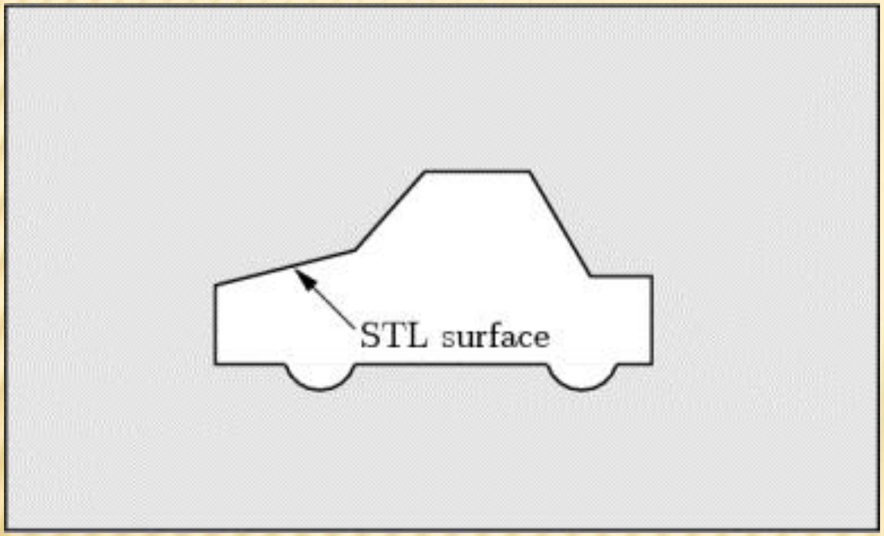
- **Requirement for snappyHexMesh to work:**
  - snappyHexMeshDict in /system/
  - Geometry data in constant/triSurface
  - Hexahedral base mesh (decomposed if running in parallel)
  - decomposeParDic file in /system/
  - All system dictionaries (e.g. controlDict, fvSchems, fvSolutions)

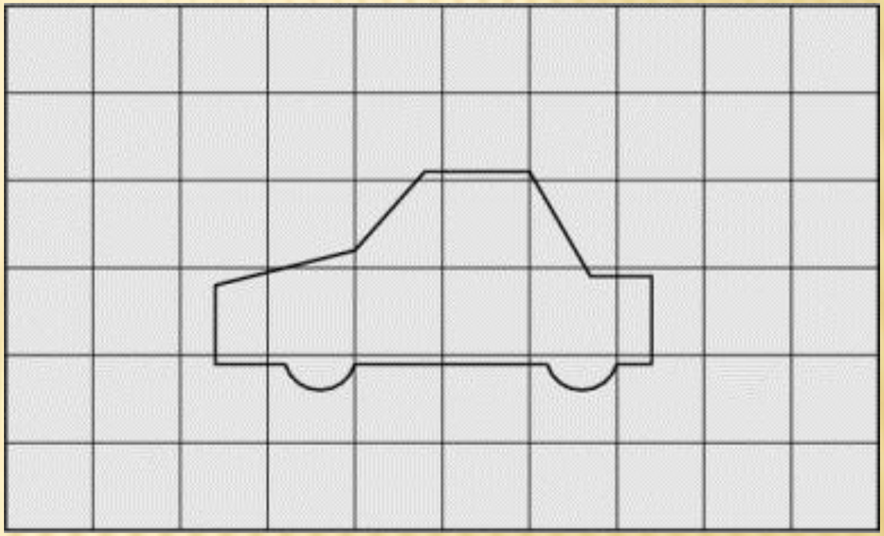
A tutorial for snappyHexMesh.

- **5 simple steps:**
  - Create base mesh
  - Refine base mesh
  - Rmove unused cells
  - Snap mesh to surface
  - Add layers

snappyHexMesh: step 1

- **Creation of a grid surrounding the stl surface.**



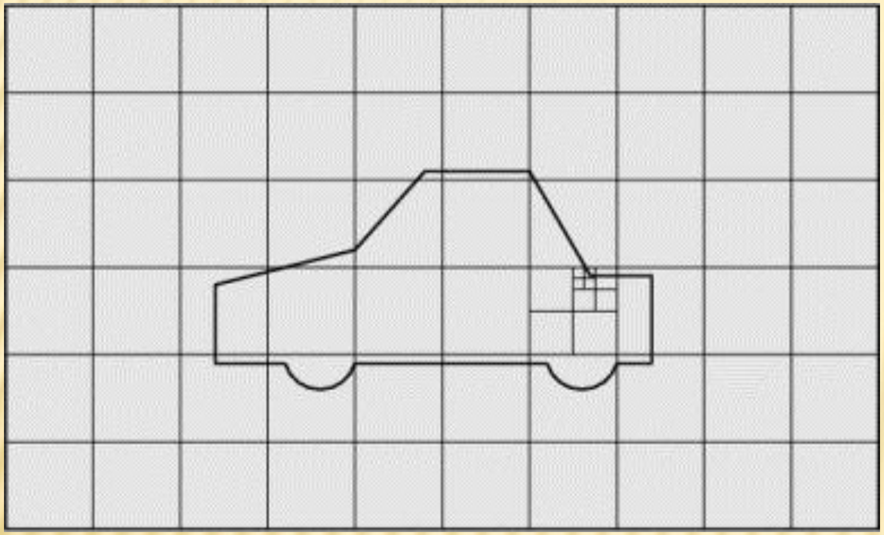


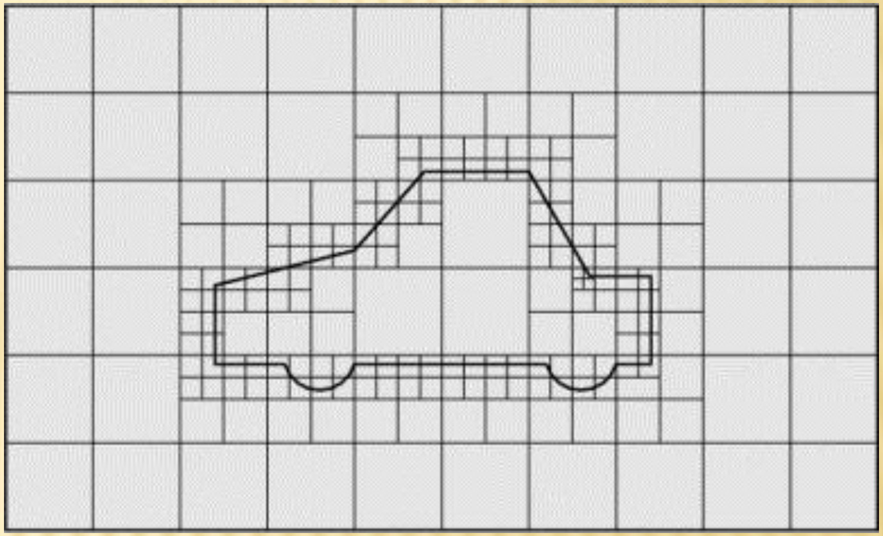
Characteristics of the grid:

✓ *The aspect ratio of the grid cells should be around 1.*
✓ *More than one cell in the z direction.*
✓ *At least on cell's edge should intersect the surface.*
✓ *There can not be empty patches, it is a 3D mesher.*

SnappyHexMesh: step 2

- **Refine the base mesh.**





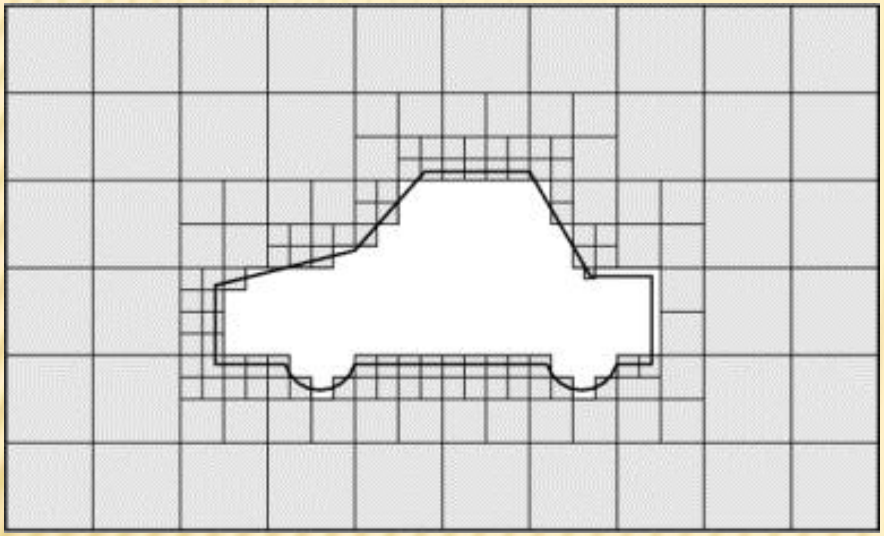
- Start the splitting from **locationInMesh** feature.
- This edge must be **inside the region to be meshed** and **must no coincide with a cell face.**

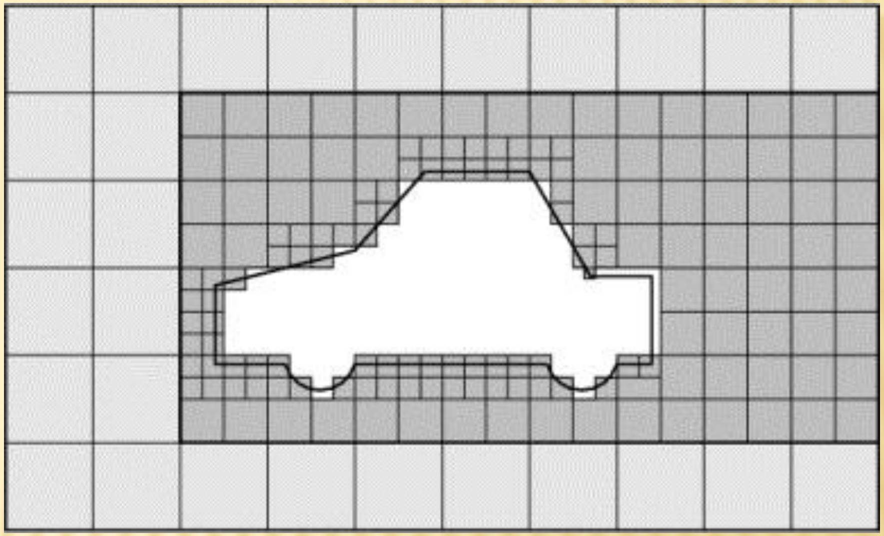
- Splitting the cells around the surface according to :

```
refinementSurfaces
  {
     file.stl
     {
         level (2 2); // default (min max) refinement for  surface
      }
  }
```

Snappyhexmesh: step 3

- **Remove unused cells.**





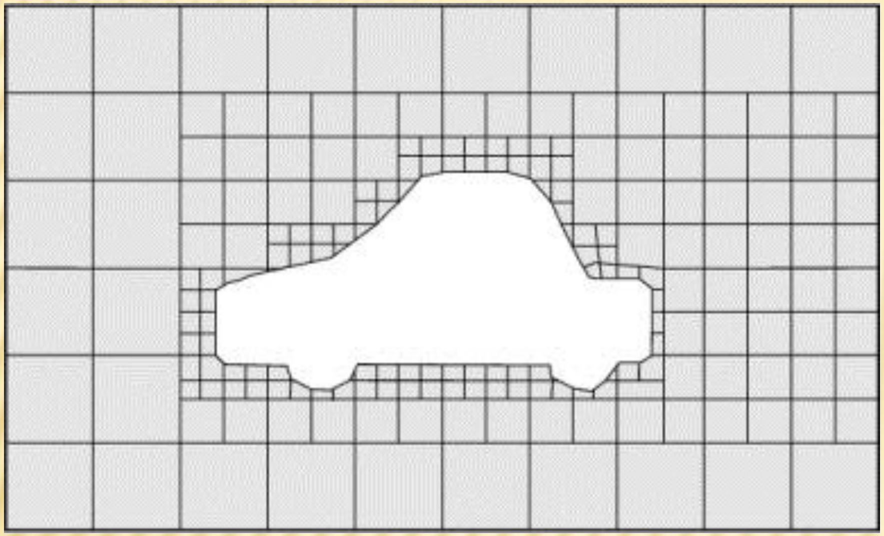
• Keep the side of the mesh defined by **locationInMesh.**
• Remove all cells that have above 50% of their volumes in the meshed region.

The region refined is specified by:
❖ **inside**: inside the volume region.
❖ **outside**: outside the volume region.
❖ **distance**: according to distance to the surface.
The region is defined first as geometry.

This first step is saved into the time folder 1.

Snappyhexmesh: step 4

- **Snapping to surface.**

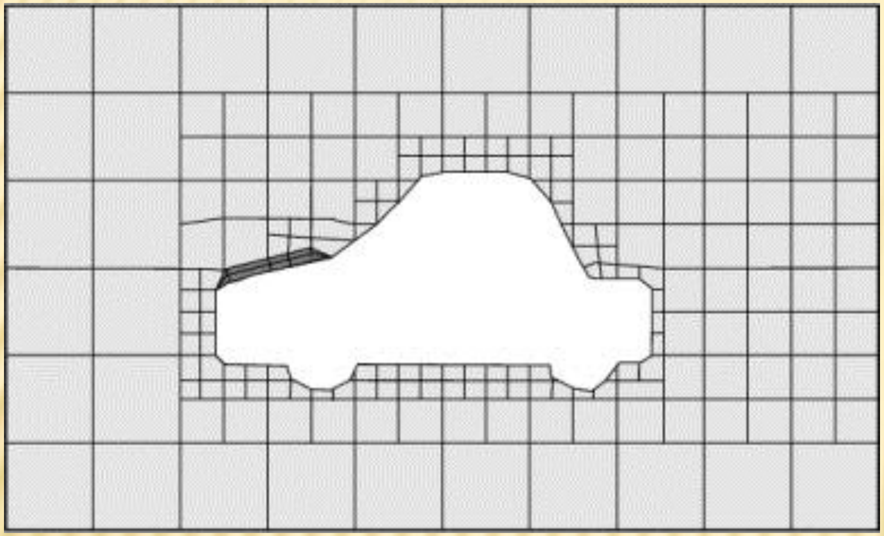


The steps to snap to surface are:
1. **Snap** the vertices onto the STL surface.
2. **Solve** for relaxation of the internal mesh.
3. **Iterate** using the snapControls in SnappyHexMeshDict.

This second step is saved into the time folder 2.

Snappyhexmesh: step 5

- **Boundary layer addition.**



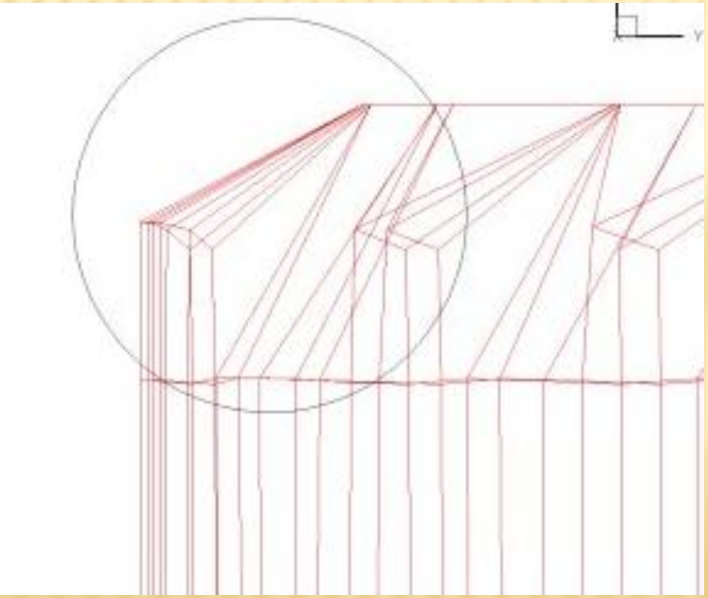
The boundary are applied on patches, not on surface!!

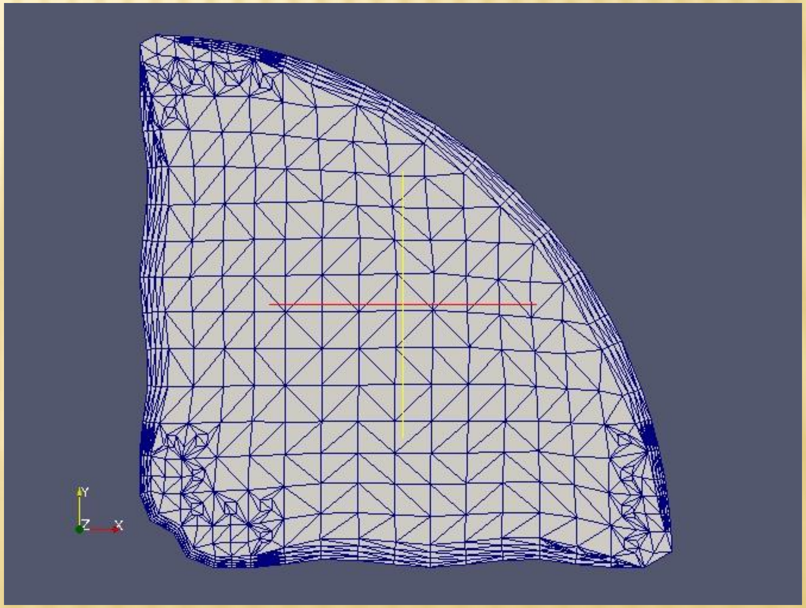
- Mesh projection back from the surface using a specified thickness normal to the surface.
- Solve for relaxation of the internal mesh with the latest projected boundary vertices.
- Check if validation criteria are validated.
- If the validation criteria can be satisfied, insert mesh layers.

This last step is saved into the time folder 3.

Snappyhexmesh: pros and cons.

- Possibilities of multiple refinements that make it very robust.
- It runs in parallel.
- Need of a good quality STL surface, with more than one region/patch.
- The feature line is now available. SurfaceFeatureExtract
- A Boundary Layer mesh is not easy to obtain, it requires experience and trial and error method.



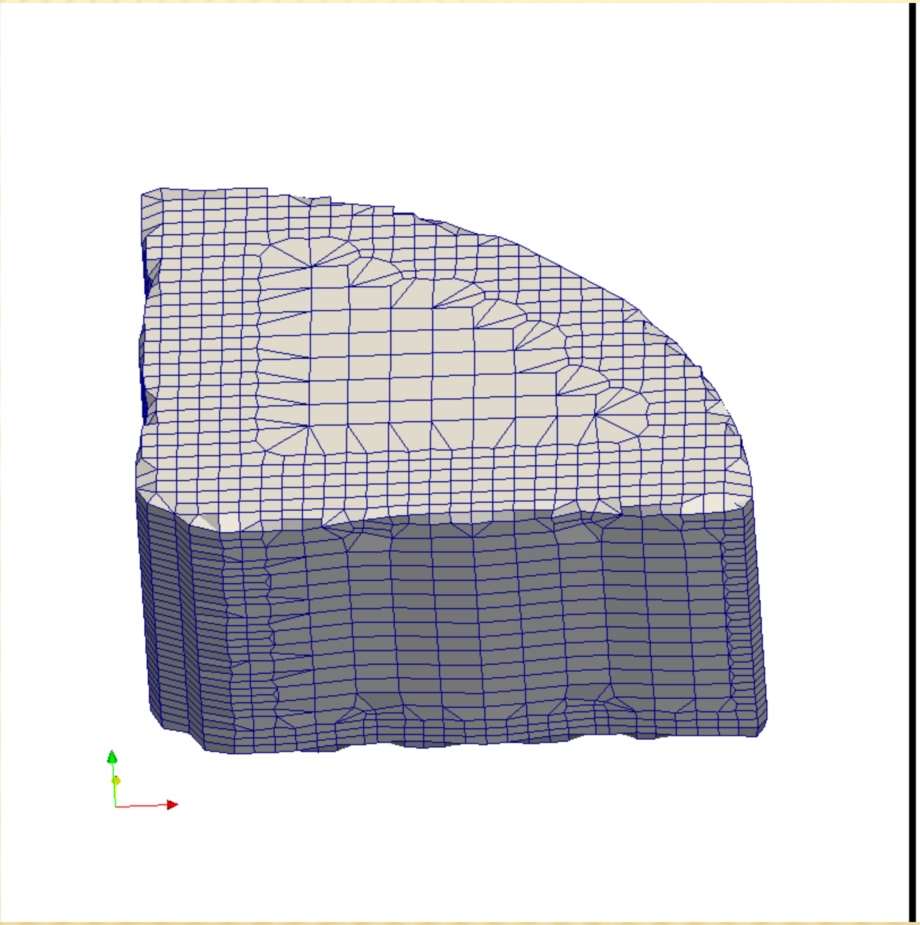


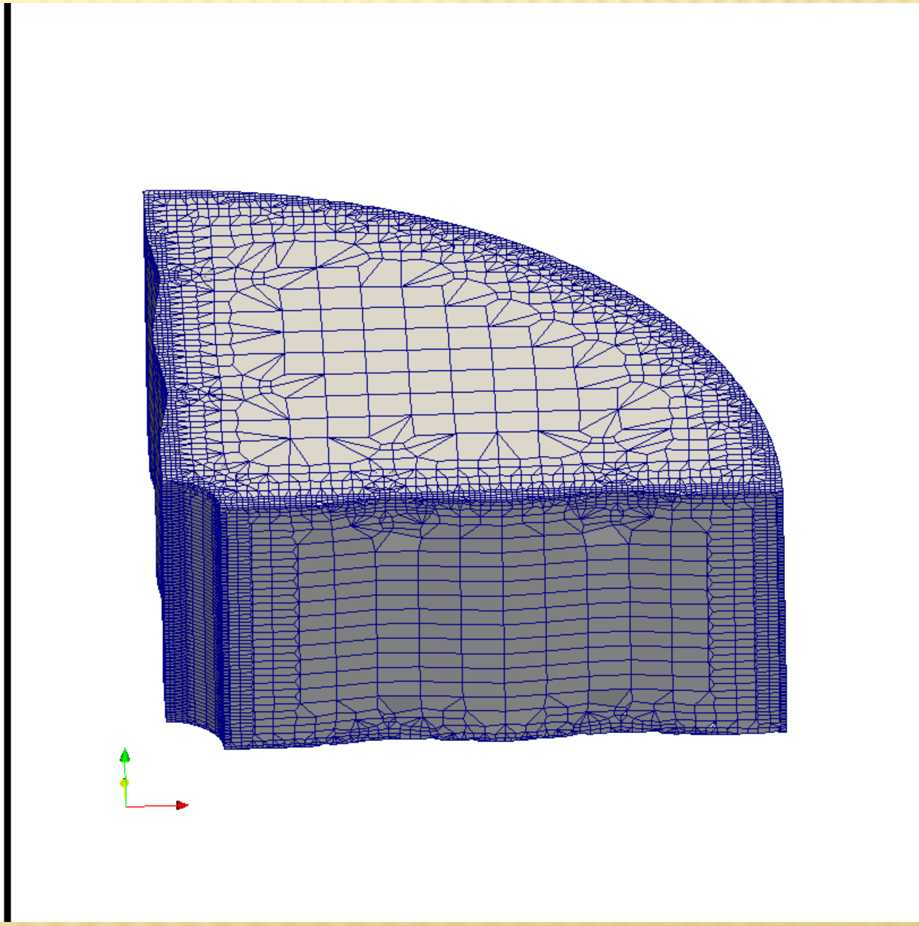
SurfaceFeatureExtract

- Need a dictionary in system/ called ExtractFeatureDict.
- In the test case, the executable is surfaceFeatureExtract.
- Create a *.emesh in constant/triSurface, and a new folder in constant/ called extendedFeatureEdgeMesh if the option writeObj is selected in surfaceExtractFeatureDict .
- In the snappyHexMeshDict dictionary, specify in features:

```
{
        file "pump_simplified.eMesh";
        level 3; // level of refinement
}
```

SurfaceFeatureExtract



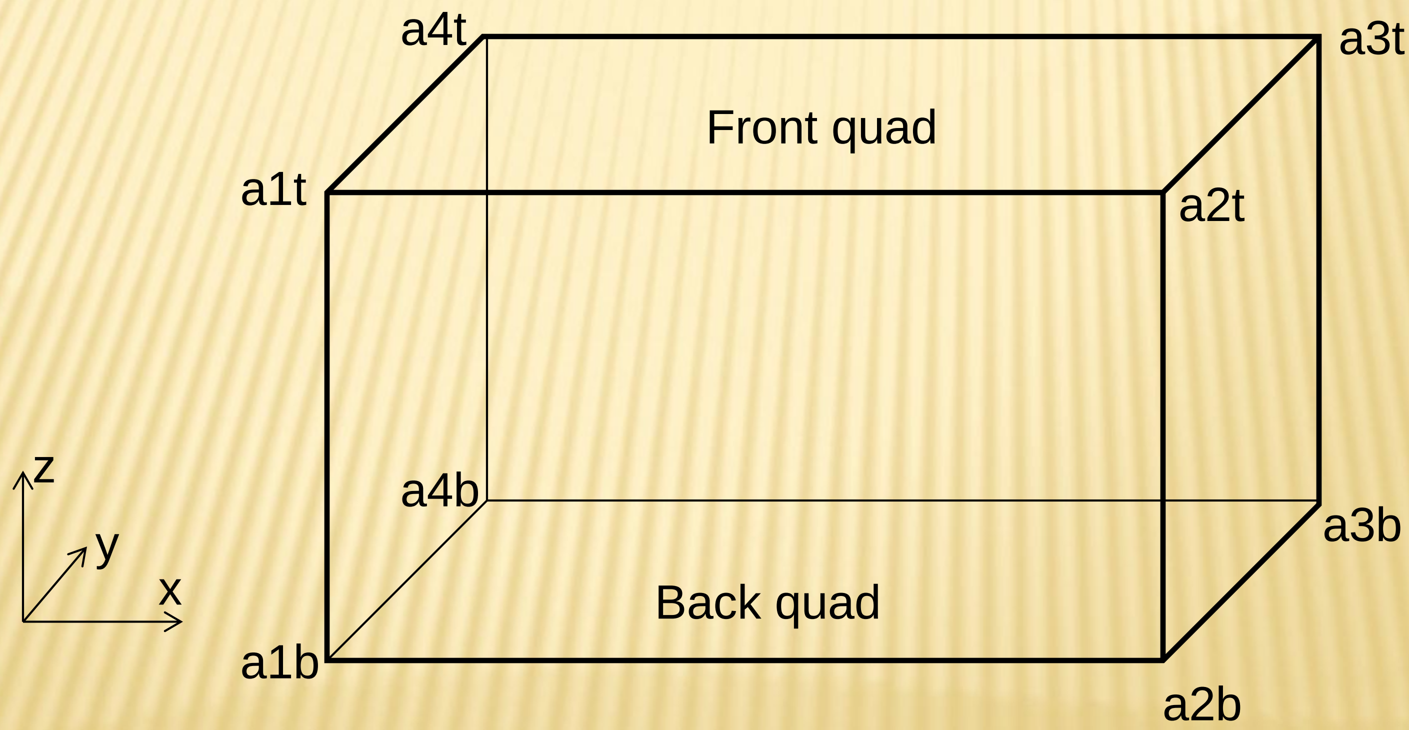


EXTERNAL LINKS FOR SNAPPYHEXMESH

- http://www.openfoam.org/docs/user/snappyHexMesh.php
- http://openfoamwiki.net/images/f/f0/Final-AndrewJacksonSlidesOFW7.pdf

blockMesh/m4

- m4: allow a parametrization of the case, easy to change.
- Let us take an example. Simple geometry: a pipe . The m4 file is found in m4_python/mesh/2D.m4.



blockMesh/m4: pros and cons.

- Very easy way to create an simple geometry.
- The parametrization with m4 allows to create different geometries from the same m4 file.
- Not enough precision in the meshing parameters.
- Easy to go wrong on the orientation of the faces and blocks.

Python

- **Python** is a general-purpose, high-level programming language. It emphasizes code readabilty.
- Like other dynamic languages, Python is often used as a scripting language.
- Commonly coupled with OpenFOAM, very useful to execute, analyse, manipulate parameters/simulations in OpenFOAM.
- Most commonly used python library in OpenFOAM is pyFoam.

Using Python to create a test case

- Main script: ./ChooseShape.py
- Needs 2 arguments. To know which one: **./ChooseShape.py –h**
- arg1 is (2D, 3D, symmetry), arg2 is (rectangle, cylinder)
- Call an other script in pythonScript folder: geometrySetup.py
- The script generates the chosen geometry, changes the m4 file, and do blockMesh

Using Python

- Many different useful applications
- pyFoam is a very useful compilation of library. http://openfoamwiki.net/index.php/Contrib_PyFoam
- Interesting tutorial by Eric Paterson, http://www.personal.psu.edu/egp11/Eric_Paterson/Blog/Entries/2009/2/2_Python_Scripting_for_Gluing_CFD_Applications__A_Case_Study_Demonstrating_Automation_of_Grid_Generation%2C_Parameter_Variation%2CFlow_Simulation%2C_Analysis%2C_and_Plotting.html. It can also be found at http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/
- post-processing example for turbomachinery can be found at http://openfoamwiki.net/index.php/Sig_Turbomachinery_/_Timisoara_Swirl_Generator#Post-processing_using_Python

Import the mesh: pros and cons

- Need of an other software to create the mesh.
- Few converters:
  - **fluentMeshToFoam**, **fluent3DMeshToFoam** for Gambit mesh types.
  - **starToFoam** for STAR-CD mesh types.
  - **ideasToFoam** for I-DEAS mesh types
  - **cfx4ToFoam** for CFX mesh types.
  - **CGNSToFoam** for CGNS files (can import more than meshes), developped by users.