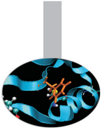
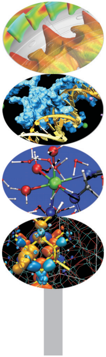


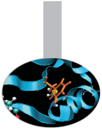
A tool for pre-processing: snappyHexMesh



Roberto Pieri - *SCS Italy*

16-18 June 2014

Overview



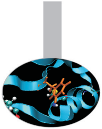
SnappyHexMeshDict

- Geometry
- CastellatedMesh
- Surface Snapping
- Layer addition

Mesh quality

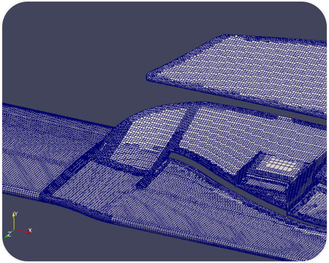
Building 2D mesh

Tutorial session

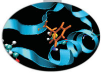


SnappyHexMesh

- Mesh generation utility of OpenFOAM.
- Automatic generation of 3D hex-dominant meshes.
- Preservation of geometry edges.
- Addition of layers for wall resolution.
- Parallel.

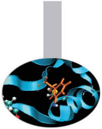


How does snappyHexMesh work?



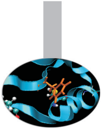
- Background mesh made of hexahedra generated by the utility *blockMesh*.
- CastellatedMesh phase:
    - Refinement in prescribed regions by the user.
    - Detection of the fluid domain.
    - Removal of cells outside the domain.
- SnapMesh phase:
    - Mesh morphing to follow the provided geometry.
- Possibly, layers addition phase.

Dictionary definition



- Dictionary file in *system/snappyHexMeshDict*.
- Divided in five sections:
  - *geometry:* input geometry;
  - *castellatedMeshControls:* refinement regions and the fluid domain;
  - *snapControls:* parameters related to morphing phase;
  - *addLayersControls:* settings for the layer addition (number of layers, grow rate, ...);
  - *MeshQualityControls:* where the user defines the quality required for the final mesh.

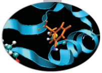
PAY ATTENTION IN REQUIRING QUALITY CONSTRAINTS



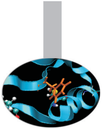
Geometry submission

- Geometry must be provided in Stereolithography (*.stl*) or Nastran (*.nas*) format.
- Working with a good quality CAD is mandatory (snappyHexMesh is not able to modify CAD)
- It has to be provided in the *constant/triSurface* directory.
- Other geometries (cylinder, box, sphere...) can be easily defined.

Geometry checking



- Before starting meshing it is important to check integrity of your CAD.
- surfaceCheck name_CAD.stl utility can check the geometry submitted.
- Main issues related to CAD:
  - non-closed CAD (snappyHexMesh will mesh inside the surface);
  - overlapping triangles.
- With the same utility you can receive informations related to surface bounding-box.



snappyHexMeshDict

geometry

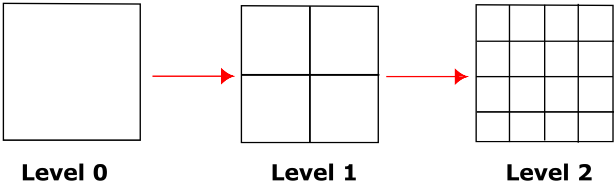
```
geometry
{
    NLR-7301.stl
    {
        type triSurfaceMesh;
        name airfoil;

        patchInfo
        {
            type wall;
        }
    }
};
```

- Name of the surface.
- Type definition.
- Definition of the name of the derived patch.
- Definition of the type of the derived patch.

snappyHexMeshDict

refinement

The first step is the refinement of cells in prescribed regions in castellatedMeshControls sub-dictionary.

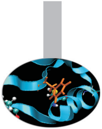


snappyHexMeshDict

castellatedMesh (I)

```
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    maxLocalCells 1000000;

    maxGlobalCells 10000000;

    minRefinementCells 0;

    maxLoadUnbalance 0.10;

    nCellsBetweenLevels 6;
```

- Maximum number of cells.
- Minimum number of cells for the surface refinement loop to stop.
- Number of cells between two adjacent refinement regions.



snappyHexMeshDict

castellatedMesh (II)

```
// Explicit feature edge refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

features
(
 {
  file "NLR-7301.eMesh";
  level 6;
 }
);
```

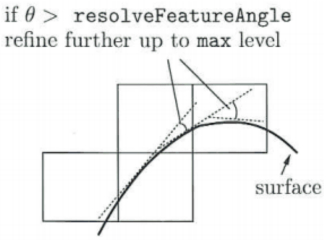
- Refinement in proximity of edges.
- Meshing with snappy may generate difficulties in reaching good resolution on edges.

The generation of the *.eMesh* file is obtained using the command surfaceFeatureExtract.

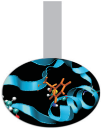
snappyHexMeshDict

castellatedMesh (III)

```
// Surface based refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~

refinementSurfaces
{
    airfoil
    {
        level (6 6);
    }
}

resolveFeatureAngle 50;
```


if $\theta >$ resolveFeatureAngle
refine further up to max level
surface

Feature angle refinement



snappyHexMeshDict

castellatedMesh (IV)

```
// Region—wise refinement
// ~~~~~~~~~~~~~~~~~~~~~~

refinementRegions
{
    airfoil
    {
        mode distance;
        levels  ((0.5  5)(0.8  4));
    }
}
```

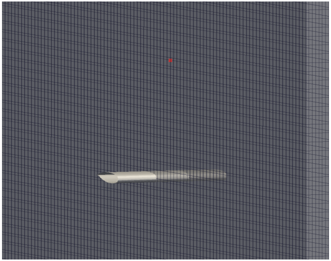
- The keyword *levels* specifies per distance to the surface the wanted refinement level.
- Other ways for refinement region:
  - inside
  - outside

snappyHexMeshDict

castellatedMesh (V)

```
    // Mesh selection
    // ~~~~~~~~~~~~~~

    locationInMesh (0.1 2 0.0);
```

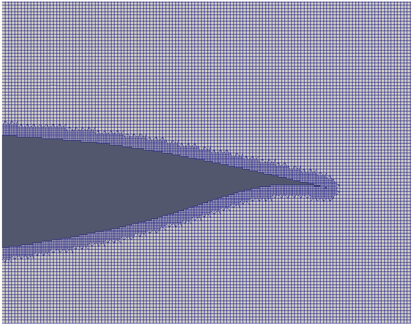
- Definition of a point inside the fluid domain.



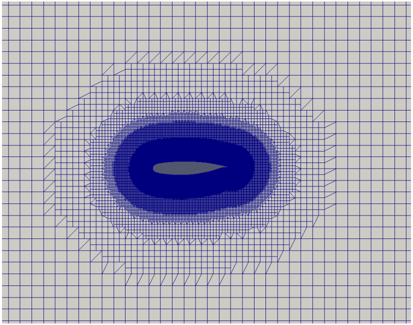
Definition of fluid domain

snappyHexMeshDict

castellatedMesh (VI)



Castellated phase

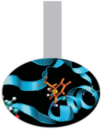


Levels in castellated phase

snappyHexMeshDict

castellatedMesh (VII)

```
Mesh stats
    points:           2027850
    faces:            5760631
    internal faces:   5529545
    cells:            1867241
    faces per cell:   6.04645
    boundary patches: 7
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     1838095
    prisms:        0
    wedges:        0
    pyramids:      0
    tet wedges:    0
    tetrahedra:    0
    polyhedra:     29146
```

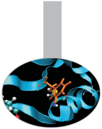


snappyHexMeshDict

Surface Snapping (I)

```
// Settings for the snapping.
snapControls
{

    nSmoothPatch 3;

    tolerance 4.0;

    nSolveIter 50;

    nRelaxIter 5;

}
```

- Number of patch smoothing before projecting on the surface.
- Scale factor of edge length for points to be attracted by surface.
- Smoothing iterations for mesh displacement relaxation.
- Maximum number of snapping iterations.



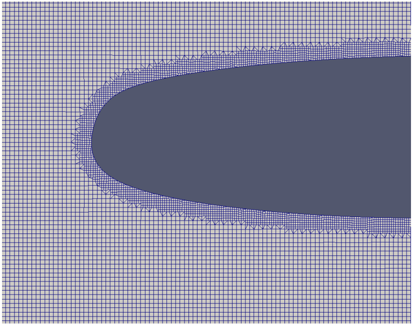
snappyHexMeshDict

Surface Snapping (III)

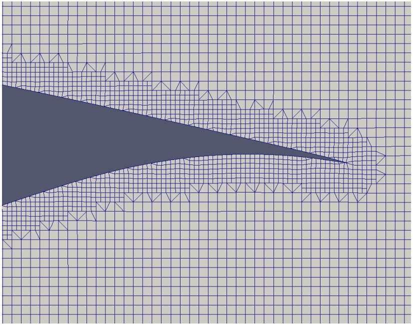
```
Mesh stats
    points:            2019140
    faces:             5752055
    internal faces:    5529545
    cells:             1867241
    faces per cell:    6.04186
    boundary patches: 7
    point zones:       0
    face zones:        0
    cell zones:        0

Overall number of cells of each type:
    hexahedra:      1829519
    prisms:         8576
    wedges:         0
    pyramids:       0
    tet wedges:     0
    tetrahedra:     0
    polyhedra:      29146
```

snappyHexMeshDict

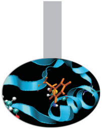
Surface Snapping (II)



Leading edge snap



Trailing edge snap



snappyHexMeshDict

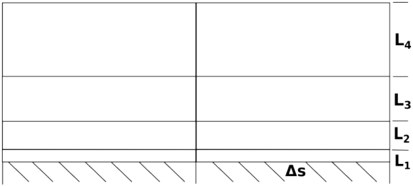
Layer Addition (I)

```
// Settings for the layer addition.
addLayersControls
{

    relativeSizes true;

    layers
    {
        airfoil
        {
            nSurfaceLayers 10;
        }
    }

    expansionRatio 1.1;

    finalLayerThickness 0.5;

    minThickness 0.05;
```

- Number of layers on selected patches.
- Expansion ratio of layers.
- Minimum layer thickness below which leyers are not added.

snappyHexMeshDict

Layer Addition (II)

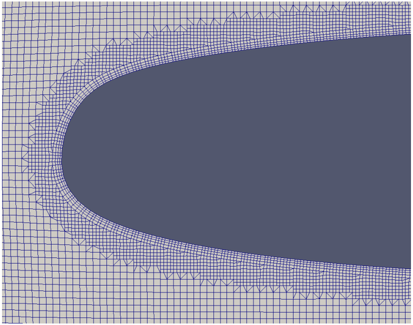


$$\text{expansionRatio} = \frac{L_2}{L_1} = \frac{L_3}{L_2} = \ldots$$

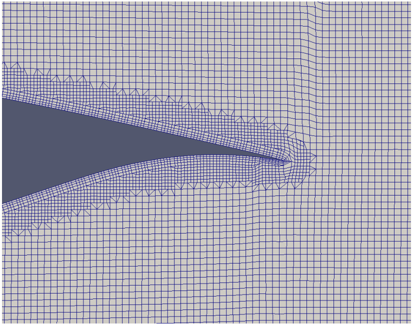
$$\text{finalLayerThickness} = \frac{L_4}{\Delta s}$$

snappyHexMeshDict

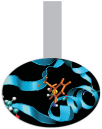
Layer Addition (III)



Layers on leading edge



Layers on trailing edge



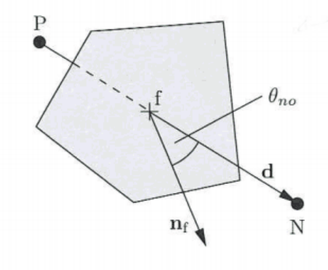
snappyHexMeshDict

Layer Addition (IV)

```
Mesh stats
    points:           2246380
    faces:            6426783
    internal faces:   6197281
    cells:            2090985
    faces per cell:   6.03738
    boundary patches: 7
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     2053263
    prisms:        8576
    wedges:        0
    pyramids:      0
    tet wedges:    0
    tetrahedra:    0
    polyhedra:     29146
```

Final mesh quality

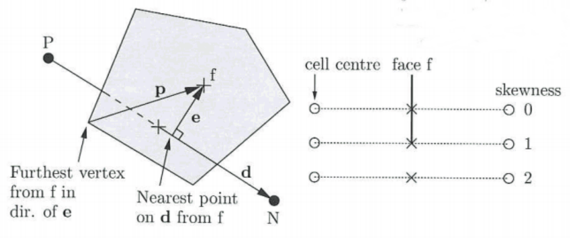
Orthogonality in OpenFOAM



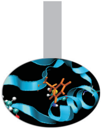
Non orthogonal faces

Final mesh quality

Skewness in OpenFOAM



Skew faces



Final mesh quality

Mesh checking in OpenFOAM (I)

```
   Mesh stats
    points:           2246380
    faces:            6426783
    internal faces:   6197281
    cells:            2090985
    faces per cell:   6.03738
    boundary patches: 7
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     2053263
    prisms:        8576
    wedges:        0
    pyramids:      0
    tet wedges:    0
    tetrahedra:    0
    polyhedra:     29146
```

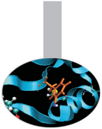
- checkMesh is an *OpenFOAM* utility to check the mesh quality.
- Number of cells.
- Number of patches.
- Cells divided by type.

Final mesh quality

Mesh checking in OpenFOAM (II)

```
Checking geometry...
    Overall domain bounding box (-6.8 -9.6 -0.075) (20.8 9.6 0.075)
    Mesh (non-empty, non-wedge) directions (1 1 0)
    Mesh (non-empty) directions (1 1 0)
 ***Number of edges not aligned with or perpendicular to non-empty directions: 1021342
  <<Writing 1419579 points on non-aligned edges to set nonAlignedEdges
    Boundary openness (3.1225e-19 -2.28772e-18 -3.18147e-14) OK.
    Max cell openness = 3.15737e-16 OK.
    Max aspect ratio = 5.58547 OK.
    Minimum face area = 6.20679e-07. Maximum face area = 0.022825.  Face area magnitudes OK.
    Min volume = 1.46414e-09. Max volume = 0.00342376.  Total volume = 79.4698.  Cell volumes OK.
    Mesh non-orthogonality Max: 64.3537 average: 3.90383
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 2.38186 OK.
    Coupled point location match (average 0) OK.

Failed 1 mesh checks.

End
```

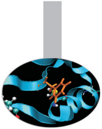
How can I build 2D meshes?



- *OpenFOAM* always works with 3D meshes (even in case you want to simulate a 2D phenomenon).
- In case you want to simulate a 2D case you have to build a mesh with only one cell in the uniform direction.
- Using extrudeMesh tool is possible to obtain 2D meshes (extrusion of 1 cell).
- It is necessary to define an *extrudeMeshDict* dictionary to tell *OpenFOAM* which patch has to be extruded.

extrudeMeshDict

```
constructFrom patch;
sourceCase "../tutorialNLR-7301_snappy";
sourcePatches (front);

// If construct from patch: patch to use for back (can be same as sourcePatch)
exposedPatchName back;

// Flip surface normals before usage. Valid only for extrude from surface or
// patch.
flipNormals false;

//- Linear extrusion in point-normal direction
extrudeModel        linearNormal;

nLayers             1;

expansionRatio      1.0;

linearNormalCoeffs
{
    thickness       0.05;
}

// Do front and back need to be merged? Usually only makes sense for 360
// degree wedges.
mergeFaces false;   //true;

// Merge small edges. Fraction of bounding box.
mergeTol 0;
```



snappyHexMesh tutorial

- Copy the CAD into the right directory.
- Extract edges from CAD using the appropriate tool.
- Open snappyHexMesh dictionary to set right refinement and layers options.
- Build background mesh.
- Run snappyHexMesh (use the flag -overwrite).
- Check the mesh quality
- Edit the extrudeMeshDict to extrude front patch.
- Extrude one patch from the previous mesh to build a 2D mesh.
- Check the mesh quality of the last mesh.
- Run the command renumberMesh -overwrite, discussion n this tool...