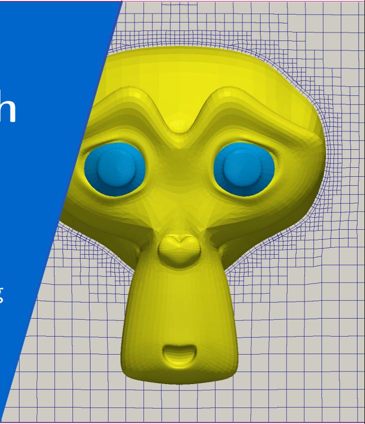
CE597D Special Topics, Fall, 2014

Chapter 4, Part 5: Mesh Generation

Xiaofeng Liu, Ph.D., P.E.
Assistant Professor
Department of Civil and Environmental Engineering
Pennsylvania State University
xliu@engr.psu.edu



Outline

Mesh Generation
- General Introduction
- Available meshing tools in OpenFOAM
- snappyHexMesh tutorial

Computational Mesh

- A computational mesh represents a description of spatial domain in the simulation: boundary and spatial discretization
- Mesh generation is not easy.
- Mesh generation usually takes more than half the effort in CFD simulations

Mesh Structure and Organisation

Mesh Types:

- Structured mesh
- Unstructured grid
- Overset meshes

From another angle, the domain or objects can be modelled using

- Explicit method: generating body-fitted mesh
- Implicit method: e.g., immersed boundary method on a fixed background mesh
- Mesh-less: e.g., LBM, SPH

Meshing Requirements

What defines the mesh requirement on size and quality?

- The most important factor is the physics you want to capture!
- Where to focus?
  - High resolution in area of interest
  - High resolution where properties change fast (high gradient)
- Dimensionality: 2D or 3D?
- For turbulent flow, how do you want to model it and how much you can afford?
  - DNS, LES or RANS
  - How to deal with regions with sharp gradient such as wall?

Mesh Generation

Typical process (steps) of mesh generation

- Geometry preparation
  - CAD
  - Scanning and digitization
- Geometry preparation
  - Refinement
  - Coarsening: too much unnecessary details
- Import the surface mesh into mesh generation software
  - The surface mesh usually defines part of the boundary you want to model
  - Combine the surface mesh(s) with other boundaries
  - Partitioning the surface into different patches, e.g., inlet, outlet, etc.
  - If the surface is simple, it can be generated in most of the mesh generation software

Mesh Generation

Typical process (steps) of mesh generation

- Volume mesh generation
  - The domain will be fully filled with non overlapping cells
  - The location of the cells defines where you will get the discrete solution
  - So you need to have some knowledge of the solution field for planning your mesh
  - Quality of the mesh extremely important for a good solution
  - In many cases, spending more time to get a decent mesh pays off later on.
  - Usual mesh quality measures
    - Cell aspect ratio
    - Non-orthogonality
    - Skewness
    - . . . *etc.*

Static vs. Dynamic Mesh

- Static: nothing changes, no deformation, no topological changes
- Dynamic: mesh deformation or topological change. Could be due to:
  - Moving parts in the domain
  - Adaptive mesh refinement/coarsening
- Common types of topological changes:
  - Attach/detach boundary
  - Cell layer addition/removal
  - Sliding interface

Adaptive Mesh Refinement/Coarsening

- To better capture of the area of interest (might change)
  1. Prepare an initial mesh
  2. March the solution forward
  3. Adjust mesh resolution (refinement or coarsening) based on the solution
  4. Repeat until desired accuracy is achieved or simulation is finished
- Adaptive mesh refinement/coarsening should be automatics based on some criterions:
  - Gradient
  - Error estimation
  - ...

Available meshing tools in OpenFOAM

- Source code in `applications/utilities/mesh`
- Mesh generation (in `applications/utilities/mesh/generation`):
  - `blockMesh`
  - `snappyHexMesh`
- Mesh conversion (in `applications/utilities/mesh/conversion`):
  - `fluentMeshToFoam`
  - `starToFoam`
  - `gambitToFoam`
  - `ideasToFoam`
  - `cfx4ToFoam`
  - ...
- Mesh tools in `applications/utilities/mesh/advanced`:
  - refineWallLayer: refine cells next to patches
  - collapseEdges: collapse short edges and combine edges that are in line
  - ...

Available meshing tools in OpenFOAM

- Mesh manipulation tools in `applications/utilities/mesh/manipulation`:
  - checkMesh: check the mesh quality
  - topoSet: operations (create, delete, invert, subset, etc.) on cellSets/faceSets/pointSets
  - refineMesh: refine the mesh, usually working with cellSet from `topoSet`
  - transformPoints: transforms the mesh points in the polyMesh directory according to the translate, rotate and scale options.
  - moveMesh: mesh deformation according to the specified motion on the boundary
  - renumberMesh: reduce the bandwidth of matrix A, as a result, reduce memory usage and improve convergence. Example in `incompressible/pisoFoam/les/motorBike/motorBike`
  - createPatch: create patches out of selected boundary faces.
  - ...

Available meshing tools in OpenFOAM

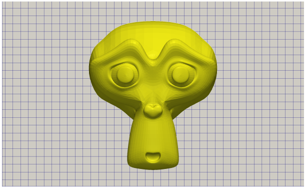
- OpenFOAM® also comes with a lot of tools to operate on surfaces (STL, OBJ, etc.)
- Source code in

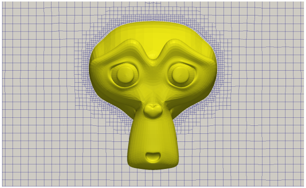
  `applications/utilities/surface`

  - surfaceCheck: checks surface for incorrect topology. Checks normals of neighbouring faces.
  - surfaceConvert: Converts surfaces to/from various formats
  - surfaceTransformPoints: Transform (scale/rotate) a surface
  - surfaceSmooth: Laplacian smoothing on surface vertices
  - surfaceCoarsen: Surface coarsening
  - ...

snappyHexMesh tutorial: Introduction

- A mesh generation utility comes with OpenFOAM®
- It *automatically* generates 3D meshes from triangulated surface geometries in Stereolithography (STL) format
- The mesh approximately conforms to the surface by iteratively refining a starting mesh and morphing the resulting split-hex mesh to the surface.
- Optional layer addition near the surface. Good for boundary layer flows.
- It can preserve feature edges.
- It runs in parallel. So the generation of large meshes are possible and fast.





snappyHexMesh : Basic usage

- Basic usage:

```
snappyHexMesh [OPTIONS]
options:
  -case <dir>          specify alternate case directory, default is the cwd
  -checkGeometry       check all surface geometry for quality
  -noFunctionObjects
                       do not execute functionObjects
  -overwrite           overwrite existing mesh/results files
  -parallel            run in parallel
  -roots <(dir1 .. dirN)>
                       slave root directories for distributed running
  -writeLevel          write pointLevel and cellLevel postprocessing files
  -srcDoc              display source code in browser
  -doc                 display application documentation in browser
  -help                print the usage
```

- Example: *snappyHexMesh* -overwrite. This command will overwrite the background mesh in the "*constant/polyMesh*" directory.

snappyHexMesh : Basic usage

Relevant input and control files:

- control file (more details later): "*system/snappyHexMeshDict*"
- A background mesh (could be from *blockMesh* or other means)
- Surface to snap to
  - A surface in STL format inside "*constant/triSurface*"
  - One of the generic surface types, e.g., sphere, cylinder, box, plane, etc.
  - Combination of above
- Other proper setup for an OpenFOAM® case in its *constant* and *system* directories

snappyHexMesh : Basic usage

Special instructions for parallel:

- *snappyHexMesh* can be run in parallel, which is needed for large meshes resulted from too much refinement or already refined background mesh or refined surface mesh.
- Parallel run is similar to other OpenFOAM® parallel runs. Example of using 8 processors:

```
mpirun -np 8 snappyHexMesh -parallel
```

- The background mesh case should already been decomposed using *decomposePar*
- Parallel run of *snappyHexMesh* will also read the *system/decomposeParDict* file.

snappyHexMesh : Processes

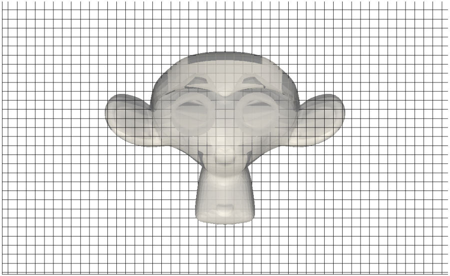
There are some basic steps when run *snappyHexMesh* :

- prepare a background mesh
- definition of surfaces and regions (for later use)
- run *snappyHexMesh* :
  - castellatedMesh
    - local refinement around the surface or designated regions
    - removal of cells not in the simulation domain
  - snap: snap the nearby points to the surface
  - addLayers: add more layers near the surface

Processes: Background mesh

Prepare a background mesh:

- `blockMesh` or other third-party mesh generator
- Background mesh requirements:
  - MUST be purely hex
  - Cell aspect ratio should be approximately 1
  - At least one intersection of cell edge with the STL surface (to start the cutting process)



- The background mesh serves as the Level 0 mesh
- The surface file in STL located in `constant/triSurface`.
- The surface file can contain more than one separated objects.

Definition of surface and regions

Surfaces and regions are defined for later use:

- in sub-dictionary `geometry`
- These surfaces and regions will later be used:
  - to snap the mesh to the surface
  - or for local refinement when the cell intersecting it
  - or for local refinement when cell is inside/outside/near it
- Example:

```
geometry
{
    box1
    {
        type searchableBox;
        min (1.5 1 -0.5);
        max (3.5 2 0.5);
    }
```

```
  monkey.stl
   {
       type triSurfaceMesh;
       name monkey;
   }

   sphere2
   {
       type searchableSphere;
       centre  (1.5 1.5 1.5);
       radius  1.03;
   }
};
```

PENNSTATE

Definition of surface and regions

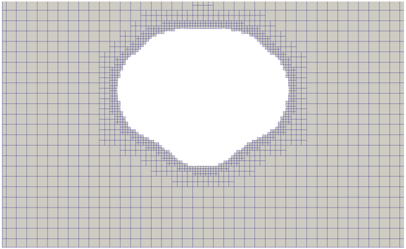
Surfaces and regions are defined for later use:

- in sub-dictionary `geometry`
- These surfaces and regions will later be used:
  - to snap the mesh to the surface
  - or for local refinement when the cell intersecting it
  - or for local refinement when cell is inside/outside/near it
- Example:
- All the entries inside `geometry` should be an object inherited from class `searchableSurface`
  1. searchableBox: box
  2. searchableCylinder: cylinder
  3. searchablePlane: plane
  4. searchablePlate: plate
  5. searchableSphere: sphere
  6. searchableSurfaceCollection:
  7. searchableSurfaceWithGaps:
  8. triSurfaceMesh: of course a STL surface

Processes

Mesh castellation:

- enabled by setting `castellatedMesh true`
- controlled by sub-dictionary `castellatedMeshControls`
  - local refinement before cell removal
  - Removal of cells not needed; depends how inside and outside are defined.

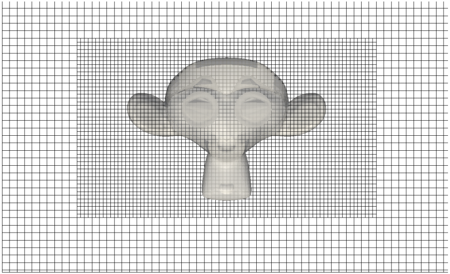


- `locationInMesh` defines the region which will be kept (not removed)
- If approximately 50% of a cell's volume in the kept region, it will be kept.
- Different areas/regions can have different level of cell refinement.
- The changes in the refinement level is continuous and gradual with a specified transition layer (`nCellsBetweenLevels`)

Processes

Local refinement with `refinementSurfaces` and `refinementRegions`:

- Refine around a surface
- Refine inside a volume

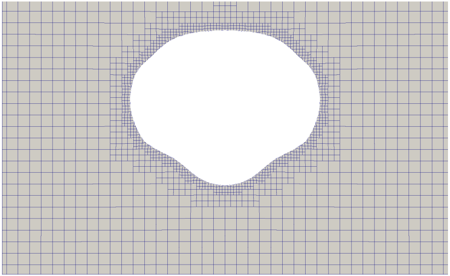


- Cells within one or more specified regions will be further refined
- Dictionary entries define `inside/outside` and refinement levels

Processes

Snap:

- enabled by setting `snap true`
- controlled by sub-dictionary `snapControls`
- Vertex points nearby will be moved onto the surface, i.e. snap
- Also move the internal points to smooth out the distribution of vertex (through mesh motion solvers)
- Any mesh quality violations due to the point motion will be rectified through iterations

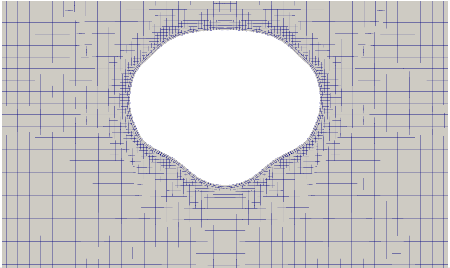


- Can preserve geometrical feature on the surface
- Other parameters controls the iterations and tolerance

Processes

Layer addition:

- enabled by setting `addLayer true`
- controlled by sub-dictionary `addLayersControls`
- Good for boundary layer simulations
- Also try to eliminate the irregular cells along the boundary surface from the snapping phase
- Mesh is pushed back from the surface (shrinking of the existing mesh) and layers of cells are inserted
- Mesh quality is monitored and any violation will be rectified through iterations



- Control file defines how many layers, layer size and growth rate
- Quality check and iteration numbers

General mesh quality

snappyHexMesh monitors the quality of the mesh during the process:

- Any violation (sub-quality or invalidicity) due to mesh motion or topological change will be recorded.
- Remedial action is mainly through "undo" to revert to the previous error free mesh status
- Mesh quality metrics are defined in the sub-dictionary `meshQualityControls`
- Most of the mesh quality calculations are done in the help class `primitiveMeshTools`
- Definition in `$FOAM_SRC/OpenFOAM/meshes/primitiveMesh/primitiveMeshCheck`

Mesh quality

Example of meshQualityControls (part)

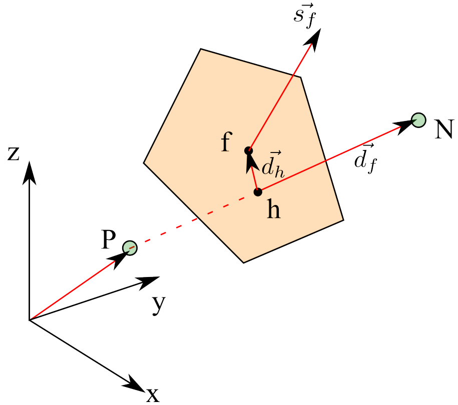
```
meshQualityControls
{
    // Maximum non-orthogonality allowed. Set to 180 to disable.
    maxNonOrtho 65;

    // Max skewness allowed. Set to <0 to disable.
    maxBoundarySkewness 20;
    maxInternalSkewness 4;

    // Max concaveness allowed. Is angle (in degrees) below which concavity
    // is allowed. 0 is straight face, <0 would be convex face.
    // Set to 180 to disable.
    maxConcave 80;

    // Minimum pyramid volume. Is absolute volume of cell pyramid.
    // Set to a sensible fraction of the smallest cell volume expected.
    // Set to very negative number (e.g. -1E30) to disable.
    minVol 1e-13;

    // Minimum quality of the tet formed by the face-centre
    // and variable base point minimum decomposition triangles and
    // the cell centre. This has to be a positive number for tracking
    // to work. Set to very negative number (e.g. -1E30) to
```

PENNSTATE

Mesh quality

Item `maxNonOrtho`:

- maximum angle between the line $\vec{d_f}$ and $\vec{s_f}$
- $\vec{s_f}$: face normal vector (pointing outwards relative to the current cell P)
- $\vec{d_f}$: vector connecting neighboring cell centers P and N



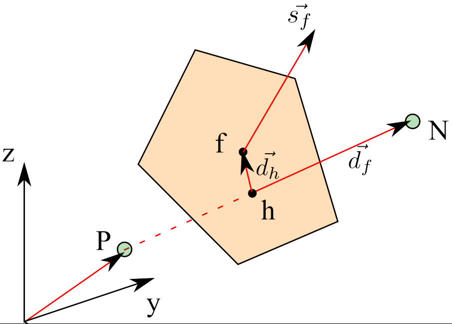
- Non-orthogonal angle

$$\cos(\theta) = \frac{\vec{s_f} \cdot \vec{d_f}}{|\vec{s_f}||\vec{d_f}|}$$

PENNSTATE

Items `maxBoundarySkewness` and `maxInternalSkewness`:

- Both skewness are referring to faces (internal or boundary) and normalized
- Normalisation distance calculated as the approximate distance from the face centre to the edge of the face in the direction of the skewness
- Skewness reduces the order of face integration but without stability implications. Meshes with highly skewed cells work better with special gradient calculation schemes: use least square gradient (with limiter).
- $\vec{d_h}$: non-normalized skewness



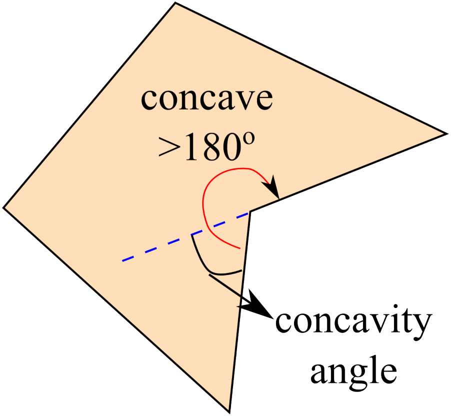
- normalized skewness

$$= \frac{|\vec{d_h}|}{normalization}$$

Mesh quality

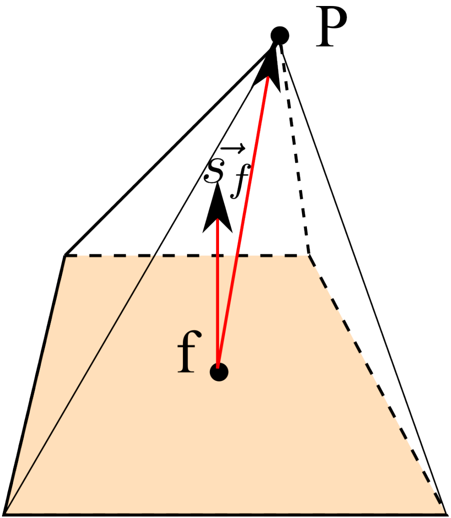
Item `maxConcave`:

- Maximum concave angle allowed between two consecutive edges of a face



Item `minVol`:

- Minimum pyramid volume.



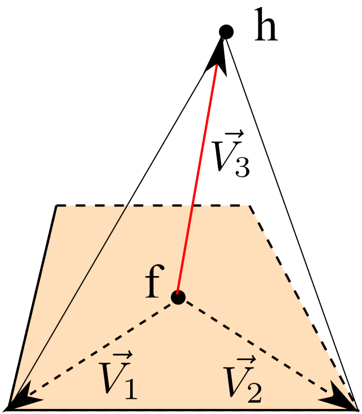
$$= \frac{1}{3}\vec{Pf} \cdot \vec{s_f}$$

Item minArea: minimum area of face.

Mesh quality

Item `minTetQuality`:

- Quality of a tetrahedron: Ratio of tetrahedron and circum-sphere volume. It is scaled so that a regular tetrahedron has a quality of 1 .
- Cells decomposed into tetrahedra by using the cell centre and face centre
- A mesh must have positive tet quality for tracking to work



$$\kappa_1 = |\vec{V}_3|^2 - (V_1 \cdot V_3)$$

$$\kappa_2 = |\vec{V}_2|^2 - (V_1 \cdot V_2)$$

$$r = \left| \frac{1}{2} \left(\vec{V}_1 + \frac{\kappa_1(\vec{V}_2 \times \vec{V}_1) - \kappa_2(\vec{V}_3 \times}{\vec{V}_3 \cdot (\vec{V}_2 \times \vec{V}_1)} \right.\right.$$

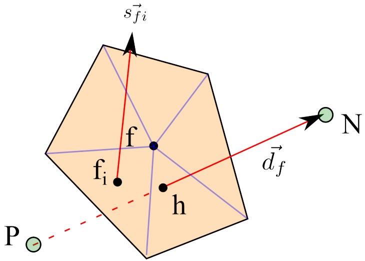
$$v = \frac{1}{6}\left[\left(\vec{V}_1 \times \vec{V}_2 \cdot \vec{V}_3\right)\right]$$

$$tetquality = \frac{v}{\frac{8}{9\times\sqrt{3}} r^3}$$

Mesh quality

Item `minTwist`:

- dot product of vector line *PN* and face centre triangles normal
- Definition in
  `$FOAM_SRC/dynamicMesh/motionSmoother/polyMeshGeometry`

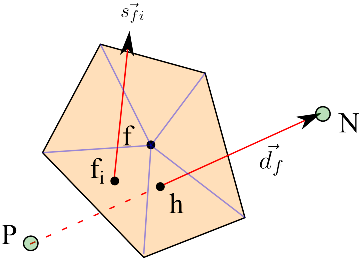


$$= \vec{d_f} \cdot \vec{s_{fi}}$$

Mesh quality

Item `minDeterminant`:

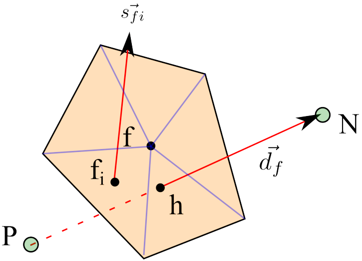
- Cell determinant is calculated by taking the determinant of the tensor calculated from the face area vectors.
- The calculation of the cell determinant is used during a fvc::reconstruct so must be well conditioned
- Definition in `$FOAM_SRC/OpenFOAM/meshes/primitiveMesh/primitiveMeshCheck`



Mesh quality

Item `minFaceWeight`:

- Interpolation weights (0.5 for regular mesh)
- Definition in
  `$FOAM_SRC/dynamicMesh/motionSmoother/polyMeshGeometry`



General mesh quality

Some other mesh quality metrics not directly specified in the control dictionary:

- Cell aspect ratio: the ratio of longest to shortest edge length. In many cases, it is desirable to align the cell with solution gradient.
- Defined in `$FOAM_SRC/OpenFOAM/meshes/primitiveMesh/primitiveMeshCheck`

```
scalarField openness;
scalarField aspectRatio;
primitiveMeshTools::cellClosedness
(
    *this,
    meshD,
    faceAreas,
    cellVolumes,
    openness,
    aspectRatio
);
```

```
void Foam::primitiveMeshTools::cellClosedness
(
    const primitiveMesh& mesh,
    const Vector<label>& meshD,
    const vectorField& areas,
    const scalarField& vols,

    scalarField& openness,
    scalarField& aratio
)
```

General mesh quality

Some other notes about mesh quality:

- From the point of view of numerical solutions, the (most) important mesh quality metrics are: orthogonality, volume ratio, delta ratio, skewness.
- Other metrics are of course also relevant:
- Face concavity:
  - Face concavity does not mean the cell is concave.
  - This is the edge angle between adjacent edges in a face, not between adjacent faces.
  - A concave polygon will always have an interior angle with a measure that is greater than 180 degrees
  - This property does not influence the discretisation directly.
  - Lowering the maxConcave value will "improve" the cell quality, but the mesh might not conform as well to the surface as what might otherwise be the case.
- Cell size grading: It affects the numerical accuracy if the cell size is not uniform. It is well known. You can actually proof this.

Summary for *snappyHexMesh*

Summary:

- `snappyHexMesh` is a good tool to generate body fitted meshes
- The drawback is that there are far too many parameters and input items
- Sometime is very slow and parallel computation can help