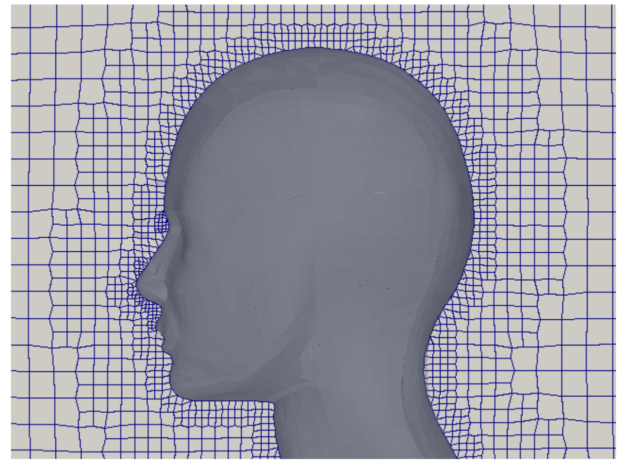
A Comprehensive Tour of snappyHexMesh

7th OpenFOAM Workshop

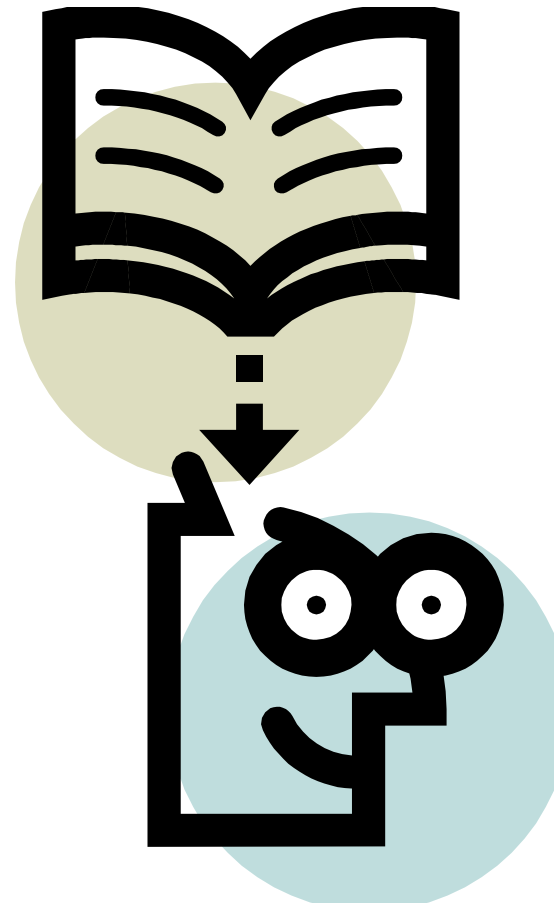
25 June 2012



info@engys.eu | Tel: +44 (0)20 32393041 | Fax: +44 (0)20 33573123 | www.engys.eu

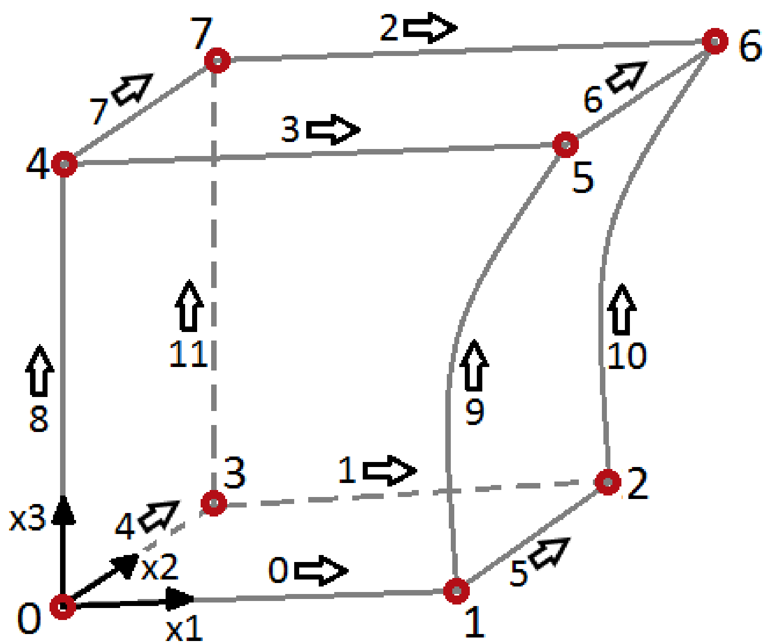
You Will Learn About...

- Mesh Creation
  - blockMesh
  - snappyHexMesh
- Mesh Checks
  - checkMesh



blockMesh | Definition

- Utility `blockMesh` is used to create simple block based fully-structured hexahedral meshes
- Controlled by dictionary *constant/polyMesh/blockMeshDict*
- Each mesh block consists of 8 vertices and 12 edges in a fixed order:



blockMesh | Usage

- Define *blockMeshDict* → Execute `blockMesh`
- Execution:

  blockMesh [-dict dictionary] [-case dir] [-blockTopology] [-region name] [-help]

  - No parallel execution
- Requirements:
  - Dictionary file *constant/polyMesh/blockMeshDict*
  - Dictionary *system/controlDict*

blockMeshDict | Overview

Dictionary file consists of five main sections:

- `vertices` → Prescribe vertex locations for blocks
- `blocks` → Prescribe block topology and mesh settings
- `edges` → Prescribe curved edges
- `patches` → Prescribe surface patches for boundary conditions
- `mergePatchPairs` → Merge disconnected meshed blocks

blockMeshDict | Headings

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant/polyMesh";
    object      blockMeshDict;
}
convertToMeters 1.0;

vertices
(
    (0.0 0.0 -0.038)
    (0.038 0.0 -0.038)
    (0.076 0.0 -0.038)
    (0.0 1.09 -0.038)
    (0.038 1.09 -0.038)
    (0.076 1.09 -0.038)
    (0.0 2.18 -0.038)
    (0.038 2.18 -0.038)
    (0.076 2.18 -0.038)
    (0.0 0.0 0.038)
    (0.038 0.0 0.038)
    (0.076 0.0 0.038)
    (0.0 1.09 0.038)
    (0.038 1.09 0.038)
    (0.076 1.09 0.038)
    (0.0 2.18 0.038)
    (0.038 2.18 0.038)
    (0.076 2.18 0.038)
);
```

→ File header

→ Keyword `convertToMeters`

- Scale factor
- Final mesh always in meters!

blockMeshDict | vertices

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant/polyMesh";
    object      blockMeshDict;
}
convertToMeters 1.0;

vertices
(
    (0.0 0.0 -0.038)
    (0.038 0.0 -0.038)
    (0.076 0.0 -0.038)
    (0.0 1.09 -0.038)
    (0.038 1.09 -0.038)
    (0.076 1.09 -0.038)
    (0.0 2.18 -0.038)
    (0.038 2.18 -0.038)
    (0.076 2.18 -0.038)
    (0.0 0.0 0.038)
    (0.038 0.0 0.038)
    (0.076 0.0 0.038)
    (0.0 1.09 0.038)
    (0.038 1.09 0.038)
    (0.076 1.09 0.038)
    (0.0 2.18 0.038)
    (0.038 2.18 0.038)
    (0.076 2.18 0.038)
);
```

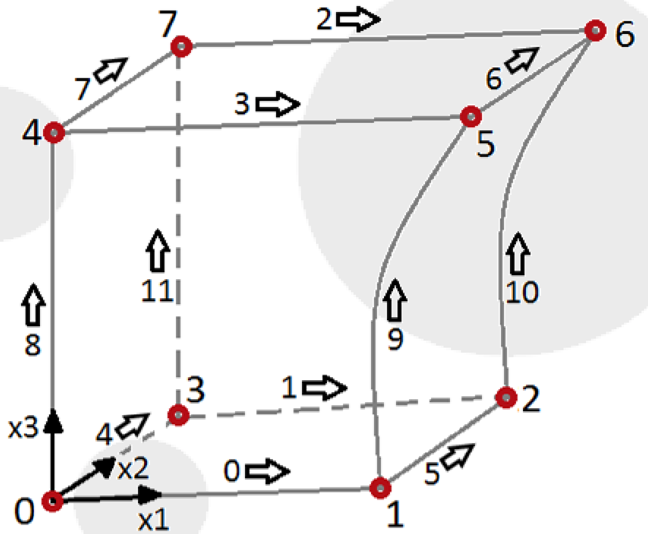
→ List of all vertices on each block

- Cartesian coordinates (x, y, z)
- Each vertex in list = 1 index

blockMeshDict | blocks

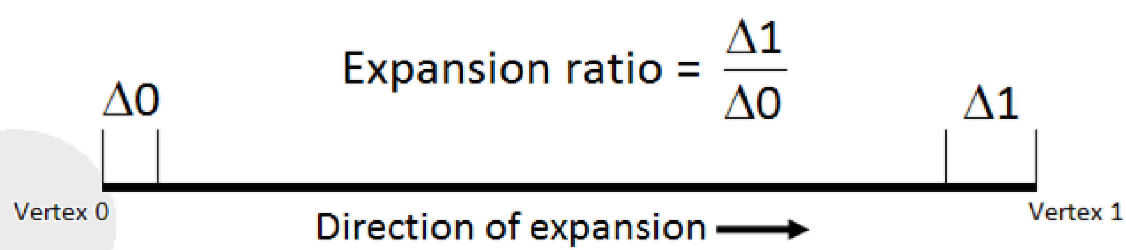
```
blocks
(
    hex (0 1 4 3 9 10 13 12) (20 400 1) simpleGrading (8 8 1)
    hex (1 2 5 4 10 11 14 13) (20 400 1) simpleGrading (0.125 8 1)
    hex (3 4 7 6 12 13 16 15) (20 400 1) simpleGrading (8 0.125 1)
    hex (4 5 8 7 13 14 17 16) (20 400 1) simpleGrading (0.125 0.125 1)
);
```

All vertices in order



Number of nodes per side (x, y, z)

Mesh grading



engys

blockMeshDict | patches

```
edges();
```

```
patches
(
    wall ffmaxy
    (
        (6 15 16 7)
        (7 16 17 8)
    )
    wall ffminy
    (
        (0 1 10 9)
        (1 2 11 10)
    )
    wall cold
    (
        (3 12 15 6)
        (0 9 12 3)
    )
    wall hot
    (
        (2 5 14 11)
        (5 8 17 14)
    )
    empty frontAndBack
    (
        (0 3 4 1)
        (1 4 5 2)
        (3 6 7 4)
        (4 7 8 5)
        (9 10 13 12)
        (10 11 14 13)
        (12 13 16 15)
        (13 14 17 16)
    )
);
```

```
mergePatchPairs();
```

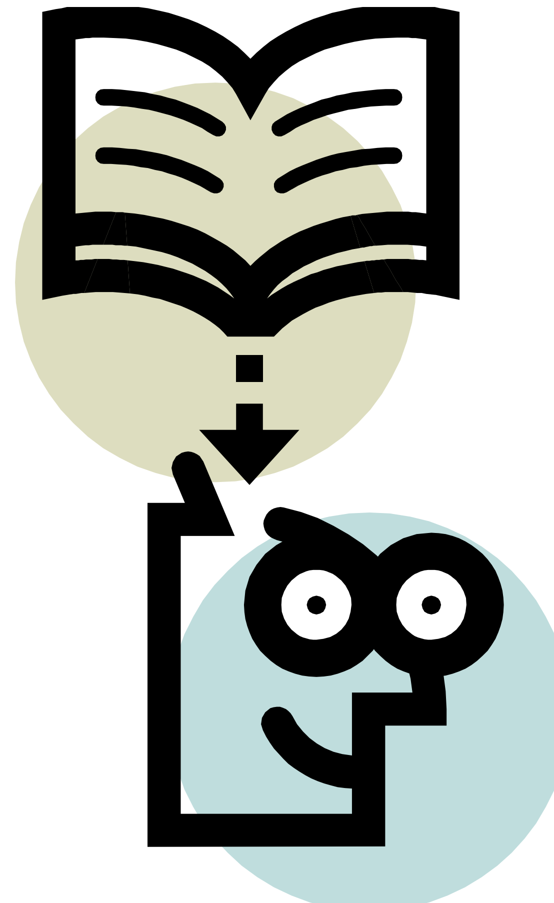
Define curved edges
- Edge is straight if not listed
- Arc and Spline available

Patches
- Boundary (base) type
- Patch name
- List of vertices (one per patch)
- Order follows right-hand rule

Merge multiple (separated) blocks into one mesh
- One-to-one correspondence if blocks are not listed

You Will Learn About...

- Mesh Creation
  - blockMesh
  - snappyHexMesh
- Mesh Checks



snappyHexMesh | Background

- Utility `snappyHexMesh` was developed by Mattijs Janssens, Eugene de Villiers and Andrew Jackson
- Engys continue to develop a version with enhanced features
  - Enhanced feature capturing and automation
  - Improved layers and layer specification methods
  - Layers growing up patches
  - Generation of Internal layers
  - Proximity based refinement
  - Multi layer addition
  - Automatic block mesh creation and decomposition
  - Mesh wrapping and small leak closure
  - and many other extensions

snappyHexMesh | Definition

- Utility `snappyHexMesh` is used to create high quality hex-dominant meshes based on arbitrary geometry
- Controlled by dictionary *system/snappyHexMeshDict*
- This utility has the following key features:
  - Fully parallel execution
  - STL and Nastran (.nas) files support for geometry data
  - Preservation of feature edges
  - Addition of wall layers
  - Zonal meshing for support of porous media and MRF
  - Quality guaranteed final mesh that will run in OPENFOAM®

snappyHexMesh | Usage

- Define *snappyHexMeshDict* → Execute `snappyHexMesh`
- Execution:

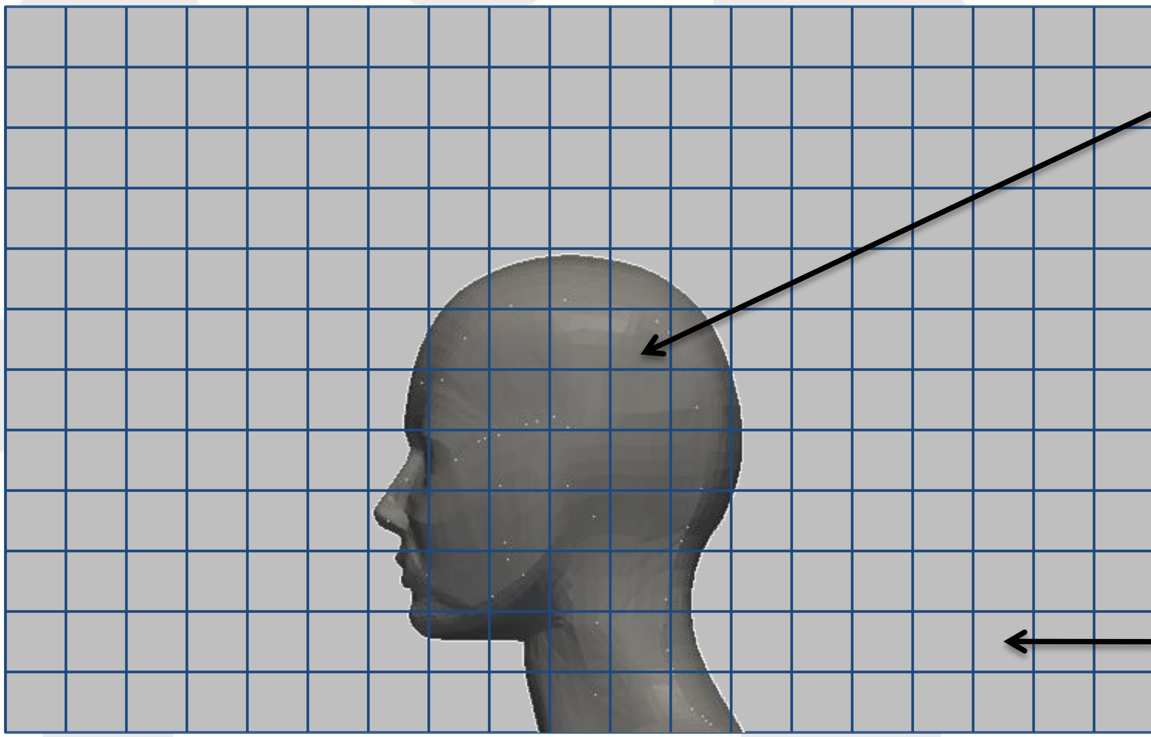
  snappyHexMesh [-noFunctionObjects][-overwrite] [-parallel] [-case dir] [-roots <(dir1 .. dirN)>] [-help]

  - Parallel execution available using **mpirun**
- Requirements:
  - Dictionary file *system/snappyHexMeshDict*
  - Geometry data (stl, nas, obj) in *constant/triSurface*
  - Hexahedral base mesh (decomposed if running in parallel)
  - Dictionary file *system/decomposeParDict* for parallel runs
  - All *system* dictionaries (e.g *controlDict, fvSchemes, fvSolutions*)

snappyHexMesh | Methodology

- Step 1: Create base mesh
  - Custom made → Using utility `blockMesh`

Note: Cells should be close to unit Aspect Ratio for optimum behaviour



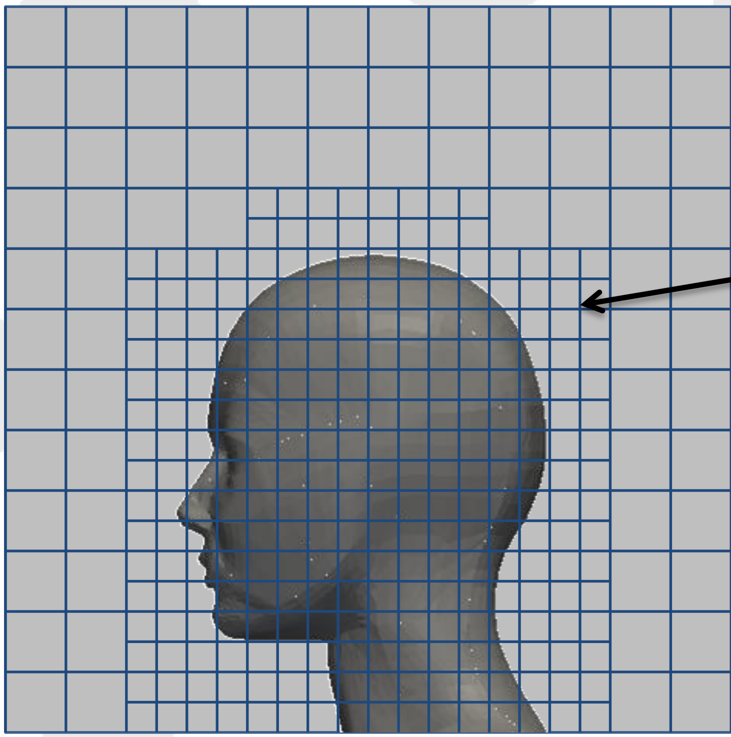
STL or Nastran (*constant/triSurface*)

NOTE: File names must not contain any spaces, unusual characters or begin with a number. The same applies to the names of the parts and regions defined within the geometry files.

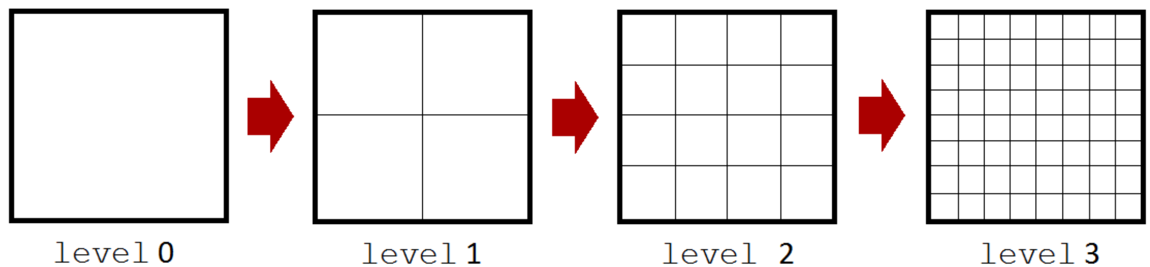
Hexahedral base mesh → level 0 size

snappyHexMesh | Methodology

- Step 2: Refine base mesh
  - Surface refinement → feature lines, proximity & curvature
  - Volume refinement → closed surfaces, geometric shapes

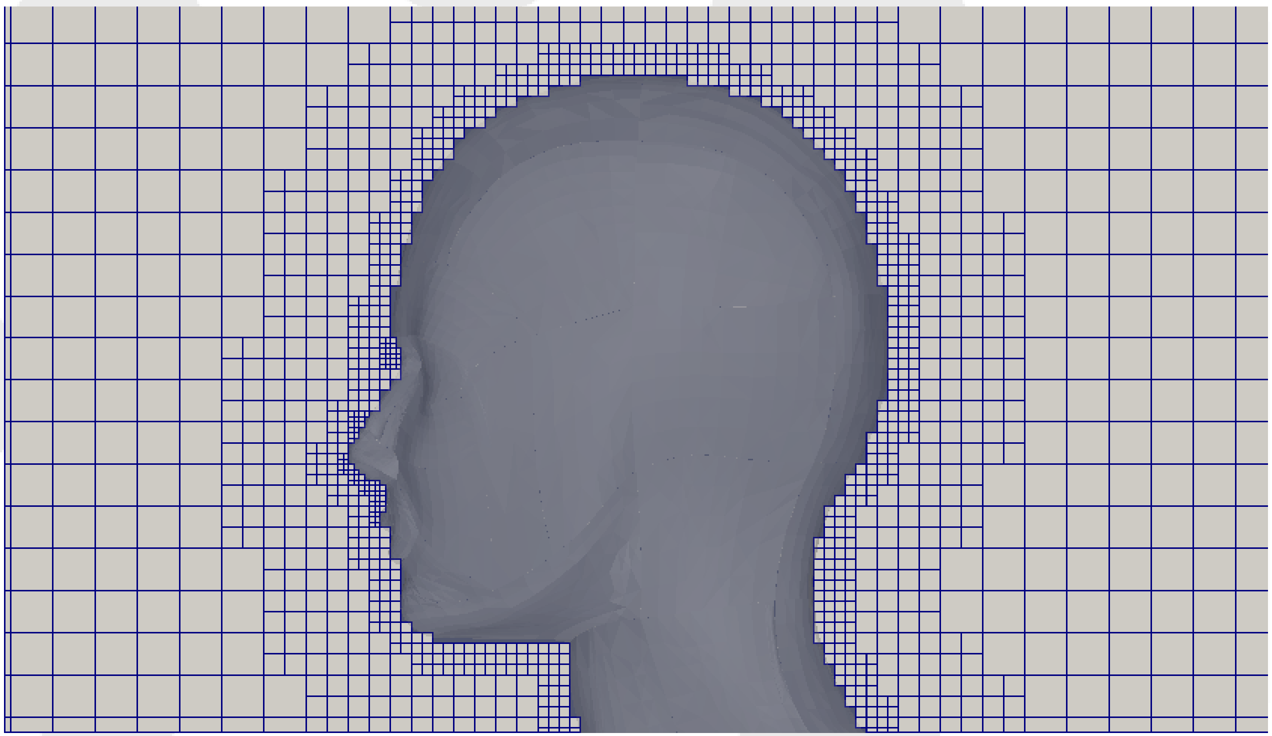


Local mesh refinements

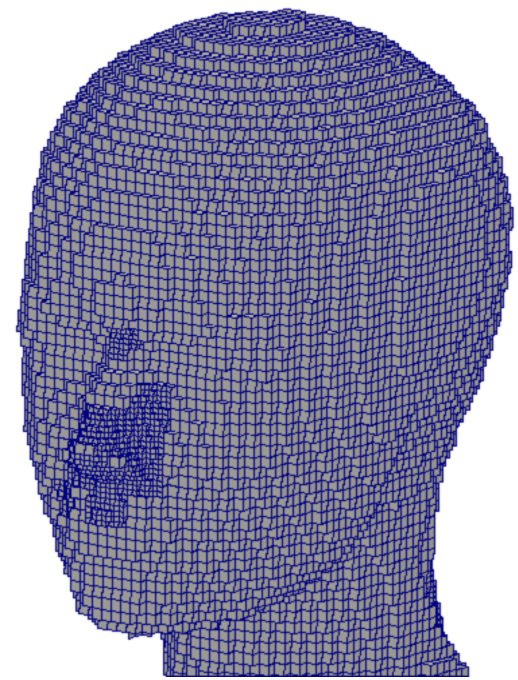


snappyHexMesh | Methodology

- Step 3: Remove unused cells
  - User defines keep point

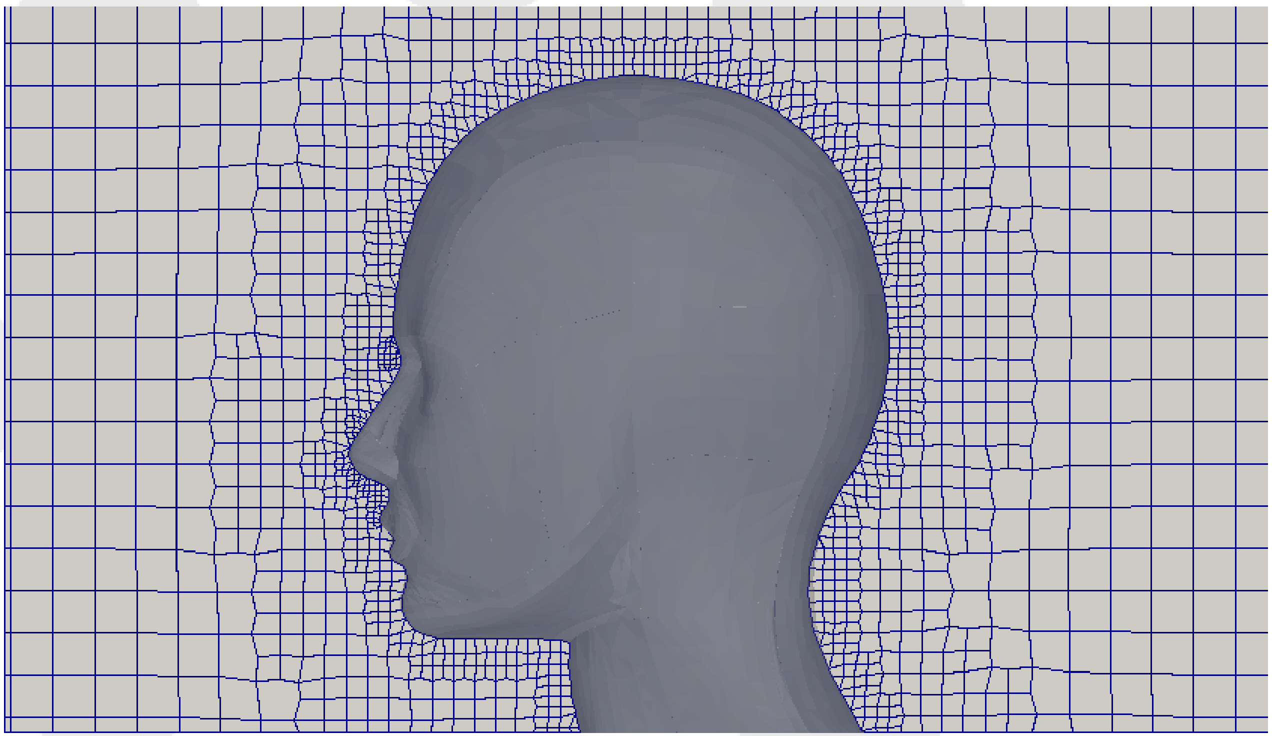


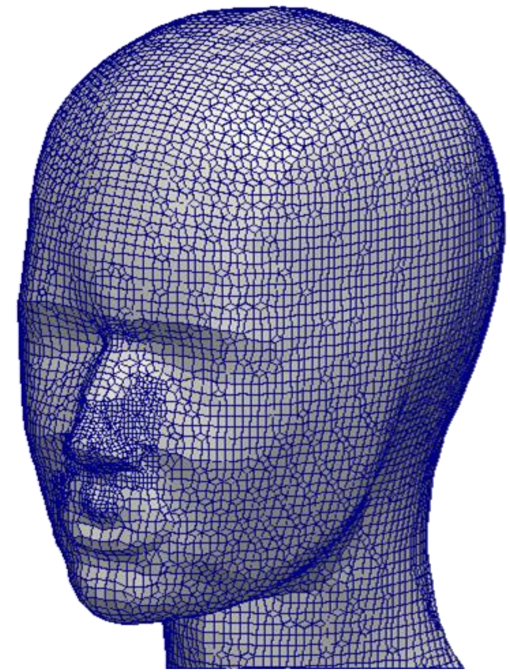
Castellated mesh



snappyHexMesh | Methodology

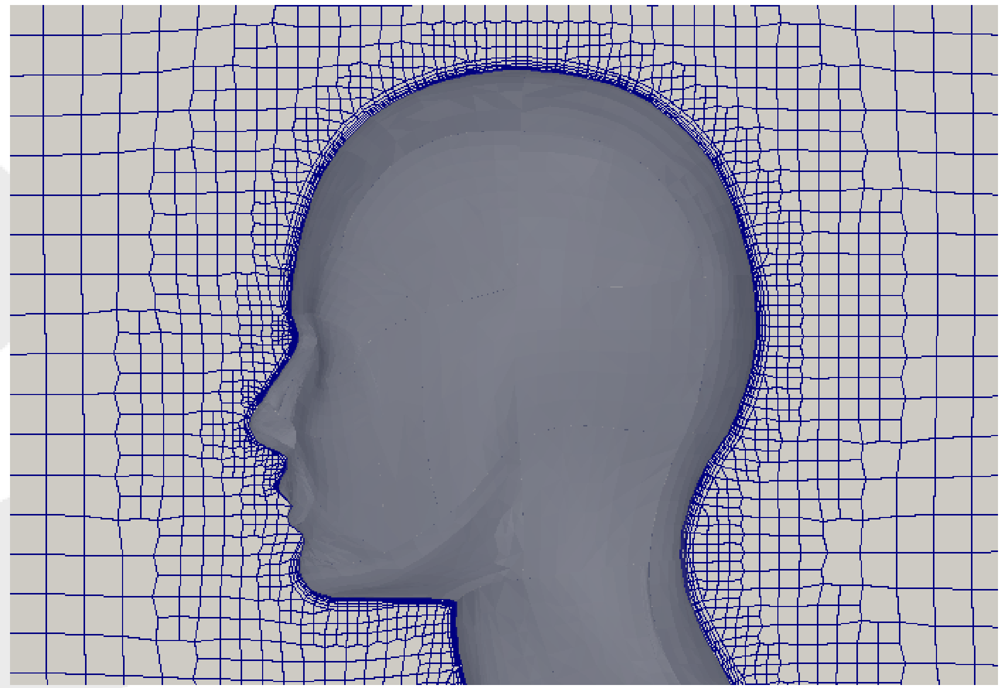
- Step 4: Snap mesh to surface
  - Implicit wrapping → Preserve features
  - Smooth & Merge faces





snappyHexMesh | Methodology

- Step 5: Add layers
  - Push mesh away from surface
  - Add layers
  - Check quality
  - Scale back displacement if errors
  - Repeat until all quality checks pass
- Step 6: Final load balance
  - Output to file



Note: All steps are automatic. Reproduced here for information purposes only.

snappyHexMeshDict | Overview

Dictionary file consists of five main sections:

- `geometry` → Prescribe geometry entities for meshing
- `castellatedMeshControls` → Prescribe feature, surface and volume mesh refinements
- `snapControls` → Control mesh surface snapping
- `addLayersControls` → Control boundary layer mesh growth
- `meshQualityControls` → Control mesh quality metrics

snappyHexMeshDict | Basic Controls

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   object     autoHexMeshDict;
}
castellatedMesh true;
snap            true;
addLayers       false;

geometry
{
   flange.stl
   {
      type triSurfaceMesh;
      name flange;
   }
   sphereA
   {
      type searchableSphere;
      centre (0 0 -0.012);
      radius 0.003;
   }
}
```

File header

Keywords

- Switch on/off mesh steps

snappyHexMeshDict | geometry

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   object     autoHexMeshDict;
}
castellatedMesh true;
snap            true;
addLayers       false;
```

```
geometry
{
   flange.stl
   {
      type triSurfaceMesh;
      name flange;
   }
   sphereA
   {
      type searchableSphere;
      centre (0 0 -0.012);
      radius 0.003;
   }
}
```

Definition of geometry types

- STL and Nastran files → serial or distributed
- Basic shapes → box, cylinder, sphere...

snappyHexMeshDict | Supported Types

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

```
geomB.stl
{
    type              distributedTriSurfaceMesh;
    distributionType follow;
    name              geomB;
}
```

Triangulated (e.g. Nastran, STL, OBJ)

- The standard type “**triSurfaceMesh**” reads a copy of each surface on to each processor when running in parallel.

- A distributed surface type exists “**distributedTriSurfaceMesh**” which can reduce the memory overhead for large surfaces
- Utility **surfaceRedistributePar** is used to initially decompose the surface
- Three distribution methods available
  independent: distribution independent of mesh to produce best memory balance
  follow: distribution based on mesh bounding box to reduce communication
  frozen: distribution remains unchanged

snappyHexMeshDict | Supported Types

```
box
{
    type searchableBox;
    min (-0.2 -0.2 -0.02);
    max (0.44 0.2 0.32);
}

sphere
{
    type searchableSphere;
    centre (3 3 0);
    radius 4;
}

cylinder
{
    type        searchableCylinder;
    point1       (0 0 0);
    point2       (1 0 0);
    radius      0.1;
}
```

User defined shapes

- Basic shapes → box, cylinder and sphere

snappyHexMeshDict | Supported Types

```
plane
{
    type          searchablePlane;

    planeType     pointAndNormal;
    pointAndNormalDict
    {
        basePoint      (0 0 0);
        normalVector   (0 1 0);
    }
}

plate
{
    type        searchablePlate;
    origin      (0 0 0);
    span        (0.5 0.5 0);
}
```

User defined shapes

- Basic shapes → plane and plate

snappyHexMeshDict | Supported Types

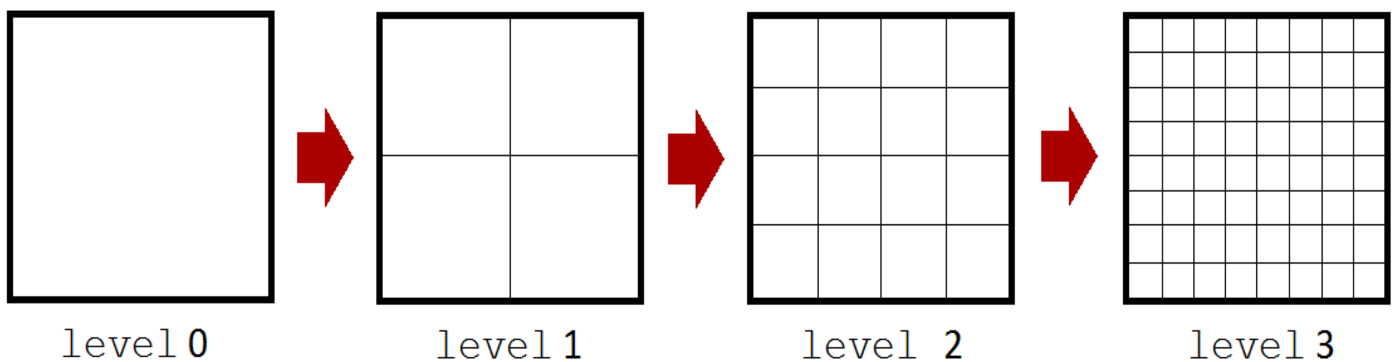
```
twoBoxes
{
    type searchableSurfaceCollection;
    mergeSubRegions true;

    boxA
    {
      surface box;
      scale (1.0 1.0 2.1);
      transform
      {
        type    cartesian;
        origin  (2 2 0);
        e1      (1 0 0);
        e3      (0 0 1);
      }
    }
    boxB
    {
      surface box;
      scale (1.0 1.0 2.1);
      transform
      {
        type    cartesian;
        origin  (3.5 3 0);
        e1      (1 0 0);
        e3      (0 0 1);
      }
    }
  }
```

User defined shapes

- Complex shapes → Collection of basic shapes scaled and transformed

snappyHexMeshDict | Refinement

The first meshing stage is called "Refinement". This is where the initial block mesh is refined based on surface and volumetric refinement settings in the **castellatedMeshControls** sub-dictionary

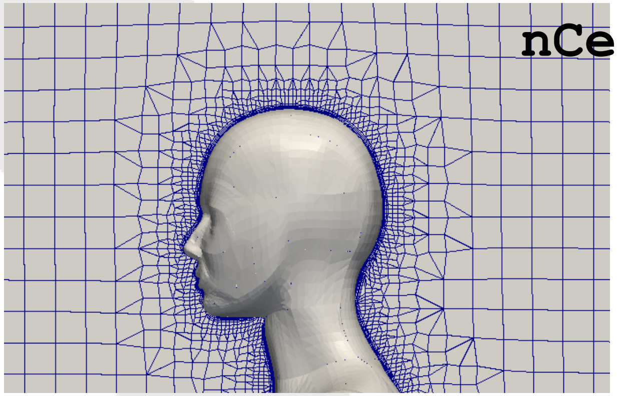


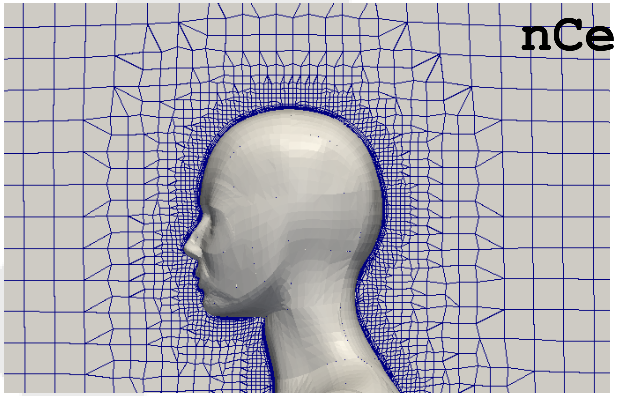
snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
   maxGlobalCells 2000000;
   minRefinementCells 0;
   nCellsBetweenLevels 1;

   features();

   refinementSurfaces
   {
      flange
      {
          level (2 3);
          regions{"*.inlet|*.outlet"{level(3,4);}}
      }
      sphereA
      {
         level (3 3);
         faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
      }
   }

   resolveFeatureAngle 30;

   refinementRegions
   {
      sphereA
      {
         mode inside;
         levels ((1E15 3));
      }
   }
   locationInMesh (-9.23149e-05 -0.0025 -0.0025);
   allowFreeStandingZoneFaces true;
}
```

Mesh control keywords:

- Global mesh size controls
- Buffer layers





snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
   maxGlobalCells 2000000;
   minRefinementCells 0;
   nCellsBetweenLevels 1;

   features();

   refinementSurfaces
   {
      flange
      {
          level (2 3);
          regions{"*.inlet|*.outlet"{level(3,4);}}
      }
      sphereA
      {
         level (3 3);
         faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
      }
   }

   resolveFeatureAngle 30;

   refinementRegions
   {
      sphereA
      {
         mode inside;
         levels ((1E15 3));
      }
   }
   locationInMesh (-9.23149e-05 -0.0025 -0.0025);
   allowFreeStandingZoneFaces true;
}
```

User-defined edge refinements

```
features
 (
     {
        file "flange.eMesh";
        level 3;
     }
 );
```

Example .eMesh file

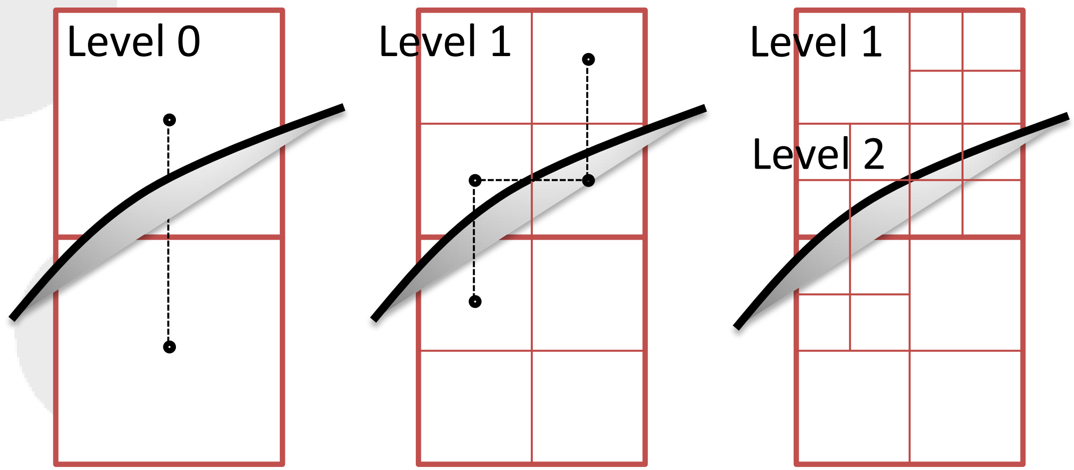
```
FoamFile
{
   version     2.0;
   format      ascii;
   class       featureEdgeMesh;
   location    "constant/triSurface";
   object      flange.eMesh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

3
(
(0.0065 0.0075 -0.02375)
(0.0065 0.0075 0.00225)
(-0.0065 0.0075 -0.02375)
)

2
(
(0 1)
(1 2)
)
```

snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
  maxGlobalCells 2000000;
  minRefinementCells 0;
  nCellsBetweenLevels 1;

  features();

  refinementSurfaces
  {
    flange
    {
       level (2 3);
       regions{"*.inlet|*.outlet"{level(3,4);}}
    }
    sphereA
    {
      level (3 3);
      faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
    }
  }

  resolveFeatureAngle 30;

  refinementRegions
  {
    sphereA
    {
      mode inside;
      levels ((1E15 3));
    }
  }
  locationInMesh (-9.23149e-05 -0.0025 -0.0025);
  allowFreeStandingZoneFaces true;
}
```

Surface based refinements:

- Global min. and max. refinements
- Refinement by patch (region)



Surface Mesh Refinements

engys

snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
  maxGlobalCells 2000000;
  minRefinementCells 0;
  nCellsBetweenLevels 1;

  features();

  refinementSurfaces
  {
    flange
    {
       level (2 3);
       regions{"*.inlet|*.outlet"{level(3,4);}}
    }
    sphereA
    {
      level (3 3);
      faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
    }
  }

  resolveFeatureAngle 30;

  refinementRegions
  {
    sphereA
    {
      mode inside;
      levels ((1E15 3));
    }
  }
  locationInMesh (-9.23149e-05 -0.0025 -0.0025);
  allowFreeStandingZoneFaces true;
}
```

Surface based refinements:

- POSIX regular expresssions supported
- patchInfo keyword can be used to set the boundary type on a per surface basis

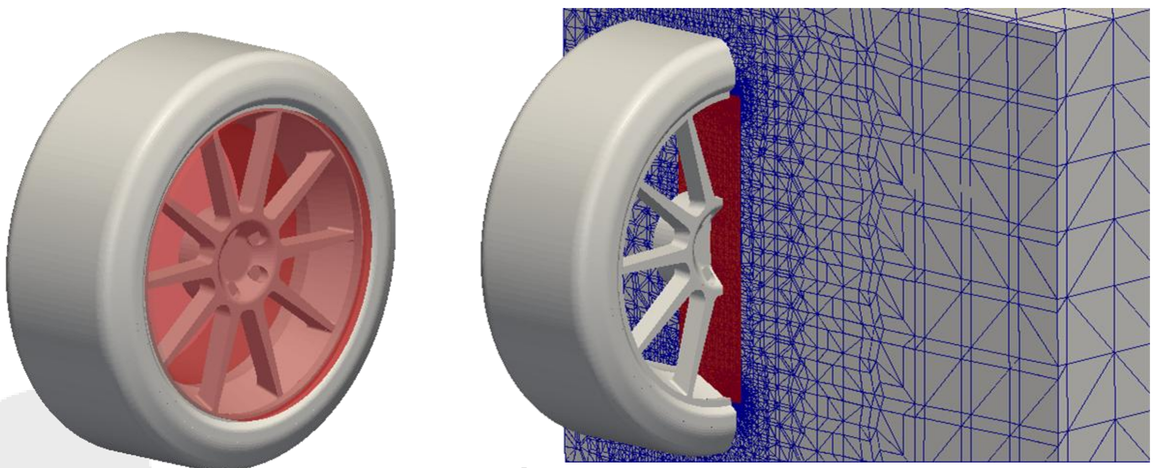
```
refinementSurfaces
  {
    flange
    {
      level (2 3);
      patchInfo
      {
        type wall;
      }
      regions
      {
        "*.inlet|*.outlet"
        {
          level(3,4);
        }
      }
    }
}
```

snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
   maxGlobalCells 2000000;
   minRefinementCells 0;
   nCellsBetweenLevels 1;

   features();

   refinementSurfaces
   {
     flange
     {
         level (2 3);
         regions{"*.inlet|*.outlet"{level(3,4);}}
     }
     sphereA
     {
        level (3 3);
        faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
     }
   }

   resolveFeatureAngle 30;

   refinementRegions
   {
     sphereA
     {
        mode inside;
        levels ((1E15 3));
     }
   }
   locationInMesh (-9.23149e-05 -0.0025 -0.0025);
   allowFreeStandingZoneFaces true;
}
```

Definition of mesh zones:

- Min. and max. refinement levels
- Cell zone name
- Face zone name
- Area selection: inside, outside or insidepoint



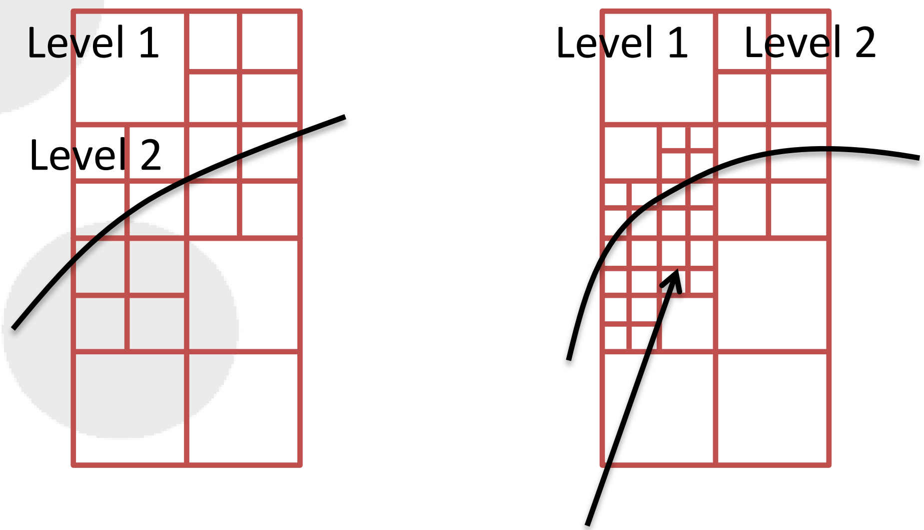
Mesh Zones

snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
  maxGlobalCells 2000000;
  minRefinementCells 0;
  nCellsBetweenLevels 1;

  features();

  refinementSurfaces
  {
    flange
    {
      level (2 3);
      regions{"*.inlet|*.outlet"{level(3,4);}}
    }
    sphereA
    {
      level (3 3);
      faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
    }
  }

  resolveFeatureAngle 30;

  refinementRegions
  {
    sphereA
    {
      mode inside;
      levels ((1E15 3));
    }
  }
  locationInMesh (-9.23149e-05 -0.0025 -0.0025);
  allowFreeStandingZoneFaces true;
}
```

Additional feature refinements:

- Local curvature
- Feature angle refinement



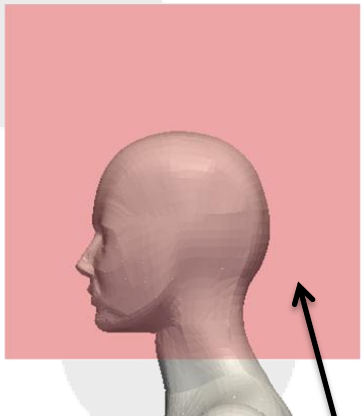
Level 3 → Local curvature refinement

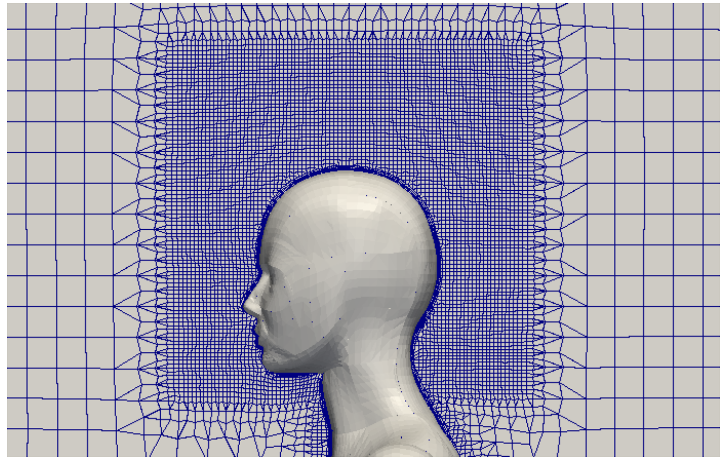
snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
   maxGlobalCells 2000000;
   minRefinementCells 0;
   nCellsBetweenLevels 1;

   features();

   refinementSurfaces
   {
      flange
      {
          level (2 3);
          regions{"*.inlet|*.outlet"{level(3,4);}}
      }
      sphereA
      {
         level (3 3);
         faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
      }
   }

   resolveFeatureAngle 30;

   refinementRegions
   {
      sphereA
      {
         mode inside;
         levels ((1E15 3));
      }
   }
   locationInMesh (-9.23149e-05 -0.0025 -0.0025);
   allowFreeStandingZoneFaces true;
}
```

Volume refinements

- inside (outside)
- distance



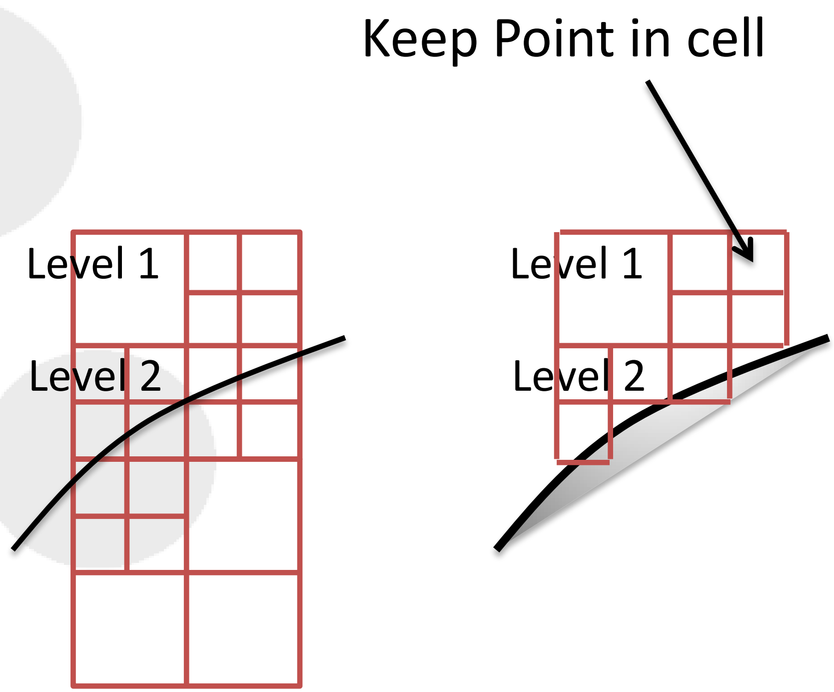


mode inside;
levels ((1E15 3));

snappyHexMeshDict | castellatedMesh

```
castellatedMeshControls
{
   maxGlobalCells 2000000;
   minRefinementCells 0;
   nCellsBetweenLevels 1;

   features();

   refinementSurfaces
   {
      flange
      {
          level (2 3);
          regions{"*.inlet|*.outlet"{level(3,4);}}
      }
      sphereA
      {
         level (3 3);
         faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
      }
   }

   resolveFeatureAngle 30;

   refinementRegions
   {
      sphereA
      {
         mode inside;
         levels ((1E15 3));
      }
   }
   locationInMesh (-9.23149e-05 -0.0025 -0.0025);
   allowFreeStandingZoneFaces true;
}
```

Cartesian point (x, y, z) to retain required volume mesh

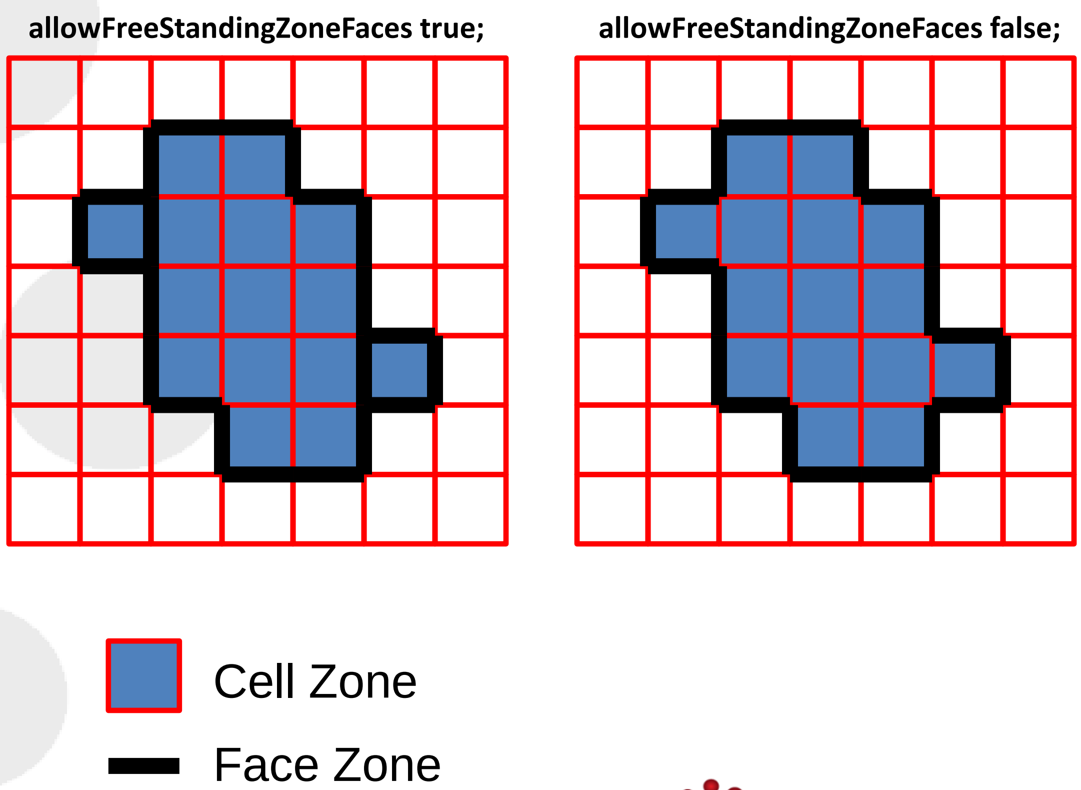


engys

snappyHexMeshDict | castellatedMesh

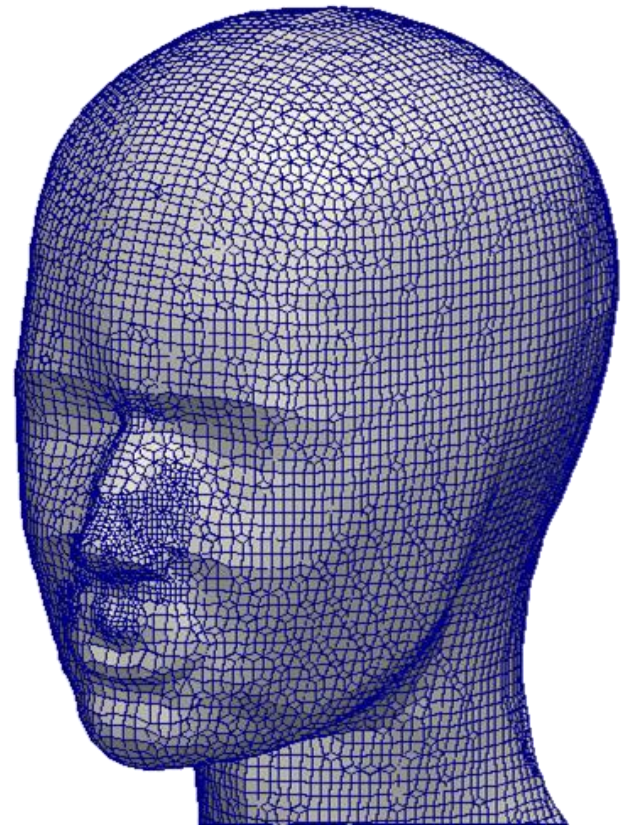
```
castellatedMeshControls
{
  maxGlobalCells 2000000;
  minRefinementCells 0;
  nCellsBetweenLevels 1;

  features();

  refinementSurfaces
  {
    flange
    {
      level (2 3);
      regions{"*.inlet|*.outlet"{level(3,4);}}
    }
    sphereA
    {
      level (3 3);
      faceZone  zoneA; cellZone zoneA; cellZoneInside inside;
    }
  }

  resolveFeatureAngle 30;

  refinementRegions
  {
    sphereA
    {
      mode inside;
      levels ((1E15 3));
    }
  }
  locationInMesh (-9.23149e-05 -0.0025 -0.0025);
  allowFreeStandingZoneFaces true;
}
```

Whether to allow zone faces that share the same owner and neighbour cell zone. If kept these can cause quality issues when the zone faces are snapped to the surface



snappyHexMeshDict | Surface Snapping

The second meshing stage is called "Snapping" where patch faces are projected down onto the surface geometry. This stage is controlled by settings in the **snapControls** sub-dictionary



snappyHexMeshDict | snapControls

```
snapControls
{
    nSmoothPatch 3;
    tolerance 1.0;
    nSolveIter 300;
    nRelaxIter 5;
    nFeatureSnapIter 10;
}
```

- nSmoothPatch: Number of pre smoothing iterations of patch points before projection to the surface is performed
- tolerance: Scaling of the maximum edge length for attraction to the surface
- nSolveIter: Number of interior smoothing iterations applied to snapped displacement field
- nRelaxIter: Controls number of scaling back iterations for error reduction stage

snappyHexMeshDict | snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolveIter 300;

    nRelaxIter 5;

   nFeatureSnapIter 10;
}
```

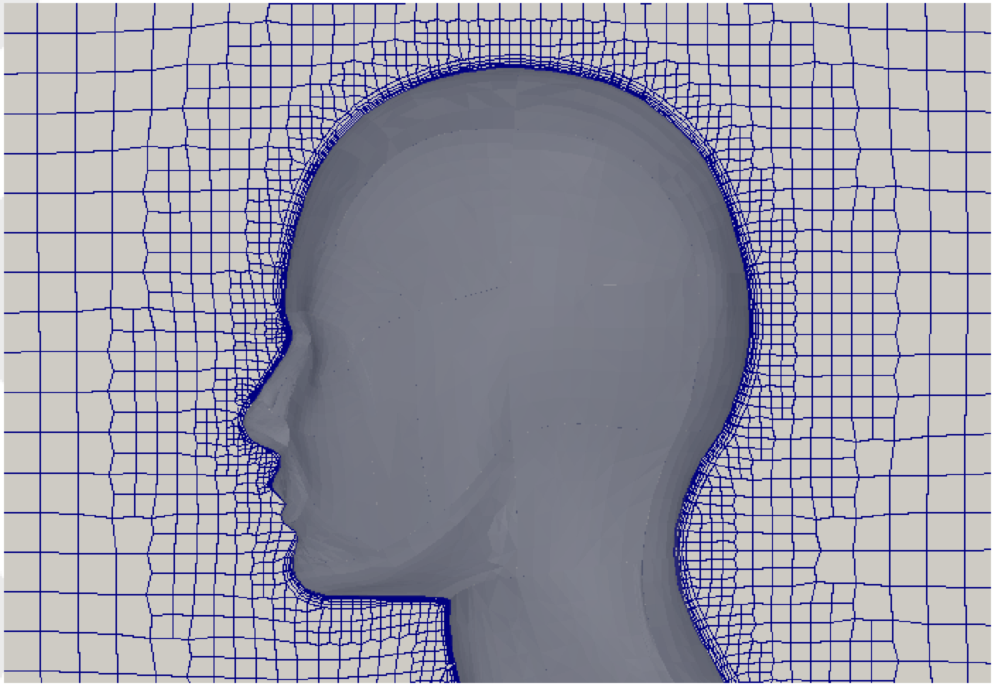
Number of feature snapping iterations to perform. Features edges to attract to are defined by an .eMesh file setup in **castellatedMeshControls** which can also be used for feature refinement.

To extract an eMesh file containing the feature edge information about a particular surface the utility **surfaceFeatureExtract** can be used e.g.

surfaceFeatureExtract -includedAngle 150 <surface> <output set>

snappyHexMeshDict | Layers

The final meshing stage is called "Layer addition" where a layer of cells is added to a specified set of boundary patches. This stage is controlled by the settings in the **addLayersControls** sub-dictionary



snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

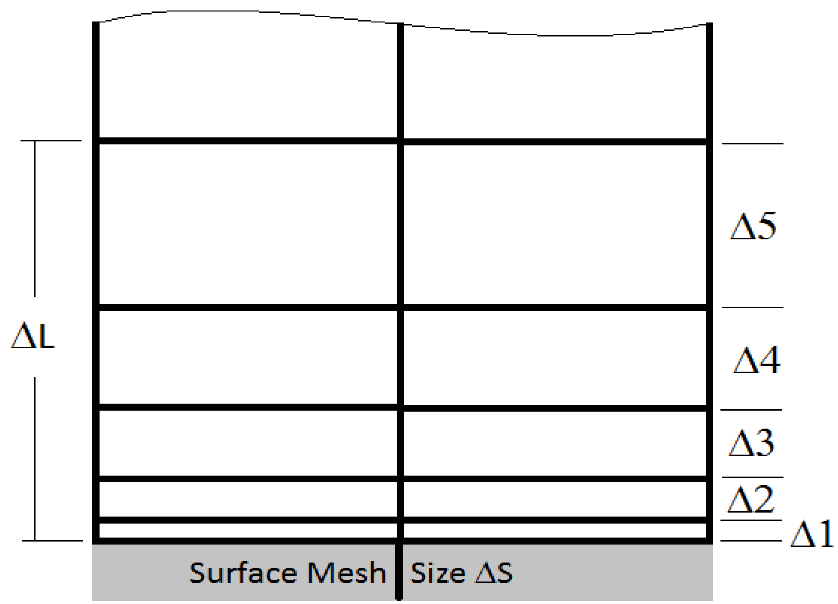
Specification of the number of layers to be grown on each patch. Supports regular expression syntax

snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

finalLayerThickness is the ratio of the final layer height relative to the adjacent surface mesh size, i.e. $\frac{\Delta 5}{\Delta S}$

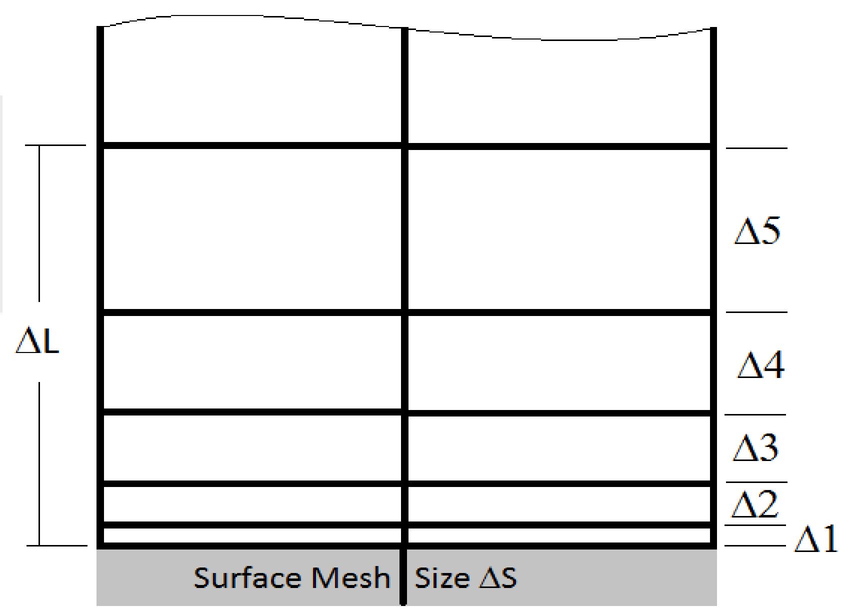
expansionRatio is the ratio of heights from one layer to the next consecutive layer in the direction away from the surface, i.e. $\frac{\Delta 2}{\Delta 1} = \frac{\Delta 3}{\Delta 2} = \frac{\Delta 4}{\Delta 3} = \frac{\Delta 5}{\Delta 4}$



snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

Specification of the number of layers, the final layer thickness and expansion ratio uniquely defines the layer profile and is used to calculate the first cell height Δ1 and total layer thickness ΔL



engys

snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

Specification of a minimum layer thickness below which height layers will automatically be collapsed.

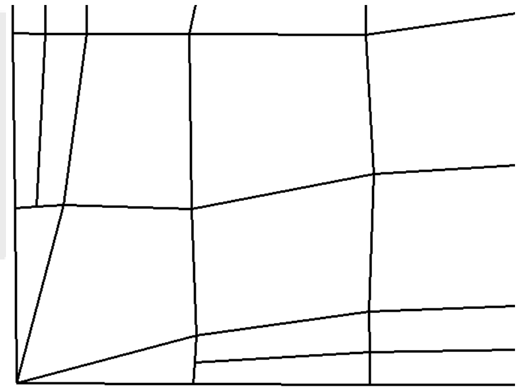
The final layer thickness and minimum thickness can be defined as either being relative (true) to the background spacing ΔS or defined as an absolute (false) length.

engys

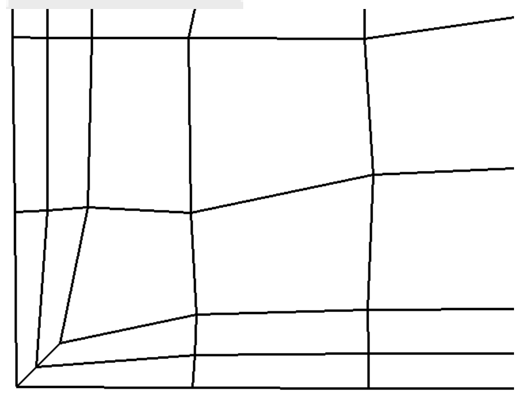
snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

Specification of feature angle above which layers are collapsed automatically



featureAngle 45;

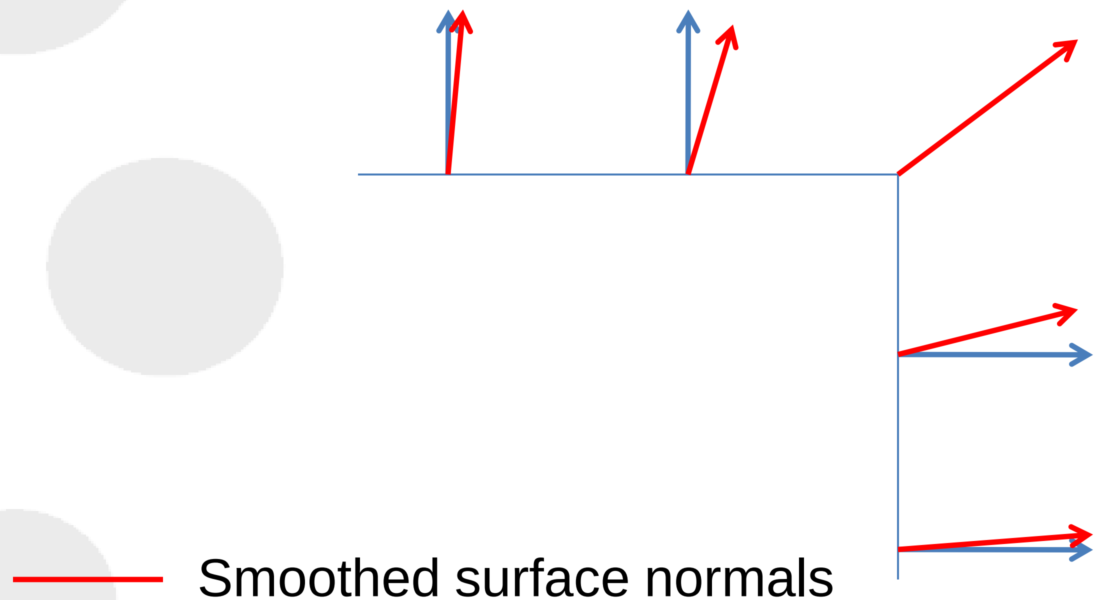


featureAngle 180;

snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

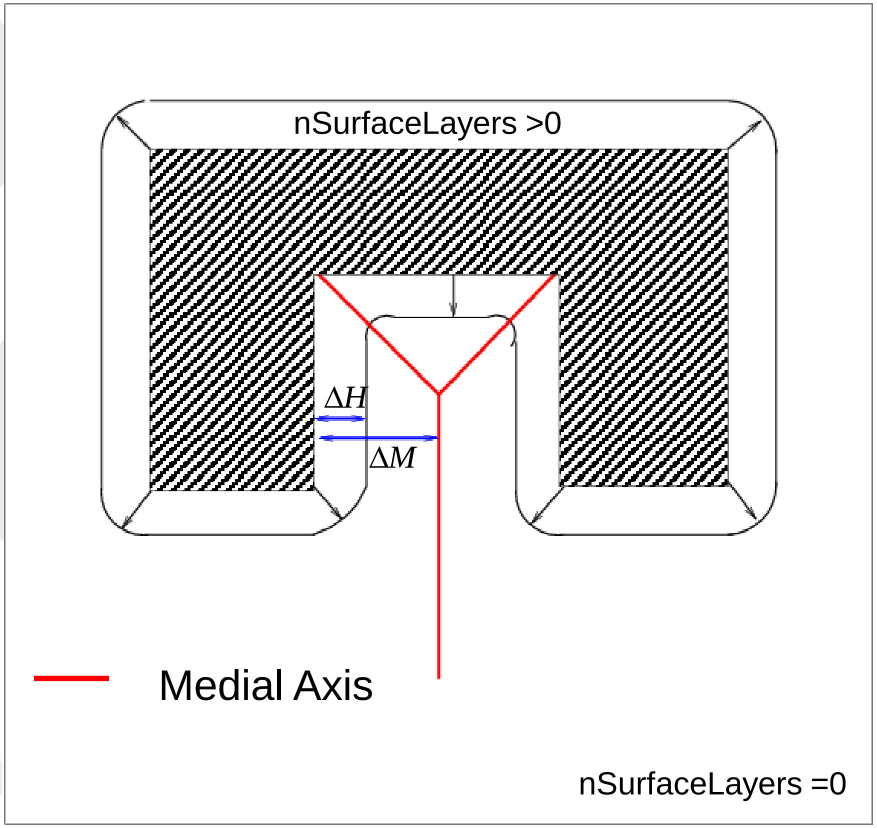
Smoothing can be performed on the surface point normals (nSmoothSurfaceNormals), layer thickness (nSmoothThickness) and the interior displacement field (nSmoothNormals) e.g.



snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

This angle is used to define a medial axis which is used when moving the mesh away from the surface

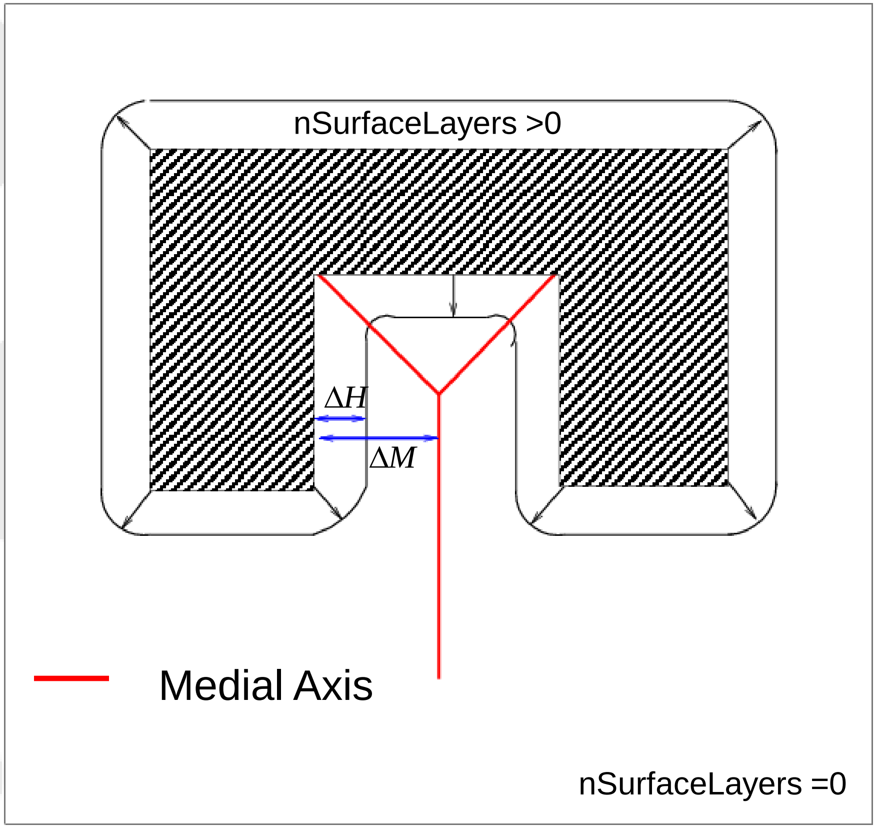


engys

snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;

  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

Used to reduce the layer thickness where the ratio of layer thickness to distance to medial axis (ΔH/ΔM) becomes too large



snappyHexMeshDict | addLayersControls

```
addLayersControls
{
   layers
   {
      "flange_.*"{nSurfaceLayers 1;}
   }

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
   featureAngle 30;

   nSmoothSurfaceNormals 1;
   nSmoothNormals 3;
   nSmoothThickness 10;

   minMedianAxisAngle 80;
   maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

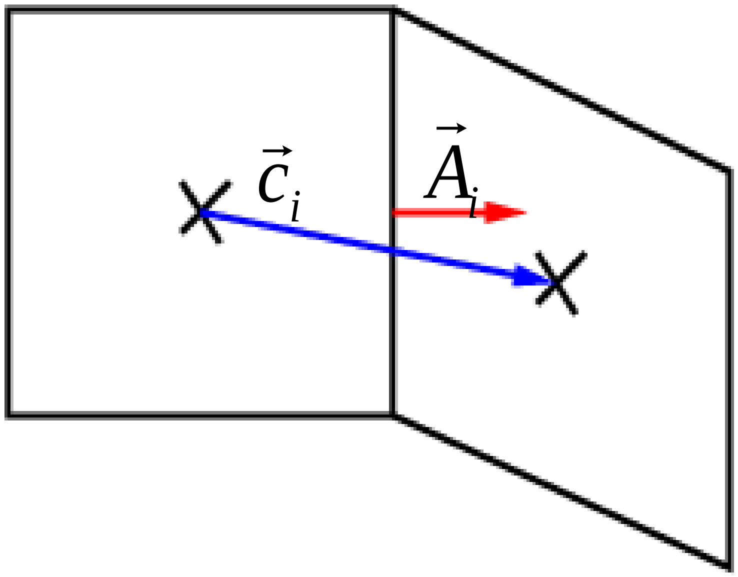
- maxFaceThicknessRatio: Used to identify warped faces and terminate layers on these faces
- nLayerIter: If the layer iteration has not converged after a certain number of iterations exit the layer addition loop early with the currently generated layer
- meshQualityControls::relaxed. nRelaxedIter: If layer iteration has not converged after a specified number of iterations then use a set of relaxed mesh quality metrics, set in **meshQualityControls,** to achieve convergence
- nRelaxIter: Controls number of scaling back iterations during error reduction stage

snappyHexMeshDict | meshQualityControls

During a run of snappyHexMesh the mesh quality is constantly monitored.

If a mesh motion or topology change introduces a poor quality cell or face the motion or topology change is undone to revert the mesh back to a previously valid error free state

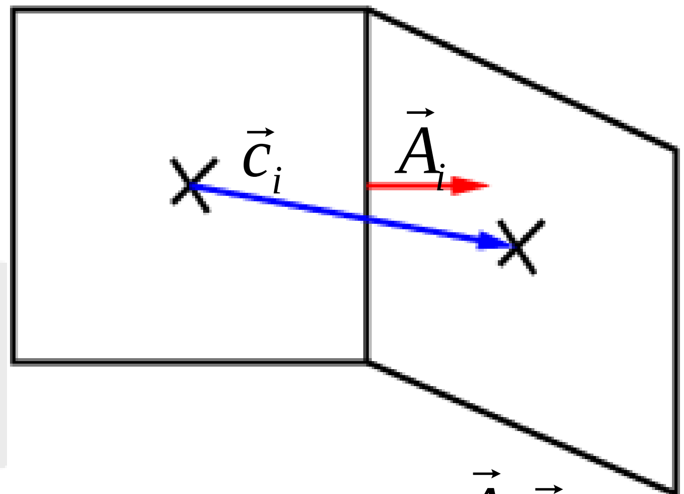
The mesh quality metrics used for the checks are set in the sub-dictionary **meshQualityControls**



snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{
  maxNonOrtho 65;

  maxBoundarySkewness 20;
  maxInternalSkewness 4;

  maxConcave 80;

   minVol 1e-13;

   minArea  1e-13;

   minTetQuality 1e-30;

  minTwist 0.05;

  minDeterminant 0.001;

  minFaceWeight 0.05;

 minVolRatio 0.01;

 minTriangleTwist -1;

  //minVolCollapseRatio 0.5;

  // Advanced
  nSmoothScale 4;
  errorReduction 0.75;

  relaxed
  {
     maxNonOrtho 75;
  }
}
```

Face orthogonality is calculated as the normalised dot product of the face area vector with a vector from the centroid of the cell to centroid of the adjacent cell



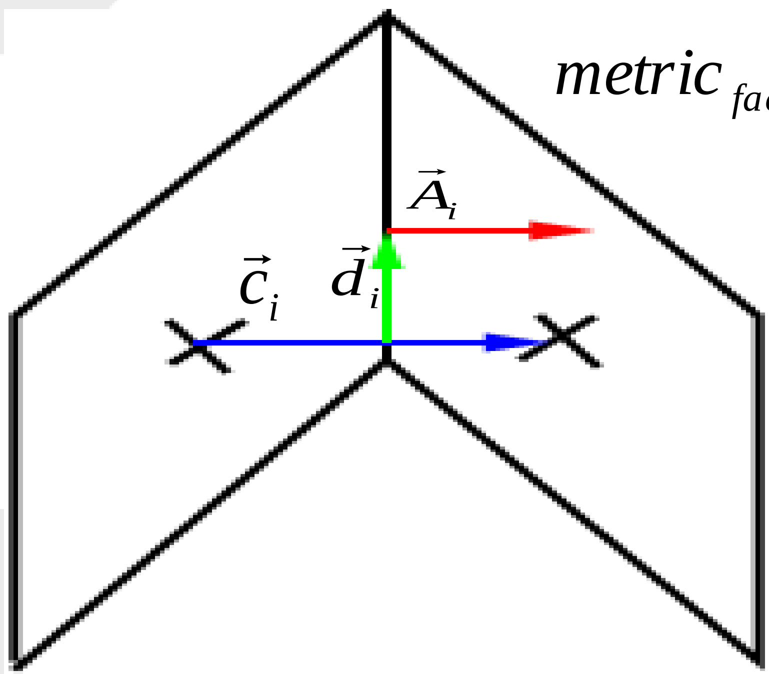
$$metric_{face,orthogonality} = \frac{\vec{A}_i . \vec{c}_i}{|\vec{A}_i||\vec{c}_i|}$$

snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

Face skewness is calculated as the distance from the face centre to the cell-centre to cell-centre face intersection point normalised by the distance from centroid of the cell to centroid of the adjacent cell

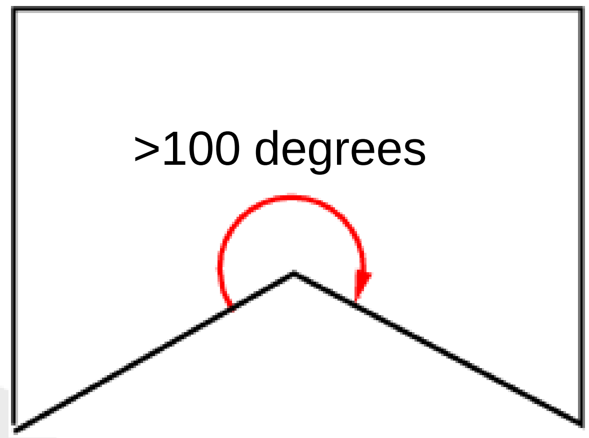
$$metric_{face,skewness} = \frac{\left|\vec{d}_i\right|}{\left|\vec{c}_i\right|}$$



snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

Face concavity makes a check of the interior angles making up the face.

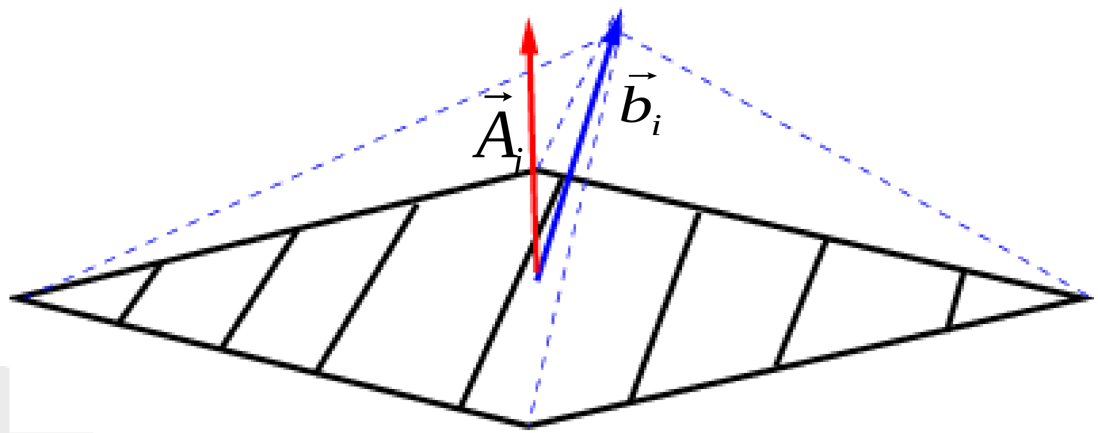


snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
  maxNonOrtho 65;

  maxBoundarySkewness 20;
  maxInternalSkewness 4;

  maxConcave 80;

   minVol 1e-13;

   minArea  1e-13;

   minTetQuality 1e-30;

  minTwist 0.05;

  minDeterminant 0.001;

  minFaceWeight 0.05;

 minVolRatio 0.01;

 minTriangleTwist -1;

  //minVolCollapseRatio 0.5;

  // Advanced
  nSmoothScale 4;
  errorReduction 0.75;

  relaxed
  {
     maxNonOrtho 75;
  }
}
```

Checks on the minimum face area and minimum cell pyramid volume. Pyramid volume is calculated from the dot product of the face area vector with the cell centre to face centre vector

$$metric_{face,\text{pyramid volume}} = \frac{1}{3}\left(\vec{A}_i \cdot \vec{b}_i\right)$$



snappyHexMeshDict | meshQualityControls

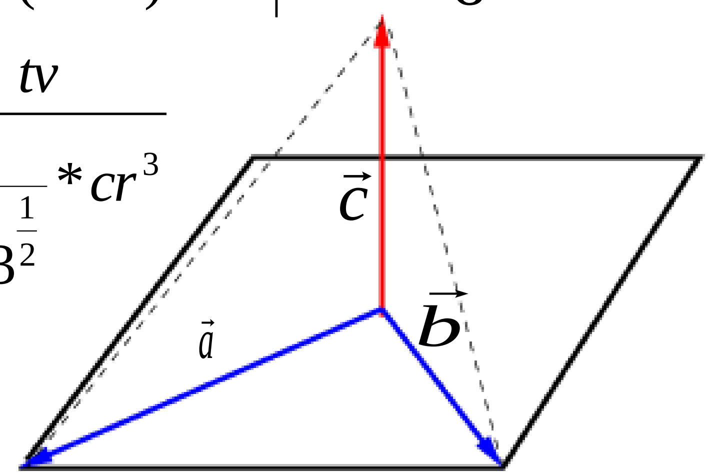
```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

Cells are decomposed into tetrahedra by using the cell centre and face centre. Tet quality is then calculated from the circumferential radius (cr) and the tetrahedral volume (tv).

$$\lambda = \left|\vec{c}\right|^2 - (\vec{a}\cdot\vec{c}) \qquad \mu = \left|\vec{b}\right|^2 - (\vec{a}\cdot\vec{b})$$

$$cr = \left| \frac{1}{2} * \left(\vec{a} + \frac{\lambda(\vec{b}\times\vec{a}) - \mu(\vec{c}\times\vec{a})}{\vec{c}\cdot(\vec{b}\times\vec{a})}\right) \right| \qquad tv = \frac{1}{6}\left(\left(\vec{a}\times\vec{b}\right)\cdot\vec{c}\right)$$

$$metric_{face,tet\ quality} \frac{tv}{\frac{8}{9*3^{\frac{1}{2}}} * cr^3}$$



snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

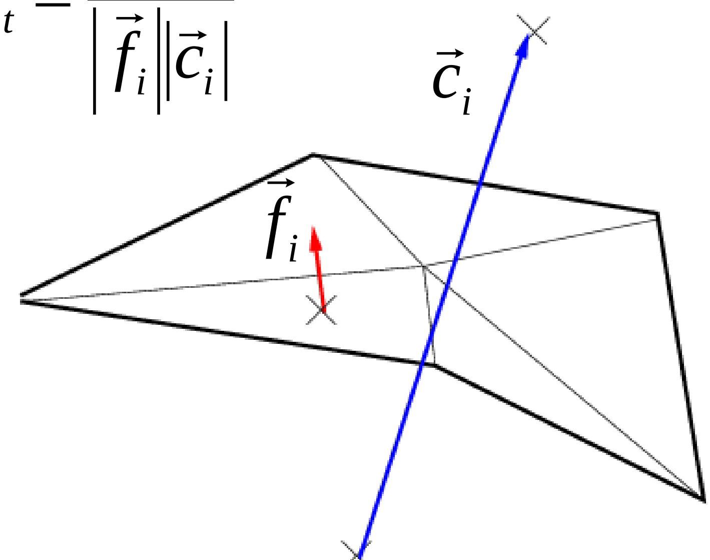
This metric needs to be set to a small positive value to guarantee an inside cell check works correctly for tracking routines e.g. streamlines

snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
  maxNonOrtho 65;

  maxBoundarySkewness 20;
  maxInternalSkewness 4;

  maxConcave 80;

   minVol 1e-13;

   minArea  1e-13;

   minTetQuality 1e-30;

  minTwist 0.05;

  minDeterminant 0.001;

  minFaceWeight 0.05;

 minVolRatio 0.01;

 minTriangleTwist -1;

  //minVolCollapseRatio 0.5;

  // Advanced
  nSmoothScale 4;
  errorReduction 0.75;

  relaxed
  {
    maxNonOrtho 75;
  }
}
```

Faces are decomposed into triangular elements using the face centre. The face twist metric is then calculated as the normalised dot product of the cell centre to adjacent cell centre vector with the triangular face area vector

$$metric_{face,twist} = \frac{\vec{f}_i . \vec{c}_i}{\left|\vec{f}_i\right|\left|\vec{c}_i\right|}$$



engys

snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

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

$$A = \sum_{faces} \left| \vec{A}_i \right| \qquad t_{i,j} = \sum_{faces} \vec{A}_i * \frac{\vec{A}_i}{\left| \vec{A}_i \right|}$$

$$metric_{cell,\mathrm{determinant}} = \left| \frac{t_{i,j}}{A} \right| / 0.03703$$

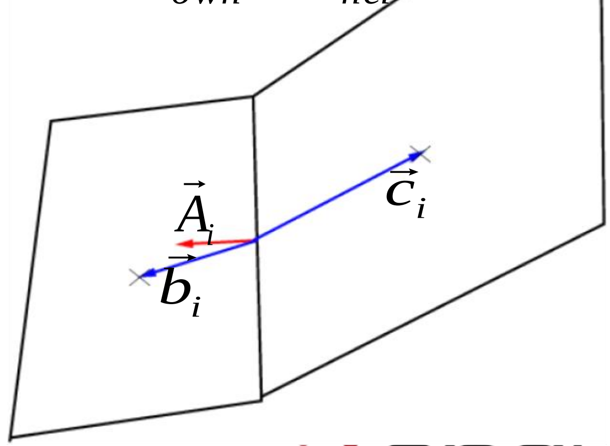
snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
  maxNonOrtho 65;

  maxBoundarySkewness 20;
  maxInternalSkewness 4;

  maxConcave 80;

   minVol 1e-13;

   minArea  1e-13;

   minTetQuality 1e-30;

  minTwist 0.05;

  minDeterminant 0.001;

  minFaceWeight 0.05;

 minVolRatio 0.01;

 minTriangleTwist -1;

  //minVolCollapseRatio 0.5;

  // Advanced
  nSmoothScale 4;
  errorReduction 0.75;

  relaxed
  {
    maxNonOrtho 75;
  }
}
```

The face weight metric is calculated as the minimum of the projected owner cell centre to face centre length and neighbour cell centre to face centre length divided by the sum of the two lengths

$$d_{own} = \frac{\left|\vec{A}_i \cdot \vec{b}_i\right|}{\left|\vec{A}_i\right|} \qquad d_{nei} = \frac{\left|\vec{A}_i \cdot \vec{c}_i\right|}{\left|\vec{A}_i\right|}$$

$$metric_{face,\mathrm{weights}} = \frac{\min(d_{nei}, d_{own})}{d_{own} + d_{nei}}$$



snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

The minimum face volume ratio metric is calculated as the ratio of the minimum of the owner and neighbour volume divided by the maximum of the two

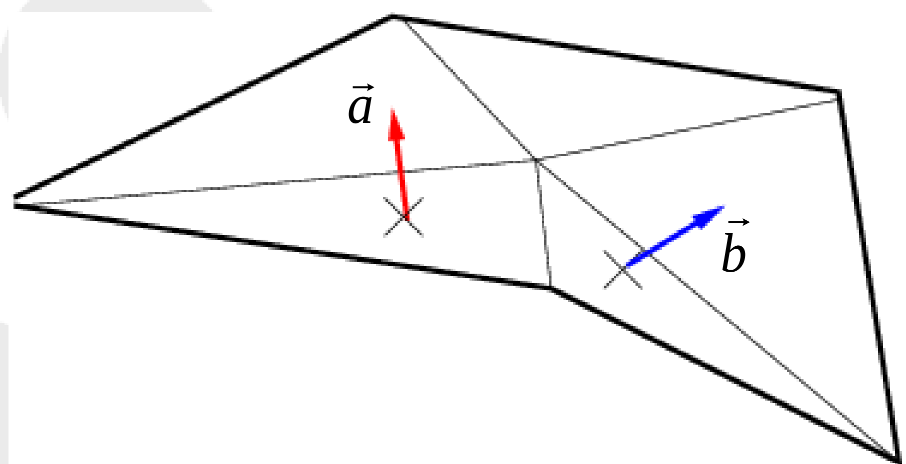
$$metric_{face,\mathrm{volumeratio}} = \frac{\min(V_{nei}, V_{own})}{\max(V_{nei}, V_{own})}$$

snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

The face triangle twist is calculated by decomposing the face into triangular elements using the face centre and then calculating the dot product from neighbouring triangular element unit normal's

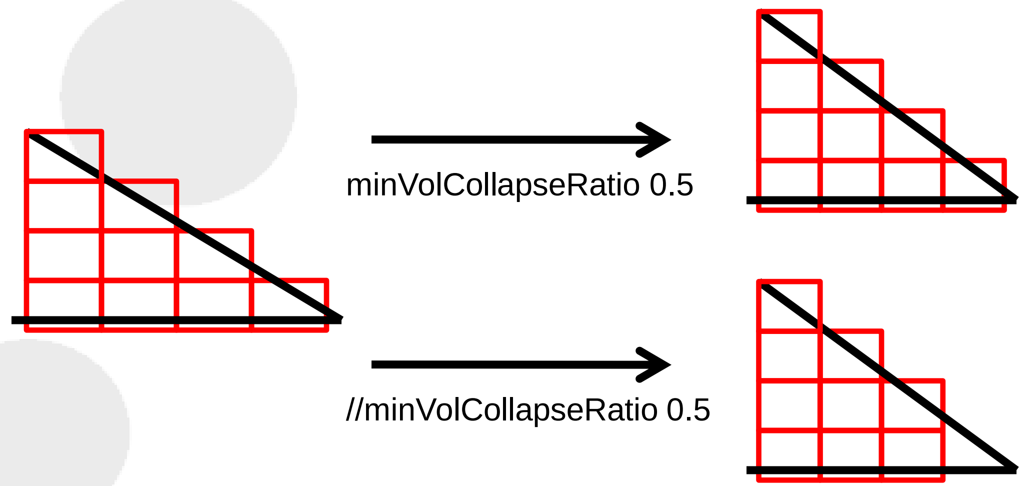
$$metric_{face,\mathrm{triangle\,twist}} = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$$



snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
  maxNonOrtho 65;

  maxBoundarySkewness 20;
  maxInternalSkewness 4;

  maxConcave 80;

   minVol 1e-13;

   minArea  1e-13;

   minTetQuality 1e-30;

  minTwist 0.05;

  minDeterminant 0.001;

  minFaceWeight 0.05;

 minVolRatio 0.01;

 minTriangleTwist -1;

  //minVolCollapseRatio 0.5;

  // Advanced
  nSmoothScale 4;
  errorReduction 0.75;

  relaxed
  {
    maxNonOrtho 75;
  }
}
```

At the end of mesh refinement cells are removed if all points on the cell are boundary ones. Enabling this keyword allows cells to be kept if boundary points projected to the surface produce a volume change less than that specified. Enabling this switch can cause stability issues because of poorly connected regions that may form



snappyHexMeshDict | meshQualityControls

```
meshQualityControls
{.
   maxNonOrtho 65;

   maxBoundarySkewness 20;
   maxInternalSkewness 4;

   maxConcave 80;

    minVol 1e-13;

    minArea  1e-13;

    minTetQuality 1e-30;

   minTwist 0.05;

   minDeterminant 0.001;

   minFaceWeight 0.05;

  minVolRatio 0.01;

  minTriangleTwist -1;

   //minVolCollapseRatio 0.5;

   // Advanced
   nSmoothScale 4;
   errorReduction 0.75;

   relaxed
   {
      maxNonOrtho 75;
   }
}
```

At the core of snappyHexMesh is the ability to scale the mesh back locally to a previously valid state where all the mesh quality criteria are known to be satisfied.

The keyword errorReduction scales the displacement field locally where ther are errors by the specified amount for each recovery iteration.

The keyword nSmoothScale applies smoothing to the displacement scaling field during each recovery iteration

snappyHexMeshDict | meshQualityControls

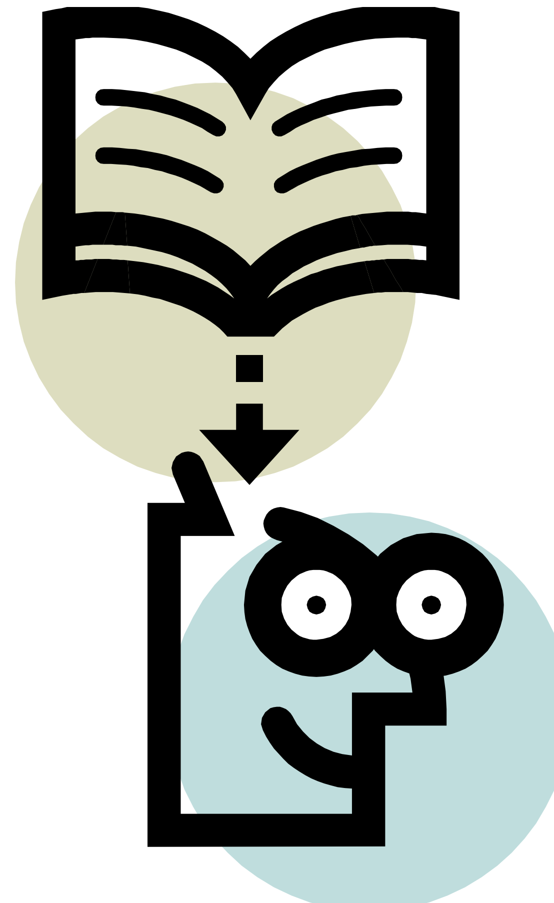
```
meshQualityControls
{.
  maxNonOrtho 65;

  maxBoundarySkewness 20;
  maxInternalSkewness 4;

  maxConcave 80;

   minVol 1e-13;

   minArea  1e-13;

   minTetQuality 1e-30;

  minTwist 0.05;

  minDeterminant 0.001;

  minFaceWeight 0.05;

 minVolRatio 0.01;

 minTriangleTwist -1;

  //minVolCollapseRatio 0.5;

  // Advanced
  nSmoothScale 4;
  errorReduction 0.75;

  relaxed
  {
    maxNonOrtho 75;
  }
}
```

A final mesh that does not satisfy all the mesh quality constraints can cause solver stability issues and divergence.

As a guide to the importance of the various mesh quality metrics a colour coding is used here. This varies from red where it is important to satisfy the mesh quality metric settings to green where satisfying the metric may not be as critical

You Will Learn About...

- Mesh Creation
- Mesh Checks
  - checkMesh



checkMesh | Definition

- Utility `checkMesh` produces a full report with mesh quality metrics and element count statistics
- Command execution only (no dictionary file required)
- Utility checks:
  - Mesh topology → Failure = Crash!
  - Patch topology for multiple connected surfaces → Failure = Crash!
  - Geometric parameters:
    - Skewness, aspect ratio, minimum face area → Failure = might still work
    - Boundary openness, non-orthogonality, minimum volume, face pyramids → Failure = Crash!
- All failed cells/faces written to *sets* in polyMesh

engys

checkMesh | Usage

- Execution:

  checkMesh [-noTopology] [-allTopology] [-latestTime] [-time ranges] [-parallel] [-constant] [-noZero] [-allGeometry] [-case dir] [-region name] [-help]

  - Parallel execution available using **mpirun**

- Requirements:
  - OPENFOAM® mesh files in *polyMesh* folder(s)
  - Dictionary *system/controlDict*

checkMesh | Output Header

```
Build  : 1.6.x
Exec   : checkMesh -parallel
Date   : Jan 01 2000
Time   : 00:00:00
Host   : node1
PID    : 21350
Case   : /home/test/exampleMesh
nProcs : 4
Slaves :
3
(
node1.21351
node1.21352
node1.21353
)

Pstream initialized with:
    floatTransfer      : 0
    nProcsSimpleSum    : 0
    commsType          : nonBlocking
```

Standard output header

checkMesh | Output Mesh Stats

```
Create time

Create polyMesh for time = 0

Time = 0
```

```
Mesh stats
    points:           8795486
    faces:            25036274
    internal faces:   24362562
    cells:            8135708
    boundary patches: 20
    point zones:      0
    face zones:       0
    cell zones:       0
Overall number of cells of each type:
    hexahedra:     7604591
    prisms:        108704
    wedges:        93
    pyramids:      11
    tet wedges:    4868
    tetrahedra:    2
    polyhedra:     417439
```

Mesh location (e.g. *constant*, time *0*, etc)

Mesh statistics:

- No. points, faces, cells...
- Global cell count

engys

checkMesh | Output Topology

```
Checking topology...
    Boundary definition OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).
```

- Check boundary mesh to ensure patch face addressing is consistent
- Check for unused points in mesh
- Check that internal faces are in upper triangular order
- Check face vertices are within point range and they are unique
- Check number of mesh regions

checkMesh | Output Patch Topology

```
Checking patch topology for multiply connected surfaces ...
```

Patch	Faces	Points	Surface topology
ceiling_Cube.003	256	263	ok (non-closed singly connected)
floor_Cube.006	623	584	ok (non-closed singly connected)
human_Sphere	1775	1652	ok (non-closed singly connected)
xmax_Cube.002	256	286	ok (non-closed singly connected)
xmin_Cube.001	256	285	ok (non-closed singly connected)

Check only shown when run in serial

Check connectivity of mesh patches

Check connectivity of mesh patches:

- closed (singly connected)
- non-closed (singly connected)
- multiply connected through a shared point
- multiply connected through a shared edge

engys

checkMesh | Output Geometry

```
Checking geometry...
```

```
Overall domain bounding box (-1.0 -1.0 -1.0) (1.0 1.0 1.0)
```
→ Bounding box dimensions

```
Mesh (non-empty, non-wedge) directions (1 1 1)

Mesh (non-empty) directions (1 1 1)
```
→ Non-constrained vectors

```
Boundary openness (2.668e-17 -4.843e-17 5.412e-17) OK.

Max cell openness = 3.720816407e-16 OK.
```
→ Openness areas

```
Max aspect ratio = 7.638021671 OK.
```
→ Max. cell aspect ratio

```
Minumum face area = 1.6762e-05. Maximum face area = 0.02712.
Face area magnitudes OK.
```
→ Face area

```
Min volume = 7.86e-07. Max volume = 0.00285. Total volume = 7.8099.
Cell volumes OK.
```
→ Cell volume

```
Mesh non-orthogonality Max: 66.35930358 average: 9.798338367

Non-orthogonality check OK.
```
→ Face orthogonality

```
Face pyramids OK.
```
→ Face pyramid volume

```
***Max skewness = 7.396435537, 109 highly skew faces detected which
may impair the quality of the results
<<Writing 109 skew faces to set skewFaces
```
→ Face skewness.

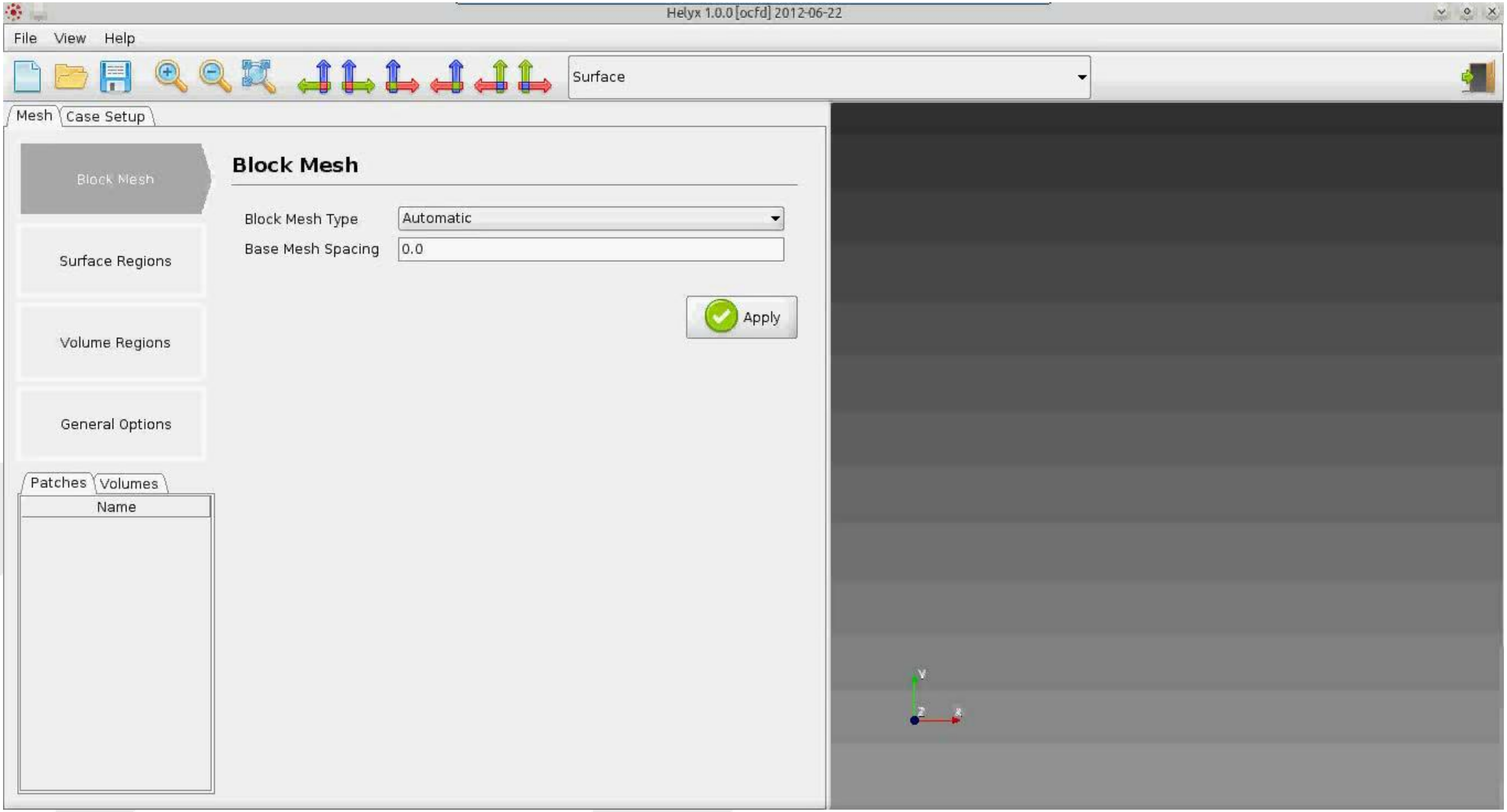
```
Failed 1 mesh checks.

End
```
→ Summary of checks →
All failed written to *sets*

engys

Meshing | Misc

- Surface Manipulation
  - surfaceCheck
  - surfaceTransformPoints
  - surfaceMeshTriangulate
- Mesh Manipulation
  - reconstructParMesh
  - renumberMesh (before using mesh)
  - topoSet with splitMeshRegions for CHT
  - transformPoints

Meshing | GUI



Helyx 1.0.0 [ocfd] 2012-06-22
File
View
Help
Surface
Mesh
Case Setup
Block Mesh
Surface Regions
Volume Regions
General Options
Patches
Volumes
Name
Block Mesh
Block Mesh Type
Automatic
Base Mesh Spacing
0.0
Apply

Questions?