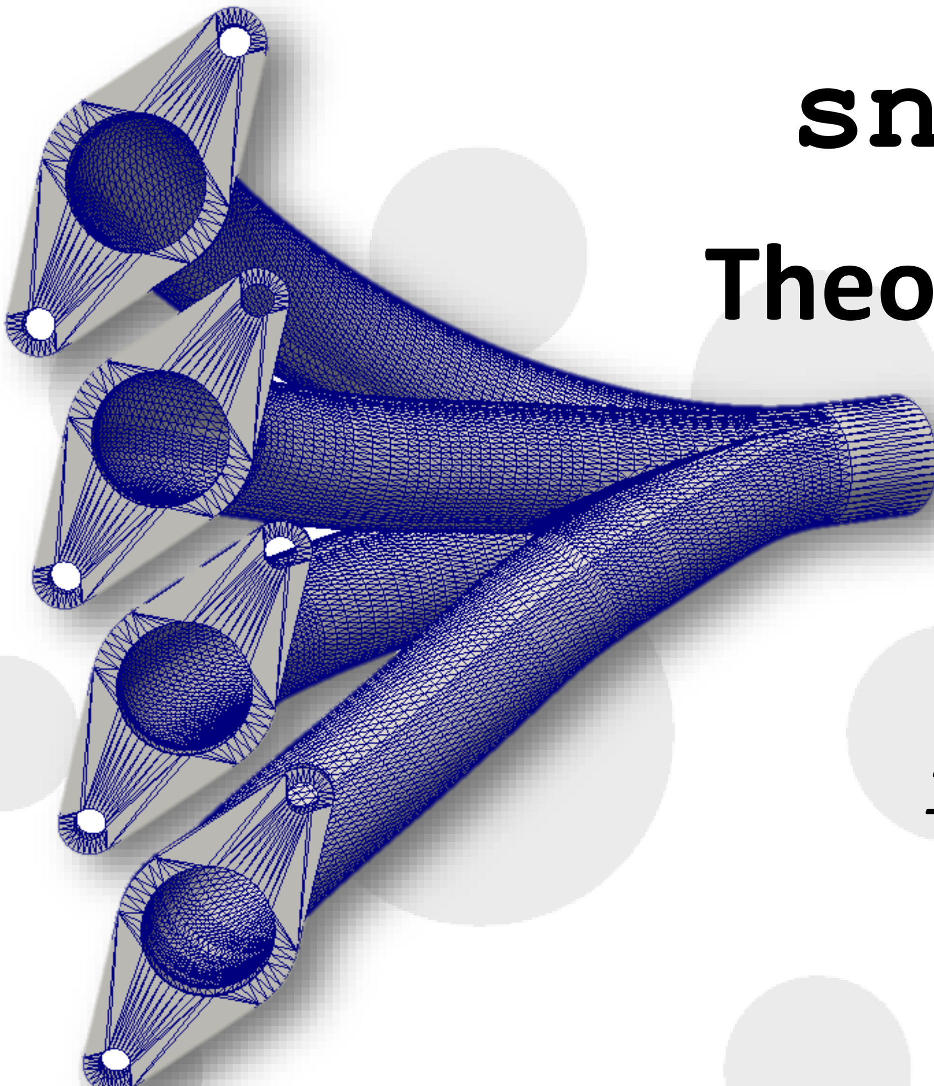


snappyHexMesh

Theory and Application

Andrew Jackson

Daniel P. Combest

11th OpenFOAM Workshop

26 June – 30 June 2016

Guimarães, Portugal

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

Disclaimer

OPENFOAM® is a trademark of OpenCFD (ESI Group). This presentation is not approved or endorsed by OpenCFD or ESI Group, the owner of the OPENFOAM® trademark.

Contents

engys

Contents

- **`snappyHexMesh`**
  - Description and Key Features
  - Background, Origin, and Forks
  - Brief Code Overview
  - Methodology Overview
- Manual Setup
- Meshing with HELYX-OS
  - Backward Facing Step
  - Pipe
  - Reactor Geometry
- Closing Remarks

engys

Description and Key Features

Utility **`snappyHexMesh`** is used to create automatically high quality three-dimensional hex-dominant meshes from input of triangulated surfaces or simple primitive shapes

Description and Key Features

Fully parallel execution

- Base mesh is created in serial and then distributed to n-processors
- True parallel performance depends on mesh composition but tens of millions of cells can be created rapidly on complex geometries
- Optimal decomposition via ptScotch and dynamic load balancing allows for improved performance

Description and Key Features

Surface, Volume, Edge Refinement

- Surface refinement based on curvature of the input geometry
- Volume refinement (inside/outside/distance) based on primitive objects or additional imported geometry
- Edge refinement based on eMesh description

Description and Key Features

Feature Edge and Surface Detail Preservation

- Automatic surface mesh created during edge, surface, and volume refinement and feature snapping phase
- Users provide STL, OBJ, or NASTRAN surface mesh files
- Surface detail is controlled by surface geometry detail and local cell size

engys

Description and Key Features

Zonal Meshing

- Allows for creation of cellZones for source terms e.g. porous media, MRF and other fvOptions
- Enables multi-region meshing for conjugate heat transfer and/or dynamic mesh cases with automatic generation of coupling via AMI patches

Description and Key Features

Wall Layer Addition

To allow for better modeling of near wall phenomena e.g. boundary layer formation

- Near wall first cell height, total layer thickness, number of layers, etc. to be specified

Description and Key Features

Quality guaranteed final mesh that will run in OPENFOAM®

Key mesh quality metrics include:

- Orthogonality
- Pyramid Volume
- Concavity
- Face Area
- Skewness
- Tet Decomposition Quality
- Face Twist
- Determinants
- Face Weights
- Volume Ratios

Meshes that do not meet **quality criteria** are scaled back to previous "high-quality" state

engys

Background | Original and Current Dev.

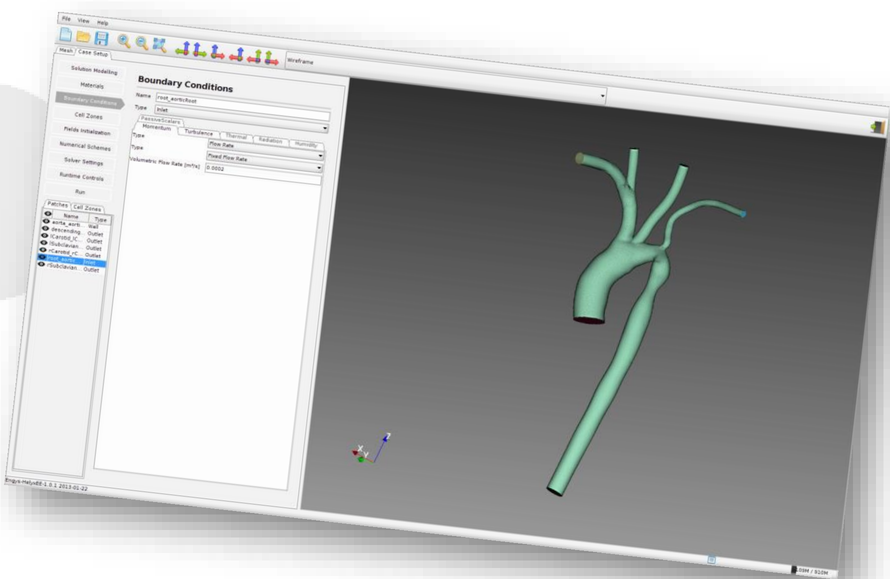
- Introduced 3rd OpenFOAM Workshop in Milan 2008
  - "Automatic Parallel Polyhedral Mesh Generation on Complex Geometries in OpenFOAM" E. de Villiers, A. Jackson (Engys), M. Janssens (OpenCFD Ltd)
- A version of snappyHexMesh called helyxHexMesh continues to be developed by Engys

Background | Original and Current Dev.

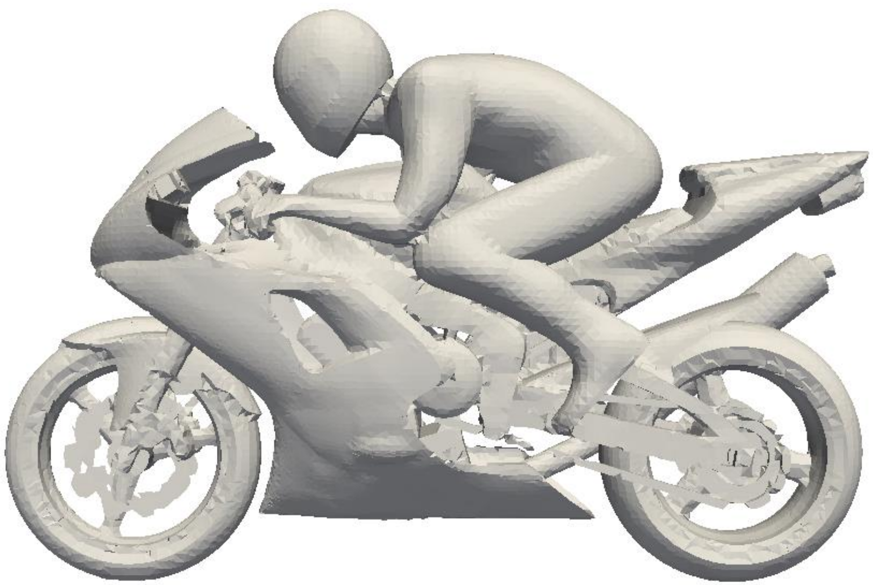
ENGYS' Continued Developments on an In-House Version

- Enhanced feature capturing and automation
- Improved layers and layer specification methods
- Anisotropic volumetric refinement
- Generation of Internal layers
- Topological improvements
- Automatic block mesh creation and decomposition
- Mesh wrapping and small leak closure
- and many other extensions

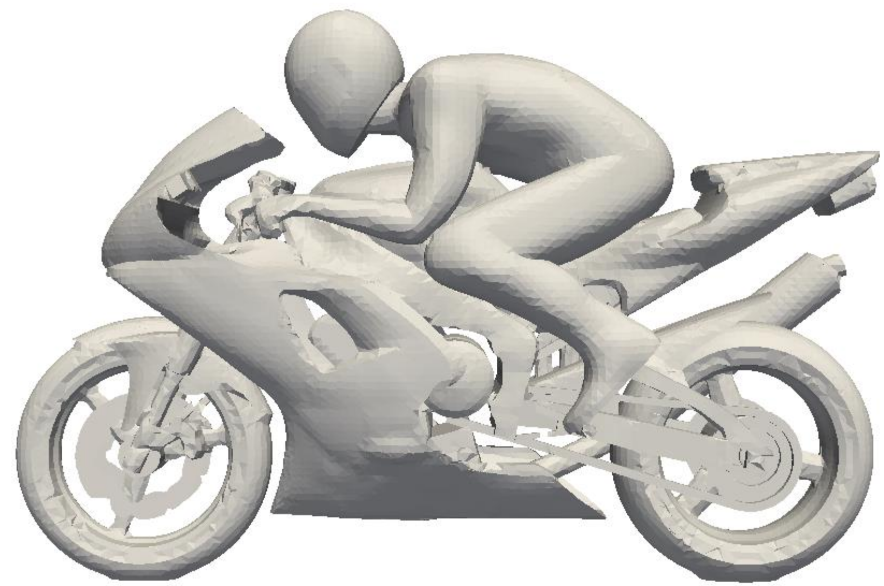
- Enhanced parallel performance
  - Reduced overall memory usage
  - Improved scaling for 32+ processors
- Graphical User interface Integration
  - HELYX
  - HELYX-OS
  - ELEMENTS



Background | Motorbike Tutorial Case

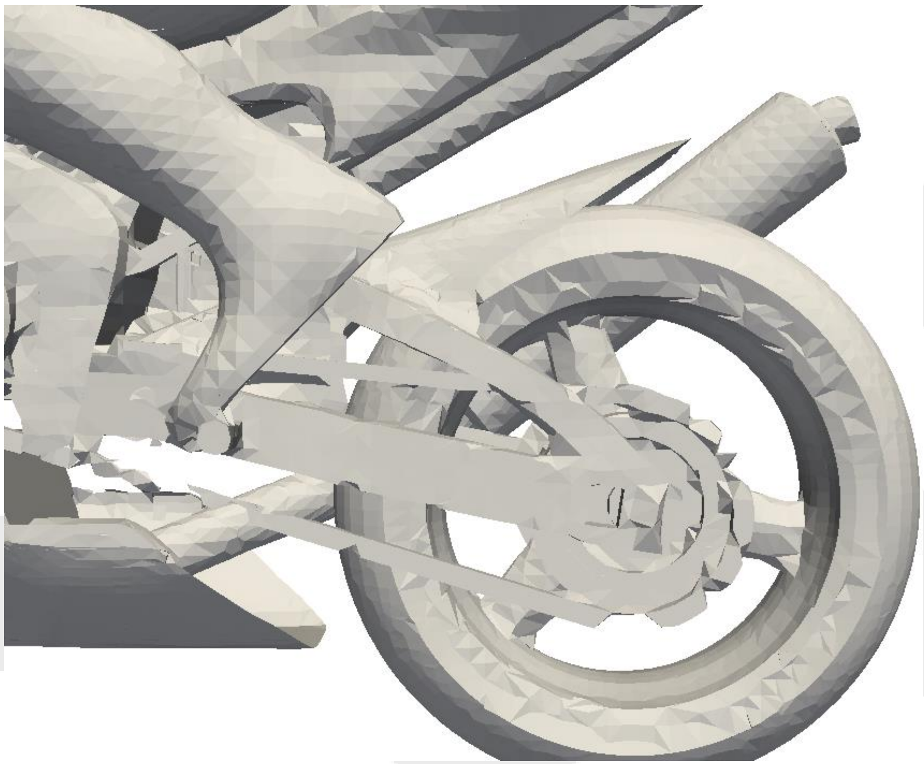


OpenFOAM-Plus snappyHexMesh
Number Cells 352K
Layer coverage 57%

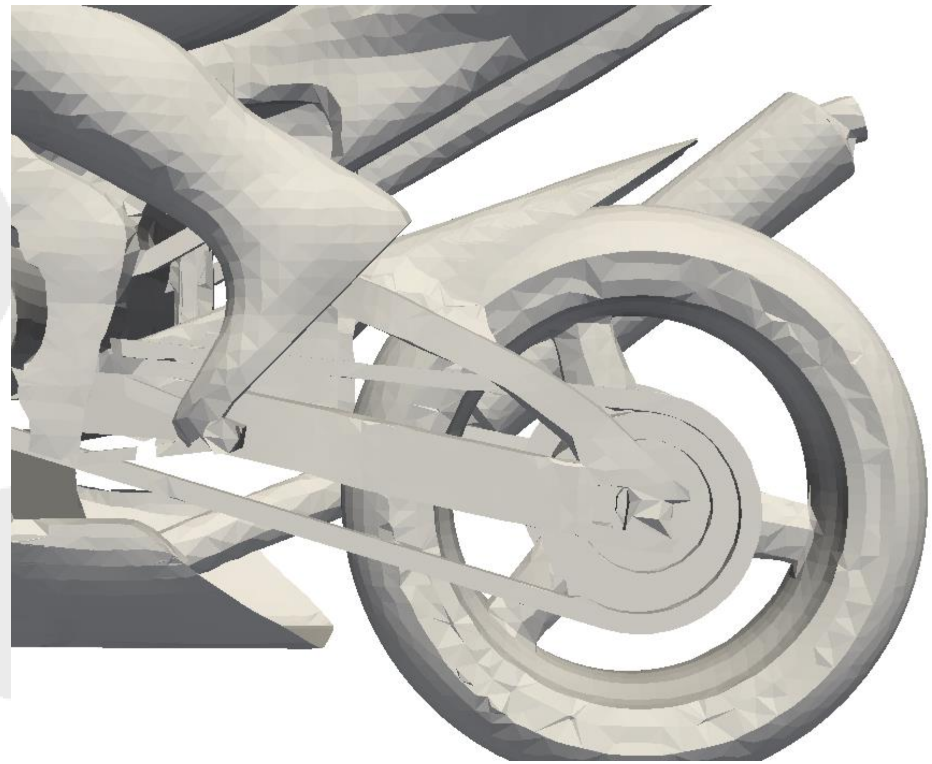


helyxHexMesh
Number Cells 362K
Layer coverage 85%

Background | Motorbike Tutorial Case



OpenFOAM-Plus snappyHexMesh
Number Cells 352K
Layer coverage 57%



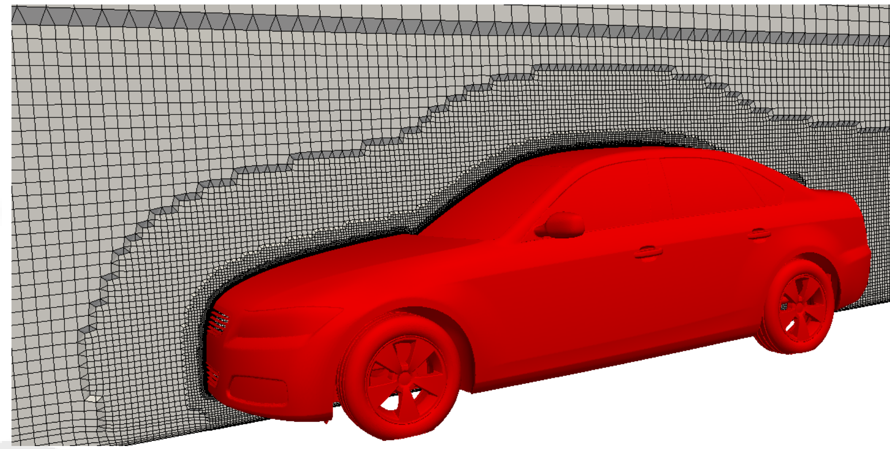
helyxHexMesh
Number Cells 362K
Layer coverage 85%

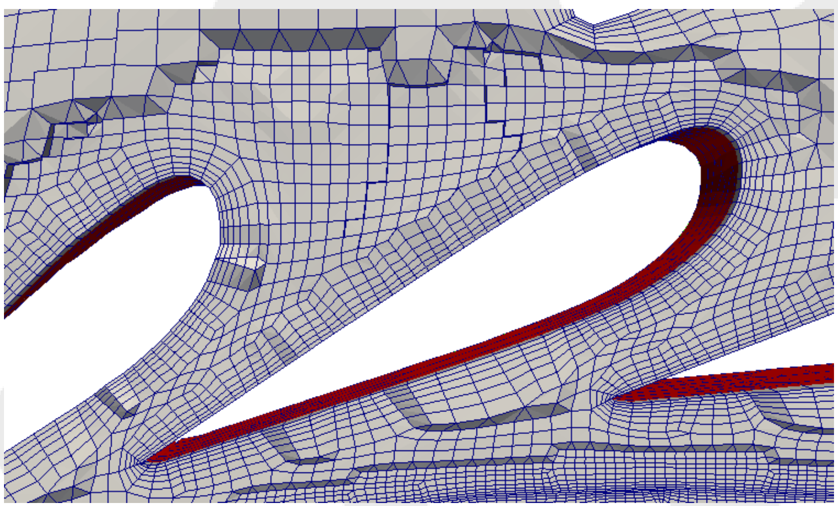
engys

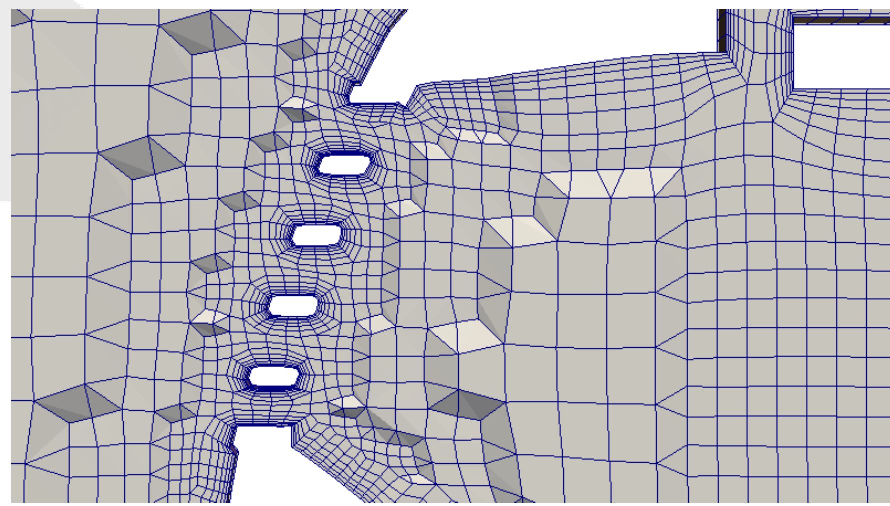
Background | Original and Current Dev.

ENGYS currently working on new dualised mesh generator

- Improved layer coverage
- Reduced jump in cell volume at refinement interfaces
- Optimisation to improve mesh quality
- Improved surface topology







Code Overview | snappyHexMesh

Overview of snappyHexMesh.C

- Reads the base mesh
- Reads the geometry files
- Reads all user provided information from system/snappyHexMeshDict
- Instantiates and calls mesh refinement, snapping, and layer addition drivers
- Outputs balanced mesh

Majority of the work is performed in separate classes constructed in this utility

Code Overview | C++ Classes

Main Library `autoMesh`

- Contains refinement, snapping and layer addition routines in 44k lines of code

Core OpenFOAM libraries

- Leveraging data structures and methods for performing low level tasks including
  - changing the mesh topology
  - octree's for surface intersection checking
  - Determine optimal decomposition during load balancing

Code Overview | C++ Classes

Geometry

- ***searchableSurfaces.H*** container for searchable objects with methods for finding nearest and line point intersections with the surface
  - ***triSurfaceMesh.H*** constructs triangulated surface and uses ***indexedOctree.H*** class for hierarchical recursive search on these
  - ***searchableCylinder.H*** and other primitive shape (e.g **searchableSphere.H** and **searchablePlane.H**) classes are available

engys

Code Overview | C++ Classes

Refinement Containers

- ***refinementSurfaces.H*** class is container for data used for surface driven refinement e.g. refinement level
- ***refinementFeatures.H*** class is a container for data used for feature based queries
- ***shellSurfaces.H*** class is a container for performing queries for volumetric based refinement

Code Overview | C++ Classes

Refinement

- ***autoRefineDriver.H*** *class* controls which refinement methods are used
- ***meshRefinement.H*** class for selecting which cells to refine and zone. Controlled by separate included files
  - Cells to refine (meshRefinementRefine.C)
  - Cells to zone (meshRefinementBaffles.C)
  - Cells remove based on topology and geometric checks (meshRefinementProblemCells.C)
  - Patch faces to merge (meshRefinementMerge.C)
- ***hexRef8.H*** class performs the actual refinement of (split) hexes using the ***polyTopoChange.H*** class. This is also the same engine used by the dynamic mesh refinement

Code Overview | C++ Classes

Snapping

- ***snapParameters.H*** class is container for snap specific information
- ***autoSnapDriver.H*** class performs snapping of mesh to the surface and methods for feature extraction (see autoSnapDriveFeature.C)

Code Overview | C++ Classes

Layers

- ***layerParameters.H*** class is container for layer specific information
- ***autoLayerDriver.H*** controls methods for layer generation
- ***externalDisplacementMeshMover.H*** is virtual base class for mesh motion away from the surface. The default method is based on a medial axis calculation (see medialAxisMeshMover.H) . A displacement motion solver method is also available (see displacmentMotionSolverMeshMover.H)
- ***addPatchCellLayers.H*** class performs addition of layer of cells to an indirect primitive patch (class is also used by extrudeMesh utility)

Code Overview | C++ Classes

Mesh Quality

- ***motionSmoother.H*** class part of the dynamicMesh library is used for performing mesh scaling back based on mesh quality checks (see meshSmootherAlgoCheck.C)
- ***polyMeshGeometry.H*** class performs actual mesh quality checks and flags faces and cells that are in error

Methodology Overview | Usage

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

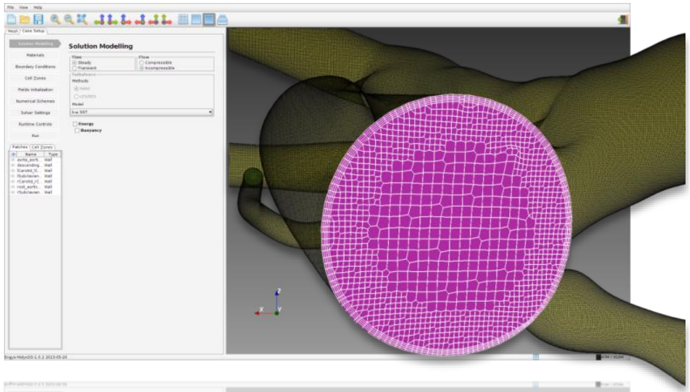
  snappyHexMesh [-noFunctionObjects][-overwrite] [-parallel] [-checkGeometry][-surfaceSimplify][-case dir] [-roots <(dir1 .. dirN)>] [-help]

  - 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*)

Methodology Overview | Workflow

- Define blockMeshDict
- Define snappyHexMeshDict
- Define decomposeParDict
- Execute `blockMesh`
- Execute `decomposePar`
- Execute `snappyHexMesh`

- Manually
- Scripted
- Graphical User Interface

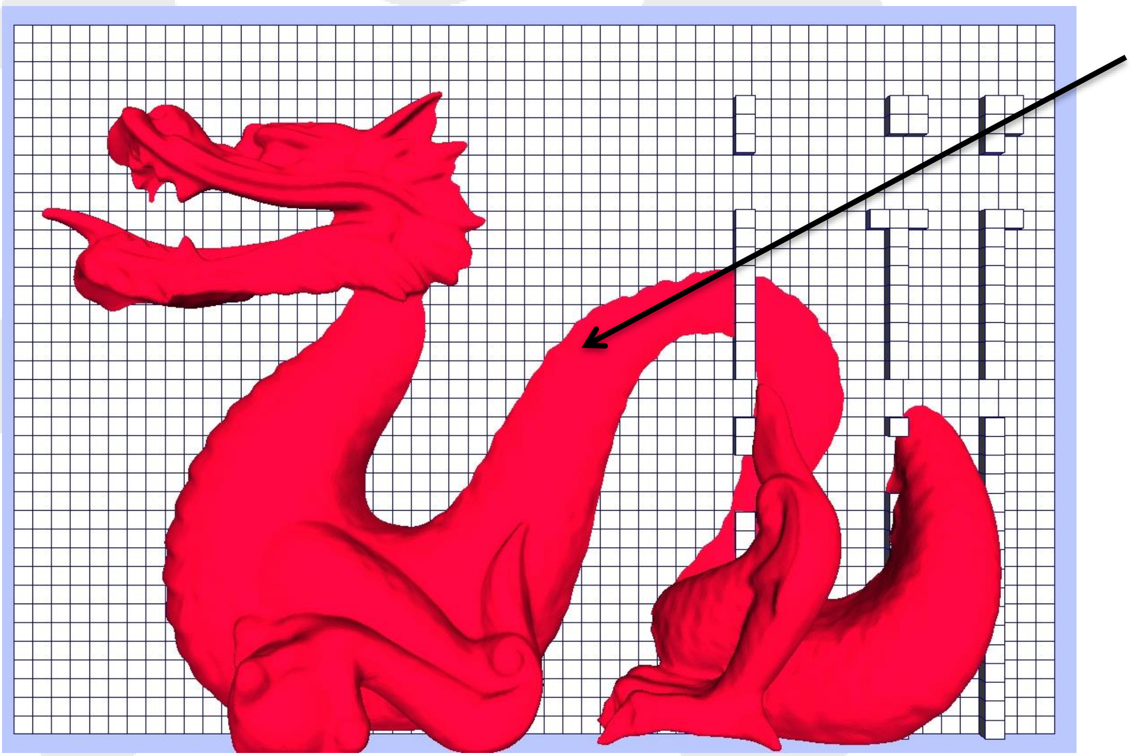


engys

Methodology Overview | Base Mesh

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

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



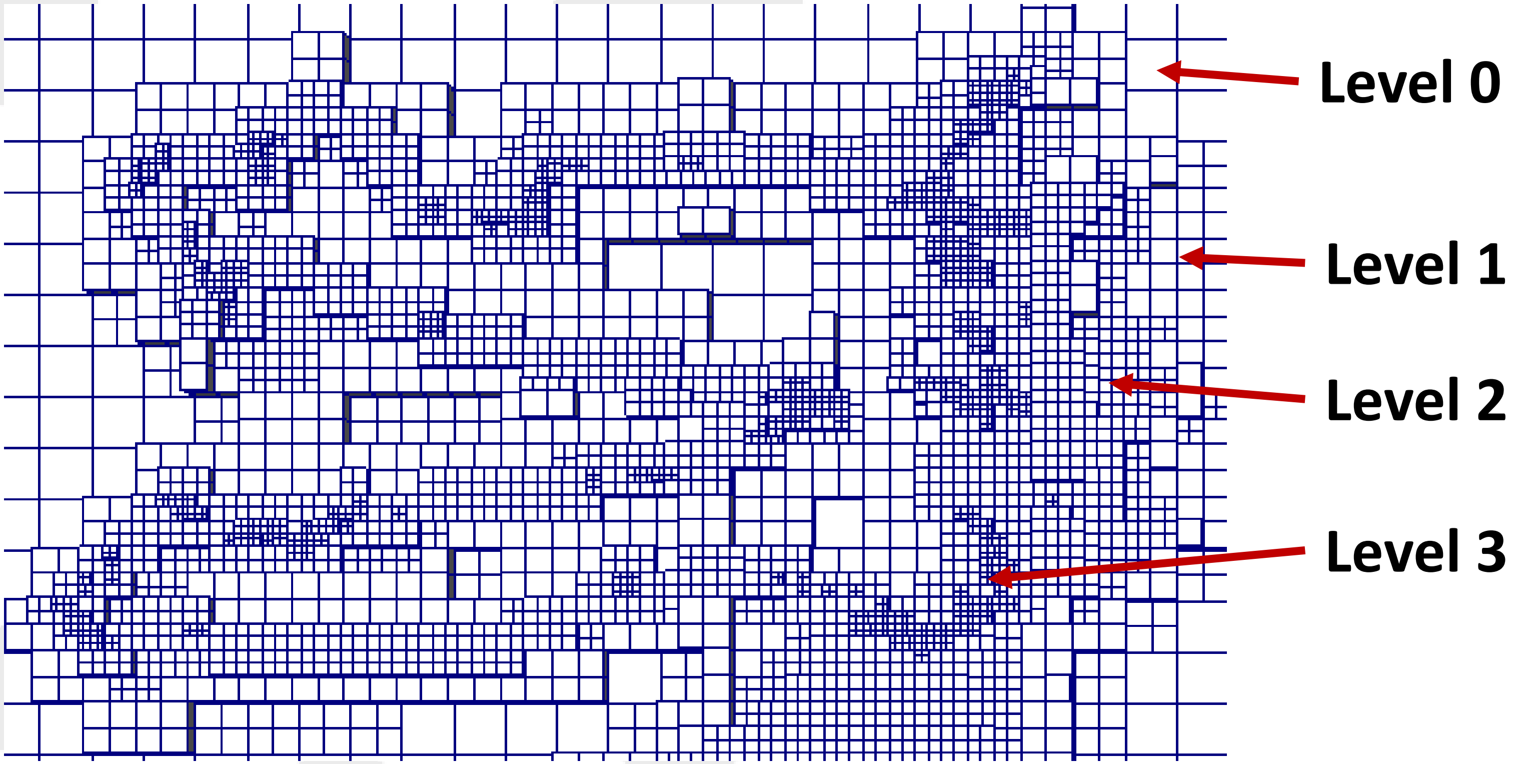
STL or Nastran (*constant/triSurface*)

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

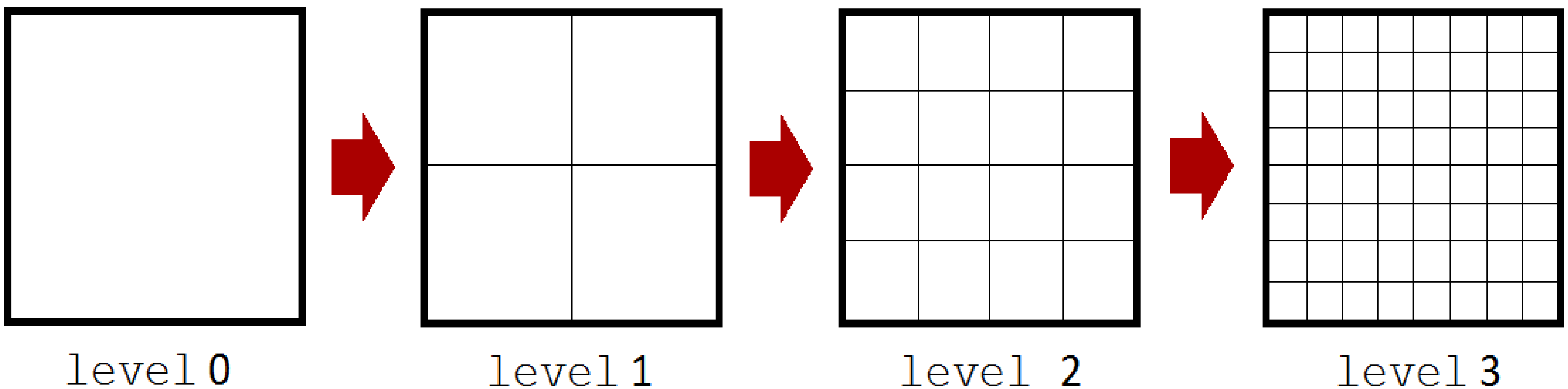
Methodology Overview | Base Refine

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



Methodology Overview | Base Refine

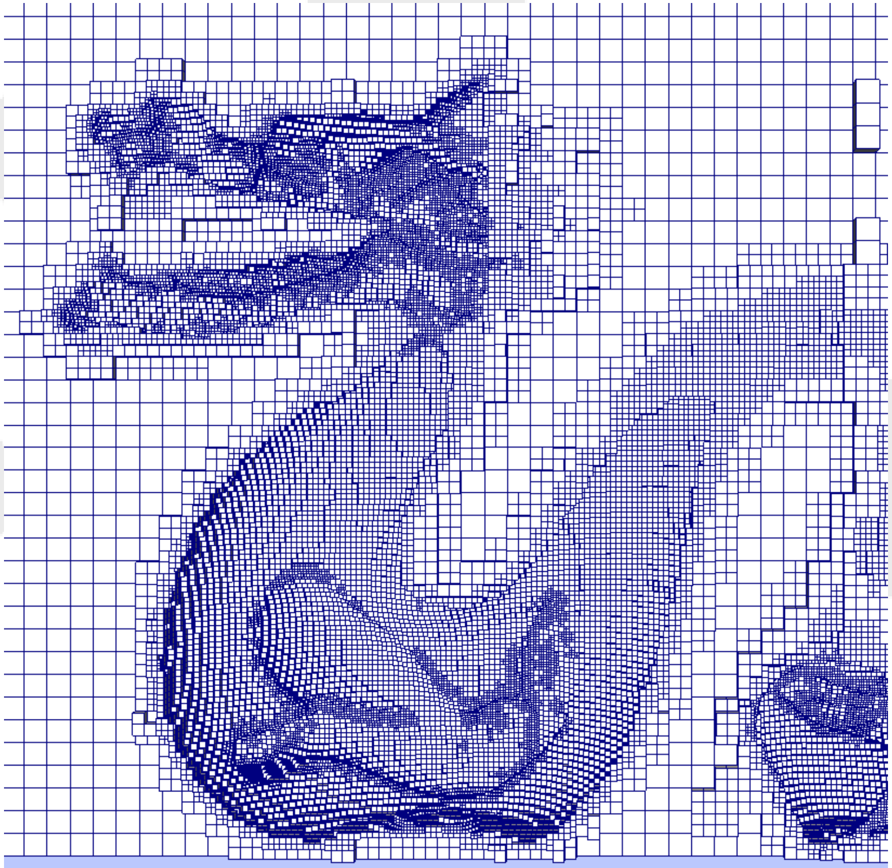
- Step 2: Refine base mesh
  - Surface refinement → feature lines, proximity & curvature
  - Volume refinement → closed surfaces, geometric shapes



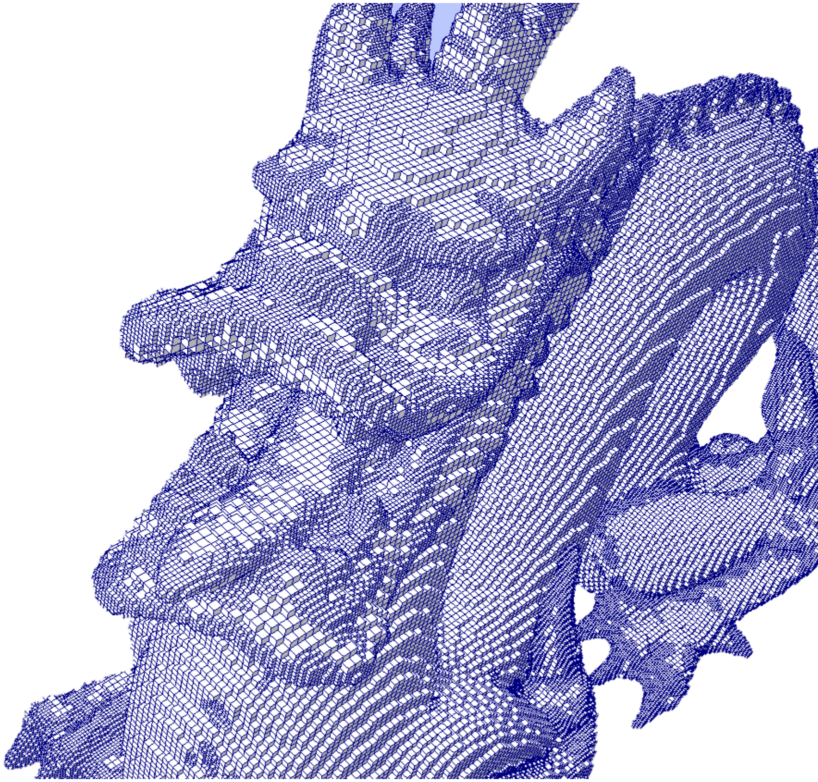
$$\sqrt[3]{V_{Cell}} = L_n = \frac{L_0}{2^n}$$

Methodology Overview | Remove Cells

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



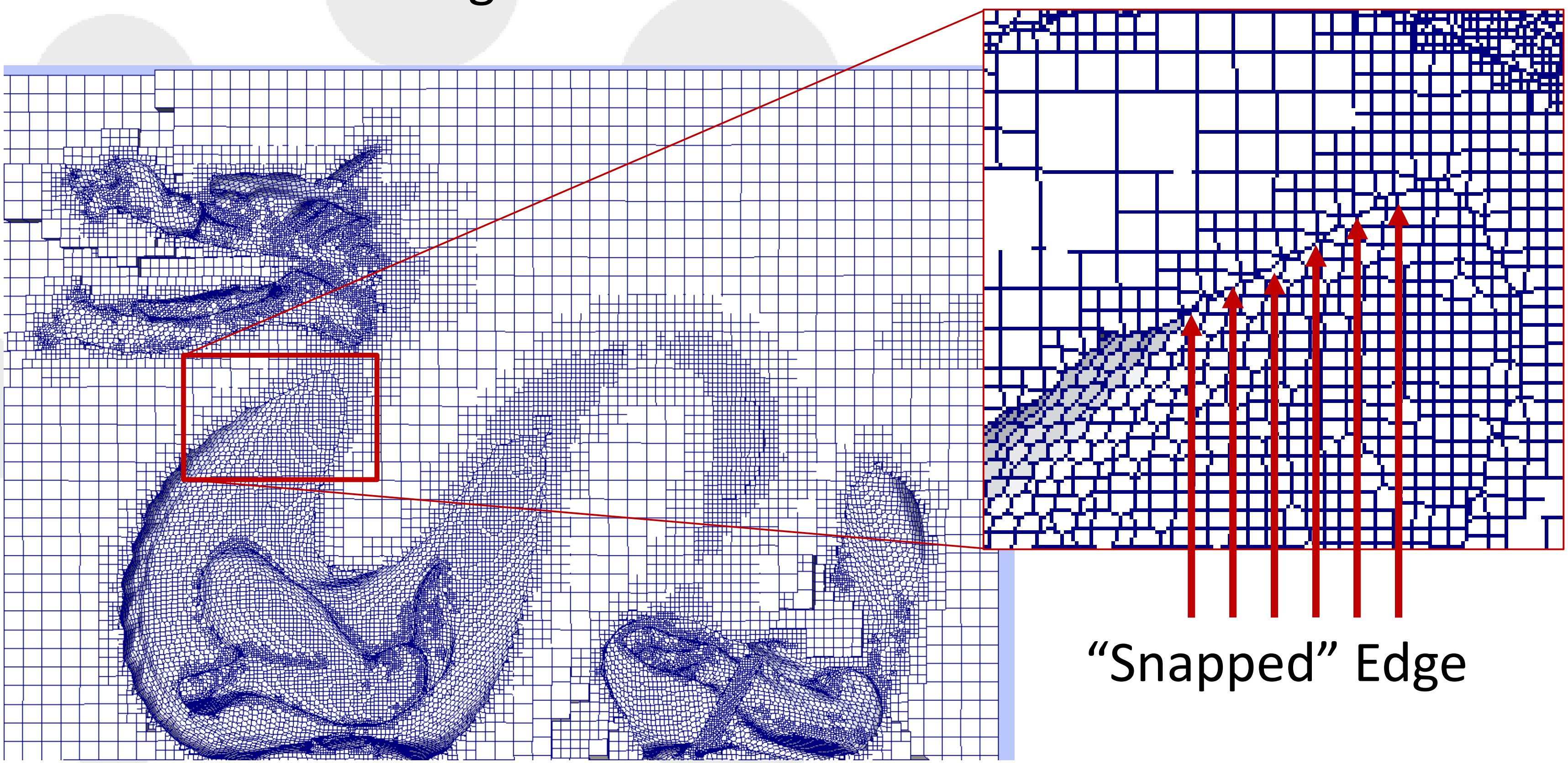
Castellated mesh



engys

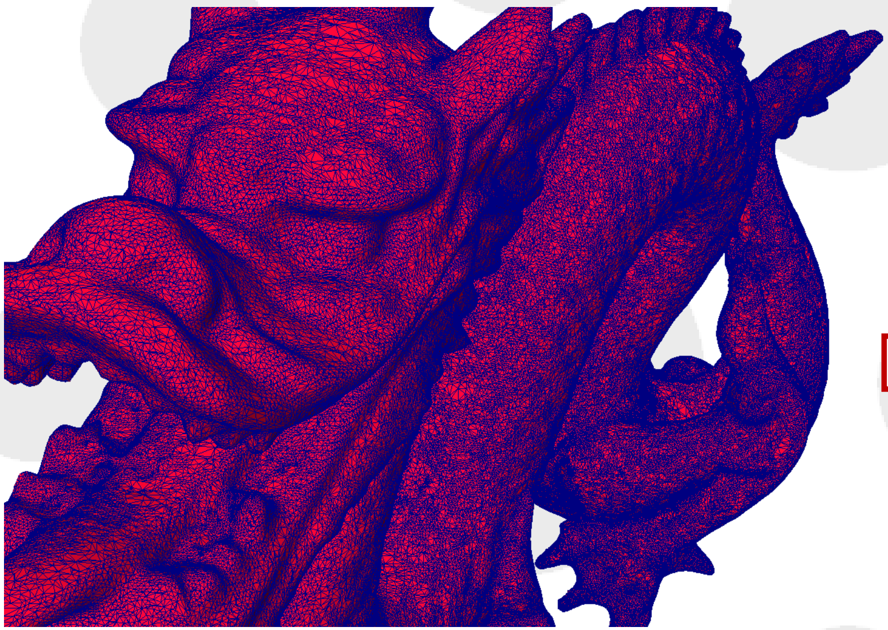
Methodology Overview | Snapping

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

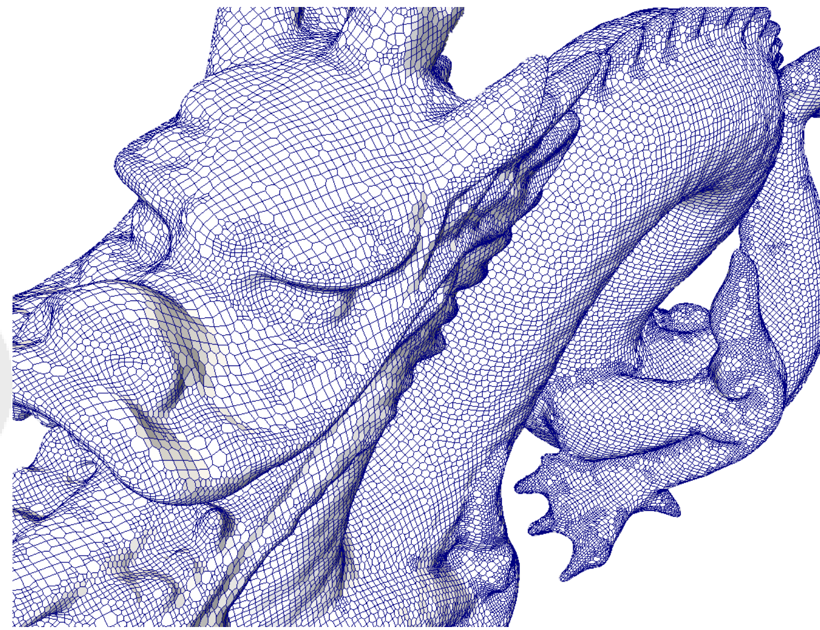


Methodology Overview | Snapping

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



Original STL Surface

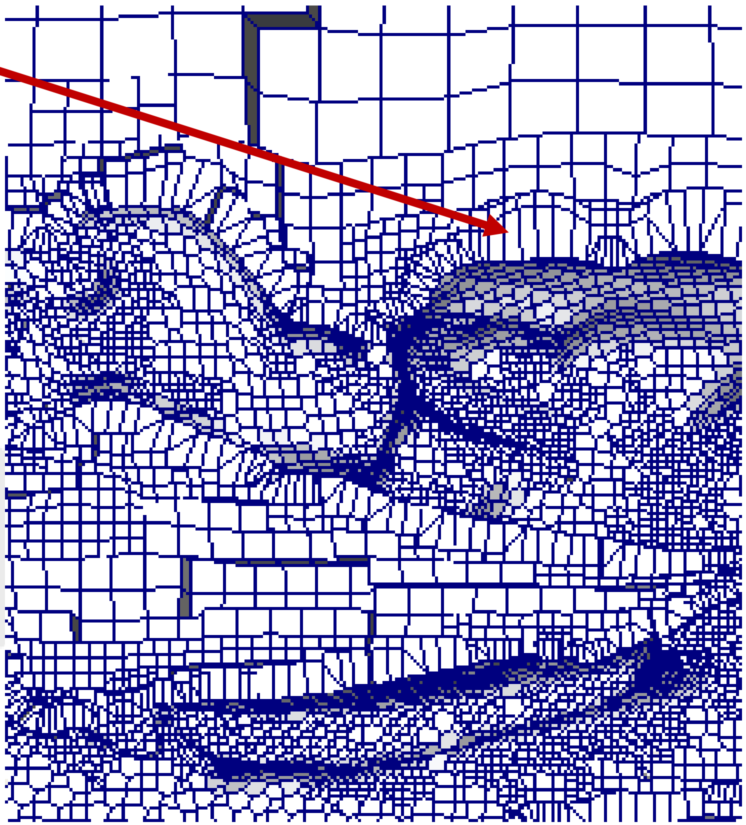


Snapped Surface

engys

Methodology Overview | Layers + Final

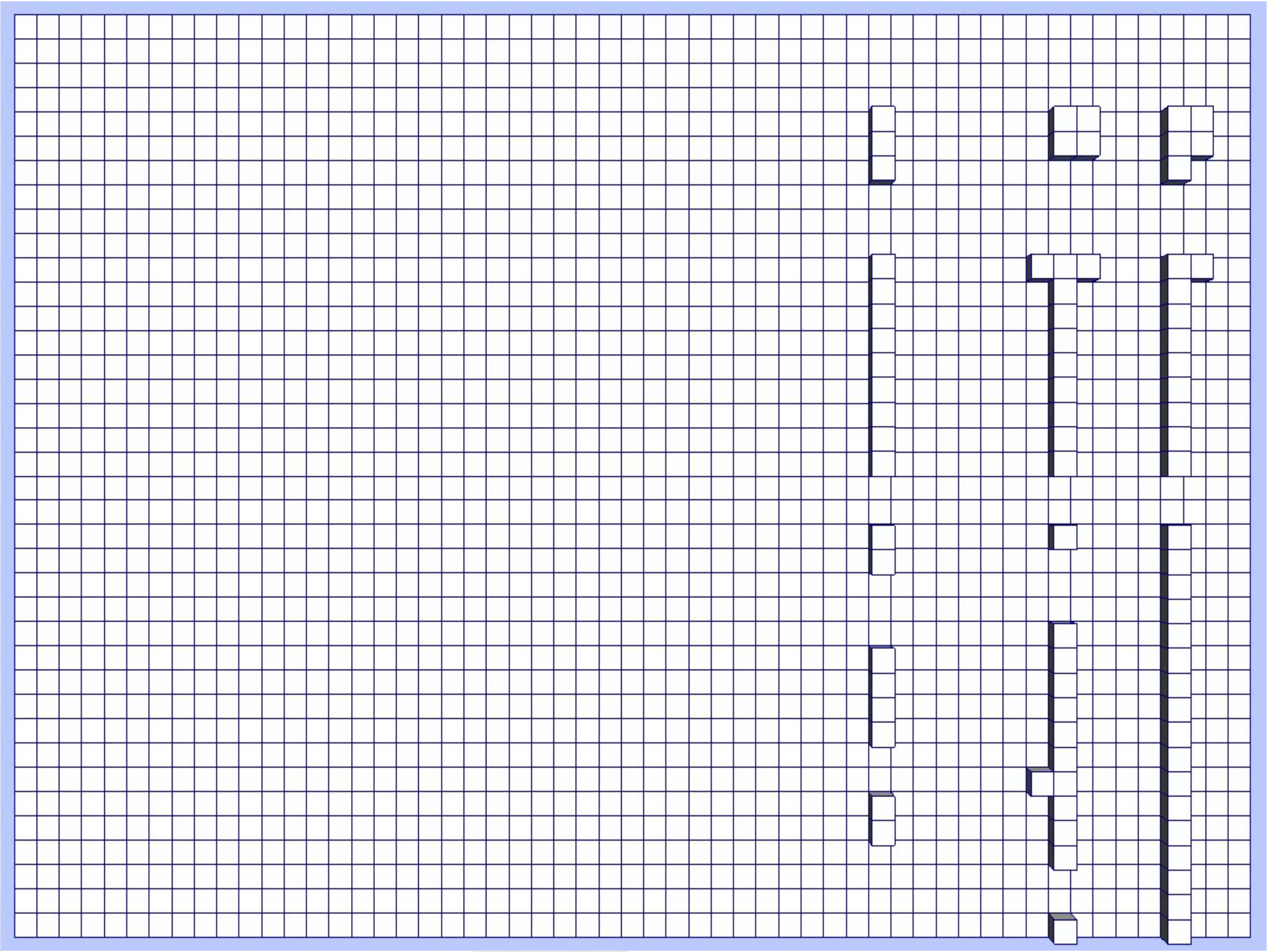
- Step 5: Add layers to the surface
  - Push mesh away from surface
  - Add layers
  - Check quality
  - Scale back displacement if errors
  - Repeat until all quality checks pass



- Step 6: Final load balance
  - Output to file

engys

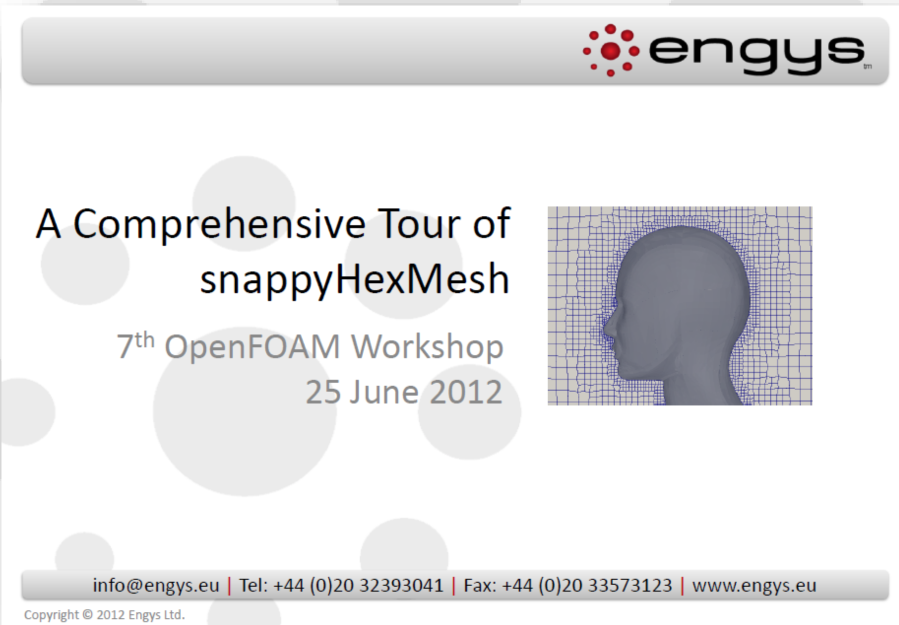
Methodology Overview | Overall

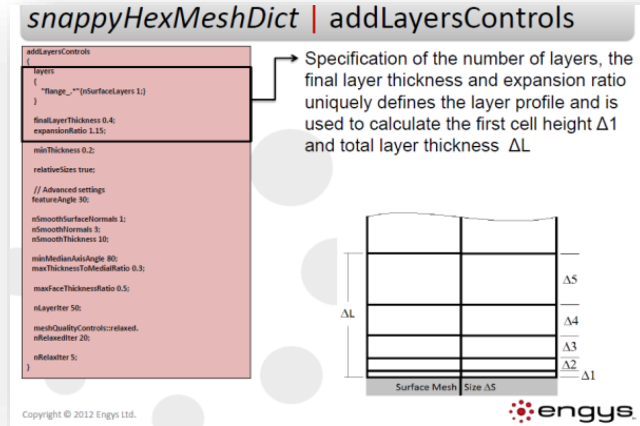


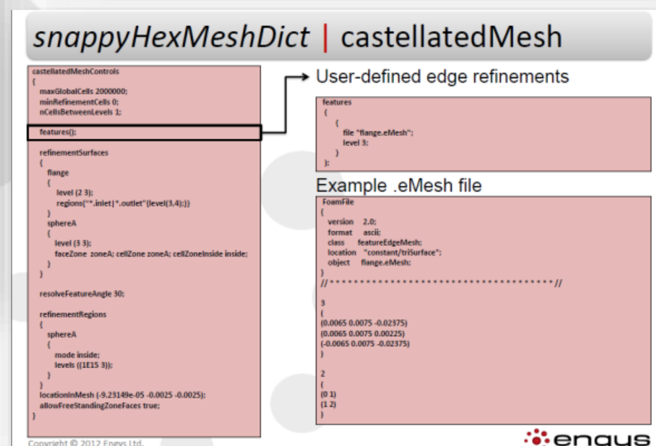
Contents

Manual Setup

Manually specify `system/snappyHexMeshDict`







engys

Contents

- snappyHexMesh
  - Description and Key Features
  - Background, Origin, and Forks
  - Brief Code Overview
  - Methodology Overview
- Manual Setup
- Meshing with HELYX-OS
  - Backward Facing Step
  - Reactor Geometry
  - Tube-Bend
- Closing Remarks

Meshing with HELYX-OS

Overall Goal of Session

- Connect meshing methodology with HELYX-OS controls
- Technology preview of the new version of HELYX-OS

Skills Obtained

✓ Importing STLs
✓ Creating base meshes
✓ Setting surface refinement
✓ Extracting feature lines
✓ Creating volume refinement boxes
✓ Adding layers

engys

Meshing with HELYX-OS

Requirements

✓Installed version of HELYX-OS

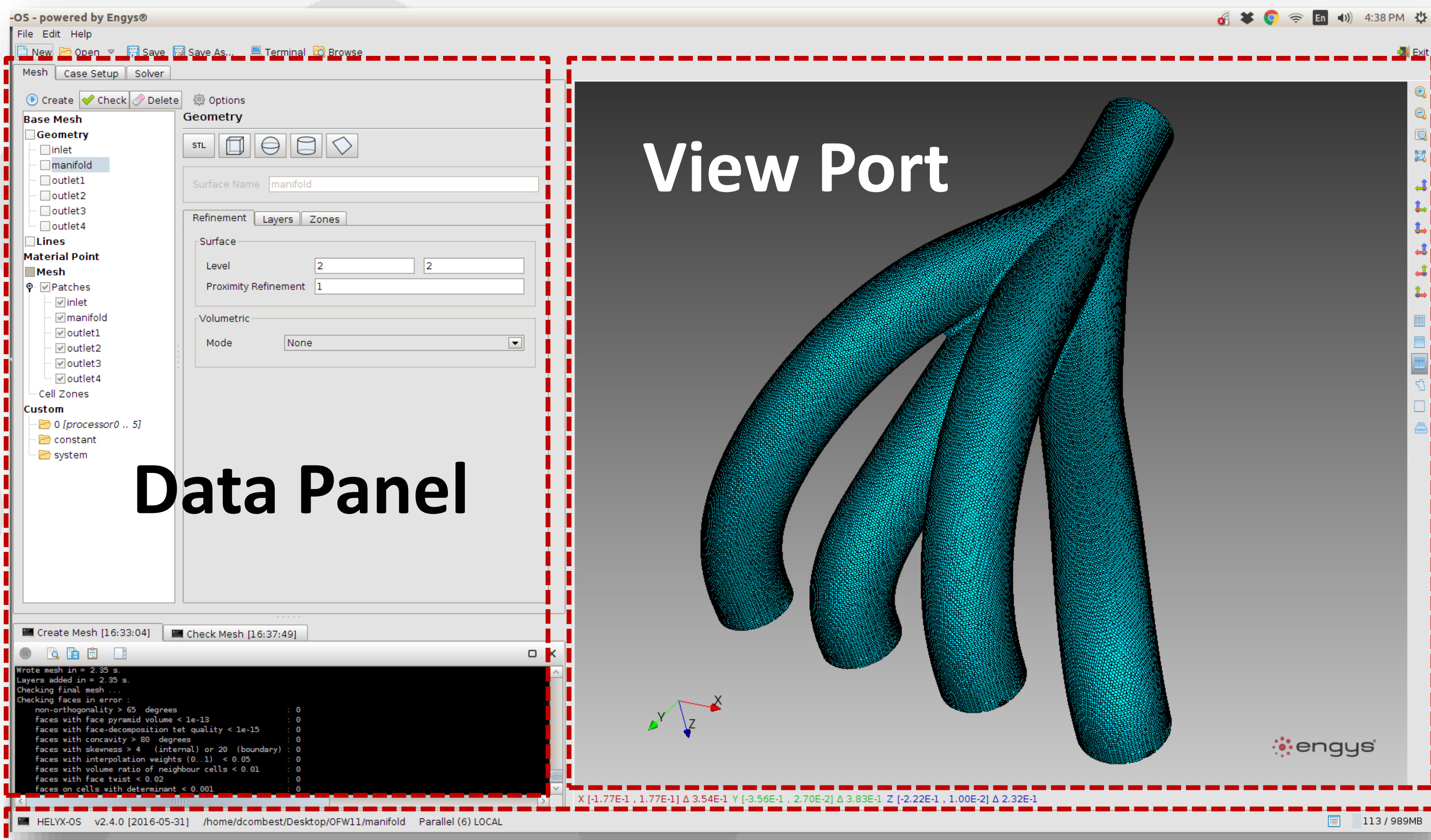
- Technology preview of **HELYX-OS v2.4.0 on workshop image**
- Or **HELYX-OS v 2.3.1** http://engys.github.io/HELYX-OS/

✓Minimum of 1 core, suggested multiple cores

✓Download the geometry files from the workshop website

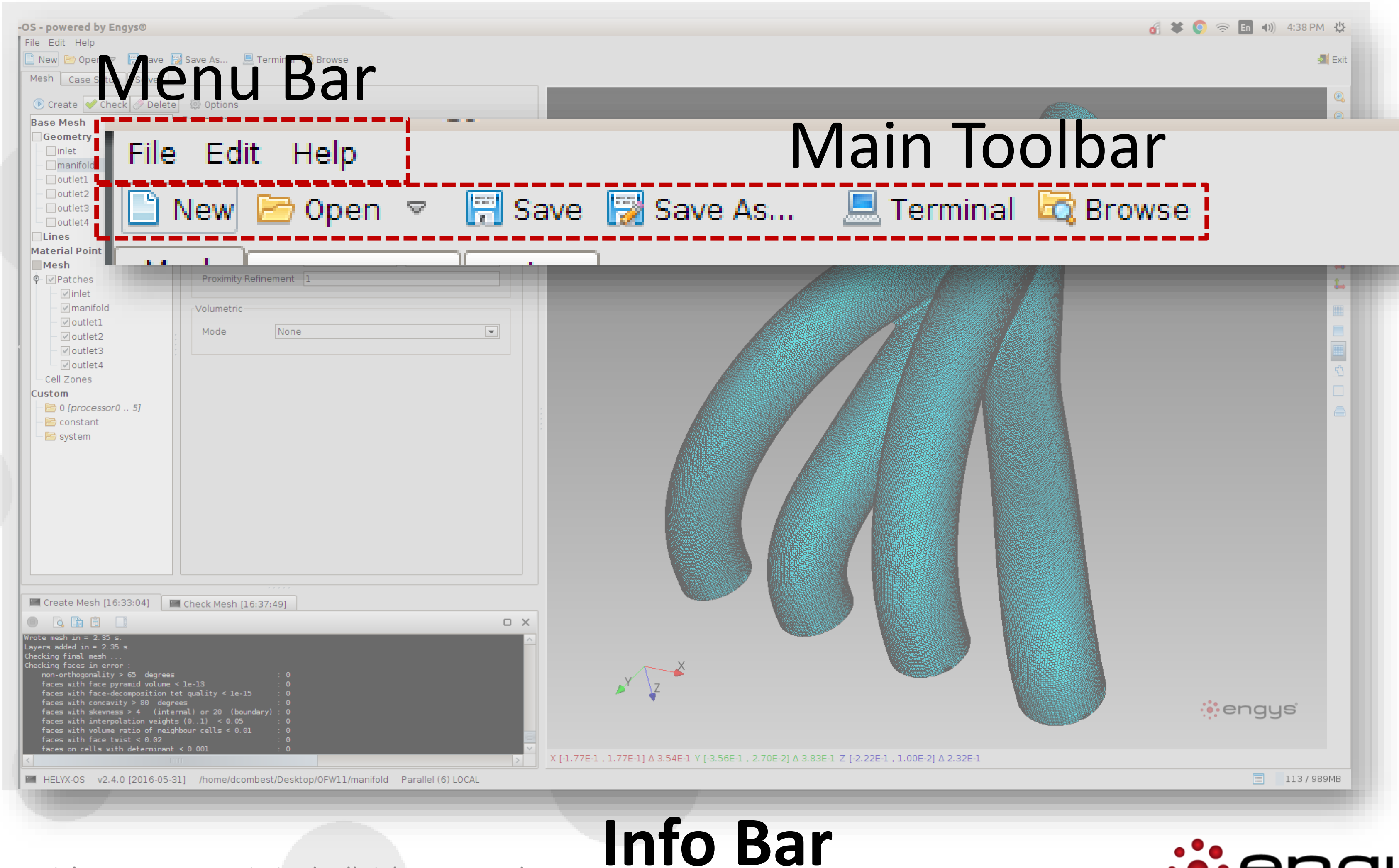
Meshing with HELYX-OS

The Interface



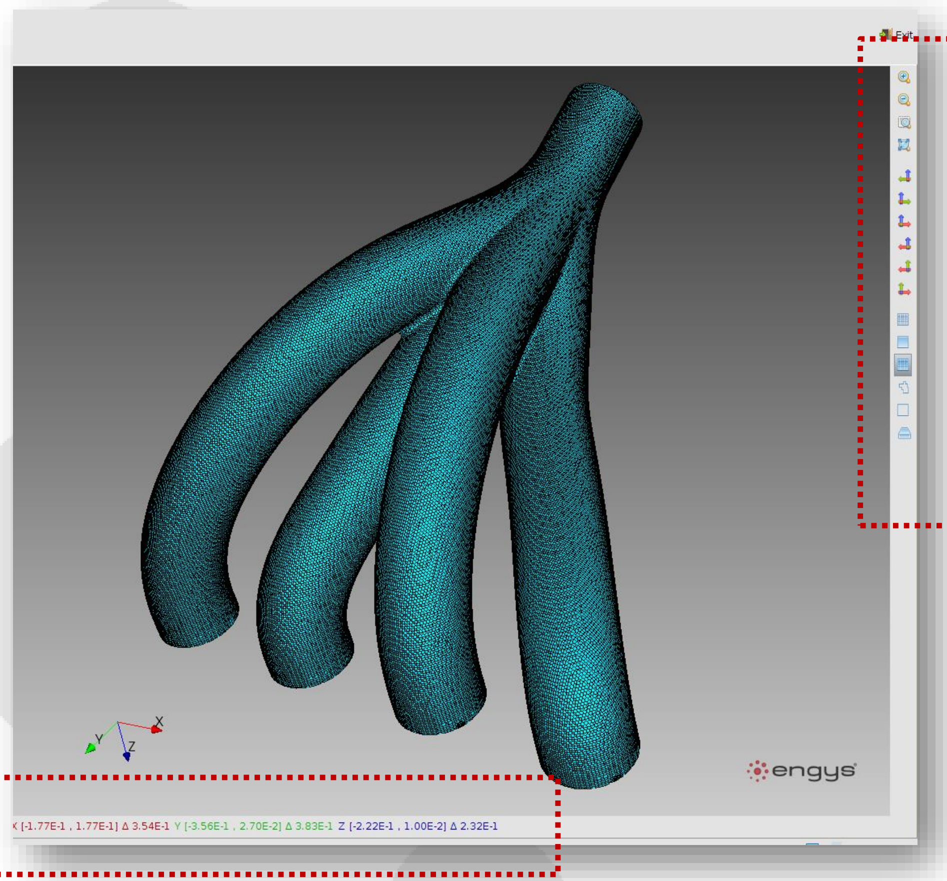
Meshing with HELYX-OS

The Interface

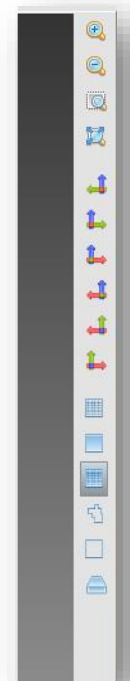


Meshing with HELYX-OS

The Interface



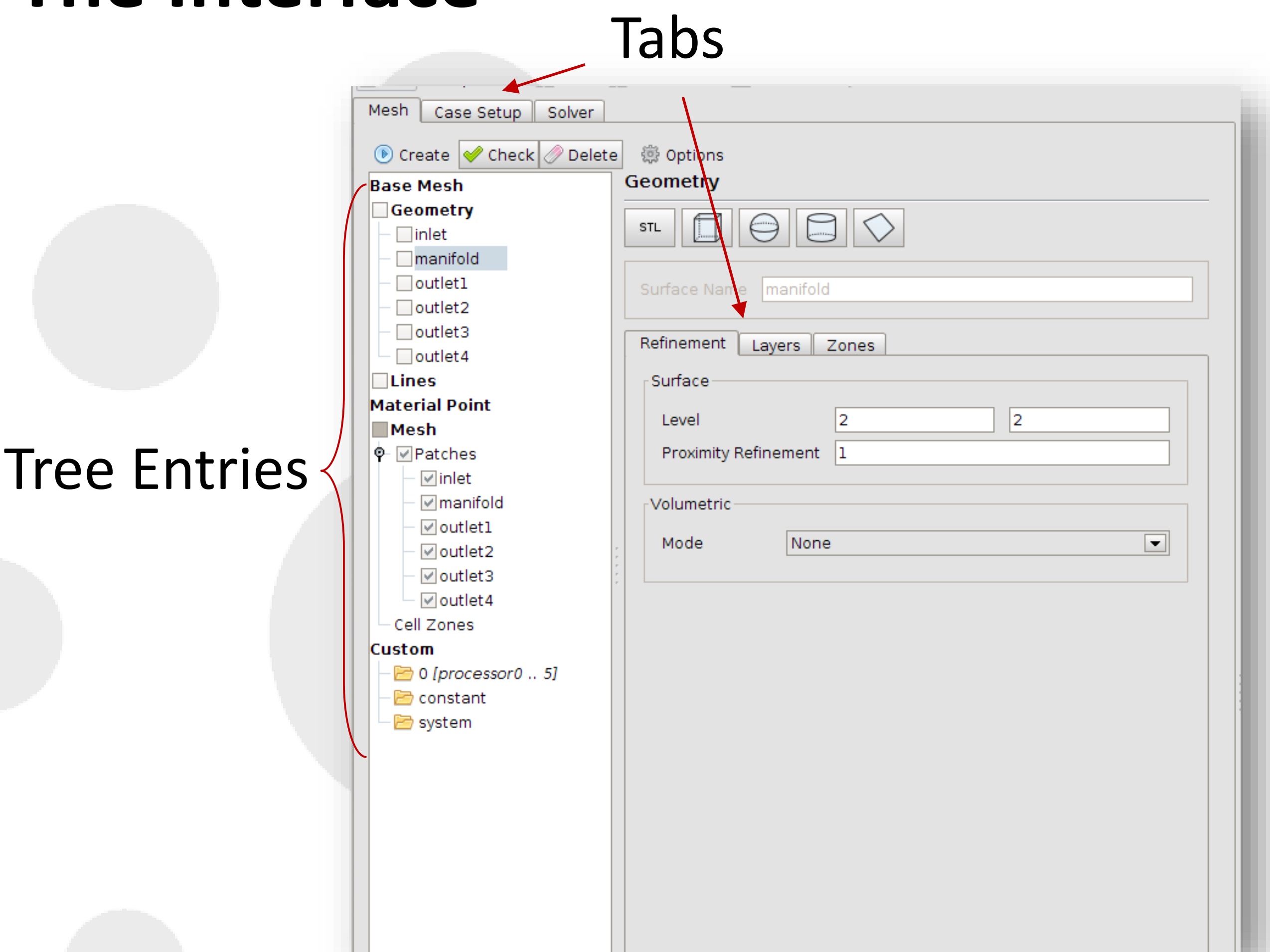
Domain/ Geometry Bounds



View Controls

Meshing with HELYX-OS

The Interface



Start on the meshing tab and work our way down the tree

Backward Facing Step | Usage Intro

Overall Goal of Session

- Introduce controls of HELYX-OS

Skills Obtained

✓ Creating base meshes

✓ Setting surface refinement

✓ Extracting feature lines

✓ Creating volume refinement boxes

Backward Facing Step | Usage Intro

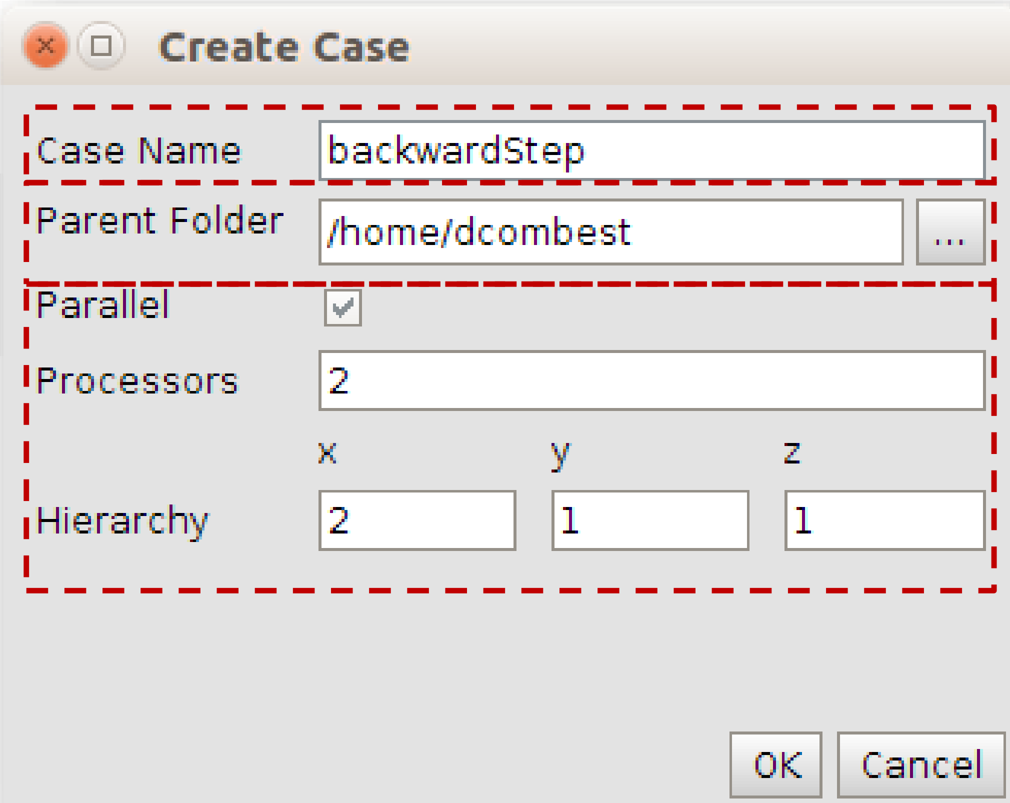
- New
- Open
- Exit

Recent Cases

Starting Screen

Backward Facing Step | Usage Intro

Case Creation

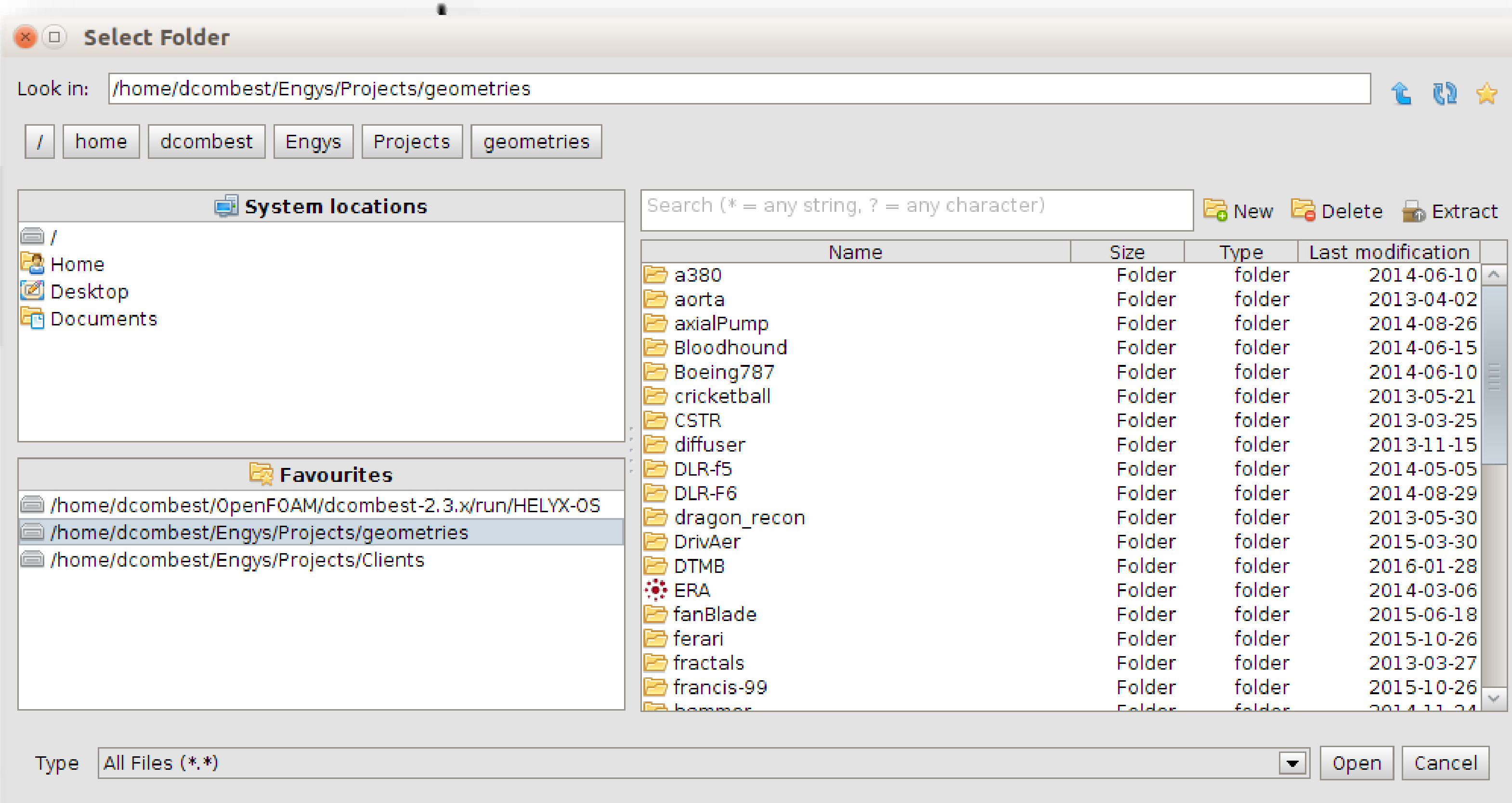


Case Name

Parent Folder

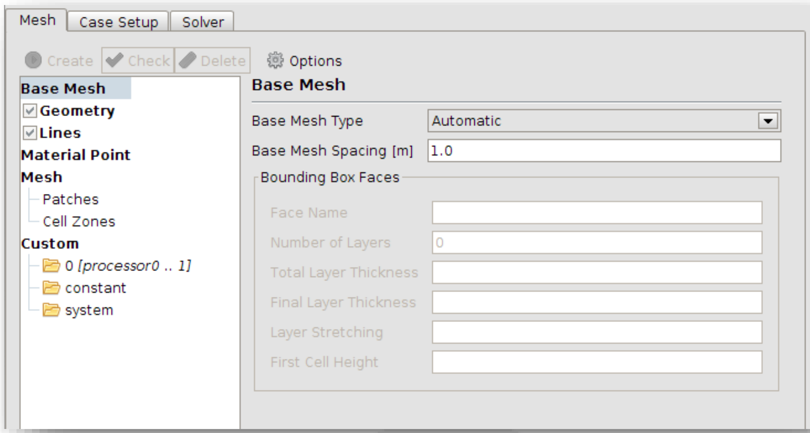
Parallel Settings

Backward Facing Step | Usage Intro



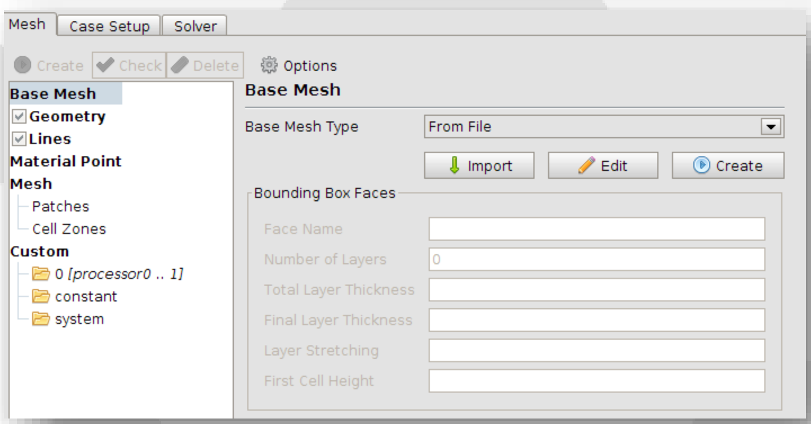
Backward Facing Step | Usage Intro

Defining a Base Mesh



Automatic

- Creates background mesh
- Used for close geometries



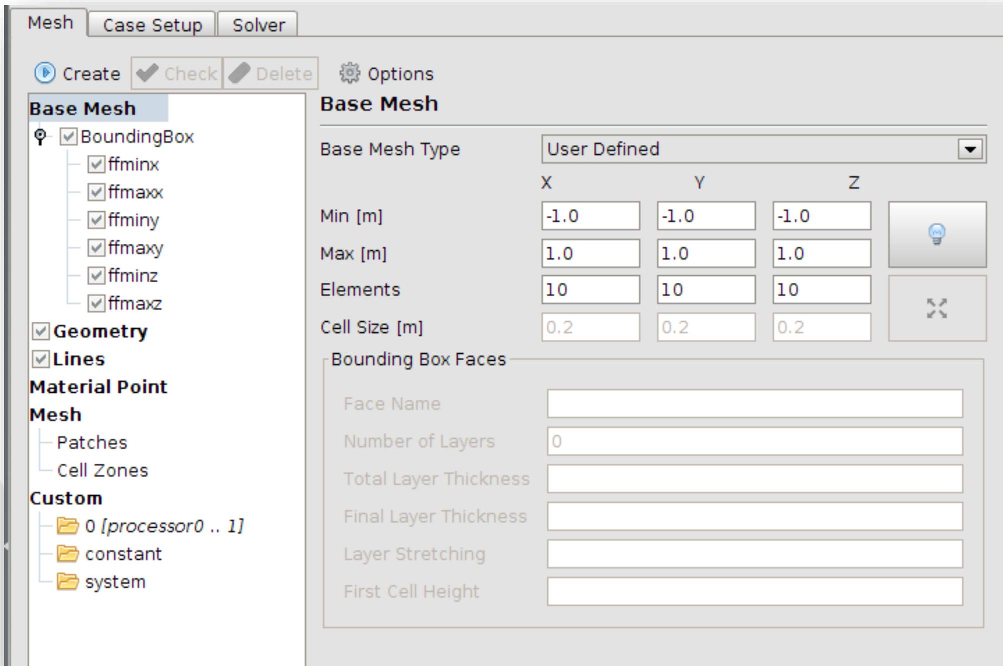
From File

- Lets you to create, import, edit existing `blockMeshDict` files

engys

Backward Facing Step | Usage Intro

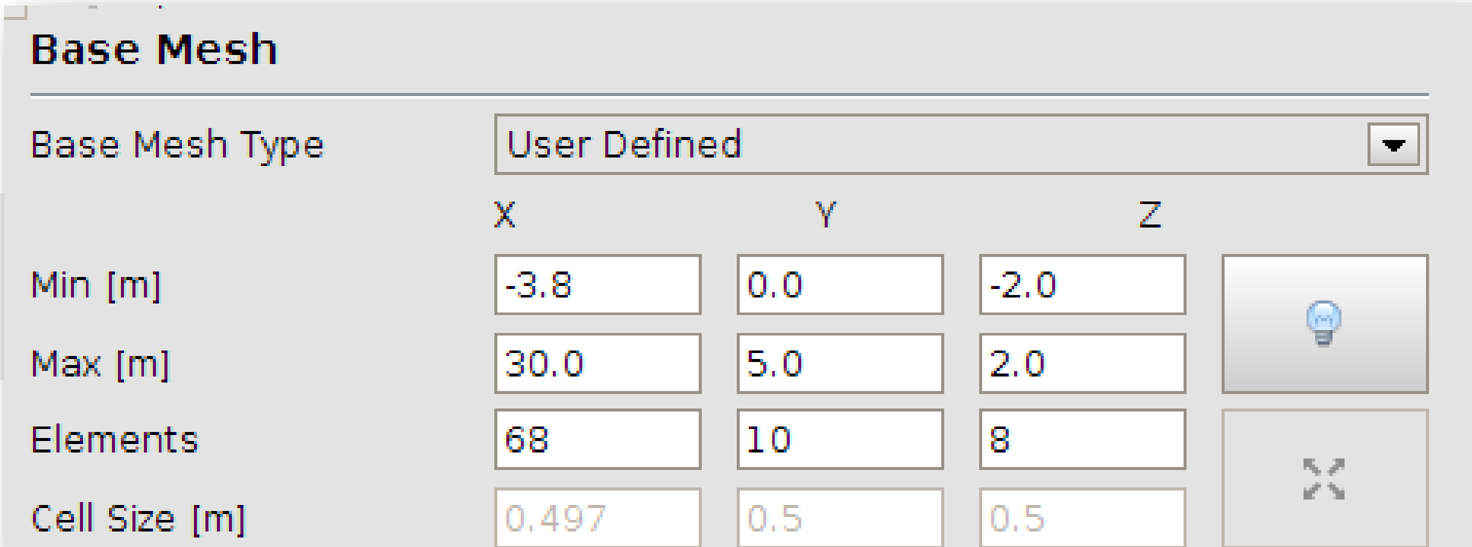
Defining a Base Mesh



User Defined

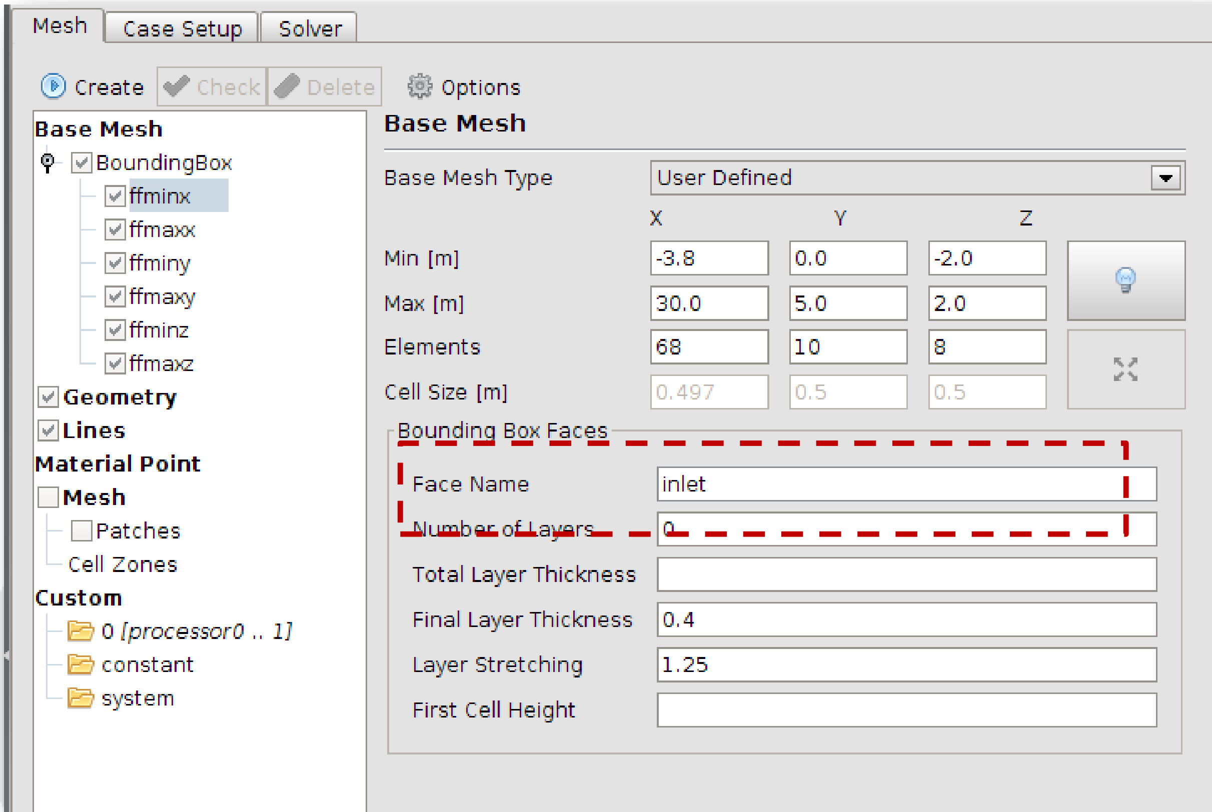
- User provides min and max bounds of x,y,z
- User provides number of elements in each direction

Defining a Base Mesh



1. Choose "User Defined"
2. Configure a mesh with these settings

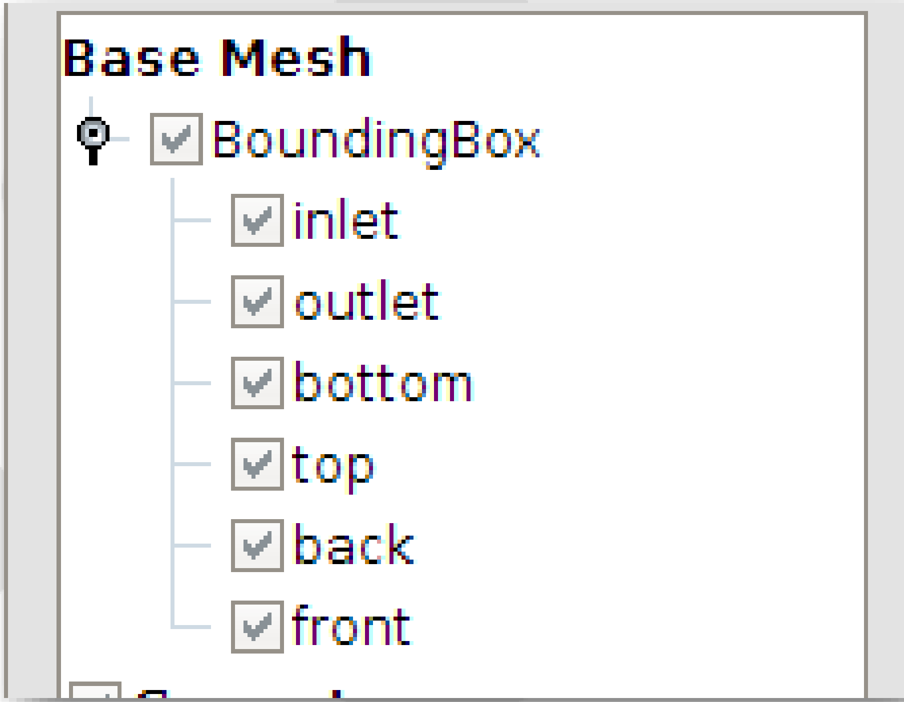
Backward Facing Step | Usage Intro



3. Click on "ffminx" in the tree
4. Change the Face Name to **inlet**

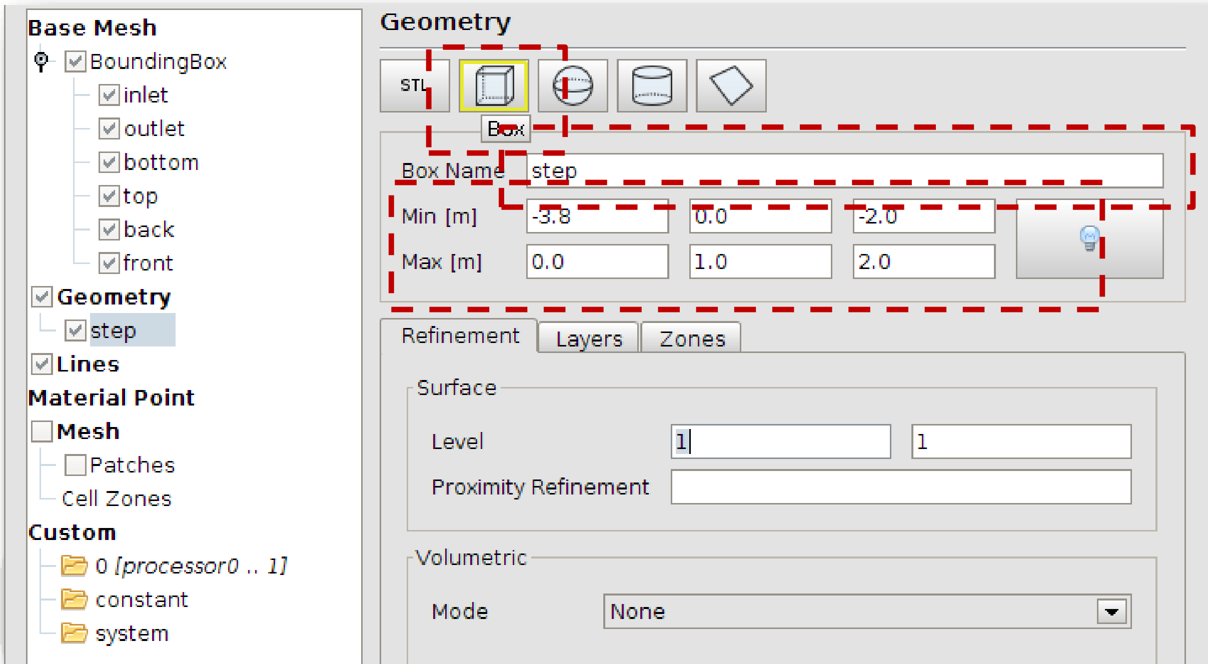
Backward Facing Step | Usage Intro

Defining a Base Mesh



4. Repeat for each of the following

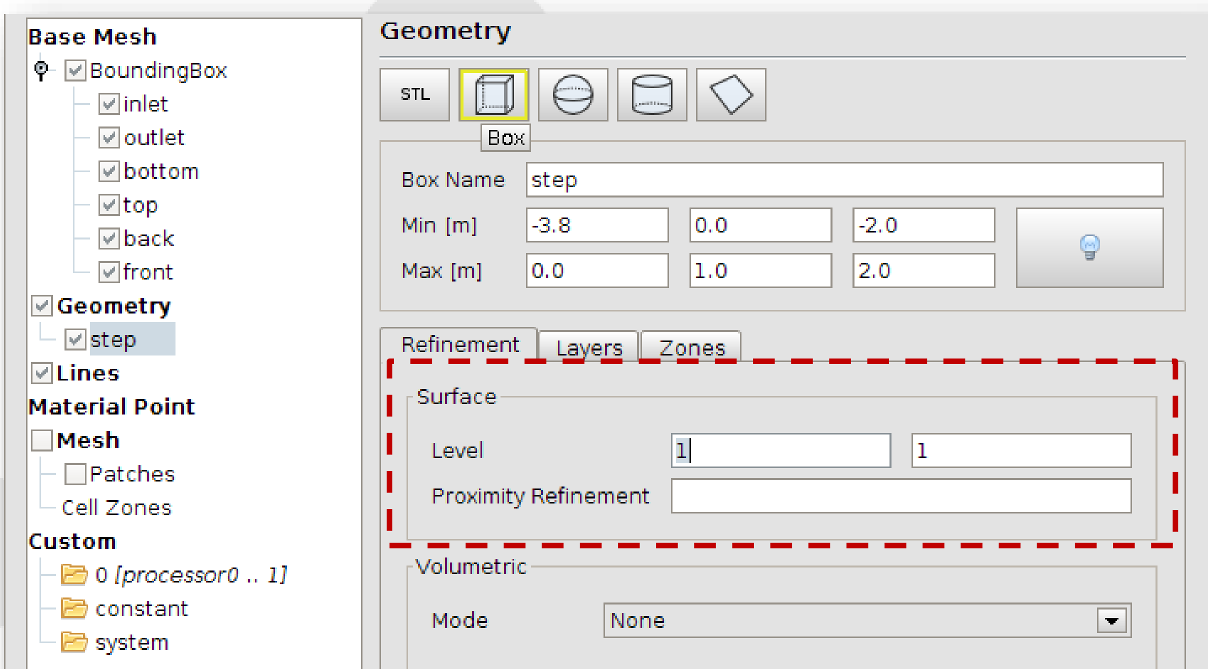
Defining a Box Primitive Object

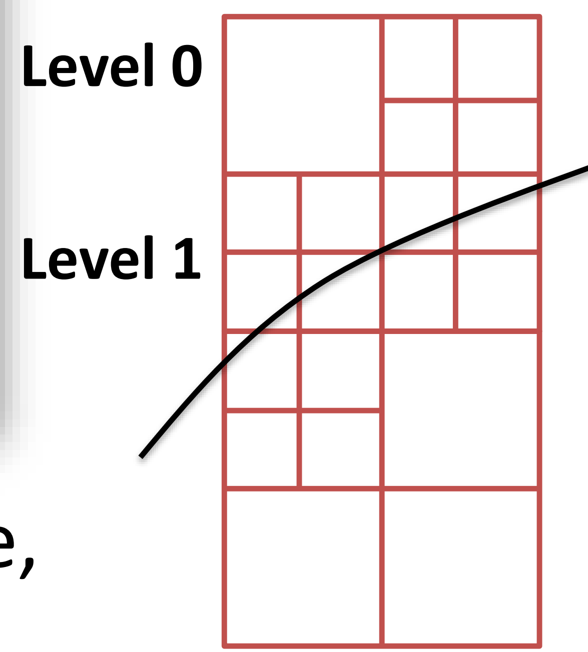


5. In the geometry tree entry add a box primitive object named **step**
6. Set with above min and max

Backward Facing Step | Usage Intro

Defining Surface Refinement



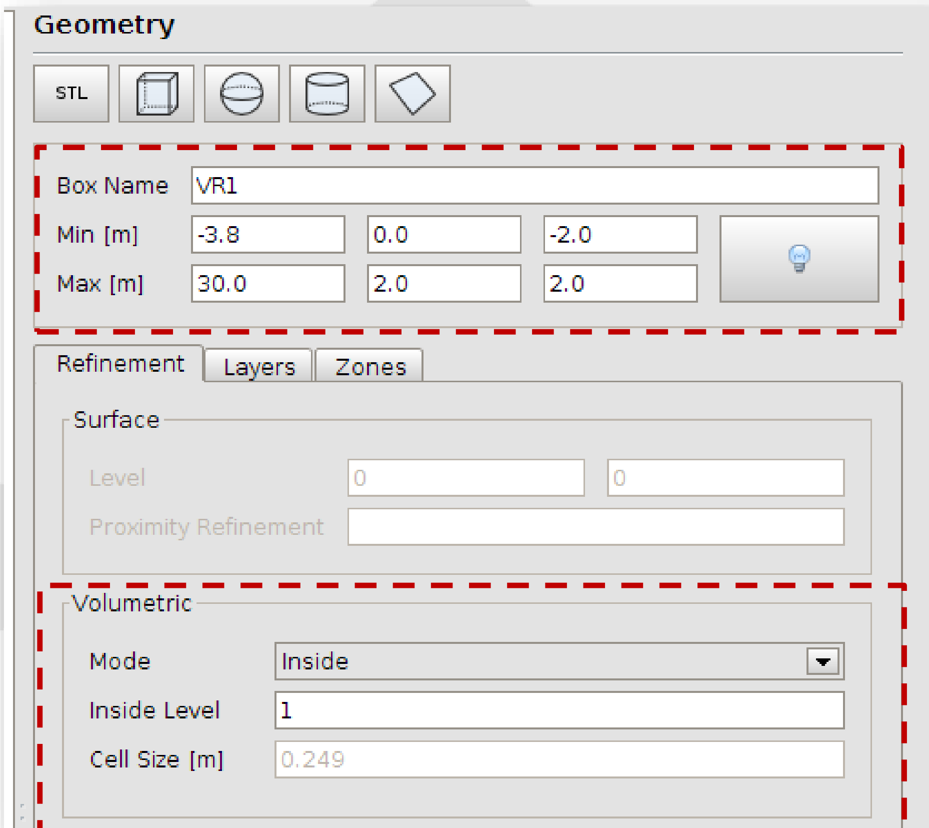


7. In refinement tab, under surface, set min and max levels to 1

engys

Backward Facing Step | Usage Intro

Volume Refinement

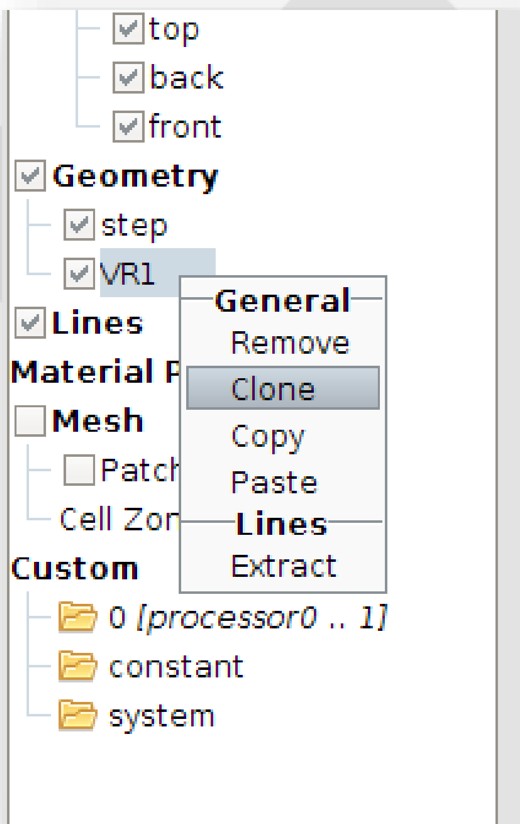


8. Create a box primitive object
9. Named VR1
10. Set a volumetric refinement mode **inside**
11. Set a refinement **level of 1**

engys

Backward Facing Step | Usage Intro

Volume Refinement

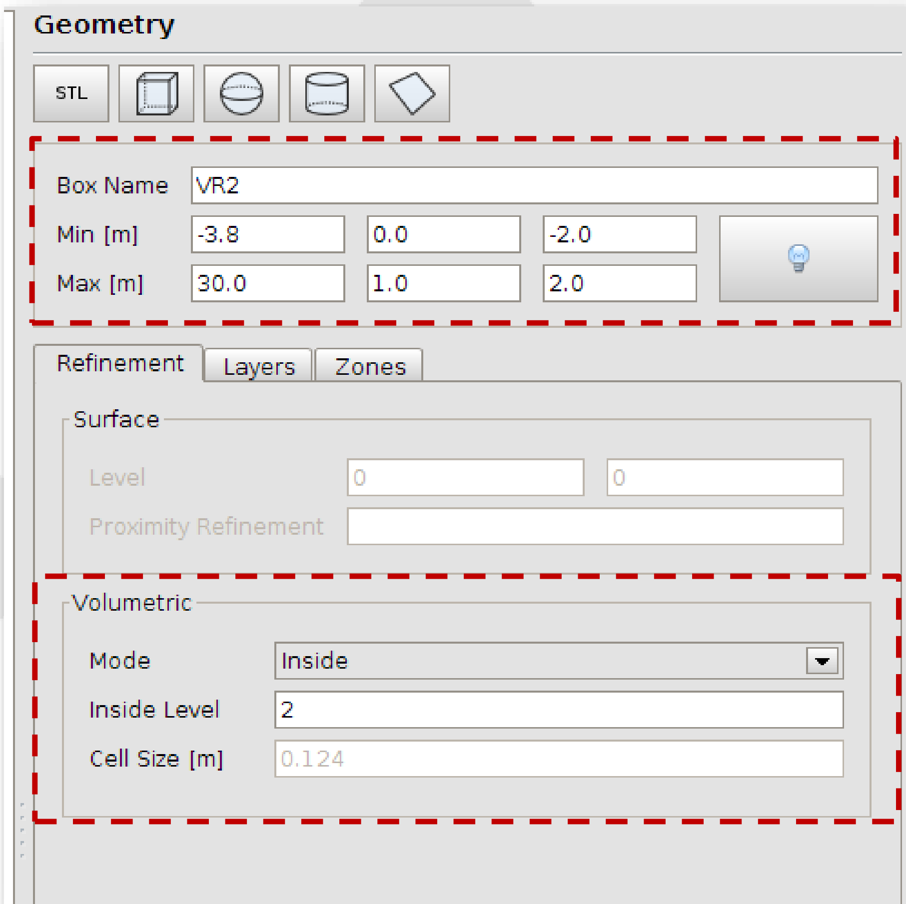


12. Clone VR1 by RMB and select clone

engys

Backward Facing Step | Usage Intro

Volume Refinement

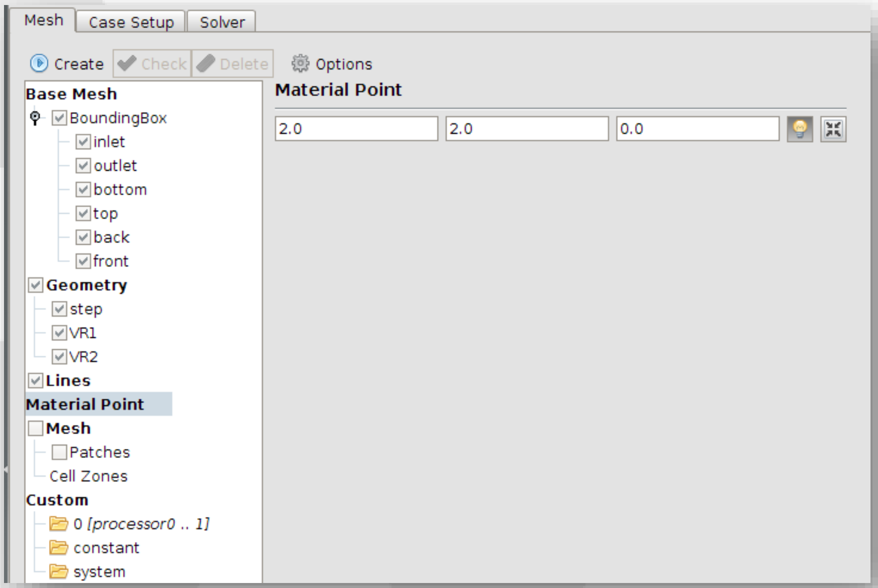


13. Create a box primitive object
14. Named VR2
15. Set a volumetric refinement mode **inside**
16. Set a refinement **level of 2**

engys

Backward Facing Step | Usage Intro

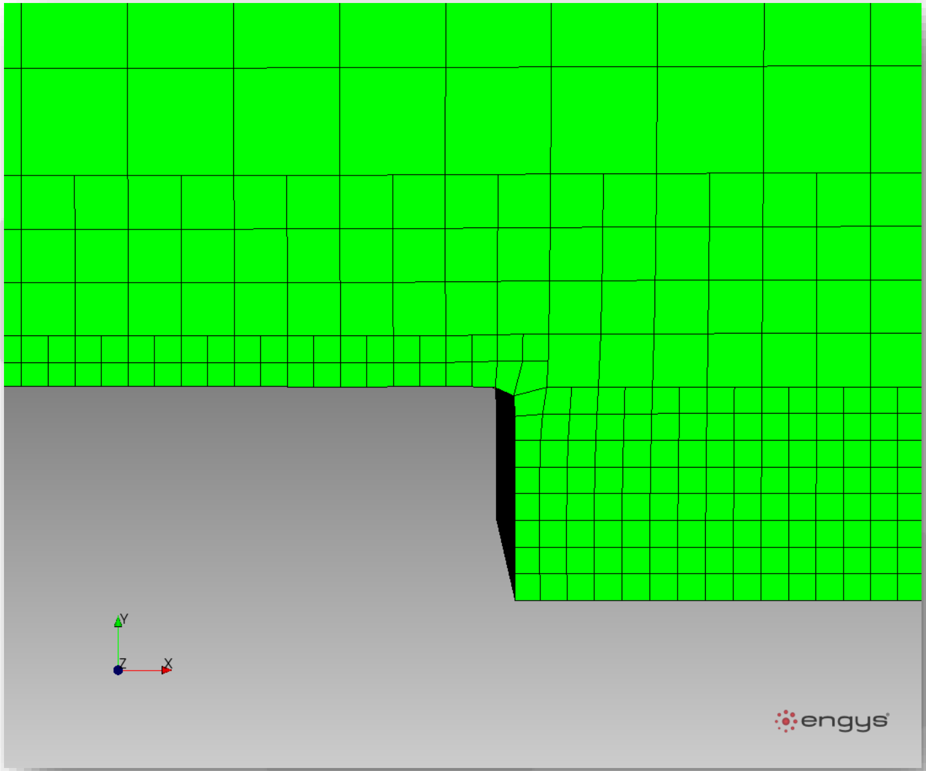
Set the Material Point



17. Select a material point inside the domain **(2,2,0)**
18. Select the **create** button

engys

Backward Facing Step | Usage Intro

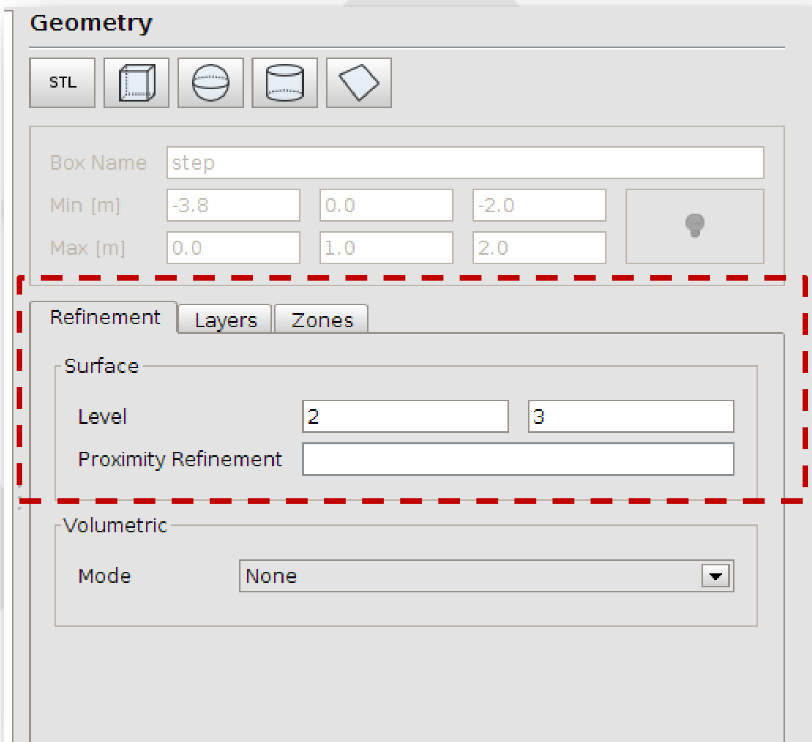


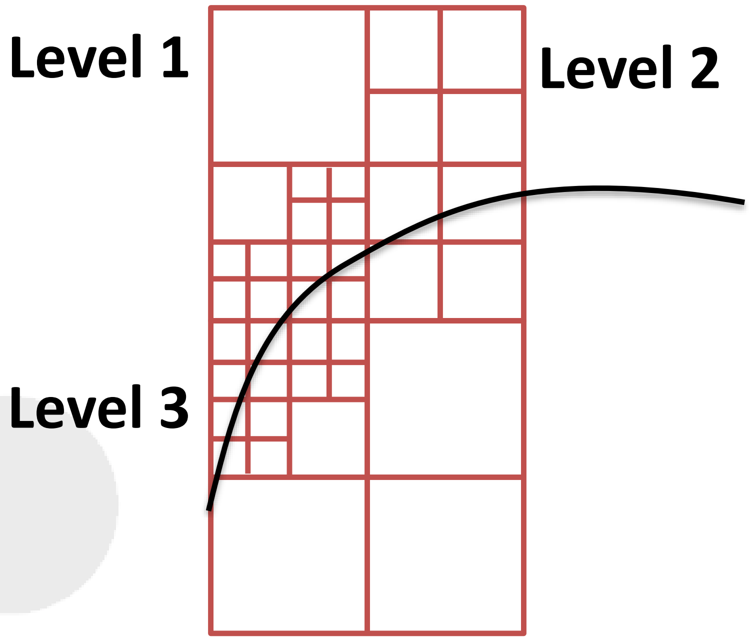
- How do we fix the issue of snapping at the top of the step?
- Increase refinement?

engys

Backward Facing Step | Usage Intro

Surface Refinement

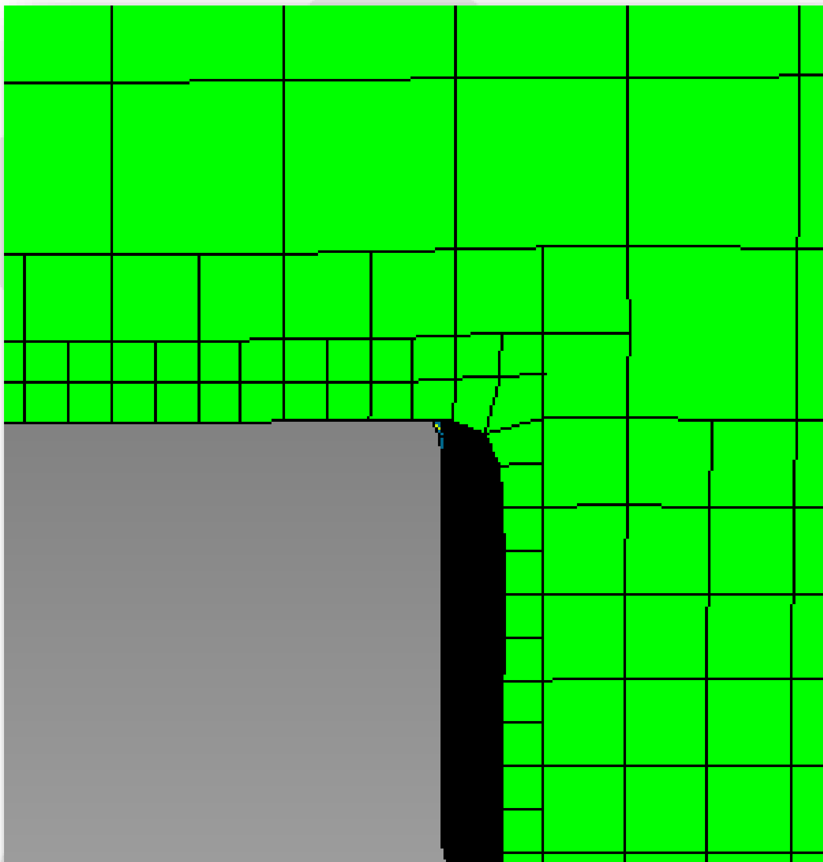




We could increase the surface refinement levels

engys

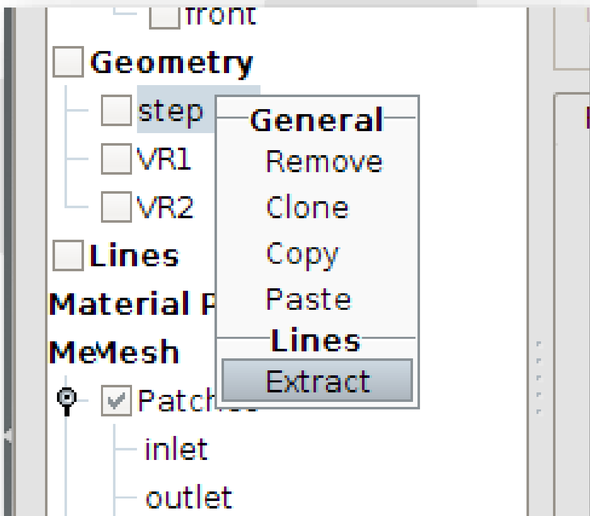
Backward Facing Step | Usage Intro

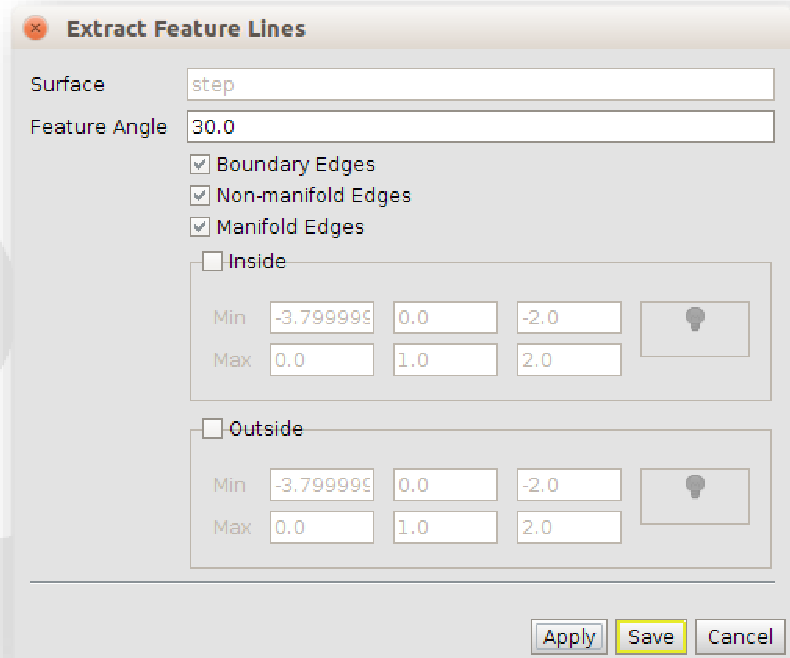


- This will increase meshing time and still not solve our issue in this case

Backward Facing Step | Usage Intro

Extract Feature Lines

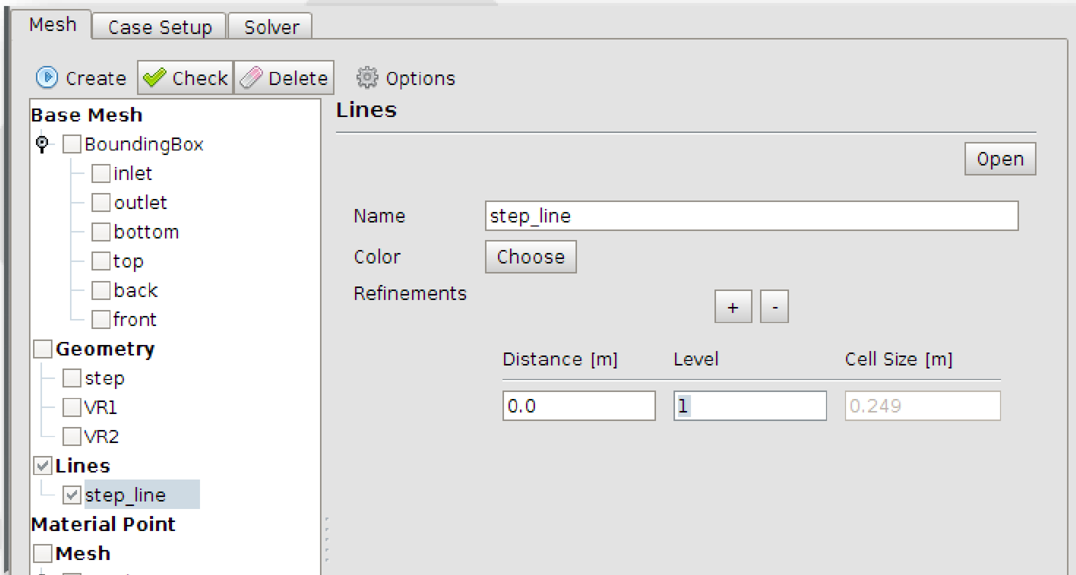




19. In the step, RMB and select extract
20. Select **apply and then** save in the extract feature lines tool

Backward Facing Step | Usage Intro

Extract Feature Lines

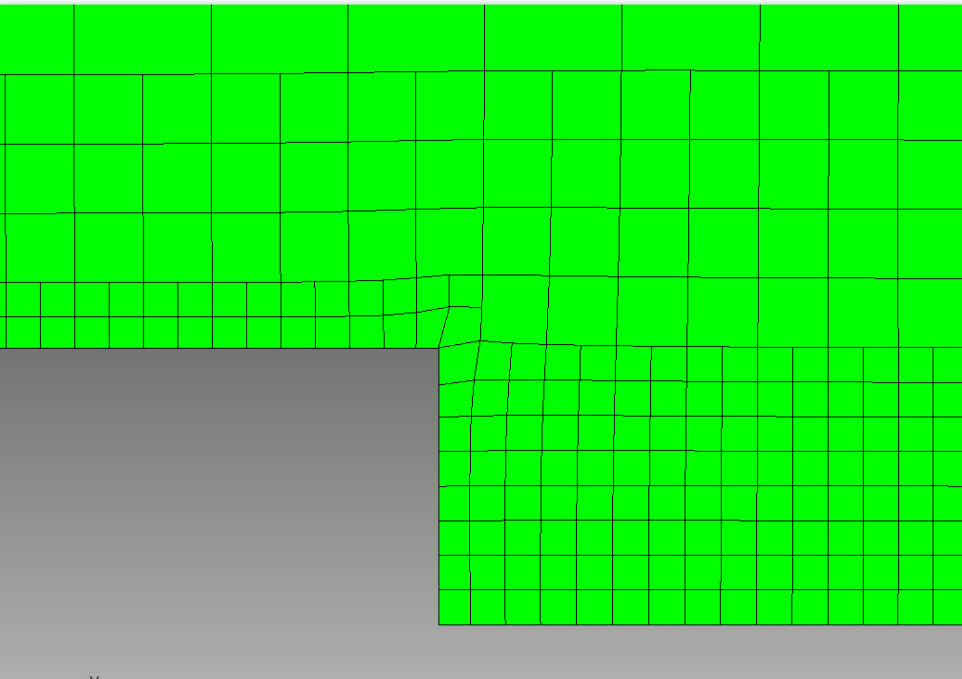


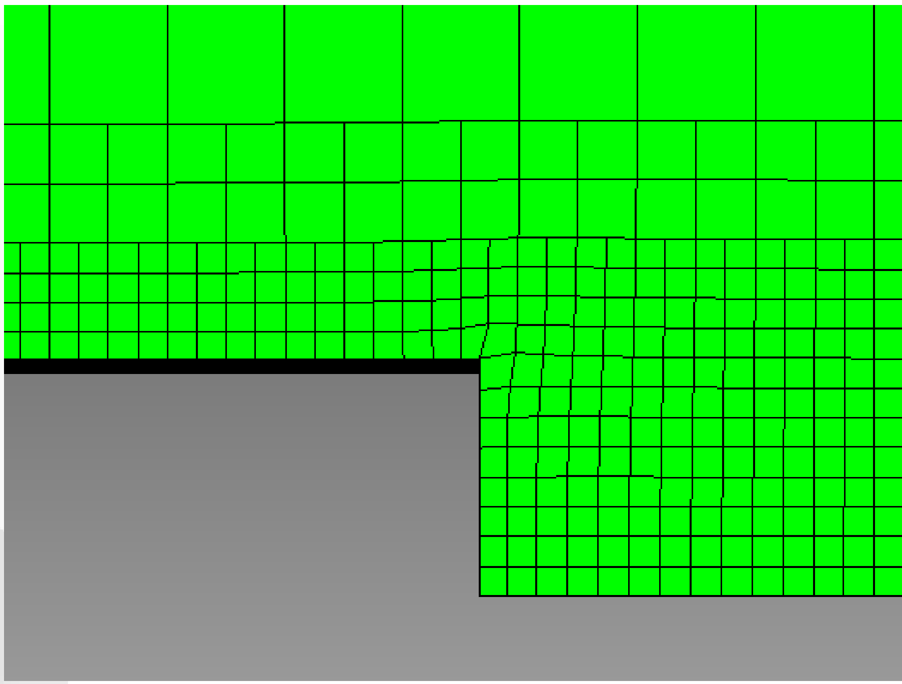
Note

As distance is increased, decrease levels

20. Set a **distance** of **0** and a **level** of **1**

Backward Facing Step | Usage Intro





- Snapping is improved
- Can further move up VR* boxes to refine more

engys

Backward Facing Step | Usage Intro

- Background meshes can be made
  - Automatically, with only a cell size
  - By defining as a simple bounding box and # of elements
  - Provided as a blockMeshDict file for complex base meshes
- We can build simple domains with primitive object
- Extracting feature lines can improve snapping as well as increasing surface refinement

engys

Straight Pipe | Adding Layers

Overall Goal of Session

- Basics of layer addition

Skills Obtained

✓Creating base meshes

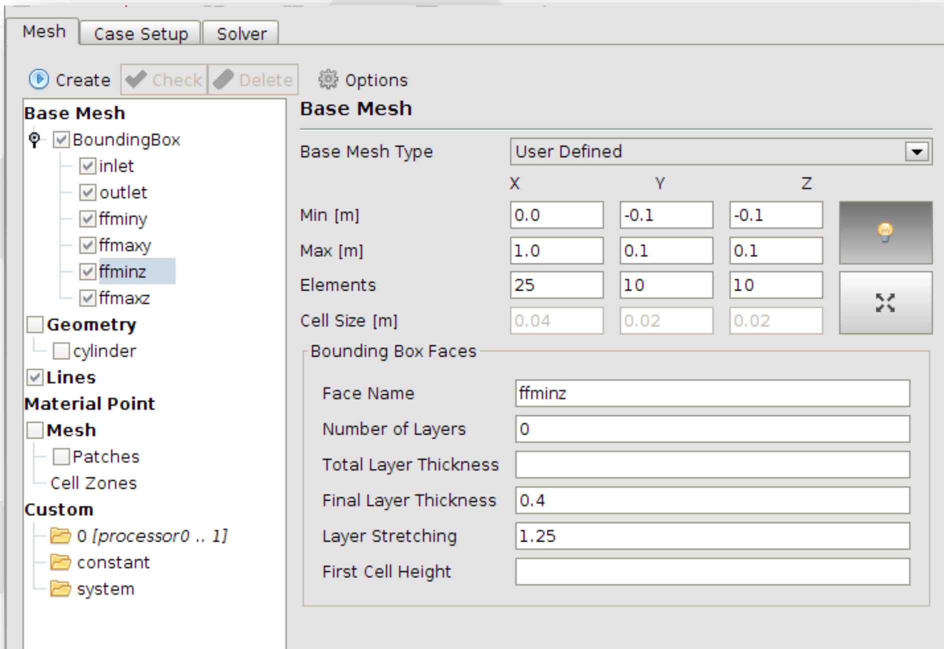
✓Setting surface refinement

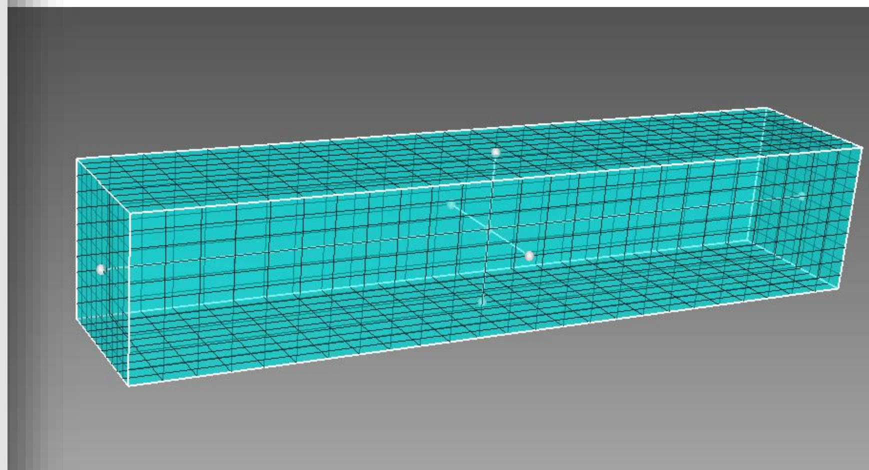
✓Adjusting layer controls and perform a parametric study

engys

Straight Pipe | Adding Layers

Define a Base Mesh

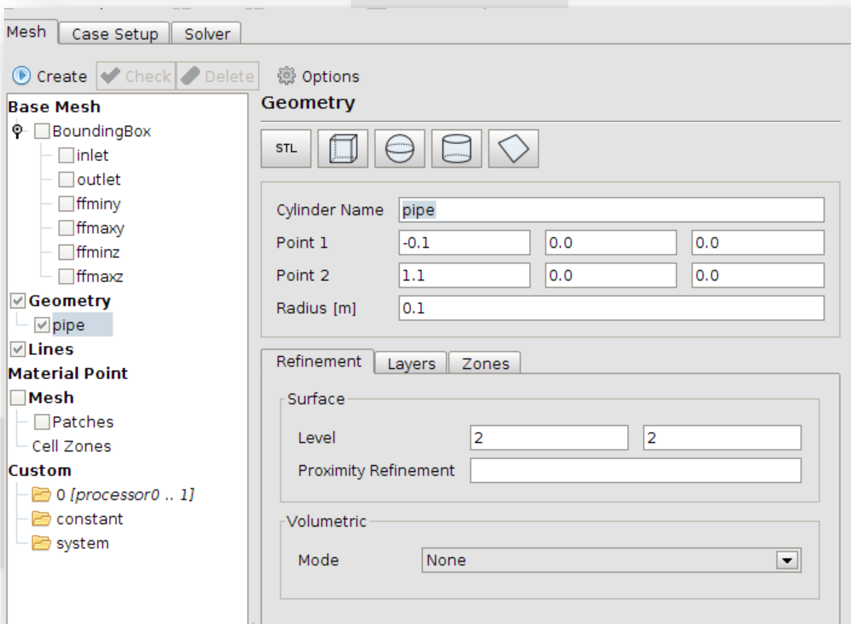


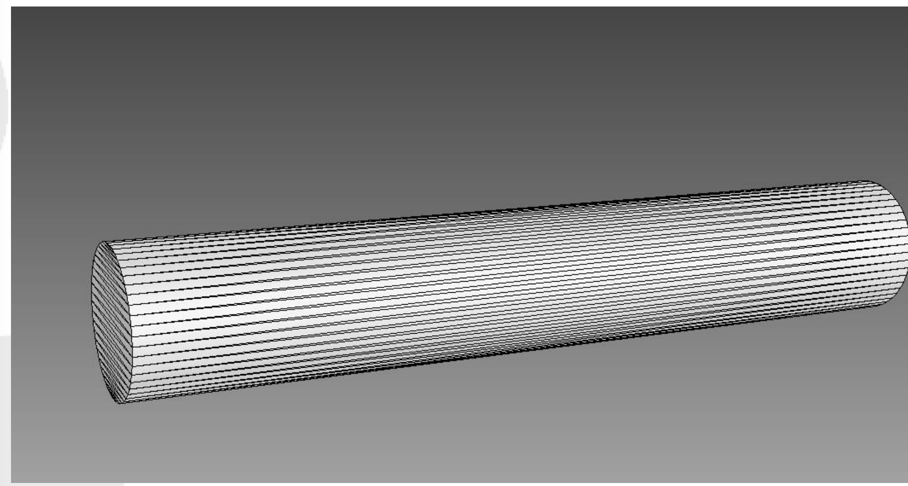


1. Create a **user defined** based mesh
2. Change the **ffminx** and **ffmaxx** to inlet and outlet

Straight Pipe | Adding Layers

Define a Cylinder Primitive Object

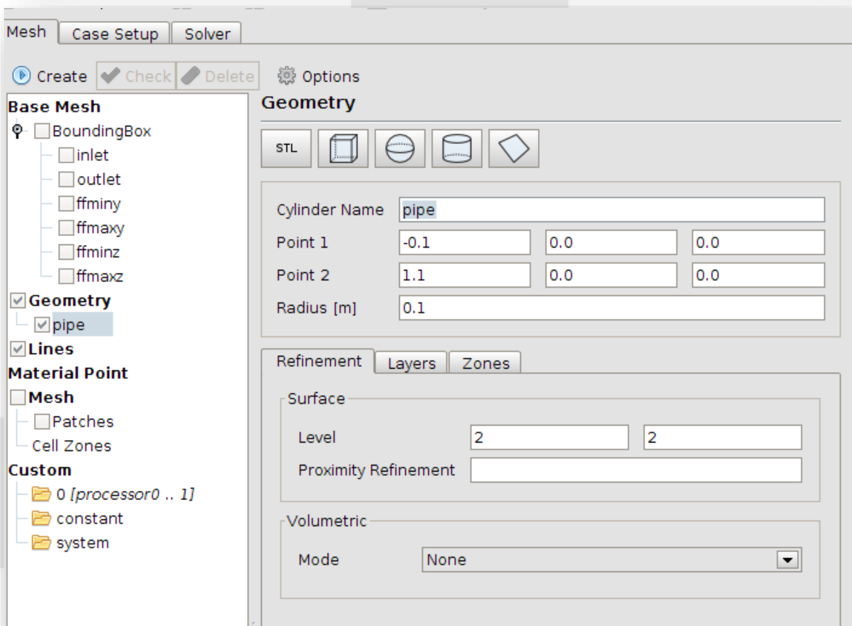


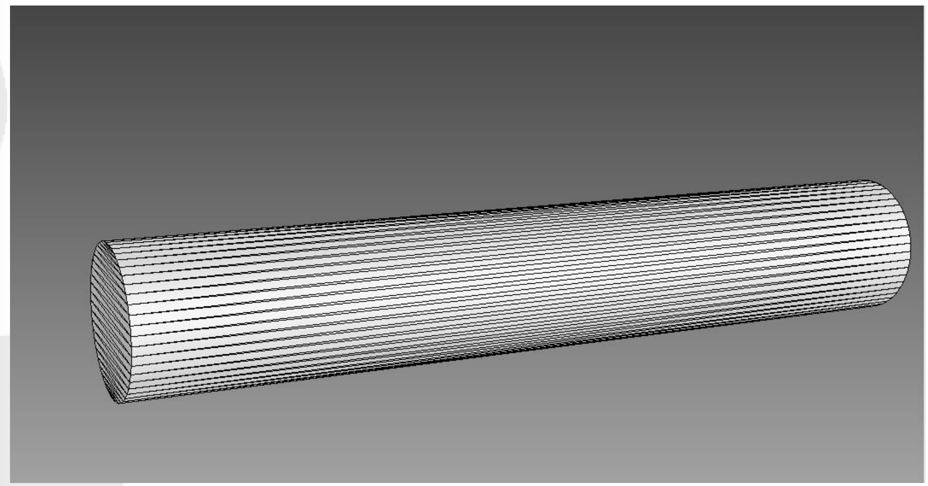


3. Define a cylinder primitive object named pipe

Straight Pipe | Adding Layers

Set the Surface Refinement

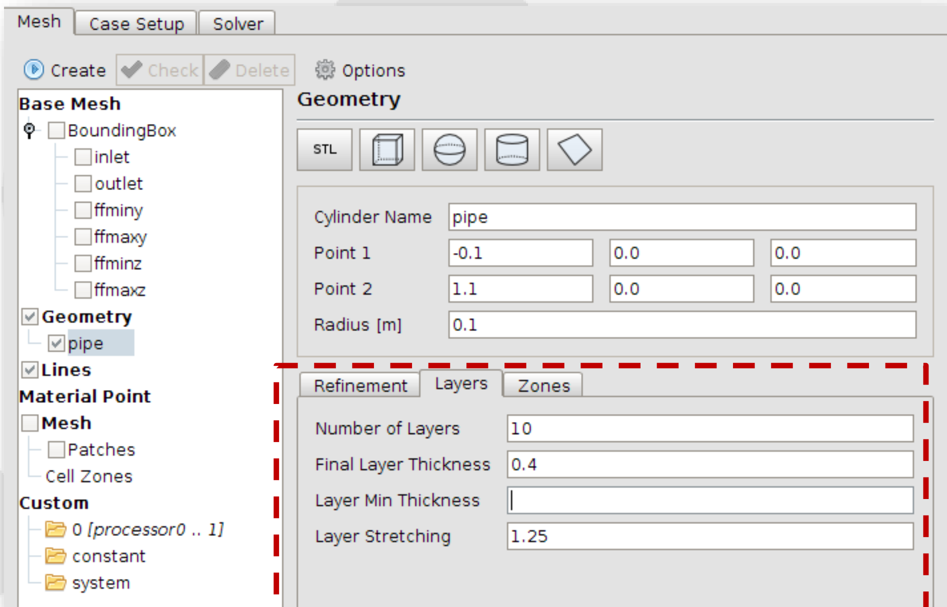




4. Set the surface refinement for the pipe as **2,2**

Straight Pipe | Adding Layers

Adjust the Layer Settings



Required

Number of Layers

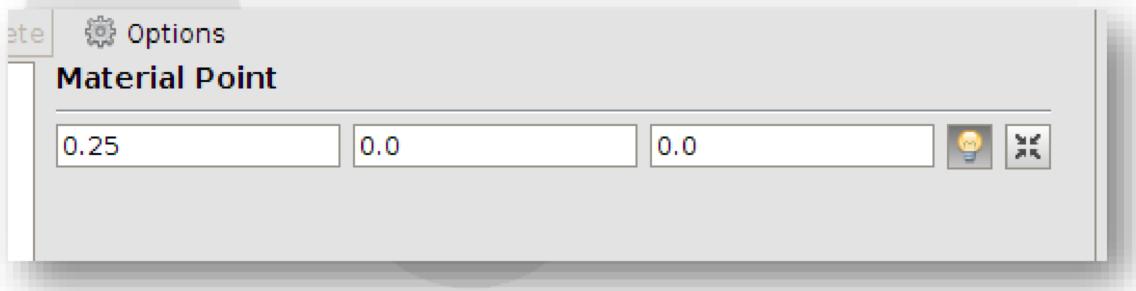
- first and expansion
- final and expansion

5. Set the number of **layers** to **10** and leave defaults

engys

Straight Pipe | Adding Layers

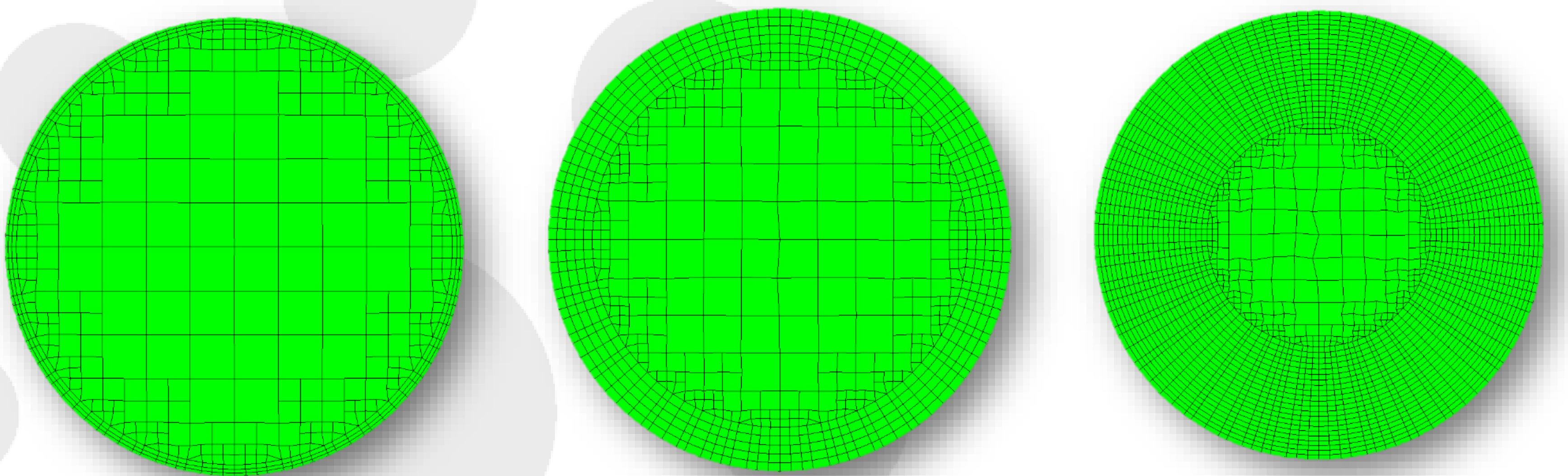
Set the Material Point



6. Set the material point to **(0.25 0 0)**
7. Hit **create**

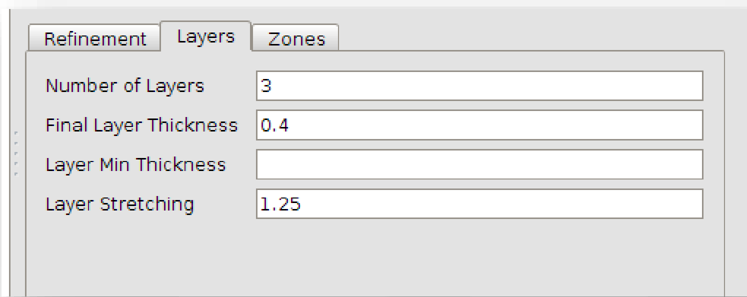
Straight Pipe | Adding Layers

Layer Settings Parameter Study

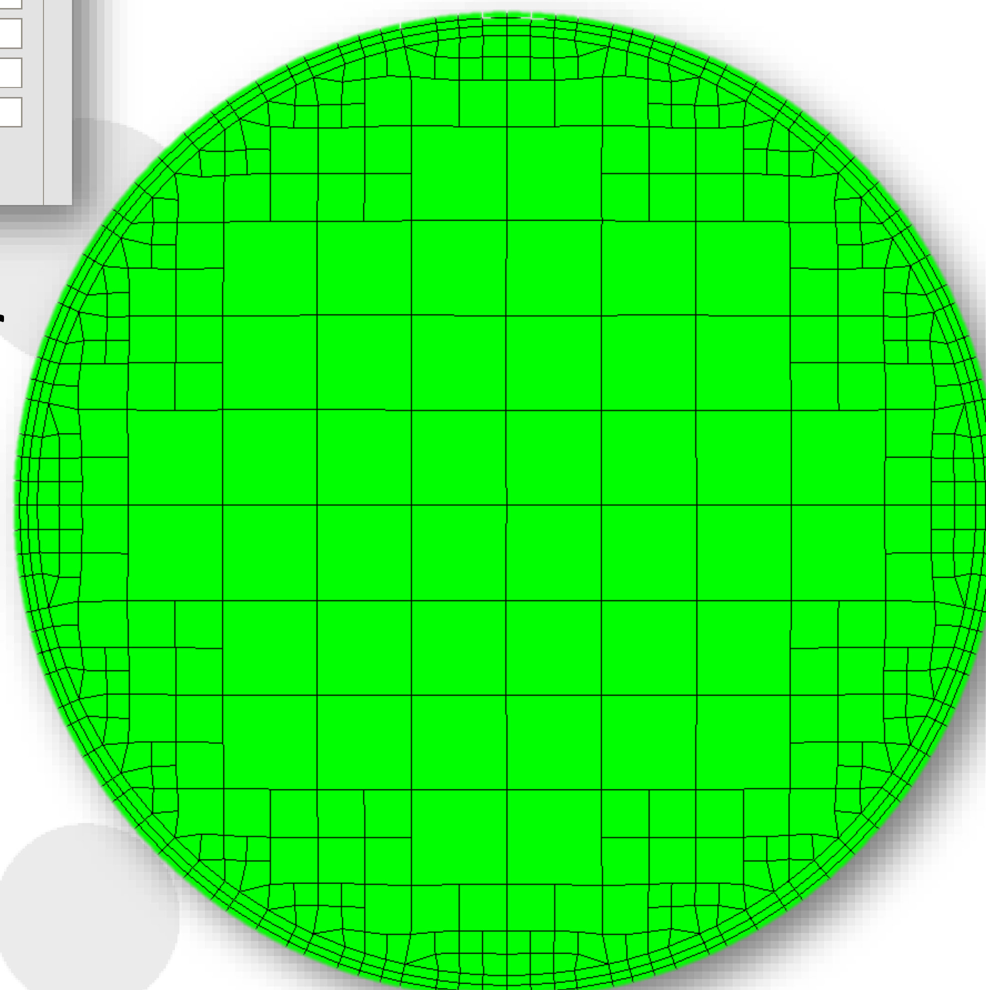


See how each parameter influences layer addition

Straight Pipe | Adding Layers



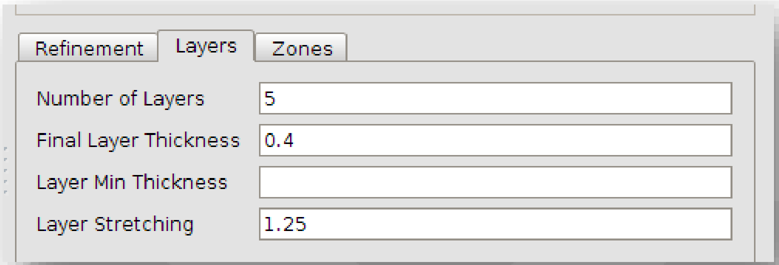
Parameter Study



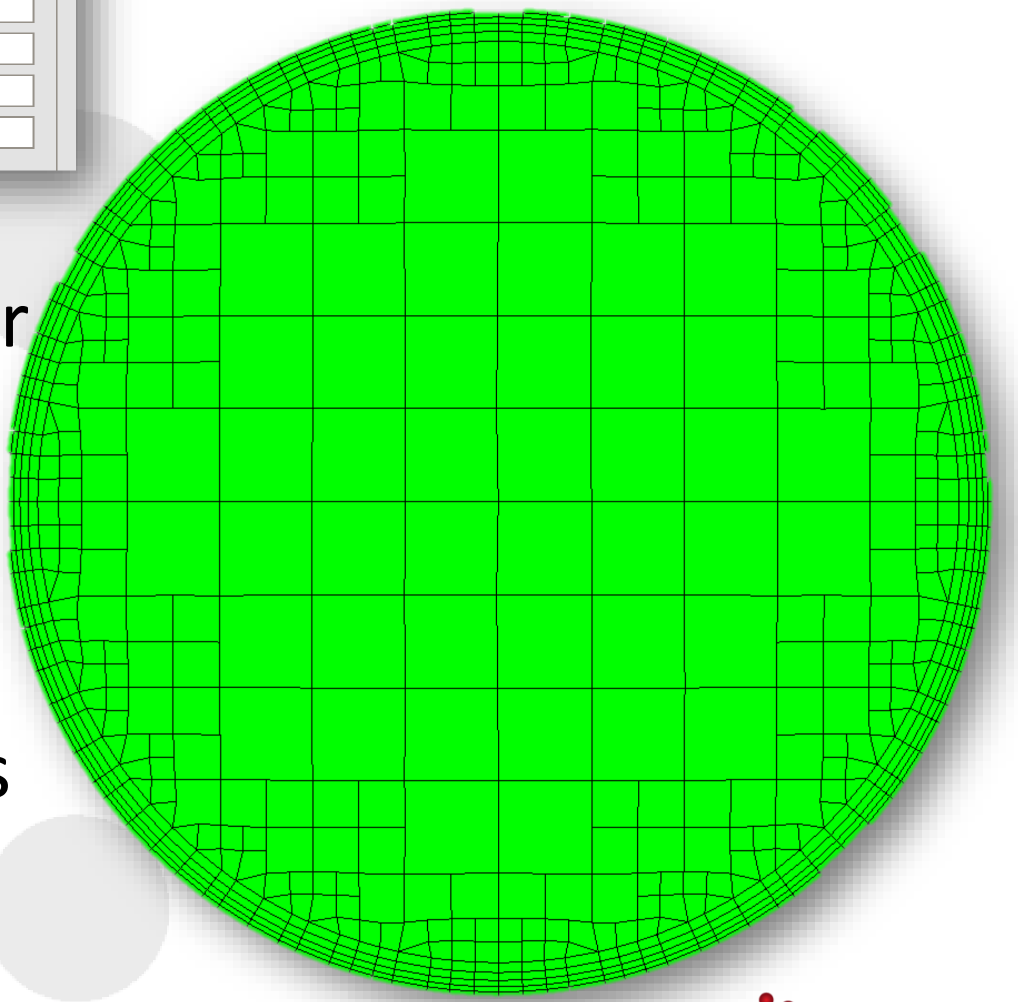
Reduce the Number of Layers to 3

Overall wall layers and height reduced

Straight Pipe | Adding Layers



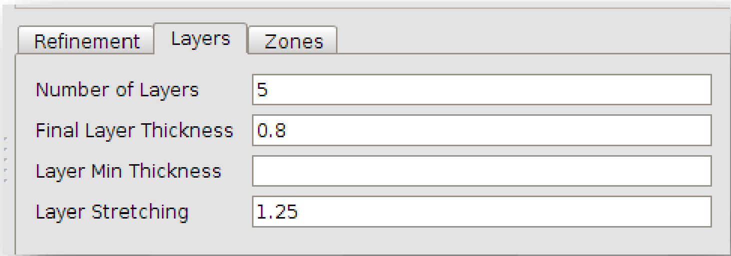
Parameter Study



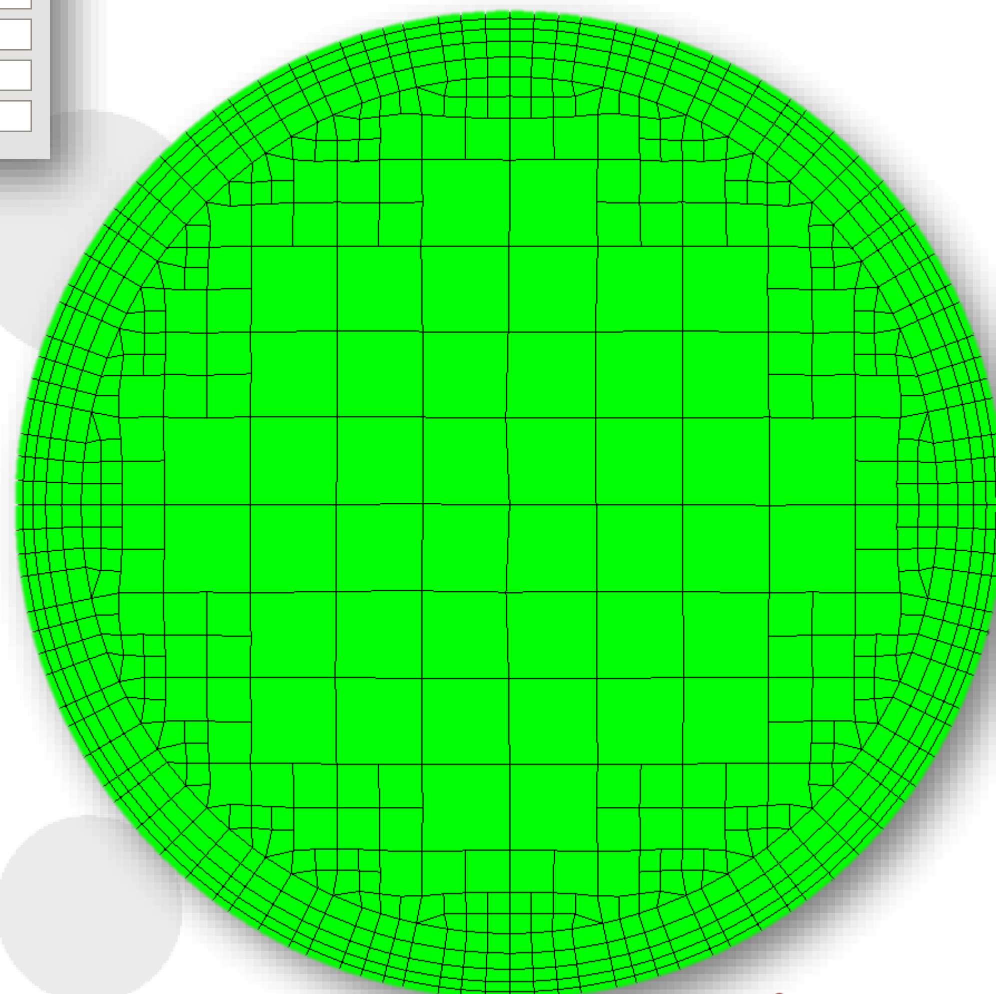
Increase the Number of Layers to 5

Overall wall layers and height increases

Straight Pipe | Adding Layers



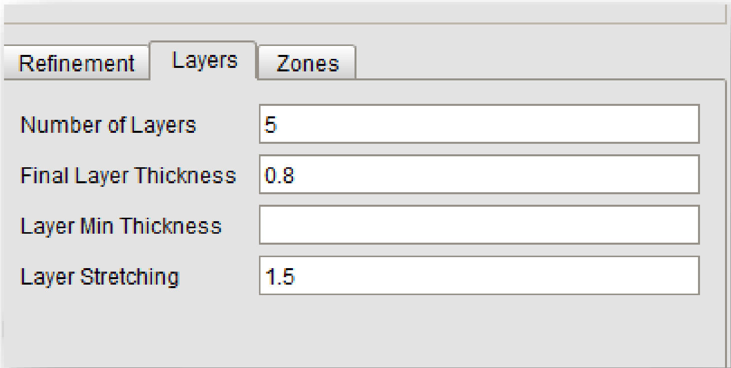
Parameter Study



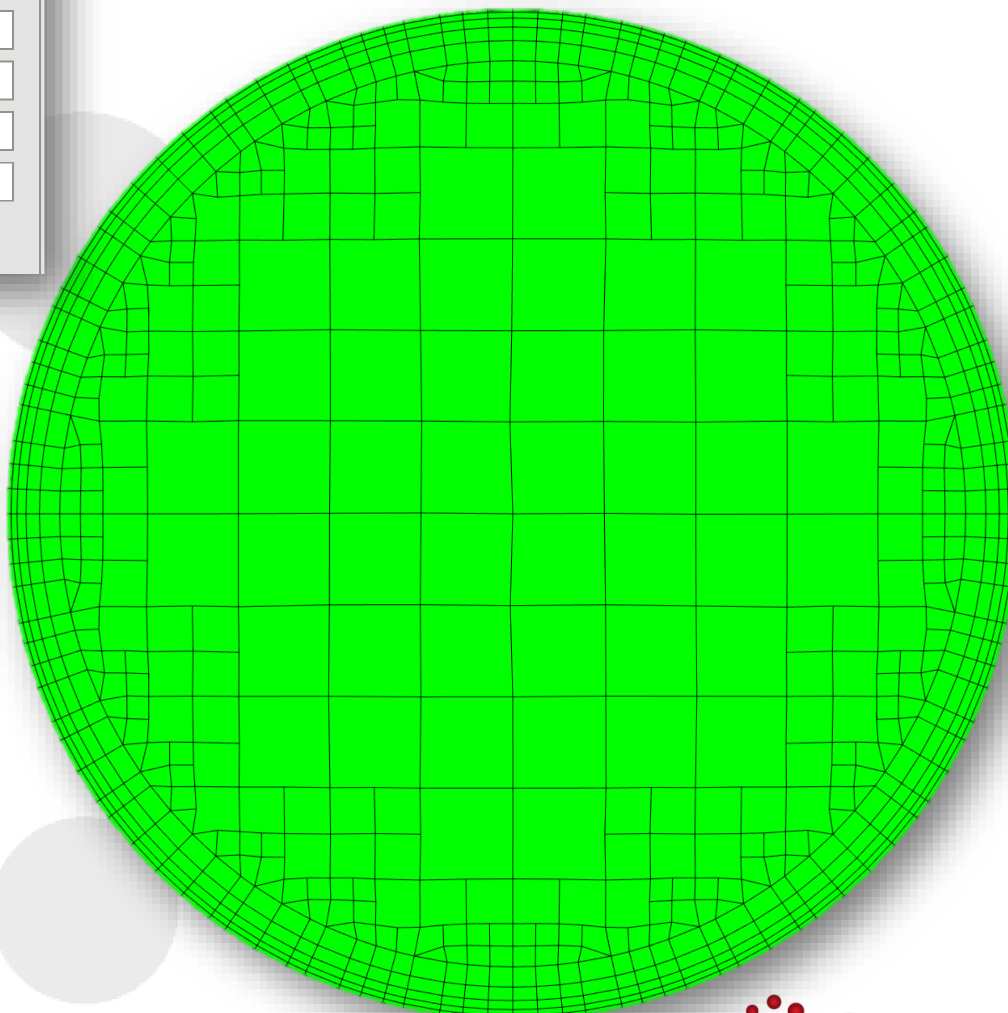
Increase FLT to 0.8

Overall wall layers stays the same and height increases

Straight Pipe | Adding Layers



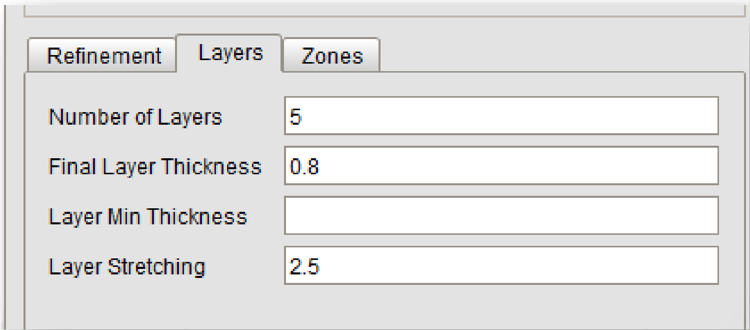
Parameter Study



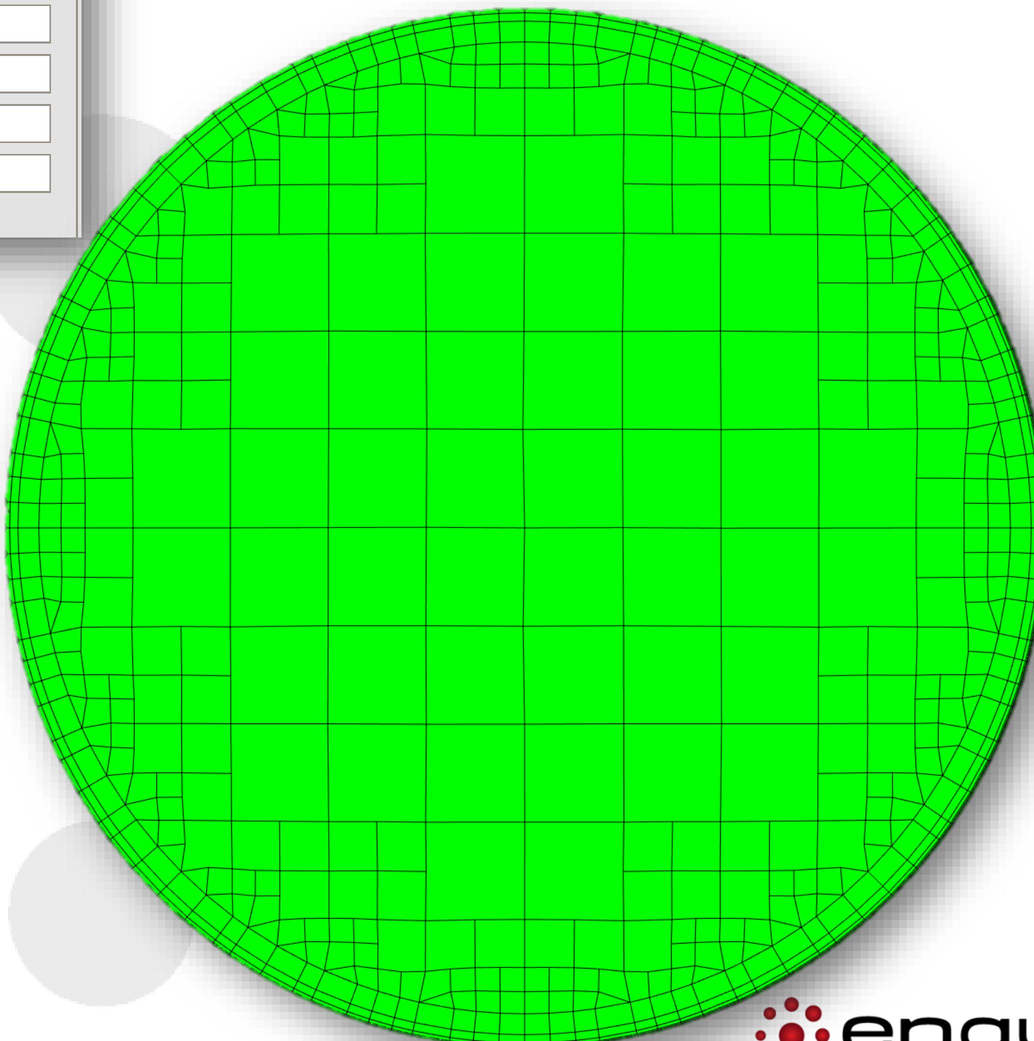
Increase layer stretching to 1.5

Overall wall layers stays the same and height decreases

Straight Pipe | Adding Layers



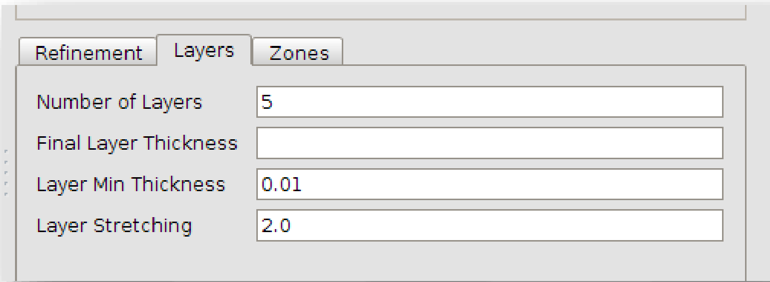
Parameter Study



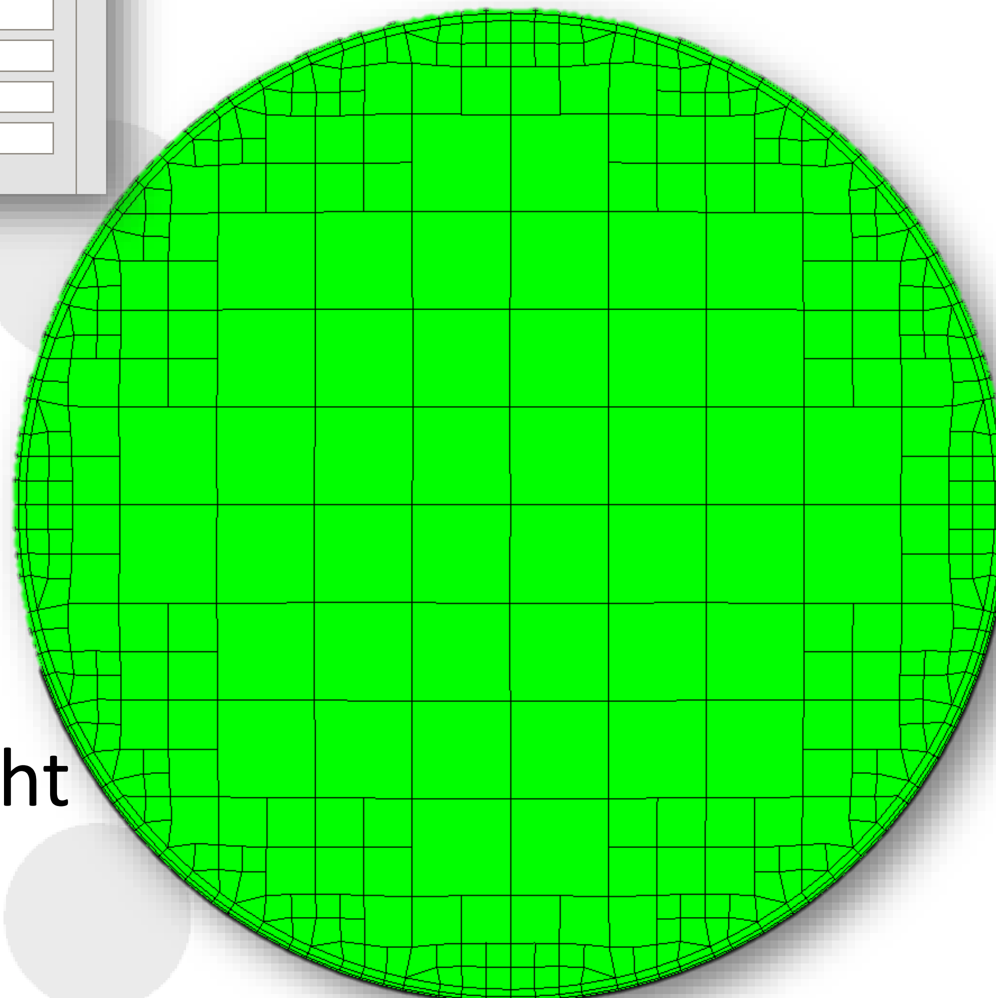
Increase layer stretching to 2.5

Drastically squeeze the first few layers

Straight Pipe | Adding Layers



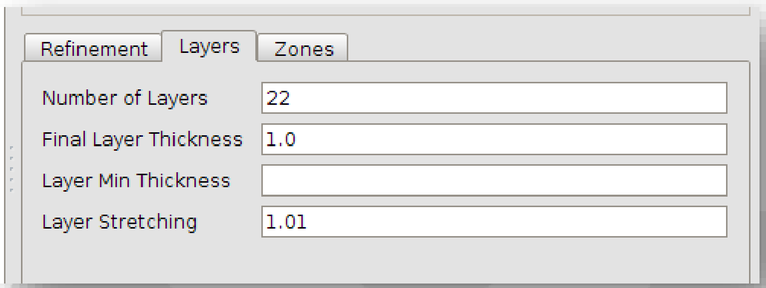
Parameter Study



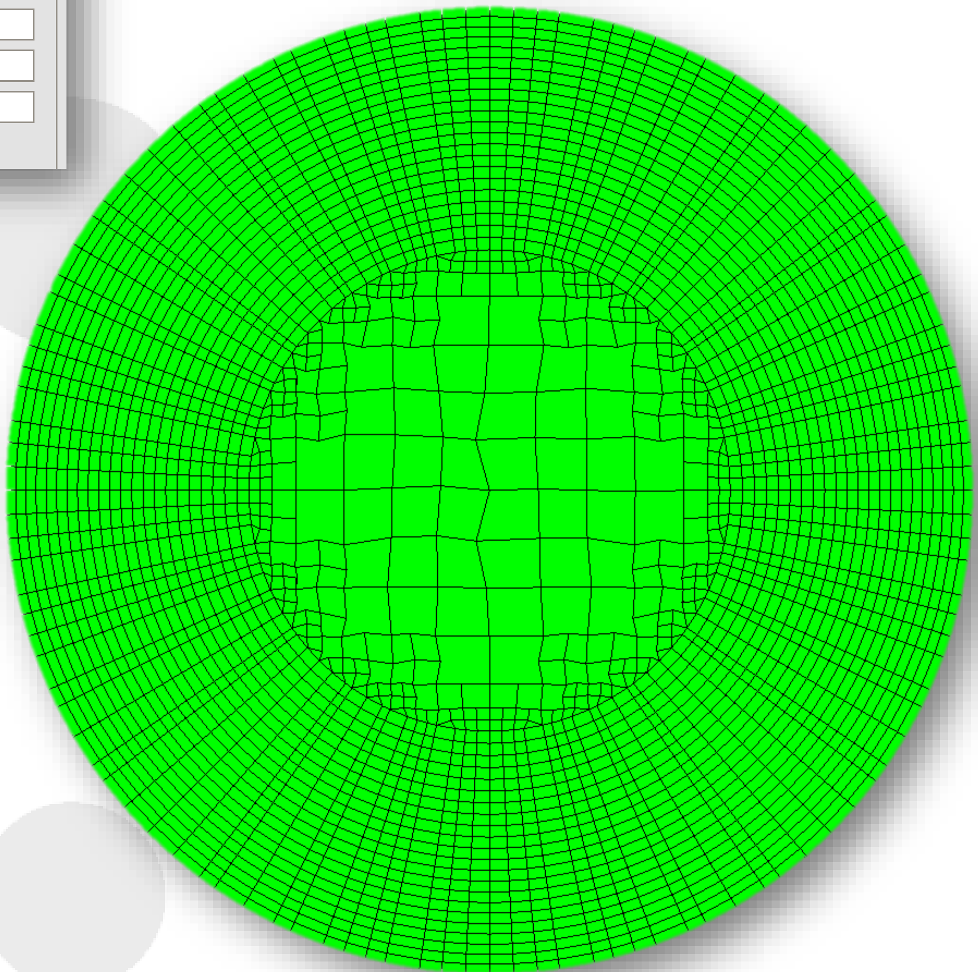
Delete FLT and add small LMT or 0.01

Overall wall layers stays the same height reduced

Straight Pipe | Adding Layers



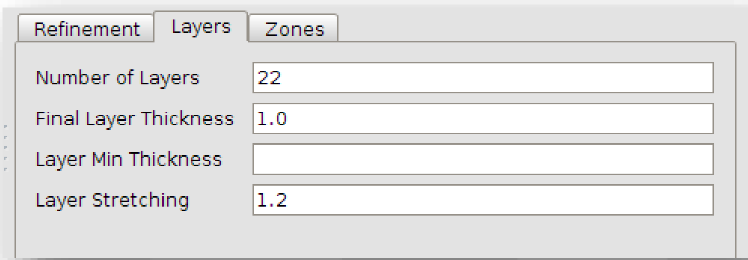
Parameter Study



Test the limits of layer addition

Many layers with similar ratio.

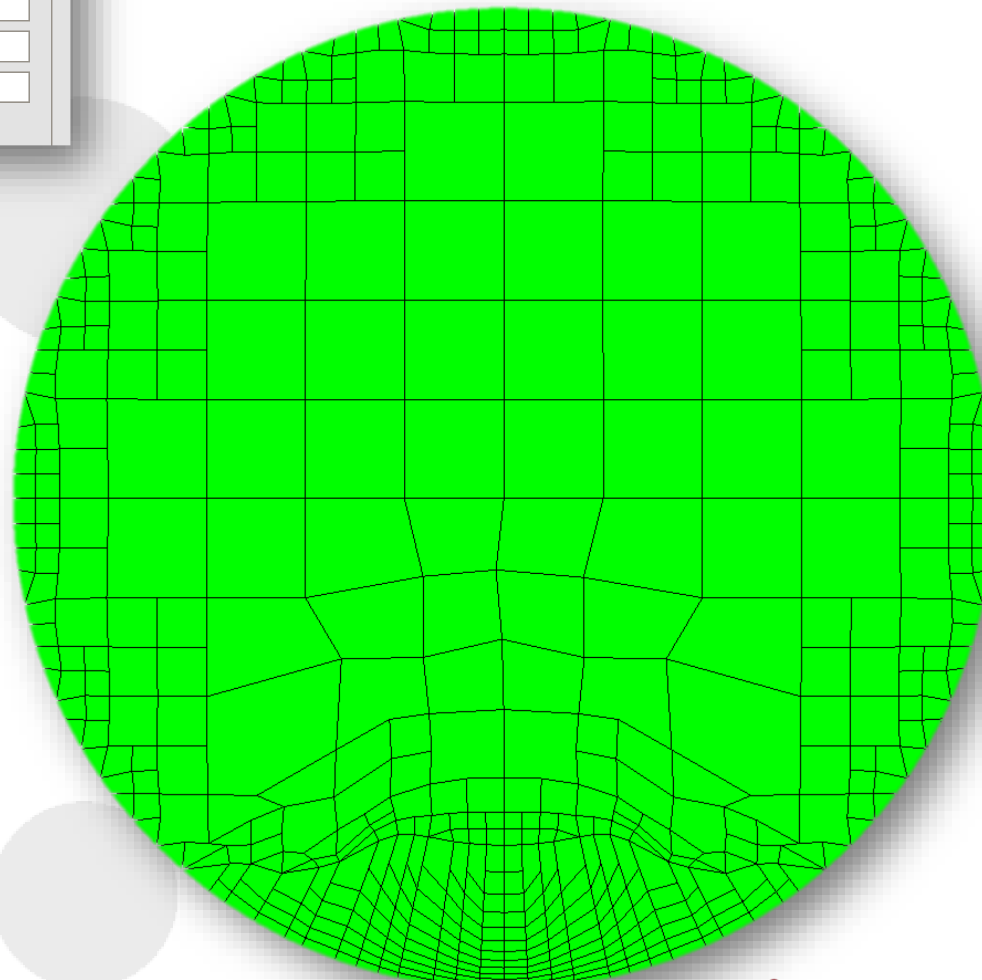
Straight Pipe | Adding Layers



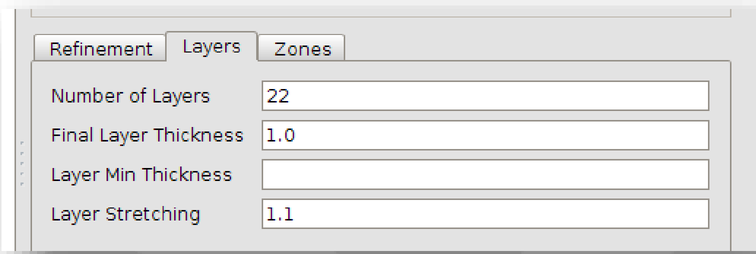
Parameter Study

Increase the layer stretching to 1.2

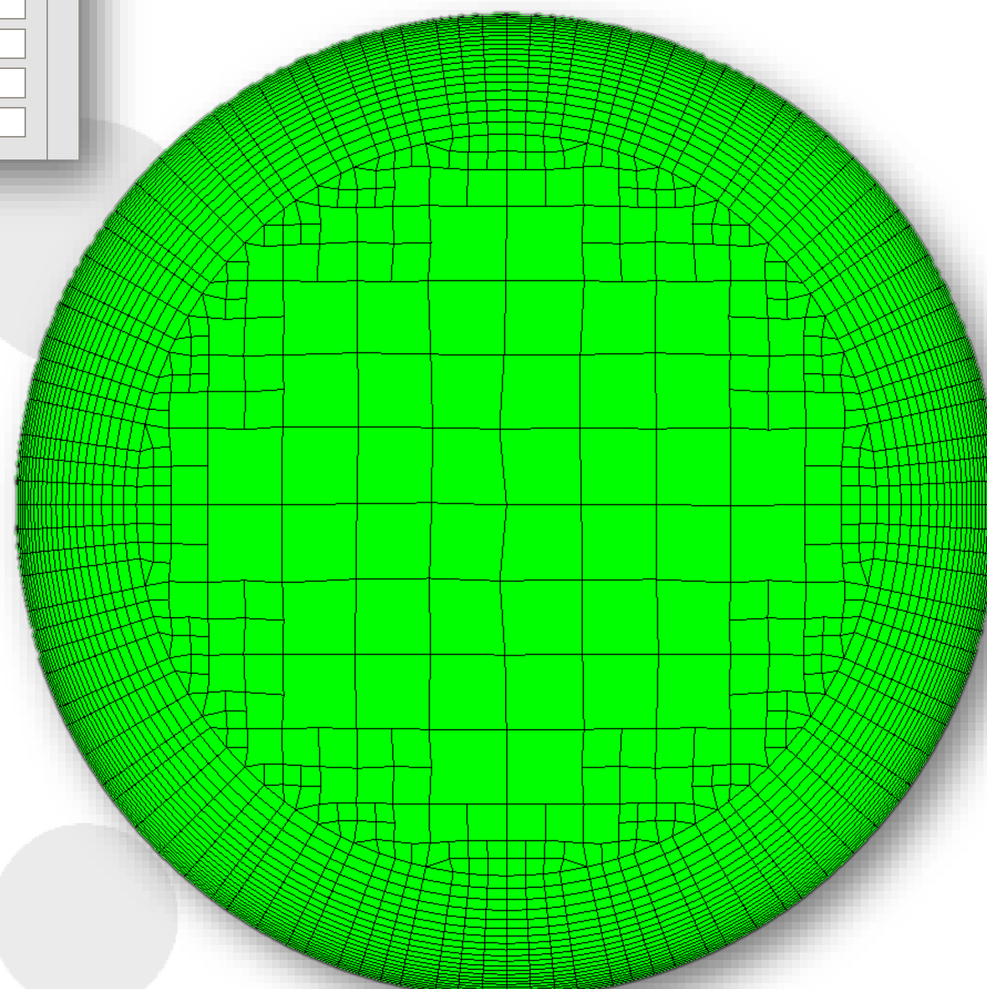
Layers completely collapsed



Straight Pipe | Adding Layers



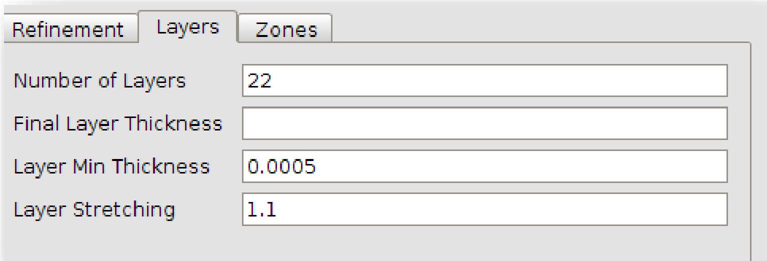
Parameter Study



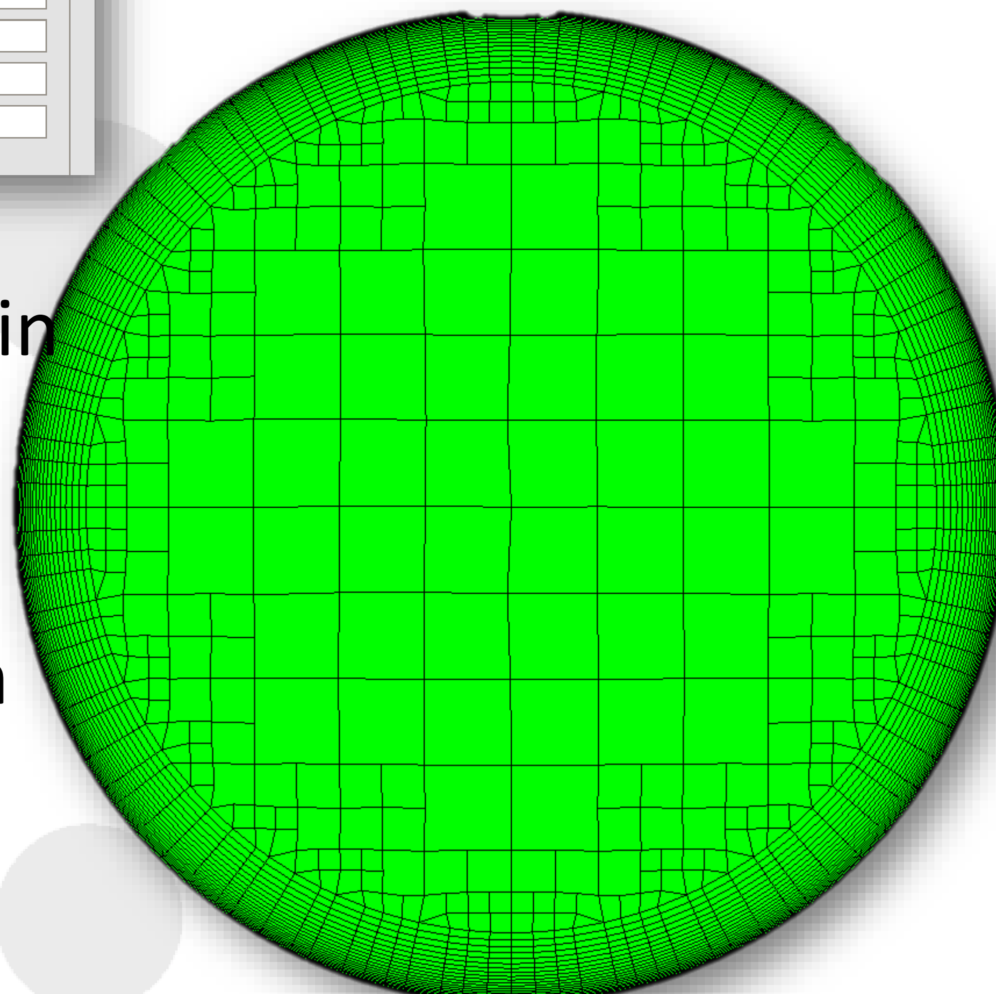
Reduce layer stretching

Many layers added

Straight Pipe | Adding Layers



Parameter Study



Reduce the Layer min thickness

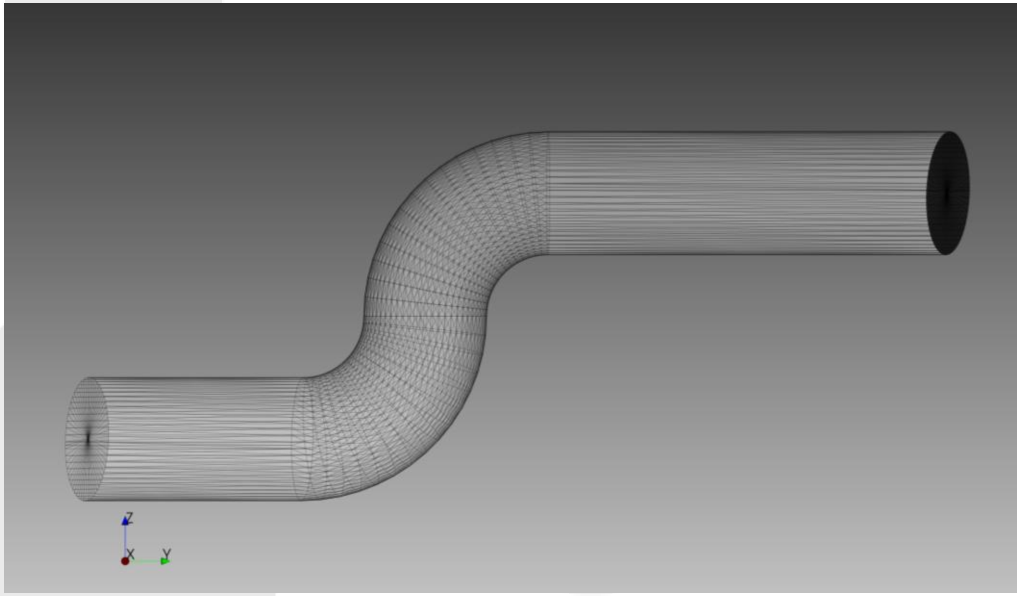
Small layers, growth to lager layers, total coverage

Straight Pipe | Adding Layers

- It is often helpful to choose simplified geometries to prototype settings and understand how controls work
- Layer addition can be a very computationally time consuming action
- Layer collapse is caused by merely the fact that adding layers will create a worse mesh than without layers

Straight Pipe | Adding Layers

- Try a similar study on a bent pipe and you will notice different behavior



- Geometry is included in the first link of the training **tube_bend.stl**

Overall Goal of Session

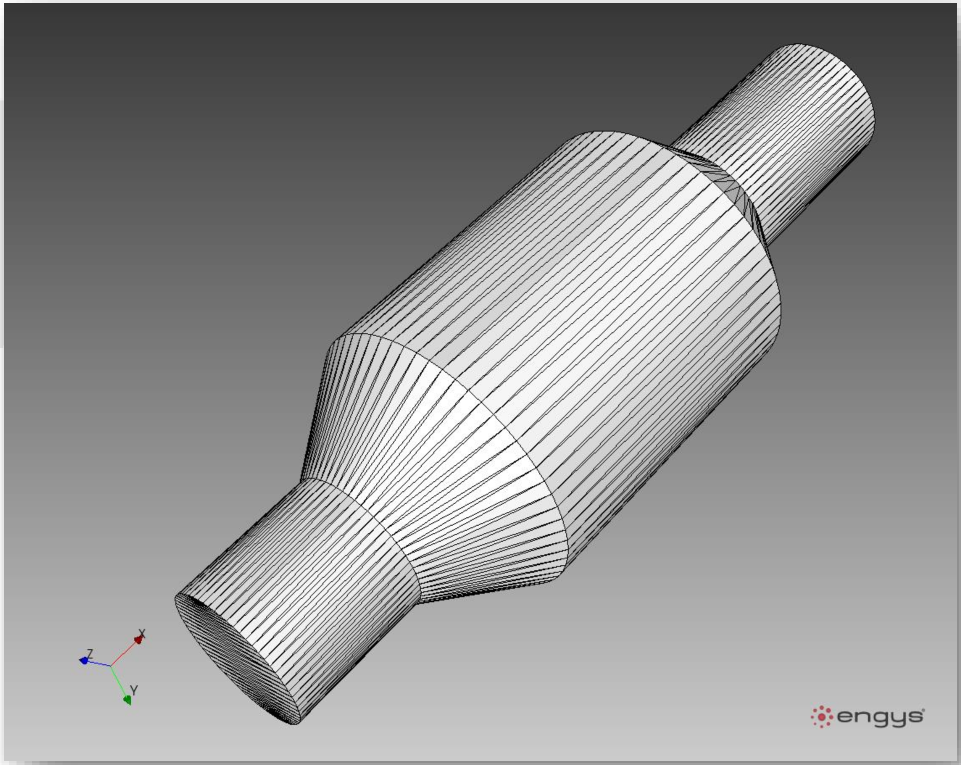
- More practice on feature snapping
- Importing STL's

Skills Obtained

✓ Creating base meshes
✓ Setting surface refinement
✓ Adjusting layer controls and perform a parametric study
✓ Be able to improve bad feature capturing

Reactor Geometry | Snapping Practice

Load the geometry files located in reactor.zip



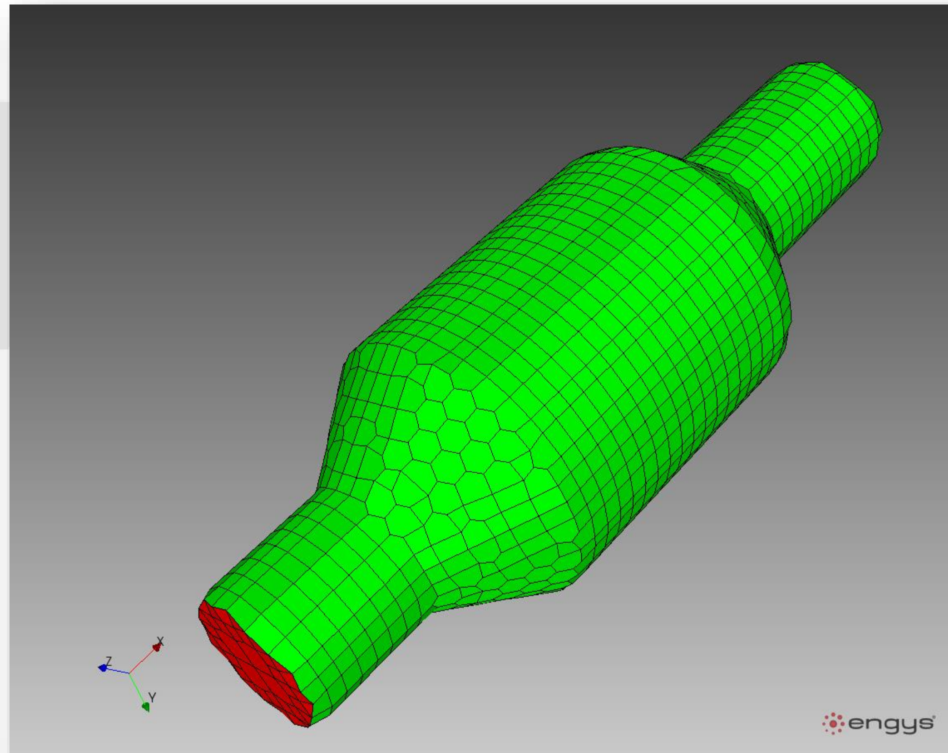
1. **Inlet**
2. **Outlet**
3. **Walls**

Set base mesh to "automatic" and 0.5 and mesh

Reactor Geometry | Snapping Practice

- Change the min and max levels of **walls** to (1,1)

- Lack of resolution?
- Is snapping bad?
- Suggestions?

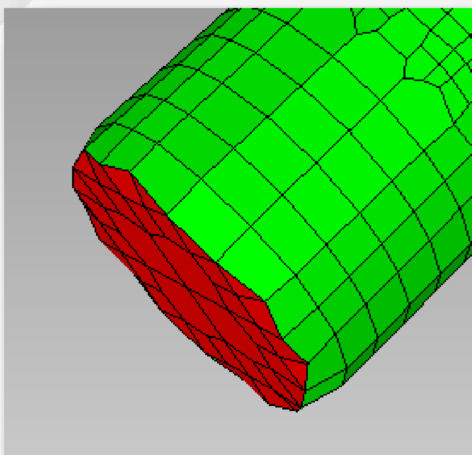


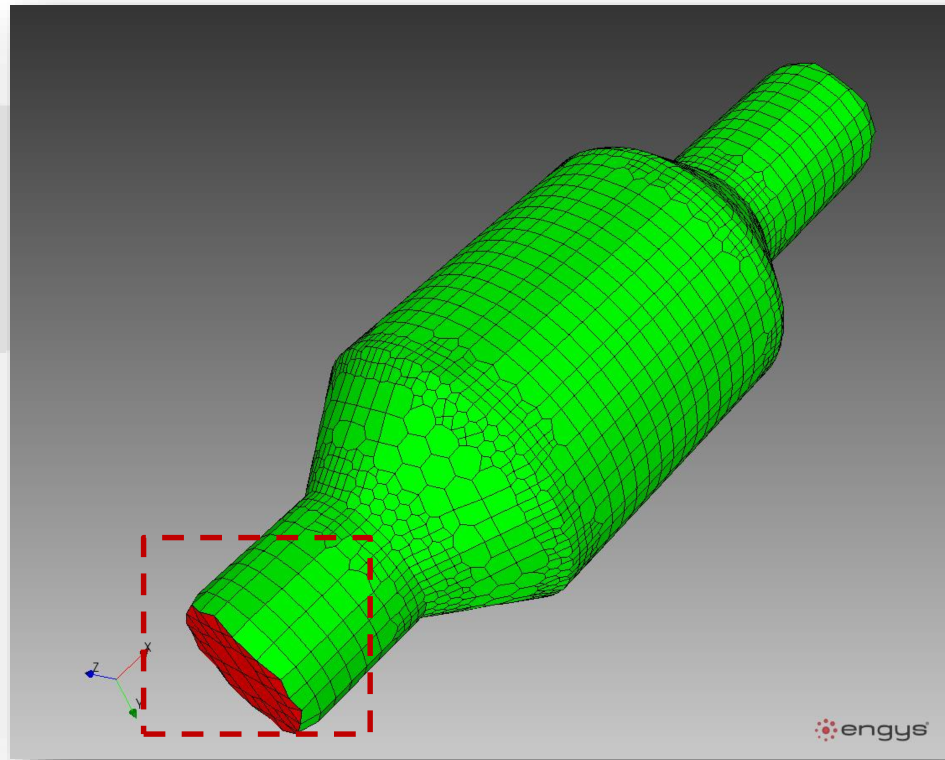
Try increasing the refinement levels and remesh

Reactor Geometry | Snapping Practice

- Change the min and max levels of **walls** to (1,2)

Poor edge capturing

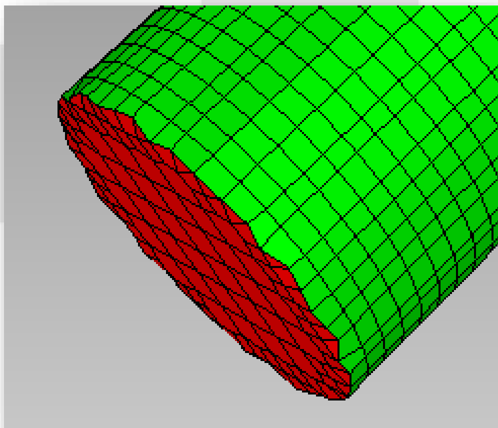




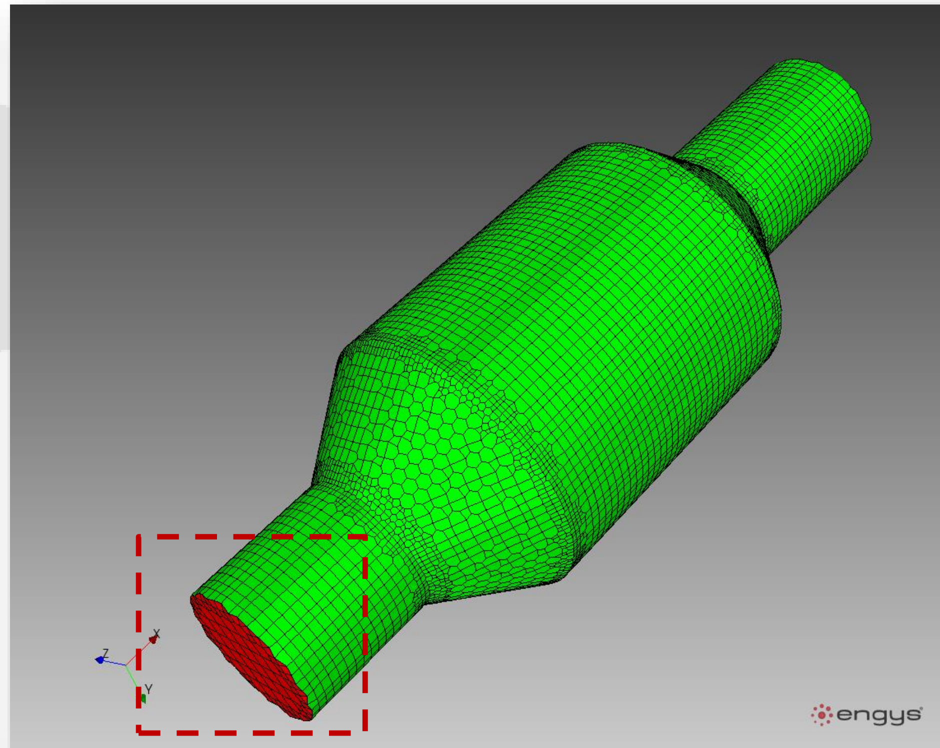
Try increasing the refinement again and remesh

Reactor Geometry | Snapping Practice

- Change the min and max levels of **walls** to (2,3)



Sufficient?

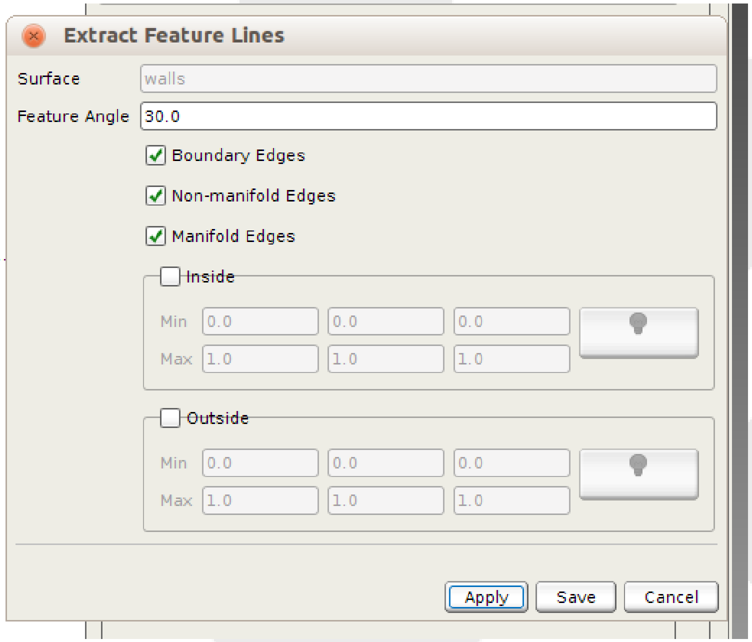


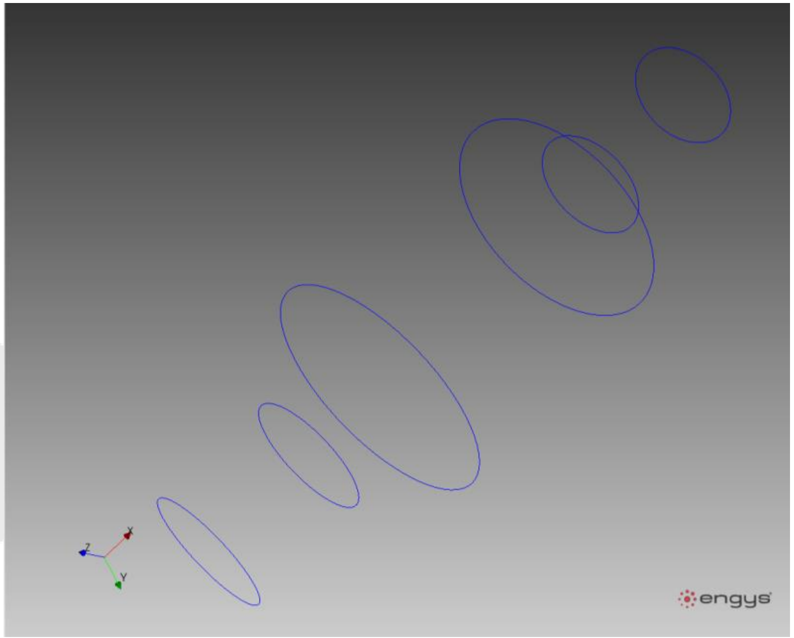
Use the feature edge extraction tool

engys

Reactor Geometry | Snapping Practice

- Right click on STL surface, then select "extract"



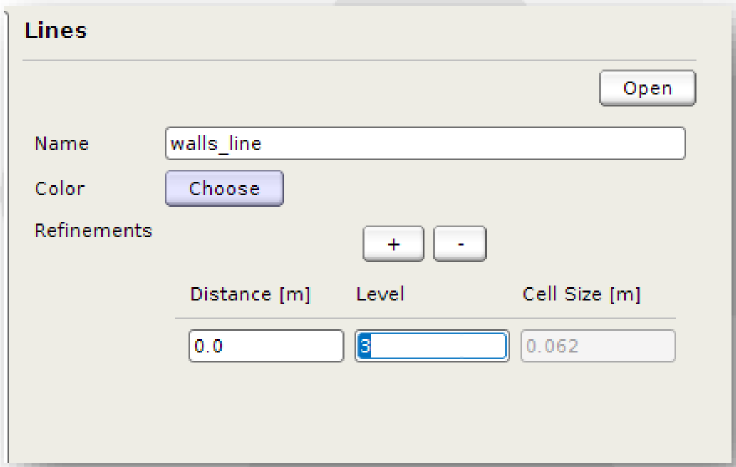


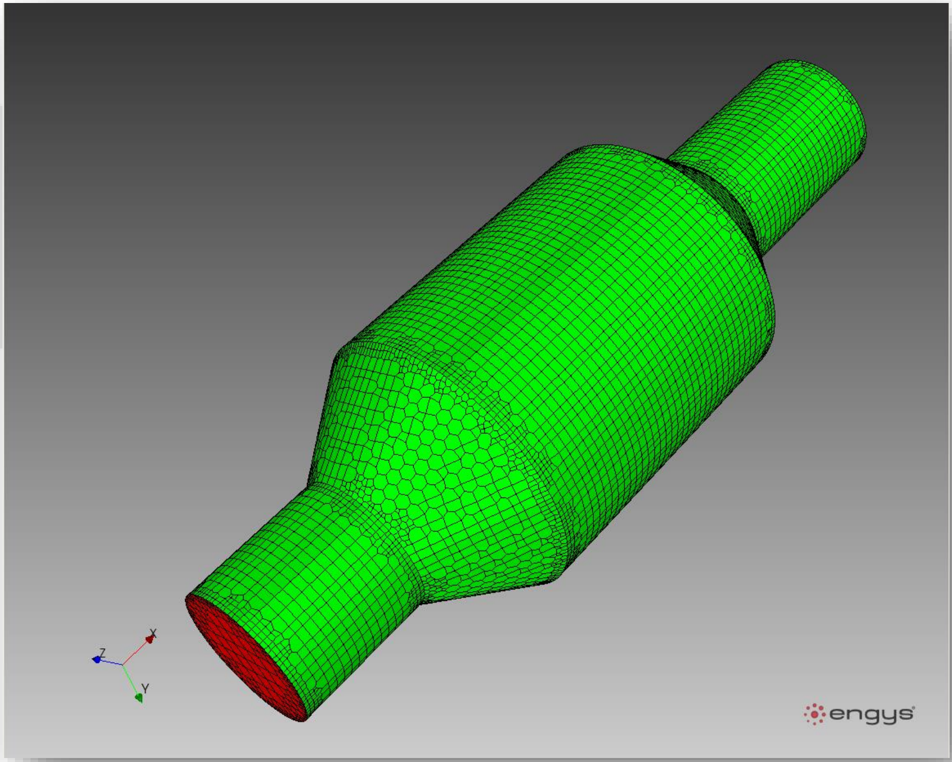
Hit apply to preview and save to save an emesh

engys

Reactor Geometry | Snapping Practice

- In the lines dialog on the newly created emesh





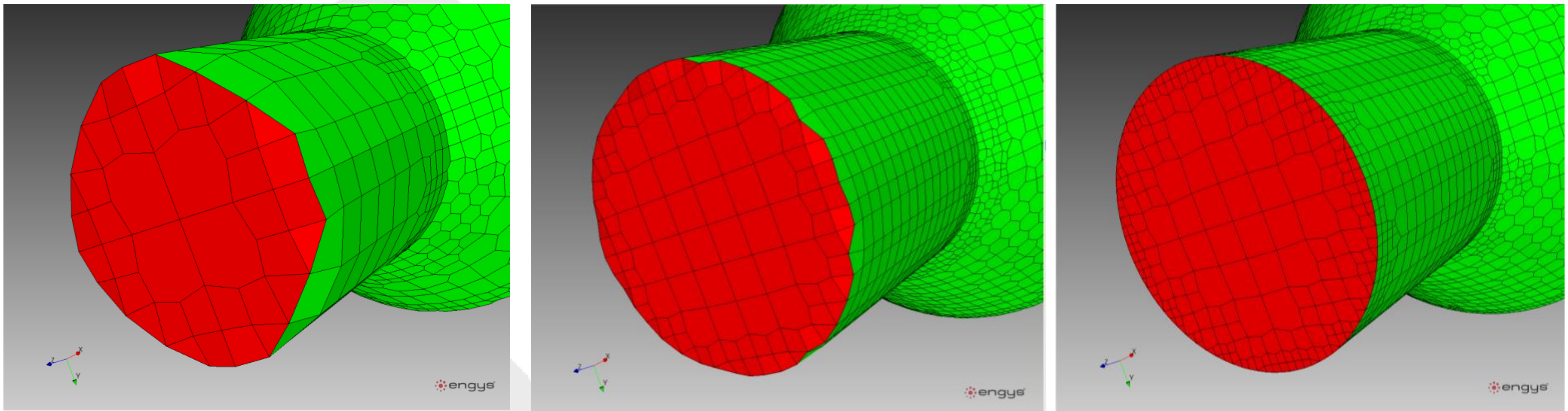
- Set to level 3
- Remesh

We now have a much improved mesh

engys

Reactor Geometry | Snapping Practice

Progression of refinement



- Base mesh size chosen by geometry size and available computing power
- Increase levels accordingly and extract feature edges
- Requires very little adjustment to metrics and defaults

Contents

Closing Remarks | Common Problems

- Castellation prior to snapping unsuccessful
- Trouble snapping to features
- Difficulty in adding layers
- Layers collapsing in certain regions
- Difficulty in choosing default values

Closing Remarks | Common Problems

- Castellation prior to snapping unsuccessful
  - Often a result in using an STL surface that is not water-tight. This prevents the algorithm from deciding what cells are in the region occupied by the keep point.
  - STL surface must be repaired and gaps closed to continue

Closing Remarks | Common Problems

- Castellation prior to snapping unsuccessful
- Trouble snapping to features
  - Often caused by too coarse a base-mesh or surfaces that are not easily identified by a feature angle
  - Lower your base-mesh spacing to a smaller value
  - Try distance refinement near the surface having trouble snapping
  - Increase your levels on that particular surface
  - Use explicit feature snapping and create eMesh or feature edge extraction utility

Try these remedies separately to begin

Closing Remarks | Common Problems

- Castellation prior to snapping unsuccessful
- Trouble snapping to features
- Difficulty in adding layers
  - Caused purely by the addition of a layer does not meet the quality criteria
  - Further caused by near wall cells being too large or poor quality
  - Remedied by refining near surface of interest or adding a smaller number of layers to begin

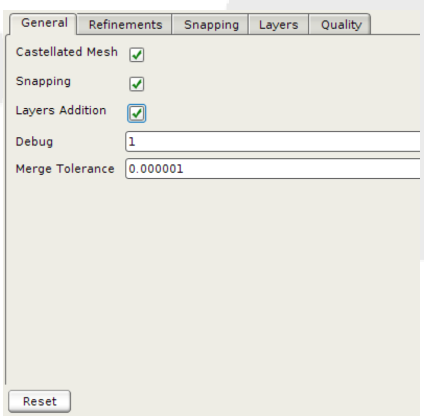
engys

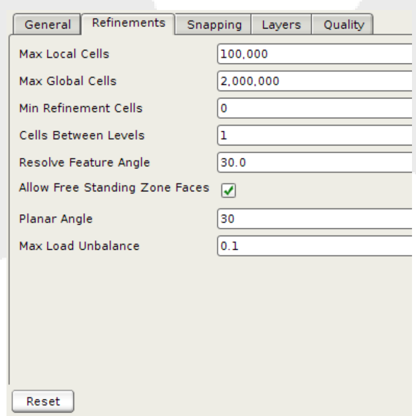
Closing Remarks | Common Problems

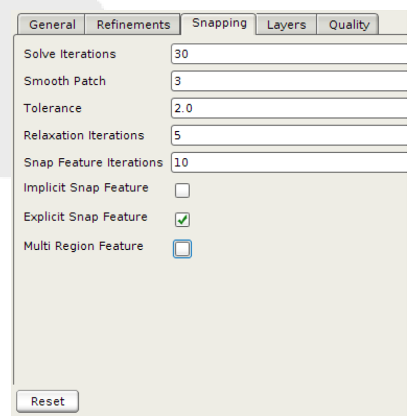
- Castellation prior to snapping unsuccessful
- Trouble snapping to features
- Difficulty in adding layers
- Layers collapsing in certain regions
  - Again caused by poor quality cells or too large of cells
  - Refine cells near the region of interest

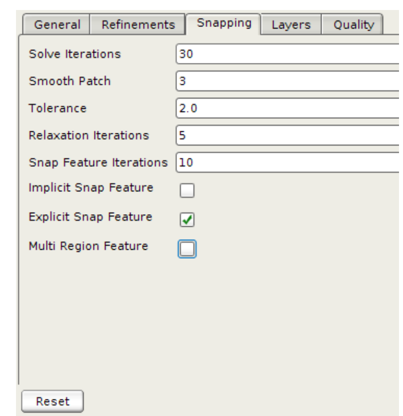
Closing Remarks | Common Problems

- Castellation prior to snapping unsuccessful
- Trouble snapping to features
- Difficulty in adding layers
- Layers collapsing in certain regions
- Difficulty in choosing default values
  - Use existing tutorials or use the defaults set by HELYX-OS



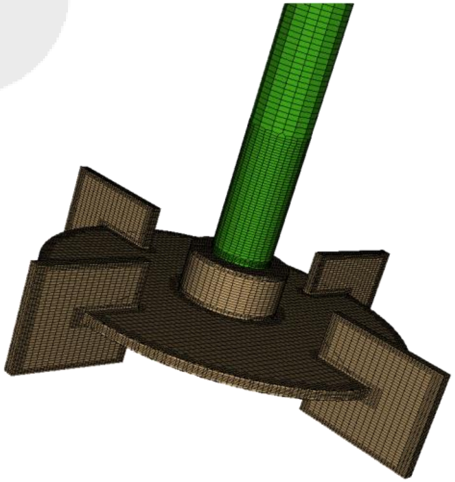






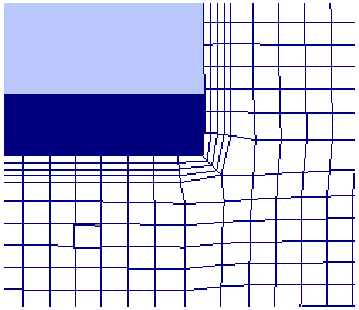
Closing Remarks | Overall

- **Overall**
  - ✓ Coordinate system aligned surface are easier to mesh
  - ✓ It is best **<u>not to change</u>** the meshQuality settings too much
  - ✓ Wedges can be difficult since cells are collapsed in narrow regions
  - ✓ Distance refinement is a per STL surface feature and not a per patch feature.
    - Get around this by importing separate STL files for each patch



Closing Remarks | Overall

- Snapping
  - ✓ Increasing min and max for a particular surface has the same affect as decreasing level 0 size
  - ✓ eMeshes can help fully resolve edges, only if base mesh is sufficiently fine
  - ✓ We can further increase snapping if we increase snapping iterations see appendix
- Layer Addition
  - ✓ Thinner layers are easier to insert
  - ✓ Use relative size rather than absolute layer size
  - ✓ More uniform cells near a surface will have a more uniform boundary layer meshes



engys

Disclaimer

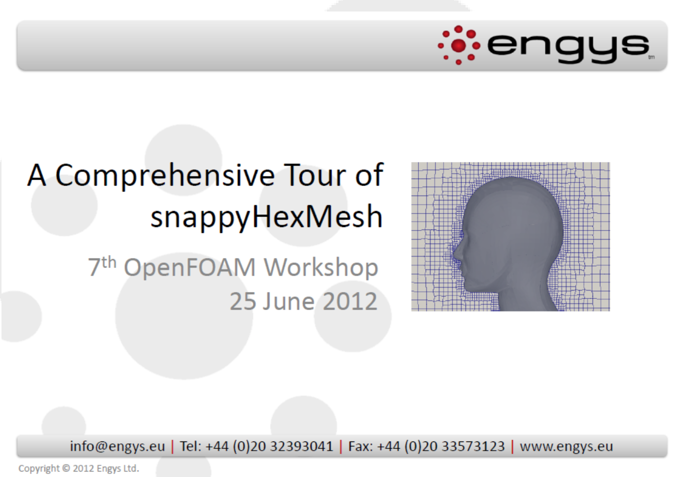
ENGYS Limited is the proprietor of the copyright subsisting in this work. No part of this work may be translated, reprinted or reproduced or utilised in any material form either in whole or in part or by any electronic, mechanical or other means, now known or invented in the future, including photocopying and recording, or in any information storage and retrieval system, without prior written permission from ENGYS Limited.

Applications for permission to reproduce any part of this work should be addressed to ENGYS Limited at info@engys.com

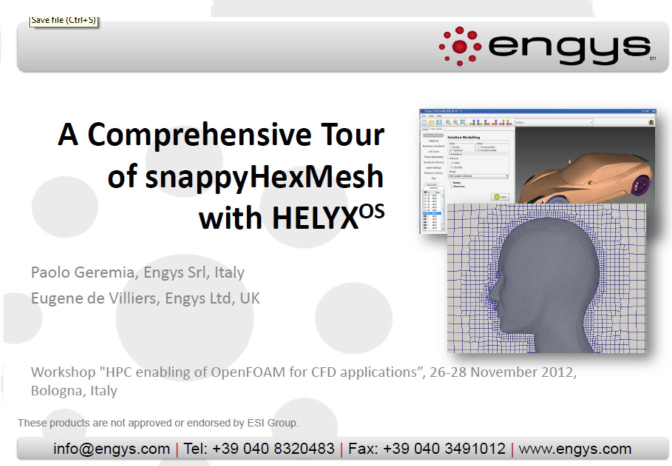
info@engys.com | Tel: +44 (0)20 32393041 | Fax: +44 (0)20 33573123 | www.engys.com

A1: snappyHexMeshDict

Appendix A1: References for snappyHexMeshDict



Andrew Jackson. *A Comprehensive Tour of snappyHexMesh*. 7th OpenFOAM Workshop. June 25 2013. Darmstadt Germany.



Paolo Geremia and Eugene de Villiers. *A Comprehensive Tour of snappyHexMesh with HELYX-OS.* Workshop "HPC enabling of OpenFOAM for CFD applications", 26-28 November 2012, Bologna, Italy

A1: snappyHexMeshDict

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

A1: 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

A1: 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...

engys

A1: 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

A1: 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

A1: 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

A1: Supported Types

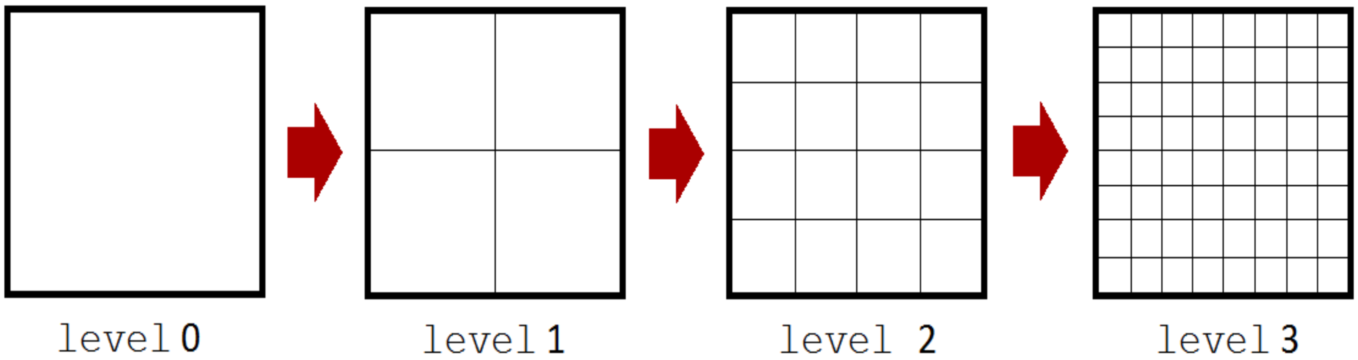
```
twoBoxes
{
    type searchableSurfaceCollection;
    mergeSubRegions true;

    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

A1: 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

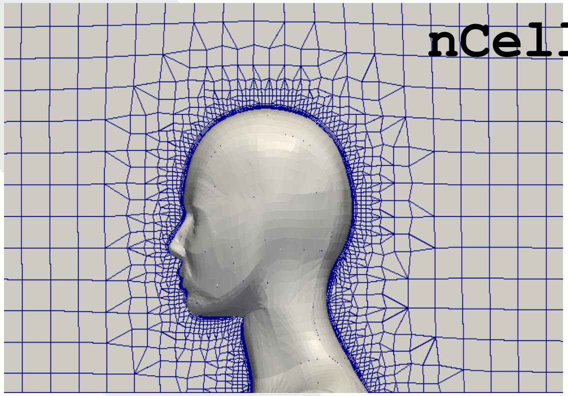


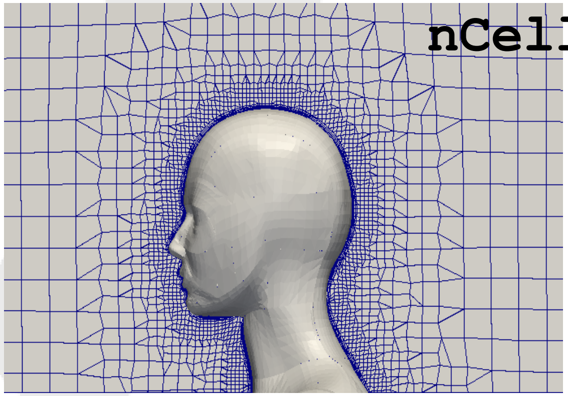
A1: castellatedMesh

```
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





engys

A1: 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;
        //or levels ( (0.1 3) (0.33 2) ) //distance refinement, new in 2.2.x
     }
 );
```

Example .eMesh file

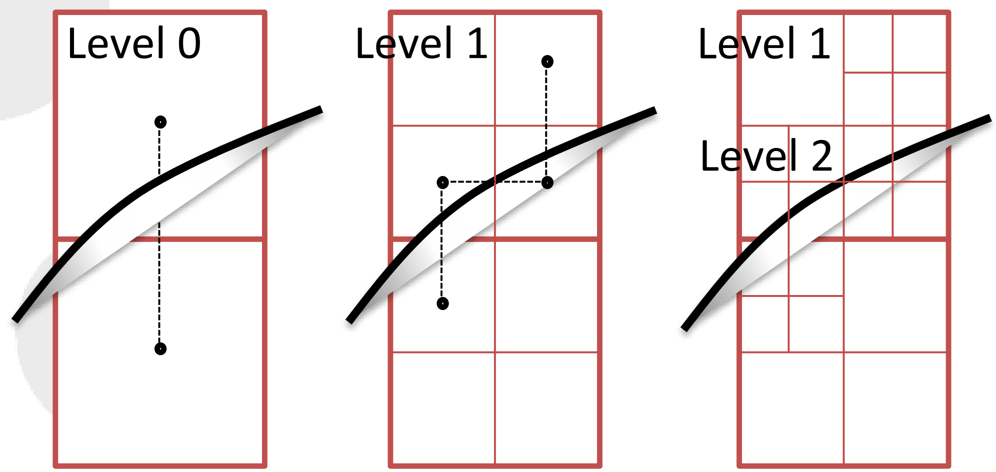
```
FoamFile
{
   version    2.0;
   format     ascii;
   class      featureEdgeMesh;
   location   "constant/triSurface";
   object     flange.eMesh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

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

A1: 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

A1: 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

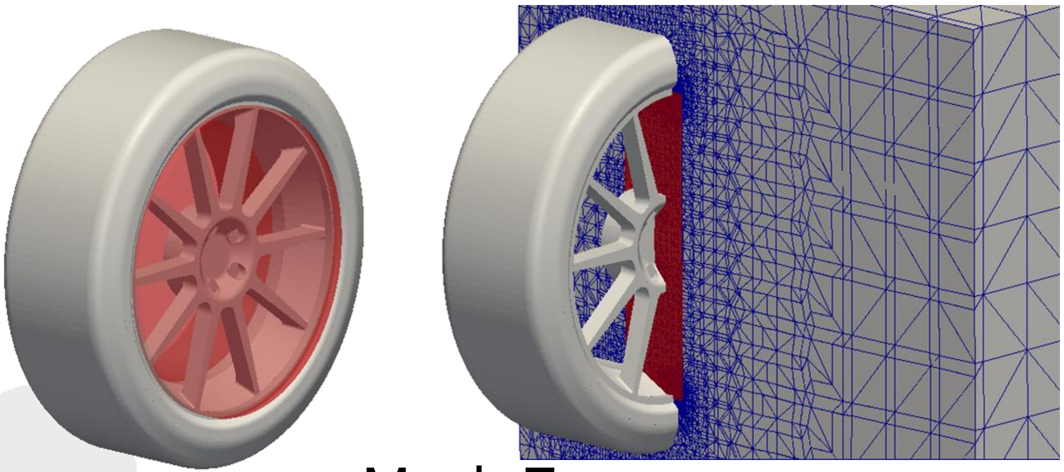
```
refinementSurfaces
   {
      flange
      {
          level (2 3);
          patchInfo
          {
              type wall;
          }
          regions
          {
              "*.inlet|*.outlet"
              {
                  level(3,4);
              }
          }
      }
}
```

A1: 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



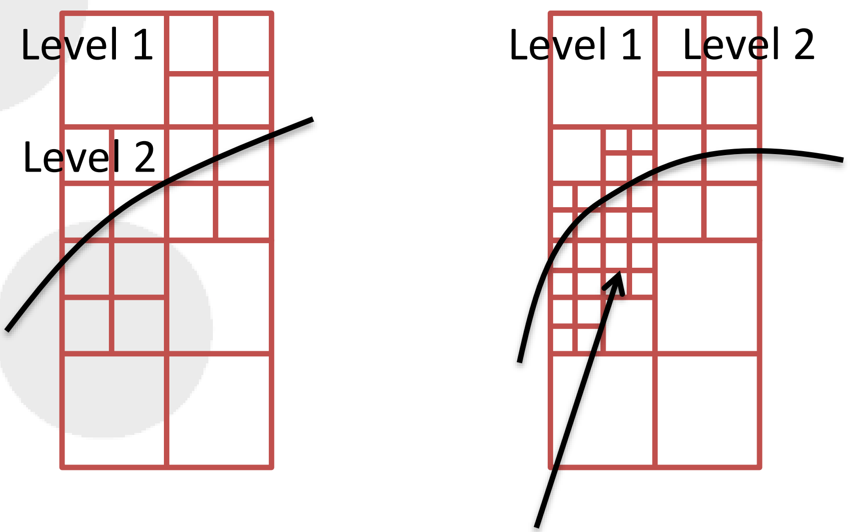
Mesh Zones

A1: 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

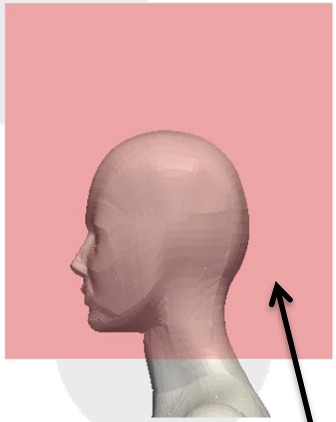
engys

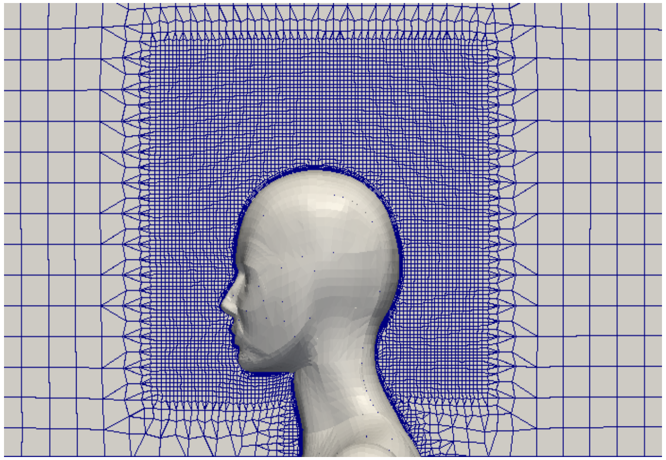
A1: castellatedMesh

```
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



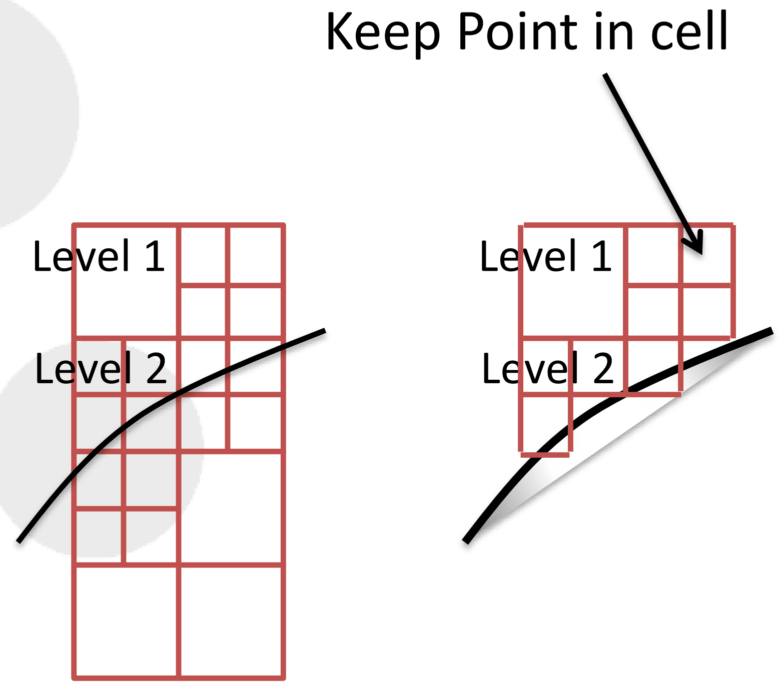


mode inside;
levels ((1E15 3));

A1: 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

A1: castellatedMesh

```
castellatedMeshControls
{
  ...
  refinementSurfaces
  {
    flange
    {
      level (2 3);

      faceZone flange;

      faceType boundary;

      cellZone flange;

      cellZoneInside inside;

    }
}
```

New functionality in 2.2.x

Used to define either

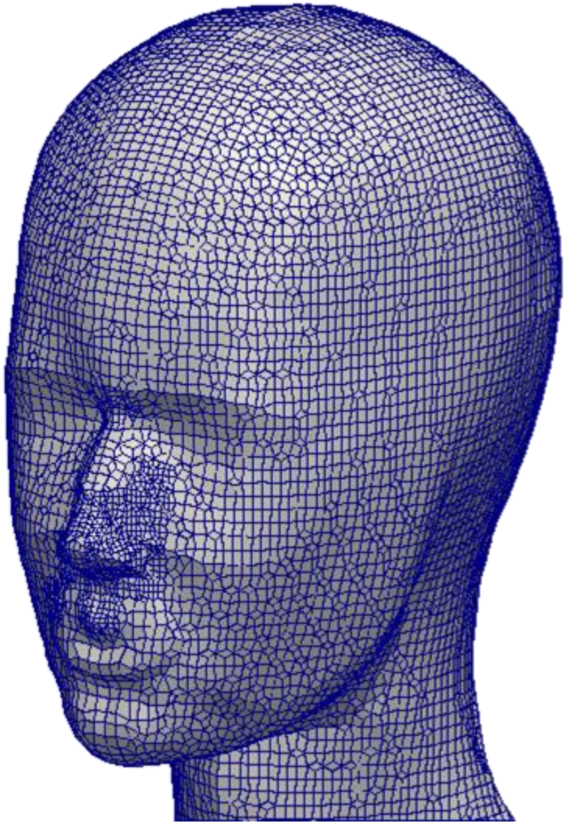
- "baffle"
  creates a pair of faces which match one-to-one.

- "boundary"
  Creates a pair of faces that do not match one-to-one. Less constraint on meshing to create more high-quality meshes.

- "internal"
  keeps faces of faceZone as internal faces.

A1: 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



A1: snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolverIter 300;

    nRelaxIter 5;

    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

Number of pre smoothing iterations of patch points before projection to the surface is performed

A1: snapControls

```
snapControls
{
   nSmoothPatch 3;

   tolerance 1.0;

   nSolverIter 300;

   nRelaxIter 5;

   nFeatureSnapIter 10;

   implicitFeatureSnap false;

   explicitFeatureSnap true;

   multiRegionFeatureSnap false;
}
```

Scaling of the maximum edge length for attraction to the surface

A1: snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolverIter 300;

    nRelaxIter 5;

    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

Number of interior smoothing iterations applied to snapped displacement field

A1: snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolverIter 300;

    nRelaxIter 5;

    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

Controls number of scaling back iterations for error reduction stage

A1: snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolverIter 300;

    nRelaxIter 5;

    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

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>

A1: snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolverIter 300;

    nRelaxIter 5;

    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

New functionality in 2.2.x

Uses resolveFeatureAngle to detect changes in the features to find “creases”. Snapping is then performed on a “representation” of the feature from the local topology. (default = false)

A1: snapControls

```
snapControls
{
   nSmoothPatch 3;

   tolerance 1.0;

   nSolverIter 300;

   nRelaxIter 5;

   nFeatureSnapIter 10;

   implicitFeatureSnap false;

   explicitFeatureSnap true;

   multiRegionFeatureSnap false;
}
```

Indicates that an eMesh file is to be used to approximate features within the mesh i.e. uses features defined in castellatedMeshControls
(default = true;

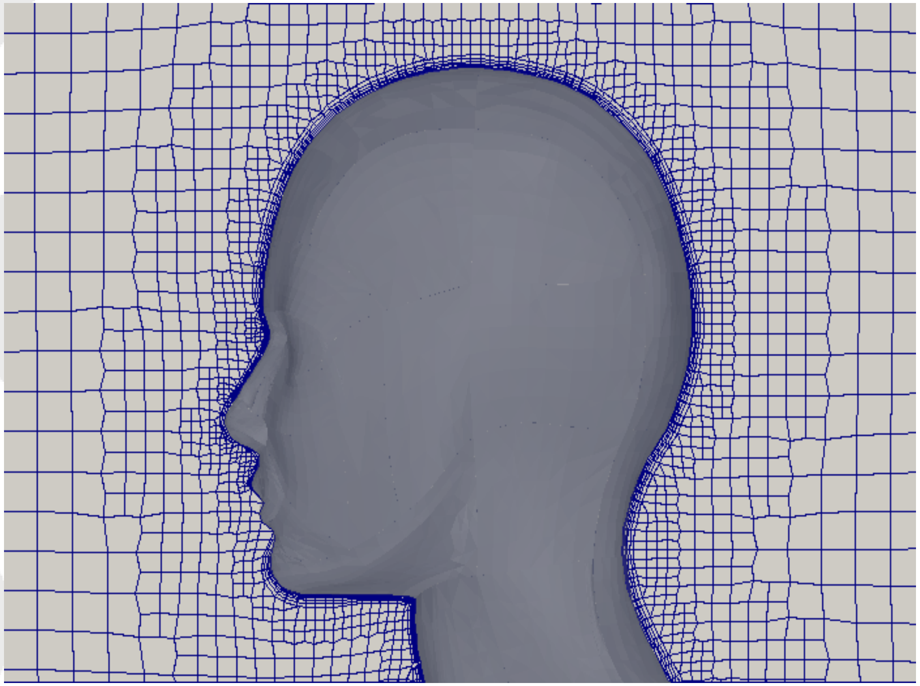
A1: snapControls

```
snapControls
{
    nSmoothPatch 3;

    tolerance 1.0;

    nSolverIter 300;

    nRelaxIter 5;

    nFeatureSnapIter 10;

    implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

In conjunction with explicitFeatureSnap, this is used to detect the features between multiple surfaces.

A1: 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



A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 30;
  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

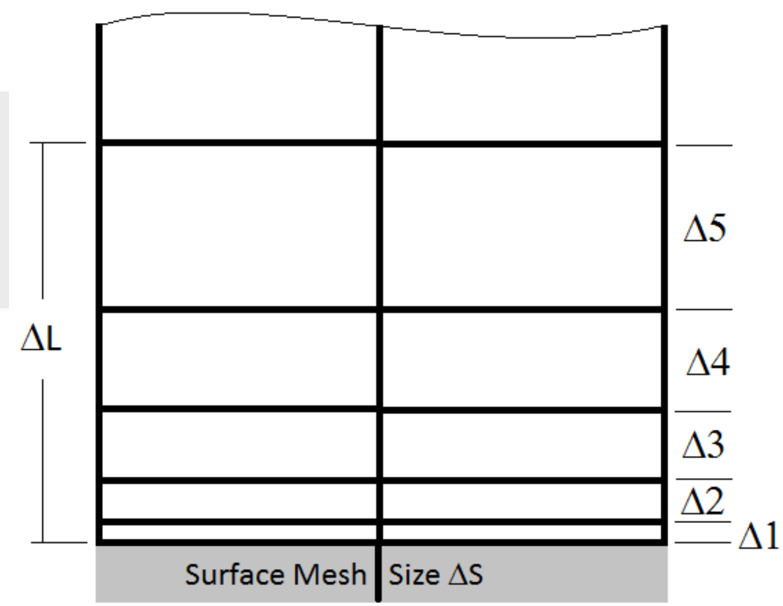
Specification of the number of layers to be grown on each patch. Supports regular expression syntax

A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 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. $\frac{\Delta 5}{\Delta S}$

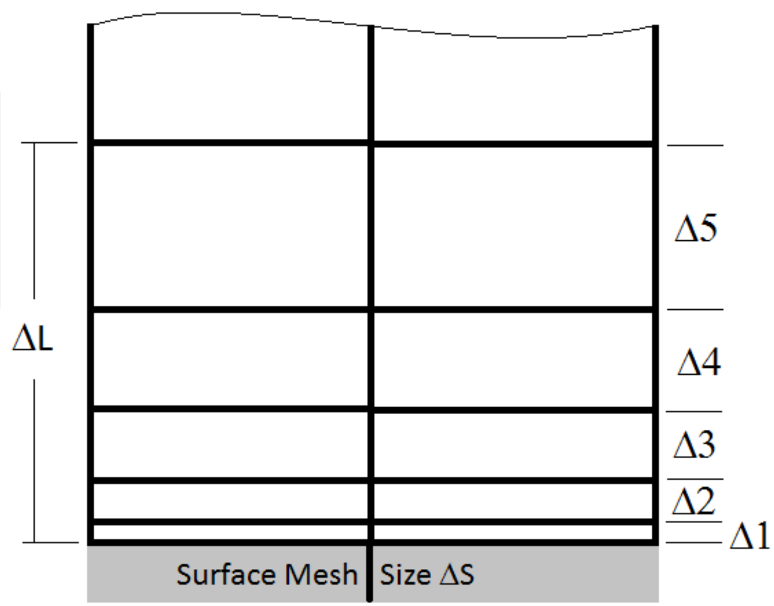
expansionRatio is the ratio of heights from one layer to the next consecutive layer in the direction away from the surface, i.e. $\frac{\Delta 2}{\Delta 1} = \frac{\Delta 3}{\Delta 2} = \frac{\Delta 4}{\Delta 3} = \frac{\Delta 5}{\Delta 4}$



A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 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



A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 30;
  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

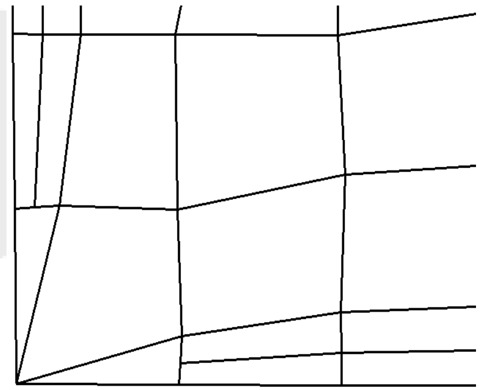
Specification of a minimum layer thickness below which height layers will automatically be collapsed.

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.

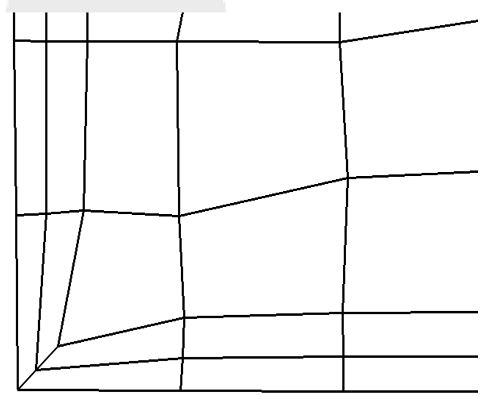
A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 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;

A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 30;
  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

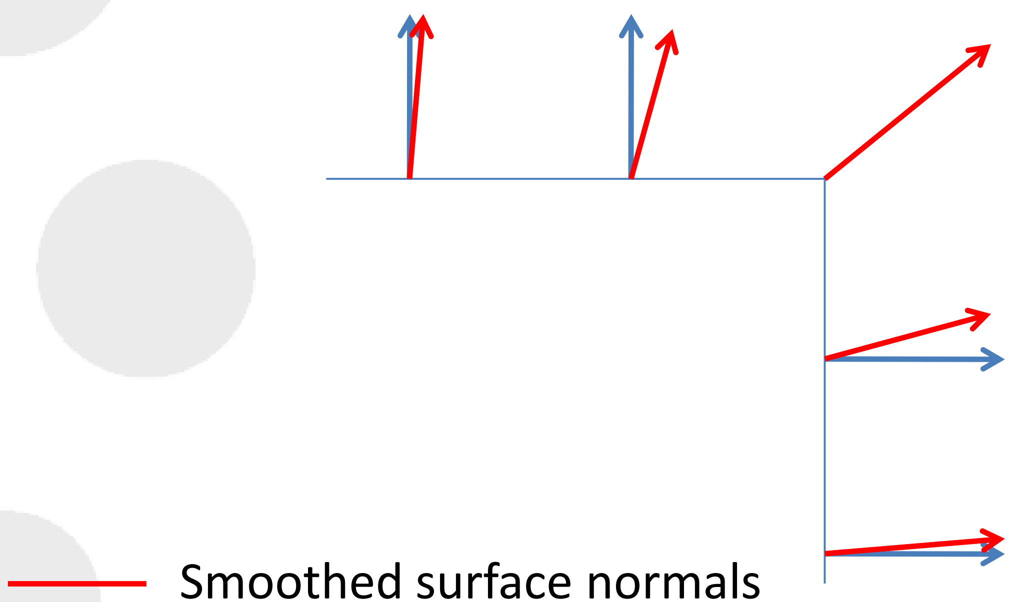
Specifies what feature angle to allow layers to slip "perpendicularly" up a patch

i.e. "at non-patch sides, allow mesh to slip if extrusion direction makes an angle larger than slipFeatureAngle"

A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
   featureAngle 30;
   slipFeatureAngle 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.

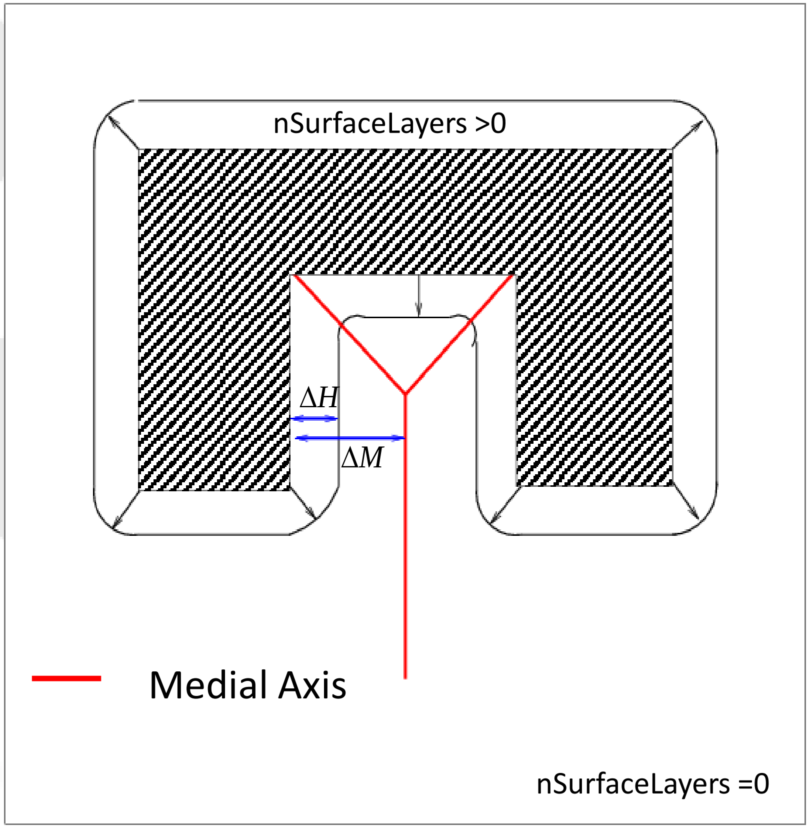


engys

A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 30;
  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;

  minMedianAxisAngle 80;
  maxThicknessToMedialRatio 0.3;

   maxFaceThicknessRatio 0.5;

   nLayerIter 50;

   meshQualityControls::relaxed.
   nRelaxedIter 20;

   nRelaxIter 5;
}
```

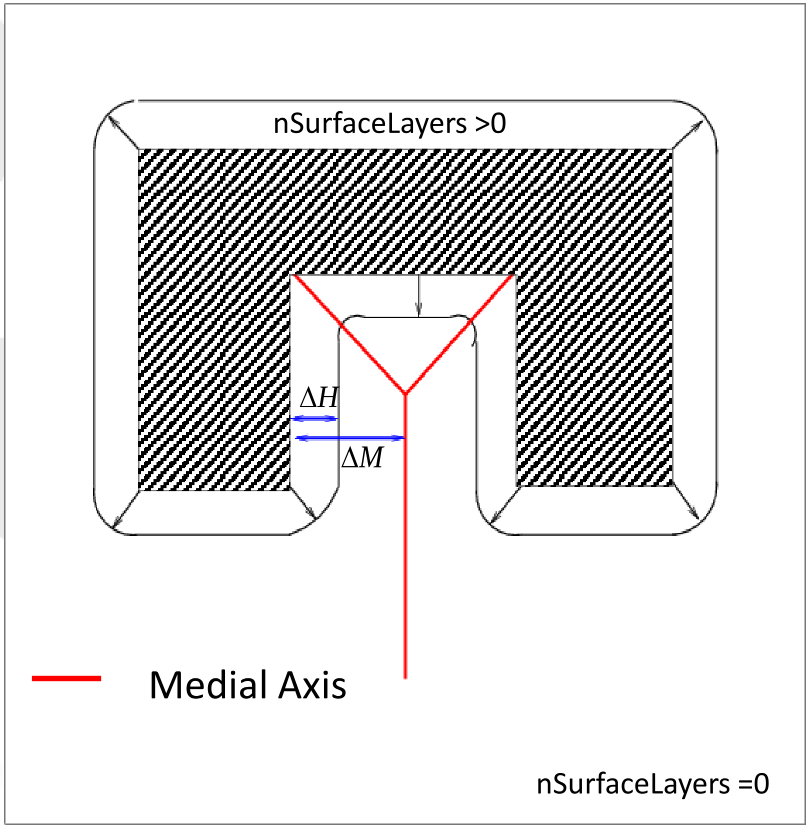
This angle is used to define a medial axis which is used when moving the mesh away from the surface



A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 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



A1: addLayersControls

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

   finalLayerThickness 0.4;
   expansionRatio 1.15;

   minThickness 0.2;

   relativeSizes true;

   // Advanced settings
  featureAngle 30;
  slipFeatureAngle 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