GMSH Meshing Workshop for OpenFOAM

Fred Morin

The completed files for this workshop can be downloaded here:
https://github.com/fredmorin18/GmshTutorial

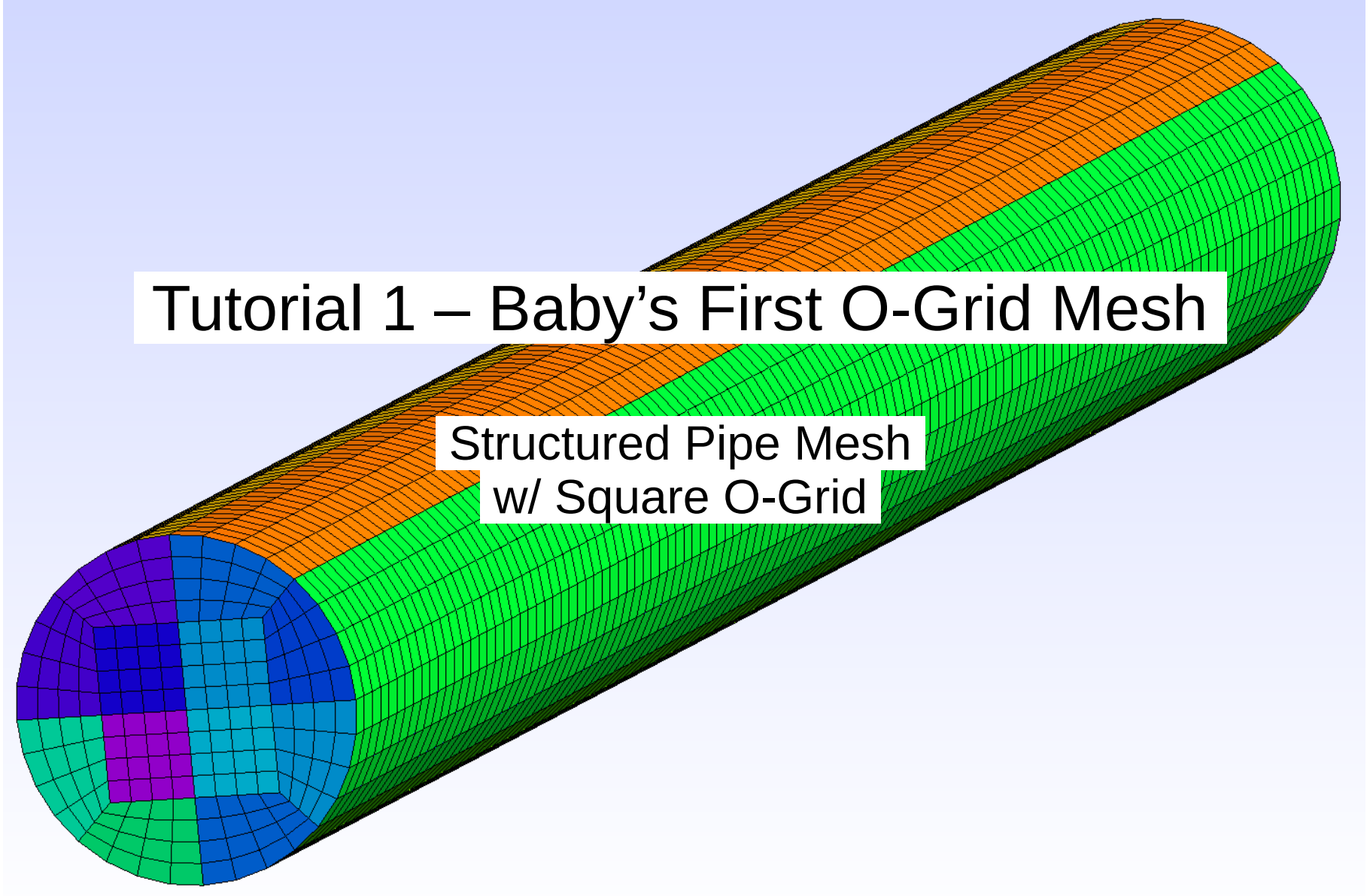
GMSH Installaiton

- Install instructions: https://gmsh.info/#Download
- Ubuntu
  - sudo apt-get install gmsh
  - or from website: https://gmsh.info/bin/Linux/gmsh-4.13.1-Linux64.tgz
- Windows
  - Latest release from website: https://gmsh.info/bin/Windows/gmsh-4.13.1-Windows64.zip
- With Python via pip
  - pip install --upgrade gmsh

Optional software

- OpenFOAM 11 or 12 (instructions: https://openfoam.org/download/)
- Will be used to test the meshes, but not strictly necessary just to make them

Tutorial 1 – Baby's First O-Grid Mesh



Structured Pipe Mesh
w/ Square O-Grid

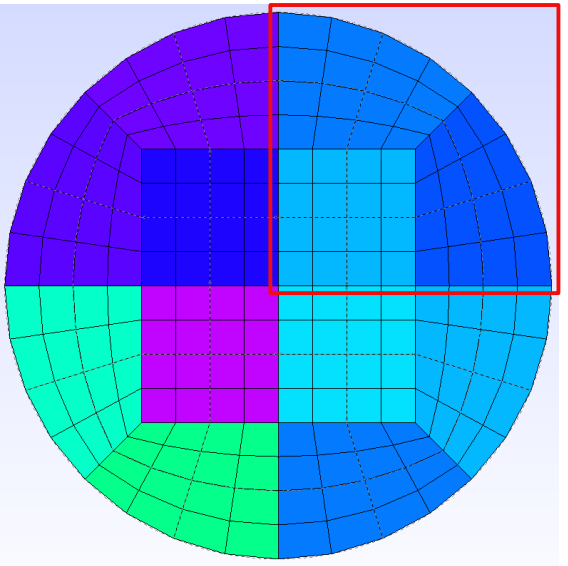
Getting Started

- To create a mesh in GMSH, you must specify the geometry to mesh either by importing a CAD model or creating one in GMSH using a .geo file
  - We will create one in GMSH here
- To get started, follow these steps:

1) Open GMSH via the terminal or app launcher
2) Create a .geo file (File → New then create a .geo file by typing Tutorial1.geo and save it)
3) Open the new file (Modules → Geometry → Edit script)
4) You're ready to make your mesh now :)

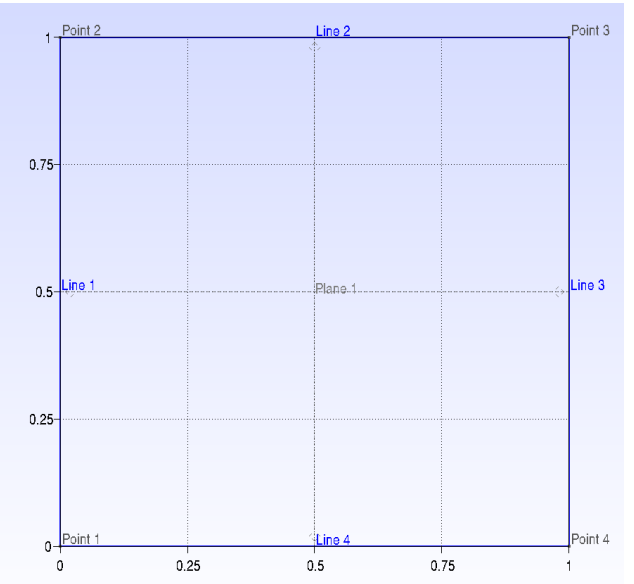
Making Our First Surface Mesh

- We'll be making a pipe with a diameter of 2 and a length of 40
- Let's start by making the surface mesh for the upper-right corner of the pipe inlet (notice symmetry in shape)
- Afterward, we can just copy-rotate our mesh to fill in the other three quarters
- Our inlet will be incident to the xy-plane and centered at the origin (flow in z-axis)



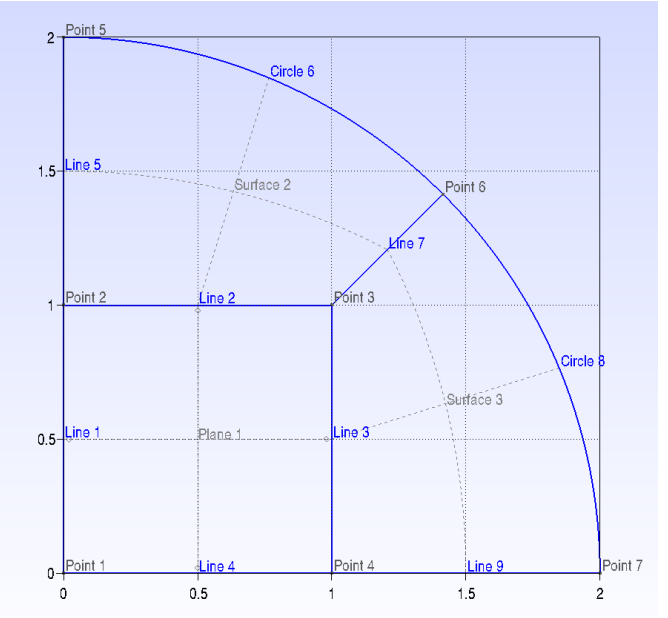
Making Our First Surface Mesh Cont...

- To start, let's make four points representing the vertices of the square in our quarter of the inlet:
  - Point(1) = {0, 0, 0};
  - Point(2) = {0, 1, 0};
  - Point(3) = {1, 1, 0};
  - Point(4) = {1, 0, 0};
- Now let's make four lines from those points
  - Line(1) = {1, 2};
  - Line(2) = {2, 3};
  - Line(3) = {3, 4};
  - Line(4) = {4, 1};
- The last step is to make a Curve Loop that defines the edges of our surface, then create a surface from that:
  - Curve Loop(1) = {1, 2 ,3, 4};
  - Plane Surface(1) = {1};



Making Our First Surface Mesh Cont...

- Let's make the curved wall portions now. First, make points for the vertices on the arcs:
  - Point(5) = {0, 2, 0};
  - Point(6) = {2*Cos(Pi/4), 2*Cos(Pi/4), 0};
  - Point(7) = {2, 0, 0};
- Next, let's make the remaining lines and arcs:
  - Line(5) = {2, 5};
  - Circle(6) = {5, 1, 6}; // Creates an arc starting from point 5 to point 6 centred on point 1 in
  - Line(7) = {6, 3};
  - Circle(8) = {6, 1, 7};
  - Line(9) = {7, 4};
- Finally, let's make the two new surfaces as before:
  - Curve Loop(2) = {5, 6, 7, -2};
  - Surface(2) = {2};
  - Curve Loop(3) = {8, 9, -3, -7};
  - Surface(3) = {3};

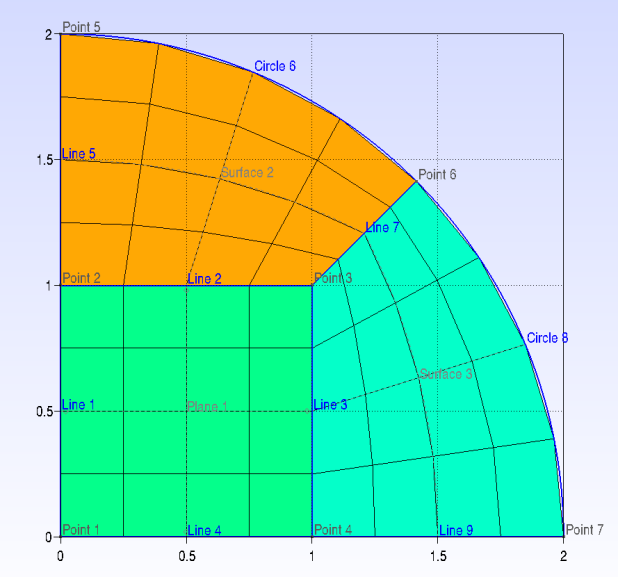


Wait!! Wanna know what this (-) sign is for?
GMSH needs all lines/curves in a loop to be oriented tip-to-tail so that it knows not to cross lines over (like a bowtie). All lines have an orientation from point a to point b, and we can reverse its direction with a (-) sign

Making Our First Surface Mesh Cont...

- Now we're all ready to mesh the surface.
- Let's start by defining some refinement parameters for the angular/center and radial discretizations:
  - dX = 5; // Angular & center
  - dRad = 5; // Radial

This tells GMSH how many nodes, meaning cells in that direction - 1



- Then, we can discretize our lines (and surfaces as a result):
  - Transfinite Curve {1, 2, 3, 4, 6, 8} = dX;
  - Transfinite Curve {5, 7, 9} = dRad;

We'll mesh all the curves opposite each other with the same discretization

  - Transfinite Surface {1, 2, 3};

Now we tell GMSH to discretize the surface from those discretized lines

  - Recombine Surface {1, 2, 3};

Break up the surface into quadrilaterals (default is triangles)

  - Mesh 2;

Create the 2D mesh

Lists go in curly braces {}!

Making Our First Surface Mesh Cont...

- Now to make the other three quarters.
- Start by using this command, which tells GMSH to copy the meshing method along with the geometry:
  - Geometry.CopyMeshingMethod = 1;
- Then, time to copy the mesh:
  - topLeft[] = Rotate{{0, 0, 1}, {0,0,0}, Pi/2} {Duplicata{Surface{1,2,3};}};
  - bottomLeft[] = Rotate{{0, 0, 1}, {0,0,0}, Pi} {Duplicata{Surface{1,2,3};}};
  - bottomRight[] = Rotate{{0, 0, 1}, {0,0,0}, 3*Pi/2} {Duplicata{Surface{1,2,3};}};
  - Mesh 2;
  - Coherence;

This action produces a list that can be stored in an object[]

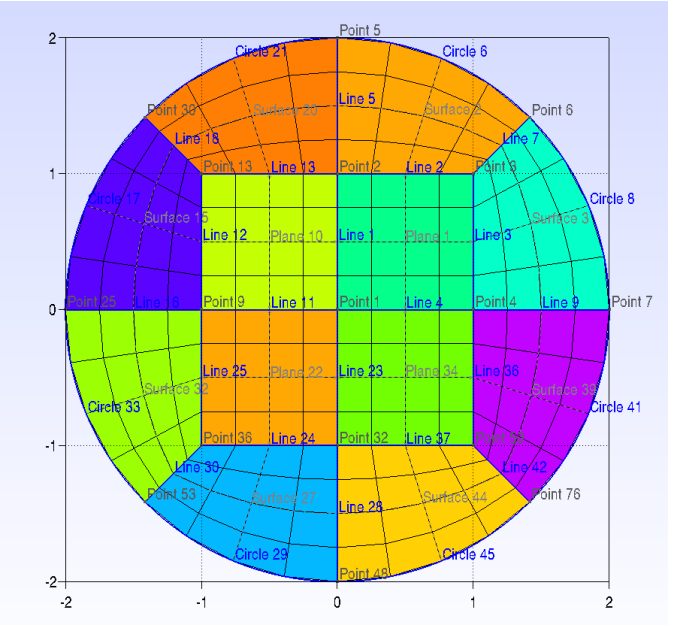
Axis of rotation

Angle of rotation

Rotates a copy instead of the original

Point around which to rotate

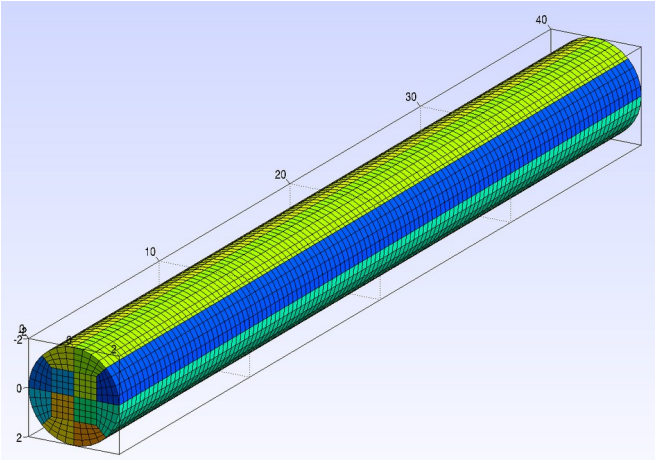
Tells GMSH to merge overlapping lines, nodes, etc.



For more GMSH commands, go here: https://gmsh.info/doc/texinfo/gmsh.html
Be careful! GMSH command syntax is *very specific* and somewhat inconsistent.

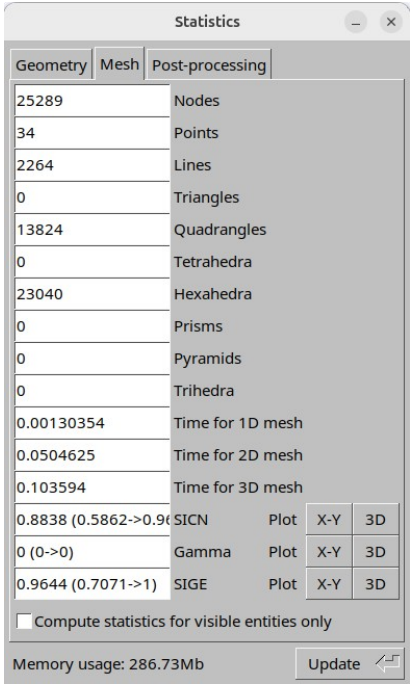
Making the Volume Mesh

- It’s finally time to make the volume mesh.
- Here, we’ll simply extrude our 2D mesh in the z-direction:
  - pipe[] = Extrude{0, 0, 40} {Surface{:}; Layers{120}; Recombine;};
  - Mesh 3;
- Congrats! You’ve made a 3D mesh!
- We could now export the mesh for use in OpenFOAM 12, but we can do better than this mesh, which is under-resolved near the walls and has heavily skewed cells around the corners of the square
- Let’s go over some basic mesh quality checks in the next slide

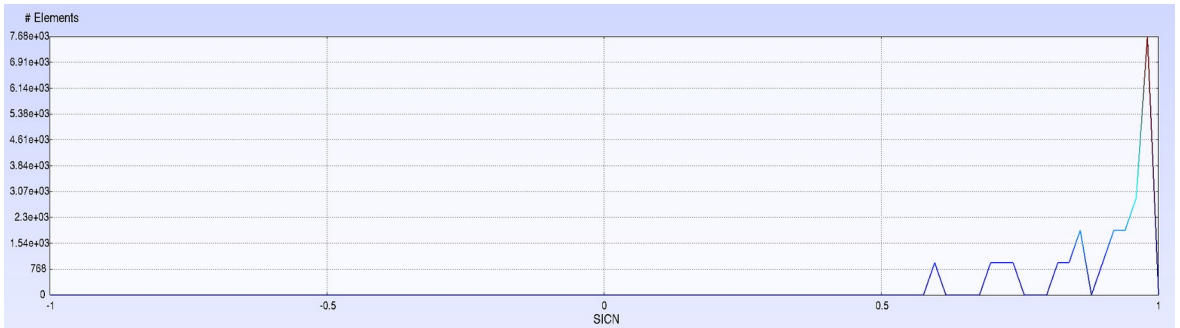


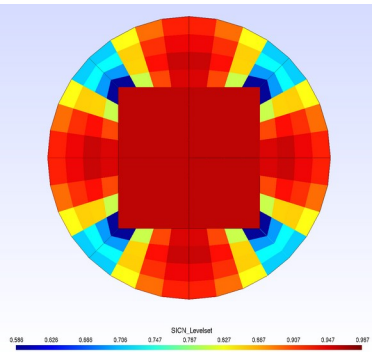
Assessing Mesh Quality in GMSH

- Go to Tools → Statistics
- In the Mesh tab, click Update
- For hexahedral meshes like this one, we will look at SICN and SIGE
  - SICN = Signed inverse condition number (lower = harder to solve for this element)
  - SIGE = Signed inverse gradient error
  - For both of these, lower = worse
  - We will generate histograms of these indicators (X-Y Pot) and see where in the mesh the worst elements are (3D Plot)
- There are other quality metrics such as skewness, orthogonality, etc, and these can be obtained from OpenFOAM after import.

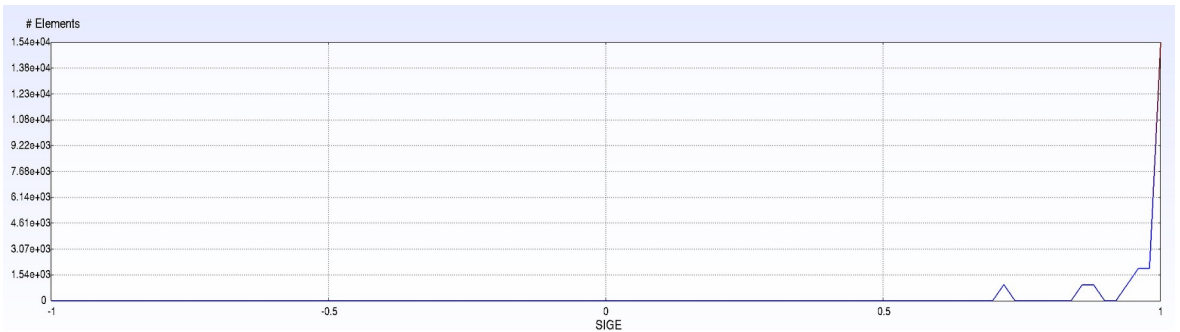


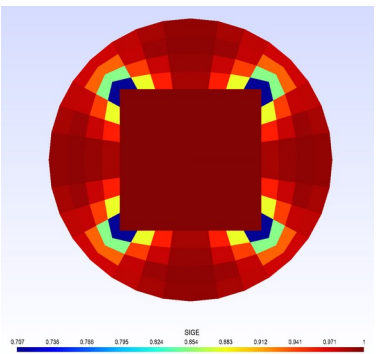
Assessing Mesh Quality in GMSH





The SICN metric will show you the quality of the surface elements in the 3D view, so you need to apply a clipping plane (Tools→ Plugins→ CuttingPlane) to see the inside of the mesh



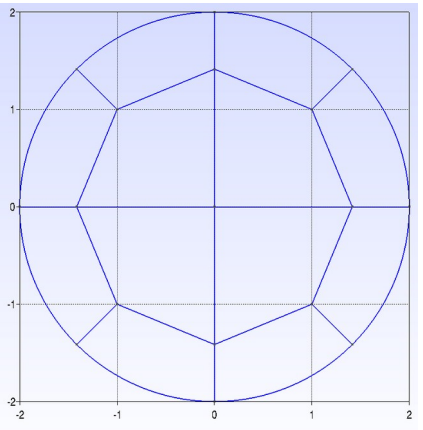


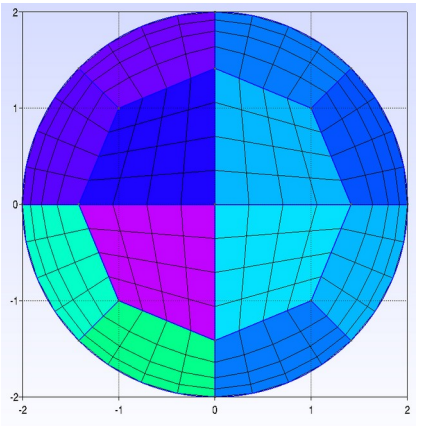
- Clearly, the sharp corners are creating some lower-quality elements, and this doesn't show us the lack of wall-resolution

Making a Higher-Quality O-Grid

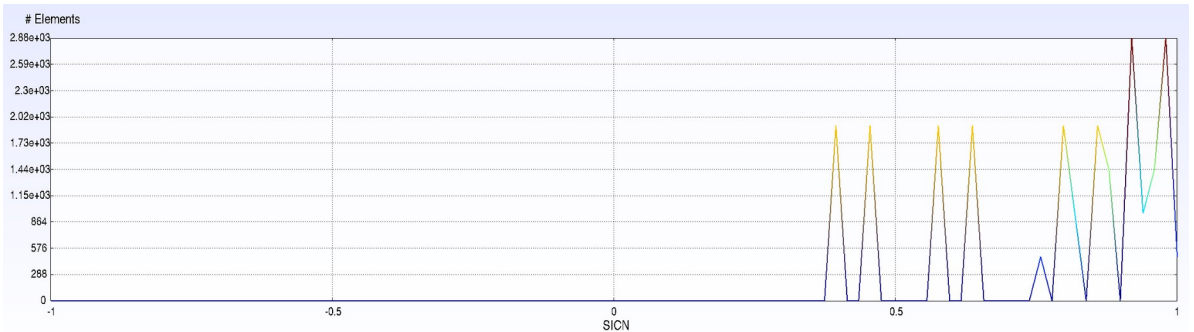
- First, let's even out the distance of the inner grid to the walls:
  - Point(1) = {0, 0, 0};
  - Point(2) = {0, 1/Cos(Pi/4), 0};
  - Point(3) = {1, 1, 0};
  - Point(4) = {1/Cos(Pi/4), 0, 0};
- Then, let's add a progression to our radial discretization to make the grid spacing non-uniform
  - Transfinite Curve {1, 2, 3, 4, 6, 8} = dX;
  - Transfinite Curve {-5, 7, 9} = dRad Using Progression 1.4;
- Now let's check the mesh quality again

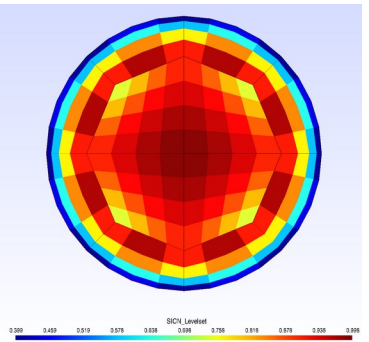
-5 otherwise progression goes opposite way (lines are oriented)

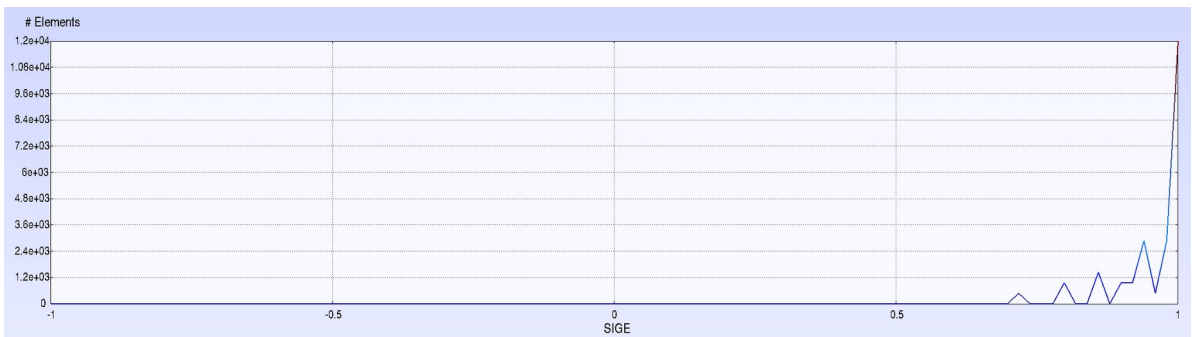


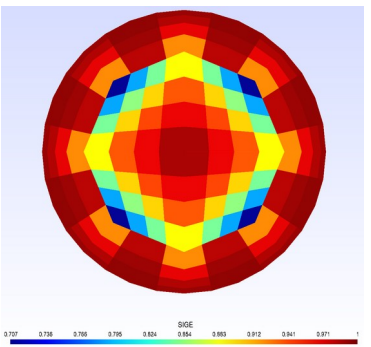


New Mesh Quality









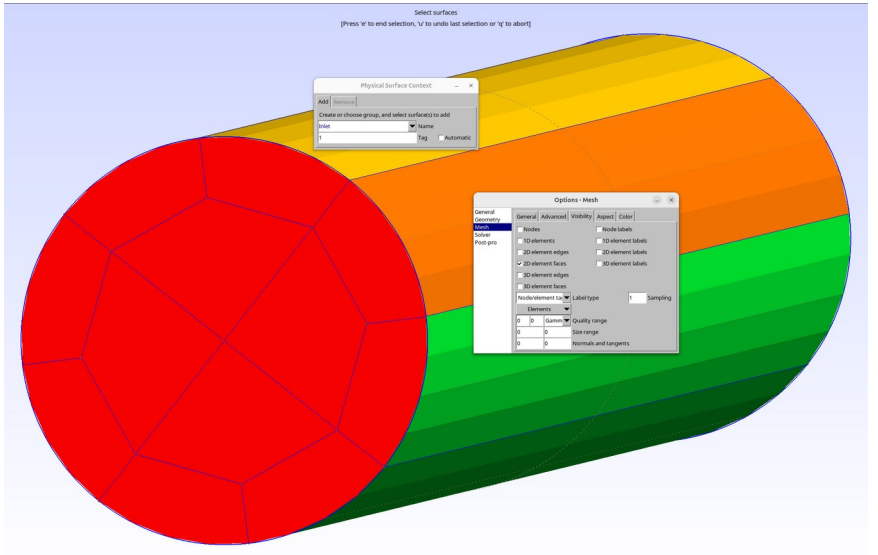
- Wait a second! This looks worse than the square mesh! >:(

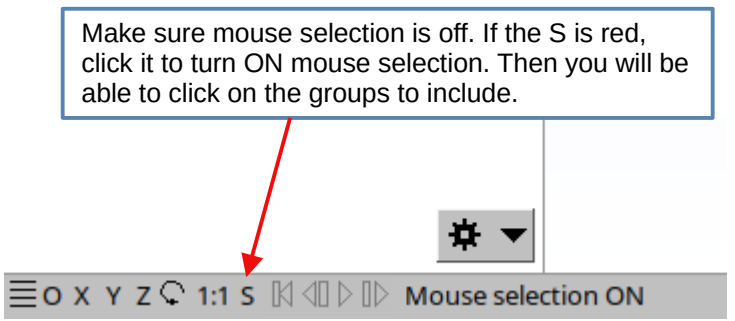
The (Sometimes) Deceiving Nature of Quality Metrics

- In general, quality metrics are reserved for telling you when there is a problem with your mesh
  - For example, if a Jacobian of a cell is negative, that cell is invalid and will cause the CFD simulation to diverge
- Specific quality metrics are useful for a given solver, but this is highly application-specific
  - E.g., high-non-orthogonality in cells may require non-orthogonal corrections in the solver
- Quality metrics are generally not good at telling you if a given mesh is better than another (aside from one being invalid, etc.)
- To further illustrate this, we will import the mesh to OpenFOAM, check the quality metrics there, and run a simulation and compare the results.

Importing the Mesh

- First, we have to define physical surfaces and volumes in Gmsh
  - These will become boundary groups and the mesh volume, respectively
  - Geometry → Physical groups → Surface/Volume
  - You may have to toggle the visibility of some geometry groups to select the appropriate ones (Tools → Options → Geometry/Mesh → Visibility)
  - You may have to toggle on the 2D element faces for the physical surface creation, then toggle them off for the volume creation
- We will define four groups as follows:
  - Physical Surface("Inlet", 1) = {27, 44, 22, 34, 39, 3, 1, 2, 20, 10, 15, 32};
  - Physical Surface("Outlet", 2) = {177, 133, 155, 243, 199, 221, 309, 265, 67, 89, 111, 287};
  - Physical Surface("Walls", 3) = {164, 146, 230, 80, 278, 98, 296, 212};
  - Physical Volume("Pipe", 4) = {8, 9, 7, 12, 5, 10, 4, 11, 6, 1, 3, 2};
  - *Note that the numbering of your surfaces/volumes may not be the same, so please select the appropriate ones when you define the physical groups





Importing the Mesh

- Finally, we will save the mesh. First, type the following anywhere in your .geo file
  - Mesh.MshFileVersion = 2.2;
  - This will set the file format to v2 ASCII, the only format OpenFOAM can read.
  - Save your file and reload the script in the GUI. Finally, go to Mesh then click Save.
- Next, we will import the mesh to OpenFOAM
  - Move your new .msh file to the folder containing the OpenFOAM case
  - Open a terminal and run the command “gmshToFoam [your file].msh”
  - Run “checkMesh” to ensure the mesh is free of errors
  - Congrats! You are now ready to run your simulation.

Comparing Meshes in OpenFOAM

- Here are some checkMesh outputs for various configurations of the mesh

```
Checking geometry...
    Overall domain bounding box (-2 -2 0) (2 2 40)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Max cell openness = 1.60691e-16 OK.
    Max aspect ratio = 3.13776 OK.
    Minimum face area = 0.0375662. Maximum face area = 0.130716.  Face area magnitudes OK.
    Min volume = 0.0125208. Max volume = 0.0306605.  Total volume = 499.431.  Cell volumes OK.
    Mesh non-orthogonality Max: 22.6432 average: 5.50689
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 1.12541 OK.
    Coupled point location match (average 0) OK.

Mesh OK.
```

Square core w/out bias

```
Checking geometry...
    Overall domain bounding box (-2 -2 0) (2 2 40)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Max cell openness = 1.37974e-16 OK.
    Max aspect ratio = 4.42406 OK.
    Minimum face area = 0.0240298. Maximum face area = 0.130716.  Face area magnitudes OK.
    Min volume = 0.00980044. Max volume = 0.0386234.  Total volume = 499.431.  Cell volumes OK.
    Mesh non-orthogonality Max: 34.7364 average: 9.95385
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 0.476661 OK.
    Coupled point location match (average 0) OK.

Mesh OK.
```

Square core w/ bias

```
Checking geometry...
    Overall domain bounding box (-2 -2 0) (2 2 40)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Max cell openness = 1.90849e-16 OK.
    Max aspect ratio = 3.73626 OK.
    Minimum face area = 0.0240298. Maximum face area = 0.138488.  Face area magnitudes OK.
    Min volume = 0.0103358. Max volume = 0.0379888.  Total volume = 499.431.  Cell volumes OK.
    Mesh non-orthogonality Max: 18.9794 average: 4.54035
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 0.617182 OK.
    Coupled point location match (average 0) OK.

Mesh OK.
```

Octagonal core w/ bias

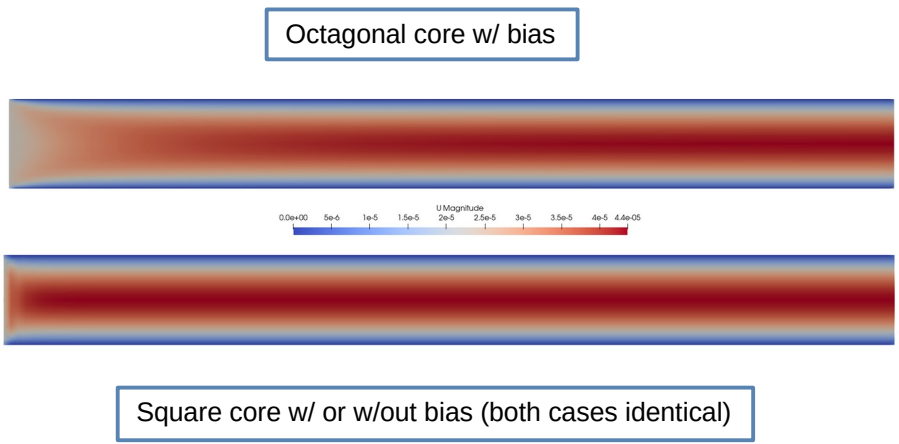
Save for the max skewness, the octagonal core mesh is worse in every quality metric here.

Let’s see if that holds up to a simple benchmark.

Mesh Benchmark – Developing Laminar Flow

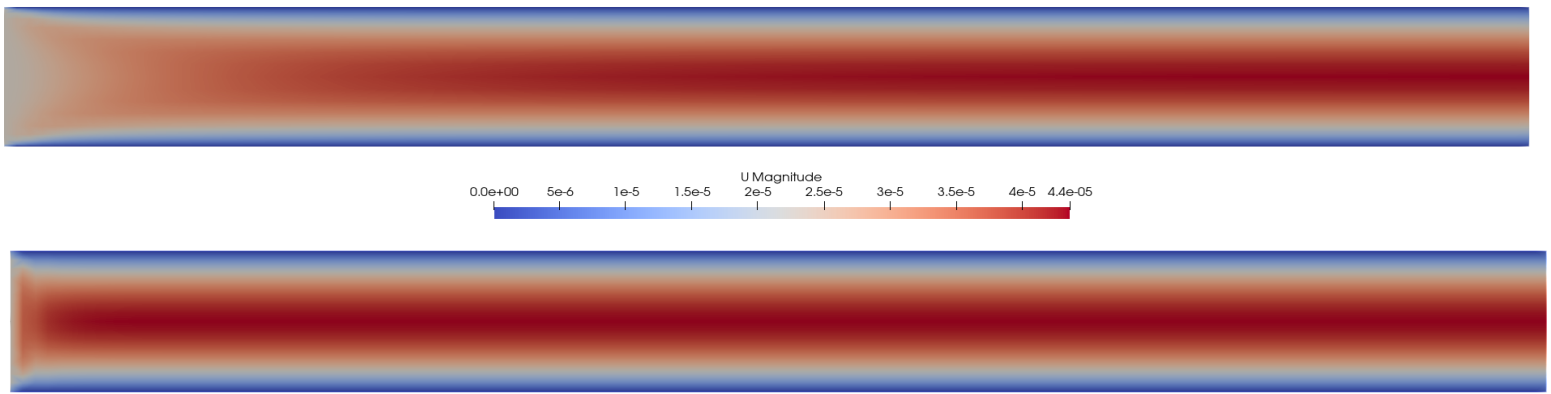
- The benchmark we will be using is a simple developing laminar flow simulation at Re= 100 (from an inviscid inlet condition).
- The case is already set up. Simply run “foamRun” to execute.
- If you run all three geometries, you will obtain the following:

Mesh configuration	Number of iterations required for convergence
Octagonal core w/ bias	299
Square core w/out bias	651
Square core w/ bias	715



Mesh Benchmark – Developing Laminar Flow

- As we can see, benchmarking reveals that the better mesh is the octagonal core o-grid, despite the mesh metrics indicating the opposite
- The square core o-grids even fail to accurately capture the developing region of the flow
- In reality, the corners of the square grid are problematic for convergence, and poorer resolution near the wall makes for difficulty capturing steep gradients (as are present in the developing region)
- The result is a mesh that causes a “skip” to developed flow and requires more iterations to converge
- In short, don’t rely solely on quality metrics!



Octagonal core w/ bias

Square core w/ or w/out bias (both cases identical)

Thank You! :)