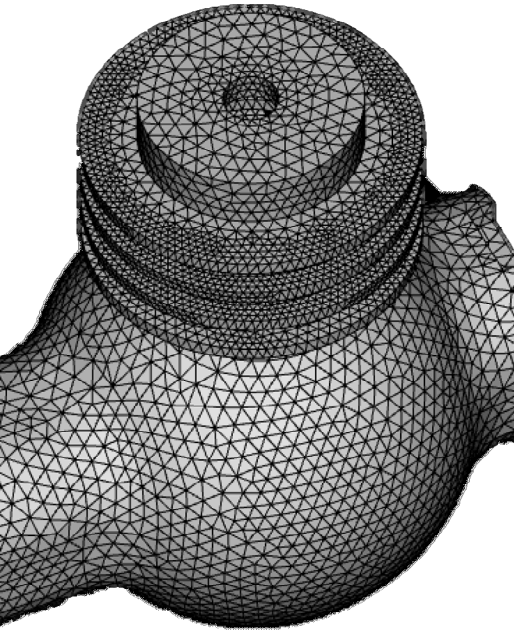
SAND2017-7232C



TetMesh Scaling for Solution Verification

W. Roshan Quadros and Brian Carnes

July 17-20, 2017

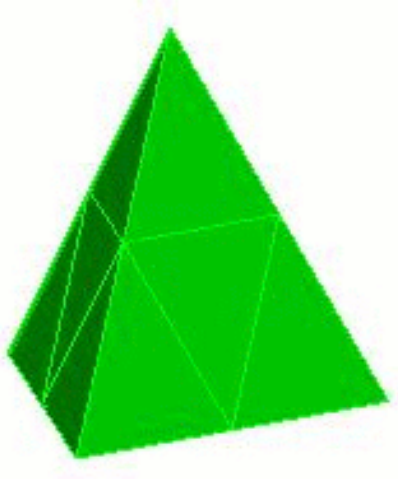
14th US National Congress on Computational Mechanics

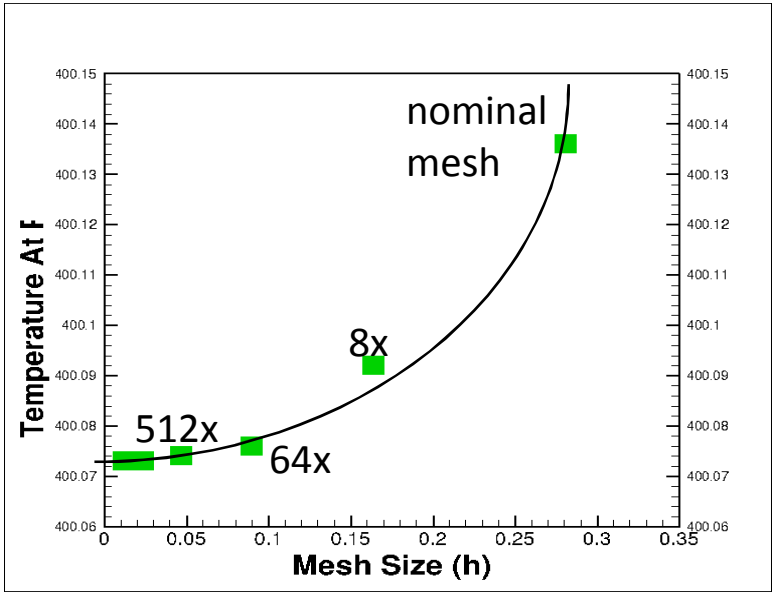
Montreal, Canada

Sandia National Laboratories is a multi mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

UMR is Inefficient in Solution Verification

- Uniform Mesh Refinement (UMR) increases d.o.f. and CPU time exponentially

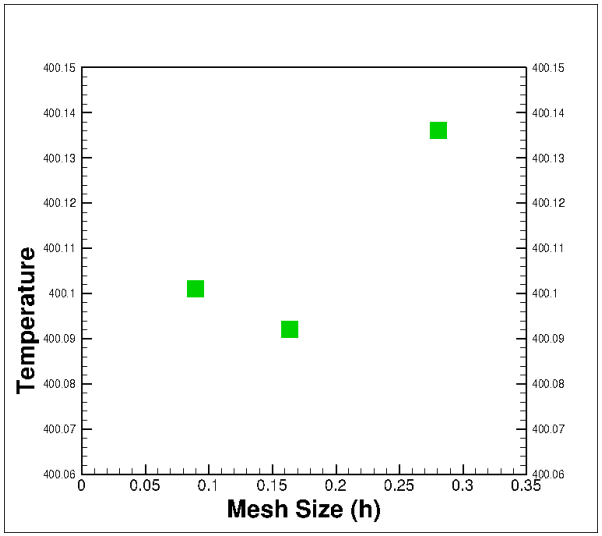




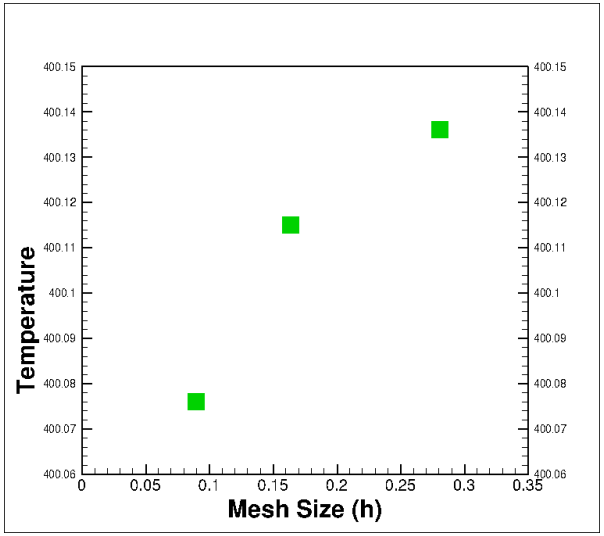
UMR level	d.o.f	Solver CPU time
1	8X	16X
2	64X	272X
3	512X	4,368X
4	4096X	69,904X

Solution Verification using UMR is Failure Prone

- Need at least 3 meshes, more preferably



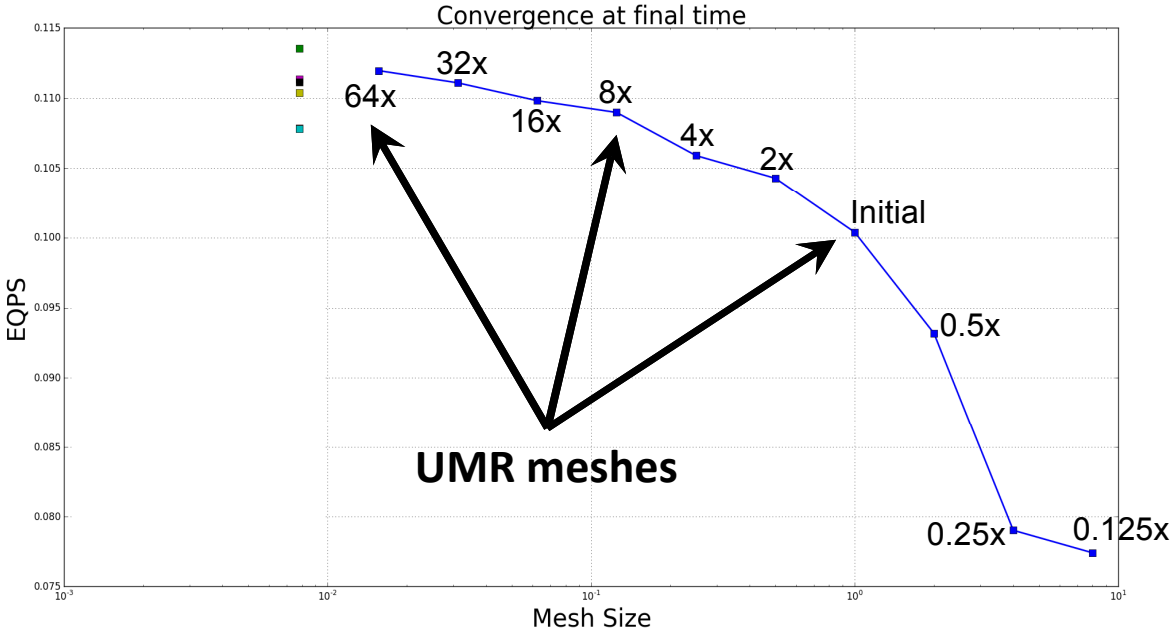
Data oscillation - yields undefined convergence rate

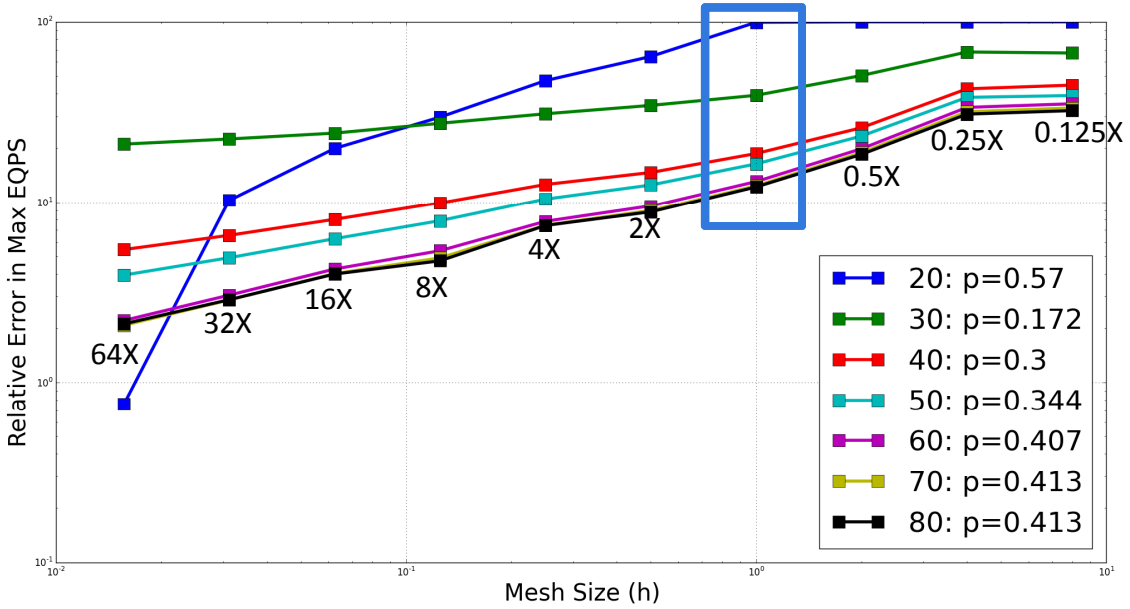


Data is diverging – yields negative convergence rate

Advantages of Mesh Scaling

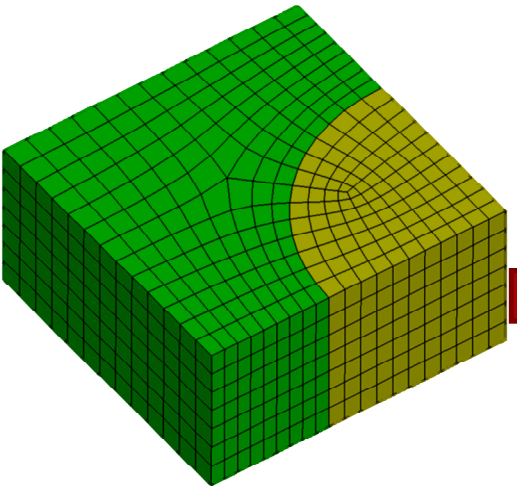
- More points on convergence plots
- Extrapolation with different combinations of points
- More trust in error estimates
- 64% saving in CPU hours over UMR for equivalent error estimate





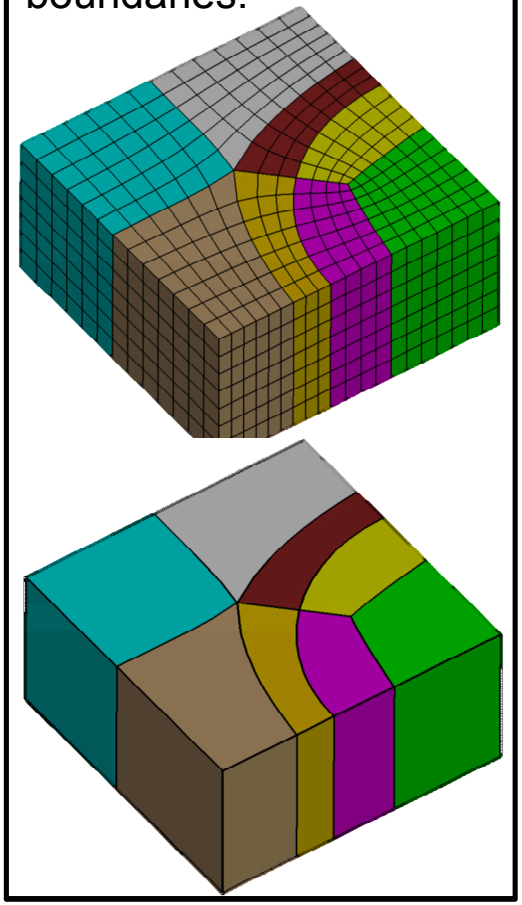
Related Work: HexMesh Scaling

Staten, Matt, Brian Carnes, Corey McBride, Clint Stimpson, Jim Cox, "Mesh scaling for affordable solution verification", *Proceedings, 25th International Meshing Roundtable*, September 26-30 2016

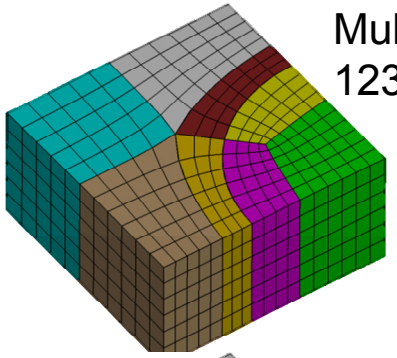


Initial Mesh
1869 hexes

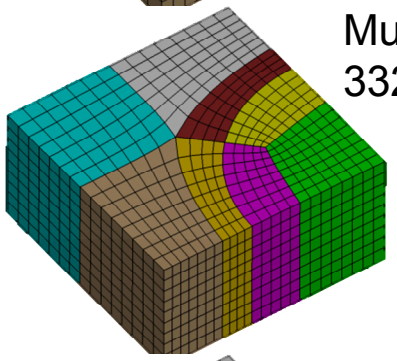
Block Decomposition is determined by both mesh irregularities and volume boundaries.



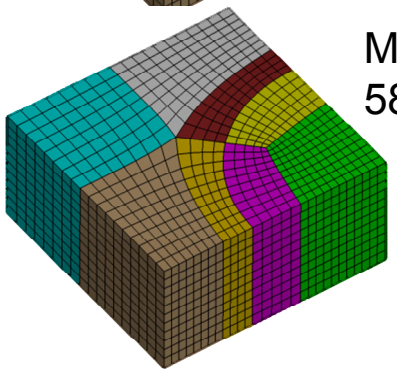
STEP 1: Extract Block Decomposition



Multiplier 0.667
1230 hexes



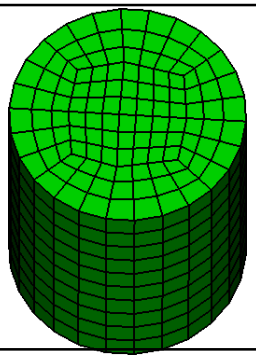
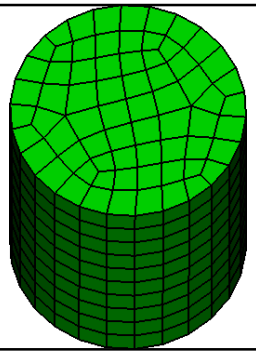
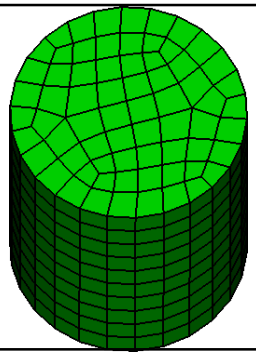
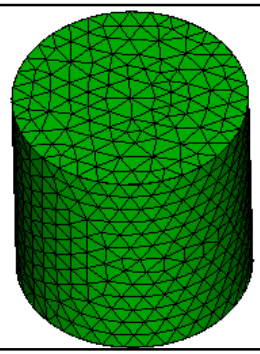
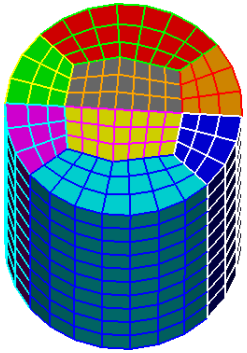
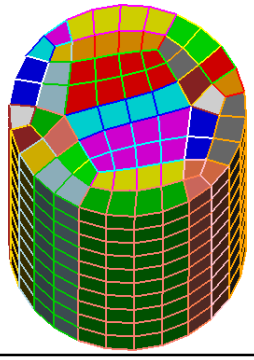
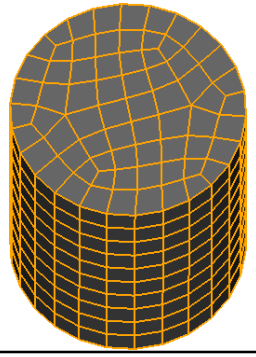
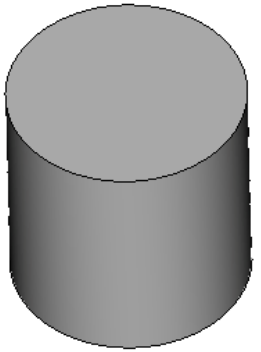
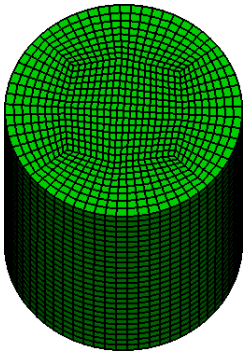
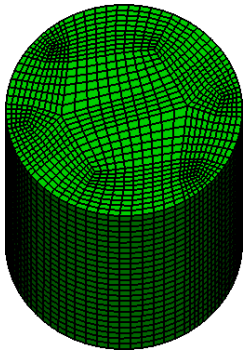
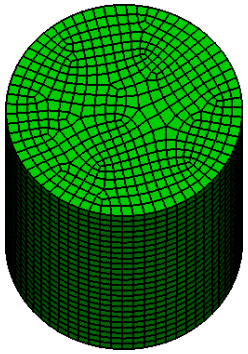
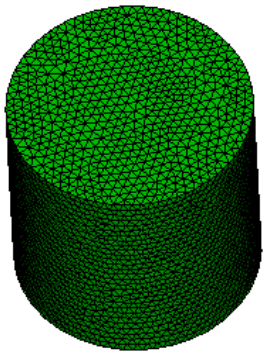
Multiplier 1.75
3321 hexes



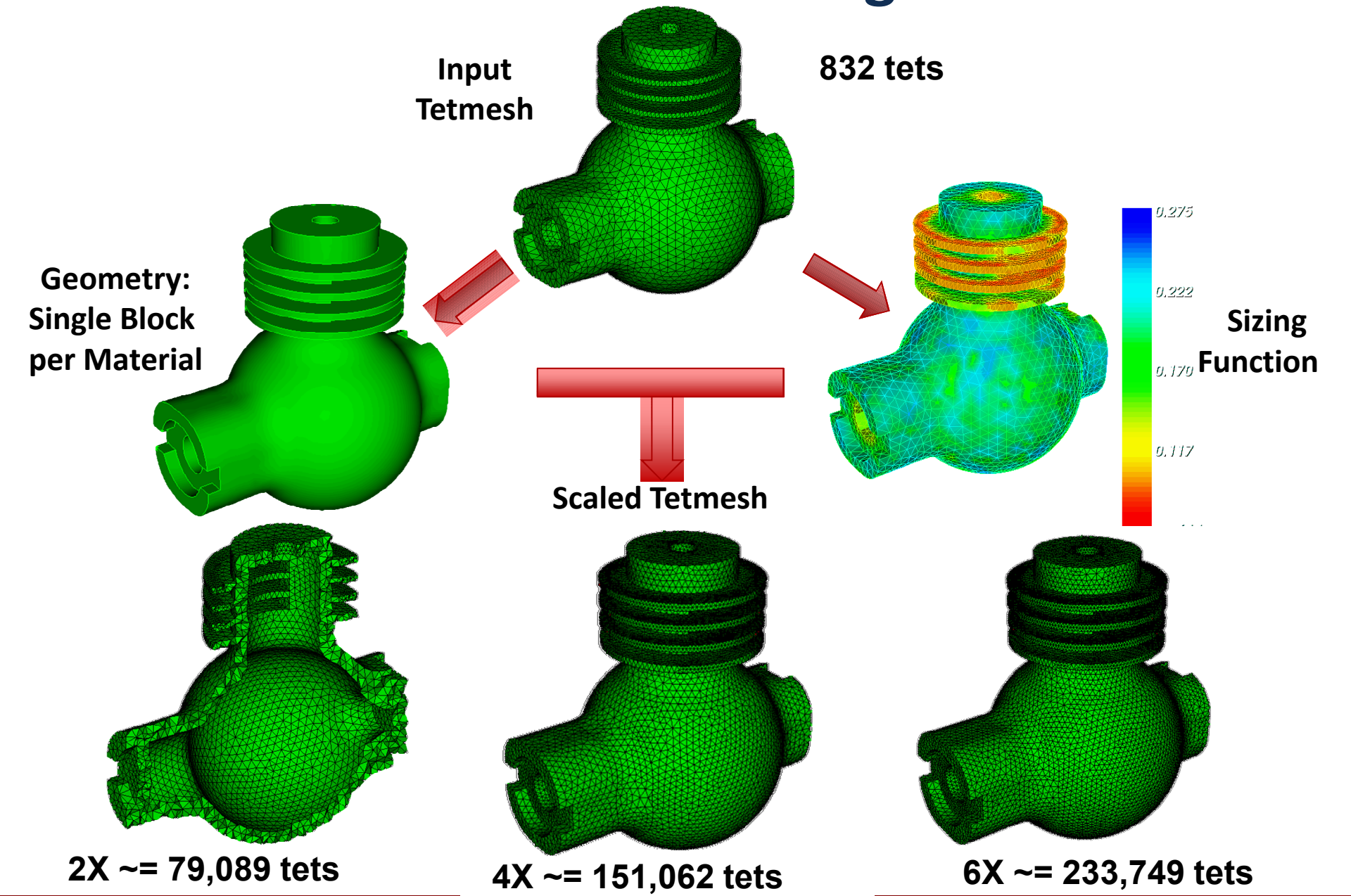
Multiplier 3.0
5841 hexes

STEP 2: Remesh Blocks

Structured Hexmesh vs Unstructured Tetmesh

	More structure & Less Blocks	Less Structure & More Blocks	Less Structure & Single Block	Unstructured & Single Block
Original Mesh				
Block Decomposition				
16X				

Overview of TetMesh Scaling

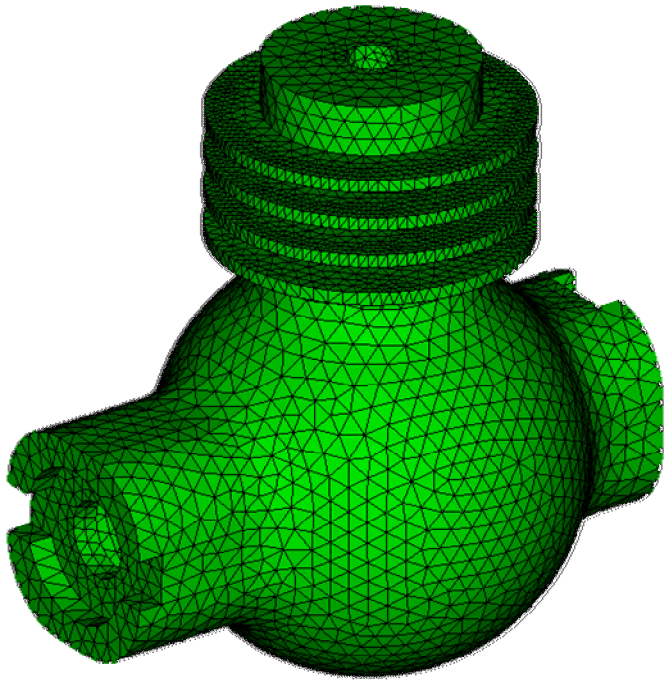


Overview of TetMesh Scaling

- STEP 1 : Import Tetmesh & get Multiplier
- STEP 2 : Build or Import Geometry
- STEP 3 : Build Sizing Function from input Tetmesh
- STEP 4 : Remesh MBG using Sizing Function

STEP 1: Import Tetmesh & get Multiplier

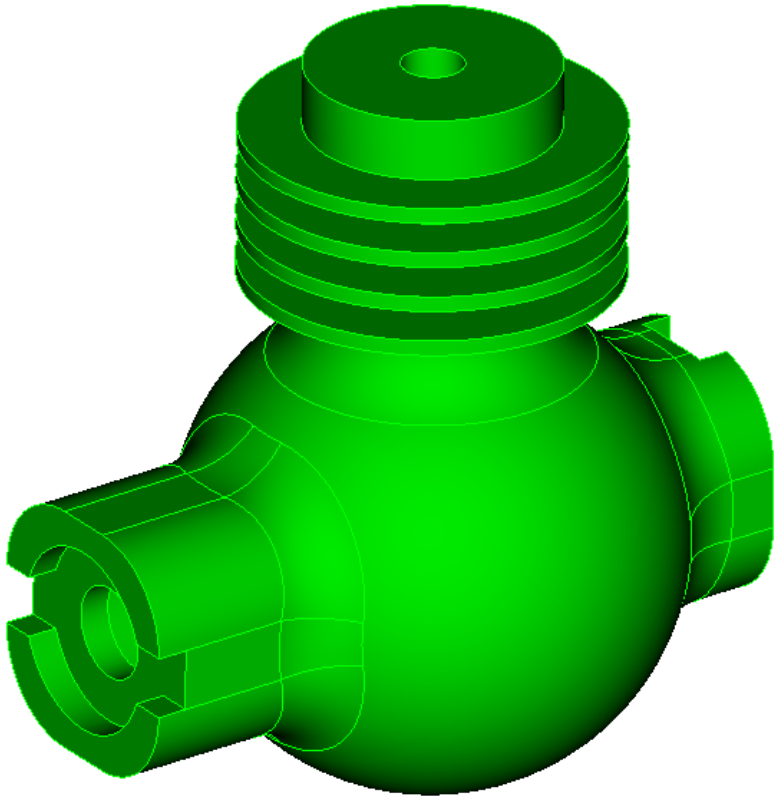
- CUBIT imports following mesh formats
  - 2D Exodus II Files
  - Exodus II Files
  - Patran Files
  - I-DEAS Files
  - Abaqus Files
  - Nastran Files
  - Fluent Files



- Get scaling Multiplier ‘M’
  - 10,000 tets scaled by 3X results in roughly 30,000 tets
  - Different multiplier on different geometric entities

STEP 2 : Import or Build Geometry

- CUBIT imports following geometry formats
  - ACIS Files
  - FASTQ Files
  - STEP Files
  - IGES Files
  - Facet Files
  - STL Files
  - Other Formats



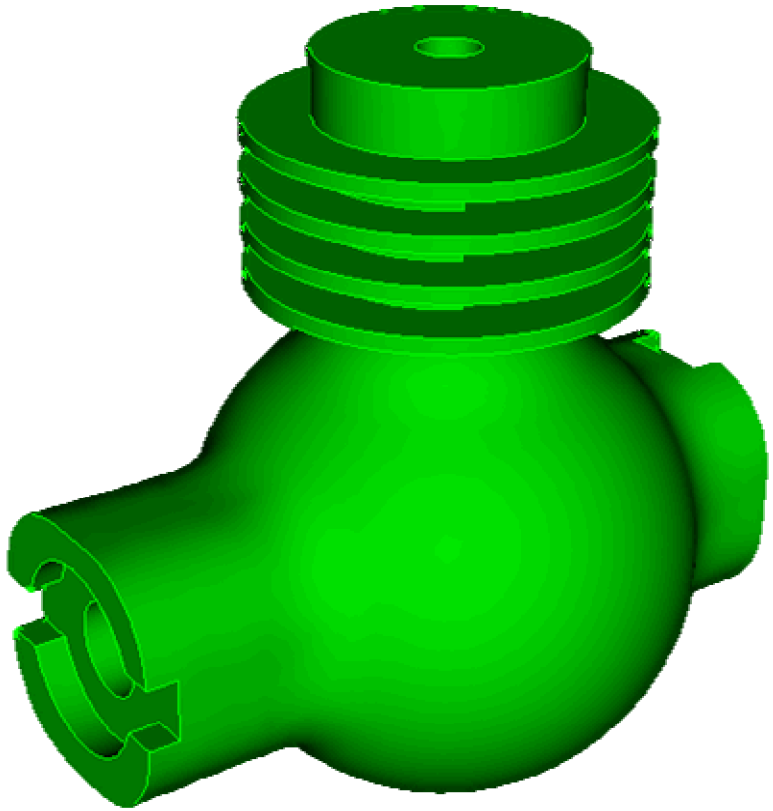
STEP 2 : Import or Build Geometry

- Build Mesh Based Geometry

- Exodus II contains a mesh representation that may include 3D elements, 2D elements, 1D elements and even 0D elements

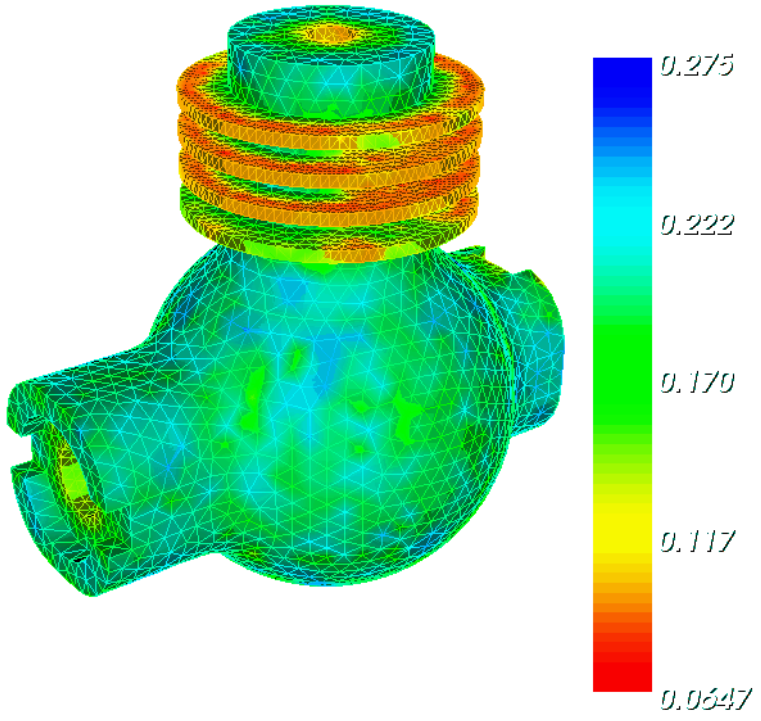
- Create a complete solid model using the mesh faces as the basis for the surface representations

- Assign appropriate ownership of the mesh entities to their geometry owners



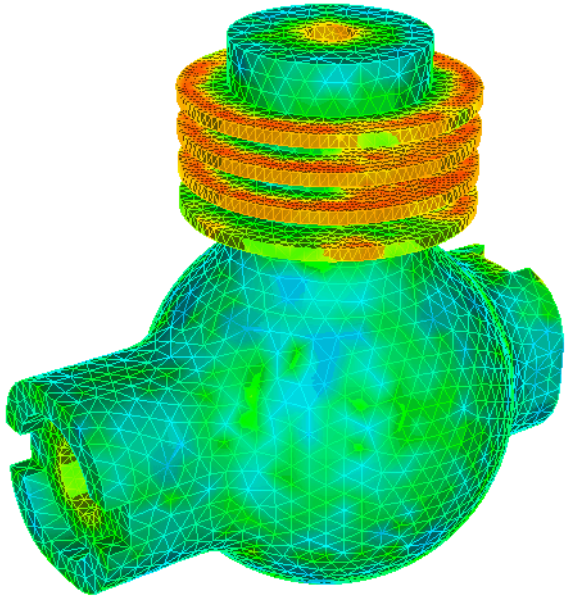
Overview of TetMesh Scaling

- STEP 1 : Import Tetmesh & get Multiplier
- STEP 2 : Build or Import Geometry
- STEP 3 : Build Sizing Function from input Tetmesh
- STEP 4 : Remesh MBG using Sizing Function



STEP 3: Build Sizing Function

- Use input mesh as Background mesh to store size field
- Calculate mesh size at each node to capture mesh characteristics
  - User specified mesh size
  - User specified mesh gradation
- Combine different multipliers on volumes & surfaces
  - Scale size field associated with geometric entities using multipliers
  - Use gradient limiting to avoid abrupt transition



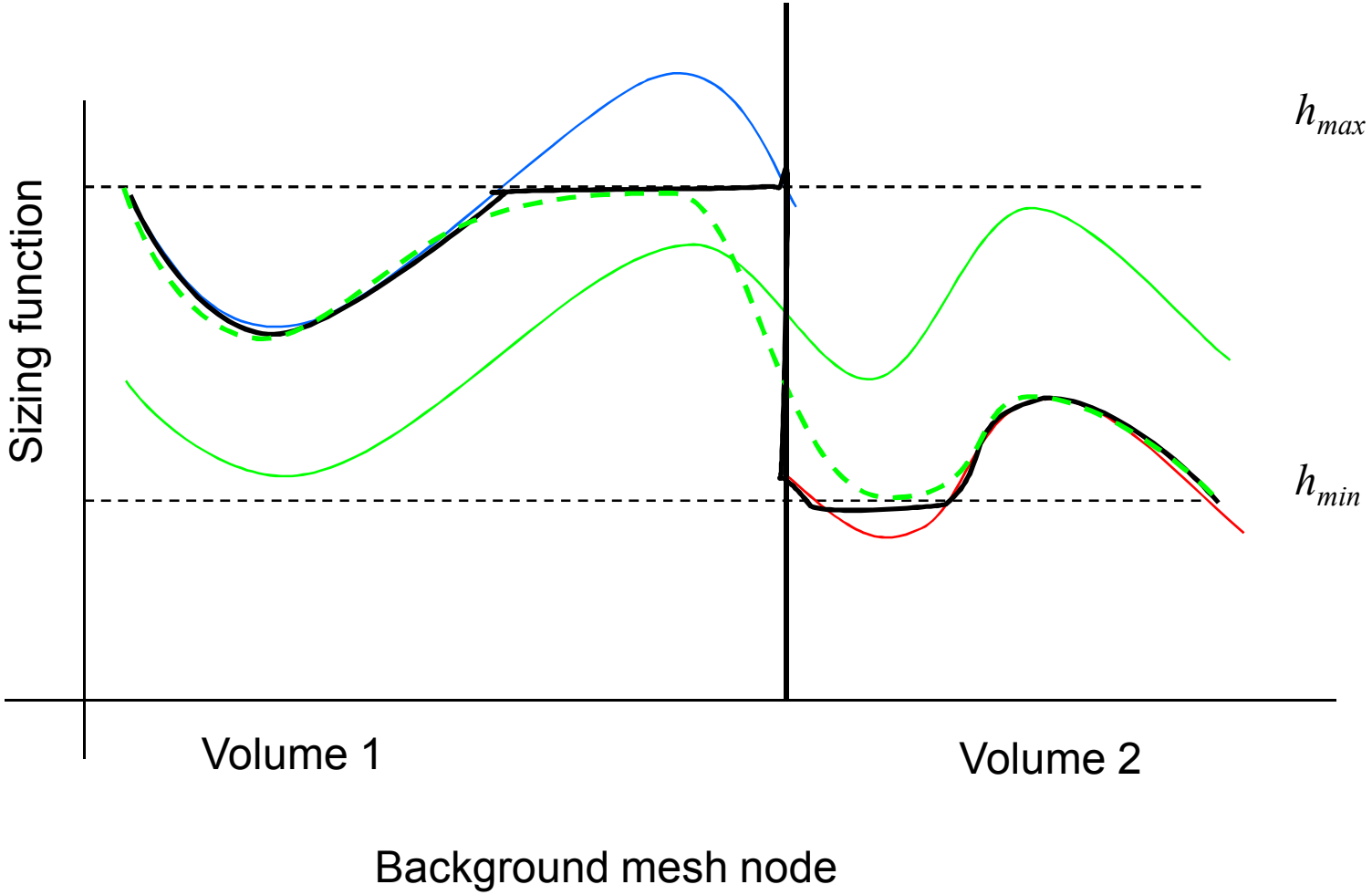
Build initial size field at nodes

$h(\mathbf{x_i}) = \frac{\sum_{j=0}^{N} \|e_{ij}\|}{N}$, where $\mathbf{x_i}$ is coordinates of node i and e_{ij} are edges incident on node i

$\mu = \frac{1}{\sqrt[3]{M}}$,where M is user specified multiplier

$h(\mathbf{x_i}) = h(\mathbf{x_i}) \times \mu$

Combining Different Multipliers



Limiting Gradient of Combined Sizing Function

$h_0(\mathbf{x})$ is the combined sizing function between h_{min} and h_{max}

The final sizing function $h(\mathbf{x})$ should satisfy

For every point $\mathbf{x} \in \Omega$

(1) $h(\mathbf{x}) \le h_0(\mathbf{x})$

(2) $\left|\nabla h(\mathbf{x})\right| \le g$

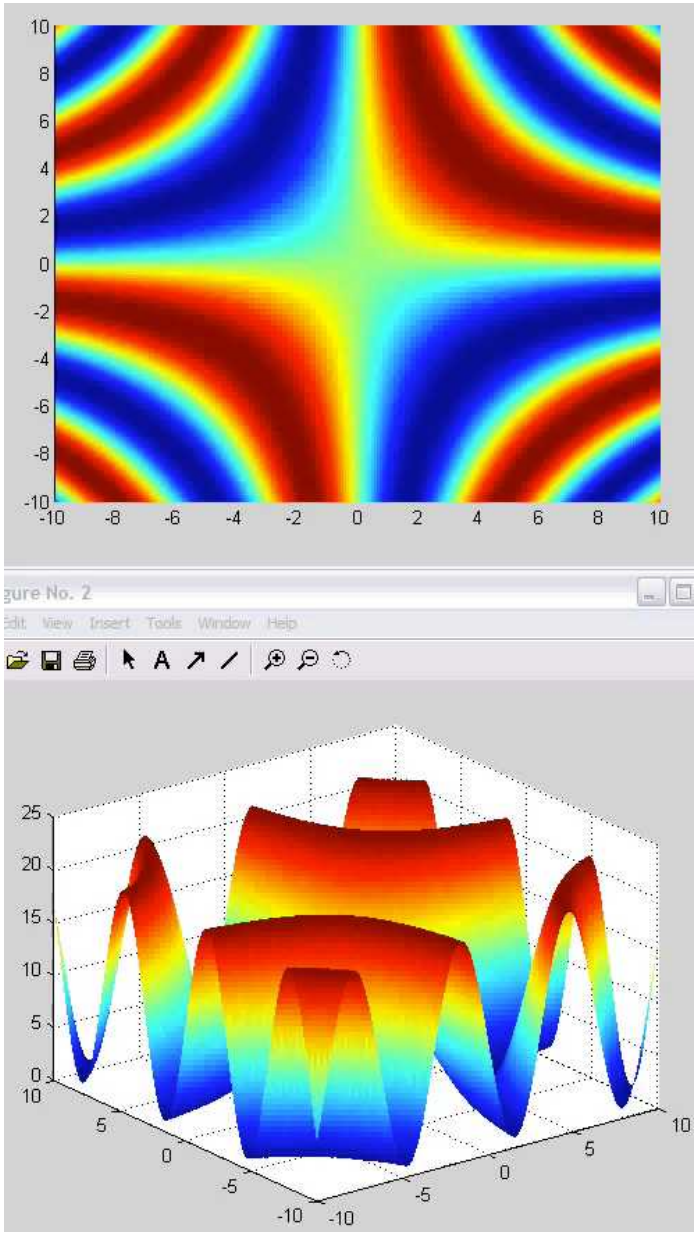
(3) $h(\mathbf{x})$ is as large as possible

Hamilton-Jacobi Method [Persson, 2004]

$$\frac{\partial h}{\partial t} + \left|\nabla h\right| = \min\left(\left|\nabla h\right|, g\right)$$

$$h^{n+1} = h^n + \Delta t\left(\min\left(\nabla^+, g\right) - \nabla^+\right)$$

where $\Delta t = \dfrac{\min(\Delta \mathbf{x})}{2}$ ∇^+ is discrete gradient

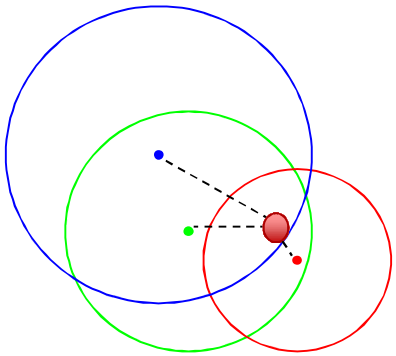


STEP 4: Remesh using Sizing Function

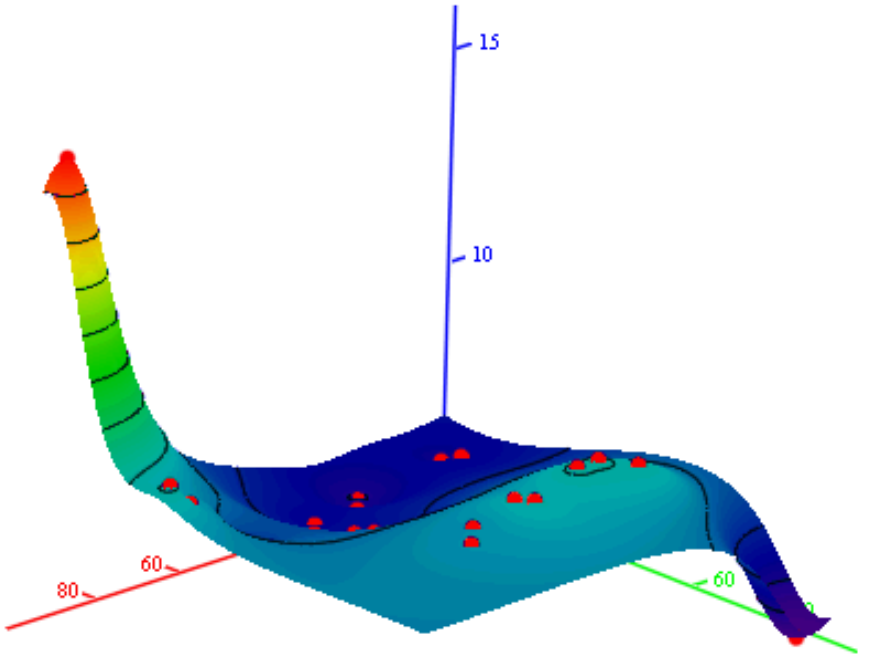
- Remesh MBG or Other geometry definition
  - Using bottom-up meshing
  - Common Sizing Function
- Interpolate mesh size on background mesh
  - Use size at nearest n-neighbor nodes of global background mesh
  - Use size at nodes of associated local mesh entities

Interpolate Size at a Point

Inverse distance weighting (quadratic)



$$h(\mathbf{x}) = \sum_{i=1}^{N} f_i(d_i) \times W_i \text{ where } W_i = \frac{\frac{1}{d_i^{\,t}}}{\sum_{j=1}^{j=m} \frac{1}{d_j^{\,t}}}, \; t = 2$$



Interpolated size on background mesh

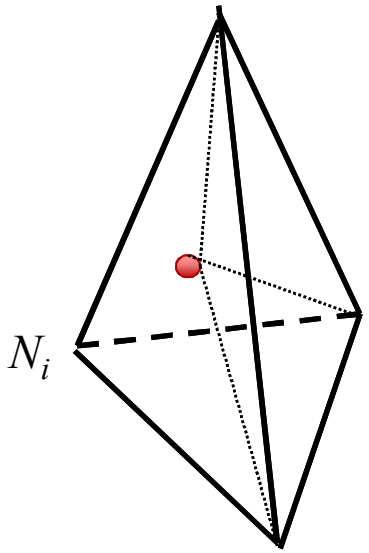
d_i : Distance between i^{th} node and input point **x**

f_i : Local sizing function of i^{th} node

Interpolate size at a point inside a tet

Use Barycentric coordinates

$$s(x) = \sum_{i=1}^{i=4} s(N_i) W_i \qquad W_i = \frac{V_i}{V}$$



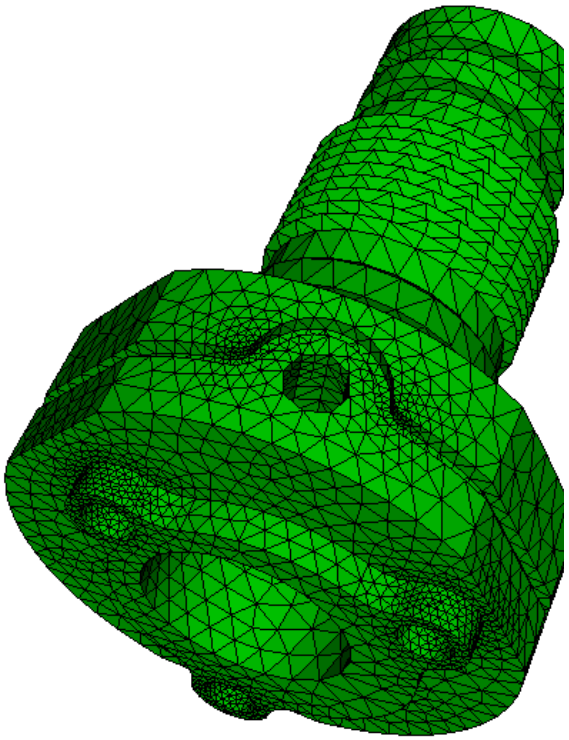
$s(x)$: Size at point x

W_i : Barycentric coordiantes

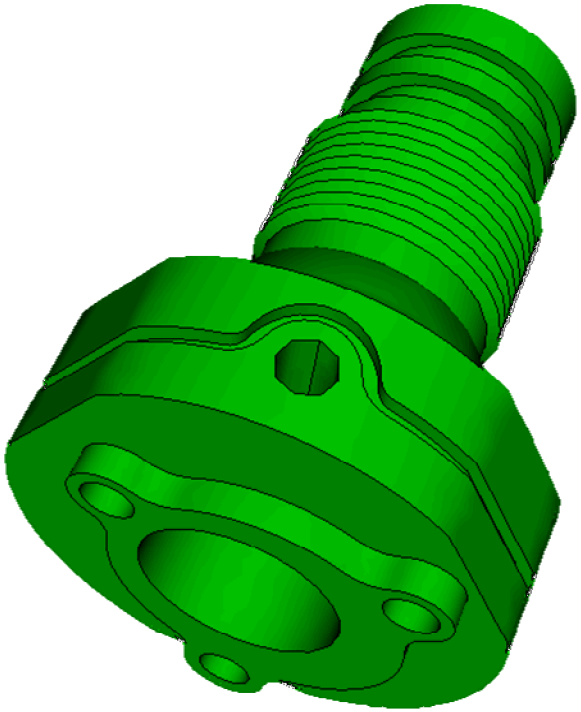
V_i : Volume of subdivided tet opposite to node node N_i

V : Volume of tet

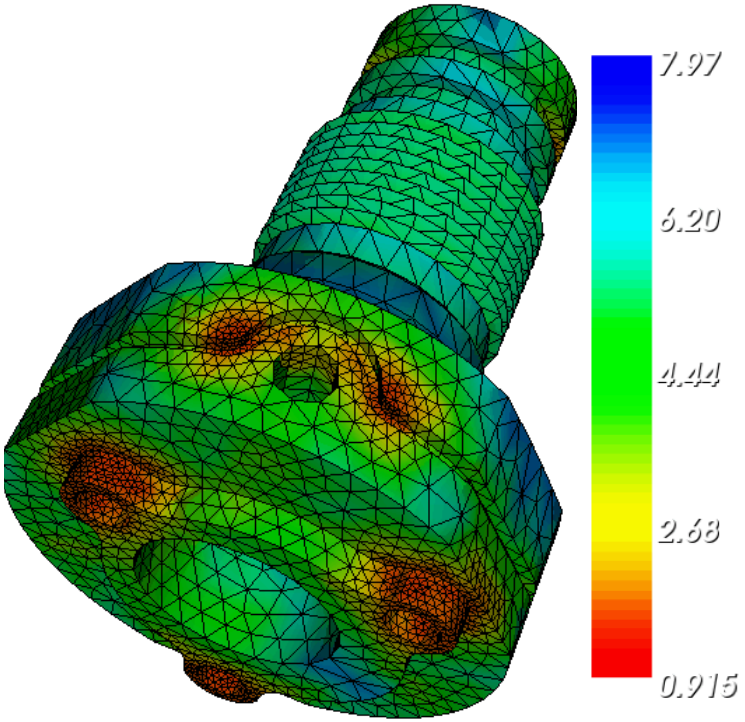
Result



Input TetMesh

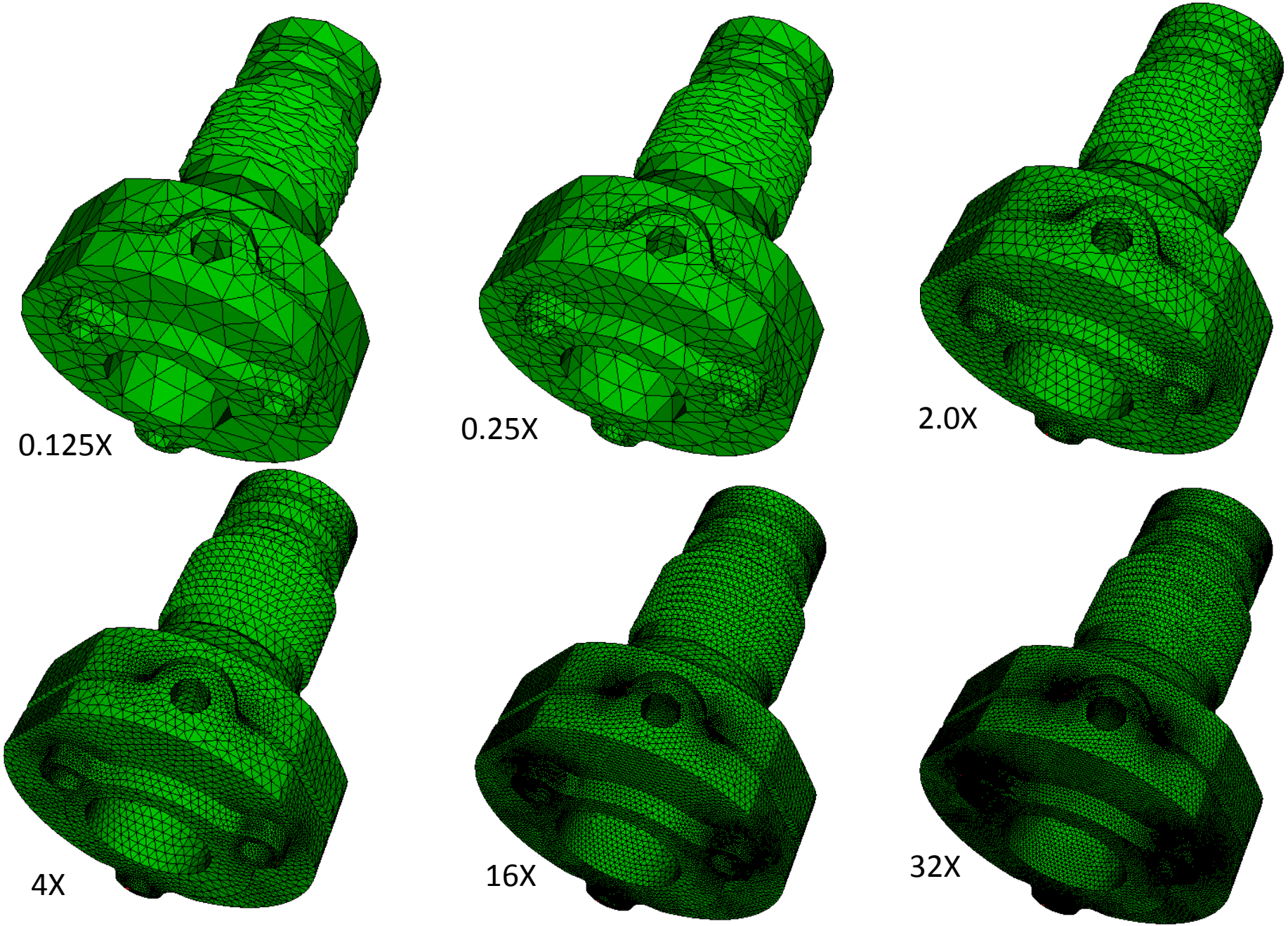


Mesh Based Geometry

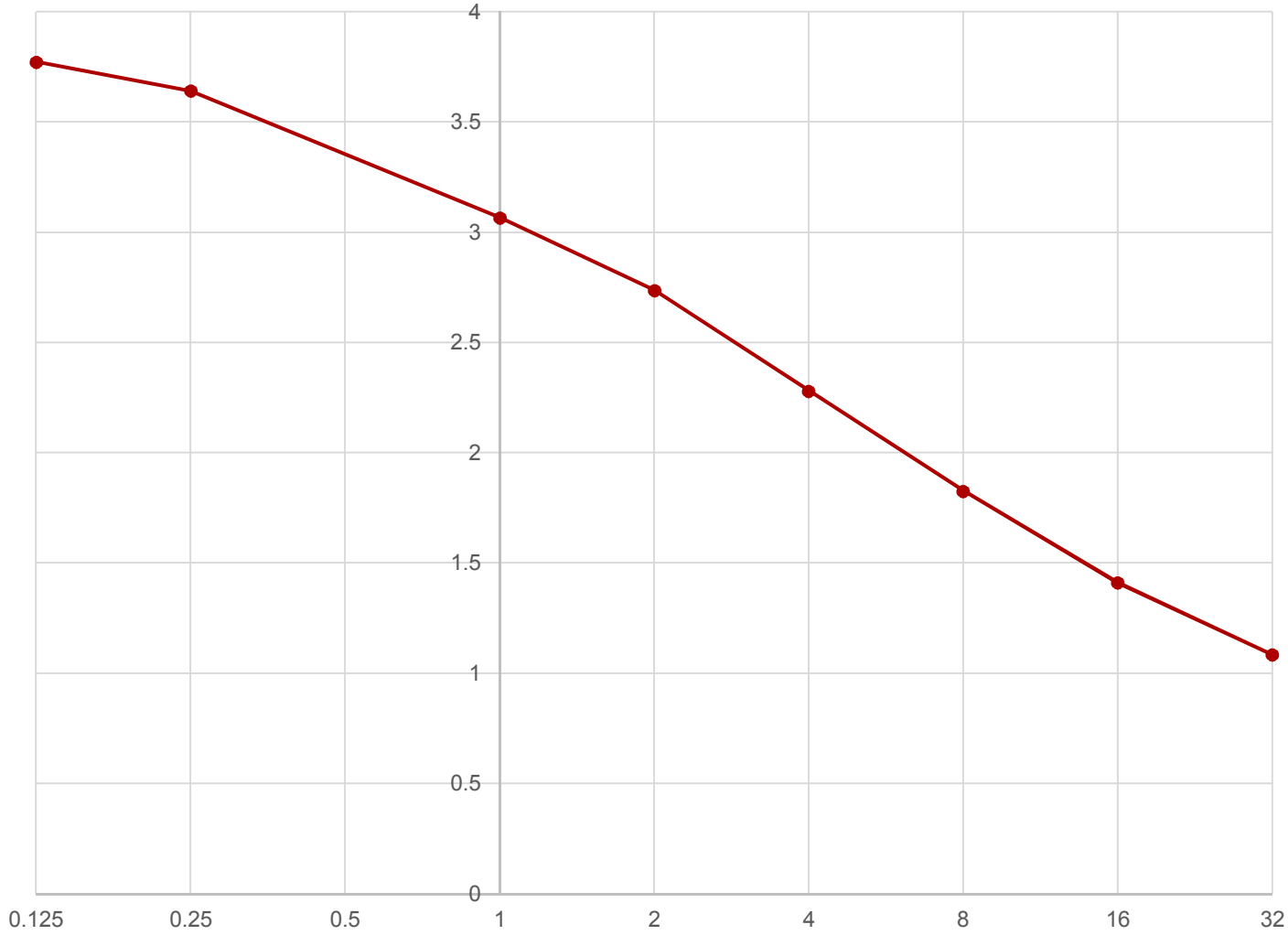


Mesh Sizing Function

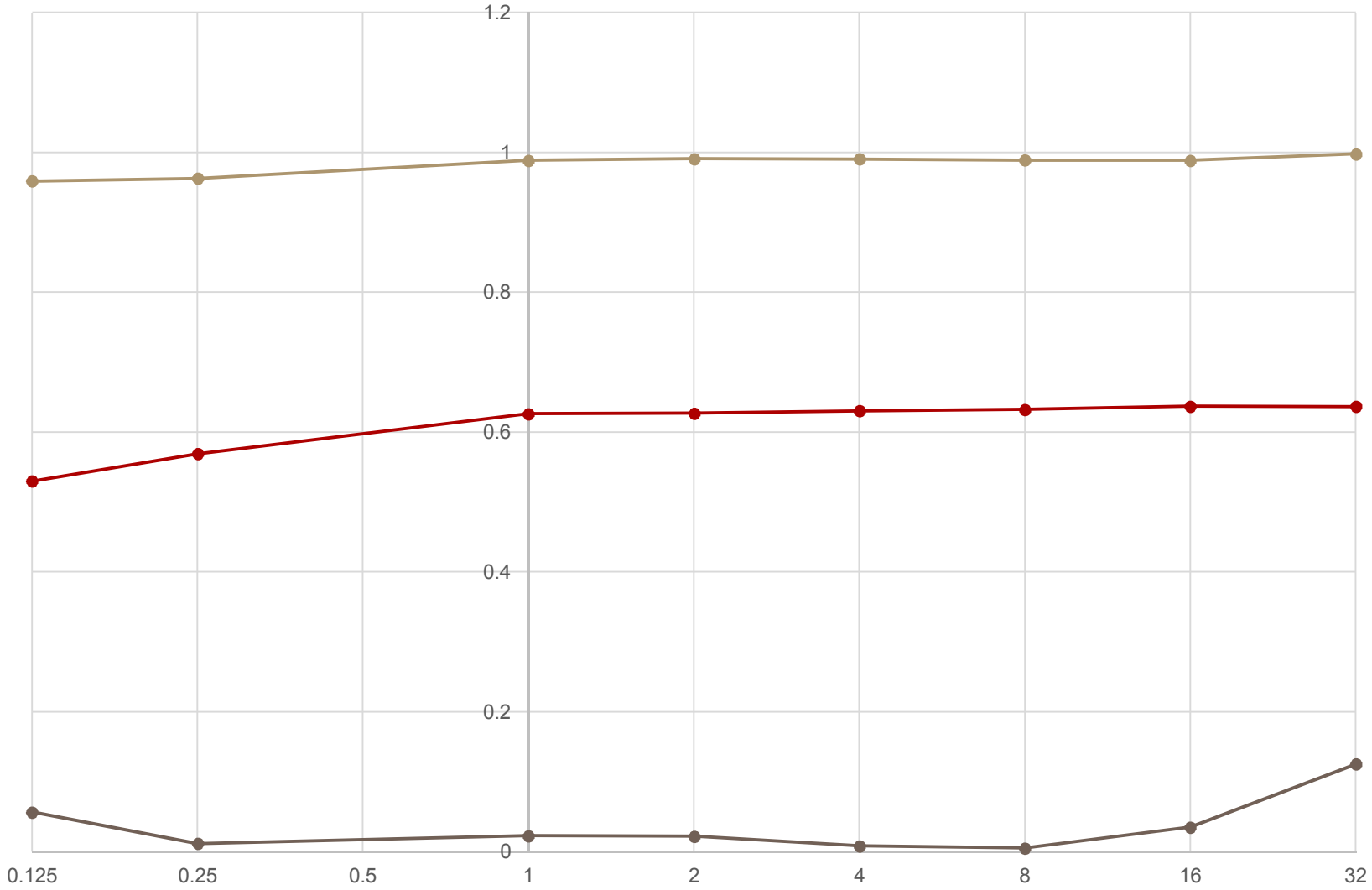
Scaled Tetmesh



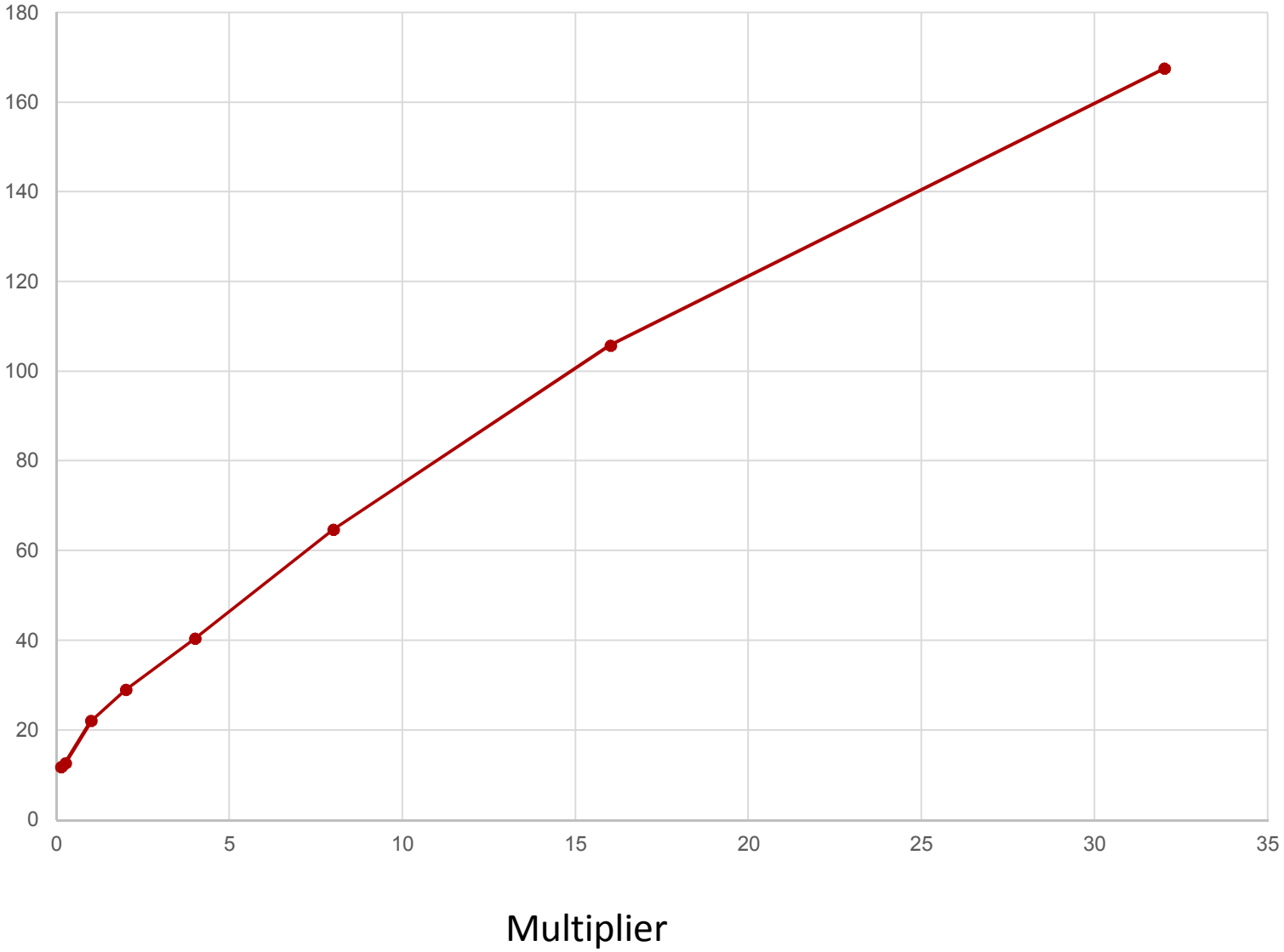
Average Edge Length



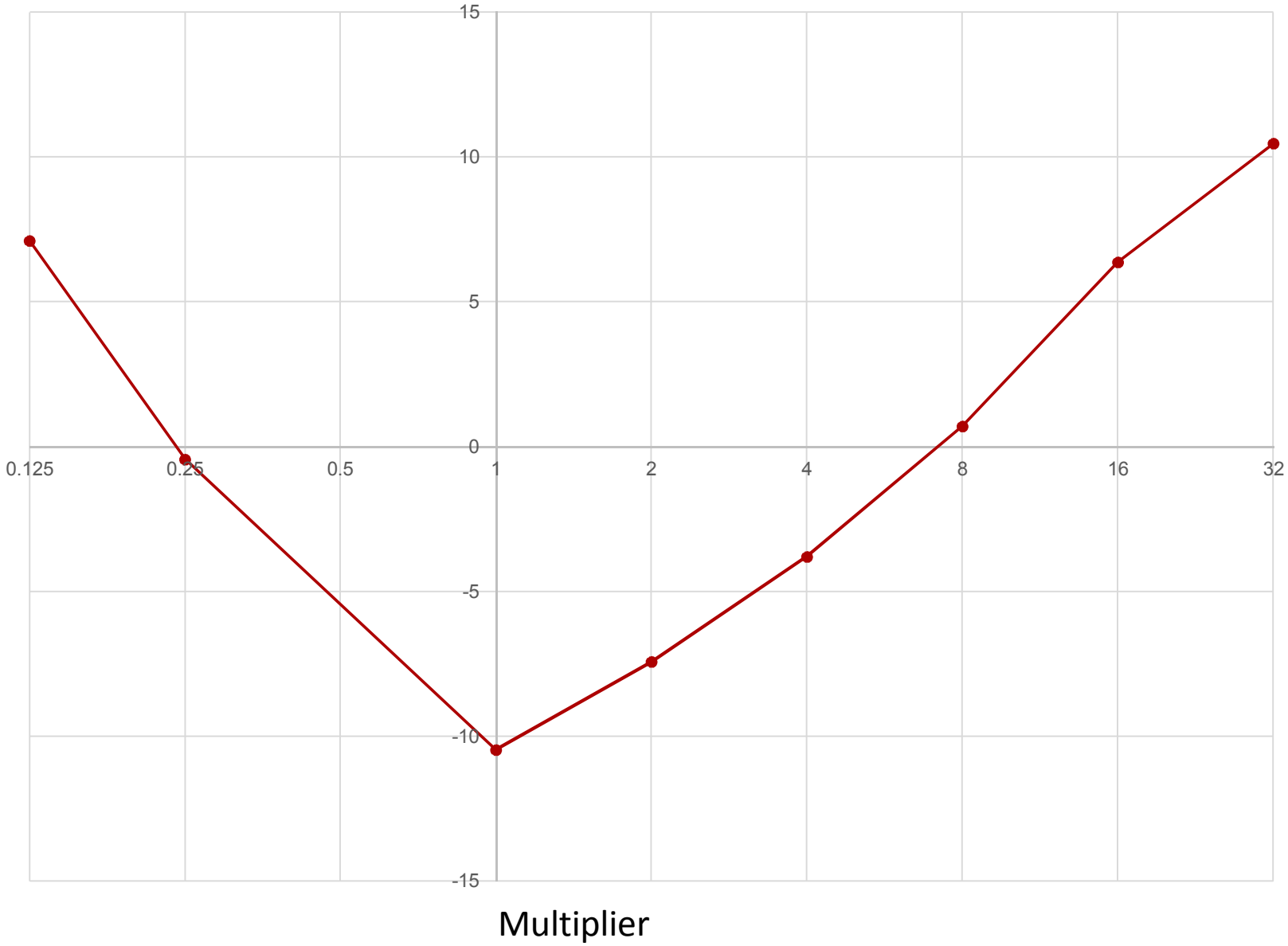
Scaled Jacobian (min, max, avg)



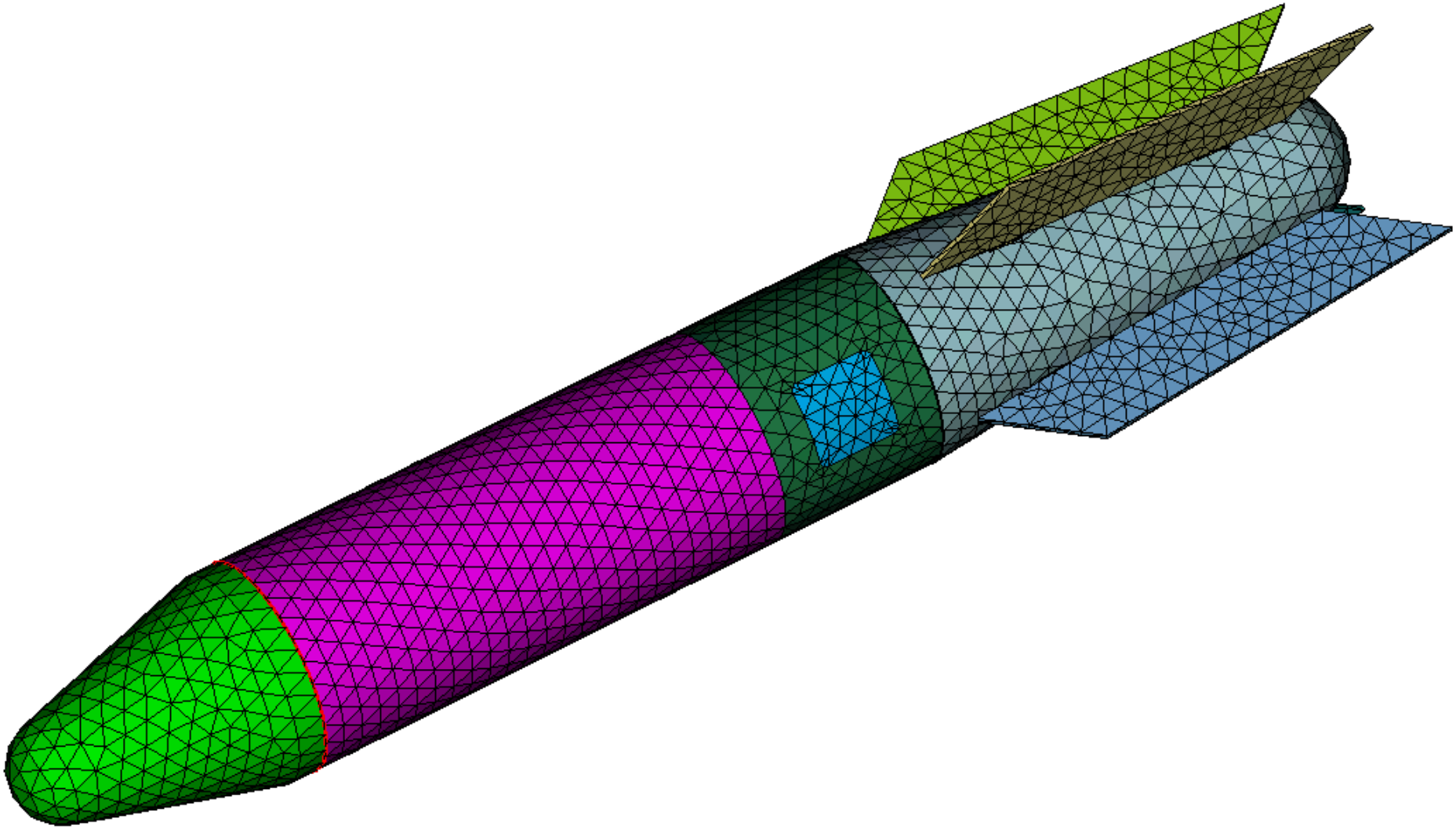
Meshing Time (sec)



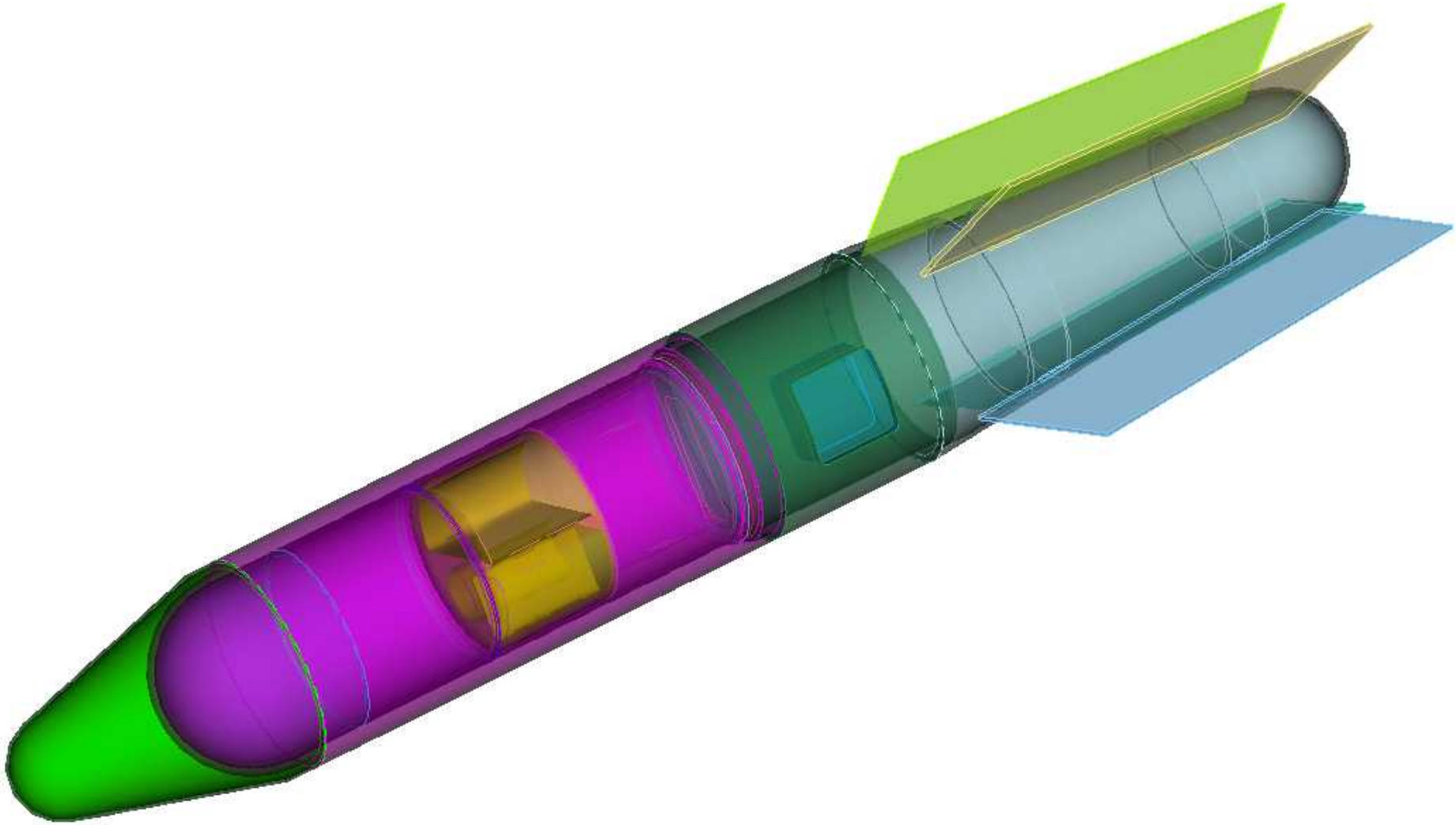
% Deviation from Expected Number of Tets



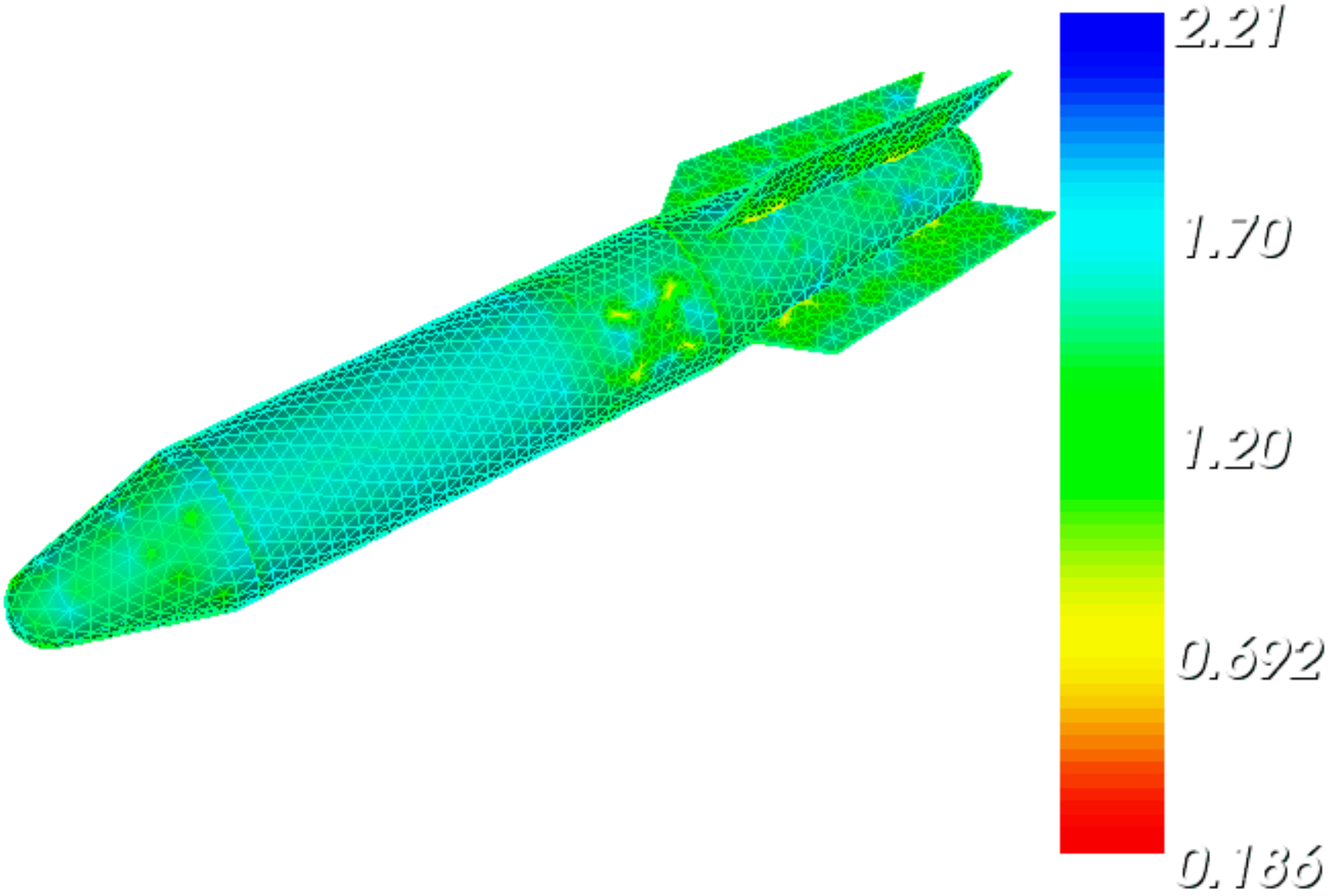
Assembly Model: Input Tetmesh



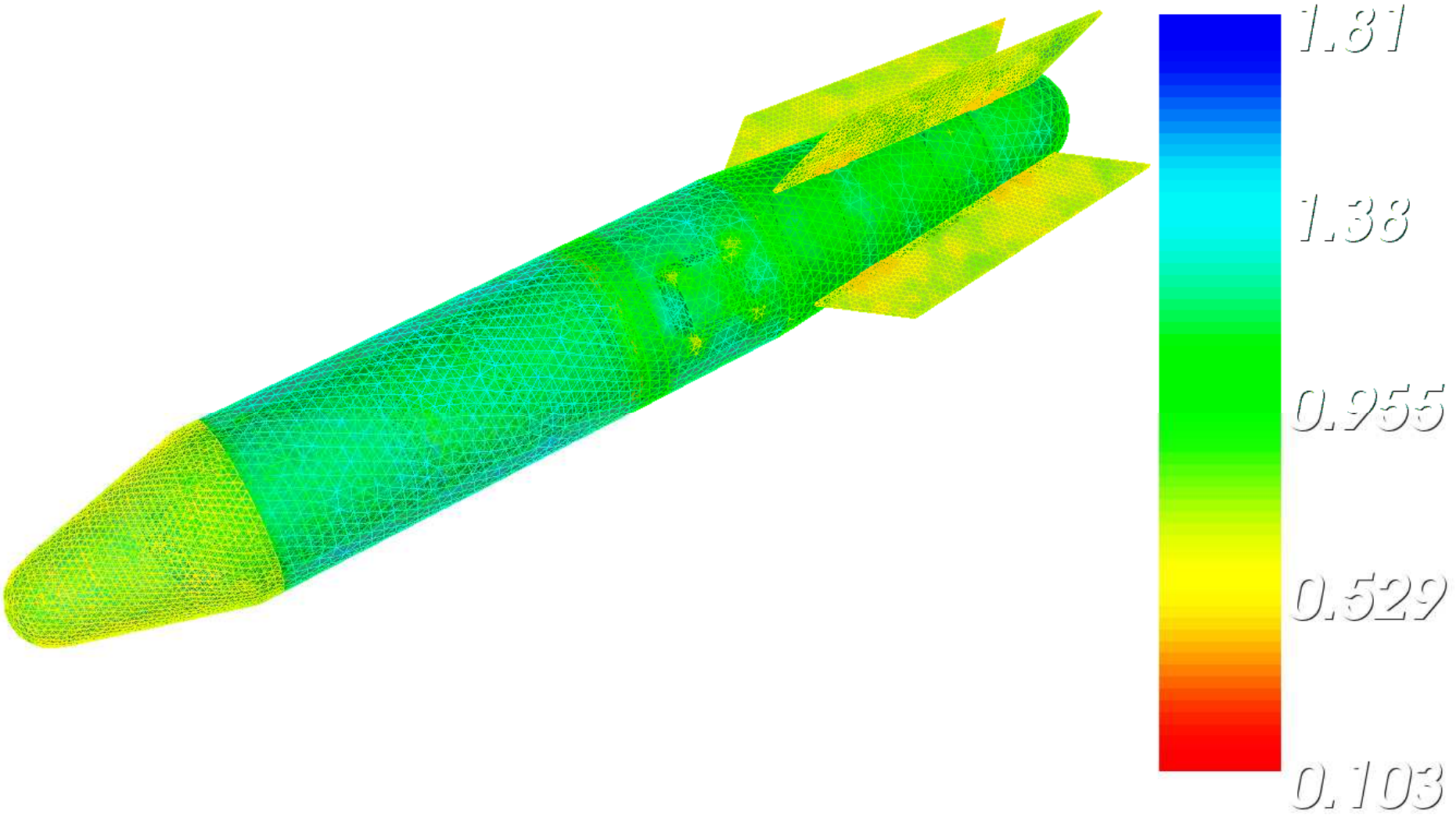
Assembly Model: Mesh Based Geometry



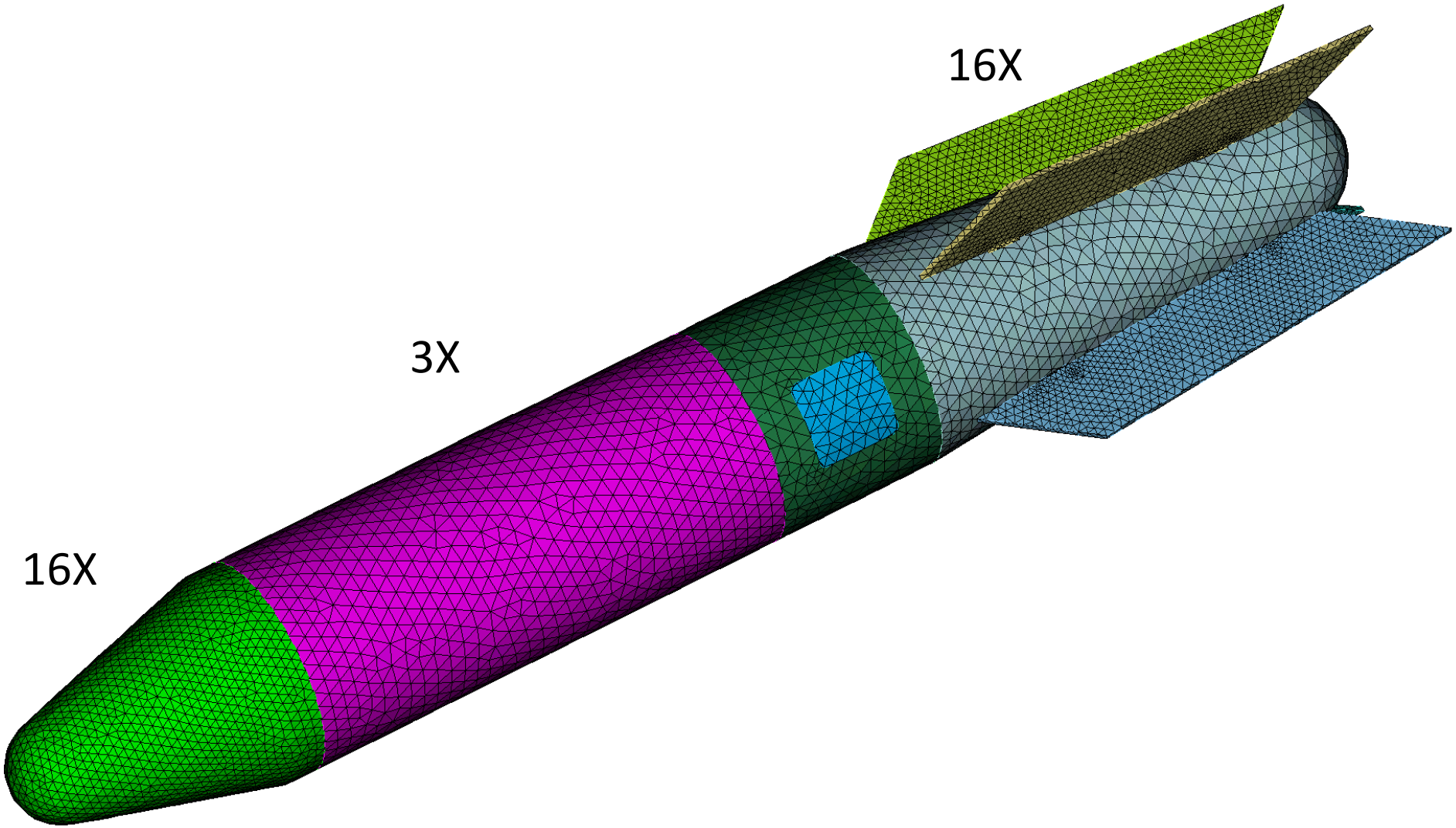
Assembly Model: Sizing Function



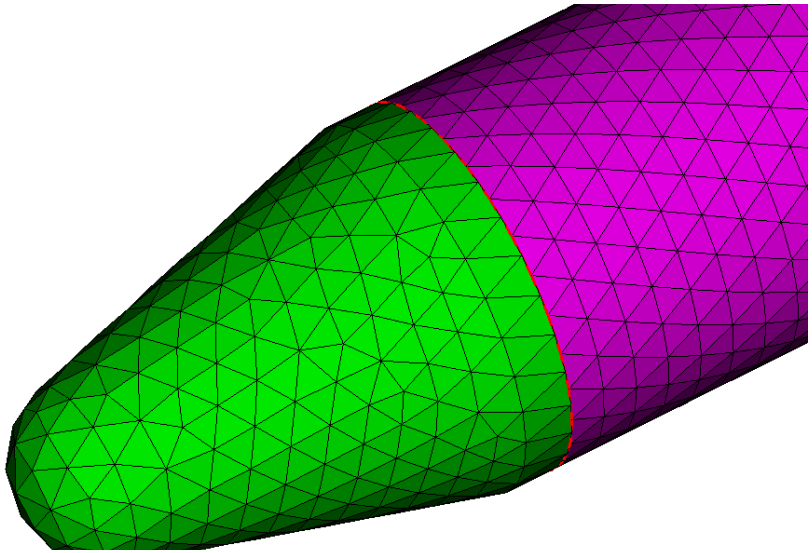
Assembly Model: Scaled Sizing Function

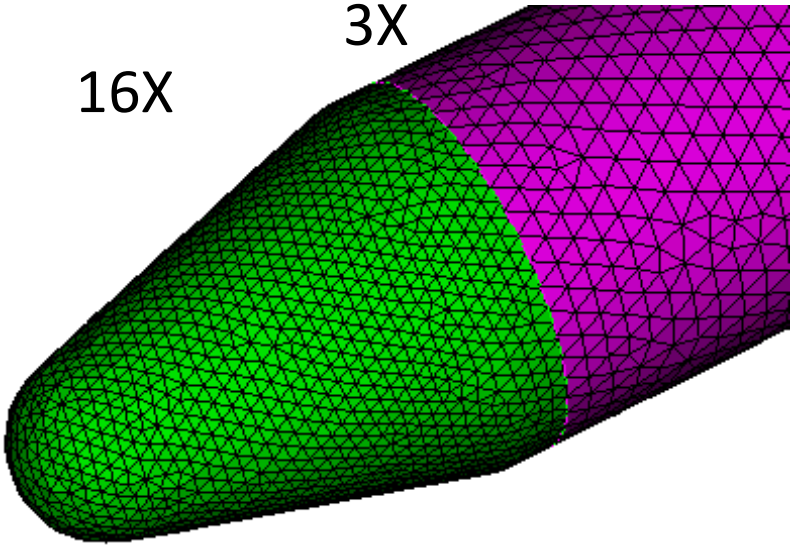


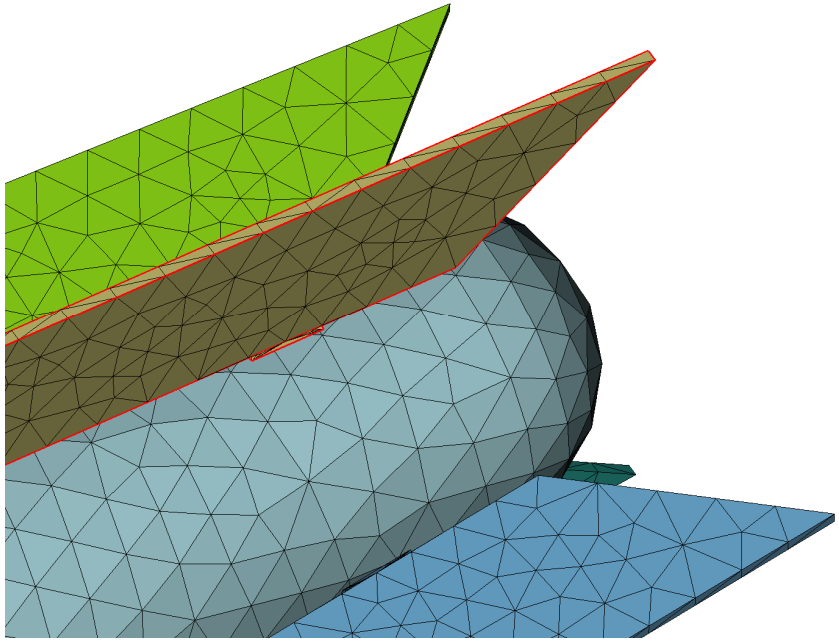
Assembly Model: Output Tetmesh

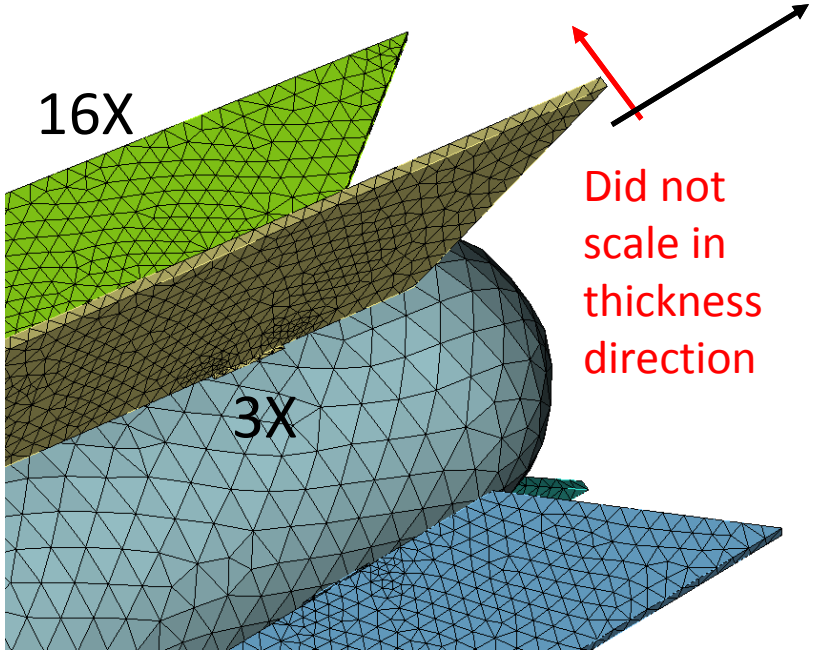


Assembly Model: Zoom Views



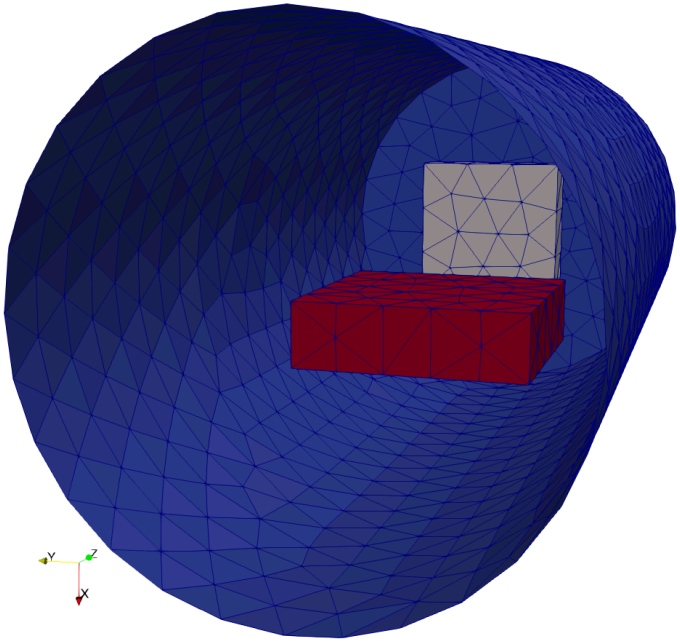




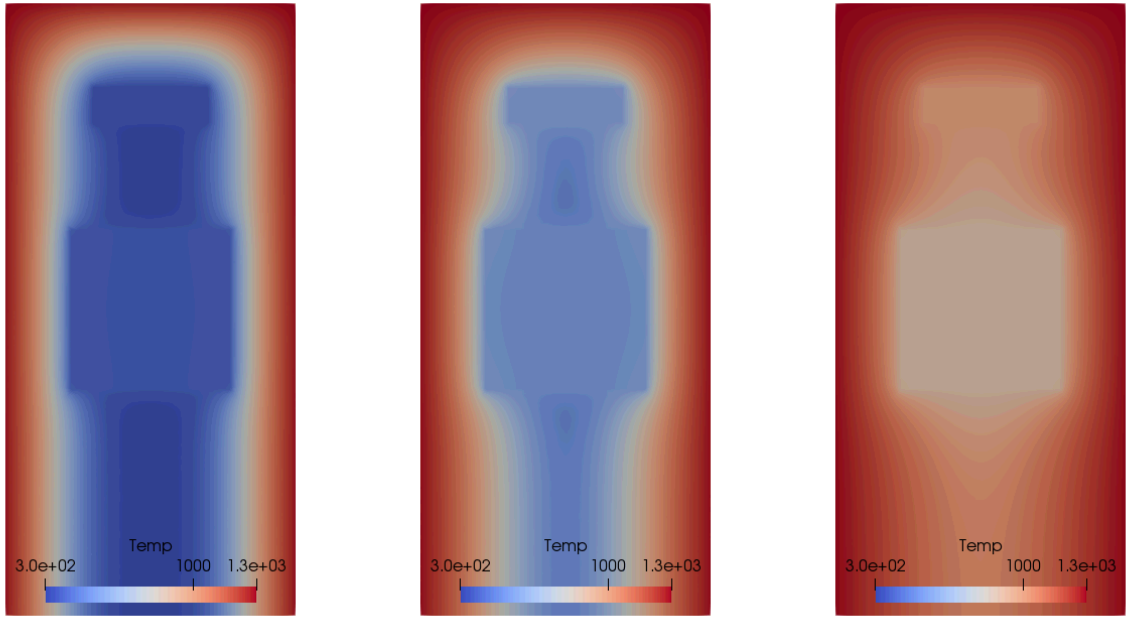


Thermal Analysis Example

- Metal mock components embedded in foam
- Outer metal shell subjected to radiative heat flux
- Nominal mesh has 2K nodes, 3 blocks
- QoIs: Max/min/average/point value of temperature
- Goal: compare mesh scaling to uniform mesh refinement (UMR)



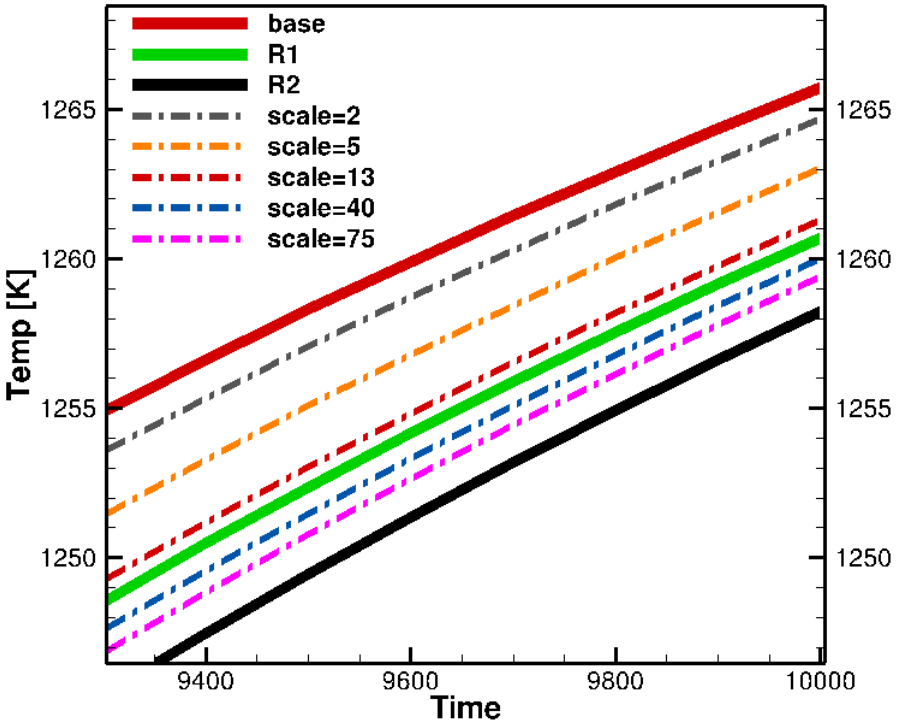
Original mesh showing exterior boundary and two mock components (foam omitted)



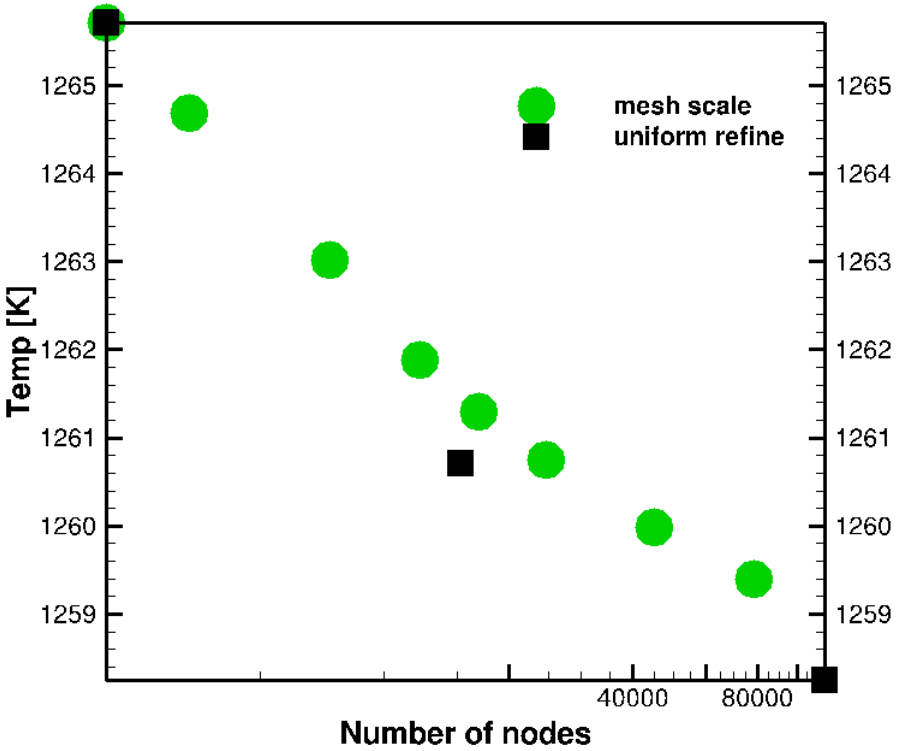
Temperature output on cut plane for increasing time

Mesh Scaling is Useful in Mesh Convergence Studies

- Similar results for all QoIs; here we show max temperature on a block
- The results using mesh scaling should be adequate for doing Richardson extrapolation to estimate numerical error



Time histories of QoIs from scaled meshes follow the same trend as UMR meshes as meshes are refined



We demonstrate the convergence properties as a function of number of nodes

Future Work

- Improve “Effective Multiplier” to match “Specified Multiplier”
  - Constrained gradient limiting at different multipliers
- Handle anisotropic meshes
  - Build initial size field at nodes using local background meshes
- Parallel implementation
  - Size at node calculation is thread safe
  - Domain decomposition
  - Load balancing
- Performance & memory analysis on large customer models

Conclusion

TetMesh scaling provides:

- Incrementally finer meshes without 8X multiplier
- Incrementally coarser meshes can be generated
- Multiple multipliers can be supported via gradient limiting
- More meshes with fewer elements
- More data points on solution convergence plots

Thank You