Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2016) 000–000

SAND2016-5412C

Procedia Engineering

www.elsevier.com/locate/procedia

25th International Meshing Roundtable

Mesh scaling for affordable solution verification

Matt Staten[a,*], Brian Carnes[a,*], Corey Mcbride[b], Clint Stimpson[b], Jim Cox[a]

[a]*Sandia National Labs, Albuquerque, NM, USA*
[b]*Elemental Technologies, Inc., American Fork, Utah, USA*

Abstract

Solution verification is the process of verifying the solution of a finite element analysis by performing a series of analyses on meshes of increasing mesh densities, to determine if the solution is converging. Solution verification has historically been too expensive, relying upon refinement templates resulting in an 8X multiplier in the number of elements. For even simple convergence studies, the 8X and 64X meshes must be solved, quickly exhausting computational resources. In this paper, we introduce Mesh Scaling, a new global mesh refinement technique for building series of all-hexahedral meshes for solution verification, without the 8X multiplier. Mesh Scaling reverse engineers the block decomposition of existing all-hexahedral meshes followed by remeshing the block decomposition using the original mesh as the sizing function multiplied by any positive floating number (e.g. 0.5X, 2X, 4X, 6X, etc.), enabling larger series of meshes to be constructed with fewer elements, making solution verification tractable.

Keywords: Mesh Refinement; Solution Verification; Remeshing; Hexahedra

1. Introduction

In order to quantify the numerical error in finite element models, a sequence of meshes is typically required with increasing resolution (reduced element size). The current state-of-the-art approach is to take a base mesh and produce finer meshes using uniform mesh refinement (UMR), under which every element is subdivided according to a refinement pattern. For example a hexahedral element is typically split into eight child elements. In practice, the cost to run the computational model on even a sequence of three meshes (base mesh plus two levels of UMR: 8X, 64X) is prohibitive requiring at least a factor of 100 times the base computational cost, making solution verification with UMR on industrial models infeasible. In addition, UMR cannot be used to coarsen a mesh.

In this paper, we introduce Mesh Scaling, which globally increases or decreases the number of elements in all-hexahedral meshes by any positive floating point multiplier. By allowing arbitrary scale factors, larger series of meshes with fewer elements can be produced significantly reducing the cost of numerical error estimation. As illustrated in Figure 1, Mesh Scaling traverses an existing mesh identifying and propagating constraints from CAD associativity, material assignments, boundary conditions, and mesh irregularities to construct a block decomposition. Each block in the decomposition is either a structured block (Figure 2a) or a swept block (Figure 2b) which can be meshed with

* Corresponding author
E-mail address: mlstate@sandia.gov, bcarnes@sandia.gov

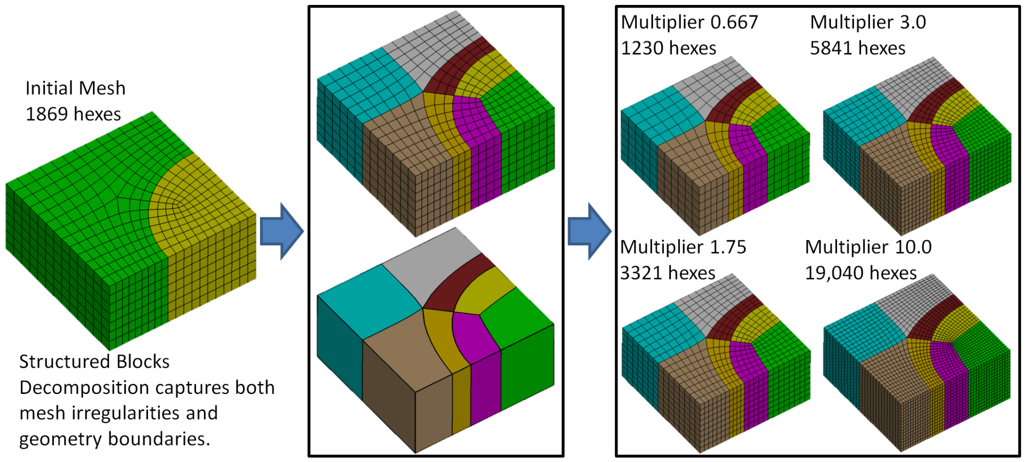

Fig. 1. Overview of Mesh Scaling with a structured block decomposition.

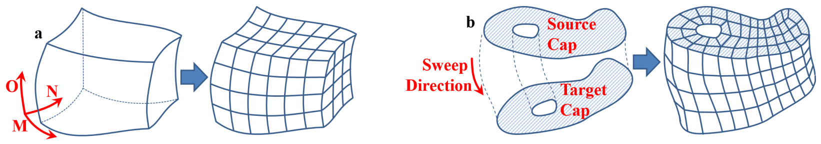

Fig. 2. Block types used in Mesh Scaling. a) a structured block. b) a swept block.

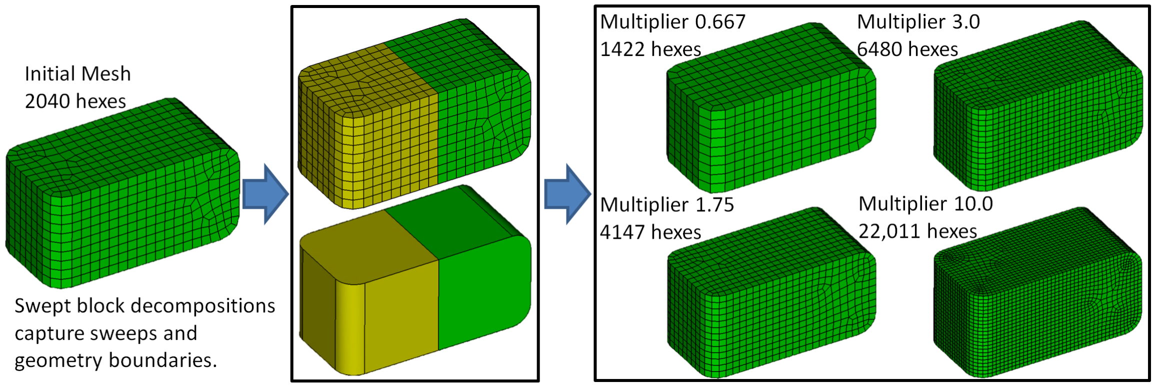

Fig. 3. Overview of Mesh Scaling with a swept block decomposition.

a simple all-hex meshing algorithm. The original mesh is then deleted and the block decomposition is remeshed with a sizing function based on sizes from the original mesh, scaled by an input multiplier.

Figure 1 demonstrates Mesh Scaling with structured blocks only, which provides evenly refined meshes for highly-structured original meshes. However, on other, less-structured, models, structured blocks alone results in a streaky, uneven refinement. Figure 3 demonstrates Mesh Scaling with swept blocks, which provides evenly refined meshes for both structured and unstructured meshes, but changes the number and location of irregular nodes (i.e. nodes with a non-optimal valence [12]).

In some cases, a tool such as Mesh Scaling can be avoided by simply re-applying the original recipe (script) for generating the mesh at different sizes. However, this is not always possible. The recipe may no longer be available. Even, if the recipe is available, running it at a different size is often unsuccessful because of the propagating constraints in hexahedral meshes. Also, re-applying the recipe requires access to the original CAD, and the original CAD decomposition upon which the recipe is built, which may not be available either. In addition, mesh modifications such as smoothing, refinement, pillowing, etc. may have been done to the mesh after the initial generation and these would not be re-applied by running the recipe. Finally, the mesh generation recipe most likely does not include application of the material properties, loads, and boundary conditions, and would have to be re-applied if the recipe were re-run. In contrast, Mesh Scaling operates on any all-hex mesh with or without the original CAD, including any refinements, or local topology modifications, and can automatically re-apply all analysis constraints upon the new mesh.

Other than re-applying the original recipe, we are unaware of any prior work that allows for global refinement or coarsening of hexahedral meshes that does not require the 8X refinement template. Cargemel et. al. [3] introduce a coarsening method which coarsens all-hexahedral meshes by repeated merging of adjacent sheets in the dual of the mesh [7] to reduce simulation computational cost, without explicit computation of a block decomposition, but provides no extension for refinement. Construction of block decompositions has been a standard in commercial and research software for decades for generation of block structured all-hexahedral meshes [14–17]. However, these highly user-interactive tools construct the block decomposition directly from the geometric model and the surrounding void space. In contrast, Mesh Scaling automatically reverse engineers a coarse block decomposition from an existing mesh.

In recent years, research on frame fields [11] has opened the possibilty of automatically constructing block decompositions directly from 3D arbitrary geometry [8, 9]. While research on 2D planar models has proven to be quite robust, significant open research issues remain for robust automatic 3D block decomposition generation from arbitrary geometry. If successful, this original blocking could be used for not just the initial mesh generation, but also for construction of subsequent meshes for solution verification.

2. Mesh Scaling

Mesh Scaling is performed using the following eight steps:

2.1. STEP 1: Identify Swept Mesh Toplogy

The first step in Mesh Scaling is to identify the sweeps in the mesh, as illustrated in Figure 4. A sweep is identified by finding topologically parallel [13] strings of simply-connected interior singular edges as illustrated in Figure 4a.

Definition (Simply Connected Edge Set): Given a set of edges, $E = \{e_1, e_2, ...\}$, we define node set, $N = \{n_1, n_2, ...\}$ such that N contains all nodes on any edge in E. E is simply connected if every node, $n \in N$, is adjacent to either one or two edges in E. If a node, $n \in N$, is connected to three or more edges in E, then E is not simply-connected.

Definition (Interior Singular Edge): An edge interior to the mesh adjacent to something other than four hexahedra.

All singular edges interior to a geometric volume are identified and sorted into sets based on their inter-connection. Any set which is not simply-connected is discarded. The remaining edge sets form *singular edge strings*.

Swept blocks are formed by collecting hexahedra adjacent to each singular edge string, and iteratively expanding and merging with adjacent swept blocks as illustrated in Figures 4b-h. While expanding and merging, checks are required to maintain a swept topology in each swept block. Priority is given first to expansions which will reduce or eliminate topological block concavities and second to expansions which increase the size of smaller swept blocks. For example, the expansions in Figures 4b-d are all in concavities, resulting in six convex swept blocks in Figure 4d. Expansion in Figure 4e results in a merge with an adjacent block which in turn creates new concavities to prioritize. In Figures 4f-h, the priority of expanding smaller blocks ensures that the two swept blocks are roughly the same size.

Swept blocks are extracted as one-to-one sweeps. Circular sweeps are cut into two non-circular sweeps that share two end caps to simplify re-meshing. We see no advantage to extracting many-to-one or many-to-many sweeps [1] since they would need to later be decomposed back to one-to-one sweeps as part of the re-meshing process.

Swept blocks are allowed to expand as long as they maintain swept mesh topology and do not cross any geometric or analysis constraint which must be maintained in the scaled mesh. Unlike the simple model in Figure 4, most production models do not have swept blocks covering 100% of the model. Rather, after swept blocks are identified, structured blocks must be formed in the remaining mesh zones, for a mix of swept and structured blocks in the final block decomposition. Each swept block b is stored in a set of swept blocks B_{swept}, and used to idenfity constraints for the subsequent structured block identification.

2.2. STEP 2: Identify Constraints

Constraints include anything in the original model that must be maintained during Mesh Scaling, including topology from the associated CAD model, boundaries of swept blocks in B_{swept} identified in Step 1, and analysis constraints

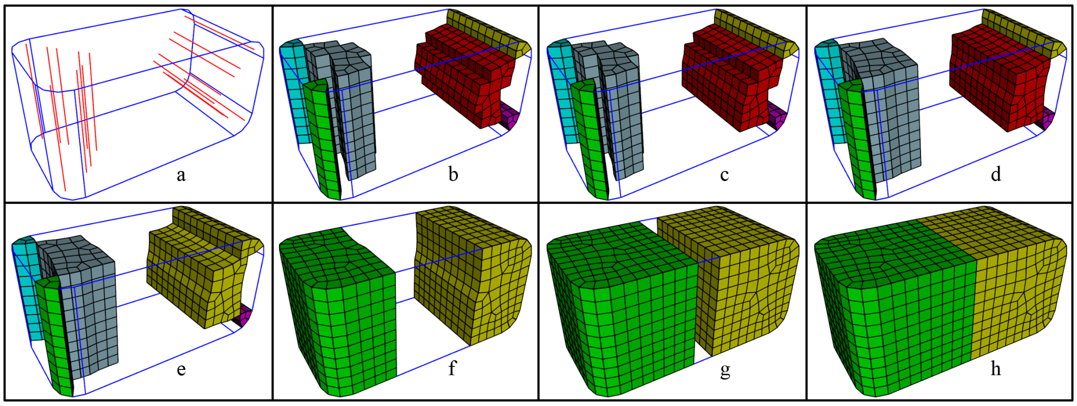

Fig. 4. Finding sweeps in the mesh from Figure 3. a) Find topologically parallel strings of simply-connected singular edges. b) Form six swept blocks with adjacent hexahedra. c) Expand to fill a concavity on the red block. d) Further expansion to remove all concavities. e) Expand and merge. f-h) Further expansion.

such as boundary conditions and material boundaries. We define a new model topology used for mesh scaling, T_{ms}, which is a set of points, P_{ms}, curves, C_{ms}, surfaces, S_{ms}, and volumes, V_{ms}. We imprint into T_{ms} the topology of all constraints. For example, every CAD vertex is imprinted as a point, $p \in P_{ms}$, every CAD curve as a curve, $c \in C_{ms}$, etc. A point load location is imprinted as a point, $p \in P_{ms}$. Similarly, if a CAD volume contains multiple swept blocks, each swept block will be represented as separate volume, $v \in V_{ms}$. After construction, T_{ms} contains the CAD topology, if available, plus additional topology required to capture the analysis constraints and swept blocking.

We define the following subsets to simplify the propagation of the constraints:

- V_{swept}, which contains any volume, $v \in V_{ms}$, which maps to a swept block from B_{swept}.
- S_{swept}, which contains any surface, $s \in S_{ms}$, which is adjacent to any volume, $v \in V_{swept}$.
- $V_{struct} = V_{ms} - V_{swept}$
- $S_{struct} = S_{ms} - S_{swept}$
- E_{ribs}, which contains any edge defining a rib [10] of a swept block, $b \in B_{swept}$. Ribs are the simply-connected sets of edge strings on the boundary of a sweep which run parallel to the sweep direction [10].

We translate these topology contraints into mesh entity constraints in the form of a node set N_C, an edge set E_C, and a quad set Q_C containing nodes, edges, and quads from the original mesh as follows.

- Q_C contains all quads on all surfaces in S_{ms}.
- E_C contains all edges:
  - on any curve in C_{ms}, or
  - on any surface in S_{ms}, adjacent to n hexes in any one structured volume, $v \in V_{struct}$, $n \in integers \neq 2$, or
  - which are interior singular edges in a structured volume, $v \in V_{struct}$.
- N_C contains all nodes:
  - on any point in P_{ms}, or
  - on any curve in C_{ms}, adjacent to n quads in any one structured surface, $s \in S_{struct}$, $n \in integers \neq 2$, or
  - on any structured surface, $s \in S_{struct}$, which has n adjacent quads in s, $n \in integers \neq 4$, or
  - in any structured volume in V_{struct}, which has n adjacent edges $\in E_C$, $n \in integers \neq 2$.

2.3. STEP 3: Propagating Block Constraints For Structured Block Decompositions

After all mesh constraints are identified and stored in sets N_C, E_C, and Q_C in STEP 2, they are propagated through the mesh to form the boundaries of the blocks to be created in the decomposition using Algorithm 1.

On line 6 of Algorithm 1, edge e is guaranteed to have exactly four adjacent hexahedra, otherwise, it would have been placed in E_C in STEP 2. Figure 5 illustrates how to find quad q_{new}

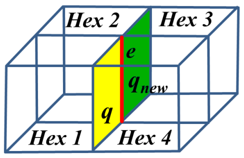

Fig. 5. Given quad q, in yellow, and adjacent edge e, in red, we seek q_{new}, in green. By finding the four hexahedra (i.e. Hex 1, 2, 3, and 4) adjacent to e, Hex 2 and Hex 3 are identified as adjacent to e, but not adjacent to q. q_{new} is the quad common to Hex 2 and Hex 3.

Algorithm 1 Propagate Block Constraints For Structured Block Decompositions

```
1: Add to $E_C$ all edges adjacent to any $n \in N_C$
2: Add to $Q_C$ all quads adjacent to any $e \in E_C$, except quads adjacent to 2 hexes in a single swept block, $b \in B_{swept}$
3: for all quad $q \in Q_C$ do
4:     for all edge $e$ adjacent to $q$ do
5:         if $e \notin E_C$ AND $e \notin E_{ribs}$ then
6:             Find the quad $q_{new}$, adjacent to $e$ and has no common adjacent hex with $q$ as illustrated in Figure 5
7:             if $q_{new} \notin Q_C$ then Insert $q_{new} \rightarrow Q_C$
```

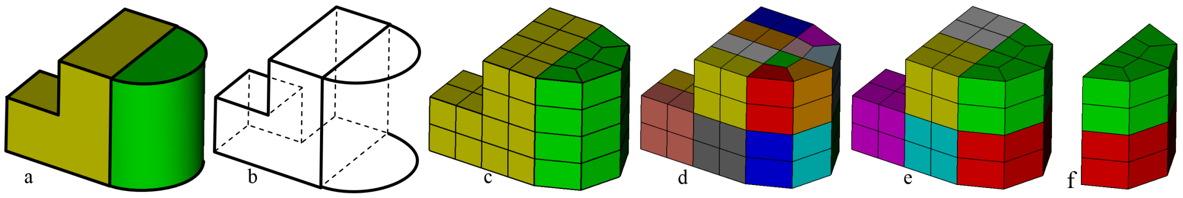

Fig. 6. Example block decompositions. a,b) Simple CAD model with two material zones (volumes). c) Mesh on model. d) Structured Block Decomposition, contains 22 structured blocks. e) Swept Block Decomposition, contains seven blocks. f) two swept blocks = one sweep group.

2.4. STEP 4: Construct Block Decomposition

After constraint propagation, Q_C contains all the quads on the boundary of all blocks in the decomposition. Blocks are constructed by grouping hexahedra such that no group crosses any quads in Q_C as defined in Algorithm 2. The resulting blocks will form either an MxNxO structured block (Figure 2a), or a one-to-one swept block (Figure 2b). The blocks in the decomposition are constructed with a full vertex, curve, and surface topology. Do not confuse this block decomposition topology with the Mesh Scaling Topology, T_{ms}, the block decomposition topology defined in Section 2.2. T_{ms} contains only the non-mesh topology constraints, and the volumes in T_{ms} may not be one of the block types defined in Figure 2. The block decomposition topology contains all constraints from T_{ms} and all meshing contraints, propagated to cut the model into either structured or swept blocks.

Figure 6 illustrates a simple CAD model with two volumes. Figure 6d illustrates the resulting block decomposition with the 22 blocks required if only structured blocks are allowed, with the hexahedra color-coded based on which block they are assigned to. Figure 6e illustrates the resulting block decomposition with the seven blocks required if swept blocks are also allowed.

In addition, we subclassify each surface in the block decomposition as either a structured surface (i.e. adjacent to a structured block or a linking surface of a swept block) or as an unstructured surface (i.e. an end cap on a swept block). The swept blocks are grouped into sweep groups which have interdependent end caps. For example, the two swept blocks in Figure 6e are grouped together into a single swept block group illustrated in Figure 6f.

2.5. STEP 5: Extract Sizing Informtaion

Sizing information is extracted from the original mesh for each block decomposition entity in order to pass non-uniform mesh sizes onto the scaled meshes. Sizing for curves, structured surfaces, and structured blocks is accomplished by defining piecewise linear parameterizations based on the nodes in the original mesh. Figure 7 illustrates a block decomposition curve meshed with five biased mesh edges in the original mesh. The six black dots represent the six nodes on the curve. Each node is assigned a t parameter based on its topological location along the curve. During

Algorithm 2 Group Block Hexahedra

```
 1: Construct a set $H$, containing all hexes in the mesh
 2: Construct an empty set, $H_{visited}$
 3: for all hex $h \in H$ do
 4:     if $h \notin H_{visited}$ then
 5:         Construct an empty set, $H_{block}$
 6:         Insert $h \rightarrow H_{block}$
 7:         for all hex $h_{block} \in H_{block}$ do
 8:             for all quad $q$ adjacent to $h_{block}$ do
 9:                 if $q \notin Q_C$ then
10:                     Find the hex, $h_{other}$, adjacent to $q$ such that $h_{other} \neq h_{block}$
11:                     if $h_{other} \neq \emptyset$ and $h_{other} \notin H_{block}$ then
12:                         Insert $h_{other} \rightarrow H_{block}$
13:                         Insert $h_{other} \rightarrow H_{visited}$
14:         Construct a block in the decomposition containing all hex in $H_{block}$
```

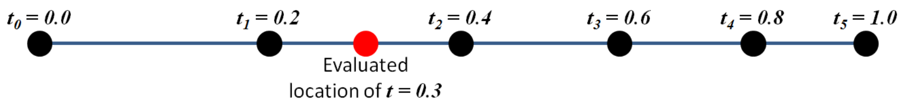

Fig. 7. Curve Parameterization for a block decomposition curve originally meshed with 5 mesh edges. Each original node (black dots) is assigned a t-parameter based on the node's topological order divided by the number of curve intervals. The red dot represented the evaluated location of t=0.3 during a subsequent remesh.

remeshing (STEP 8), the parameter, t_{new} of a new node is determined, and the parameters, t_a and t_b of the two nodes between which t_{new} lies are identified. The location, $\vec{x}(t_{new})$, of the new node is computed using Equation 1.

$$\vec{x}(t_{new}) = \left(\frac{t_b - t}{t_b - t_a}\right)\vec{x}(t_a) + \left(\frac{t - t_a}{t_b - t_a}\right)\vec{x}(t_b) \tag{1}$$

This location, $\vec{x}(t_{new})$, is then projected to the underlying CAD model. 2D and 3D versions of this curve parameterization are used for structured surfaces and volumes in the block decomposition. This piecewise linear interpolation seems to adequately approximate the biases in the original mesh for the solid mechanics models tested.

Sizing functions for unstructured surfaces are constructed by computing the average edge length adjacent to every node in the original mesh in each unstructured surface, and then storing these sizes at the nodes for subsequent interpolation upon sizing queries. We also store a triangle mesh, which is the original quad mesh split into triangles, for use during the interpolation.

2.6. STEP 6: Delete the Original Mesh

After sizing information is extracted, the original mesh can be deleted to free up memory to store the scaled mesh.

2.7. STEP 7: Compute New Curve Intervals

In order to compute the new scaled number of edges (i.e. curve intervals) on each curve in the block decomposition, curve groups must be formed. A curve group is a group of inter-dependent block decomposition curves coupled such that all curves in the group require the same number of intervals. The model in Figure 1 has seven curve groups, three of which are illustrated in Figure 8. Curve groups are formed by propagating from a starting block decomposition curve to opposite curves through adjacent structured block decomposition surfaces. Curve group creation stops once all curves are in a curve group. Each curve will be in exactly one curve group.

The computation of the new number of intervals on each curve group is a function of the number of intervals on the curve group in the original mesh, an input *Multiplier*, an input *MinInt* parameter, and *sum even* constraints on all adjacent unstructured surfaces. The input *Multiplier* defines the target increase in the number of hexahedral elements.

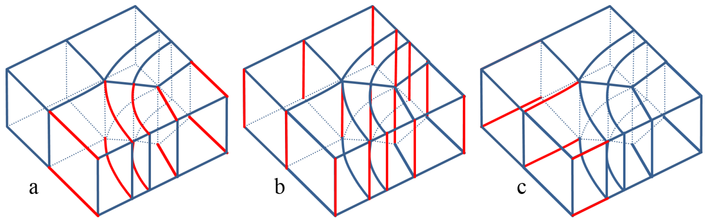

Fig. 8. a) A curve group with 10 curves in red. b) Another curve group with 15 curves. c) Another curve group with six curves.

For example, a mesh with 1000 hexahedra scaled with a *Multiplier* of 2X should result in approximately 2000 hexahedra. It follows that the number of intervals in each curve group should increase by approximately $\sqrt[3]{Multiplier}$. It is approximate because an integer number of intervals must be maintained on each curve.

The *MinInt* parameter defines the minimum number of intervals that must be added to each curve group, and can be used to ensure that every part of the model will have a scaled mesh in all three dimensions locally, avoiding plateaus on subsequent convergence plots. Consider the case of scaling a model with a curve group, g, originally with one interval (i.e. $Intervals^g_{orig} = 1$) with a $Multiplier = 2X$. The simple new interval calculation on this curve group would be $Intervals^g_{orig} * \sqrt[3]{Multiplier} = (1) * \sqrt[3]{2.0} = 1.26$, which will get necessarily rounded to the nearest integer of 1, thus having no change on this curve group. If intervals do not change in physics critical regions of the model, a plateau on the convergence plot could result. Specifying $MinInt = 1$, and using Equation 2 would cause the intervals to change by the minimal amount.

For solution verification of global quantities, each mesh in the series should be finer everywhere from the previous mesh in the series. To achieve this, a $(Multiplier, MinInt)$ set such as (2X,1), (4X,2), (8X,3) could be applied to the original mesh.

We use interval assignment [5, 6] to compute the new intervals. We create a single variable representing the intervals on each curve group, rather than a separate variable for each curve, with an allowed minimum and maximum for each variable as defined by Equations 2 and 3, where S is an integer slack variable ≥ 0. When the block decomposition is remeshed in STEP 8, an unstructured quad surface meshing algorithm, such as paving [2], will be called for one unstructured cap surface from each sweep group. Thus, one *sum even* constraint (Equation 4) is also defined for each swept block group defined in Section 2.4. In a swept block group, only one of the cap surfaces will be sent to paving, with the other caps recieving a transformed copy of the result to conform to sweeping requirements [10]. Thus only one *sum even* constraint equation per sweep group is needed. An interval assignment solve is first attempted with $S = 0$. If the problem is over-constrained, we increment S by one and solve again, repeating until the problem becomes unconstrained.

$$Intervals^g_{min} = MAXIMUM\left(\left\lfloor Intervals^g_{orig} * \sqrt[3]{Multiplier} \right\rceil, MinInt\right) \tag{2}$$

$$Intervals^g_{max} = Intervals^g_{min} + S \tag{3}$$

$$\sum_{i=1}^{num\_curves\_on\_surface} Intervals^g_i = 2k \mid k \in integers \geq 2 \tag{4}$$

With this formulation, interval assignment gets close, but does not guarantee the input *Multiplier* will be met exactly. The exact number of elements in a scaled mesh cannot be calculated precisely until one cap surface of each sweep group has been remeshed as described in Section 2.8, which is done after the intervals are assigned.

2.8. STEP 8: Remesh the block decomposition

Remeshing the block decomposition is performed by first remeshing all of the vertices, curves, structured surfaces, and structured blocks by evaluating the sizing function parameterizations defined in Section 2.5 as follows:

- Nodes on Vertices: Place the node at the original location of the vertex in the block decomposition.

- Nodes on Curves: Evaluate the curve using Equation 1 at t parameters $1/n$, $2/n$, ... $(n-1)/n$, where n is the new number of intervals assigned to the corresponding curve group in STEP 7.
- Nodes on Structured Surfaces: Evaluate the surface using a 2D surface equivelant of Equation 1 at parameters pairs $(1/m, 1/n)$, $(2/m, 1/n)$, $(1/m, 2/n)$, $(2/m, 2/n)$, ... $((m-1)/m, (n-1)/n)$, where m and n are the new number of intervals assigned to the two corresponding curve groups in STEP 7.
- Nodes in Structured Blocks: Evaluate the block using a 3D volumetric equivelant of Equation 1 at parameter tuples in a 3D pattern similar to that done for curves and structured surfaces.

Next, one cap surface for each sweep group is remeshed with an unstructured quad mesh using the sizing function defined for the surface. The resulting unstructured mesh is then copied to the other end caps in the sweep group. Each swept block is then swept as a one-to-one sweep [4, 10].

2.9. Examples: Structured Blocks vs. Swept Blocks

Figures 9-12 illustrate the advantages and disadvantages of using swept blocks vs. structured blocks only on a simple cylinder model. Figures 9 and 10 both scale CylStruct, a structured hexahedral mesh. Figure 9 scales CylStruct with structured blocks only and Figure 10 scales CylStruct with a single swept block. Because the original mesh has so much structure, CylStruct scales very nicely with structured blocks as illustrated in Figure 9. Further, Table 1 shows that scaling CylStruct with structured blocks closely achieves the desired *Multiplier*, maintains the minimum scaled Jacobian exactly, and only slightly changes the maximum aspect ratio. Further, using structured blocks results in scaled meshes that closely resemble the original mesh visually, because the block decomposition (Figure 9b) maintains all of the irregular nodes. The additional cut in the block decomposition (Figure 9b) is due a CAD vertex on the outer circular curve of the cylinder end surface.

In contrast, scaling the CylStruct model with swept blocks results in a single swept block in the decomposition (Figure 10b). The cap surfaces of this swept block are remeshed with the paver [2] changing the number and location of irregular nodes. Table 1 shows that scaling the CylStruct model with swept blocks sometimes under-achieves the desired *Multiplier* due to the unpredictability of paving [2], with only limited increase in both minimum scaled Jacobian and the maximum aspect ratio.

Figures 11 and 12 both scale CylUnstruct, a unstructured hexahedral mesh. Figure 11 scales CylUnstruct with structured blocks only and Figure 12 scales CylUnstruct with swept blocks. Because the original mesh has so many irregular nodes, when scaling with structured blocks, the block decomposition (Figure 11b) has a lot more blocks when compared to scaling the more structured CylStruct model. When scaling CylUnstruct at *Multiplier*=2X and *MinInt*=0 the blocks indicated with red dots in Figure 11c remain unchanged in two dimensions. In addition, the resulting scaled meshes (Figures 11c-g) are streaky and uneven. Table 2 further shows that scaling CylUnstruct with structured blocks often significantly overshoots the desired *Multiplier* (by more than double in some cases) and raises the maximum aspect ratio, but maintains the minimum scaled Jacobian exactly.

In contrast, scaling the CylUnstruct model with swept blocks results in a single swept block in the decomposition (Figure 12b). The cap surface of this swept block is remeshed with the paver. Since the cap surface was originally meshed with an unstructured quadrilateral mesh, remeshing with the paver, while changing the location and number of irregular nodes, results in a mesh very similar to the cap mesh in the original mesh. Table 2 shows that scaling CylUnstruct with swept blocks maintains the minimum scaled Jacobian closely (although not exactly), closely matches the desired *Multiplier*, with only moderate increase in the maximum aspect ratio.

We conclude that the ideal type of blocks to use during Mesh Scaling is dependent upon the original mesh. If the original mesh was carefully constructed with lots of structure, then Mesh Scaling should probably use structured blocks only. If the original mesh has unstructured paved surfaces, then swept blocks should probably be used. In practice, users are allowed to specify what types of blocks to use both globally and locally. If the model has a lot of structure in some regions, and paved meshes in another, users can specify that swept blocks be used everywhere except in user-specified highly structured regions.

Coarsening can be achieved by specifying a *Multiplier* < 1.0. However, Mesh Scaling can not coarsen beyond the topology of the block decomposition. Since using structured blocks only results in more blocks in the decomposition than using swept blocks, using swept blocks will allow for more coarsening than structured blocks only.

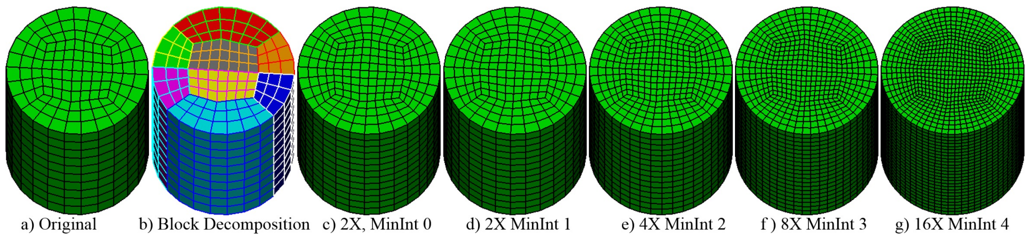

Fig. 9. CylStruct model scaled with structured blocks only.

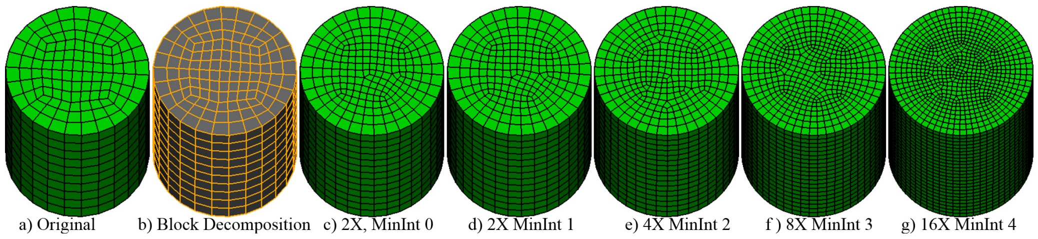

Fig. 10. CylStruct model scaled with swept blocks.

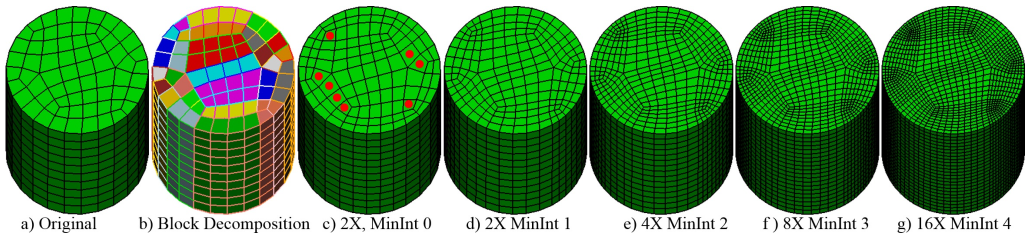

Fig. 11. CylUnstruct model scaled with structured blocks only.

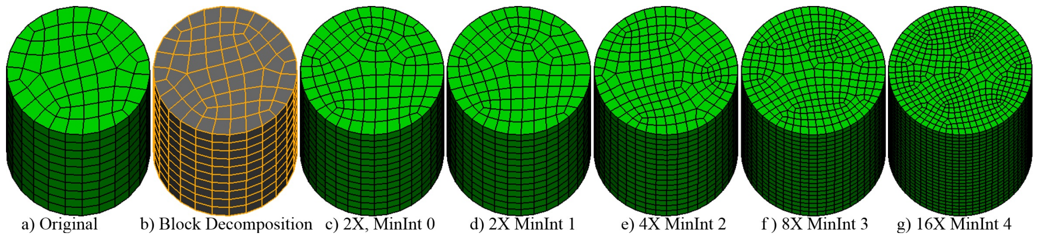

Fig. 12. CylUnstruct model scaled with swept blocks.

3. Solution Verification Using Scaled Meshes

In this section we provide results for solution verification using Mesh Scaling for two example problems. Both problems use a nonlinear large deformation solid mechanics code to produce the results using a quasistatic solver. The material models used are incremental elastic-plastic models.

The Notch model (Figure 13) is for a tension test using a notched cylinder. The original mesh has 22,860 elements. Using scale factors that were multiples of 2.0 and 0.5 we generated over eight additional meshes using swept blocks. Examples of the meshes produced are shown in Figure 14 with data about the scaled meshes in Table 3.

We used an error model with three parameters. If h is the mesh size, $Q(h)$ is the approximate quantity of interest (QoI) for mesh size h, and $E(h)$ is the error, then the error estimate takes the form in Equation 5.

$$E(h) = Q - Q(h) = C\, h^p \tag{5}$$

Traditional extrapolation requires three meshes to estimate the three parameters in this error model. We considered a nonlinear least squares fitting of the error model using three or more meshes. In Figure 15 we plot the estimated errors in one QoI using a fitting based on five meshes centered around the base mesh (two coarsened meshes and two refined meshes). Here the QoIs are the maximum element value of a material parameter called equivalent plastic

Table 1. CylStruct model scaled data.

	Structured Blocks Only (Figure 9)				Swept Blocks (Figure 10)			
	#elems	Min Scaled Jacobian	Max Aspect Ratio	Effective Multiplier	#elems	Min Scaled Jacobian	Max Aspect Ratio	Effective Multiplier
original Mesh	1,200	0.7931	2.019	-	1,200	0.7931	2.019	-
scaled (2X, *MinInt* 0)	2,496	0.7931	2.110	2.08	2,756	0.8070	2.798	2.30
scaled (2X, *MinInt* 1)	2,496	0.7931	2.110	2.08	2,756	0.8070	2.798	2.30
scaled (4X, *MinInt* 0)	5,088	0.7931	2.166	4.24	4,816	0.7712	2.589	4.01
scaled (4X, *MinInt* 1)	5,088	0.7931	2.166	4.24	4,816	0.7712	2.589	4.01
scaled (4X, *MinInt* 2)	4,800	0.7931	2.310	4.00	4,816	0.7712	2.589	4.01
scaled (8X, *MinInt* 3)	9,600	0.7931	2.093	8.00	9,240	0.7725	2.747	7.70
scaled (16, *MinInt* 4)	19,864	0.7931	2.157	16.55	18,575	0.7834	2.799	15.48

Table 2. CylUnstruct model scaled data.

	Structured Blocks Only (Figure 11)				Swept Blocks (Figure 12)			
	#elems	Min Scaled Jacobian	Max Aspect Ratio	Effective Multiplier	#elems	Min Scaled Jacobian	Max Aspect Ratio	Effective Multiplier
original Mesh	680	0.7947	1.547	-	680	0.7947	1.547	-
scaled (2X, *MinInt* 0)	1,932	0.7947	2.673	2.84	1,586	0.7672	1.953	2.332
scaled (2X, *MinInt* 1)	2,304	0.7947	2.800	3.39	1,586	0.7672	1.953	2.332
scaled (4X, *MinInt* 0)	2,790	0.7947	2.589	4.10	2,864	0.7848	2.093	4.212
scaled (4X, *MinInt* 1)	3,120	0.7947	2.239	4.59	2,864	0.7848	2.093	4.212
scaled (4X, *MinInt* 2)	5,670	0.7947	3.452	8.34	2,864	0.7848	2.093	4.212
scaled (8X, *MinInt* 3)	13,120	0.7947	3.497	19.29	5,760	0.7375	2.047	8.471
scaled (16, *MinInt* 4)	25,250	0.7947	3.523	37.13	11,275	0.8107	2.068	16.581

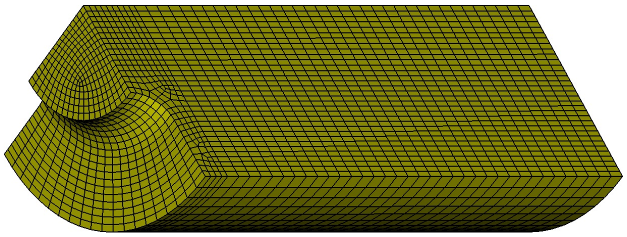

Fig. 13. 1/8th symmetry Notch model: base mesh.

strain (EQPS) and the integrated surface reaction force. Each line represents a point in the load history (running from 20 to 80), where we plot relative error as a function of mesh size. Based on such data, an analyst can select a mesh with desired accuracy. For example, we estimate the relative error in max EQPS using the base mesh to range from 30-80% over the entire load history.

While we report data on a total of ten meshes, only five were used in the solution verification and the finest mesh used had only four times the number of elements used in the coarse mesh. In addition, Figure 15 includes the estimated convergence rate p, expected to be near one for this application indicating first order convergence under mesh refinement. With the exception of the earliest times in the load history (20 and 30), all the errors in the QoIs reduce at a relatively smooth rate, even though the meshes are not nested. This is a confirmation that meshes produced by mesh scaling can produce meshes suitable for solution verification with scale factors other than powers of eight.

Our second example is the more complex PCAP model with base mesh of nearly two million hex elements. In this case the loading was using a time-dependent pressure on the interior of the structure. The QoIs of interest were the max EQPS within two regions representing weld zones. The original mesh and five scaled meshes are shown

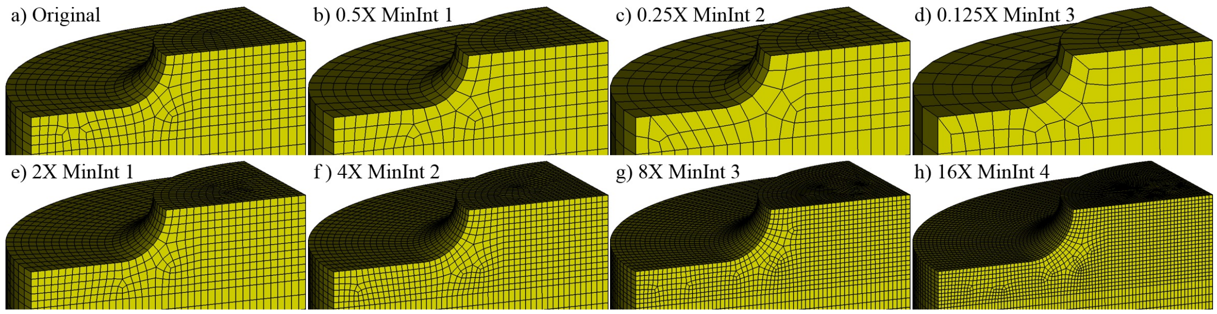

Fig. 14. 1/8th symmetry Notch model: scaled meshes.

Table 3. Notch model scaled data, scaled with swept blocks.

	#elems	Min Scaled Jacobian	Max Aspect Ratio	Effective Multiplier
original Mesh	22,860	0.73	3.55	-
scaled (0.125X, *MinInt* 3)	3,189	0.55	3.05	0.14
scaled (0.25X, *MinInt* 2)	6,216	0.72	4.12	0.27
scaled (0.50X, *MinInt* 1)	11,675	0.76	3.61	0.51
scaled (2X, *MinInt* 1)	46,395	0.75	3.76	2.03
scaled (4X, *MinInt* 2)	88,752	0.73	3.73	3.88
scaled (8X, *MinInt* 3)	178,768	0.72	3.97	7.82
scaled (16X, *MinInt* 4)	362,074	0.72	4.59	15.84

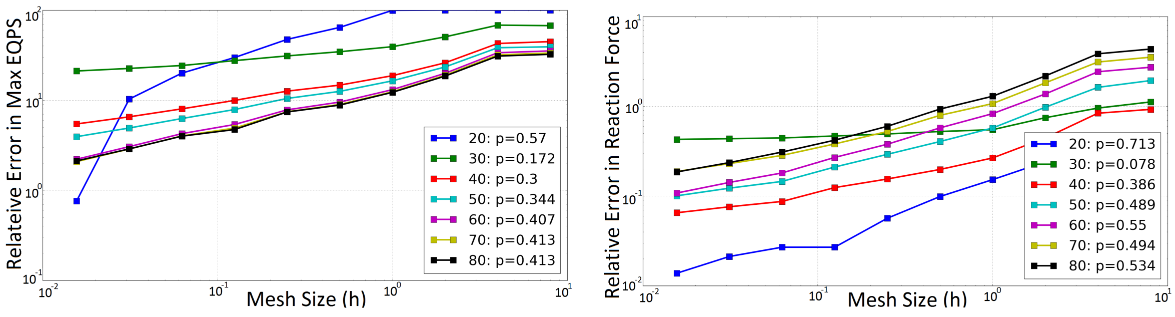

Fig. 15. Notch model: estimated errors in max EQPS (left) and reaction force (right) using five meshes centered at the base mesh.

in Figure 16 with data about the scaled meshes in Table 4, indicating the unstructured nature of the meshes and the complexity of the mesh.

As in the previous example, we estimate errors in max EQPS over the load history and present the results in Figure 17. Here we have plotted the max EQPS over the load history and have added the error estimate as an error bar. The error estimate is computed as before using nonlinear regression, but the results are presented in a more analyst-specific format. This enables the change in the error estimate over the load history to be more apparent. The scale factors used in the solution verification were 0.5, 1, 2, and 4.

4. Conclusions

This paper presents Mesh Scaling, a new global mesh refinement/coarsening algorithm for generating series of all-hexahedral meshes for solution verification without the limitations of the UMR 8X template. Mesh Scaling also provides limited coarsening, which is not possible with standard UMR. Solution verification on two models has been presented demostrating the applicability of meshes produced by Mesh Scaling to solution verification. Mesh Scaling provides the flexibility of scaling the mesh by any positive floating point multiplier, allowing analysts to create a much

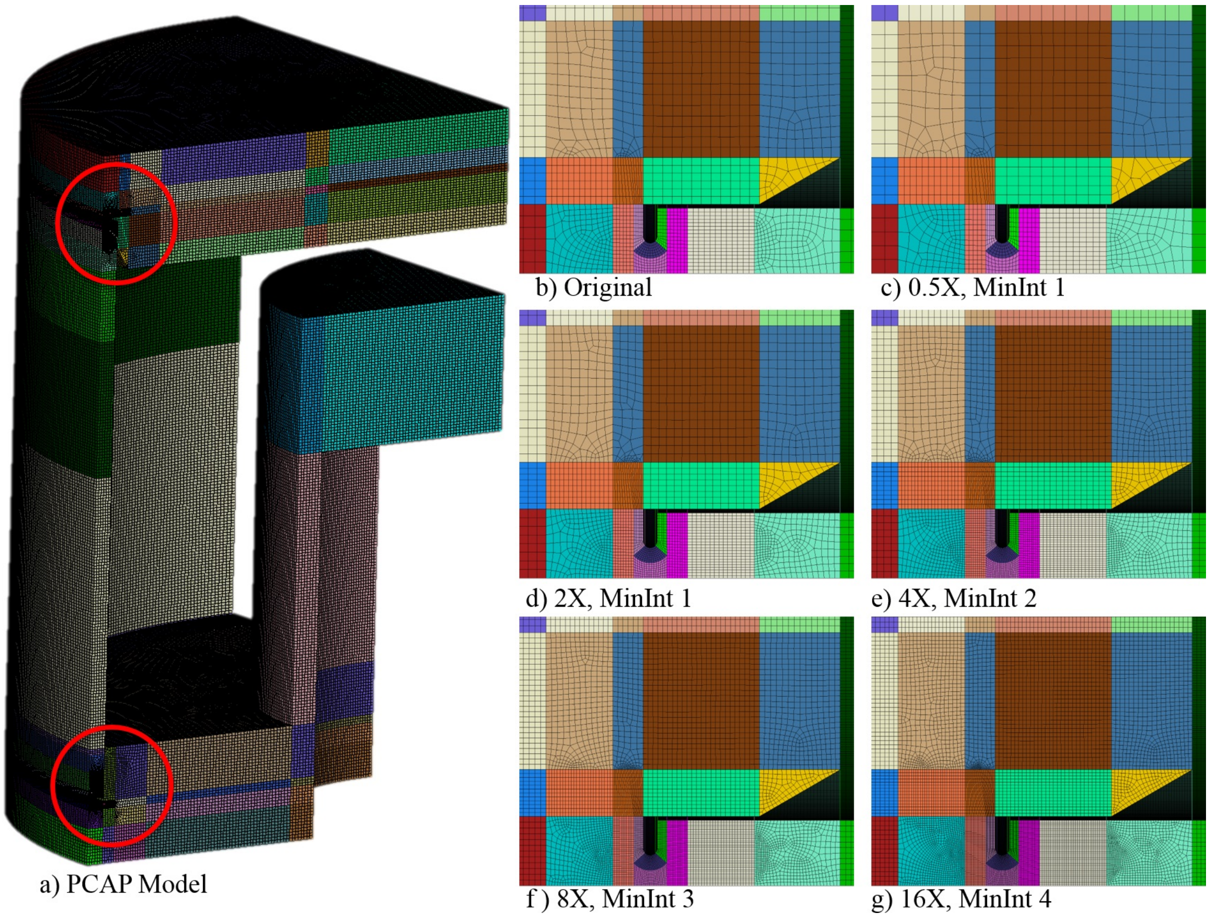

Fig. 16. 1/4 symmetry PCAP model.

Table 4. PCAP model scaled data, scaled with swept blocks.

	#elems	Min Scaled Jacobian	Max Aspect Ratio	Effective Multiplier
original Mesh	1,850,944	0.50	13.81	-
scaled (0.25X, *MinInt* 2)	898,787	0.43	22.40	0.48
scaled (0.50X, *MinInt* 1)	1,144,872	0.43	15.67	0.61
scaled (2X, *MinInt* 1)	3,869,929	0.50	14.73	2.09
scaled (4X, *MinInt* 2)	7,412,602	0.50	17.53	4.00
scaled (8X, *MinInt* 3)	14,807,552	0.50	14.94	8.00
scaled (16X, *MinInt* 4)	30,337,006	0.50	14.21	16.39

larger series of meshes, providing more data points for solution convergence plots, at a fraction of the computational cost required when using UMR.

The weakest point in the Mesh Scaling algorithm is its reliance upon the paver [2]. Paving's non-determistic results cause Mesh Scaling to undershoot and overshoot the target multiplier. Further, parallelization of Mesh Scaling for HPC is limited as long as it depends upon paving. We have already begun implementation on a new *Hybrid* version of Mesh Scaling which constructs two block decompositions, one with entirely structured blocks, and one with swept blocks. We anticipate that mapping between these two decompositions will allow for complete elimination of the paver, enabling parallelization, and providing for more deterministic scaled meshes.

Mesh Scaling could also be reconfigured for localized mesh refinement by computing new intervals on some block decomposition curves, while leaving other intervals unchanged.

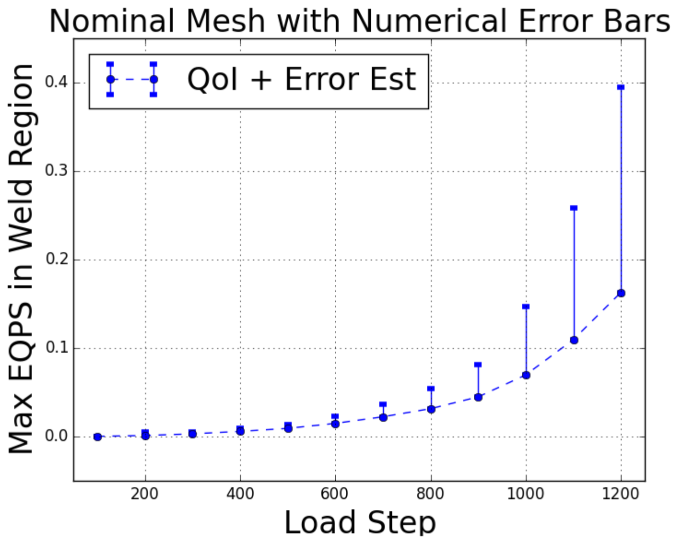

Fig. 17. PCAP model: max EQPS for base mesh over load history with error estimate.

Acknowledgements

- Notch model provided by George Orient, Sandia National Laboratories.
- PCAP model provided by Amalia Black, Sandia National Laboratories.
- Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. XXXX-XXXXX.

References

[1] T. D. Blacker, The Cooper Tool, Proceedings 5th International Meshing Roundtable, (1996) 13–30.
[2] T. D. Blacker, M. B. Stephenson, Paving: A New Approach to Automated Quadrilateral Mesh Generation, INT J NUMER METH ENG. 32(4) (1991) 811–847.
[3] P. Cargemel, Y. Mesri, H. Guillard, Conformal Hexahedral Mesh Coarening by Agglomeration, Research Note, 23rd International Meshing Roundtable, (2014).
[4] P. M. Knupp, Next-Generation Sweep Tool: A Method for Generating All-Hex meshes on Two-And-One-Half Dimensional Geometries, Proceedings 7th International Meshing Roundtable, (1998) 505–513.
[5] S. A. Mitchell, High Fidelity Interval Assignment, Proceedings 6th International Meshing Roundtable, (1997) 33–44.
[6] S. A. Mitchell, Simple and Fast Interval Assignment Using Nonlinear and Piecewise Linear Objectives, Proceedings 22nd International Meshing Roundtable, (2014) 203–221.
[7] P. Murdoch, S. E. Benzley, T. D. Blacker, and S. A. Mitchell. The spatial twist continuum: A connectivity based method for representing all hexaedral finite element meshes. FINITE ELEM ANAL DES. 28(137) (1997) 137–149.
[8] N. Kowalski, F. Ledoux, P. Frey, Automatic Domain Partitioning for Quadrilateral Meshing With Line Constraints, ENG COMPUT. 31 (2015) 405–421.
[9] M. Lyon, D. Bommes, L. Kobbelt, HexEx: Robust Hexahedral Mesh Extraction, SIGGRAPH 2016, ACM Transactions on Graphics. 35(4) (2016).
[10] M. L. Staten, S. A. Canann, S. J. Owen, BMSweep: Locating Interior Nodes During Sweeping, ENG COMPUT. 15 (1999) 212–218.
[11] N. Ray, D. Sokolov, On Smooth 3D Frame Field Design, Cornell University Library. http://arxiv.org/abs/1507.03351.
[12] M. L. Staten, K. Shimada, A Close Look at Valences in Hexahedral Meshes, INT J NUMER METH ENG. 83(7) (2010) 899–914.
[13] M. L. Staten, J. F. Shepherd, F. Ledoux, K. Shimada, Hexahedral Mesh Matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces, INT J NUMER METH ENG. 82(12) (2009) 1475–1509.
[14] Los Alamos National Laboratory, KIVA-3: A KIVA Program with Block-structured Mesh for Complex Geometries, (1993).
[15] http://www.cd-adapco.com/.
[16] http://www.pointwise.com/pw.
[17] http://www.truegrid.com/.