Engineering with Computers

Parallel Mesh Adaptation for High-Order Finite Element Methods with Curved Element Geometry

--Manuscript Draft--

Manuscript Number:	
Full Title:	Parallel Mesh Adaptation for High-Order Finite Element Methods with Curved Element Geometry
Article Type:	S.I.: IMR21 2012
Corresponding Author:	Qiukai Lu Rensselaer Polytechnic Institute Troy, NY UNITED STATES
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Rensselaer Polytechnic Institute
Corresponding Author's Secondary Institution:	
First Author:	Qiukai Lu
First Author Secondary Information:	
Order of Authors:	Qiukai Lu
	Mark S. Shephard, Dr
	Saurabh Tendulkar
	Mark W. Beall, Dr
Order of Authors Secondary Information:	
Abstract:	This paper presents a parallel adaptive mesh control procedure designed to operate with high-order finite element analysis packages to enable large scale automated simulations on massively parallel computers. The curved mesh adaptation procedure uses curved entity mesh modification operations that explicitly consider the influence of the curved mesh entities on element shape. Applications of the curved mesh adaptation procedure have been developed to support the parallel automated adaptive accelerator simulations at SLAC National Accelerator Laboratory.
Suggested Reviewers:	

Engineering with Computers manuscript No.
(will be inserted by the editor)

Parallel Mesh Adaptation for High-Order Finite Element Methods with Curved Element Geometry

Qiukai Lu · Mark S. Shephard · Saurabh Tendulkar · Mark W. Beall

Received: date / Accepted: date

Abstract This paper presents a parallel adaptive mesh control procedure designed to operate with high-order finite element analysis packages to enable large scale automated simulations on massively parallel computers. The curved mesh adaptation procedure uses curved entity mesh modification operations that explicitly consider the influence of the curved mesh entities on element shape. Applications of the curved mesh adaptation procedure have been developed to support the parallel automated adaptive accelerator simulations at SLAC National Accelerator Laboratory.

Keywords Curved Mesh · High-order Finite Element · Parallel Mesh Adaptation

1 Introduction

High-order finite element methods have the advantage of achieving exponential rates of convergence to problems of interest [33]. In order to fully realize the benefits of the methods in the simulation on general 3D domains with curved boundary geometry, techniques must be developed to construct valid meshes with properly curved elements that approximate the curved domain geometry to the correct order. For simulations that require the numerical solutions to have extremely high accuracy, high mesh resolution is required in critical regions. Even when taking advantage of the benefits of adaptively refined meshes, element counts in the millions are common. Such meshes can only be created and analyzed using large scale parallel computing systems, which requires effective parallel mesh generation and adaptation techniques [4].

The majority of the efforts on mesh generation and adaptation have been focused on dealing with conventional meshes with all straight-sided elements. To date there has been limited effort devoted to studying mesh generation and adaptation techniques for fully unstructured curved meshes [6,10,11,28]. The work presented in this paper tackles the challenges of developing effective adaptive mesh control procedures for distributed curved meshes to support large scale parallel simulations using high-order finite elements. The paper is organized as follows. Section 2 presents a general element shape measure that is generally applicable to both straight-sided and curved elements. Based on the shape measure, two validity check algorithms for detecting invalid curved elements are presented in Section 3. Section 4 presents a set of local mesh modification operations that are used by the curved mesh adaptation procedure. Section 5 outlines the parallel execution of curved mesh adaptation and presents an adaptive simulation loop which utilizes the developed parallel curved mesh adaptation procedures. Application examples are presented to demonstrate the effectiveness of the adaptive loop are given in Section 6.

The nomenclature used in this paper is defined as follows:

Q. Lu
Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY.
E-mail: luq3@rpi.edu

M.S. Shephard
Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY.
E-mail: shephard@rpi.edu

S. Tendulkar
Simmetrix Inc. Clifton Park, NY.
E-mail: saurabh@simmetrix.com

M.W. Beall
Simmetrix Inc. Clifton Park, NY.
E-mail: mbeall@simmetrix.com

Ω_v	Domain of interest, $v = G, M$ where G denotes the geometric model and M denotes the mesh model
$\partial\Omega_v$	Boundary of the domain Ω_v
G_i^d	ith geometric model entity of dimension d.
M_i^d	ith mesh entity of dimension d. $d = 0, 1, 2, 3$ and represents mesh vertex, edge, face and region respectively.
$\sqsubset$	Classification symbol used to indicate the association of one or more entities from the mesh model M with the geometric model G.
M^d	Unordered group of mesh topological entities of dimension d.
$M_i^{d_i}\{M_j^d\}$	First order adjacency sets of individual mesh entity $M_i^{d_i}$ defined as the set of mesh entities of dimension d_j adjacent to mesh entity $M_i^{d_i}$.
$P_i^{(n)}(M_j^d)$	the ith control point of a nth order Bézier polynomial associated with the mesh entity M_j^d.
$X^{(n)}(M_j^3)$	the nth order Bézier polynomial representation of a general tetrahedron.

2 A General Element Shape Quality Measure

There has been substantial effort in developing mesh quality measures for straight-sided simplex meshes. Knupp defines a mesh quality metric as a scalar function that measures certain geometric property of that given element [12]. In the case of a straight-sided tetrahedral element, the geometric shape is uniquely determined by the positions of the four vertices. It is straight-forward to calculate various geometric quantities to evaluate the shape of the element, such as edge length, solid angle. Several commonly used straight-sided mesh entity shape measures are reviewed in the references [14,15, 20]. Ciarlet [3] and Shewchuk [32] have discussed element quality measures focusing on interpolation errors. Knupp [13] introduces the application-specific algebraic mesh quality metrics that contain an user defined reference that maps local geometric quality to quality characteristics that are relevant to specific application problems.

On the other hand There has been little consideration given to shape measures for curved elements largely due to the complexities involved in calculating the geometric quantities such as length of a curved edge, area of a curved surface or volume of a curved region. One shape measure for curved tetrahedron evaluates the shape of a curved mesh entity by calculating the scaled variations of the determinant of Jacobian over the element domain [6,11,23,28]. The shape measure is defined as:

$$q_c = \frac{\min_{\xi \in \Omega^e} det(J(\xi))}{\max_{\xi \in \Omega^e} det(J(\xi))} \tag{1}$$

This measure gives important information about how distorted the tetrahedron is with reference to the curving of mesh edges and faces. However it does not take into account the shape of the underlying straight-sided frame of the curved element. For straight-sided elements, this shape measure q_c always reports the optimal value: 1, regardless of whether the element is highly anisotropic and/or close to being degenerated. Therefore, q_c can not serve as a general shape measure for curved elements.

A general shape measure that overcomes the aforementioned issue can be constructed using a multiplicative element shape measure that combines a straight-sided entity shape measure and a curved mesh entity shape measure as follows:

$$Q_{sc} = q_s \times q_c \tag{2}$$

where q_s is a selected normalized shape measure for straight-sided elements [15, 20] and q_c for curved elements.

In the case of a straight-sided element where $q_c = 1$, $Q_{sc} = q_s$ measures the element shape. For a curved element, q_c contributes to Q_{sc} along with the straight-sided shape q_s. This general shape measure is adopted to serve as the basis to support the curved mesh modification operations and the mesh quality improvement stages of the curved mesh adaptation algorithm presented in Section 5.4.

3 Curved Mesh Validity Checks

A valid curved mesh is essential to the successful execution of high-order finite element simulations. To verify a valid curved tetrahedral element independent of the numerical integration scheme being used, it is necessary to ensure the determinant of the Jacobian is positive throughout the domain of the element. One effective method to address this evaluation is to evaluate a lower bound for the determinant of Jacobian.

By applying the Bézier polynomial based entity representation to a high-order curved tetrahedron, the determinant of the Jacobian $det(J)$ can be represented as a scalar Bézier polynomial of order $3(p-1)$ defined over the element domain, where p is the order of the vector Bézier polynomial representing the element. According to the convex hull property [7] the $det(J)$ is bounded

by the maximum and minimum values evaluated at the control points of the order $3(p-1)$ Bézier polynomial. In the case of a quadratic tetrahedral element, the following inequality holds:

$$\min\{P^{(3)}_{|i|}\} \leq det(J) \leq \max\{P^{(3)}_{|i|}\} \tag{3}$$

where $P^{(3)}_{|i|}$ represents the scalar value at the ith control point. Naturally, a sufficient condition to ensure positive determinant of Jacobian for a pth order curved tetrahedron is that the minimum value of all control points $\min\{P^{(3(p-1))}_{|i|}\}$ is positive. More specifically for a quadratic tetrahedron: $\min\{P^{(3)}_{|i|}\} > 0$

The uniform validity check algorithm [23] is based on the above condition by checking all the control points of the Bézier polynomial of $det(J)$. In the case of a quadratic tetrahedral element, the total number of control points is 20. This algorithm is computationally efficient and is independent of the numerical integration schemes. However, since it uses a sufficient condition that's based on the lower bound of $det(J)$ in the Bézier form, there can be overly-conservative in situations where the lower bound is not tight. In such cases, the actual value of $\min\{det(J)\}$ could be positive over the element domain whereas $\min\{P^{(3)}_{|i|}\}$ is negative and leads to false negative report by the uniform validity check.

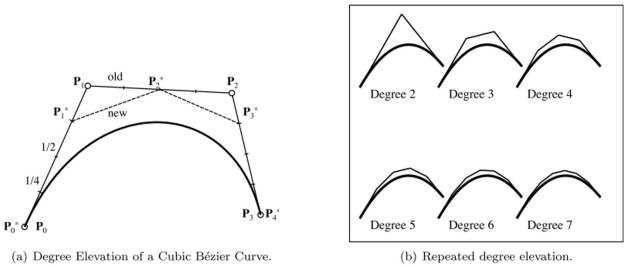
Although the uniform validity check is sometimes overly-conservative, it is nevertheless an effective method to determine the candidate mesh entities with which the potential invalidity is associated. Given a negative lower bound at a specific control point it is possible to identify which curved mesh entities are causing the bound to be negative. Specifically, if the control point is associated with a mesh vertex M^0, the edges connected to the vertex are the candidate entities $M^0\{M^1\}$. If the control point is associated with a mesh edge M^1, that particular edge M^1 is the candidate entity. If the control point is associated with a mesh face M^2, the edges that bound the face are candidates $M^2\{M^1\}$ [23]. Once the candidate mesh entities are identified, one of the approximation improvement methods is applied to obtain tighter bounds for $det(J)$ to ultimately determine if there is a negative Jacobian at some point in the element.

3.1 Adaptive Validity Check By Degree Elevation

The degree elevation algorithm [30] increases the total number of control points of a Bézier polynomial by elevating the polynomial degree, making the control polygon converge to the actual polynomial as the number of control points approaches infinity [7]. It can be applied to the key entities to refine its control polygon of its Bézier representation to give a tighter lower bound. For example, if $\min\{P^{3(p-1)}_{|i|}\} < 0$ is reported at an edge control point, the degree elevation check will elevate the degree of the polynomial representing that edge based on all the control points associated with it. For a quadratic curved element, the original representation of $det(J)$ is a 3rd-order Bézier polynomial, therefore 4 control points are associated with an edge, denoted by $P^{(3)}_{|3000|}$, $P^{(3)}_{|2001|}$, $.P^{(3)}_{|1002|}$, $P^{(3)}_{|0003|}$. The control points after one step of degree elevation from order 3 to order 4 can be calculated by [7]:

$$\begin{aligned}
P^{(4)}_{|4000|} &= P^{(3)}_{|3000|} \\
P^{(4)}_{|0004|} &= P^{(3)}_{|0003|} \\
P^{(4)}_{|3001|} &= \tfrac{1}{4}P^{(3)}_{|3000|} + \tfrac{3}{4}P^{(3)}_{|2001|} \\
P^{(4)}_{|1003|} &= \tfrac{3}{4}P^{(3)}_{|1002|} + \tfrac{1}{4}P^{(3)}_{|0003|} \\
P^{(4)}_{|2002|} &= \tfrac{1}{2}P^{(3)}_{|2001|} + \tfrac{1}{2}P^{(3)}_{|1002|}
\end{aligned} \tag{4}$$

This can be generalized to obtain control points of any degree n elevated from degree $n-1$. As the polynomial degree gets elevated step by step, the number of control points increases and the control polygon becomes closer to the actual curve, and therefore gives tighter lower and upper bounds. According to [25,29], the authors showed that the convergence rate is $O(\frac{1}{\nu})$, where ν is the polynomial order. Fig 1 illustrates the convergence process of degree elevation to a 2D Bézier curve [30].



(a) Degree Elevation of a Cubic Bézier Curve. (b) Repeated degree elevation.

Fig. 1 Convergence of Degree Elevation [30]

3.2 Adaptive Validity Check By Subdivision

In addition to the degree elevation algorithm, the polynomial subdivision algorithm [30] also yields more control points and tighter control polygon for a Bézier polynomial. Thus it can also be applied to the key entities to improve the lower bound [21]. Take an edge as the

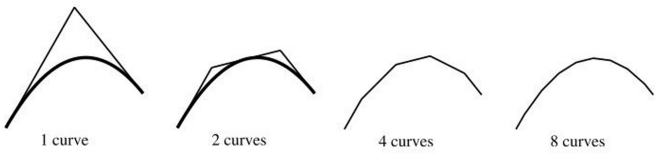


Fig. 2 Convergence of polynomial subdivision [30]

example entity again, one step of the subdivision algorithm divides the original 3rd-order Bézier polynomial associated with the edge into two 3rd-order polynomials by applying the *de Casteljau* algorithm [7] as follows:

$$
\begin{aligned}
P_0^1 &= \tfrac{1}{2}P^{(3)}_{|3000|} + \tfrac{1}{2}P^{(3)}_{|2001|}, P_0^2 = \tfrac{1}{2}P_0^1 + \tfrac{1}{2}P_1^1 \\
P_1^1 &= \tfrac{1}{2}P^{(3)}_{|2001|} + \tfrac{1}{2}P^{(3)}_{|1002|}, P_1^2 = \tfrac{1}{2}P_1^1 + \tfrac{1}{2}P_2^1 \\
P_2^1 &= \tfrac{1}{2}P^{(3)}_{|1002|} + \tfrac{1}{2}P^{(3)}_{|0003|}, P_0^3 = \tfrac{1}{2}P_0^2 + \tfrac{1}{2}P_1^2
\end{aligned} \tag{5}
$$

The new sets of Control points for the two polynomials are then: $\{P^{(3)}_{|3000|},\ P_0^1,\ P_0^2,\ P_0^3\}$ and $\{P_0^3,\ P_1^2,\ P_2^1,\ P^{(3)}_{|0003|}\}$. Note that P_1^1 is not used as a new control point. A recent paper by George et al also discusses this method [10].

It is straight-forward to obtain more control points if one keeps doing subdivision recursively. And it is obvious that the control polygon gets closer to the polynomial itself as the subdivision steps continue. According to [25,29], the control polygon converges to the curve with the rate of convergence $O(\frac{1}{2^i})$, where i is the number of subdivision steps. Fig 2 gives an example of a 2D Bézier curve and its control polygons after several steps of subdivision [30].

3.3 Stopping Criteria for the Adaptive Validity Checks

After applying either of the adaptive validity check algorithms, if $\min\{P^{(n)}_{|i|}\}$ is found to be positive, the element is valid. On the other hand, the element is invalid if negative $\min\{P^{(n)}_{|i|}\}$ is found at any control point that is interpolating the value of $det(J)$. In both cases, the algorithm will stop accordingly. However if negative $\min\{P^{(n)}_{|i|}\}$ appears at a non-interpolating control point while positive values are found at all interpolating control points during finite steps, it is necessary to terminate the algorithm to avoid infinite loops of checking and refinement. The stopping criterion used is to evaluate the increment of the lower bound $\Delta\min\{P^{(n)}_{|i|}\}$ after each step. If negative $\min\{P^{(n)}_{|i|}\}$ is still reported after $\Delta\min\{P^{(n)}_{|i|}\} < \epsilon$, where ϵ is a prescribed tolerance, then the element is regarded as invalid and the algorithm stops at the current step. See Algorithm 1.

```
Data: A quadratic curved tetrahedral element,
      prescribed tolerance for relative increments ε
compute min{P^(n)_|i|} ;
if min{P^(n)_|i|} > 0 then
    return: the element is VALID ;
else
    if negative min{P^(n)_|i|} at interpolating point then
        return: VALID ;
    else
        while relative increment > ε do
            get the mesh entity associated with the
            negative control point ;
            apply subdivision to the mesh entity ;
            update the new min{P^(n)_|i|} ;
            if min{P^(n)_|i|} > 0 then
                return: VALID ;
            else
                if new negative min{P^(n)_|i|} at
                interpolating point then
                    return: INVALID ;
                else
                    compute the relative increment ;
                    if relative increment < ε then
                        return: INVALID ;
                    else
                        update with new relative
                        increment ;
                    end
                end
            end
        end
    end
end
```

Algorithm 1: Algorithm for the termination of the adaptive subdivision validity check method

It is worth noting that the subdivision algorithm gives more interpolation points in addition to the vertex control points. This property could serve as an addition to the stopping criterion. If any of the newly-computed interpolating control points after a subdivision step has negative value, it indicates that $det(J)$ at this point is negative and the adaptive check stops and reports the element as invalid.

3.4 Numerical Results and Some Observations

A mesh with 4 partitions of around 45k tetrahedral elements including quadratic curved tetrahedral elements are used and numerical experiments are carried out to study the 4 worst-shaped regions of the 4 parts (one on each part) reported by the uniform validity check method. In each case, a brute-force search was used to find the exact value of $det(J)$ at a large number of sample points, and $\min\{det(J)\}$ was found in order to com-

pare the results with the adaptive checks. For the ones with $\min\{P^{(3)}_{|i|}\}$ found at an edge, the adaptive checks using degree elevation and subdivision algorithms were applied respectively. Results are given in Table 1.

In all four cases, $\min\{P^{(3)}_{|i|}\}$ appeared at the control point associated with either a mesh vertex or a mesh edge denoted as key entity. In each case, the key entity (vertex or edge) determined by $\min\{P^{(3)}_{|i|}\}$ of the uniform validity check was consistent with the key entity found by the brute-force search of $\min\{det(J)\}$. This shows the effectiveness of the uniform validity check to determine key entities.

In the second case that $\min\{det(J)\}$ was found at a mesh vertex by brute-force search, the $\min\{P^{(3)}_{|i|}\}$ had the exact same value at the control point of the vertex as $\min\{det(J)\}$. This is because the control points at the vertices are interpolation points. Therefore when the invalidity ($\min\{det(J)\} \leq 0$) happens at a vertex, the uniform validity check method is able to accurately detect it, and is not conservative in those cases.

In the first, third and fourth cases that $\min\{det(J)\}$ were at a mesh edge and were all positive, $\min\{P^{(3)}_{|i|}\}$ gave conservative lower bounds with negative values. The adaptive validity checks were called to iteratively refine the control polygons and tighten the lower bounds. The stopping criterion was set to stop the refinement after the step in which all positive valued control points were obtained.[1] In the third and fourth cases, only one step of refinement (degree elevation or subdivision) was needed to obtain all positive control points (i.e. positive $\min\{det(J)\}$). However, in the first case, four steps of degree elevation were needed to raise the polynomial representation to 7th-order to get all positive control points. In this same case, only one step of subdivision was needed to get all positive control points. One could also observe that in these three cases, after one step of refinement, subdivision always gave a tighter lower bound than what degree elevation gave. The results are consistent with the theoretical rate of convergence for subdivision and degree elevation algorithms, i.e., subdivision converges faster than degree elevation [25,29].

In terms of the efficiency of the adaptive checks, one could also estimate the additional cost by counting the floating point operations (flops) required. Doing the original uniform validity check requires to perform $1 \times 4 = 4$ box products at mesh vertices, $3 \times 12 = 36$ box products at the edges, and $6 \times 4 = 24$ at the faces, which is 64 box products in total. On the other hand, the subdivision or degree elevation algorithm is essentially calculating the weighted average of the 3rd-order control points. Therefore, if only one step of adaptive refinement is needed in addition to the uniform validity check calculation, the added computational cost is 9 flops for degree elevation and 12 flops for subdivision algorithm, thus is non-substantial compared with what the uniform check requires.

A test case that compares the performance of the uniform validity check and the one with adaptive subdivision was conducted on a test mesh with 60k curved elements. Four repeated runs were done for both methods and the performance results in terms of mesh reading, writing and validity check are given in Table 2 and Table 3. A slight increase of computation time for the adaptive validity check is observed from 14.0% of the total time to 14.6%, which is relatively small and quite acceptable.

[1] Note that this criterion is different from the one being discussed in Section 3.3 and can be used only in such particular cases that the element is in fact valid since it could eventually stop. In other cases, the incremental criterion should be used.

4 Mesh Modification of Curved Mesh Entities

Automated adaptive finite element simulations using unstructured 3D meshes rely on mesh adaptation to alter the connectivity and/or the geometry of the mesh dictated as by a mesh size field produced by *a posteriori* error estimation and correction indication procedures [18,19]. An effective approach to obtain such an adapted mesh with the desired element sizes is to apply local mesh modification operations to appropriate groups of mesh entities referred to as mesh cavities [2, 19]. Specific mesh modification operations have been designed and implemented to deal with curved mesh entities.

4.1 Entity Geometry Modification Operations

A set of entity geometry modification operations have been designed and implemented to work with curved mesh elements. These operations only change element shape and do not alter the mesh connectivity. Geometry modification operation is required to increase geometric approximation accuracy to the curved model domain when element approximation order is increased. They can also be of great value to eliminate mesh invalidity, or improve mesh shape quality.

4.1.1 Mesh Curving Operation for Surface Entities Classified on Model Boundaries

A mesh curving operation converts a linear straight-sided mesh entity to a high-order curved entity by introducing high-order nodes. For mesh entities classified on curved model boundaries, those nodes are snapped

Table 1 Numerical results of the fixed and adaptive validity checks on four test regions

Element	min{$det(J)$}	max{$det(J)$}	min{$P^{(3)}_{\|i\|}$}	max{$P^{(3)}_{\|i\|}$}
Region1	0.2809	12.7292	-0.4283	12.7292
Key entity	M^1_3	M^0_2	M^1_3	M^0_2
Region2	0.5324e-3	2.4996	0.5324e-3	2.4996
Key entity	M^0_3	M^0_0	M^0_3	M^0_0
Region3	36.0230	176.0338	-0.2049	176.0338
Key entity	M^1_3	M^0_2	M^1_3	M^0_2
Region4	0.7574	13.5732	-0.1702	13.5732
Key entity	M^1_3	M^0_0	M^1_3	M^0_0

Element	min{$P^{(3)}_{\|i\|}$} after sub.div.	min{$P^{(4)}_{\|i\|}$} (4th-order)	min{$P^{(7)}_{\|i\|}$} (7th-order)
Region1	0.6952e-1	-0.1459	0.1755e-1
Region2	0.5324e-3	0.5324e-3	n/a
Region3	26.4814	15.6570	n/a
Region4	0.6762	0.7857e-4	n/a

Table 2 Timing results of non-adaptive validity check (units: sec)

	Test1	Test2	Test3	Test4	Average	Percentage
Read	2.298	2.318	2.311	2.297	2.306	27.6%
Check	1.156	1.185	1.165	1.164	1.167	14.0%
Write	4.844	4.881	4.787	4.958	4.867	58.4%

Table 3 Timing results of adaptive check with subdivision (units: sec)

	Test1	Test2	Test3	Test4	Average	Percentage
Read	2.282	2.287	2.271	2.275	2.278	27.1%
Check	1.232	1.229	1.237	1.221	1.229	14.6%
Write	4.979	4.895	4.829	4.919	4.905	58.3%

to the curved geometry to ensure geometric approximation accuracy. Dey et al presented a mesh curving algorithm in [6], which interacts directly with the underlying CAD modeling engine to obtain the proper geometric location of the high-order nodes on the model boundaries based on parametric interrogations.

Only curving the boundary mesh entities may cause self-intersecting mesh entities which leads to invalid elements. In such occasions, interior mesh entities may be selected to be curved to correct the invalidity. For example in a 2D case shown in Fig 3, mesh edges M^1_0 and M^1_1 are curved to model edge G^1_0. However elements M^2_0 and M^2_1 become invalid (Fig 3(b)). In this case, the interior edges M^1_2 and M^1_3 must be reshaped into a curved shape (Fig 3(c)) to ensure element validity since performing an edge swap will still yield an invalid element and a collapse is not possible. Although this case required the application of element reshaping, there can be cases where applying a swap or collapse operation can be preferred.

4.1.2 Entity Reshaping Operation for Interior Mesh Entities

For a linear straight-sided element, the shapes of its edges and faces are uniquely defined by the end vertices. In this case one can only reshape a straight-sided mesh entity by repositioning its end vertices. There have been extensive efforts devoted to developing various algorithms for the vertex reposition or smoothing operations [5,8,15,19]. For a high-order curved mesh, the shape of the curved mesh edges and faces depend not only on the position of the end vertices, but also the high-order nodes associated with the mesh entities or other entity shape parameters such as the control polygon. The developed curved element reshaping algorithm considers curved shape parameters in addition to the vertices. It computes the shape quality as defined in Equation 2 based on the calculation of q_s and q_c re-

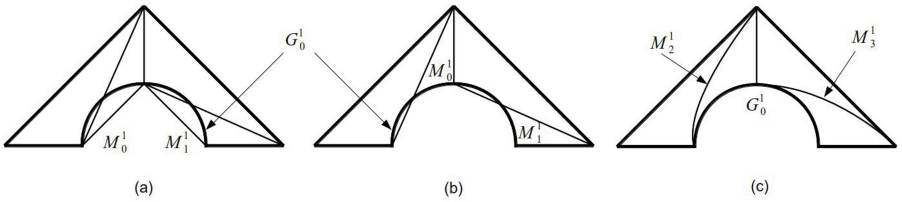


Fig. 3 Curving interior mesh edges to resolve the invalidity caused by curving boundary edges

spectively. The mean ratio shape measure [18] is used to compute q_s as defined in Equation 6.

$$q_s = \frac{15552 \times V^2}{\left(\sum l_i^2\right)^3} \quad (i = 1, 2, ..., 6) \tag{6}$$

The scaled Jacobian variation shape measure is used to compute q_c as defined in Equation 1. The calculated q_s and q_c are used to compare with an application-specified shape quality threshold q_{limit} (0.1 used in the examples presented in this paper). The curved element reshaping algorithm deals with the following cases:

Case 1: If $q_s < q_{limit}$ *and* $q_c > q_{limit}$, in this case, it indicates the straight-sided shape quality q_s is not acceptable. Thus a higher priority is given to improve q_s firstly for the current element. Such a tetrahedron will be processed by the shape improvement algorithm for straight-sided elements. Refer to [5,19] for details.

Case 2: If $q_s > q_{limit}$ *and* $q_c < q_{limit}$, the straight-sided shape component of this region q_s is acceptable to the application. Thus consideration is given to improving the curved component of the shape quality q_c. The algorithm proceeds with the following 3 steps:

Step 1: Choose the candidate mesh entities to be reshaped.

As being defined by Equation 1, q_c can be improved by increasing $\min\{det(J)\}$. Therefore the candidate mesh entities chosen to be reshaped are the ones directly associated with the control point that has the minimum value of $det(J)$. If the control point is associated with a mesh vertex M^0, then the edges connected to the vertex are the candidate entities $M^0\{M^1\}$. If the control point is associated with a mesh edge M^1, this particular edge M^1 is the candidate entity. If the control point is associated with a mesh face M^2, the bounding edges of the mesh face are candidates $M^2\{M^1\}$.

Step 2: Determine the direction of motion.

Persson and Peraire proposed a curving procedure that requires solving an auxiliary Lagrange solid mechanics problem for each element that has interior edges that need to be curved [28]. Although it produces promising results, the cost of such a procedure is expensive for large meshes. Nevertheless, since the reason for reshaping interior edges is due to the fact that some other

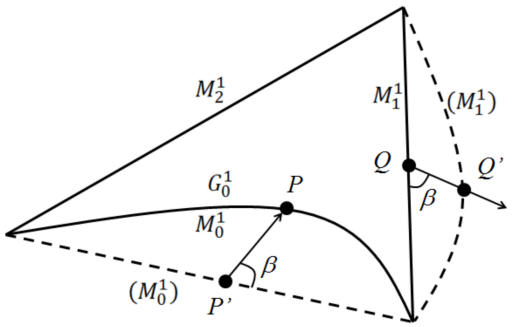


Fig. 4 Example case of an edge reshaping operation in 2D with one curved boundary edge M_0^1 classified on geometric model edge G_0^1

mesh entities on the boundary are highly curved and curving interior edge is the best option to ensure acceptable element shape quality, the direction of motion can be selected by taking into account the curved geometry of the boundary entity. More specifically in this algorithm, the direction of motion is determined to be along the direction defined by the edge mid-points of the curved boundary entity and its straight-edged counterpart. As an example shown in Fig 4, the mesh edge M_0^1 is classified on a curved model edge G_0^1, and is curved already. In order to improve the shape quality of the triangular element, one interior edge M_1^1 is curved as well. The angle β is calculated based on the movement of the edge mid-point of M_0^1: $\overline{P'P}$ and the straight-sided version of (M_0^1), which represents the curving direction of the boundary edge M_0^1. The direction of motion $\overline{QQ'}$ for curving M_1^1 is then determined such that it forms the same angle β with the straight-sided edge M_1^1 itself. Note that if there are two curved boundary edges, the direction of motion is taken to be the average of the two direction vectors of the curved boundary edges.

Step 3: Search the interval of acceptable motion for the local optimal location.

When reshaping the curved entity, the shape quality of the neighboring elements changes and often becomes lower. The limit of curving would be when the neighboring element becomes invalid which happens when the moving edge control point reaches the first problem plane, which is the first mesh face it will intersect

with. Reference [17] gives the precise definition of the first problem plane and its calculation with respect to a vertex relocation operation. For the reshaping operation, the first problem plane is defined and calculated in the same way for the edge to be curved. To avoid the invalidity caused by intersecting with the first problem plane, the interval of acceptable motion for the edge control point is defined to be the non-intersection line segment along the direction of motion from the original location of the control point to the intersecting point with the first problem plane. Given the interval of acceptable motion as the domain, the objective function for the local optimal search is set to maximize the minimum shape quality among all the affected elements as given in Equation 7.

$$maximize\{Q_{sc}(M_i^3)\forall M_i^3 \in M^3\{M_0^1\}\} \quad (7)$$

A golden section algorithm is adopted here to perform the search [5].

Case 3: If $q_s < q_{limit}$ and $q_c < q_{limit}$, There is the possible case of both q_s and q_c are not acceptable. In such a case, it is usually more effective to apply local mesh modification operations to improve the shape quality, thus this algorithm assigns higher priority to improving the straight-sided shape quality by applying possible local mesh modification operations such as collapse and/or swap. References [18,19] give detailed discussion of procedures used to improve the shape quality of straight-sided elements. After improving q_s to be over the q_{limit} threshold, the algorithm then focuses on improving q_c by following the same steps of Case 2. Note that such a priority is set also by taking into account the cost-effectiveness since the explicit search algorithm required in Step 3 of Case 2 is very computationally intensive [5].

4.2 Curved Local Mesh Modification Operations

Local mesh modification operations that change the connectivity of appropriate local mesh cavities have been shown to be effective and efficient for mesh adaptation procedures [18,19]. The specific local mesh modification operations used in this work consist of three unit operations: 1) splitting, 2) collapsing and 3) swapping, as well as compound operations which combine the unit operators in an ordered sequence.

Typically, A split operator works on edges with face and region also possible in three dimensional cases. The split operation inserts one or more new vertices on a target entity and subdivides the entity and the higher dimensional entities it bounds. In cases of edge/face split operations, if the target mesh edge or face to be split is a curved boundary entity that approximates the curved geometric model boundary, the newly-created vertex is placed on the model boundary [17,18]. The new edges and faces will also be curved accordingly to conform to the curved geometry [6,22]. There are also situations when splitting interior high-order curved mesh entities is desired. In such cases, the interior entity can be parametrized using the Bézier representation and split in the parametric space.

An edge collapse operation eliminates a target edge by collapsing one of the two end vertices into the other, and the high dimensional entities being bounded by the edge and vertices will be modified accordingly. Different from the split operation, a collapse operation is not always performable due to various constraints to ensure compatibility. In the context of high-order curved meshes, collapsing a curved boundary edge should consider the effect to the geometry approximation of model domain it will introduce. The curved edge collapse operation can be viewed as appending a set of mesh curving operations at the end of the straight-sided edge collapse operation. More specifically, when a curved boundary edge is to be collapsed, the topological modifications made to the mesh cavity stay the same as if it were a straight-sided mesh cavity. After obtaining a new cavity after collapsing, all the boundary edges that M_1^0 bounds will be processed and reshaped to conform to the model boundary using steps of parametric interrogations. If in some cases invalid self-intersecting elements are introduced by the newly curved boundary edges, curving of selected interior edges is necessary to correct the invalidity.

Swap operations change the connectivity of a local mesh cavity defined by all the regions that use the target entity (usually an edge or a face) $M_i^d\{M^3\}, d = 1, 2$. Similar to the collapse operation, when swap operation is to be performed on high-order curved mesh entities classified on geometric model boundary, one must address the new entity shapes after swapping to ensure that the mesh geometry still properly approximates the geometric model. Same as for collapse, selected interior edges are curved accordingly if invalid self-intersecting elements are introduced by the newly curved boundary edges.

4.2.1 The Full Set of Operations

In the application of curved mesh adaptation procedures, the curved local mesh modification operations are heavily used to refine and/or coarsen the initial mesh to conform to the desired mesh sizes. The entity geometry modification operations play an important

role in geometric approximation of the curved mesh boundary. In the process of shape quality improvement, often times, the two set of operations are combined to be carried out in a desired sequence. For instance, when curving an interior mesh entity to improve the element shape quality, it is possible that the current mesh configuration is such that there's not enough space for the entity to move in the desired direction of motion to effectively increase the shape quality of the element. In such a case, a collapse or swap operation is often effective to create extra space for the curving to be executed. Another typical scenario is when a target element for shape improvement has too strict geometric and/or topological constraints. In such cases, split operations are used to split the element into several new elements that each has less constraint for the mesh modification operations to carry out. Section 5.4 discusses the application of the full set of curved mesh modification operations

5 Parallel Adaptive Simulation Loop

The work flow for an adaptive simulation starts with a definition of the problem domain of interest. In computer-aided design and engineering, the domain definition is typically a solid model constructed in a CAD system [7]. The analysis attributes (loads, boundary conditions, material properties) are specified with respect to the solid model. Initial mesh control attributes that guide the mesh generation process can also be specified with respect to the solid model [2]. Based on the meshing attributes, an initial mesh is generated with the required geometric approximation accuracy to the model. In the case of parallel simulations, load balancing is performed to maintain balanced distribution of workloads among the multiple processes. The finite element analysis procedure computes the solution fields of interest. To adaptively improve solution accuracy, error estimation and correction indication procedures are used to calculate a mesh size field, which is used to drive the mesh adaptation procedure to obtain an adapted mesh. After mesh adaptation parallel dynamic load balancing is applied and the finite element analysis procedure is performed, and a new set of solution fields can be obtained with improved resolution and accuracy [1]. The adaptive simulation loop continues until the desired solution accuracy is achieved as shown in Fig 5. Finally the results of the solution fields can be post-processed and visualized. Details of the individual steps are discussed in Section 5.1 through Section 5.4.

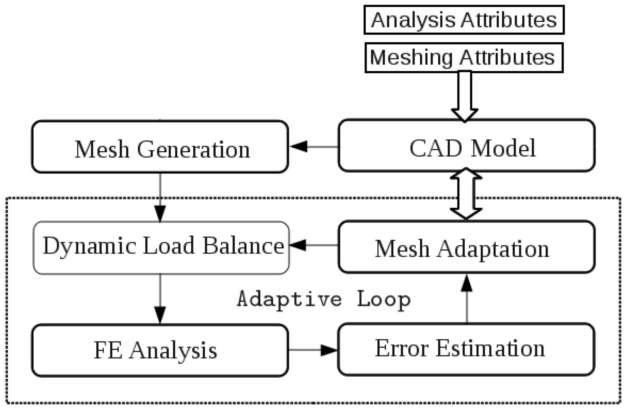


Fig. 5 The workflow for an adaptive simulation

5.1 Invalidity Correction for the Initial Curved Mesh

For problems with fixed domain geometry, the first stage to carry out before performing the first finite element analysis is to check the input mesh for element validity. The invalidity correction algorithm detects the invalid elements by using the validity check methods introduced in Section 3. It chooses the proper local mesh modification and element reshaping operations to correct the invalidity of the curved elements. Details of the specific invalidity correction algorithm are discussed in References [21,23].

5.2 Element Shape Quality Improvement

After obtaining a valid input mesh, this stage is carried out to check for unsatisfactory element shape quality, and if there is any, improve the shape quality by using proper local mesh modification operation such as edge/face swap, and/or curved element reshape operations [19,21,23]. Details of the curved local mesh modification operation is presented in Section 4.2. Curved element reshaping is discussed in Section 4.1.2. Given a pre-defined element quality threshold Q_{th}, the elements whose shape quality Q_{sc} is below Q_{th} are collected into a list and processed iteratively until either the list is empty or no further local mesh modification operations can be applied to improve the remaining elements in the list. Note that since the reshaping operation for curved elements is extremely demanding in terms of computation costs due to the explicit search procedure involved, it is used at the end of the process should the swap operations fail to improve the curved element shape quality. After obtaining a valid input mesh with properly curved elements with and satisfactory shape quality, the finite element analysis process can be performed.

5.3 Parallel SPR Based Error Estimation

After a mesh is analyzed by the finite element procedure, *a posteriori* error estimation is executed to calculate the information used to compute the new mesh sizes that are input to the mesh modification procedures. A parallel error estimation procedure has been developed based on the Super-convergent Patch Recovery (SPR) scheme [35,36]. In the error estimation procedure, a C^0 continuous gradient field is recovered from the original C^{-1} discontinuous field, and the error is defined as the difference between the recovered and original fields.

A key step of the recovery of a given nodal degree of freedom (associated with a mesh vertex, M_i^0) employs a local least-square fitting scheme over the patch of elements (mesh regions $\{M_i^0\{M^3\}\}$) surrounding the node (or the mesh vertex M_i^0). Therefore, the complete field information of the nodal patch is required. In the context of a parallel analysis based upon a distributed mesh, some of the mesh vertices are located on mesh partition boundaries. Consequently, the corresponding nodal patch of such a vertex is distributed among several mesh partitions, thus not complete within a local part. In order to form a complete patch on a local mesh part in a way that avoids extensive communication cost between mesh parts, the parallel SPR based error estimator takes advantage of the ghosting functionality supported by the Flexible distributed Mesh DataBase (FMDB) [31], which creates one extra layer of ghost mesh entities at the part boundary [26].

Data: Distributed Mesh M, Solution field data S
Result: Calculate the error field, elemental error indicator and new desirable mesh size
1 Load the mesh and solution data, setup the nodal field correctly;
2 Create one layer of ghost regions with bridge vertices;
3 Conduct the on-part SPR procedure on M and S;
4 Calculate the error field, elemental error indicator and new desirable mesh size using Equations 8, 10 and 9;
5 Delete the ghost entities;

Algorithm 2: Parallel SPR Based Error Estimation with Ghosting

Algorithm 2 gives the steps for the error estimation procedure using ghosting. Step 1 loads the mesh and attaches the finite element solution field data to the specific mesh entities. The field data associated with a corner node is attached to the corresponding mesh vertex, and for higher order elements, the field data associated with an edge and/or a face node is attached to the mesh edge/face. Step 2 creates one ghost layer of mesh regions on part boundaries as well as the mesh vertices, edges and faces that bound the ghost regions. The field data attached to the mesh entities is also carried along with the ghosts onto local processors so that after the ghosting process is done, it is guaranteed that each mesh vertex on any local part (including the ones on part boundary) has a complete local patch of elements on the same mesh part. And the complete local patch has all the field data information needed to carry out the SPR procedure. Therefore no extra inter-part communication is required to form the local nodal patch. In Step 3, each local mesh part with the ghosted mesh entities and field data is regarded as an independent local mesh with no inter-processor communication necessary. The SPR procedure recovers a C^0 continuous gradient field ε^* based on the least-square fitting of the discontinuous finite field at element integration points over the complete nodal patch [34]. Step 4 calculates the error field, elemental error indicator and desirable mesh size. Since the exact solution ϵ is unknown, the error e_ϵ can only be estimated. In the SPR based approach, the recovered solution ε^* is used to replace the exact solution. The approximated error field is computed under certain norm. For instance, L_2 norm is used to measure the error field in Equation 8.

$$\|e_\epsilon\|_{L_2} \approx \|e_\varepsilon^*\|_{L_2} = \left\|\varepsilon^* - \varepsilon^h\right\|_{L_2} \tag{8}$$

After the error field computation, the elemental error indication is carried out by integrating of the error over the element domain. The desirable mesh size is calculated through an h-adaptive procedure based on Equation 9 [34].

$$h_e^{new} = h_e^{current} \times r_e \tag{9}$$

where h_e^{new} and $h_e^{current}$ denote the new and current mesh sizes respectively. And the size scaling factor r_e is computed based on the Equation 10 [34].

$$r_e = \|e_\varepsilon\|_e^{\frac{2}{2p+d}} \frac{\eta^2 \|\varepsilon^*\|^2}{\sum_{i=1}^{n} \|e_\varepsilon\|_i^{\frac{2d}{2p+d}}}^{\frac{1}{2p}} \tag{10}$$

where d and p are respectively the dimension and polynomial order of elements in the part mesh. e is an element in the mesh. The goal is to ensure the relative percentage error in L_2 norm of the error is below a given limit.

With the aid of the ghost layers, there is no inter-part communication required in Step 4. After the error estimation procedure is completed, Step 5 deletes the ghost entities that were created as part of Step 2. With the application of ghosting, the overall communication cost of the parallel error estimator can be reduced to the one-time ghosting process in Step 2 as there is no other form of inter-part communication required.

5.4 Parallel Curved Mesh Adaptation with Mesh Size Field

Given the current curved mesh and a desired mesh size field (for instance computed by an error estimator such as the SPR based procedure discussed in Section 5.3 the curved mesh adaptation procedure is carried out to adapt the mesh using a set of parallel mesh modification operations and element reshaping operations. The parallel adaptation produces a distributed curved mesh that satisfies the size field and preserves the geometric approximation to the right order.

5.4.1 Parallel Mesh Modification Operations

The set of local mesh modification operations for curved meshes discussed in Section 4.2 is executed in parallel. The primary consideration in parallelizing those operations is to properly determine appropriate geometric shapes for the new mesh entities created during those operations among multiple mesh partitions. Technical modifications are made to the parallel straight-sided mesh modification operations [1,4,18].

In the case of splitting a curved mesh entity, any newly created mesh entities on the curved geometric model boundaries must be properly curved to the model boundary to ensure the geometric approximation of the resulting mesh is maintained [23]. In some cases, the curving process changes the original mesh geometry as shown in Fig 6, and it could possibly invalidate the neighboring elements as shown in Fig 7. When splitting a curved entity which is shared by multiple mesh parts in parallel, the affected elements are on different mesh parts as well. Therefore it is possible that curving the new entities dictated by the decision of the owner copy would cause the elements on the remote copy to become invalid. For instance, M_0^2 and M_1^2 in Fig 7 happens to be on two separate mesh parts P_0 and P_1 while the curved edge to be split is on part boundary, the splitting and curving decision made by M_0^2 on P_0 would invalidate the new element M_4^2 on the P_1. In order to effectively avoid such cases in parallel, the parallel curved split operation requires that, if the target curved entity to be split is shared by multiple mesh parts, all the affected elements (which are the mesh regions that the target entity bounds on all mesh parts) must be migrated to a single mesh part. The parallel split operation is described in Algorithm 3:

If any of the entities to be eliminated by a collapse or swap operation is on a mesh partition boundary, the local mesh cavity is distributed on more than one mesh part. In this case, the entities of the cavity will be migrated to a single mesh part first as in the straight edge case [1,4]. In the curved mesh entity case the migration process must also migrate the curved mesh entity shape information. The idea of extending the edge swap operation to deal with parallel curved meshes is essentially the same as for the edge collapse operator in the sense that (i) geometric approximation accuracy needs to be considered when curving newly created entities, and (ii) curved entity migrations will be needed for the cases that the cavity is across partition boundaries. The process for parallel curved edge collapse or swap operations is summarized as in Algorithm 4:

Fig 8 demonstrates a 2D example of a combination of parallel curved edge swap operation. Note that the collapse/swap operation is in essence local, the full cav-

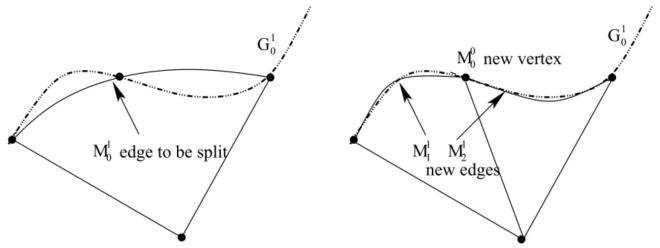


Fig. 6 Before (left) and after (right) splitting a quadratic curved mesh edge on model edge. New mesh edges have been properly curved [23]

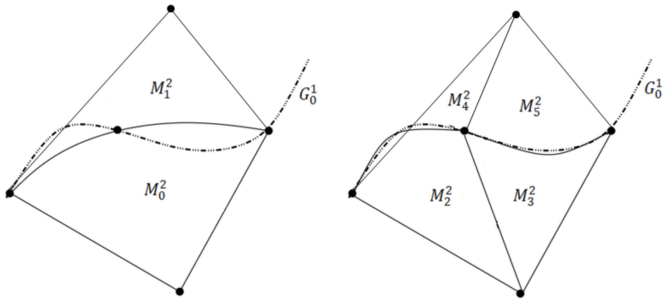


Fig. 7 Curving new mesh edges after splitting could cause invalid element as M_4^2

Data: A quadratic curved edge shared by two mesh parts
Result: Two quadratic edges after split
1 Determine the mesh regions that the target edge bounds on each mesh part;
2 Migrate all the regions to a single mesh part to form a complete cavity;
3 Split the curved edge into two new edges as is done in a serial split operation;
4 Curve the new edges to the proper location;
5 Check if the curving process leads to invalid element, if so, fix the invalidity by local mesh modification and/or element reshaping operation;

Algorithm 3: Parallel split operation for a curved edge on partition boundary

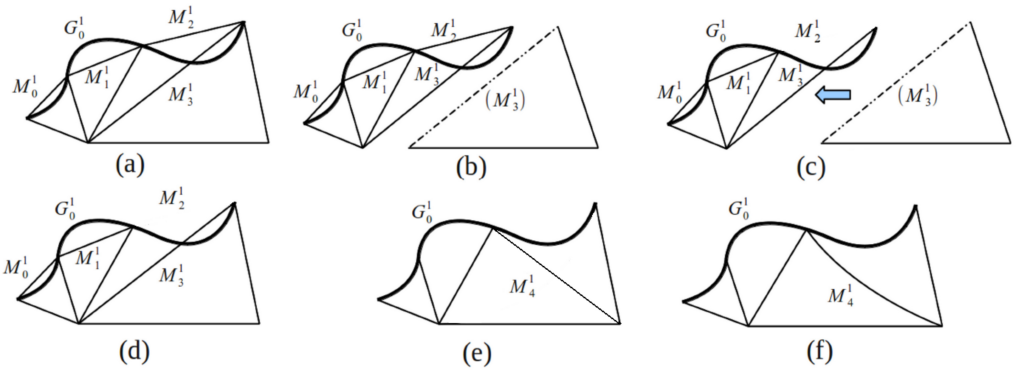


Fig. 8 A 2D example of parallel curved edge swap operation (a) an initial straight-sided 2D mesh, (b) the mesh is distributed to two parts, (c) boundary mesh edges M_0^1, M_1^1, M_2^1 are curved to the model edge G_0^1, interior straight edge M_3^1 is causing element invalidity by intersecting with M_2^1 (d) to fix the invalidity, M_3^1 is to be swapped, so the distributed mesh entities are migrated to form a local cavity on a single part, (e) M_3^1 is swapped and the new edge M_4^1 is created to be straight-sided (f) to improve element shape quality, the interior edge M_4^1 is curved

Data: Curved edge to be collapsed or swapped
Result: New mesh cavity after collapse or swap operation
1 Determine if the operator is performable by topology and geometry checks;
2 If the local mesh cavity is across part boundary, migrate the entities to a single part by curved mesh migration operations;
3 Perform topological modifications: collapse/swap the target entity and modify the connectivity of the cavity;
4 Perform geometric modifications if the original cavity has high-order curved entities. Properly curve the newly created entities based on the geometric model information or the mesh geometry;

Algorithm 4: Parallel curved collapse/swap operation

ity of all the to-be-affected mesh regions is known before applying the operation. There is no need to expand the cavity during a single collapse/swap operation.

5.4.2 Parallel Curved Mesh Adaptation

The parallel curved mesh adaptation procedure consists of three steps (see Algorithm 5): 1) coarsening, 2) iterative refinement, and 3) shape quality improvement.

The coarsening process eliminates the mesh edges that are shorter than the desired length specified by the size field [19]. It is accomplished by performing entity collapse operations on the identified short edges. Serial (on-part) curved entity collapse operation are introduced in [23]. Its parallelization is discussed in Section 5.4.1. When collapsing curved mesh entities classified on curved geometric model boundary, it is necessary to curve the new entities to conform to the boundary after collapsing. Curving the new entity may lead to intersection with another existing mesh entity which

Data: Initial curved mesh, geometry domain, and mesh metric field
Result: Adapted curved mesh satisfying the metric field
1 traverse mesh edges and determine the edges with length shorter than L_{low};
2 perform coarsening algorithm to those edges;
3 traverse mesh edges and determine the edges with length longer than L_{up};
4 perform iterative refinement algorithm to those edges;
5 traverse mesh regions and create a list of regions that have shape quality $Q_{sc} < Q_{threshold}$;
6 **while** *there is still unprocessed region(s) in the list* **do**
7 evaluate the best local mesh modification operations applicable to improve the shape of the region;
8 **if** *no operation is applicable* **then**
9 tag the region as processed;
10 **end**
11 **end**
12 create a list of of the regions still with quality $Q_{sc} < Q_{threshold}$;
13 perform entity shape improvement by mesh modifications and element reshaping;

Algorithm 5: Overall algorithm of curved mesh adaptation

causes element invalidity. To identify such cases, curved mesh validity checks (refer to Section 3) are always applied after a collapse operation. Invalidity is then resolved by the invalidity correction algorithm [23,21] as discussed in the previous stage of this section.

After coarsening, the iterative refinement algorithm incrementally reduces the edge lengths that are longer than the desired size field by as many iterations as needed to satisfy the local mesh metric. Entity split operations are applied to achieve the goal [19,23]. In the case that curved entities are to be refined, the new

entities should be curved properly as well. The shape of the new entities are determined through parametric interrogations to the CAD modeler [6] and/or by calculation in the parametric space using Bézier parametrization of curved tetrahedrons [10,23,21]. Edge length is checked after each refinement iteration. Edge collapse operations are performed to eliminate the shorter-than-desired edges introduced by the refinement operations.

After iterative refinement, element shape quality improvement is carried out to control the overall quality of the adapted mesh. It uses the proper combination of the local mesh modification operations, primarily the swap operation, and element reshaping operation as discussed in Section 5.2.

The adapted mesh is analyzed again by the finite element solution procedure to obtain more accurate solution field data. If the solution still does not reach the desired accuracy, the adaptive loop continues with another iteration of error estimation as discussed in Section 5.3, mesh adaptation as discussed in Section 5.4, and finite element solution procedure.

6 Parallel Adaptive Mesh Examples

The Advanced Computations Department (ACD) of the SLAC National Accelerator Laboratory (SLAC) is developing a suite of high-order finite element procedures (ACE3P) for accelerator simulations that have demonstrated the ability to accurately model a variety of accelerator problems [23,24]. The level of discretization to obtain reliable predictions in ACE3P simulations often requires meshes with upwards of hundreds of millions of elements. To meet the requirements, the Scientific Computation Research Center (SCOREC) at RPI, in collaboration with Simmetrix Inc., is working with SLAC on providing the full range of parallel curved mesh generation and adaptation tools needed to work with the ACE3P simulation tools.

Fig 9 gives an example of a distributed curved mesh generated over a geometry of two linear accelerator cavities. The largest application so far has been a partitioned curved mesh of 180 million tetrahedral elements generated on 64 processors in less than 12 minutes (not counting I/O).

Fig 10 shows a small example of a parallel curved mesh refinement process with an isotropic analytic size field. The geometry is a linear accelerator cavity. The mesh to be refined on the left is of a relatively coarse global mesh size with finer mesh being generated locally at regions of large curvature. The parallel refinement focuses at the coarse mesh regions and brings the global mesh to a finer size while keeping the locally refined mesh regions unchanged.

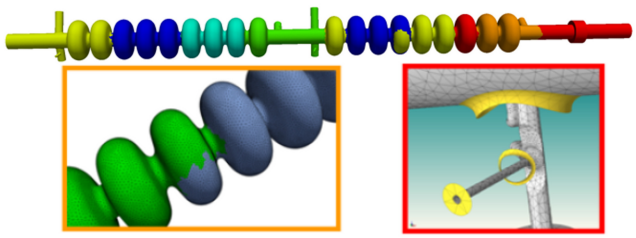


Fig. 9 Overview and close-ups of a partitioned curved mesh of linear accelerator cavities

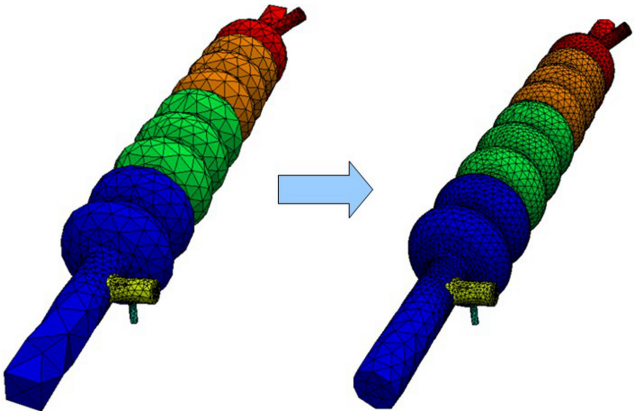


Fig. 10 An example of parallel curved mesh refinement on a four part mesh

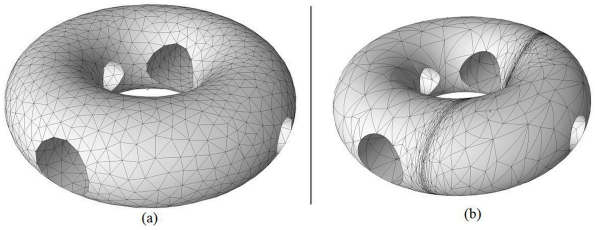


Fig. 11 An example of curved mesh adaptation with an anisotropic size field

Fig 11 gives an example of adapting an isotropic initial curved mesh to an anisotropic analytic size field representing a planar shock.

Fig 12 gives a set of pictures: (a) initial mesh of 8 parts, (b) solution field, (c) new size field and (d) 8-part adapted mesh of a pillbox model. Figs 12(e) and (f) are close-up views of the region where relatively small mesh sizes are needed to get higher resolution and extensive curved mesh refinement is applied. The initial mesh in this case has 22k elements and the mesh after adaptation has 106k elements. The wall clock time of mesh adaptation for this 8-part mesh in parallel is 6.73 seconds. The total time for the same adaptation process in serial is 22.89 seconds. Both the serial and parallel cases run on 2.3GHz Opteron processors.

Fig 13 shows an initial unadapted mesh with 32 mesh parts of the linear electron-positron collider model

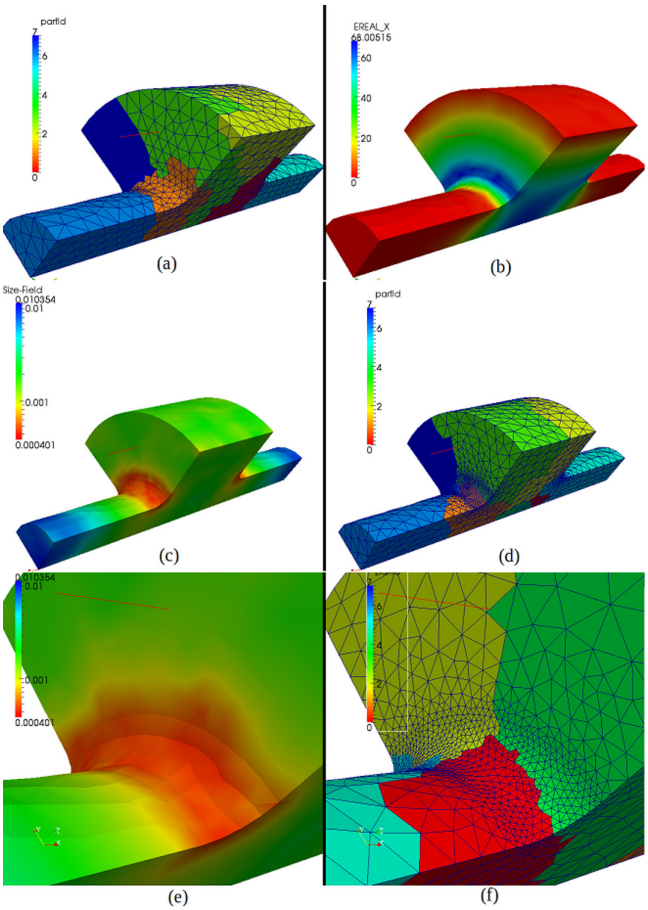


Fig. 12 Example of one iteration of the parallel adaptive loop

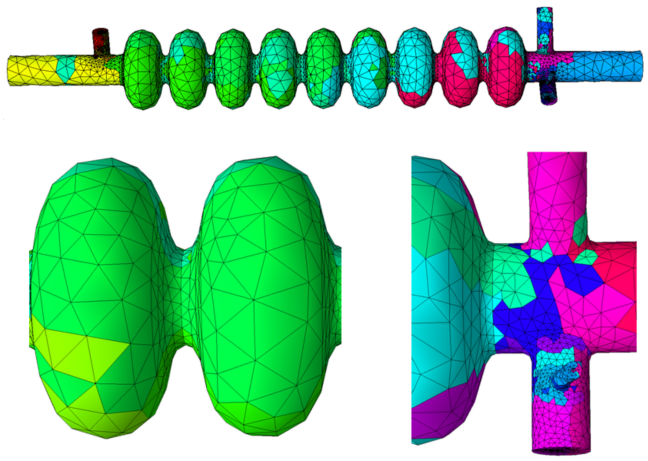


Fig. 13 32-part initial mesh of the Tesla accelerator cavity model with 178k elements

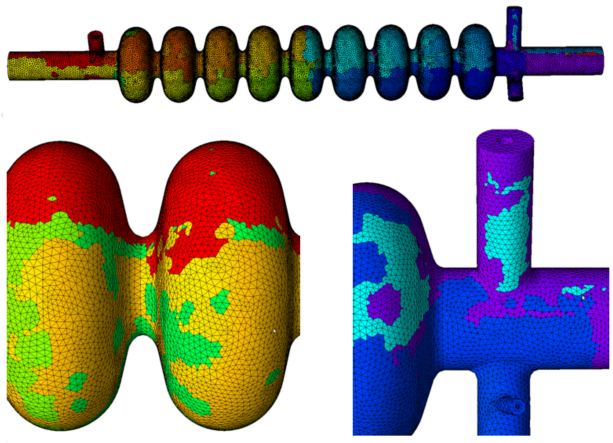


Fig. 14 Adapted 32-part mesh with 1.65 million elements

TESLA, which is based on 9-cell superconducting niobium cavities. The initial unadapted mesh has about 178k elements in total. The high-order finite element solver package Omega3P [16] is used to compute the nominal dipole properties of the cavities. The important requirements for computing the accelerating mode frequency is that it has to be known to within 0.01% of the designed value [27], and that the wall loss to be calculated as accurately as possible for efficiency and thermal management reasons [9].

The initial mesh of 178k elements with 32 partitions is used to obtain a set of preliminary solution fields. Based on the solution field data, the parallel ZZ-SPR based error estimator and curved mesh adaptation procedures are performed. Fig 14 shows an adapted 32-part mesh with 1.65 million elements. The adapted mesh is used as input for Omega3P to re-run the finite element computation and get more accurate solution fields. The adaptive loop iterations continue until the eigenvalue solution converges within the 0.01% tolerance. Fig 15 shows the computed solution of electric field of Omega3P analysis on an adapted mesh with 5.14 million elements. Table 4 shows the series of eigenvalue solutions from the adaptive loop iterations that start from less than 200k degrees of freedom (DOFs) to more than 33 million DOFs. As a comparison with the error-based adaptation loop, another series of uniformly refined meshes are studied for the same problem. Solutions are obtained and listed in Table 5.

Fig 16 shows the convergence plot of the eigenvalue solution data obtained by error-based adaptation loops (AMR) as given in Table 4 and uniform refinement meshes (UMR) as given in Table 5 respectively. One can clearly observe that the solution converges much faster with many fewer DOFs using the error-based mesh adaptation procedure. For example the adapted

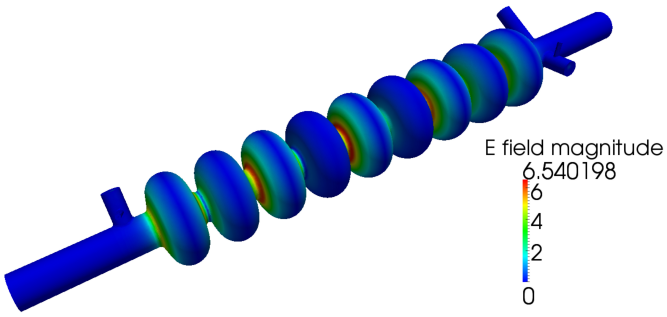


Fig. 15 Solution field on the 32-part mesh

Table 4 Solutions obtained by error based mesh adaptation

No. Elems	No. DOFs	Eigenvalue (e+09)	Rel Error
178k	191k	1.29377	n/a
429k	468k	1.29214	0.126%
755k	834k	1.29146	0.053%
1.65m	1.84m	1.29108	0.029%
5.14m	5.82m	1.29088	0.015%
29.1m	33.4m	1.29084	0.003%

Table 5 Solutions obtained by uniformly refined meshes

No. of elements	No. of DOFs	Eigenvalue (e+09)
178k	191k	1.29377
1.42m	1.69m	1.29142
11.4m	13.4m	1.29090

Table 6 Cost of the adaptive loop steps. Unit:(seconds)

No. Elems	No. Procs	Analysis Time	Adaptation Time
178k	32	31	10
429k	32	133	27
755k	32	367	65
1.65m	32	839	163
5.14m	32	3155	252
29.1m	256	4017	n/a

mesh of 1.65m elements gives the same relative error as a uniformly refined mesh of 11.4m elements.

The total wall clock time for analyzing and adapting the meshes from 178k elements to 29.1m elements is given in Table 6. The analysis and adaptation steps are carried out on the Cray XE6 supercomputer Hopper at the National Energy Research Scientific Computing Center (NERSC). Note that the final 29.1-million-element mesh is obtained by repartitioning the 32-part 5.14m mesh to 256 parts and adapting the 256-part mesh. The 252 seconds given in Table 6 is the time for the actual adaptation process with 256 processors.

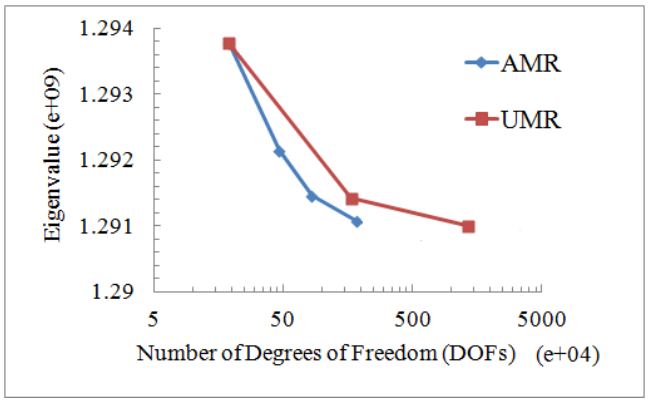


Fig. 16 Convergence of the solution under error-based adaptation and uniform refinement

7 Closing Remarks

A curved mesh adaptation procedure designed to operate on massively parallel computers was presented. The core of the procedure are two classes of mesh modification: entity geometry modification and local mesh modification for curved meshes. For the entity geometry modification, curved entity reshape operations that explicitly resolve element invalidity and improve the shape quality of curved elements are presented. The local mesh modification operations for curved meshes were extended from the operations for straight-sided meshes with additional consideration and treatment of curve boundary entities and selected curved interior entities. The parallel curved mesh adaptation technique is being used to support the automated adaptive accelerator simulations at SLAC National Accelerator Laboratory.

Acknowledgements This work is supported by the US Department of Energy. The RPI portions of the work are supported by the SciDAC grant NO. DE-FC02-06ER25769 and a DOE SBIR grant NO. BEE101/DE-SC0002089. The Simmetrix portions of the work are supported by the SBIR grant. The authors would like to thank Dr Lixin Ge, Dr Cho-Kuen Ng and Dr Kwok Ko at SLAC National Accelerator Laboratory for providing the accelerator models and access to the ACE3P solvers.

References

1. Alauzet, F., Li, X., Seol, E.S., Shephard, M.S.: Parallel anisotropic 3d mesh adaptation by mesh modification. Engineering with Computers **21**, 247–258 (2006)
2. Beall, M.W., Shephard, M.S.: An object-oriented framework for reliable numerical simulations. Engineering with Computers **15**(1), 61–72 (1999)
3. Ciarlet, P.G.: The Finite Element Method for Elliptic Problems. SIAM (2002)
4. de Cougny, H.L., Shephard., M.S.: Parallel refinement and coarsening of tetrahedral meshes. International Jour-

nal for Numerical Methods in Engineering. **46**(7), 1101–1125 (1999)
5. de Cougny, H.L., Shephard, M.S., Georges., M.K.: Explicit node point mesh smoothing within the octree mesh generator. Tech. rep., Scientfic Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY (1990)
6. Dey, S., O'Bara, R.M., Shephard., M.S.: Curvilinear mesh generation in 3d. Computer-Aided Design. **33**, 199–209 (2001)
7. Farin, G.E.: Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide, Third Edition. Academic Press. Waltham, MA. (1992.)
8. Freitag, L.A., Knupp., P.M.: Tetrahedral element shape optimization via the jacobian determinant and condition number. In: Proceeding of the 8th International Meshing Roundtable. South Lake Tahoe, CA., pp. 247–258 (1999)
9. Ge, L., Lee, L.Q., Li, Z., Ng, C., Ko, K., Luo, Y., Shephard, M.S.: Adaptive mesh refinement for high accuracy wall loss determination in accelerating cavity design. Tech. rep., SLAC National Accelerator Laboratory. Menlo Park, CA. (2004)
10. George, P.L., Borouchaki, H.: Construction of tetrahedral meshes of degree two. International Journal for Numerical Methods in Engineering **90**, 1156–1182 (2012)
11. Johnen, A., Remacle, J.F., Geuzaine, C.: Geometrical validity of curvilinear finite elements. In: Proceedings of the 20th International Meshing Roundtable. Paris, France. (2011)
12. Knupp, P.M.: Algebraic mesh quality metrics. SIAM Journal on Scientific Computing. **23**(1), 193–218 (2001)
13. Knupp, P.M.: Algebraic mesh quality metrics for unstructured initial meshes. Finite Elements in Analysis and Design. **39**, 217–241 (2003)
14. Knupp, P.M.: Remarks on mesh quality. In: 45th AIAA Aerospace Sciences Meeting and Exhibit. Reno, NV (2007)
15. Knupp, P.M.: Introducing the target-matrix paradigm for mesh optimization via node-movement. In: Proceedings of the 19th International Meshing Roundtable. Chattanooga, TN., pp. 67–84. Chattanooga, TN. (2010)
16. Lee, L.Q., Li, Z., Ng, C., Ko, K.: Omega3p: A parallel finite-element eigenmode analysis code for accelerator cavities. slac-pub-13529. Tech. rep., SLAC National Accelerator Laboratory. Menlo Park, CA., Menlo Park, CA. (2009)
17. Li, X.: Mesh modification procedures for general 3d non-manifold domains. Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY. (2003)
18. Li, X., Shephard, M.S., Beall, M.W.: Accounting for curved domains in mesh adaptation. International Journal for Numerical Methods in Engineering **58**(2), 247–276 (2003)
19. Li, X., Shephard, M.S., Beall, M.W.: 3d anisotropic mesh adaptation by mesh modification. Computer Methods in Applied Mechanics and Engineering **194**, 4915–4950 (2005)
20. Liu, A., Joe, B.: Relationship between tetrahedron shape measures. BIT Numerical Mathematics **34**(2), 268–287 (1994)
21. Lu, Q.: Developments of parallel curved meshing for high-order finite element simulations. Master's thesis, Rensselaer Polytechnic Institute., Troy, NY (2011)
22. Luo, X.: An automatic adaptive directional variable p-version method in 3d curved domains. Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY., Troy, NY. (2005)
23. Luo, X., Shephard, M.S., Lee, L.Q., Ge, L., Ng, C.: Moving curved mesh adaptation for higher-order finite element simulations. Engineering with Computers **27**(1), 41–50 (2010)
24. Luo, X., Shephard, M.S., Yin, L.Z., O'Bara, R.M., Nastasi, R., Beall, M.W.: Construction of near optimal meshes for 3d curved domains with thin sections and singularities for p-version method. Engineering with Computers **22**(1), 41–50 (2010)
25. Morin, G., Goldman, R.: On the smooth convergence of subdivision and degree elevation for bezier curves. Computer Aided Geometric Design. **18**, 657–666 (2001)
26. Mubarak, M., Seol, S., Lu, Q., Shephard, M.S.: A parallel ghosting algorithm for the flexible distributed mesh database. Submitted to Scientific Programming (2012)
27. Ng, C., Akcelik, V., Candel, A., Chen, S., Folwell, N., Ge, L., Guetz, A., Jiang, H., Kabel, A., Lee, L.Q., Li, Z., Prudencio, E., Schussman, G., Uplenchwar, R., Xiao, L., Ko, K.: State of the art in em field computation. slac-pub-12020. Tech. rep., SLAC National Accelerator Laboratory. Menlo Park, CA., Menlo Park, CA. (2006)
28. Persson, P.O., Peraire, J.: Curved mesh generation and mesh refinement using lagrangian solid mechanics. In: Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit. (2009)
29. Prautzsch, H., Kobbelt, L.: Convergence of subdivision and degree elevation. Advances in Computational Mathematics **2**, 143–154 (1994)
30. Sederberg, T.W.: Computer Aided Geometric Design (2011). Http://tom.cs.byu.edu/ 557/text/cagd.pdf; accessed May 31, 2012
31. Seol, E.S., Shephard, M.S.: Efficient distributed mesh data structure for parallel automated adaptive analysis. Engineering with Computers. **22**(3), 197–213 (2006)
32. Shewchuk, J.: What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (2002). Preprint
33. Szabo, B.A., Babuska, I.: Finite Element Analysis. John Wiley & Sons Inc, New York, NY (1991)
34. Wan, J.: An automatic adaptive procedure for 3d metal forming simulations. Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY. (2006)
35. Zienkiewicz, O.C., Zhu, J.Z.: The superconvergent patch recovery and a posteriori error estimates. part 1. the recovery technique. International Journal for Numerical Methods in Engineering. **33**, 1331–1361 (1992)
36. Zienkiewicz, O.C., Zhu, J.Z.: The superconvergent patch recovery and a posteriori error estimates. part 2. error estimates and adaptivity. International Journal for Numerical Methods in Engineering. **33**, 1365–1382 (1992)