Parallel Curved Mesh Adaptation for Large Scale High-order Finite Element Simulations

Qiukai Lu[1], Mark S. Shephard[1], Saurabh Tendulkar[2] and Mark W. Beall[2]

[1] Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY. luq3@rpi.edu, shephard@rpi.edu.
[2] Simmetrix Inc. saurabh@simmetrix.com, mbeall@simmetrix.com.

Summary. This paper presents the development of 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. Applications of the curved mesh adaptation procedure have been developed to support the parallel automated adaptive accelerator simulations at SLAC National Accelerator Laboratory.

1 Introduction

High-order finite element methods have the advantage of achieving exponential rates of convergence to problems of interest [21]. In order to fully realize the benefits of the methods over 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 resolution in certain critical regions, meshes with substantially large amount of entities are needed. Such meshes can only be created using distributed parallel computing systems, which requires effective parallel mesh generation and adaptation techniques [2].

There have been extensive studies in the area of automatic mesh generation and adaptation techniques. The majority of the efforts have been focused on dealing with conventional meshes with all straight-sided elements. To support large scale simulations, parallel mesh generation and adaptation on distributed computers has been under development, and the primary focus has been on the linear straight-sided meshes as well. To date there has been limited effort devoted to studying meshing techniques for fully unstructured curved meshes [4, 7, 8, 18], not to mention curved meshes in parallel. 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 the parallel curved mesh adaptation procedures from the aspects of specific local mesh modification operations and general curved mesh adaptation strategies. Section 3 introduces a prototype adaptive simulation loop which utilizes the developed parallel curved mesh adaptation technique. A couple of application examples are presented to demonstrate the effectiveness of the adaptive loop. Section 4 gives the closing remarks and points out the potential future developments.

2 Mesh Modification of Curved Mesh Entities

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

2.1 A General Entity Shape Measure

Given a straight-sided tetrahedral element, its geometric shape is uniquely defined by the positions of its four vertices. It is straight-forward to calculate various geometric quantities to evaluate the shape of the element, such as edge length, solid angle. Plenty of shape measures for straight-sided elements have been proposed that are based upon such geometric quantities. Several commonly used measures are reviewed in the following papers [9, 12]. There are relatively less geometric shape measures proposed 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 popular 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 [4, 8, 14, 18]. 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 specific tetrahedron is in the physical space with respect to the reference tetrahedron in the parametric space. 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 or close to being degenerated. Therefore, q_c can not serve as a general shape measure for a mesh that has both straight-sided and curved elements.

In order to find a general shape measure that overcomes the aforementioned issue, a multiplicative element shape measure is defined which 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 [9, 12] 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 will be computed and contributes equally to Q_{sc} with the underlying 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 which will be presented in the later sections.

2.2 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 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 itself. According to the convex hull property [5] 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 [14] 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 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.

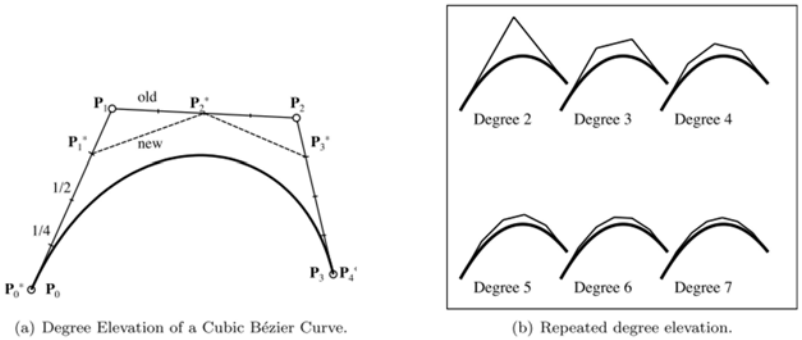
Although the uniform validity check is sometimes overly-conservative, it is nevertheless an effective method to determine the candidate mesh entities with which the potential invalidity is associated. Depending on which control point exhibits the negative lower bound, a specific set of mesh edges corresponding to that control point are identified. 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 $M^2\{M^1\}$ [14]. Once the candidate mesh entities are identified, one of the following approximation improvement methods is applied to obtain tighter bounds for $det(J)$.

Adaptive Validity Check By Degree Elevation

The degree elevation algorithm 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 [5]. It can be applied to the key entity 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 [5]:

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

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



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

Fig. 1. Convergence of Degree Elevation [20]

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 [16, 19], 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 [20].

Adaptive Validity Check By Subdivision

In addition to the degree elevation algorithm, polynomial subdivision algorithm also yields more control points and tighter control polygon for a Bézier polynomial. Thus it can also be applied to the key entity to improve the lower bound [13]. Take an edge as the 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 [5] as follows:

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

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 has discussions regarding this topic [7].

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 [16, 19],

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 [20].

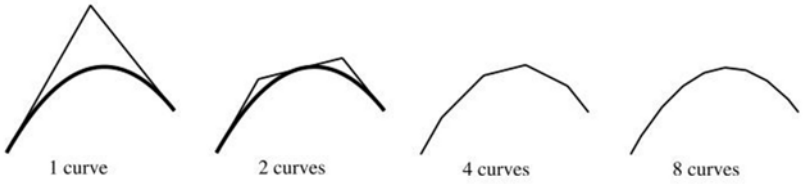


Fig. 2. Convergence of polynomial subdivision [20]

Stopping Criteria for the Adaptive Validity Checks

After applying either of the adaptive validity check algorithms, if $\min\{P_{|i|}^{(n)}\}$ is found to be positive, the element is valid. On the other hand, the element is invalid if negative $\min\{P_{|i|}^{(n)}\}$ 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 $\min\{P_{|i|}^{(n)}\}$ appears at a non-interpolating control point while positive values are found at all interpolating control points during finite steps, it is also necessary to terminate the algorithm without doing infinite loops of checking and refinement. The stopping criterion used is to evaluate the increment of the lower bound $\Delta \min\{P_{|i|}^{(n)}\}$ after each step. If negative $\min\{P_{|i|}^{(n)}\}$ is still reported after $\Delta \min\{P_{|i|}^{(n)}\} < \epsilon$, where ϵ is a prescribed tolerance, then the element is regarded as invalid and the algorithm stops at the current step.

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.

2.3 Curved Entity Geometry Modification Operations

A set of entity level geometry modification operations have been designed and implemented to work with curved mesh elements. These operations are different from the local mesh modification operations for straight-sided elements introduced in [11] in that they involve purely geometric shape modifications

while keeping the local mesh connectivity unchanged. An entity geometry modification operation is often used to increase geometric approximation accuracy to the curved model domain, eliminate mesh invalidity, or improve mesh shape quality.

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 to the curved geometry to ensure geometric approximation accuracy. Dey et al presented a mesh curving algorithm in [4], 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.

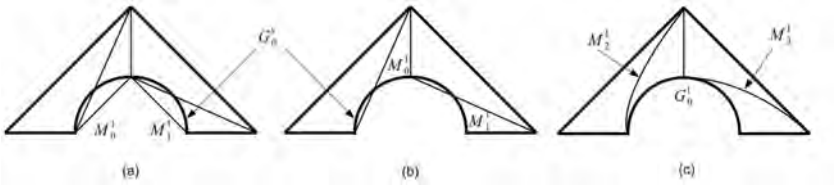


Fig. 3. Curving interior mesh edges to resolve the invalidity caused by curving boundary edges

Only curving the boundary mesh entities may cause self-intersecting mesh entities which leads to invalid elements. In such occasions, interior mesh entities are selected to be curved to correct the invalidity. For example in a 2D case shown in Fig 3, mesh edges M_0^1 and M_1^1 are curved to model edge G_0^1. However elements M_0^2 and M_1^2 become invalid (Fig 3(b)). In this case, the interior edges M_2^1 and M_3^1 can be curved (Fig 3(c)) to ensure element validity. The procedures for such element curving are referred to as entity reshaping operations, which are discussed in the following section.

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 [3, 6, 9, 11]. 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. Therefore the developed curved element reshaping algorithm considers curved shape parameters.

Input to this algorithm is a curved tetrahedral element whose shape quality is evaluated by shape measure in Eq 2. The algorithm computes the shape measures q_s and q_c respectively. The entity reshaping operation works independently with the specific selection of shape measure for evaluating q_s and q_c as long as the selected shape measure is normalized to range from 0 to 1 with 0 being the limit of an invalid element and 1 being an optimally shaped element.

The algorithm compares q_s and q_c with an application-specified shape quality threshold q_{limit} (0.1 used in the examples presented in this paper) and 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 [3, 11] for details. It will be removed from the current list and put into another list of tetrahedrons for a straight-sided element shape improvement procedure.

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 considerations are 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 Eq. 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 minimum value of $det(J)$. The selection criterion is the same as used in the curved validity check as discussed in Section 2.2.

Step 2: Determine the line of motion. As discussed in Section 2.1, the optimal value of q_c is reached when a given tetrahedron is straight-sided, i.e. $det(J)$ is constant over the element domain. Thus, it is a useful choice to reshape the curved entities back to be straight-sided assuming they can all move toward straight-sided (Fig 4(a)). On the other hand, certain curved entities may have specific geometric constraints that prevent them from being straightened, e.g. curved edges and/or faces classified on curved geometric model boundaries (M_1^1 in Fig 4(b)). In such cases, curving other less constrained entities along in a direction that improves q_c must be applied. The current algorithm limits the direction to move a high-order node of a curved edge to be along a straight line, referred to as Line of Motion (LOM). Fig 4(b) gives such an example in which M_1^1 is classified on model boundary, therefore it is curved to conform to

the boundary geometry. M_0^1 and M_2^1 are interior edges. If M_0^1 is straightened in this case, the element becomes invalid due to self-intersection. Therefore it has to be curved to resolve the invalidity and improve q_c. This is achieved by moving the edge high-order node along the LOM defined by P and P', which are the positions of the high-order nodes associated with M_0^1 and its straight-edged counterpart. The optimal location P" where M_0^1 should be curved is determined by the next step.

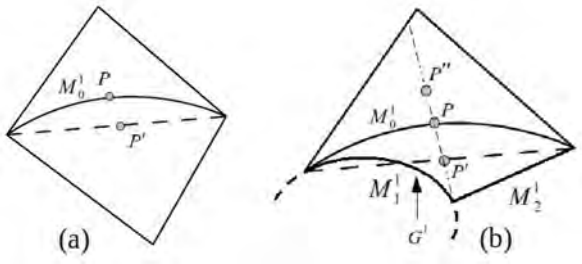


Fig. 4. Two cases of 2D mesh edge reshaping. (a) without further geometric constraints, (b) one additional edge classified on model boundary G^1

Step 3: Search the interval of uncertainty for the local optimal location. As reshaping the curved entity, the shape quality of the neighboring elements changes and may drop as well. When the reshaped entity reaches a position that intersects with another mesh entity, invalidity occurs. To avoid invalidity, the non-intersecting segment of the line of motion is considered as the search interval, referred to as interval of uncertainty, for the local optimal position. For instance, the dash-dot line in Fig 4(b) is the interval of uncertainty for **P** to move. In the meantime, to avoid deteriorating the overall shape quality of the affected elements while reshaping the target entity, the objective for the local optimal search is set to maximize the minimum shape quality among all the affected elements. Therefore the local optimal location for the entity to be reshaped is where the overall shape quality of the elements in the local cavity is improved to the highest possible value. A golden section algorithm is adopted here to perform the search [3].

Case 3: If $q_s < q_{limit}$ *and* $q_c < q_{limit}$

Of course there is the possible case of both q_s and q_c are not acceptable. In such a case, the algorithm chooses to improve the straight-sided shape quality first as in Case 1. Such a preference is set by considering of the cost-effectiveness of the algorithm since the explicit search algorithm required in Step 3 of Case 2 is often very computational intensive [3].

3 Parallel Adaptive Simulation Loop

The desired workflow 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. The various analysis attributes (loads, boundary conditions, material properties) are best 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. Based on the attributes, an initial mesh can be generated with desired 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 thus ensuring the efficiency of the workflow. The finite element analysis procedure computes the solution fields of interest. To adaptively improve solution accuracy, error estimation/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 dynamically balancing the work loads in parallel, the finite element analysis procedures can be performed again with the adapted mesh and a new set of solution fields can be obtained with improved resolution and accuracy. The adaptive simulation loop continues until the desired solution accuracy is achieved. Finally the results of the solution fields can be post-processed and visualized.

3.1 Modifications to Straight-sided Parallel Mesh Operations

This section discusses the parallelization of a set of local mesh modification operations for distributed curved meshes such as split, collapse and swap operations. The primary complexities introduced by curved mesh geometry in parallel is to properly determine appropriate geometric shapes for the new mesh entities created during those operations among multiple mesh parts. Technical modifications are made to the parallel straight-sided mesh modification operations [1, 2, 10].

Splitting curved mesh entities in parallel

In the case of splitting a curved mesh entity, the resultant entities created by subdividing the target entity have to be curved as well. When an entity to be split is at the partition boundary of multiple mesh parts, only the owner of the target entity can carry out the split operations and must inform copies of the curved mesh entity shape information of affected entities.

Take the example of a quadratic curved edge shared by two mesh parts P_0 and P_1. If a split operation is to be performed by inserting a new vertex at the mid-point location of the edge in the parametric space, the process can be summarized in the following steps:

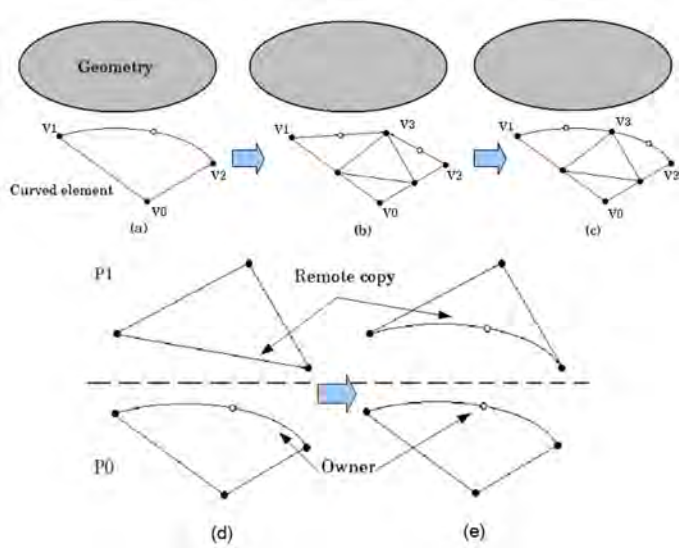


Fig. 5. Curved edge split operation and Synchronization of curved entities on partition boundary: (a) curved element to be split, (b) split the element uniformly and treat the newly created edges as straight-sided, (c) curve the new edges which are classified on curved geometric model boundary, (d) before sync, owner determines the target location of edge middle point. (e) after sync, remote copies receive data from owner and update their local copy

1. Calculate the x,y,z coordinates of the edge middle point for the new vertex to be introduced. See Fig 5(a).
2. Create a new vertex at the target location, two new edges connecting the new vertex and the other two existing vertices, and other new sub-faces. This is done for the owner and all the remote copies. See Fig 5(b).
3. For the owner copy, attach high-order nodes to the new edges and place the nodes properly based on the geometric model or the mesh geometry. See Fig 5(c).
4. Synchronize the remote copies with the owner across the parts to update the high-order node information. See Fig 5(d) and (e).

Collapsing or Swapping curved mesh entities in parallel

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 order to avoid communication overheads, the entities of the cavity will be migrated to a single mesh part first as in the straight edge case [1, 2]. 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 can be summarized as the following steps:

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.

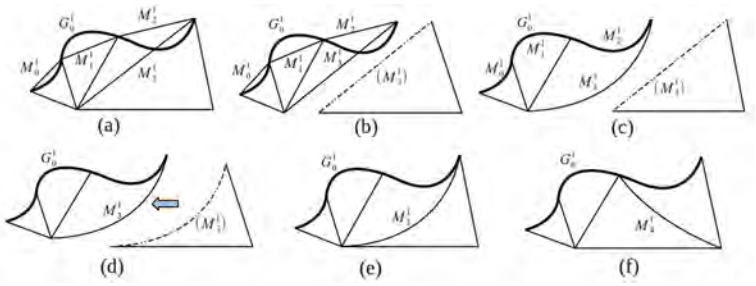


Fig. 6. A 2D example of parallel mesh curving and curved edge swap operation (a) an initial straight-sided 2D mesh, (b) the mesh is distributed to two parts, (c) boundary mesh edges M_0^1, M_1^1, M_2^1 are curved to the model edge G_0^1, interior edge M_3^1 is curved to avoid element invalidity, (d) the remote copy of M_3^1 is synchronized by its owner to also be curved, and to improve the mesh quality, M_3^1 is to be swapped, (e) the distributed mesh entities are migrated to form a local cavity on a single part, (f) M_3^1 is swapped and the new edge M_4^1 has been curved accordingly

Fig 6 demonstrates a 2D example of a combination of parallel mesh curving and curved edge swap operations. Note that the collapse/swap operation is in essence local, the full cavity 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.

3.2 Parallel SPR Based Error Estimation

In a parallel adaptive finite element simulation, a posteriori error estimation and correction indication is an important component. A parallel error estimation procedure has been developed based on the Superconvergent Patch

Recovery (SPR) scheme [22]. 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, a set of mesh vertices is characterized as being 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, and in the meantime, to avoid extensive communication cost between mesh parts, the parallel SPR based error estimator has been developed to take advantage of the ghosting functionality supported by the Flexible distributed Mesh DataBase (FMDB) [17].

3.3 Parallel Curved Mesh Adaptation with Mesh Size Field

Given an initial curved mesh in parallel and a desired mesh size field, curved mesh adaptation in parallel produces a distributed curved mesh that satisfies the size field and preserves the geometric approximation to the right order. The procedure consists of three stages: 1) curved mesh invalidity correction, 2) coarsening and iterative refinement, and 3) shape quality improvement. After the mesh adaptation, a load-balancing step is performed by using either Zoltan or ParMETIS.

Invalidity Correction for the Initial Curved Mesh

This stage eliminates all invalid elements prior to performing mesh modifications, since invalidity undermines the proper execution of the mesh modification operations. The invalidity correction algorithm detects the invalid elements by the validity checks introduced in Section 2.2. It chooses the proper local mesh modification and entity reshape operations to correct the invalidity of the curved elements. Details of the specific invalidity correction algorithm can be found in [14, 13].

Coarsening and Refinement

The coarsening process eliminates the mesh edges that are shorter than the desired length specified by the size field [11]. It is accomplished by performing entity collapse operations on the identified short edges. Serial (on-part) curved entity collapse operation are introduced in [14]. Its parallelization is discussed in Section 3.1. 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 causes element invalidity. To identify such cases, curved mesh validity checks (refer to Section 2.2) are always applied after a collapse operation. Invalidity is then resolved by the invalidity correction algorithm [14, 13] as discussed in the previous stage of this section.

After coarsening, the 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 [11, 14]. 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 [4] and/or by calculation in the parametric space using Bézier parameterization of curved tetrahedrons [7, 14, 13]. 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.

Curved Element Quality Improvement

After obtaining a satisfactory mesh with the desired size field, this stage is conducted to further improve mesh quality by edge/face swap and/or curved entity reshape operations [11, 14, 13]. Details of the parallel swap operation is presented in Section 3.1. As mentioned in 2.3, the entity reshape operation for curved elements is extremely demanding in terms of computation costs due to the explicit search procedure. Thus, it is used in the end should the swap operations fail to improve the curved element shape quality. Given a pre-specified element quality threshold Q_{th}, the elements whose shape quality $Q_{sc} < Q_{th}$ are collected to a list and are 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.

3.4 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 [14, 15]. 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 7 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).

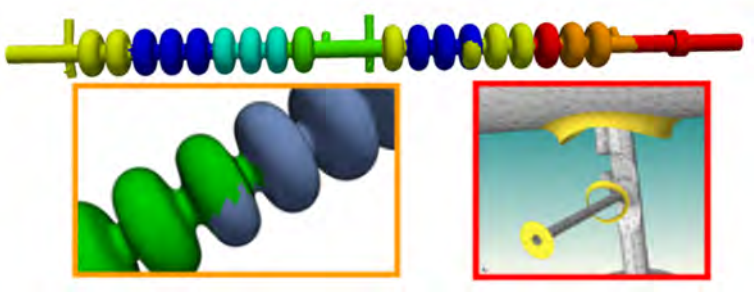


Fig. 7. Overview and close-ups of a partitioned curved mesh of linear accelerator cavities

Fig 8 shows a small example of a parallel curved mesh refinement process. 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.

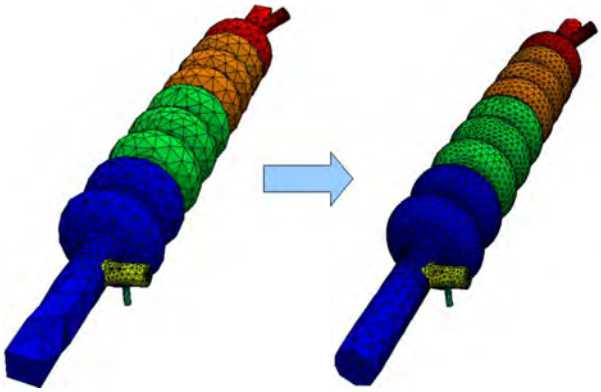


Fig. 8. An example of parallel curved mesh refinement on a four part mesh

Fig 9 gives an example of an isotropic initial curved mesh adapted to an anisotropic size field representing a planar shock.

Fig 10 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. (e)

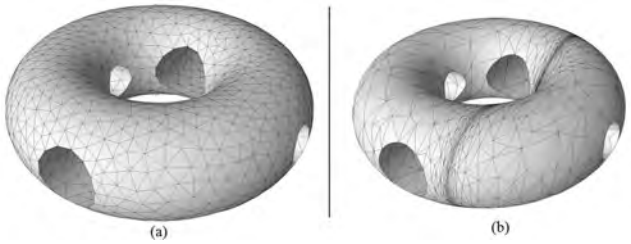


Fig. 9. An example of curved mesh adaptation with an anisotropic size field

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 pictures present a complete iteration of the developed parallel adaptive loop. 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.

4 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.

Acknowledgment

This work is supported by the US Department of Energy. The RPI portions of the work are supported through the SciDAC program through 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.

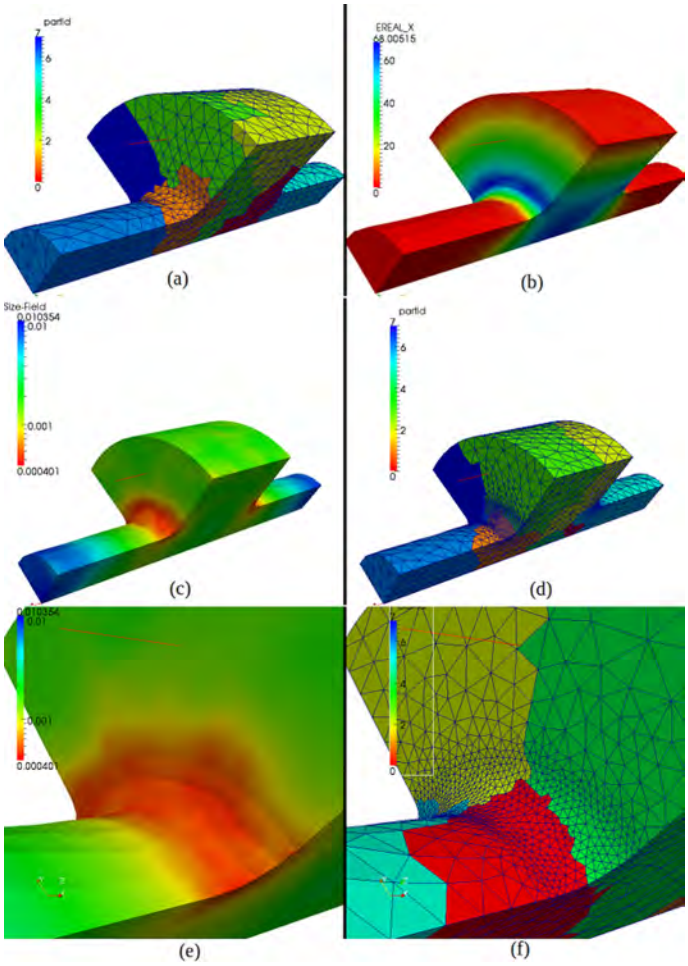


Fig. 10. Example of one iteration of the parallel adaptive loop

References

1. F. Alauzet, X. Li, E. S. Seol, and M. S. Shephard. Parallel anisotropic 3d mesh adaptation by mesh modification. *Engineering with Computers*, 21:247–258, 2006.
2. H. L. de Cougny and M. S. Shephard. Parallel refinement and coarsening of tetrahedral meshes. *International Journal for Numerical Methods in Engineering.*, 46(7):1101–1125, 1999.
3. H. L. de Cougny, M. S. Shephard, and M. K. Georges. Explicit node point mesh smoothing within the octree mesh generator. Technical report, Scientfic Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY, 1990.
4. S. Dey, R. M. O'Bara, and M. S. Shephard. Curvilinear mesh generation in 3d. *Computer-Aided Design.*, 33:199–209, 2001.

5. G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide, Third Edition.* Academic Press. Waltham, MA., 1992.
6. L. A. Freitag and P. M. Knupp. Tetrahedral element shape optimization via the jacobian determinant and condition number. In *Proceeding of the 8th International Meshing Roundtable. South Lake Tahoe, CA.*, pages 247–258, 1999.
7. P. L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 90:1156–1182, 2012.
8. A. Johnen, J.-F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. In *Proceedings of the 20th International Meshing Roundtable. Paris, France.*, 2011.
9. P. M. Knupp. Introducing the target-matrix paradigm for mesh optimization via node-movement. In *Proceedings of the 19th International Meshing Roundtable. Chattanooga, TN.*, pages 67–84, Chattanooga, TN., 2010.
10. X. Li, M. S. Shephard, and M. W. Beall. Accounting for curved domains in mesh adaptation. *International Journal for Numerical Methods in Engineering*, 58(2):247–276, 2003.
11. X. Li, M. S. Shephard, and M. W. Beall. 3d anisotropic mesh adaptation by mesh modification. *Computer Methods in Applied Mechanics and Engineering*, 194:4915–4950, 2005.
12. A. Liu and B. Joe. Relationship between tetrahedron shape measures. *BIT Numerical Mathematics*, 34(2):268–287, 1994.
13. Q. Lu. Developments of parallel curved meshing for high-order finite element simulations. Master's thesis, Rensselaer Polytechnic Institute., Troy, NY, December 2011.
14. X. Luo, M. S. Shephard, L.-Q. Lee, L. Ge, and C. Ng. Moving curved mesh adaptation for higher-order finite element simulations. *Engineering with Computers*, 27(1):41–50, 2010.
15. X. Luo, M. S. Shephard, L.-Z. Yin, R. M. O'Bara, R. Nastasi, and M. W. Beall. 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.
16. G. Morin and R. Goldman. On the smooth convergence of subdivision and degree elevation for bezier curves. *Computer Aided Geometric Design.*, 18:657–666, 2001.
17. M. Mubarak, S. Seol, Q. Lu, and M. S. Shephard. A parallel ghosting algorithm for the flexible distributed mesh database. *Submitted to Scientific Programming*, 2012.
18. P.-O. Persson and J. Peraire. Curved mesh generation and mesh refinement using lagrangian solid mechanics. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit.*, January 2009.
19. H. Prautzsch and L. Kobbelt. Convergence of subdivision and degree elevation. *Advances in Computational Mathematics*, 2:143–154, 1994.
20. T. W. Sederberg. *Computer Aided Geometric Design*. 2011. http://tom.cs.byu.edu/ 557/text/cagd.pdf; accessed May 31, 2012.
21. B. A. Szabo and I. Babuska. *Finite Element Analysis*. John Wiley & Sons Inc, New York, NY, 1991.
22. O. C. Zienkiewicz and J. Z. Zhu. 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.