DEVELOPMENTS OF PARALLEL CURVED MESHING FOR HIGH-ORDER FINITE ELEMENT SIMULATIONS

By

Qiukai Lu

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: MECHANICAL ENGINEERING

Approved:

_____

Mark S. Shephard, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

September 2011
(For Graduation December 2011)

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

The high-order finite element methods have the advantage of being able to achieve an exponential rate of convergence in the application to problems of interest that is superior to the linear rate of convergence of conventional finite element methods. However in order to fully realize the benefits of high-order finite elements for large scale simulations over general 3D curved domains, properly curved finite elements must be generated.

The work in this thesis aims to develop reliable and efficient parallel curved meshing techniques to support large scale simulations using higher-order finite element methods. The mathematical fundamentals of the Bézier polynomial based shape representation for high-order curved elements are reviewed. Based on the shape representation, a hybrid shape metric is proposed to serve for both linear straight-sided elements and high-order curved elements. Novel extensions to the Bézier control point based element validity check method are proposed and developed. Numerical experiments have shown results that they are effective with almost neglegible addition to the computational cost.

Technical developments of parallel curved meshing have been presented in terms of creating and adapting partitioned curved meshes. Two alternative approaches to create large partitioned curved meshes have been developed. Efforts have been made to extend linear straight-sided mesh modification procedures to work with high-order curved meshes in parallel. The approach starts from the existing functionalities of parallel linear mesh adaptation and serial mesh curving, both of which have been developed in the SCOREC MeshAdapt software.

CHAPTER 1
Introduction

1.1 Background and Historical Review

In the area of numerical analysis of partial differential equations, several classes of numerical methods have been extensively studied, such as the finite difference methods, finite volume methods and finite element methods [71, 29, 5], etc. Among those methods, the finite element methods (FEM) have gained the most popularity. There are several reasons that make it superior compared with other numerical methods. It deals with problems defined over general domains as long as the domain can be correctly decomposed into small 'elements'. Very little or even no code modification is needed if boundary conditions or physical parameters change [29]. Therefore, numerous simulation software packages have been implemented with finite element methods and are widely used in both academic research and industry.

The mathematical properties of the finite element methods have been well established, which involves the process of discretize the problem domain by piecewise continuous functions. Extensive theoretical studies have shown that the accuracy of the numerical solution and efficiency of the analysis process depend closely on the quality of the discretization called a finite element mesh [37, 42, 43].

In an adaptive simulation, the initial mesh is generated based on some *a priori* knowledge of the desired solution field over the domain of interest. One can evaluate the accuracy of the solution field based on the initial mesh using error estimators [4, 76, 77]. In the cases which the solution is not accurate enough, mesh adaptation techniques are used to modify the discretization based on an error estimator and/or correction indicator. The accuracy and efficiency of an adaptive simulation relies greatly on the capability of mesh generation and adaptation. Mesh adaptation procedures can be roughly categorized into h-version [13, 37], p-version [42, 43, 44] and hp-version [16, 17], etc. The h-version modifies the size of the element while the p-version alters the polynomial degree. The hp-version combines the two together. Recently as the isogeometric analysis is being studied, a so-called k-version

adaptivity is proposed as well [30]. The work presented in this thesis focuses on the p-version mesh adaptivity to support the high-order finite element methods.

1.2 Motivation and Challenges

The high-order finite element methods have the advantage of being able to achieve an exponential rate of convergence in the application to problems of interest that is superior to conventional methods [71, 5]. However in order to fully realize the benefits of high-order finite elements for large scale simulations over general 3D curved domains, properly curved finite elements must be generated [43]. Such curved meshes must have at least validity elements, and preserve sufficiently accurate geometric approximation to the curved domain boundary. Furthermore, for the problems that require the numerical solution to have really high resolution and accuracy, meshes with extremely large amount of entities are desired. Such meshes can only be created using distributed computing systems in parallel, which requires effective and efficient parallel meshing techniques to be developed [14].

There have been extensive studies in the area of automatic mesh generation and adaptation. However, the majority efforts have been focused on dealing with standard lower-order meshes with straight-sided elements. To approximate 3D curved domains with linear meshes, one has to sufficiently refine the mesh in the vicinity of key features of interest, which usually results in large number of elements [37]. The use of properly constructed curved mesh entities could effectively approximate the domain with relatively small meshes. Literatures can be found studying and proposing meshing techniques for higher-order curved meshes [43, 58].

Recent developments in parallel computing facilities and paradigms have enabled large scale simulations in the areas of molecular dynamics, computational fluid dynamics, and electromagnetic simulations etc. In order to fully make use of the computing resources to reach desired solution accuracy and resolution, these simulations can require billions of degrees of freedom. Therefore extremely large meshes have to be generated using multiple processors in parallel [73]. To date, research of parallel mesh generation and adaptation has been primarily focused on the linear straight-sided meshes.

The work in this thesis aims to develop reliable and efficient parallel curved meshing techniques to support large scale simulations using higher-order finite element methods. The current approach starts from the existing functionalities of parallel linear mesh adaptation and serial mesh curving, both of which have been developed in the SCOREC MeshAdapt software. When constructing curved mesh entities, interactions with the underlying CAD modeling engines are essential in order to approximate the geometric model boundaries more closely. In the parallel distributed computing environment, extra considerations are needed to make sure the correct information can be exchanged among the communicating processors. In addition, the entities shared by multiple processors must be carefully synchronized to maintain global consistency.

1.3 Thesis Organization

This thesis is organized as follows: Chapter 2 introduces the mathematical foundation of a Bézier polynomial based shape representation of high-order curved tetrahedrons, and discusses various properties of Bézier polynomials. Following that, Chapter 3 then presents a criterion for determining validity of a general high-order curved tetrahedron, and a validity check algorithm is presented. A novel shape metric is also proposed to be generally applicable in the cases where both linear and curved elements exist. Chapter 4 and 5 focuses on technical aspects of parallel curved meshing procedures. The design and implementations of the algorithms are discussed along with several examples. Chapter 4 focuses on the generation of a valid curved mesh starting from a given valid linear mesh, and Chapter 5 focuses on the adaptation of a given curved mesh. Applications of the parallel curved meshing techniques are given in Chapter 6. The most successful application so far is to support the parallel high-order finite element simulation techniques developed at SLAC National Accelerator Laboratory for the design of next generation large scale particle accelerators. Conclusions and future works are discussed in Chapter 7.

1.4 Nomenclature

The nomenclature used in this chapter follows what has been defined in [43]:

Ω_υ	Domain of interest, $\upsilon = G, M$ where G denotes the geometric model and M denotes the mesh model
$\partial\Omega_\upsilon$	Boundary of the domain Ω_υ
$\overline{\Omega}_\upsilon$	Closure of the domain, $\overline{\Omega}_\upsilon = (\Omega_\upsilon \cup \partial\Omega_\upsilon)$
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 $M_i^{d_i}$.
$b_i^{(n)}(t)$	the ith Bernstein basis polynomial of degree n.
$P_i^{(n)}(M_j^d)$	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.

CHAPTER 2
Overview of the Parametric Representation of High-order Curved Finite Element Volumes

2.1 Introduction

Driven by the developments of high-order finite element analysis and applications, for instance the p-version FEM, effective curved mesh generation and adaptation techniques have become an important ingredient to construct adaptive loops of automatic finite element simulations. However, compared with linear straight-sided meshing techniques, generating and adapting high-order meshes meets more complicated constraints in terms of approximating the geometry of domain boundaries. Therefore it requires more carefully designed metric to represent the curved entities and determine the validity of the mesh. A parametric representation for curved tetrahedral elements based on the Bézier polynomials has been developed and implemented [43, 44].

2.2 The Bernstein Polynomials and Bézier Curves

In computer aided geometric design, Bézier polynomials are frequently used to construct 3-dimensional curves and surface patches in parametric forms [19]. For example, a single-variable Bézier polynomial with vector-valued control points can be used to construct a general curve in the 2D or 3D physical space. In general, a n^{th} order Bézier curve can be expressed as:

$$\mathbf{B}(t) = \sum_{i=0}^{n} b_i^{(n)}(t)\mathbf{P}_i^{(n)}, t \in [0,1] \tag{2.1}$$

In the equation above, $b_i^{(n)}(t)$ is the ith Bernstein basis polynomial of degree n,

$$\mathbf{b}_i^{(n)}(t) = \binom{n}{i} t^i (1-t)^{n-i}, i = 0, ..., n; \tag{2.2}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}. \tag{2.3}$$

and $\mathbf{P}_i^{(n)}$ is the ith control point of a nth order Bézier curve.

A single-variable Bézier polynomial maintains a mapping between the representations of a curve in the physical space and parametric space. If the control points are in 3D, i.e. $\mathbf{P}_i = (P_x, P_y, P_z)^T \in R^3$, then $B(t)$ maps a one-dimensional parametric coordinate system ($t \in R$) to a 3D Cartesian coordinate system ($\mathbf{B}(t) \in R^3$), and vice versa.

Bézier polynomials have some very useful properties [19]. Key ones of importance to the development of curved mesh algorithms are listed as follows:

1. The Convex Hull Property. The max and min values of a Bézier polynomial evaluated within the domain are bounded by the max and min values of its corresponding control points,

2. The derivatives of a Bézier polynomial are still Bézier polynomials. The product of a m^{th} order Bézier polynomial and a n^{th} order Bézier polynomial is also a Bézier polynomial of order $(m + n)$,

3. Easy to perform subdivision or degree elevation. This property is useful in the h- and p-version mesh adaptation.

2.3 Bézier Polynomial Based Representation of High-order Tetrahedral Volume

It is not difficult to generalize the Bézier polynomial to represent a parametric surface or volume. Using the notation in [44], the q^{th} order Bézier polynomial based volume representation of high-order tetrahedrons can be expressed by making use of the volume coordinates as follows:

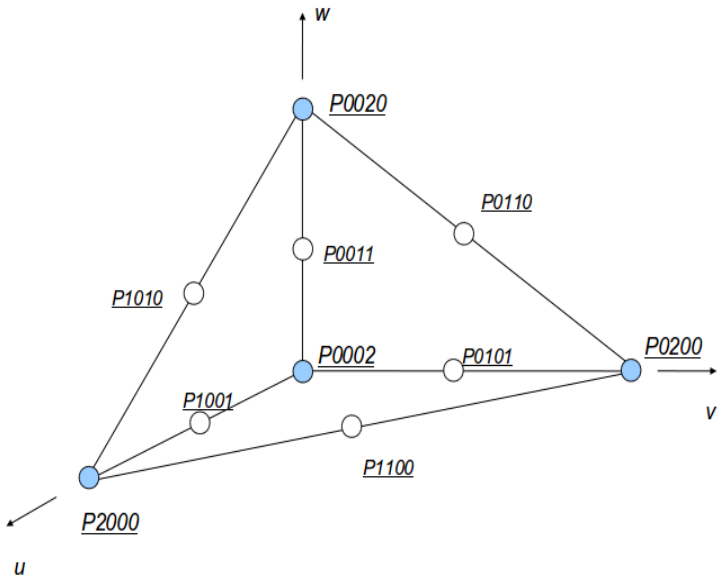


Figure 2.1: 2nd-order tetrahedron

$$\mathbf{X}^{(n)}(M_j^3) = \mathbf{X}^{(n)}(\xi) = \sum_{i=1}^{q} \mathbf{P}_i^{(n)} b_i^{(n)}(\xi) \tag{2.4}$$

where $b_i^{(n)}(\xi)$ are the Bernstein polynomials with $\xi = \{(\xi_1, \xi_2, \xi_3, \xi_4) | \xi_1 + \xi_2 + \xi_3 + \xi_0 = 1 \text{ and } \xi_i \in [0, 1]\}$. q is the total number of control points for a nth order Bézier polynomial in 3D.

More specifically in the case of a 2^{nd}-order curved tetrahedron, the Bézier representation is:

$$\begin{aligned} \mathbf{X}^{(2)}(\xi_1, \xi_2, \xi_3, \xi_4) &= \sum_{i=1}^{4} \mathbf{P}^{(2)}(M_i^0) C_1 \xi_i^2 \\ &+ \sum_{j=1}^{6} \mathbf{P}^{(2)}(M_j^1) C_2 \xi_m \xi_n \end{aligned} \tag{2.5}$$

where $\mathbf{P}_i^{(2)}(M_j^0)$ are four control points associated with the vertices, and $\mathbf{P}_i^{(2)}(M_j^1)$ are six control points associated with the edges. $C_i = \frac{2!}{i!}$ are the coefficients of the second order Bernstein polynomials.

The Bézier representation above is a vector-valued polynomial. Each value

corresponds to a point within the volume of the tetrahedron.

The Jacobian matrix is defined as the first-order partial derivatives of the physical coordinates with respect to the parametric coordinates (using indicial notation):

$$\mathbf{J} = \frac{\partial X_i}{\partial \xi_j}, \quad i, j = 1, 2, 3; \tag{2.6}$$

and its determinant is:

$$det(\mathbf{J}) = \frac{\partial \mathbf{X}}{\partial \xi_1} \times \frac{\partial \mathbf{X}}{\partial \xi_2} \cdot \frac{\partial \mathbf{X}}{\partial \xi_3} \tag{2.7}$$

The determinant of Jacobian at a given point gives important information near that point. For instance, a continuously differentiable function is invertible near a point P if the determinant of Jacobian at P is non-zero. Furthermore, if the determinant of Jacobian at P is positive, then the function preserves orientation near P; if it is negative, the function reverses orientation. The absolute value of the determinant of Jacobian at P gives us the factor by which the function expands or shrinks volumes near P.

In the specific case of a 2nd order tetrahedron, the derivatives with respect to ξ_1, ξ_2 and ξ_3 are first order Bézier polynomials:

$$\begin{aligned} \frac{\partial X}{\partial \xi_1} &= 2(P(M_2^0) - P(M_1^1))\xi_1 + 2(P(M_2^1) - P(M_3^1))\xi_2 \\ &+ 2(P(M_5^1) - P(M_4^1))\xi_3 + 2(P(M_1^1) - P(M_1^0)\xi_4 \end{aligned} \tag{2.8}$$

$$\begin{aligned} \frac{\partial X}{\partial \xi_2} &= 2(P(M_2^1) - P(M_1^1))\xi_1 + 2(P(M_3^0) - P(M_3^1))\xi_2 \\ &+ 2(P(M_6^1) - P(M_4^1))\xi_3 + 2(P(M_3^1) - P(M_1^0))\xi_4 \end{aligned} \tag{2.9}$$

$$\frac{\partial X}{\partial \xi_3} = 2(P(M_5^1) - P(M_1^1))\xi_1 + 2(P(M_6^1) - P(M_3^1))\xi_2$$

$$+ \quad 2(P(M_4^0) - P(M_4^1))\xi_3 + 2(P(M_4^1) - P(M_1^0))\xi_4 \tag{2.10}$$

Denote the vector coefficients in front of ξ_1, ξ_2, ξ_3 and ξ_4 as $a_i, b_i, c_i, d_i, i = 1, 2, 3$ respectively.

$det(J)$ is the product of the three first order Bézier polynomials, and therefore is a 3rd order Bézier polynomial, and its a scalar polynomial:

$$det(J) = \sum_{i=1}^{20} P_i^{(3)} b_i^{(3)}(\xi) \tag{2.11}$$

The control points of this 3rd order Bézier polynomial with respect to their corresponding Bernstein basis terms are given in Table 2.1. There are 20 control points in total, including 4 for the vertices, 12 for the edges, and 4 for the faces.

Table 2.1: The 20 control points and polynomial terms of a third-order three-variable Bézier polynomial

	Control Points $P^{(3)}_{\|i\|}$	Polynomials terms $b^{(3)}_{\|i\|}$
Vertex point 1:	$(a_1 \times a_2 \cdot a_3)$,	ξ_1^3
Vertex point 2:	$(b_1 \times b_2 \cdot b_3)$,	ξ_2^3
Vertex point 3:	$(c_1 \times c_2 \cdot c_3)$,	ξ_3^3
Vertex point 4:	$(d_1 \times d_2 \cdot d_3)$,	ξ_4^3
Edge point 1:	$\frac{1}{3}[(a_1 \times a_2 \cdot b_3) + (a_1 \times b_2 \cdot a_3) + (b_1 \times a_2 \cdot a_3)]$,	$3\xi_1^2\xi_2$
Edge point 2:	$\frac{1}{3}[(a_1 \times b_2 \cdot b_3) + (b_1 \times a_2 \cdot b_3) + (b_1 \times b_2 \cdot a_3)]$,	$3\xi_1\xi_2^2$
Edge point 3:	$\frac{1}{3}[(a_1 \times a_2 \cdot c_3) + (a_1 \times c_2 \cdot a_3) + (c_1 \times a_2 \cdot a_3)]$,	$3\xi_1^2\xi_3$
Edge point 4:	$\frac{1}{3}[(a_1 \times c_2 \cdot c_3) + (c_1 \times a_2 \cdot c_3) + (c_1 \times c_2 \cdot a_3)]$,	$3\xi_1\xi_3^2$
Edge point 5:	$\frac{1}{3}[(a_1 \times a_2 \cdot d_3) + (a_1 \times d_2 \cdot a_3) + (d_1 \times a_2 \cdot a_3)]$,	$3\xi_1^2\xi_4$
Edge point 6:	$\frac{1}{3}[(a_1 \times d_2 \cdot d_3) + (d_1 \times a_2 \cdot d_3) + (d_1 \times d_2 \cdot a_3)]$,	$3\xi_1\xi_4^2$
Edge point 7:	$\frac{1}{3}[(b_1 \times b_2 \cdot c_3) + (b_1 \times c_2 \cdot b_3) + (c_1 \times b_2 \cdot b_3)]$,	$3\xi_2^2\xi_3$
Edge point 8:	$\frac{1}{3}[(b_1 \times c_2 \cdot c_3) + (c_1 \times b_2 \cdot c_3) + (c_1 \times c_2 \cdot b_3)]$,	$3\xi_2\xi_3^2$
Edge point 9:	$\frac{1}{3}[(b_1 \times b_2 \cdot d_3) + (b_1 \times d_2 \cdot b_3) + (d_1 \times b_2 \cdot b_3)]$,	$3\xi_2^2\xi_4$
Edge point 10:	$\frac{1}{3}[(b_1 \times d_2 \cdot d_3) + (d_1 \times b_2 \cdot d_3) + (d_1 \times d_2 \cdot b_3)]$,	$3\xi_2\xi_4^2$
Edge point 11:	$\frac{1}{3}[(c_1 \times c_2 \cdot d_3) + (c_1 \times d_2 \cdot c_3) + (d_1 \times c_2 \cdot c_3)]$,	$3\xi_3^2\xi_4$
Edge point 12:	$\frac{1}{3}[(c_1 \times d_2 \cdot d_3) + (d_1 \times c_2 \cdot d_3) + (d_1 \times d_2 \cdot c_3)]$,	$3\xi_3\xi_4^2$
Face point 1:	$\frac{1}{6}[(a_1 \times b_2 \cdot c_3) + (a_1 \times c_2 \cdot b_3) + (b_1 \times a_2 \cdot c_3)$ $+(b_1 \times c_2 \cdot a_3) + (c_1 \times a_2 \cdot b_3) + (c_1 \times b_2 \cdot a_3)]$,	$6\xi_1\xi_2\xi_3$
Face point 2:	$\frac{1}{6}[(a_1 \times b_2 \cdot d_3) + (a_1 \times d_2 \cdot b_3) + (b_1 \times a_2 \cdot d_3)$ $+(b_1 \times d_2 \cdot a_3) + (d_1 \times a_2 \cdot b_3) + (d_1 \times b_2 \cdot a_3)]$,	$6\xi_1\xi_2\xi_4$
Face point 3:	$\frac{1}{6}[(a_1 \times d_2 \cdot c_3) + (a_1 \times c_2 \cdot d_3) + (d_1 \times a_2 \cdot c_3)$ $+(d_1 \times c_2 \cdot a_3) + (c_1 \times a_2 \cdot d_3) + (c_1 \times d_2 \cdot a_3)]$,	$6\xi_1\xi_3\xi_4$
Face point 4:	$\frac{1}{6}[(d_1 \times b_2 \cdot c_3) + (d_1 \times c_2 \cdot b_3) + (b_1 \times d_2 \cdot c_3)$ $+(b_1 \times c_2 \cdot d_3) + (c_1 \times d_2 \cdot b_3) + (c_1 \times b_2 \cdot d_3)]$,	$6\xi_2\xi_3\xi_4$

CHAPTER 3
Hybrid Element Quality Metric and Validity Check for High-order Tetrahedron

This chapter overviews the common mesh quality metrics and a validity check method based on the Bézier shape representation of high-order curved tetrahedral elements. A quality metric that combines a straight-sided and a curved shape metric is proposed and its properties discussed. An adaptive validity check is developed using the new shape metric.

3.1 Overview of Common Quality Metrics for 3D Tetrahedron

There has been substantial effort trying to put a precise definition to what an *a priori* (element geometry only) mesh quality metric is. A general definition given by Knupp [32] states that a mesh quality metric is a scalar function that measures some geometric property of that given element. Another more specific definition for tetrahedron proposed by Dompierre et al in [18] requires the metric to 1) be invariant under translation, rotation, reflection and uniform scaling of the tetrahedron, 2) have unique maximum for equilateral tetrahedron and unique minimum for degenerated tetrahedron. A number of element quality metrics that conform to the above definitions have been proposed and studied. Each metric targets certain parameters of a tetrahedron, such as edge length, dihedral angle, non-dimensional ratio parameters, etc. Regardless of what parametera are selected, a desired mesh quality metric should be 1) effective to detect all kinds of poorly shaped elements, 2) efficient in terms of computational cost, and 3) informative to drive the mesh adaptation procedures to improve the mesh quality. Since any complete measure of element quality must consider *a posteriori* information, the term poorly shaped element used here is primarily concerned with elements whose shape is such that the Jacobian over the element is likely to introduce numerical

problems into calculations performed using that element.

3.1.1 Quality Metrics for Straight-sided Tetrahedral Elements

Given a straight-sided tetrahedral mesh, the geometric shape of an element is uniquely defined once the positions of its four nodes are determined since the edges are straight lines and faces are flat planes. It is straight-forward to calculate various geometric quantities, such as edge length and solid angle, etc. Therefore, a host of mesh quality metrics for straight-sided elements have been proposed which were based upon geometric quantities. Several commonly used ones are reviewed as follows.

1. Edge Ratio:

   The Edge Ratio r is defined to be the ratio of the shortest edge over the longest edge in a given tetrahedron [18]:

   $$r = \frac{\min_{i=1..6} l(M_i^1)}{\max_{j=1..6} l(M_j^1)}, \tag{3.1}$$

   where l denotes the length of one of six edges. See Figure 3.1.

   It is obvious that $0 \leq r \leq 1$ for all elements, and r reaches 1 for an equilateral tetrahedron. On the other hand, if r is very small or even close to 0, it indicates that the tetrahedron might be highly anisotropic with one or many edges being relatively shorter/longer than others. However, this shape metric fails in cases where the element is indeed flat while none of its edges are degenerated to zero length as shown in Figure 3.2. It also can not detect inverted elements. This metric only uses mesh edges and calculates their length, therefore it is one of the most computationally efficient metrics.

2. Dihedral Angle:

   In a given tetrahedron, each of the six edges is used by two mesh faces. The dihedral angle θ is defined to be the angle of two intersecting faces [18]. It can be obtained by adding a perpendicular plane to the edge and measure the angle between the two intersecting lines. See Figure 3.3.

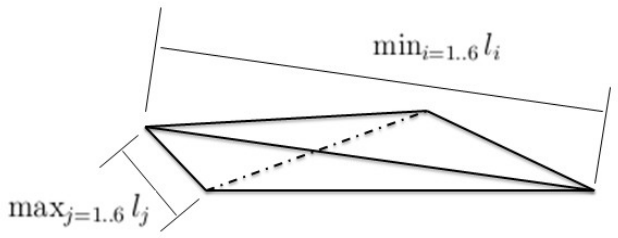


Figure 3.1: Definition of the edge ratio metric

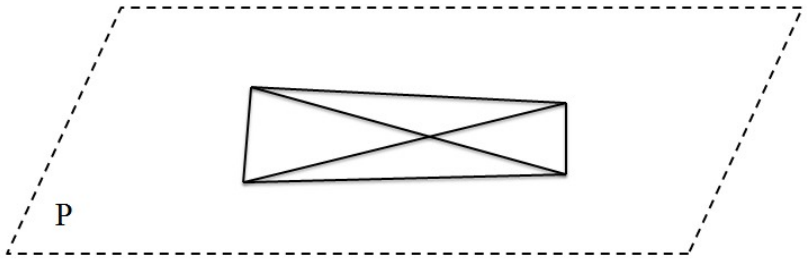


Figure 3.2: An example of a flat element in plane *P*

The dihedral angle is an effective metric to detect sliver and flat elements as θ approaches 0 or π. However it lacks information of the length scale of the element. A variant of the dihedral angle is discussed in [27] that non-dimensionalize the quantity by computing

$$q = \sin(\theta), \tag{3.2}$$

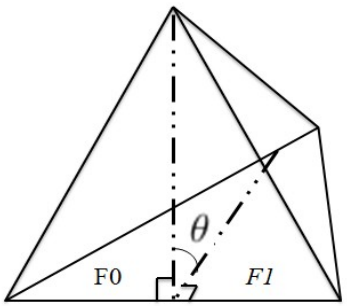


Figure 3.3: Definition of dihedral angle

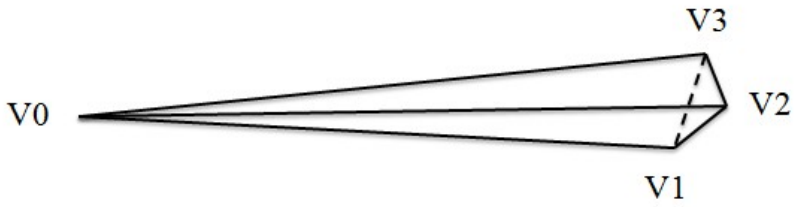


Figure 3.4: An example of needle-shaped straight-sided tetrahedron

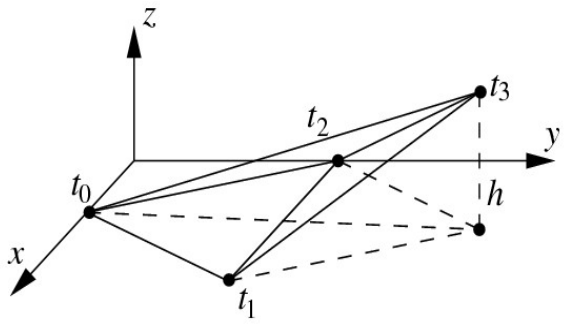


Figure 3.5: Definition of the aspect ratio metric

This metric targets at elements with small or large dihedral angles but does not detect needle-shaped tetrahedron as shown in Figure 3.4. The example tetrahedron is highly distorted, however the dihedral angles are still in an acceptable range. In addition, evaluation of the angles requires calculating trigonometric functions which is usually more expensive than normal floating point arithmetics such as additions or multiplications. Therefore this metric is more expensive than the Edge Ratio metric.

3. Aspect Ratio:

   The aspect ratio is defined as the ratio between the minimum altitude and the length of the longest edge of a given tetrahedron.

$$\rho = \frac{\sqrt{6} h_{min}}{2 l_{max}} \tag{3.3}$$

   where h_{min} is the smallest altitude, l_{max} is the longest edge length. See Figure 3.5.

   Its normalized inverse form is also used as a quality metric and is discussed

in [27]. The aspect ratio metric ranges from 0 to 1 for a valid element and is able to effectively detect slivers as well as highly anisotropic elements in which case h_{min} is much smaller than l_{max}. Therefore it is a metric that is able to detect all types of poorly shaped elements. And if h_{min} is defined to be a signed height, i.e. negative values are allowed, then it detects invalid elements as well.

4. Mean Ratio:

   The definition of the mean ratio metric takes into account the length of all six edges and the volume of the tetrahedron [39, 40, 18]. It is computed as:

$$\eta = K \frac{V(M^3)^2}{\sum_{i=1}^{6} l(M_i^1)^3} \tag{3.4}$$

   For a valid element, K is a scaling factor that multiplies to rescale η to the range [0, 1]. In the optimal case, an equilateral tetrahedron has $\eta = 1$ under this metric. A flat or degenerated element has zero volume which leads to $\eta = 0$. In the cases that an element is inverted, η becomes negative since the volume of the element is negative. Therefore, this metric has also been shown to be able to detect all types of poorly-shaped and invalid elements [27].

Since both Aspect Ratio (AR) and Mean Ratio (MR) metrics are desired in terms of effectiveness, it is worthwhile to further compare the computational cost of both metrics. Studies have shown that MR is less expensive than AR in terms of the number of basic computations $(+, -, \times, \div)$ and square roots [57]. Also, AR requires evaluation of face areas in order to obtain h_{min} while MR does not and it has been studied in [6, 65] that in unstructured tetrahedral meshes the number of mesh faces is larger that the number of edge by a ratio of about $\frac{12}{7}$. Therefore, MR is generally more efficient than AR in terms of computational cost.

Aside from measuring the mesh quality by the mesh entity geometric quantities, there is another type of geometric quality metrics focusing on evaluating the Jacobian and related matrices of a mesh element, which is referred to as algebraic mesh quality metrics in a set of publications by Knupp et al [32, 33].

In vector calculus, the Jacobian relates the derivatives of a set of variables in one coordinate system to the derivatives of the same variables in a second coordinate system. For 3D finite element analysis, the Jacobian matrix is a 3×3 square matrix defined as the first-order partial derivatives of the physical coordinates (x_1, x_2, x_3), with respect to the parametric coordinates (ξ_1, ξ_2, ξ_3):

$$\mathbf{J}(\xi) = \frac{\partial X_i}{\partial \xi_j}, \quad i, j = 1, 2, 3; \tag{3.5}$$

and its determinant can be calculated as:

$$det(\mathbf{J}) = \frac{\partial \mathbf{X}}{\partial \xi_1} \times \frac{\partial \mathbf{X}}{\partial \xi_2} \cdot \frac{\partial \mathbf{X}}{\partial \xi_3} \tag{3.6}$$

In theory, an essential requirement for a FE analysis to be performed is that the mapping between the physical space and parametric space remains one-to-one and onto over each mesh element [29], which requires the determinant of the Jacobian matrix $det(J)$ to be positive for the element. Consequently, if a negative $det(J)$ is found anywhere in the element, it is identified as invalid.

In the case of straight-sided tetrahedral meshes and piece-wise linear shape functions, the mapping between X and ξ is linear, in which case the Jacobian is constant over the element domain and easy to calculate. Important information can be derived from a Jacobian in terms of the shape, dilatation, orientation of an element in physical space with respect to its reference in the parametric space. Therefore the Jacobian and related matrices provide appropriate information for building mesh quality metrics. Determinant, trace and norm are useful operators that relate matrices to scalar quantities. Based on those operators, various Jacobian-based algebraic quality metrics have been developed. Several typical ones are [32, 33, 34]:

- Determinant: $\tau = det(J)$,
- Volume: τ^2,
- Frobenius norm: $|J|^2 = trace(J^T J)$,
- Condition number: $\kappa = |J||J^{-1}|$.

More detailed discussions of their property and applications in mesh smoothing and adaptation can be found in Knupp [32, 33] and Bucki et al [10].

3.1.2 Quality Metric for High-order Curved Tetrahedron

There are far fewer geometric mesh quality metrics proposed to date for high-order curved tetrahedral elements than for straight-sided elements, partially due to the difficulties to calculating the geometric quantities such as length of a curved edge or area of a curved surface. Besides that, none of the geometric entity based quality metric discussed above account for the influence of the mapping of high-order curved elements on the properties of the numerical system. Although the Jacobian based algebraic quality metrics do provide knowledge about the mapping of the elements, the ones in Knupp[refs here] can not be easily applied to high-order curved elements since the Jacobian is not constant through the element domain when elements have curved geometry. It is generally a function of position in the parametric coordinates.

Although many geometric and/or algebraic parameters can not be readily used for measuring the quality of a high-order tetrahedral element, the Jacobian matrix remains one of the most influential factors as it is explicitly used in the numerical integration of the stiffness matrix over the element. It is well-known that negative determinant of Jacobian $det(J)$ evaluated at integration points could easily compromise solution accuracy and case the solver to halt prematurely [67, 43, 71, 29]. Elements with such Jacobian matrix are identified as invalid. Even with positive $det(J)$, large variations of $det(J)$ over the element will contribute to numerical stiffening which will cause the solution to converge very slowly [67]. Such elements are categorized as poorly-shaped.

A type of quality metrics for high-order curved tetrahedron has been proposed and studied by Shephard et al which identifies the invalid as well as poorly-shaped high-order curved elements by evaluating the scaled variations of the determinant of Jacobian over the element domain [67, 16, 17, 44]. The metric is defined as:

$$q_c = \frac{\min_{\xi \in \Omega^e} det(J(\xi))}{\max_{\xi \in \Omega^e} det(J(\xi))} \tag{3.7}$$

This metric normalizes the variations of the determinant of the Jacobian by

rescaling the minimum value with respect to the maximum. Its range is within [0, 1] for valid curved elements, while being negative for invalid elements. It gives information about how distorted the specific tetrahedron is in the physical space. However, this scaled metric only considers the shape deviation of a curved element with respect to its underlying straight-sided counterpart, it does not consider the shape quality of the straight-sided element itself. Therefore, if a high-order tetrahedron has all straight-sided edges, then its $det(J)$ is again constant over the volume. Consequently the metric q_c reports the optimal value 1, even if the straight-sided tetrahedron is highly anisotropic or even close to being degenerated. Therefore, q_c alone does not capture all element geometric shape concerns.

3.2 The Hybrid Shape Quality Metric

In order to overcome the issue discussed above and effectively measure the quality for both straight-sided and curved meshes, a hybrid quality metric is developed which combines a straight-sided mesh quality metric and a curved mesh quality metric.

Let q_s be any selected quality metric for straight-sided elements and q_c for curved elements. m and n are selected weighting constants. The hybrid metric computes the mesh quality of curved elements as a simple product:

$$Q_{sc} = q_s^m \times q_c^n \tag{3.8}$$

This quality metric effectively combines the quality metrics for both straight-sided and curved elements. In the case of straight-sided elements where $q_c = 1$, $Q_{sc} = q_s$ functions alone to measure the element shape. For curved elements, q_c will be computed and contribute to Q_{sc} together with the underlying straight-sided shape quality q_s. The two power constants m and n can be tuned as needed to change the influence of either q_s or q_c on the overall metric Q_{sc}. Note that in the case that q_s and q_c are normalized metrics within range [0, 1], m and n has to be non-negative in order for Q_{sc} to also be a normalized metric.

Differentiating Equation 3.8 with respect to either of the component quality

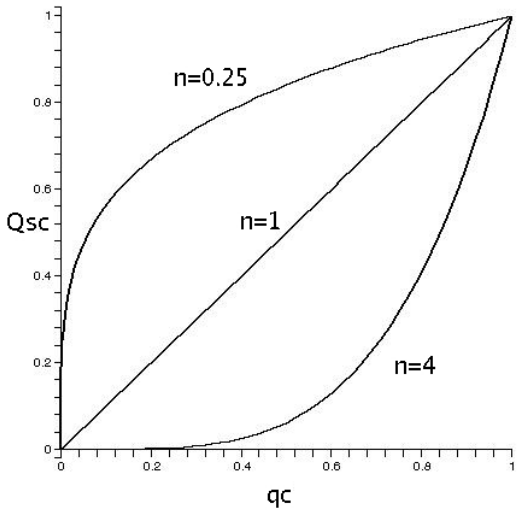


Figure 3.6: Plot of Q_{sc} with respect to q_c for different weighting constant n, assuming $q_s = 1$

metrics, say q_c, while holding the other fixed, in this case q_s. We get:

$$Q_{,c} = \frac{\partial Q_{sc}}{\partial q_c} = C q_c^{n-1} \tag{3.9}$$

where $C = n q_s^m$.

Assuming $C > 0$, Equation 3.9 computes the slope of Q_{sc} with respect to the curved quality component. For example, if one selects $n = 1$, then $Q_{,c} = C$ is constant with respect to variations in q_c. Therefore it is equally sensitive to the curved element quality within range [0, 1]. If $n > 1$ is selected, $Q_{,c}$ is large in the high quality range of q_c, and Q_{sc} drops very rapidly as q_c starts to degrade from very good quality to relatively poor quality, which leads to higher sensitivity of Q_{sc} to the curved element distortion, therefore detects poorly-shaped curved elements very effectively. On the other hand, if $0 < n < 1$, Q_{sc} starts to rapidly decrease as q_c approaches truly low quality that is close to 0. Therefore it is generally less strict on curved element distortion than the $n > 1$ cases and focuses on detecting the poor quality element affected by its straight-sided component. See Figure 3.6 for examples of the three situations.

In Chapter 4, this quality metric serves as the basis to support the explicit nodal repositioning algorithm to identity poorly-shaped tetrahedral elements and

improve element shape quality.

3.3 The Validity Condition of Curved Element

To identify a valid curved tetrahedral element regardless of the numerical integration scheme being used, it is sufficient to make sure positive determinant of Jacobian throughout the domain of the tetrahedron.

$$det(J)|_{(\xi_1,\xi_2,\xi_3,\xi_4)} > 0 \tag{3.10}$$

However, it is not feasible in practice to go through each and every point of the volume to check directly the value of determinant of Jacobian. Therefore an alternative method is used to consider the bounds of the determinant of Jacobian.

According to the Convex Hull property of the Bézier representation of a high-order curved tetrahedron discussed in Chapter 2, the determinant of the Jacobian $det(J)$ can also be represented as a Bézier polynomial of order $3(p-1)$ over the tetrahedron, where p is the order of the tetrahedral element. Therefore it 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 second-order tetrahedron, the following inequality holds:

$$\min\{P^{(3)}_{|i|}\} \leq det(J) \leq \max\{P^{(3)}_{|i|}\} \tag{3.11}$$

where $P^{(3)}_{|i|}$ represents the value at the ith control point.

Consequently, 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:

$$\min\{P^{(3)}_{|i|}\} > 0 \tag{3.12}$$

3.4 The Uniform Validity Check Method

The uniform validity check algorithm is based on the above condition by checking all the control points of a Bézier representation. The total number of control

points of a Bézier polynomial is uniquely determined by its order. In the case of second-order curved tetrahedral element, the Bézier polynomial representing $det(J)$ is of order 3. The total number of control points is 20. The algorithm is described as follows:

Given a second-order curved tetrahedron

1. Get the Bézier control points of the curved tetrahedron,
2. Calculate the vectors a_i, b_i, c_i, d_i, $i = 1, 2, 3$ as defined by Equations 2.8, 2.9 and 2.10 in Chapter 2,
3. Compute the control points $P^{(3)}_{|i|}$ (scalar) for the 3rd order Bézier polynomial $det(J)$ according to Table 2.1 in Chapter 2,
4. Find $\min\{P^{(3)}_{|i|}\}$ among the 20 control points. If $\min\{P^{(3)}_{|i|}\} > 0$, the 2nd order curved tetrahedron is valid.

This algorithm is computationally efficient and is independent of the numerical integration schemes compared with evaluating the real determinant of Jacobian at the quadrature points based on the integration rules. However, due to fact that this algorithm uses a sufficient condition that evaluates the lower bound of $det(J)$, it can be overly-conservative in cases where the lower and upper bounds are not very tight. In such cases, the actual $det(J)$ could still be positive over the entire volume of the tetrahedron even if $\min\{P^{(3)}_{|i|}\}$ is negative.

The degree of conservativeness of the lower and upper bounds obtained by this validity check algorithm depends upon various factors. Essentially, they depend on the number of control points used to represent the polynomial, the fewer the number of control points is, the more conservative the measure is. Besides that, the conservativeness also depends on the classification of the control point where the minimum value is reached. The validity check algorithm is accurate if minimum value is found at an interpolation point where $\min\{P^{(3)}_{|i|}\} = \min\{det(J)\}$, while conservative if minimum is at non-interpolating points where $\min\{P^{(3)}_{|i|}\} \leq \min\{det(J)\}$. Details are discussed in the next two subsections.

Note that for a tetrahedron of order higher than quadratic, the order of the Bézier polynomial representation for $det(J)$ is higher than 3rd order and the total

number of control points is more than 20 consequently. The above stated algorithm can be generalized to apply to the higher-order cases which still focuses at monitoring the minimum value of all control points of the corresponding Bézier polynomials.

3.4.1 $\min\{P^{(3)}_{|i|}\}$ at an Interpolating Point

According to the convex hull property given by Equation 3.11, the minimum value among the control points is generally no greater than the actual minimum of the $det(J)$, i.e. $\min\{P^{(3)}_{|i|}\} \leq \min\{det(J)\}$. Also, it is known that the control points at the ends of a Bézier polynomial are interpolation points, which gives $det(J) = P^{(3)}_{|i|}$ at these particular control points. Therefore, in the cases that $\min\{P^{(3)}_{|i|}\}$ is found at a mesh vertex control point, which is indeed an interpolating point, the minimum determinant of Jacobian can be accurately obtained be to $\min\{det(J)\} = \min\{P^{(3)}_{|i|}\}$. In other words, if the value of $\min\{P^{(3)}_{|i|}\}$ is negative at any of the vertex control points, the uniform validity check method is no longer conservative and is able to effectively detect the invalidity.

3.4.2 $\min\{P^{(3)}_{|i|}\}$ at a Non-interpolating Point

In the cases where $\min\{P^{(3)}_{|i|}\}$ is found at a non-interpolating point, e.g. edge/face control point for a quadratic tetrahedron, the uniform validity check method becomes conservative for curved element geometry. Note that it is still accurate if the quadratic tetrahedron is straight-sided in which case all the control points are interpolation points. If $\min\{P^{(3)}_{|i|}\} \geq 0$ for a particular tetrahedron, it is sufficient to determine that the element is valid according to the condition discussed in Equation 3.12. However in the cases where $\min\{P^{(3)}_{|i|}\} \leq 0$, one could not necessarily conclude that the tetrahedron is invalid. In fact in some applications, such cases have been reported that negative $\min\{P^{(3)}_{|i|}\}$ are found for valid curved elements with large curvature, which means the uniform control point based validity check needs to be refined to deal with such cases.

3.5 The Adaptive Validity Check Methods

Both degree elevation and subdivision algorithms of a Bézier polynomial increases the number of control points and the control points converge to the actual polynomial [64, 19]. Thus either method can be used to obtain tighter bounds. Taking advantages of this property, two approaches to refine the uniform validity check method are proposed and studied in the following subsections.

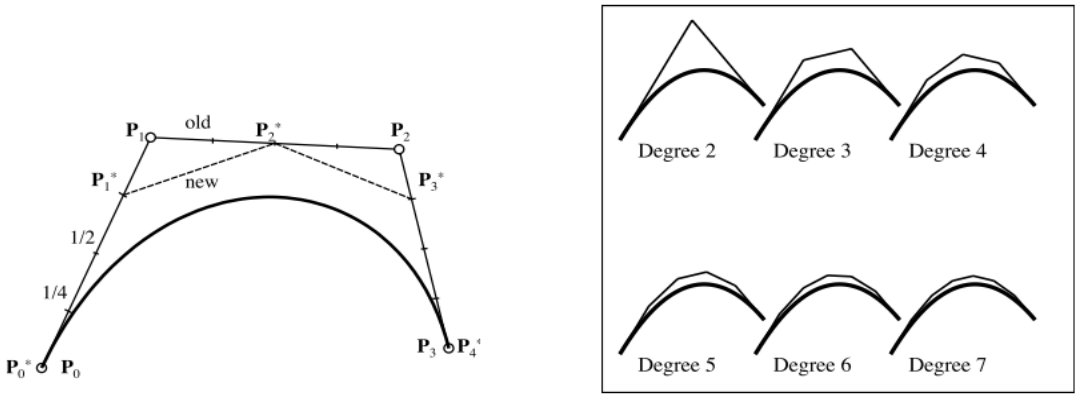
It is worth mentioning that although the uniform validity check becomes over conservative and loses accuracy in some cases, it is still an effective method to determine the key mesh entity with which the potential invalidity is associated. Given a curved tetrahedron of order p with $\min\{P_{|i|}^{(3(p-1)}\} < 0$ reported at a non-interpolating control point, the mesh entity associated with that particular control point is identified as the key mesh entity of possible invalidity.

By determining the key entity, we can avoid doing degree elevation or subdivision uniformly to all the element entities. Instead, only the key entity of interest is elevated or subdivided in an appropriate manner.

3.5.1 Adaptive Check Using Degree Elevation

After identifying a potentially invalid element and its key entity determined by doing the uniform validity check, a degree elevation check applies degree elevation algorithm to the key entity to refine the control polygon of its Bézier representation to give a tighter lower bound. For example, if $\min\{P_{|i|}^{(3(p-1)}\} < 0$ 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, i.e. $P_{|3000|}^{(3)}$, $P_{|2001|}^{(3)}$, $.P_{|1002|}^{(3)}$, $P_{|0003|}^{(3)}$. The control points after one step of degree elevation from 3rd- to 4th-order can be calculated by:

$$P_{|4000|}^{(4)} = P_{|3000|}^{(3)}$$
$$P_{|3001|}^{(4)} = \tfrac{1}{4}P_{|3000|}^{(3)} + \tfrac{3}{4}P_{|2001|}^{(3)}$$



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

Figure 3.7: Convergence of Degree Elevation

$$\begin{aligned} P^{(4)}_{|2002|} &= \tfrac{2}{4}P^{(3)}_{|2001|} + \tfrac{2}{4}P^{(3)}_{|1002|} \\ P^{(4)}_{|1003|} &= \tfrac{3}{4}P^{(3)}_{|1002|} + \tfrac{1}{4}P^{(3)}_{|0003|} \\ P^{(4)}_{|0004|} &= P^{(3)}_{|0003|} \end{aligned} \tag{3.13}$$

This can be generalized to obtain control points of any degree n elevated from degree $n-1$. According to [59, 49], the authors showed that the convergence rate is $O(\frac{1}{\nu})$, where ν is the polynomial order. A picture from [64] illustrates the convergence process of degree elevation to a 2D Bézier curve. See Figure 3.7.

As shown in the figure, 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.

3.5.2 Adaptive Check Using Subdivision

In addition to degree elevation algorithm, a subdivision algorithm can also produce more control points and tighter control polygon while maintaining the shape of the original Bézier polynomial. Take an edge as the key entity again, the check will subdivide the original 3rd-order Bézier polynomial associated with the edge as the addition of two 3rd-order sub-polynomials using the *de Casteljau algorithm* [19, 64]:

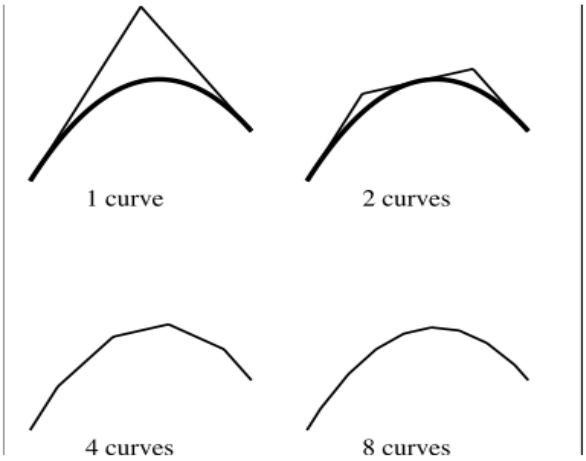


Figure 3.8: Convergence of Subdivision

$$P_0^1 = \tfrac{1}{2}P_{|3000|}^{(3)} + \tfrac{1}{2}P_{|2001|}^{(3)}, \quad P_1^1 = \tfrac{1}{2}P_{|2001|}^{(3)} + \tfrac{1}{2}P_{|1002|}^{(3)}, \quad P_2^1 = \tfrac{1}{2}P_{|1002|}^{(3)} + \tfrac{1}{2}P_{|0003|}^{(3)},$$
$$P_0^2 = \tfrac{1}{2}P_0^1 + \tfrac{1}{2}P_1^1, \qquad P_1^2 = \tfrac{1}{2}P_1^1 + \tfrac{1}{2}P_2^1,$$
$$P_0^3 = \tfrac{1}{2}P_0^2 + \tfrac{1}{2}P_1^2 \tag{3.14}$$

The new sets of Control points for the two sub-polynomials are then: $\{P_{|3000|}^{(3)}$, P_0^1, P_0^2, $P_0^3\}$ and $\{P_0^3$, P_1^2, P_2^1, $P_{|0003|}^{(3)}\}$. Note that P_1^1 is not used as a new control point.

It is also straight-forward to obtain more control points if one keeps doing subdivision recursively. And a picture from [64] gives an example of a 2D Bézier curve and its control polygons after several steps of subdivision. See Figure 3.8.

It is obvious that the control polygon gets closer to the curve after each step of subdivision, and according to [59, 49], it eventually converges to the curve with the rate of convergence $O(\frac{1}{2^i})$, where i is the number of subdivision steps.

3.5.3 The Stopping Criteria the Algorithm Description

The goal of the adaptive validity check algorithm is to effectively determine whether a given curved tetrahedron is a valid element. If $\min\{P_{|i|}^{(n)}\}$ is found to be positive, the element is valid. On the other hand, the element is invalid if negative

$\min\{P^{(n)}_{|i|}\}$ is found at any interpolating point during the checking process. In both cases, the algorithm will stop accordingly. However if negative $\min\{P^{(n)}_{|i|}\}$ appears at a non-interpolting point while no invalidity is found at all interpolating points during finite steps, it is also necessary to terminate the algorithm without doing infinite loops of checking and refinement. The current stopping criterion for such situation is based on evaluating 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.

It is worth noticing that the subdivision algorithm gives more interpolation points in addition to the vertex control points. Because after each step of subdivision, the original control polygon is divided into two parts and each part has its own interpolation points at the ends. See Figure 3.8 as an example. This gives an alternative to get the exact value of $det(J)$ at arbitrary parametric locations of mesh edges and faces by subdividing the corresponding control polygons at that location. This property could also serve as an addition to the stopping criterion. If any of the newly-computed interpolation 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.

A pseudocode description of the algorithm is given in Algorithm 1. The input is a quadratic curved tetrahedral mesh.

Data: A quadratic curved tetrahedral mesh, prescribed tolerance for relative increments ϵ

loop over the elements of the input mesh and process each of the elements;
for the current element to be processed, compute $\min\{P_{|i|}^{(n)}\}$;
if $\min\{P_{|i|}^{(n)}\} > 0$ **then**
    return: the element is VALID ;
else
    if *negative* $\min\{P_{|i|}^{(n)}\}$ *is at a interpolation point* **then**
        return: the element is VALID ;
    else
        while *relative increment* $> \epsilon$ **do**
            get the mesh entity associated with the negative control point ;
            apply subdivision to the mesh entity ;
            update the new $\min\{P_{|i|}^{(n)}\}$;
            if $\min\{P_{|i|}^{(n)}\} > 0$ **then**
                return: the element is VALID ;
            else
                if *new negative* $\min\{P_{|i|}^{(n)}\}$ *is at a interpolation point* **then**
                    return: the element is INVALID ;
                else
                    compute the relative increment of this step ;
                    if *relative increment* $< \epsilon$ **then**
                        return: the element is INVALID ;
                    else
                        update the new relative increment ;
                    end
                end
            end
        end
    end
end

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

Table 3.1: 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	E2_4	V3	E2_4	V3
Region2	0.5324e-3	2.4996	0.5324e-3	2.4996
Key entity	V4	V1	V4	V1
Region3	36.0230	176.0338	-0.2049	176.0338
Key entity	E2_4	V3	E2_4	V3
Region4	0.7574	13.5732	-0.1702	13.5732
Key entity	E2_4	V1	E2_4	V1

(V1: vertex 1; E2_4: Edge defined by vertex 2 and 4)

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

3.6 Numerical Results and Observations

A serial test mesh of around 45k tetrahedral elements including second-order curved tets was loaded and partitioned into 4 parts. Numerical experiments were done to study the 4 worst-shaped regions of the 4 parts (one on each part) reported by the 20-point validity check. In each case, a brute-force search was used to evaluate the exact value of $det(J)$ at a large number of sample points, and $\min\{det(J)\}$ was found so that we can compare the results with our check. For the ones with $\min\{P^{(3)}_{|i|}\}$ found at an edge, the refined adaptive checks using both degree elevation and subdivision algorithms were applied.

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 20-point validity check was consistent with the key entity found by the brute-force search of $\min\{det(J)\}$. This shows the effectiveness of the 20-point 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 consistent with what was discussed in previous section

since the control points at the vertices are interpolation points. Therefore when the invalidity ($\min\{det(J)\} \leq 0$) happens at a vertex, the 20-point 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 extended adaptive 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 [59, 49].

In terms of the efficiency of the extended adaptive checks, one could also estimate the additional cost by counting the flops required. Doing the original 20-point 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 based on the 20-point check, the added computational cost is not substantial at all.

A test case that compares the performance of the 20-point validity check and the one with adaptive subdivision was conducted on a test mesh with 60,390 curved elements. Four repeated runs were done for both check methods and the performance results in terms of mesh reading, writing and validity check are given in Table 3.2

[1] Note that this criterion is different from the one being discussed in the previous section 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.

Table 3.2: 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 3.3: 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%

and Table 3.3. Slight increase of computation time for the adaptive validity check is observed from 14.0% of the total time to 14.6%, which is small compared with the time spent in mesh reading and writing operations.

CHAPTER 4
Parallel High-order Curved Mesh Adaption

In this chapter, technical developments of parallel mesh adaptation procedures for partitioned high-order curved meshes are presented. Various operations of entity geometry modification and local mesh modification procedures are reviewed. Algorithms of high-order curved mesh adaptation are presented. Parallelization of those algorithms for partitioned high-order curved meshes is discussed.

4.1 Introduction

Adaptive finite element analysis using unstructured 3D meshes relies on the capability of locally changing the topology and/or geometry of a set of elements of the mesh, called mesh cavity, dictated by a mesh size field produced by *a posteriori* error estimation and indication procedures. One general approach to obtain such a mesh with the desired element sizes is to regenerate the mesh using size-controlled automatic mesh generation techniques. The drawbacks of this approach are 1) computationally expensive, and 2) require application of more expensive yet less accurate solution transfer procedures. An alternative, and more efficient, approach is to use local mesh modification procedures to refine and/or coarsen selected mesh cavities. Such procedures can be performed effectively in a desired order, and since the mesh modifications are local, the issue associated with local solution transfer can be effectively addressed [14, 23, 36, 37, 72].

4.2 Entity Geometry Modification Operations

The entity geometry modification operations focus on the geometric shape of mesh entities. It changes the shape of mesh edges and/or faces to curve the original straight-sided/planar entities, or relocates the mesh vertices. The topology of a local mesh cavity subject to such operations remains unchanged. In general, the entity geometry modification operations are used to 1) increase geometric approximation accuracy to the curved model domain, 2) eliminate mesh invalidity, and/or

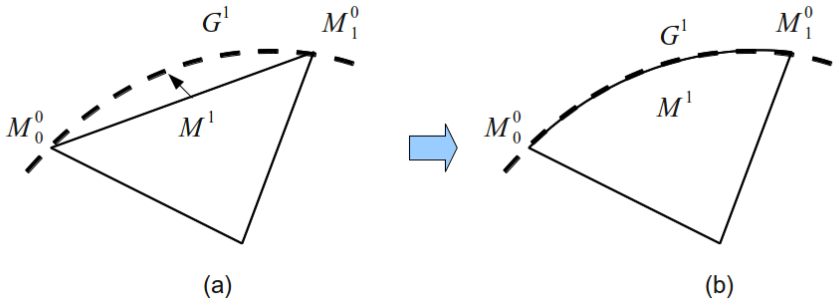


Figure 4.1: An example of curving a mesh edge to a model edge

3) improve mesh shape quality.

4.2.1 Curving Mesh Entities Classified on Curved Model Boundaries

The mesh curving operation modifies the geometry of a given mesh entity. One means to convert an initially linear geometry mesh entity to a curved one is the introduction of high-order nodes that are then positioned as desired. For mesh entities classified on curved model boundaries, their nodes are moved to the curved geometry to achieve a better geometric approximation to that geometry.

Dey et al present a mesh curving algorithm in [17] which interacts with geometric models through the underlying CAD modeling engine and calculates the proper position of a high-order node on its classified model boundary entity based on parametric interrogations.

For example, as shown in Figure 4.1, to properly curve a straight-sided mesh edge M^1 classified on a curved model edge G^1 using a quadratic Lagrange interpolation, a third point of the mesh edge has to be determined and placed properly onto the model edge. During the mesh generation process, if a mesh vertex is classified on a model edge or face, the parametric coordinates ζ of the mesh vertex with respect to the parametric space of its classified model entity are stored in the mesh data. If ζ_0 and ζ_1 are the parametric coordinates of the end vertices M_0^0 and M_1^0, their physical coordinates being $\mathbf{x_0}$ and $\mathbf{x_1}$ in the physical space, then by evaluating the mapping from parametric to physical coordinates $\mathbf{x}(\zeta)$ through the CAD modeler,

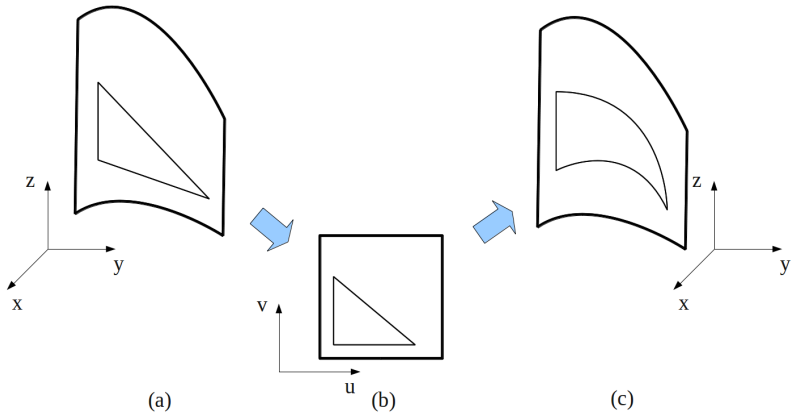


Figure 4.2: Curving boundary mesh entities by parametric interrogation. (a) is a straight-sided mesh face classified on a geometric model face in the physical space. (b) shows the mesh face in the 2D parametric space of the model face. The parametric coordinates of the edge mid-point are calculated in this space and given to the CAD modeler. (c) shows the mapping from the parametric coordinates to the physical space to get the Cartesian coordinates and the mesh face curved accordingly

the physical position of the mid-point of the mesh edge is obtained.

$$\mathbf{x_2} = \mathbf{x}(\frac{\zeta_0 + \zeta_1}{2}) \tag{4.1}$$

Therefore a high-order node associated with the mesh edge can be introduced and placed to that location. Also see Figure 4.2 for an example of curving a planar mesh face on a curved model face.

The curving of the boundary mesh entities may lead to self intersections which make elements invalid. In such occasions, either selected interior mesh entities are curved as well to correct the invalidity or other mesh modifications that eliminate the self intersections must be applied. For example in a 2D case shown in Figure 4.3, 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 (Figure 4.3(b)). In this case, the interior edges M_2^1 and M_3^1

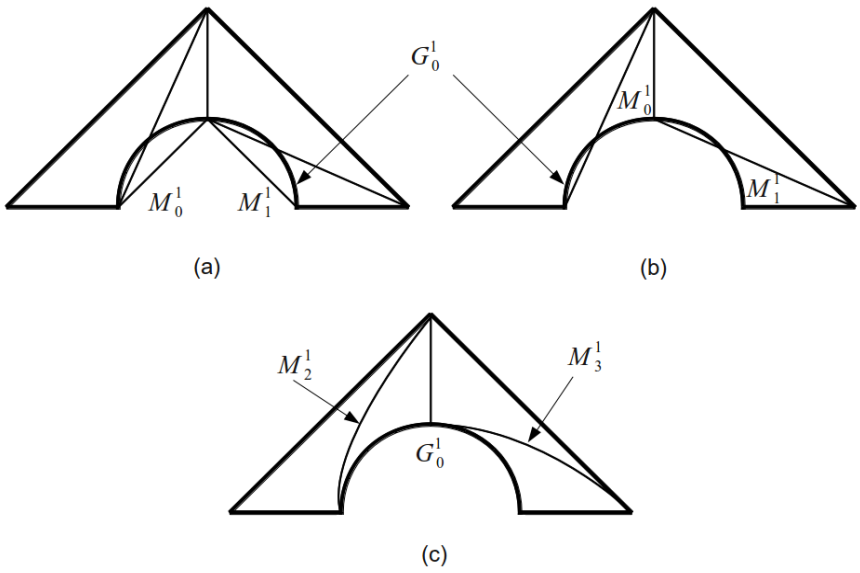


Figure 4.3: Curving interior mesh edges to resolve the invalidity caused by curving boundary edges

can be curved (Figure 4.3(c)) to ensure element validity. The procedures for such element curving are discussed in Section 4.2.2.2.

4.2.2 Reshaping Mesh Entities to Improve Mesh Quality

The mesh entity reshaping operation alters the local mesh geometry by relocating the position or changing the shape of selected mesh entities under specific mesh validity and boundary geometry constraints. It is typically used to improve element shape quality without altering the local mesh topology. The cavity of such an operation is defined as $M^i\{M^3\}$ depending on the dimension of the entity to be operated on. The possible mesh entity can be a vertex M^0, an edge M^1, or a face M^2.

4.2.2.1 Vertex Repositioning to Improve Straight-sided Element Quality

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. Generally speaking, determining the optimal location of a vertex to be repositioned is a constraint optimization problem in terms of a set of carefully selected objective functions. And there has been

extensive efforts devoted to develop various algorithms for the vertex repositioning operation, such as in [12, 21, 22, 34].

One of the algorithms which is fairly efficient and yields "better-looking" meshes is the Constraint Laplacian smoothing method. It moves the target vertex to the centroid of its cavity defined as $M^0\{M^3\}$ under the constraint of the geometric approximation. Assume there are n vertices in the cavity in addition to the vertex to be repositioned, the target location can be evaluated as:

$$\mathbf{x_0} = \frac{\sum_{i=1}^{n} \mathbf{x_i}}{n} \tag{4.2}$$

If the vertex to be repositioned is classified on a boundary entity of the geometric model, it is only allowed to move on the model boundary to ensure geometric approximation accuracy. In such cases, the computed centroid location needs to be projected back to the model entity.

Unfortunately this algorithm does not always improve the shape of some extremely poorly shaped elements [37]. In these cases an alternative vertex repositioning algorithm that guarantees a better overall mesh quality is the explicit vertex smoothing method. An effective design of such an algorithm has been introduced in [12]. However a major drawback of this algorithm is that it is computationally more demanding than the Laplacian smoothing approach. The algorithm is summarized in Algorithm 2:

Data: A list of straight-sided elements, shape quality threshold Q_{th}, maximum number of iterations allowed i_{max}

1 initialize an empty list that stores mesh vertices ;
2 traverse the list of elements and for each element compute the shape quality Q_s ;
3 **if** $Q_s < Q_{th}$ **then**
4 | put the four vertices of the element into the vertex list and avoid duplication;
5 **end**
6 traverse the vertex list and process each vertex in turn ;
7 evaluate the shape quality of the elements connected to the current vertex $M^0\{M^3\}$ and get the minimum value Q_{min} ;
8 find a direction of movement that will improve the shape quality of the element whose shape is Q_{min} ;
9 define an interval of uncertainty along this direction of movement ;
10 search the interval of uncertainty for a new location of the current vertex to move to, where the local maximum of Q_{min} of the cavity $M^0\{M^3\}$ can be reached;
11 **if** Q_{min} *can not be improved according to the search* **then**
12 | do not move the current vertex and proceed to the next one ;
13 **end**
14 repeat line 1 - 13 when the end of the vertex list is reached ;
15 **if** *the vertex list is empty* **then**
16 | exit the algorithm ;
17 **end**
18 terminate the algorithm after i_{max} iterations ;

Algorithm 2: An explicit vertex repositioning algorithm introduced in [12]

The first step of the explicit vertex repositioning algorithm is to create a list of candidate vertices to be processed. The vertices are selected from all the elements whose shape quality is below a given threshold Q_{th} (lines 1 - 5). The next step is to process the vertices one by one, and explicitly search for a local optimal location

for each vertex to move (lines 6 -13). This process involves the determination of a direction of movement and an interval of uncertainty for searching. In [12], the direction of movement is defined as a straight line by the original position of the vertex and the ideal position for the vertex to move to such that the most poorly shaped element it bounds is improved to its optimal shape. The interval for searching is defined as a segment along the direction of motion from the original position of the vertex to the position where the shape of any element of $M^0\{M^3\}$ drops below the original minimum value Q_{min}. A bisection or golden search is performed on the interval to find the local optimal position. Since this algorithm is targeting to improve the overall shape quality of a local mesh cavity, it is allowed to have the shape of shaped elements in a cavity degrade so long as the worst shaped element gets better and the overall quality get elevated.

Note that the vertex repositioning algorithm can be extended to deal with curved elements as well. However the shape of a curved element depends not only on its end vertices but also other shape parameters such as edge/face control points for a Bézier representation. Such additional shape parameter has to be considered when moving vertices of curved elements therefore increases the complexity of the optimization problem.

4.2.2.2 Curved Entity Reshaping to Improve Curved Element Quality

For a high-order curved mesh, the shapes 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. For example, the shape of a second-order curved mesh edge is uniquely determined when the position of its two end vertices plus a high-order node on the edge is fixed. Therefore the vertex repositioning operation itself is no longer sufficient to effectively manipulate the geometry of curved mesh entities. In this subsection, an entity reshaping algorithm (see Algorithm 3) is developed for the curved mesh edges and faces to improve the element shape quality of high-order curved meshes.

Data: A input mesh with curved elements, hybrid shape quality threshold Q_{th} and straight-sided shape quality threshold q_{th}

traverse the mesh and for each element compute the hybrid shape quality Q_{sc} by Equation 4.3 ;
if $Q_{sc} < Q_{th}$ **then**
    put the element into the list to be processed ;
end
traverse the element list and process each element in turn ;
compute straight-sided shape quality q_s ;
if $q_s < q_{th}$ **then**
    remove the element from the current list ;
    add the element to another list for straight-sided shape improvement procedures ;
else
    compute curved shape quality q_c by Equation 4.4 ;
    get $\min\{det(J(\xi))\}$ and $\max\{det(J(\xi))\}$;
    find the mesh edges associated with $\min\{det(J(\xi))\}$ and $\max\{det(J(\xi))\}$;
    for a candidate mesh edge, determine the line of motion ;
    define the interval of uncertainty for searching ;
    perform an explicit search algorithm such as golden search ;
    find the local optimal position to reshape the edge ;
end

Algorithm 3: An explicit entity reshaping algorithm

The input to the algorithm is a list of poorly-shaped curved tetrahedrons whose shapes are evaluated by the hybrid element quality metric

$$Q_{sc} = q_s^m \times q_c^n \tag{4.3}$$

discussed in Chapter 3, where q_s represents shape quality of the straight-sided part of a given tetrahedron and q_c represents the curved part. m and n are taken to be

1 for the sake of simplicity.

The algorithm processes the list of tetrahedrons as the following steps:

Step 1: Retrieve one tetrahedron from the list, and compute the corresponding shape quality measurement q_s and q_c respectively.

Note that, there are multiple choices for the shape metric q_s of a given straight-sided tetrahedron as discussed in Chapter 3. The explicit smoothing algorithm works independently of such choices. The shape metric q_c for a curved tetrahedron in this algorithm uses the scaled variations of the determinant of Jacobian over the element domain defined by Equation 3.7 in Chapter 3.

Step 2: Determine whether this tetrahedron should be considered for the curved entity reshaping operation. See lines 7 - 9 of Algorithm 3.

Given a shape quality threshold q_{th} for straight-sided element shape, if $q_s < q_{th}$, it indicates the straight-sided shape component is not acceptable under such a threshold, and it takes a higher priority to improve q_s first for the current element. Therefore, this tetrahedron will not be considered for the next steps. It will be removed from the current list and will be put into another list of tetrahedrons for a straight-sided-element shape improvement procedure. On the other hand, if $q_s \geq q_{th}$, it shows that the straight-sided shape component q_s of this region is good enough. Thus, considerations are given to improving the curved component of the shape quality q_c. And the algorithm continues with the next steps.

Step 3: Choose the candidate mesh entities to be reshaped. (See lines 11 - 13)

Once it is determined that the curved shape quality q_c is to be improved, it is critical to pick the proper candidate entity for reshaping. The curved shape quality q_c is defined in Chapter 3 as the following.

$$q_c = \frac{\min_{\xi \in \Omega^e} det(J(\xi))}{\max_{\xi \in \Omega^e} det(J(\xi))} \tag{4.4}$$

It is obvious that one should either increase $\min\{det(J(\xi))\}$ or decrease $\max\{det(J(\xi))\}$ in order for q_c to be improved. Therefore the candidate mesh entity should be the one(s) directly associated with the maximum or minimum value of $det(J(\xi))$.

By using the Bézier polynomial representation introduced in Chapter 2 for a

second-order curved tetrahedron, one can easily evaluate the value of $det(J(\xi))$ at 20 distinct control points associated with the mesh vertices, edges and faces, and get $\min\{det(J(\xi))\}$ and $\max\{det(J(\xi))\}$.

Depending on which control point the maximum or minimum is at, different candidate mesh entities are chosen. 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 $M^2\{M^1\}$.

Note that, the current algorithm only deals with the reshaping operation in the physical space, therefore the edges that are classified on model boundaries are not included as candidate edges since it involves the mapping of the entity shape between the physical space and the parametric space of model entities.

Step 4: Determine the line of motion and the interval of uncertainty (Lines 14 -15)

After having chosen the candidate mesh entity/entities to apply the reshaping operation, the next step is to determine how to reshape the entity. For the curved element shape metric q_c, the optimal value is always reached when a given tetrahedron is straight-sided, in which case, the value of $det(J)$ is a constant over the tetrahedron and $q_c = 1$. This indicates that for a given curved tetrahedron without further geometric constraint, it is always the best choice to reshape the curved entities back to be straight-sided. As an example in Figure 4.4(a), the optimal position for P to move to is P' so that M^1 becomes a straight-sided edge therefore q_c's for the two triangular elements reach 1.

However when reshaping a mesh entity of a curved tetrahedron with certain geometric constraints (for example some of the edges and/or faces of that tetrahedron are already curved because they are classified on curved geometric model boundaries), it is usually no longer the best choice to place the high-order node(s) of the curved entity to the mid-point position of its imaginary straight-sided counterpart. Instead, one needs to search for a local optimal location in a certain direction of motion. In order to efficiently find a local optimum, this algorithm limits the type

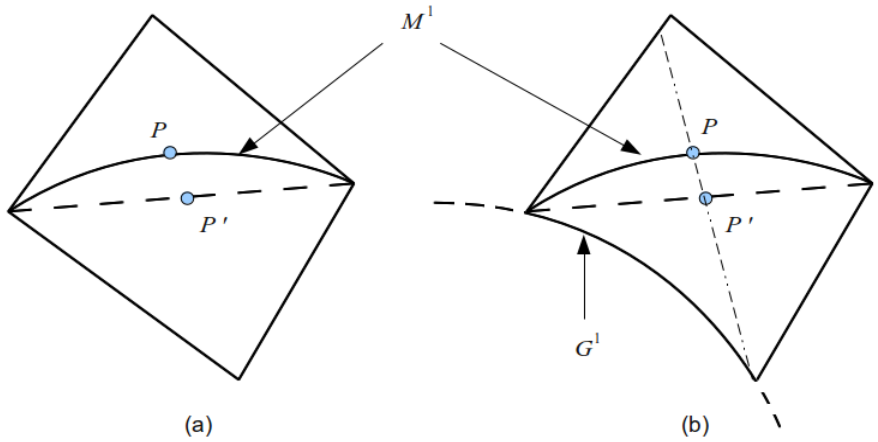


Figure 4.4: Two cases of 2D mesh edge reshaping. (a) without further geometric constraints, (b) one additional edge classified on model boundary G^1

of motion of the high-order node(s) to be on a straight line and consequently defines the line of motion by the current position of the high-order node to be moved and the mid-point position of the imaginary straight-sided entity.

$$\mathbf{r} = \mathbf{P} + (\mathbf{P} - \mathbf{P}') \cdot t \tag{4.5}$$

See Figure 4.4(b) for example.

As the high-order node moves along the line of motion to improve the shape quality of current tetrahedron, the shape quality of the neighboring tetrahedrons in the cavity changes, and is very likely to become worse at some point. The worst scenario is when the high-order node reaches a position that lies on another mesh entity, in which case, the current mesh entity associated with the high-order node will intersect with the other entity leading to mesh invalidity. Therefore the non-intersecting segment of the line of motion is considered as the interval of uncertainty for finding the local optimal position. For instance, the dash-dot line in Figure 4.4(b) is the interval of uncertainty for P to move along.

Step 5: Search the interval of uncertainty for the local optimal location. (Lines 16 -17)

With the interval of uncertainty being determined, one needs to find the op-

timal location on the line of motion for the high-order node to be moved to. As stated previously, reshaping a candidate mesh entity affects the shape quality of its neighboring tetrahedrons in the cavity, and it is very likely that the shape quality of a certain affected tetrahedrons will drop even below the lowest shape quality among the tetrahedrons of the original cavity. To avoid such situation from happening, the objective function for the search is picked to be the lowest shape quality of the tetrahedrons within the cavity defined by the entity to be reshaped. Therefore the local optimal location for the entity to be reshaped is where the lowest shape quality of the cavity is improved to the highest possible.

The golden section algorithm in [12] is used here to perform the search. This algorithm evaluates the objective function starting at the two ends of the interval of uncertainty. By comparing the values, about 38% of the interval is discarded each time and the rest serves as the interval of uncertainty for the next round of search. A piece of pseudo code is given to discribe the algorithm (see Algorithm 4).

Data: beginning and end of the initial interval of uncertainty P_0 and P_1, predefined tolerance ϵ

compute the length of the initial interval $l_0 = P_1 - P_0$;
get the two golden section points $P_2 = P1 - 0.61803 \times l_0$, $P_3 = P_0 + 0.61803 \times l_0$;
evaluate the objective function $f_2 = f(P_2); f_3 = f(P_3)$;
set the current length of interval $l = l_0$;
while $l/l_0 > \epsilon$ **do**
    if $f_2 > f_3$ **then**
        $P_1 = P_3$; $P_3 = P_2$; $f_3 = f_2$;
        $l = P_1 - P_0$;
        $P_2 = P_1 - 0.61803 \times l$;
        $f_2 = f(P_2)$;
    else
        $P_1 = P_2$; $P_2 = P_3$; $f_2 = f_3$;
        $l = P_1 - P_0$;
        $P_3 = P_0 + 0.61803 \times l$;
        $f_3 = f(P_3)$;
    end
end
if $f_2 > f_3$ **then**
    $P_{max} = P_2$;
else
    $P_{max} = P_3$;
end

Algorithm 4: Algorithm for the golden section search

4.3 Local Mesh Modification Operations for High-order Curved Meshes

In addition to purely geometric mesh modification operations, local mesh modification operations that change the local mesh connectivity as well as geometry have

been proved to be effective and efficient in mesh adaptation.

The local mesh modification operations used in this work consist of three unit operations: 1) splitting, 2) collapsing and 3) swapping, as well as several compound operations which combine the unit operators in an ordered sequence [23, 36, 37].

4.3.1 Split Operation

Typically, A split operator works on edges, faces and regions in three dimensional cases by inserting one or more new vertices to a target entity and subdivide the entity itself and the higher dimensional entities it bounds.

An edge split operation breaks an edge into two edges by introducing a new vertex to the middle of the edge, and also splits each of the connected higher order entities into two entities (See Figure 4.5). The new vertex inherits the classification of the split edge. A face split operation divides a face into three new faces. For 3D meshes, it divides each region bounded by the face into three new regions. To perform a face split, a new vertex is created inside the face. Three new faces are created by connecting the new vertex to two vertices of the original face in turn. If the face has tetrahedrons connected to it, each of the new faces is combined with the fourth vertex of the tetrahedron to form a new region (three new regions in total) [23]. See Figure 4.6 for an illustration. A region split divides a region into four new regions. The new regions are formed by each of the faces of the original element and the newly-created vertex. The newly created vertex can be classified only on the interior [23, 37]. See Figure 4.7.

In cases of edge/face split operations, if the target mesh edge or face (either straight-sided or curved) to be split is a boundary entity that approximates the curved geometric model boundary, the newly-created vertex should to be snapped onto the model boundary [36, 37]. The new edges and faces will also be curved accordingly to confirm to the curved geometry if high-order meshes are considered [17, 43]. If the geometric approximation is based on Lagrange interpolation points, the snapping and curving process are essentially the same in the sense that they both compute the mid-point(s) of a given mesh entity on model boundary. And the process is done by 1) retrieving the coordinates of the end points of the entity to

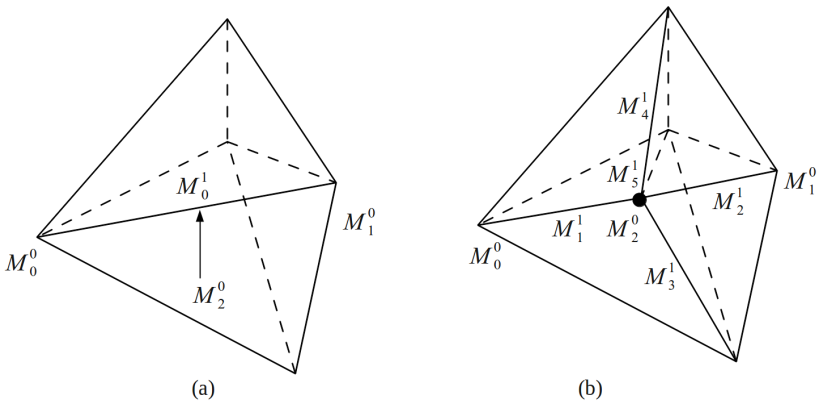


Figure 4.5: Split operator on a mesh edge. (a) M_0^1 is targeted to be split by introducing a new vertex M_2^0, (b) after edge split, new entities have been created

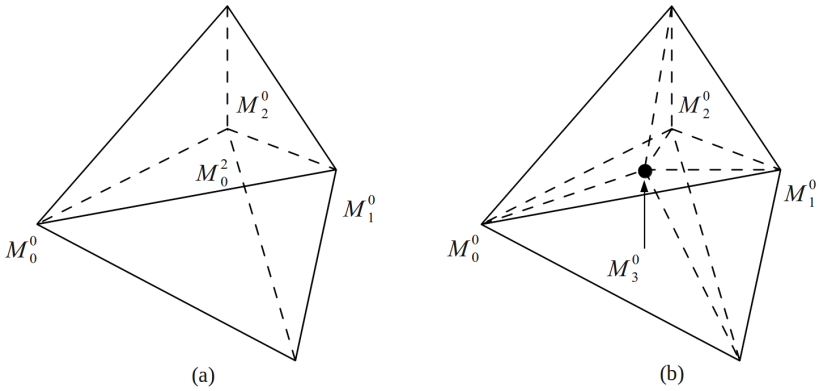


Figure 4.6: Split operator on a mesh face. (a) mesh face M_0^2 is targeted to be split, (b) a new vertex M_3^0 is introduced and M_0^2 has been split into 3 sub-faces, other new entities have been created accordingly

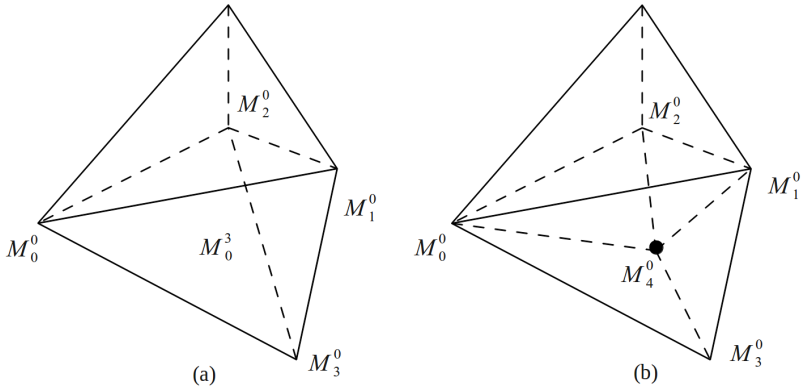


Figure 4.7: Split operator on a mesh region. (a) region M_0^3 is targeted to be split, (b) a new vertex M_4^0 is introduced and M_0^3 is split into 4 sub-regions

be split (or curved) in the parametric space that contains the geometric boundary entity, 2) computing the location of the splitting vertex (or high-order node) in that parametric space, and 3) mapping the location back to physical space.

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 discussed in Chapter 2. Based on the parametrization, the splitting process is essentially the same as splitting a boundary entity, and the geometry is preserved during the splitting.

4.3.2 Collapse Operation

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. For a collapse operation, a local mesh cavity is defined as the regions which are using either one of the two vertices of the target edge [37]. In Figure 4.8, vertex M_1^0 is collapsed to vertex M_0^0. It can be viewed as moving M_1^0 to the position of M_0^0. It is obvious that such movement will cause certain entities to overlap and certain entities to degenerate. By merging

the overlapping entities and eliminating the degenerated entities, the local mesh configuration becomes valid after collapsing.

Different from the split operation, a collapse operation is not always performable due to various topological constraints to ensure compatability. Therefore a set of procedures will be performed to check the constraints before such an operation is applied. As an example, if the two end vertices M_d^0 and M_r^0 belong to two different model faces, edge collapsing cannot be performed without violating the validity of the mesh. More details about the constraints and topological checks are discussed in [23, 37].

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 the same steps of parametric interrogations discussed in the previous section. 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. Details of the curving process is discussed in Section 4.2.2.2.

4.3.3 Swap Operation

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) $M_i^d\{M^3\}, d = 1, 2$. It is a reconnection procedure that effectively deletes the target entity and its connected elements and re-triangulates the polygon or polyhedron without the deleted entity [36, 37]. An example of the edge swap operation is given in Figure 4.9, edge M_0^1 is swapped to become a new edge M_1^1.

If an edge to be swapped is classified in a model region in 3D, the number of possible swap configurations can be determined by the number of regions the edge

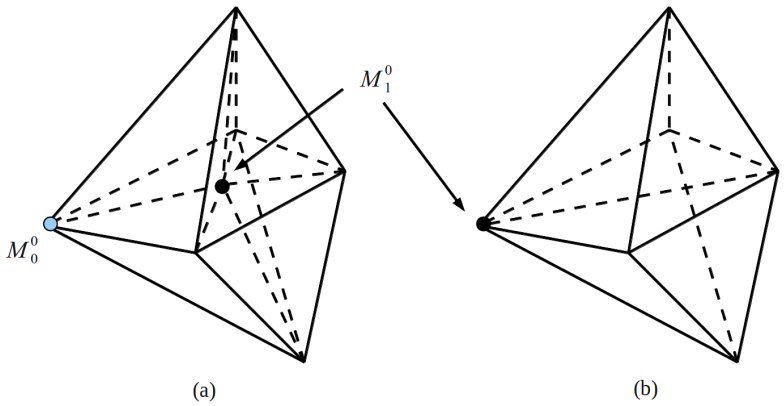


Figure 4.8: Example of an edge collapse operation. (a) before collapse, (b) after collapse

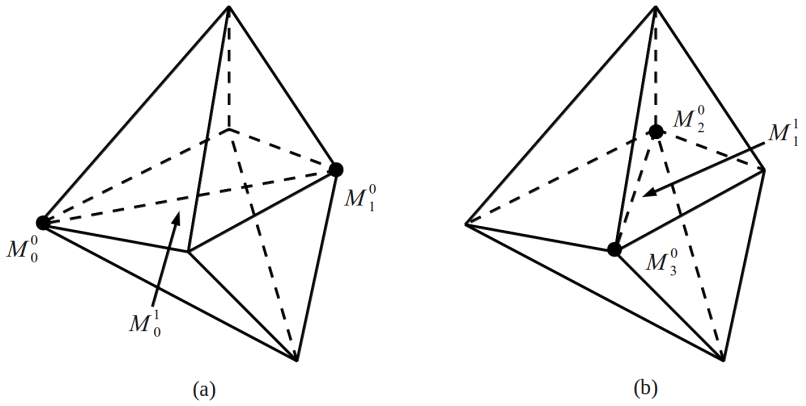


Figure 4.9: Example of an edge swap operation. (a) before swap, (b) after swap

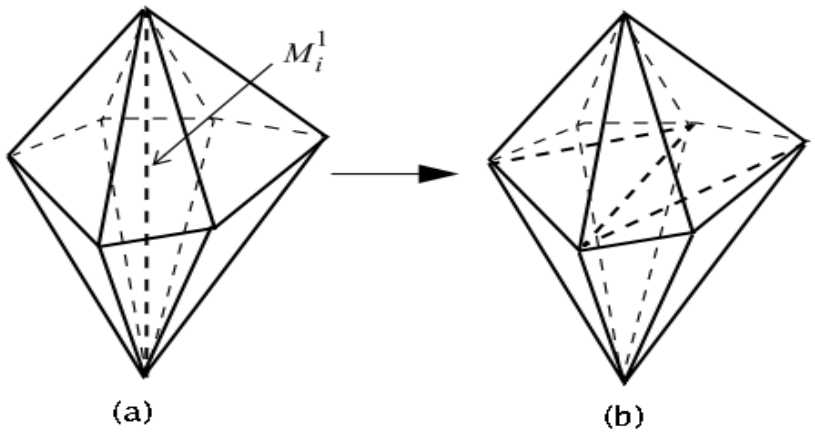


Figure 4.10: Another example of an edge swap operation

bounds ($M^1\{M^3\}$). Let n be the number of regions in cavity $M^1\{M^3\}$, formulas for the total number of possible swap configurations is given in [37, 23, 15] as

$$\begin{aligned} N_n &= \sum_{i=3}^{n} N_{i-1} \times N_{n+2-i}, n > 2 \\ N_2 &= 2 \end{aligned} \tag{4.6}$$

It is clear that when n increases, N_n increases dramatically, which leads to very expensive computational cost to determine from all the configurations. In [37], Li proposed to a limit for n for efficiency ($n \leq 6$ is used in current implementation).

Similar to the collapse operation, when swap operation is to be performed on high-order curved mesh entities classified on geometric model boundary, one should consider the new entity shapes after swapping to ensure that the mesh geometry still properly approximates the geometric model. Same as for curved edge collapse operation, various topological modification operations will be performed at first and the newly-introduced entities being classified on curved model boundary will be processed and curved properly after that. Same as for collapse, selected interior edges are curved accordingly if invalid self-intersecting elements are introduced by the newly curved boundary edges.

4.4 Compound Operations

There are cases that single unit local mesh modification operator is not capable of producing the desired mesh configuration. The compound operators are designed to deal with such cases by carefully combining a set of unit operators. Three compound operators found to be most useful [36, 37, 72] are:

- Double Split + Collapse

  Used for eliminating sliver tetrahedrons

- Split + Collapse

  Used for eliminating sliver triangular faces

- Collapse + Swap(s)

  Used in case that single edge collapse yields poorly shaped or even invalid tetrahedrons, swaps will be used to improve the local mesh quality

Typically, the compound operators are used to eliminate sliver tetrahedrons in 3D or sliver mesh faces in 2D. In some occasions, they are also needed to make space for vertex/nodal smoothing or boundary mesh curving procedure. The compound operations are not the focus of this work, nevertheless, interested readers could refer to [36, 37] in which more detailed discussions and use cases can be found.

4.5 High-order Curved Mesh Adaptation Strategy

The high-order curved mesh adaptation algorithm is built as an extension of the general straight-sided mesh adaptation procedure with specific invalidity correction and shape improvement procedures for curved meshes added.

4.5.1 Overall Procedure

Given an initial mesh of the domain and a desired mesh metric field defined over the domain, the goal of the curved mesh adaptation process is to produce a valid high-order curved mesh that satisfies the metric field while preserving the geometric approximation to the right order. The overall procedure consist of three

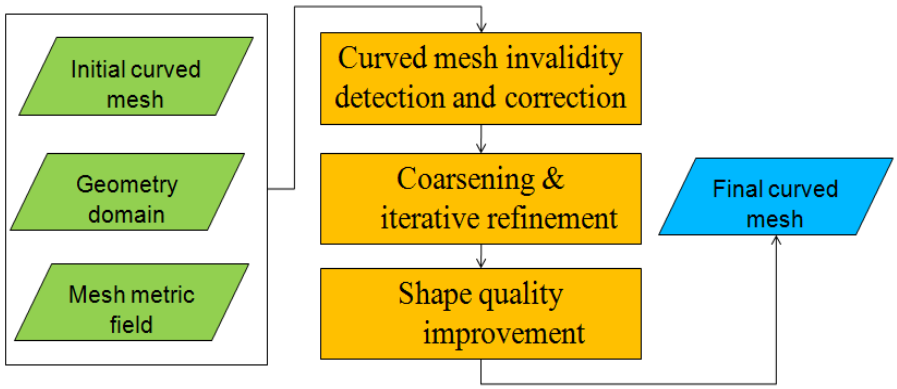


Figure 4.11: Overall procedure of high-order curved mesh adaptation

main stages (see Algorithm 5): 1) curved mesh invalidity correction (lines 1 - 4), 2) coarsening and iterative refinement (lines 5 - 8), and 3) shape quality improvement (lines 9 - 17). A flowchart is given in Figure 4.11.

Data: Initial curved mesh, geometry domain, and mesh metric field
Result: Adapted curved mesh satisfying the metric field
traverse mesh regions and create a list of all invalid elements;
while *the list is not empty* **do**
    eliminate the invalidity through curved mesh correction procedures;
end
traverse mesh edges and determine the edges with length shorter than L_{low};
perform coarsening algorithm to those edges;
traverse mesh edges and determine the edges with length longer than L_{up};
perform iterative refinement algorithm to those edges;
traverse mesh regions and create a list of regions that have shape quality $Q_{sc} < Q_{threshold}$;
while *there is still unprocessed region(s) in the list* **do**
    evaluate the best local mesh modification operations applicable to improve the shape of the region;
    if *no operation is applicable* **then**
        tag the region as processed;
    end
end
create a list of of the regions still with quality $Q_{sc} < Q_{threshold}$;
perform entity shape smoothing by geometry modifications to improve shape quality;

Algorithm 5: Overall algorithm of curved mesh adaptation

4.5.2 Invalidity Detection and Correction for the Initial Input Mesh

Due to the limitations of many current curved mesh generation techniques, there are often times that a curved mesh to be adapted has some invalid elements. Since the mesh adaption processes assume a valid initial mesh, such invalidity can undermine the proper execution of subsequent adaptation operations. Therefore it is critical to eliminate all invalid elements before performing mesh modifications to satisfy the mesh size field. The algorithm used here for curved mesh correction is

presented in Algorithm 6.

```
   input  : Initial curved mesh with invalid curved elements
   output: Corrected curved mesh without invalidity
 1 traverse mesh regions and create a list of all invalid elements;
 2 while there is still unprocessed invalid element(s) in the list do
 3 |   check if region collapse operation is applicable to eliminate invalid
   |   sliver regions at model boundary;
 4 |   check if edge collapse is applicable to eliminate invalid elements;
 5 |   check if edge/face swap is applicable;
 6 |   check if one of the compound operations such as double split plus
   |   collapse, collapse plus swap(s) is applicable;
 7 |   if all the above checks fail then
 8 |   |   perform local refinement and append newly created invalid
   |   |   elements in the list and tag them as processed;
 9 |   end
10 end
11 repeat the while loop for a number of iterations;
12 create a new list of the remaining invalid curved elements;
13 Uncurve the invalid curved elements to straight-sided elements;
```

Algorithm 6: Curved mesh correction algorithm

The order of the operations in the above algorithm is designed in consideration of cost effectiveness and likelihood of elimination of the invalidity under consideration. A operation will be applied as soon as it passes the checks and is determined to be applicable, therefore the cost to evaluate subsequent operations can be saved.

The algorithm first checks the unit operations of collapse and swap, and as discussed previously, swap operation generally has multiple possible configurations that requires additional efforts to evaluate. Therefore it is evaluated if collapse is not possible. The compound operations combine multiple unit operations, so the cost to evaluate increases consequently. They are examined if simple collapse or swap is not successful.

The local refinement step is of importance to eliminate topological and/or

geometric constraints of an invalid curved element, and gives more possibilities for the application of the other local mesh modification operations.

4.5.3 Coarsening and Iterative Refinement

Given a valid curved mesh, this stage is adopted from a straight-sided coarsening/refinement strategy introduced in [37, 38], with the addition of two aspects that account for curved meshes: 1) validity check for curved elements in the local cavities and 2) boundary mesh entity reshaping to conform to geometric model boundary as well as selective interior mesh curving to ensure validity. The coarsening with validity check and reshaping is given in Algorithm 7:

```
1  traverse mesh edges and create a list of edges whose length L(M_i^1) < L_low;
2  put the end vertices of the edges to a vertex list and avoid duplication;
3  traverse the vertex list;
4  while there is unprocessed vertex in the list do
5      get an unprocessed vertex M_i^0 and the shortest edge it bounds M_j^1;
6      evaluate the edge collapse of M_j^1 with M_i^0 removed;
7      if the local cavity defined by the edge collapse has new entities
       classified on curved model boundary then
8          curve the new entities to the model boundary;
9          check the validity of the curved elements in the cavity;
10         if invalidity is detected then
11             do not apply collapse operation;
12             tag the vertex as processed;
13         end
14     end
15 end
```

Algorithm 7: Algorithm of curved mesh coarsening

Since the coarsening process can create new mesh entities that are classified on curved geometric model boundary, it is necessary to curve those entities to the boundary. In general, this process can cause other connected curved elements to become invalid. Therefore curved element validity checks are used to detect potential

invalidity. The coarsening operation is not allowed to be apply if such invalidity happens.

The curved refinement algorithm used here is almost the same as discussed in [38] for the straight-sided meshes. It creates the new entity in a bottom-up fashion, i.e., edges first, then faces, regions last. The only difference happens in the cases that curved entities are to be refined. In such cases, the newly-introduced entities after refining the original curved entities should also be curved properly. This is done by calculating the coordinates of the new vertices and/or high-order nodes in the parametric space using either parametric interrogation through CAD modeler or the Bézier parameterization of high-order tetrahedrons discussed in Chapter 2. This not only ensures geometric approximation accuracy of curved boundary entities, but also rules out the possibility of introducing new invalidity to the mesh after the refinement stage.

4.5.4 Curved Element Quality Improvement

After getting a satisfactory mesh configuration conforming to the desired size field, this stage is performed to further improve mesh quality by curved local mesh modifications and explicit smoothing of the mesh entity geometry (see Algorithm 8).

input : Initial curved mesh with poorly-shaped curved elements, User specified quality threshold Q_{th}
output: Improved curved mesh
1 traverse mesh regions and create a list of elements with $Q_{sc} < Q_{th}$;
2 **while** *there is still unprocessed element(s) in the list* **do**
3 evaluate possible curved local mesh modification operations;
4 compare the worst shape quality among the elements of the local cavity before and after a specific local mesh modification operation;
5 **if** *there is at least one operation that improves the shape* **then**
6 apply the best operation that improves the worst shape quality to the highest;
7 append newly-created elements whose $Q_{sc} < Q_{th}$ to the list as unprocessed;
8 **end**
9 **if** *all the local mesh modification operations fail to improve the worst shape* **then**
10 perform explicit vertex/node smoothing operation;
11 append newly-created elements whose $Q_{sc} < Q_{th}$ to the list as unprocessed;
12 tag the current element at processed;
13 **end**
14 **end**

Algorithm 8: Curved mesh quality improvement algorithm

Note that in the evaluation of local mesh modification operations, the order is the same as in the mesh correction stage.

The explicit smoothing operation, as discussed previous section, is the most computationally demanding operation due to the golden search procedure to find the best location. Therefore, it is used at the end in the situation that all the other operations fail to apply. Nevertheless, it guarantees the elevation of shape quality of the local mesh cavity.

4.6 Parallelization

This section discusses the parallelization of the local mesh modification operations for distributed curved meshes. The developments are based on the basic parallelization process used for parallel straight-sided mesh modification procedures.

The key technical issues that have been addressed in parallelizing straight-sided mesh modification procedures are 1) evaluating and executing mesh modifications on or near partition boundaries and 2) the dynamic mesh load balancing. The Flexible distributed Mesh DataBase (FMDB) [20, 65] has been used to support the parallel operations of mesh entities including communications between mesh partitions, migration of mesh entities and the Zoltan [78] library is used to support dynamic load balancing.

In this section, the issue of effective evaluation and execution of curved mesh modification operations on partition boundaries is discussed. The dynamic load balancing is not considered here.

4.6.1 Parallel Curved Split Operation

In the case of splitting a curved mesh entity, the geometric shape of the target entity needs to be considered in addition to the topology, that is, 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, the strategy being used to keep the shared information consistent is that the owner copy of the target entity always makes the decision in terms of how to perform the split operation while the remote copies always follow the decision made by owner and accept the data sent from the owner copy.

Take the example of a quadratic curved edge shared by two mesh parts P0 and P1. If a split operation is to be performed by inserting a new vertex at the middle of the edge in the parametric space, the algorithm can be described in the following steps:

1. Calculate the x,y,z coordinates of the edge middle point for the new vertex to be introduced. See Figure 4.12(a).

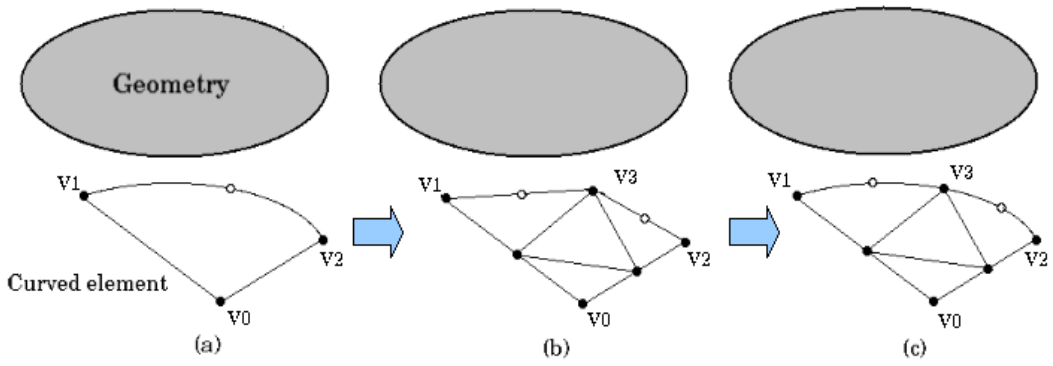


Figure 4.12: Curved edge split operation: (a) curved element to be split, (b) split the element uniformly and treat the newly created edges as straight-sided, (c) curve the new edges which are classified on curved geometric model boundary

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 Figure 4.12(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 Figure 4.12(c).

4. Synchronize the remote copies with the owner across the parts to update the high-order node information. See Figure 4.13.

4.6.2 Parallel Curved Collapse and Swap

If the target edge or either one of its end vertices is on a 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. FMDB [20] supports such migration operations by calling the migration callback functions defined for high-order curved entities.

The idea of extending the edge swap operator to deal with parallel high-order 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

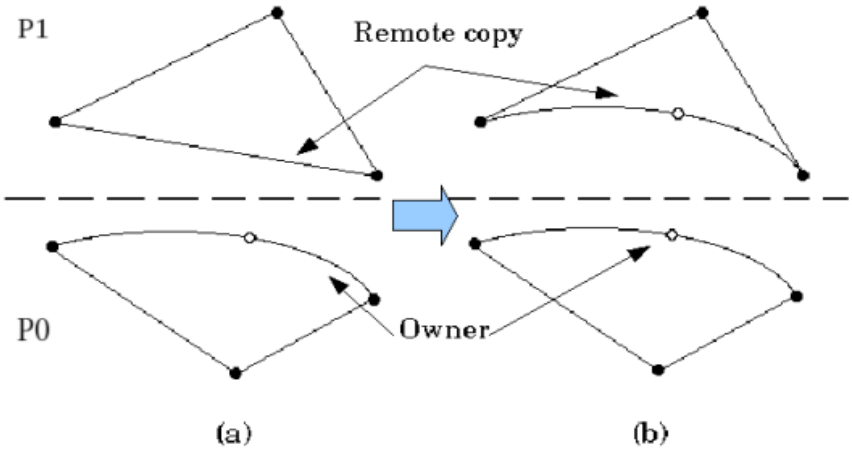


Figure 4.13: Synchronize curved entities on partition boundary: (a) before sync, owner determines the target location of edge middle point. (b) after sync, remote copies receive date from owner and update their local copy

newly created entities, and (ii) curved entity migrations will be needed for the cases that the cavity is across partition boundaries.

The overall algorithm for parallel curved edge collapse and swap operations can be summarized into three general 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.

Figure 4.14 demonstrates a 2D example of a combination of parallel mesh curving and curved edge swap operations.

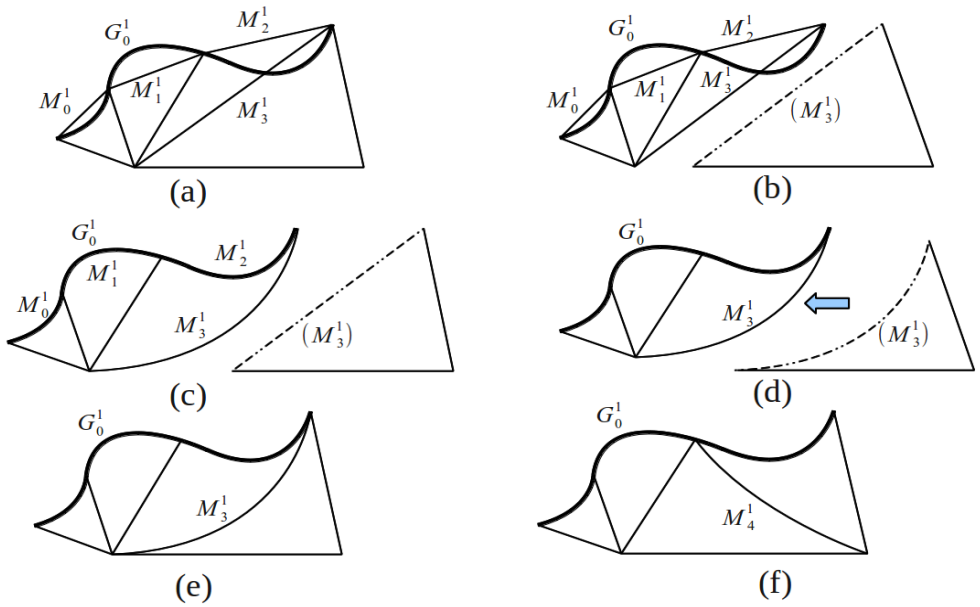


Figure 4.14: A 2D example of parallel mesh curving and curved edge swap operation (a) an initial straight-sided 2D mesh, (b) the mesh is distributed to two parts, (c) boundary mesh edges M_0^1, M_1^1, M_2^1 are curved to conform to the geometric model edge G_0^1, interior mesh edge M_3^1 is curved to avoid element invalidity, (d) the remote copy of M_3^1 is synchronized by its owner to also be curved, and to further 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

4.6.3 Parallel Curved Compound Operations

The three compound operations discussed in previous section – Double Split plus Collapse (DSPC), Split Plus Collapse (SPC) and Collapse plus Swap(s) – have also been parallelized for partitioned curved meshes. Since the operations simply chains the split, collapse and swap operations together, the steps are therefore performed together in an ordered sequence. Note that collapse operation requires mesh migration to construct local cavity, therefore migration is performed for DSPC and SPC in advance of splitting the mesh edges.

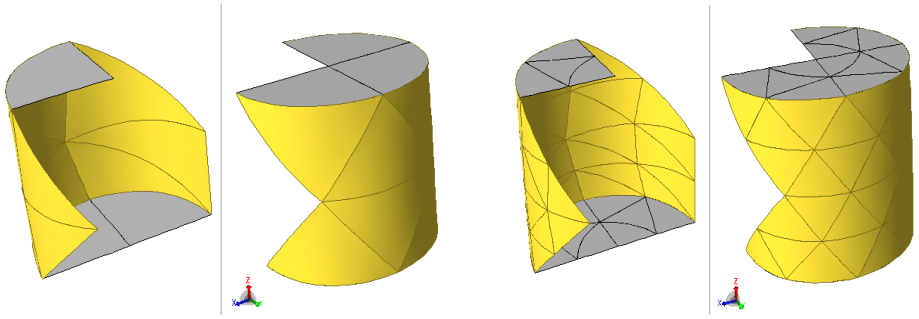


Figure 4.15: An example of parallel curved split operations on a two part distributed mesh: (left) before refinement, (right) after refinement

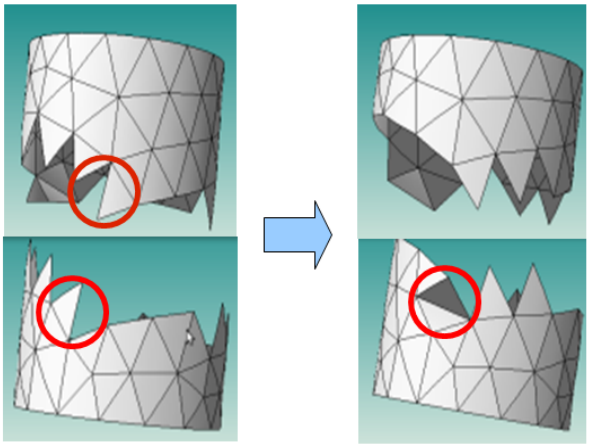


Figure 4.16: An example of parallel curved edge swap on a two part mesh

4.6.4 Examples Meshes

Parallel curved mesh examples are presented in this section to demonstrate the capability of the curved local mesh modification and adaptation procedures.

Figure 4.15 gives an example of parallel curved split operations over a 2-part distributed mesh. Parallel curved edge split, face split and region split operations have been exercised extensively in this example.

Figure 4.16 shows an example of the unit operation of parallel curved edge swap. The edge to be swapped is on a partition boundary shared by two partitions. Therefore the local mesh cavity defined the edge is migrated before the curved edge swap operation is performed within a local partition.

Figure 4.17 shows a parallel curved mesh refinement process. The geometry

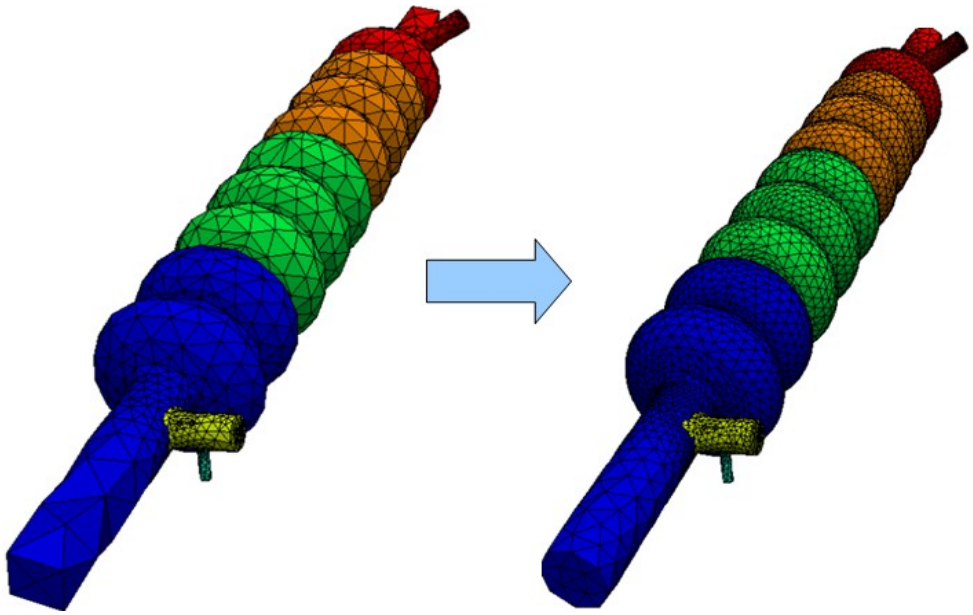


Figure 4.17: An example of parallel curved mesh refinement on a four part mesh

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.

CHAPTER 5
Application: Parallel Automatic Adaptive Accelerator Simulations

The Advanced Computations Department (ACD) of the SLAC National Accelerator Laboratory (SLAC) is developing a new generation of high-order finite element procedures (ACE3P) [1] for accelerator simulations that have demonstrated the ability to accurately model a variety of accelerator problems. Due to the nature of the ACE3P solvers, it requires properly designed high-order finite element discretizations using curved elements [44]. Moreover, the level of discretization (mesh size) to obtain reliable predictions in ACE3P simulations often requires meshes with upwards of hundreds of millions of elements, therefore massively parallel computing technologies are of critical importance.

Since available finite element meshing software does not properly deal with meshes with curved elements and there is very limited parallel meshing tools even for linear straight-sided meshes, SCOREC in collaboration with Simmetrix 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.

5.1 Desired Workflow for Automatic Adaptive Simulations

The desired workflow for an adaptive simulation starts with a definition of the problem domain of interest. In accelerator design the best domain definition is a solid model constructed in a CAD system. The various analysis attributes (loads, boundary conditions, material properties) are best specified with respect the solid model. Initial mesh control attributes that guide the mesh generation process can also be specified with respect to the solid model [7, 8, 68]. Based on the CAD model and attributes, a suitable discretization – an initial mesh – can be generated with desired geometric approximation accuracy to the model.

In case of a parallel simulation, load balancing is performed to maintain balanced distribution of work loads among the multiple processes thus ensuring the

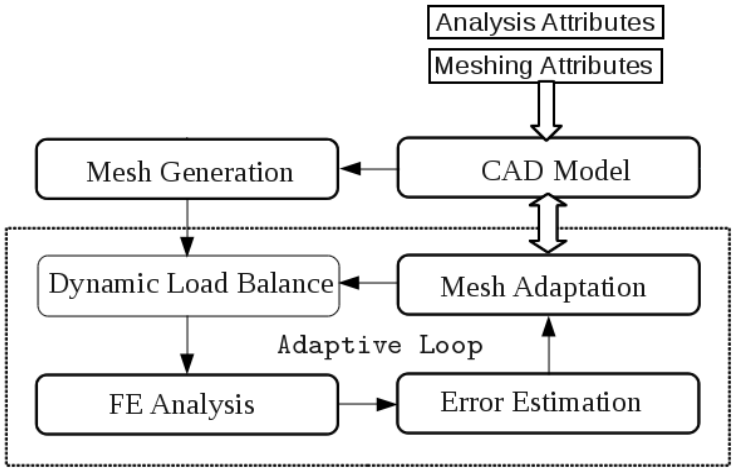


Figure 5.1: An schematic of the desired workflow for parallel adaptive simulations

efficiency of the workflow. The finite element analysis procedures compute the solution fields of interest. To adaptively improve solution accuracy, error estimation and correction indication procedures are used to extract information from the computed solution fields and that information, in the form of a mesh size-field, is used to drive the mesh adaptation procedures to obtain a better discretization – an adapted mesh. After dynamically balancing the work loads, 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 desired solution accuracy is achieved. Finally the results of the solution fields can be post-processed and visualized.

Figure 5.1 shows an schematic of the desired workflow.

5.2 Contributing Components

The interacting functional components for such an adaptive simulation workflow are:

- geometry modeling engines,
- attributes management component,
- mesh generation software,

- finite element analysis procedure,
- error estimation and correction indication component,
- mesh adaptation procedure with local solution transfer,
- dynamic load balancing procedure.

5.2.1 Desired Features

The application of this workflow on massively parallel computing systems requires that each individual component operates in parallel and can interact with parallel representations of information (such as partitioned mesh) in a consistent manner. A robust parallel mesh generator is essential to produce valid initial curved meshes in parallel used as input for the parallel FEA procedures. The components of dynamic load balancing, parallel FEA, error estimation, and parallel mesh adaptation come together as an integrated loop of the adaptive simulation workflow, as shown in the dash-line box in Figure 5.1. Note that to properly represent the geometry, the parallel mesh adaptation procedure must interact with the CAD model in parallel to perform basic geometric queries. A user-friendly graphical interface is necessary for attributes management and specification.

To effectively integrate the various components together as a complete workflow, a set of interoperable interface functions is needed for the components that provide consistent interactions with the mesh and solution data being carried on during the simulation process. Such interactions are desired to be carried out, ultimately, through the internal data structures that do not require inter-component file I/O. The design and implementation of such interfaces requires that all components provide runtime access to their information, especially the meshing and solution components.

The subsections that follow introduce pieces of software that have been, or continue to be, developed to support the individual functional components.

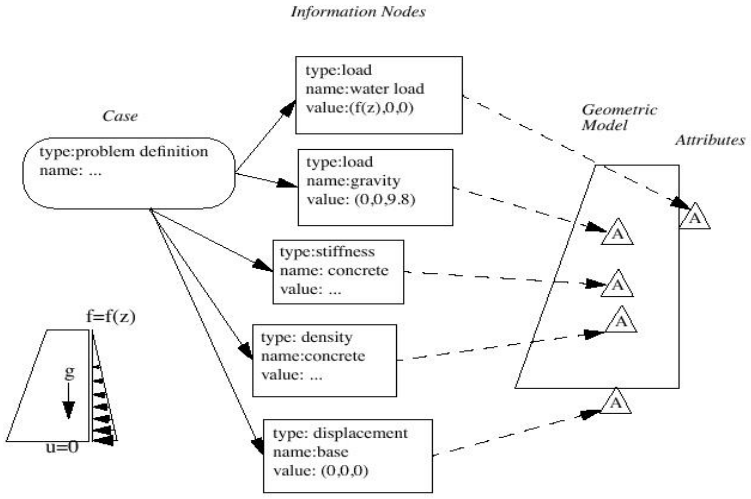


Figure 5.2: Simmetrix attribute management [8]

5.2.2 Geometry

As mentioned in section 5.1, the geometry of the problem domain is best defined by a solid model constructed in CAD systems. There are several CAD kernels that are widely used in solid modeling, such as ACIS, Parasolid, OpenCascade, with ACIS and Parasolid being used by popular commercial CAD modeling systems such as SolidWorks [70], NX [53], and also used in specific DOE tools (e.g. the CUBIT meshing system employs ACIS for geometry).

5.2.3 Attributes

To fully define an analysis problem, additional information is needed besides the geometric domain such as loads, material properties, boundary conditions and the like. Such information is defined as attributes to the CAD model [7, 8]

CUBIT [11] is a DOE mesh generation toolkit developed by Sandia National Lab (Sandia) that interacts with the ACIS kernel. It supports a way of specifying analysis attributes such as material properties and boundary conditions to the CAD model by grouping model faces and regions as “sidesets” and “blocks” and labeling the groups with reference IDs. These IDs are then associated with specific analysis attributes defined separately from CUBIT.

The Simmetrix Simulation Modeling Suite (SimModSuite) [69] consists of a set

of components that also have capabilities of geometry access, simulation attribute management as well as mesh generation and adaptation. GeomSim is one of the components which integrates with multiple CAD kernels (e.g. ACIS, Parasolid, OpenCascade) and provides a unified access to geometry and topology of CAD models. Although it does not support model construction, it is capable of loading given CAD models and supporting query geometric and topological information about the models as well as automated tools for geometry clean-up in preparation for meshing.

SimModSuite also provides a component named GeomSim Attributes that introduces an object-oriented way of specifying and managing attributes directly off the CAD model. Analysis attributes in GeomSim Attributes are managed in two levels. An abstract level defines the concept and properties of various types of attributes as informational nodes, e.g. displacement as a first order tensor. When assigning an attribute to a model entity, such as displacement of a model surface, an instance of the abstract node is created with specific functions [7]. Figure 5.2 gives an illustration of the structure. Such an object-orient way is intuitive and general, and the abstraction of attributes can be easily customized or extended.

5.2.4 Meshing

CUBIT has unstructured initial mesh generation in serial with straight-sided elements [24, 25]. It also supports curving those tetrahedrons in serial [11]. Together with the ACIS modeling capability, CUBIT has been utilized by the SLAC ACD team to generate curved initial meshes for ACE3P simulations. Figure 5.3 shows a simple example. However, invalid elements are often found in the curved initial meshes that harm the stability and efficiency of the ACE3P solvers. Additional efforts are required to fix the mesh invalidities.

The automatic mesh generation component of SimModSuite, named MeshSim [69], is able to generate valid curved meshes for general non-manifold geometric models. It fully supports the generation of valid meshes with curved quadratic elements in serial. Parallel generation and adaptation of distributed straight-sided meshes is currently supported and with the extensions needed for curved meshes

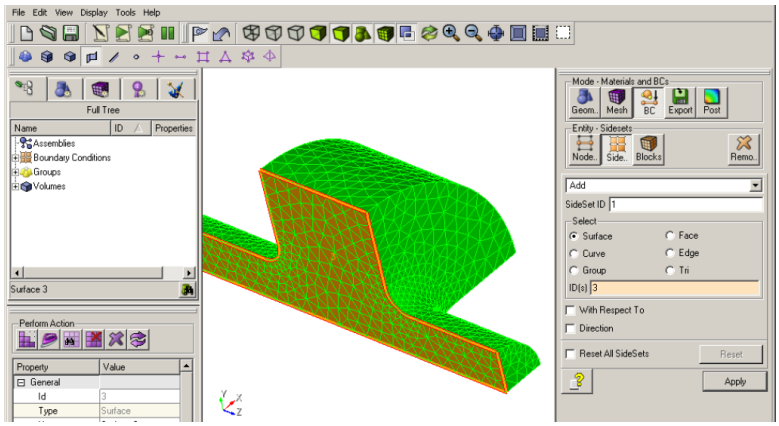


Figure 5.3: Screenshot of an example mesh generated by CUBIT

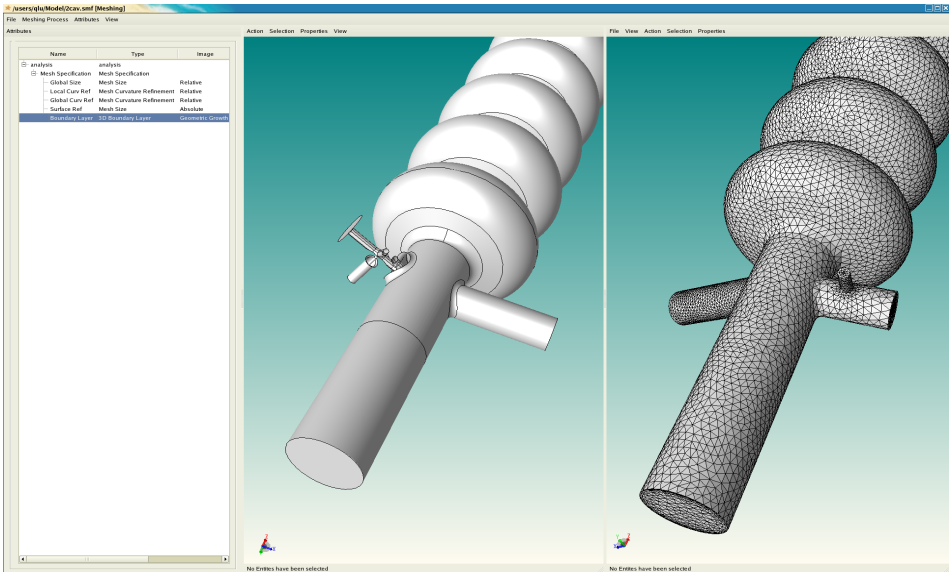


Figure 5.4: Screenshot of the Simmetrix GUI showing the attribute definitions on the left, CAD model in the middle and mesh of it on the right

under development. It also has the capability to create structured boundary layer meshes with mixed topology elements. The input data that MeshSim supports can be CAD models, image data or discrete (mesh) models. Figure 5.4 shows an example of the Simmetrix graphical user interface (GUI) that demonstrates the geometry, attributes and mesh.

5.2.5 Finite Element Analysis Procedure

The ACE3P is SLAC's suite of 3D parallel finite element based electromagnetic simulation codes for accelerator modeling [1]. It contains a set of solvers that perform time and frequency domain calculations, particle tracking and simulations as well as multiphysics simulations.

The list of ACE3P simulation codes includes [1]:

- Omega3P – Eigenvalue solver for finding the normal modes in an RF cavity
- S3P – S-parameter solver to calculate the transmission properties of open structures
- T3P – Time-domain solver to calculate transient response of driven fields and beam excitations
- Track3P – Particle tracking code with surface physics
- Pic3P – Particle-in-cell code to simulate self-consistent electrodynamics of charged particles
- TEM3P – Multi-physics module for integrated electromagnetic, thermal, and mechanical analysis

Implemented to work on parallel computing environment, the ACE3P suite has demonstrated state of the art capability for accelerator simulations and produced results of great interest to accelerator designs [35, 52].

5.2.6 Error Estimation and Correction Indication

Simmetrix provides a component, named FieldSim, for error estimation based on application-specified solution fields. The error estimation procedure supported is based on the superconvergent patch recovery method (SPR) which has been applied to electromagnetic analysis [76, 77]. Alternative error estimation procedures have also been considered and are easily introduced.

SCOREC has similar error estimation procedure based on the SPR method. In additional to that, a Hessian based strategy for error estimation has been developed

to extract directional information of error distribution [61]. The method obtains the directional information by computing the field's second derivatives and convert that information into a mesh metric field which prescribes the element size and orientation for adaptation [62, 63].

5.2.7 Mesh Adaptation

SCOREC and Simmetrix both have been developing mesh adaptation procedures driven by an anisotropic mesh sizefield. The SCOREC MeshAdapt software [47] adapts an unstructured mesh using various geometry modification and local mesh modification operations until it satisfies the prescribed mesh metric field. The metric field to be satisfied can be either isotropic or anisotropic [23, 38]. Both serial and parallel mesh adaptation procedures have the capability to work with meshes with curved quadratic-shaped elements. Detailed discussions about the parallel curved mesh adaptation strategy and operations can be found in Chapter 4. A version of MeshAdapt also supports the adaptation of structured layers of mesh elements at model boundary called boundary layer meshes [62, 63].

The Simmetrix mesh adaptation software components, MeshSim Adapt and Parallel MeshSim Adapt, also work with isotropic and anisotropic mesh metric fields. MeshSim Adapt allows the adaptive refinement and coarsening of meshes created by MeshSim in serial while Parallel MeshSim Adapt allows adaptive refinement of partitioned meshes in parallel. MeshSim Adapt also supports serial boundary layer mesh adaptivity for wedge element boundary layers [69]. Parallel versions of boundary layer and curved mesh adaptation are under development.

5.2.8 Dynamic Load Balancing

Massively parallel adaptive simulations at large core counts require that the mesh be distributed across the processors with equally balanced workload to ensure efficiency and scalability of all steps of the analyses.

A number of algorithms and software packages have been developed for unstructured mesh partitioning. The most popular approaches by far are the graph- or hypergraph-based algorithms. Zoltan [78] is a software package being developed at Sandia that employs the graph/hypergraph-based partitioning algorithms. However,

specific issues are observed for parallel applications at extreme scale. For example, typical graph/hypergraph-based partitioners only consider balancing the workload based on mesh elements (or regions), which may result in vertex imbalance and consequently imbalance of degrees of freedom (DOFs) for the FEA solver. Study shows such imbalance can be as high as 20% [75].

To address the issues, ParMA [55], a parallel iterative mesh partition improvement algorithm, is being developed to account for load balance of all mesh entities of interest. By migrating selected mesh entities from relatively heavily loaded parts, with respect to mesh entities of interest, to less loaded neighboring parts using the mesh adjacency information, ParMA can effectively eliminate heavily loaded parts (referred to as spikes) and maintain a balanced workload [75]. Results have demonstrated the ability of ParMA operating to large meshes with billions of mesh regions on massively parallel systems with more than 100,000 processors [74].

5.3 Current CUBIT-based Workflow

To both support the current user base, and be able to move forward to the desired adaptive simulation loop, two workflows are being developed simultaneously. One is based on the serial meshing capability of the CUBIT mesh generator currently utilized by SLAC, and the other focuses on taking advantage of the parallel meshing and curved mesh capabilities of Simmetrix MeshSim.

The ACIS geometry modeling kernel with the CUBIT CAE interface and mesh generation toolkit have been used by SLAC, and a CUBIT based workflow is in place to support ACE3P simulations. The workflow can be summarized as the following steps:

1. geometric model read into or constructed in CUBIT,
2. attribute management and serial initial straight-sided mesh generation is executed using the tetrahedral meshes [24] in CUBIT,
3. the straight-sided mesh has midside nodes added and moved to the CAD geometry,

4. curved element invalidity detection and correction is executed using a version of the SCOREC MeshAdapt software,

5. FE analysis using ACE3P solvers.

A prototype adaptive loop has been developed by the SCOREC team in a component-based manner for this workflow [46]. To supplement the existing steps, key developments have been made to incorporate the error estimation and size driven (isotropic) mesh adaptation procedures into the workflow to form a complete loop. The error estimation procedure was based on the superconvergent patch recovery method [76, 77]. The mesh adaptation was based on the well-established straight-sided mesh adaptation procedures [37, 38] and was extended to deal with curved meshes [46].

5.3.1 Initial Curved Mesh Generation And Element Invalidity Issue

In the applications of the CUBIT-based workflow, several issues have been observed and incremental developments have been made to address the issues and support the workflow. One of the issues has to do with the initial curved mesh generation and element invalidity.

The CUBIT meshing capability does not guarantee generation of valid curvilinear meshes to meet the basic requirements of ACE3P solvers (or any other solvers). Moveover, in some cases that a given CAD model is too complex and contains undesirable small feature, CUBIT fails to generate any meshes at all. SCOREC has worked with SLAC and developed a curved mesh validity check and invalidity correction procedure to effectively detect mesh invalidity and fix the invalid curved elements [45]. The validity check applies the Bézier representation introduced in Chapter 2 and the specific methods are discussed in Chapter 3. The mesh correction procedure uses various mesh modification operations discussed in Chapter 4. The procedure is implemented as part of the SCOREC MeshAdapt. It has been applied to ensure the creation of valid curvilinear meshes for the ACE3P applications and helps to obtain stable simulations with improved efficiency. See Figure 5.5 for an example.

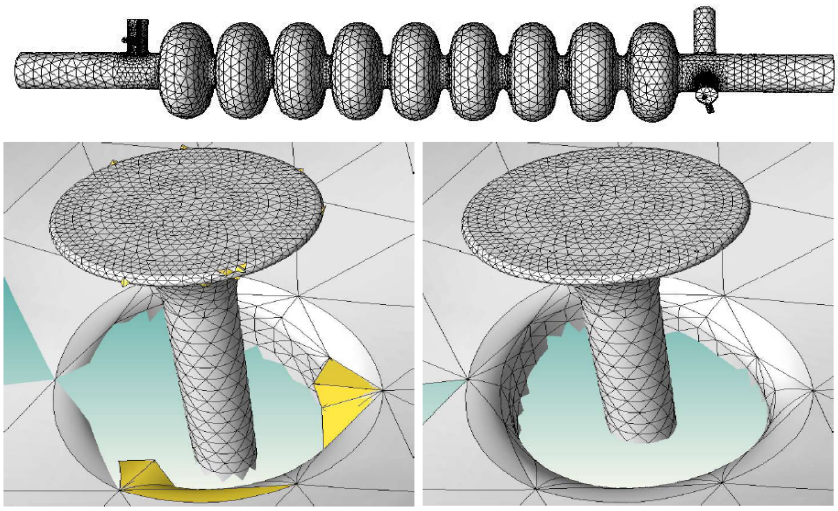


Figure 5.5: Curved mesh for an accelerator cavity, close up mesh before and after applying invalid element correction procedure [45]

Besides the combination of CUBIT mesh generation capability and SCOREC mesh correction procedure, an alternative and more straight-forward approach to obtain a valid curved initial mesh is to use the Simmetrix software components. As introduced previously, the Simmetrix MeshSim is a software component that is capable of effectively generating curved initial meshes from a given CAD geometry and it guarantees the validity of the generated meshes. Integrated with GeomSim, which is able to clean up small features in the CAD model to ensure successful mesh generation, it is the only working approach that has worked for the cases when CUBIT fails to generate any meshes from some complex CAD models with undesirable small features. The use of Simmetrix software instead of CUBIT results in a more reliable workflow which is discussed for a fully parallel version in Section 5.4.

Tests have shown the effectiveness and efficiency of Simmetrix curved mesh generation capability on various complex CAD geometries. Figure 5.6 shows an example CAD model of a complex geometry. CUBIT fails to generate initial meshes from the model while Simmetrix is able to automatically clean up the undesirable feature of the model and successfully generate valid curved meshes.

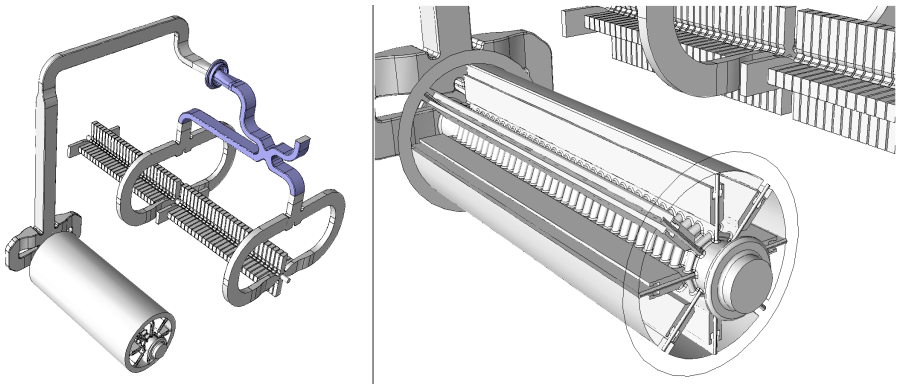


Figure 5.6: A complex geometry with undesirable small features simulated by ACE3P. (Left) an overview. (Right) close-up view of some small features

5.3.2 Additional Incremental Improvements to the CUBIT-based Workflow

Although the aforementioned issues of serial curved meshing and element invalidity can be resolved, such a workflow still has major drawbacks that prevents the application from going to massively parallel systems. The main limiting factor is that it is based on the serial meshing capability. The total number of elements in an initial serial mesh is limited by hardware constraints of a single computer, such as memory limits. Thus it can not reach the scale of hundreds of millions of elements required by some ACE3P simulations.

Section 5.4 discusses the efforts on the development of a fully parallel workflow in which interoperable component operates in parallel. Since that workflow is not yet fully in place, and there is a need to maintain the CUBIT-based workflow for the current user community, several incremental developments have been carried out. The first development is a procedure that partitions a serial curved mesh into distributed mesh parts and effectively increase the total number of elements by mesh refinement steps in parallel. In this procedure, the SCOREC Flexible distributed Mesh DataBase (FMDB) [20] is used to serve as the underlying mesh database to support various mesh based operations such as entity creation, deletion and migration. The mesh partitioning and load balancing are supported by the Zoltan and ParMETIS packages.

In the partitioning process, the shape parameters of a curved mesh entity

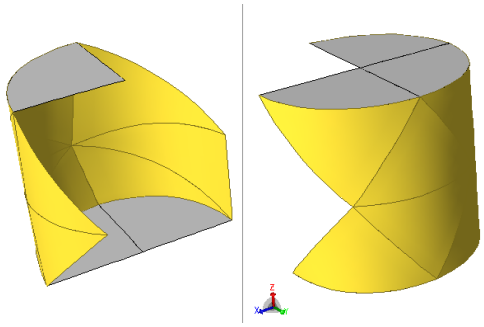


Figure 5.7: Curved mesh parts after partitioning

(such as high-order nodes or edge mid-points) are retrieved and attached to the entity itself. During the migration process of the curved entity, the attached data is migrated along with the edge from one processor to another. See Figure 5.7 for an example of the surface mesh for a curved partitioned mesh.

After partitioning the serial curved mesh, refinement steps are performed to increase the number of mesh entities simultaneously on each mesh part to reach the desired amount. The SCOREC MeshAdapt is used to perform the parallel refinement. For the simplest case that no size field information or boundary layer structure is prescribed, the most effective way is to uniformly refine the mesh.

To support the mesh refinement process, the parallel curved split operations are used based on the algorithm described in Section 4.6.1. If the geometric modeling engine is available, the new vertices and new edges on curved model boundaries are reshaped to be placed on the appropriate model entity according to the results of parametric interrogations to the CAD model. The algorithm in this case is: (Figure 4.12 illustrates the curving process.)

1. Get the coordinates of vertices V1, V2, V3 in parametric space by querying the modeling engine,

2. compute the parametric coordinates of the mid-points to be added as the average of the two vertex parametric values,

3. map the parametric coordinates back to coordinates in Cartesian space, using a query to the modeler,

4. attach the mid-point coordinates to the edges.

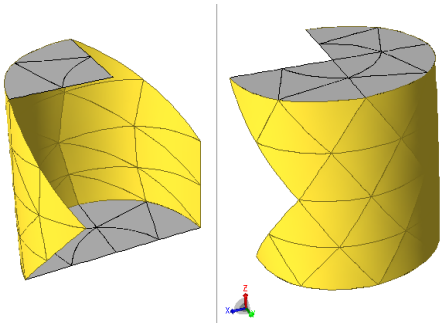


Figure 5.8: Partitioned curved mesh after uniform refinement

If the geometric model is not available the shape information of the original mesh edge is used. The algorithm is defined as the following: (One can still refer to Figure 4.12(b) and (c).)

1. Get the Cartesian coordinates of vertices V1, V2, V3,
2. fit a parabola to V1, V2, V3 by Lagrange interpolating polynomials, and parameterize the curve to 1D space [0, 1],
3. compute the Cartesian coordinates at parametric value 0.25 and 0.75, which corresponds to the desired location of the mid-points of the new edges
4. attach the mid-point coordinates to the new edges.

Although both of the algorithms are implemented, the current workflow runs with the latter version due to the fact that common CAD modeling engines are normally not available on most of the parallel computing platforms. As soon as the modeling engines become available, the latter algorithm can be easily replaced by the former one. Figure 5.8 shows the surface of an example mesh after parallel uniform refinement.

The mesh refinement procedure in MeshAdapt also serves as an important component in supporting the request from SLAC to conduct convergence studies of ACE3P analysis results. As a mesh generation software, CUBIT does not provide mesh adaptation capability, one has to re-generating new meshes from scratch every time when finer meshes are needed, which is time consuming and inefficient. By integrating the mesh refinement procedure into the workflow, refined meshes can

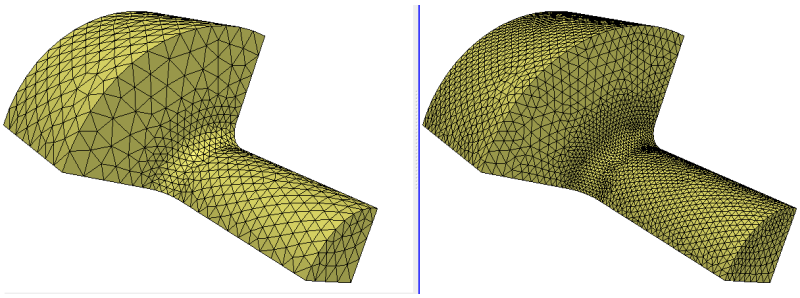


Figure 5.9: An example of the meshes generated by uniform refinement for convergence study

be created based on existing coarse meshes, and a convergence study loop can be executed incrementally and automatically. Figure 5.9 gives an example of the coarse and refined meshes generated for convergence study.

5.3.3 Meshing Results

The serial initial mesh based workflow has been applied in several cases to support the ACE3P simulations. Several example meshes are shown in previous sections of this Chapter.

Figure 5.10 gives an additional example mesh generated over a geometry of two linear accelerator cavities. The largest application so far has been a partitioned curved mesh generated on 64 processors with more than 180 million curved tetrahedral elements. The initial serial curved mesh for this application has around 3 million elements and is partitioned to 64 parts. Two steps of uniform refinement result in the final mesh.

5.4 Fully Parallel Workflow

The desired starting point of the fully parallel work flow is a CAD model of the accelerator domain to be analyzed with the analysis attributes of loads material properties and boundary conditions specified on that entities on that CAD model. From there all steps of the simulation, starting with the generation of the initial mesh, must be executed automatically, including adaptive error control, on massively parallel computing systems. The ability to construct such a fully parallel workflow requires each component operate in parallel, can interoperate with the

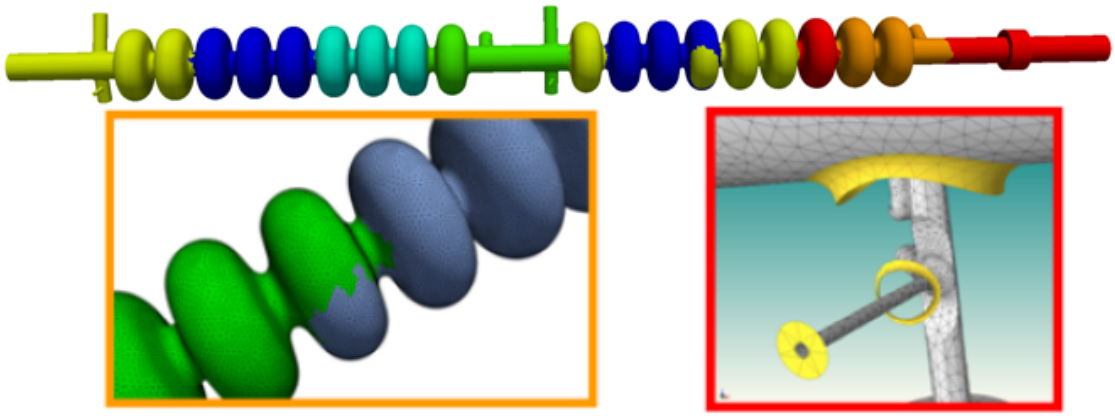


Figure 5.10: Overview and close-ups of a partitioned curved mesh of linear accelerator cavities

other components in a consistent manner, and all components share a consistent view of the parallel distribution of the information being worked on. The optimization of the such a parallel workflow also requires having the greatest level of scalability of the full set of interacting components and elimination of key unproductive bottlenecks.

From the point of the initial mesh generation through all adaptive analysis processes, the key information structure is the mesh. The parallel representation of the mesh as a partition of parts is common to all these components and is supported by the ITAPS iMeshP partition model specification [31]. Both Simmetrix [69] and FMDB [20] implementations support the needs of this partition model. Efforts are just underway to investigate making it a two level partition model to support future high core count systems in which the intra-compute node level would be threaded and the inter-compute node level would be message passing. Scalability of the entire process requires the repeated application of dynamic load balancing as the mesh and computations evolve. Currently Zoltan [78] is used when there large load imbalances and ParMA [55] is used eliminate small imbalances and/or to account for multiple criteria such as driven by the multiple steps of the analysis process. Investigation is ongoing to see if the ParMA, which is very fast, can be extended to do complete repartitions and still maintain a sizable speed advantage.

One obvious bottleneck in the construction of parallel adaptive simulation

processes is when the interactions between components employ file I/O. The more efficient alternative is to provide the information exchange between components by directly extracting the information from the component data structures. Since functional interfaces have been used in integrating components developed in this work, even when just dealing with files, the overall approach is already in place to eliminate unneeded file I/O steps. Doing this does require a more direct interaction and access to the structures within the analysis components.

The efforts carried out to date have produced important steps toward the construction of a fully parallel work flow. Parallel versions of all needed components are either available, or will be by the end of 2011. However, specific additional efforts are needed to actually put all the pieces together. These efforts are planned as part of the future work jointly with Simmetrix.

A summary of these efforts and the components involved is as follows:

A full set of geometry-based analysis attributes can be applied directly to the ACIS, or other CAD models using the Simmetrix attribute specification tools. A consistent linkage between the analysis attributes and the model, and between the model and the mesh are maintained. The graphical user interface that supports the specification of the analysis attributes can also be used to quickly and easily set mesh control parameters to be followed by the initial mesh generation procedure.

Initial parallel mesh generation of curved meshes of any desired numbers of elements will be executed by MeshSim. In addition to the extension of the Simmetrix parallel mesh generator to produce valid curved element meshes, Simmetrix is working on parallelization of aspects of the geometric model for the future cases when it may be desirable to also distribute the geometric model.

As already indicated, Zoltan and ParMA will be used to support dynamic load balancing.

Given an attributed mesh that is load balanced based on the needs of the ACE3P simulation component to be executed, that analysis component can be executed in parallel to the highest possible scalability.

Once the analysis is completed the needed simulation results can be loaded into a field structure (either Simmetrix FieldSim and SCOREC iFields) to make the

information accessible for discretization error estimation and correction indication. Currently both projection based error estimators and Hessian-based anisotropic correction indication procedures have been developed and used in various applications. The results of the parallel error estimation and/or correction procedures are used to specify the desired mesh size field over the current mesh.

Both SCOREC MeshAdapt and Simmetrix MeshSim Adapt operate in parallel and will take a current mesh and associated mesh size field and perform the mesh modifications needed to have the mesh satisfy the requested mesh size field, which can be fully anisotropic [3, 38]. MeshAdapt fully supports these operations on quadratic order curved meshes and Simmetrix MeshSim will also support quadratic order curved meshes by the end of 2011.

The development of this fully parallel workflow will be executed in two passes. In the first pass the interactions between the Simmetrix and SCOREC components will be through API's interacting directly with data structures while the interactions with the ACE3P components will be using their existing file interfaces. On the second pass we plan to work with the ACE3P developers to interact directly with analysis component data structures thus eliminating the unneeded file interactions.

CHAPTER 6
Discussion and Conclusions

This thesis has presented techniques to construct controlled curved meshes of millions of elements for use with high-order finite elements methods. This chapter summarizes the work done to achieve that goal and suggests directions for future works.

6.1 Conclusions

In this thesis, we have reviewed the mathematical fundamentals of the Bézier polynomial based shape representation for high-order curved tetrahedrons. The Bézier representation provides an easy way mapping the parametric space to the physical space. It also supports the effective evaluation of the determinant Jacobian, which is the key factor for checking mesh validity.

Based on the Bézier shape representation, A hybrid shape metric has been proposed to serve as a unified shape measure for both linear straight-sided elements and high-order curved elements. The proposed metric takes the product of a selected straight-sided metric with a curved metric. In this way, it is capable of capturing all forms of badly-shaped elements, both straight-sided and curved. This metric serves as the basis for the element validity checks as well as mesh modification operations, such as the explicit entity reshaping operation introduced in this work.

Efficient methods for curved element validity checking have been developed based on the Bézier shape representation and its control points. The adaptive methods developed utilizes degree elevation or entity subdivision operations, to adaptively refine the control polygon of the Bézier shape representation. Numerical experiments have shown results that they are effective with almost neglegible addition to the computational cost over the uniform validity check method.

Technical parallel curved meshing techniques have been presented in terms of various mesh modification operations and overall mesh adaptation strategies. The mesh modification operations include two major categories: entity geometry mod-

ification and local mesh modification for curved meshes. For the entity geometry modification operations, efforts have been devoted to develop entity reshape operations to explicitly resolve element invalidity and improve the shape quality of curved elements. The local mesh modification operations for curved meshes are extended from the operations for straight-sided meshes with additional considerations and treatments to curve boundary entities and selected interior entities to ensure geometric approximation to the model as well as element validity. The overall curved mesh adaptation strategy is built on the existing strategy for straight-sided mesh adaptation. Specific procedures such as validity check and shape improvement for curved elements have been added. The above mentioned operations have been made to work with partitioned curved meshes in parallel in order to support the application of mesh based parallel high-order finite element simulations.

Applications of the parallel curved meshing technique has been under development to support the automated adaptive accelerator simulations at SLAC National Accelerator Laboratory (SLAC). A desired workflow for parallel automated adaptive simulations has been proposed. Various functional components, such as parallel curved mesh generation and adaptation, have been discussed. In order to support the current user community, efforts have been made to build a workflow utilizing the DOE mesh generation software CUBIT. Several issues associated with such a CUBIT-based workflow have been identified such as element invalidity and lack of mesh adaptation capability. Incremental developments have been made to resolve those issues. In the meantime, in order to support large scale simulations, a fully parallel workflow for SLAC is being developed. The fully parallel workflow is designed to take full advantages of the state-of-the-art techniques, such as parallel mesh generation and adaptation, automatic geometry clean-up, dynamic load balancing. The various functional components of the workflow are designed to interact with each other through component data structures to, in the future, eliminate the major bottleneck in the parallel adaptive simulation processes – file I/O. Both the current CUBIT-based workflow and desired fully parallel workflow are supporting the accelerator simulations conducted at SLAC in a more reliable and efficient way.

6.2 Future Work

Future developments of the work presented in the thesis include:

- Studying the mathematical properties of the proposed hybrid shape metric. As summarized in Section 6.1, the multiplicative hybrid shape metric is capable of capture the characteristics of poorly shaped straight-sided and curved elements. Further analysis and numerical experiments could be conducted to reveal quantitatively how sensitive the metric is with respect to certain kinds of poorly shaped curved elements such as sliver and close-to-self-intersecting curved elements. Such a sensitivity study could provide insight to selecting the most effective mesh modification operation to improve element shape quality.

- Incorporating size field driven adaptation to partitioned curved meshes based on error estimation of analysis results. For an automatic adaptive simulation, a mesh sizefield obtained by the error estimation and indication procedures is the essential guide to the mesh adaptation procedure. The current implementation of the parallel adaptive mesh adaptation for curved meshes is capable of performing tag driven refinement or adaptation driven by prescribed size field based on the motion of domain of interest [44]. Therefore an error estimator needs to be incorporated. An interface needs to be developed between the error estimator and the mesh adaptation driver. The parallel curved mesh adaptation has to be able to support mesh modification operations based on the mesh sizefield provided by the error estimator.

- Continuing the development of the CUBIT-based and the fully parallel workflows for the accelerator simulations at SLAC. As stated before, the CUBIT-based workflow has an existing user community therefore it is important to continue to support the users. However the fully parallel workflow is more promising in terms of reliability and scalability for further large scale simulations. The future developments include developing missing functional components such as parallel curved mesh generator and parallel error estimation procedure, incorporating dynamic load balancing procedure into the parallel

adaptive loop, and integrating the various components through interfaces that employ fileless communications.

LITERATURE CITED

[1] Advanced Computations Department. "ACE3P (Advanced Computational Electromagnetics 3P) Code Suite." SLAC National Accelerator Laboratory, Menlo Park, CA. [online]. Available from *https://slacportal.slac.stanford.edu/sites/ard_public/acd/Pages/Default.aspx*. (Date last accessed: November 23, 2011)

[2] "ACIS 3D Modeler". Spatial Corp. [online]. Available from *http://www.spatial.com/products/3d-acis-modeling*. (Date last accessed: November 2, 2011)

[3] F. Alauzet, X. Li, E. S. Seol, and M. S. Shephard. "Parallel Anisotropic 3D Mesh Adaptation by Mesh Modification." *Engineering with Computers*. Vol 21, No 3. (2006): 247-258.

[4] M. Anisworth and J. T. Oden. *A Posteriori Error Estimation in Finite Flement Analysis*. New York, NY: John Wiley & Sons, Inc. September 2000.

[5] I. Babuska, B. A. Szabo and I. N. Katz. "The p-Version of the Finite Element Method." *SIAM Journal on Numerical Analysis*. Vol 18, No 3. (1981): 515-545.

[6] M. W. Beall and M. S. Shephard. "A General Topology-based Mesh Data Structure." *International Journal for Numerical Methods in Engineering*. Vol 40, Issue 9. (1997): 1573-1596.

[7] M. W. Beall and M. S. Shephard. "An Object-Oriented Framework for Reliable Numerical Simulations." *Engineering with Computers*. Vol 15, No. 1. (1999): 61-72.

[8] M. W. Beall. "An Object-Oriented Framework for the Reliable Automated Solution fo Problems in Mathematical Physics." *PhD Thesis*. Rensselaer Polytechnic Institute, Troy, NY. 1999.

[9] "Bézier Curve." Wikipedia. [online]. Available from *http://en.wikipedia.org/wiki/Bézier_curve.* (Date last accessed: October 28, 2011)

[10] M. Bucki, C. Lobos, Y. Payan and N. Hitschfeld. "Jacobian-based Repair Method for Finite Element Meshes after Registration." *Engineering with Computers*. Vol 27, No 3. (2010): 285-297.

[11] "CUBIT Geometry and Mesh Generation Toolkit". Sandia National Laboratory, Albuquerque, NM. [online]. Available from *http://cubit.sandia.gov/.* (Date last accessed: November 21, 2011)

[12] H. L. de Cougny, M. S. Shephard and M. K. Georges. "Explicit Node Point Mesh Smoothing within the Octree Mesh Generator." *SCOREC Report #10-1990.* Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY. 1990.

[13] H. L. de Cougny and M. S. Shephard. "Surface Meshing Using Vertex Insertion." *Proceedings of the 5th International Meshing Roundtable.* Pittsburgh, PA. (1996): 243-256.

[14] H. L. de Cougny and M. S. Shephard. "Parallel Refinement and Coarsening of Tetrahedral Meshes." *International Journal for Numerical Methods in Engineering.* Vol 46, Issue 7. (1999): 1101-1125.

[15] E. B. de lIsle and P. L. George. "Optimization of Tetrahedral Meshes." *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, eds. Ivo Babuska et al.* Berlin, Germany: Springer. (1993): 97-128.

[16] S. Dey. "Geometry-Based Three Dimensional hp Finite Element Modeling and Computations." *PhD Thesis.* Rensselaer Polytechnic Institute, Troy, NY. 1997.

[17] S. Dey, R. M. O'Bara and M. S. Shephard. "Curvilinear Mesh Generation in 3D." *Computer-Aided Design.* Vol 33. (2001): 199-209.

[18] J. Dompierre, P. Labbé, F. Guibault and R. Camarero. "Benchmarks for 3D Unstructured Tetrahedral Mesh Optimization." *Proceedings of the 7th International Meshing RoundTable.* Dearborn, MI. (1998): 459-478.

[19] G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide, Third Edition.* Waltham, MA: Academic Press. 1992.

[20] FMDB: Flexible Distributed Mesh DataBase. [online]. Available from *http://www.scorec.rpi.edu/FMDB.* (Date last accessed: November 11, 2011).

[21] L. A. Freitag, M. T. Jones and P. E. Plassmann. "An Efficient Parallel Algorithm for Mesh Smoothing." *Proceedings of the 4th International Meshing Roundtable.* Sandia National Laboratories, Albuquerque, NM. (1995): 47-58.

[22] L. A. Freitag and P. M. Knupp. "Tetrahedral Element Shape Optimization via the Jacobian Determinant and Condition Number." *Proceeding of the 8th International Meshing Roundtable.* South Lake Tahoe, CA. (1999): 247-258.

[23] R. V. Garimella. "Anisotropic Tetrahedral Mesh Generation." *PhD Thesis.* Rensselaer Polytechnic Institute, Troy NY. 1999.

[24] P. L. George. *Automatic Mesh Generation: Applications to Finite Element Methods.* New York, NY: John Wiley & Sons, Inc. 1992

[25] P. L. George. "Improvements on Delaunay-based Three-dimensional Automatic Mesh Generator." *Finite Elements in Analysis and Design.* Vol 25, No 3. (1997): 297-317.

[26] W. J. Gordon and C. A. Hall. "Transfinite Element Methods: Blending-funtion Interpolation over Arbitrary Curved Element Domains." *Numerische Mathematik.* Vol 21, No 2. (1973): 109-129.

[27] S. Gosselin and C. Ollivier-Gooch. "Tetrahedral Mesh Generation Using Delaunay Refinement with Non-standard Quality Measures." *International Journal for Numerical Methods in Engineering.* Vol 87, Issue 8. (2011): 795-820.

[28] R. Harber, M. S. Shephard, J. F. Abel, R. H. Gallagher and D. P. Greenberg, "A General Two-Dimensional, Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings." *Internationl Journal for Numerical Methods in Engineering.* Vol 17, Issue 7. (1981): 1015-1044.

[29] T. J. R. Hughes. *The Finite Element Method, Linear Static and Dynamic Finite Element Analysis.* Mineola, NY: Dover Publications, Inc. 2000.

[30] T. Hughes, J. Cottrell and Y. Bazilevs. "Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement." *Computer Methods in Applied Mechanics and Engineering.* Vol 194, No 39-41. (2005): 4135-4195.

[31] Interoperable Technologies for Advanced Petascale Simulations (ITAPS), iMeshP Specification. [online]. Available from: *http://www.itaps.org/software/download_interfaces.html.* (Date last accessed: September 16, 2011)

[32] P. M. Knupp. "Algebraic Mesh Quality Metrics." *SIAM Journal on Scientific Computing.* Vol 23, No 1. (2010): 193-218.

[33] P. M. Knupp. "Algebraic Mesh Quality Metrics for Unstructured Initial Meshes." *Finite Elements in Analysis and Design.* Vol 39. (2003): 217-241.

[34] P. M. Knupp. "Introducing the Target-Matrix Paradigm for Mesh Optimization via Node-Movement." *Proceedings of the 19th International Meshing Roundtable.* Chattanooga, TN. (2010): 67-84.

[35] L.-Q. Lee, Z. Li, C. Ng and K. Ko. "Omega3P: A Parallel Finite-Element Eigenmode Analysis Code for Accelerator Cavities." *SLAC-PUB-13529*, SLAC National Accelerator Laboratory. Menlo Park, CA. February 2009

[36] X. Li, M. S. Shephard and M. W. Beall. "Accounting for Curved Domains in Mesh Adaptation." *International Journal for Numerical Methods in Engineering*. Vol 58, No 2. (2003): 247-276.

[37] X. Li. "Mesh Modification Procedures for General 3d Non-manifold Domains." *PhD Thesis*. Rensselaer Polytechnic Institute, Troy, NY. 2003.

[38] X. Li, M. S. Shephard and M. W. Beall. "3D Anisotropic Mesh Adaptation by Mesh Modification." *Computer Methods in Applied Mechanics and Engineering*. Vol 194. (2005): 4915-4950.

[39] A. Liu and B. Joe. "On The Shape of Tetrahedra from Bisection." *Mathematics of Computation*. Vol 63, No 207. (1994): 141-154.

[40] A. Liu and B. Joe. "Relationship Between Tetrahedron Shape Measures." *BIT Numerical Mathematics*. Vol 34, No 2. (1994): 268-287.

[41] X. Luo, et al. "p-Version Mesh Generation Issues." *Proceedings of the 11th Meshing Roundtable.* Ithaca, NY. (2002): 343-354.

[42] X. Luo, et al. "Automatic p-Version Mesh Generation for Curved Domains." *Engineering with Computers*. Vol 20. (2004): 265-285.

[43] X. Luo. "An Automatic Adaptive Directional Variable p-Version Method in 3D Curved Domains." *PhD Thesis*. Rensselaer Polytechnic Institute, Troy, NY. 2005.

[44] X. Luo, M. S. Shephard, L.-Q. Lee, C. Ng and L. Ge. "Tracking Adaptive Moving Mesh Refinements in 3D Curved Domains for Large-Scale Higher Order Finite Element Simulations." *Proceedings of the 17th International Meshing RoundTable.* Pittsburgh, PA. (2008): 585-602.

[45] X. Luo, M. S. Shephard, L.-Q. Lee, C. Ng and L. Ge. "Curved Mesh Correction and Adaptation Tool to Improve COMPASS Electromagnetic analyses." *Journal of Physics: Conference Series*. Vol 125, No 1. (2008): 1-5.

[46] 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*. Vol 27, No 1. (2010): 41-50.

[47] MeshAdapt Service. [online]. Available from: *http://www.itaps.org/tools/services/adaptServ.html*. (Date last accessed: November 21, 2011)

[48] Mesh Curving Service. [online] Available from: *http://www.itaps.org/tools/services/curve.html*. (Date last accessed: November 21, 2011)

[49] G. Morin and R. Goldman. "On the Smooth Convergence of Subdivision and Degree Elevation for Bézier Curves." *Computer Aided Geometric Design.* Vol 18. (2001): 657-666.

[50] NetCDF (Network Common Data Form). [online]. Available from: *http://www.unidata.ucar.edu/software/netcdf/*. (Date last accessed: November 22, 2011)

[51] NERSC: National Energy Research Scientific Computing Center. [online]. Available from: *http://www.nersc.gov/*. (Date last accessed: September 5, 2011)

[52] C. Ng, et al. "State of the Art in EM Field Computation." *SLAC-PUB-12020.* SLAC National Accelerator Laboratory. Menlo Park, CA. August 2006.

[53] NX: Siemens PLM Software [online]. Available from: *http://www.plm.automation.siemens.com/en_us/products/nx/*. (Date last accessed: November 20, 2011)

[54] Parasolid: Siemens PLM Software. [online]. Available from: *http://www.plm.automation.siemens.com/en_us/products/open/parasolid/*. (Date last accessed: November 20, 2011)

[55] ParMA. Package for Parallel Mesh Partition Improvement. [online]. Available from: *http://redmine.scorec.rpi.edu/projects/pima*. (Date last accessed: November 23, 2011)

[56] ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering. [online]. Available from: *http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview*. (Date last accessed: November 23, 2011)

[57] V. N. Parthasarathy, C. M. Graichen and A. F. Hathaway. "A Comparison of Tetrahedron Quality Measures." *Finite Elements in Analysis and Design.* Vol 15. (1993): 255-261.

[58] P.-O. Persson and J. Peraire. "Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics." *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit.* (January 2009): AIAA-2009-949.

[59] H. Prautzsch and L. Kobbelt. "Convergence of Subdivision and Degree Elevation." *Advances in Computational Mathematics.* Vol 2. (1994): 143-154.

[60] J.-F. Remacle, O. Klaas, J. E. Flaherty and M. S. Shephard, "Parallel Algorithm Oriented Mesh Database." *Engineering with Computers.* Vol 18, No 3. (2002): 274284.

[61] O. Sahni. "Automated Adaptive Viscous Flow Simulations." *PhD Thesis.* Rensselaer Polytechnic Institute. Troy, NY. 2007.

[62] O. Sahni, K. E. Jansen, M. S. Shephard, C. A. Taylor and M. W. Beall. "Adaptive Boundary Layer Meshing for Viscous Flow Simulations." *Engineering with Computers.* Vol 24, No 3. (2008): 267-285.

[63] O. Sahni, X. J. Luo, K. E. Jansen and M. S. Shephard. "Curved Boundary Layer Meshing for Adaptive Viscous Flow Simulations." *Finite Elements in Analysis and Design.* Vol 46, Issues 1-2. (2010): 132-139.

[64] T. W. Sederburg. *Computer Aided Geometric Design*, 2011. [online]. Available from: *http://tom.cs.byu.edu/ 557/text/cagd.pdf.* (Date last accessed: November 23, 2011).

[65] E. S. Seol. "FMDB : Flexible Distributed Mesh Database." *PhD Thesis.* Rensselaer Polytechnic Institute, Troy, NY. 2005.

[66] E. S. Seol and M. S. Shephard. "Efficient Distributed Mesh Data Structure for Parallel Automated Adaptive Analysis." *Engineering with Computers.* Vol 22, No 3. (2006): 197213.

[67] M. S. Shephard, S. Dey and M. K. Georges. "Automatic Meshing of Curved Three-Dimensional Domains: Curving Finite Elements and Curvature-Based Mesh Control." *Proceedings of the IMA Summer program, Modeling, Mesh Generation, and Adaptive Numerical Method for Partial Differential Equations.* Vol 75. (1994): 67-96.

[68] M. S. Shephard. "Meshing Environment for Geometry-based Analysis." *International Journal for Numerical Methods in Engineering.* Vol 47, No 1-3. (2000): 169-190.

[69] Simmetric Inc. [online]. Available from: *http://www.simmtrix.com.* (Date last accessed: November 23, 2011)

[70] SolidWorks - Spatial Corp. [online]. Available from: *http://www.solidworks.com.* (Date last accessed: November 20, 2011)

[71] B. A. Szabo and I. Babuska. *Finite Element Analysis.* New York, NY: John Wiley & Sons Inc, 1991.

[72] J. Wan. "An Automatic Adaptive Procedure for 3D Metal Forming Simulations." *PhD Thesis.* Rensselaer Polytechnic Institute, Troy, NY. 2006.

[73] M. Zhou. "Petascale Adaptive Computational Fluid Dynamics." *PhD Thesis.* Rensselaer Polytechnic Institute, Troy, NY. 2009.

[74] M. Zhou, O. Sahni, T. Xie, M. S. Shephard and K. E. Jansen. "Unstructured Mesh Partition Improvement for Implicit Finite Element at Extreme Scale." *The Journal of Supercomputing.* [online first] (December 8, 2010). Available from: *http://www.springerlink.com/content/a3870064466n8418/fulltext.pdf.* (Date last accessed: November 23, 2011)

[75] M. Zhou, O. Sahni and K. D. Devine, M. S. Shephard and K. E. Jansen. "Controlling Unstructured Mesh Partitions for Massively Parallel Simulations." *SIAM Journal on Scientific Computing.* Vol 32, No 6. (2010): 3201-3227.

[76] 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.* Vol 33. (1992): 1331-1364.

[77] O. C. Zienkiewicz and J. Z. Zhu. "The Superconvergent Patch Recovery and A Posteriori Error Estimates. Part 2. Error Estimates and Adaptivity." *International Journal for Numerical Methods in Engineering.* Vol 33. (1992): 1365-1382.

[78] Zoltan: A Dynamic Load-balancing Library for Parallel Applications. Sandia National Labs. [online]. Available from: *http://www.cs.sandia.gov/zoltan.* (Date last accessed: November 20, 2011)