PARALLEL CURVED MESHING FOR HIGH-ORDER FINITE ELEMENT SIMULATIONS

By

Qiukai Lu

A Dissertation Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: MECHANICAL ENGINEERING

Examining Committee:

Dr. Mark S. Shephard, Dissertation Adviser

Dr. Assad A. Oberai, Member

Dr. Bruce R. Piper, Member

Dr. Onkar Sahni, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2017
(For Graduation May 2017)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

It is well known that high-order finite element methods (FEM) are among the most powerful methods for simulating complex engineering problems in terms of solution accuracy and rate of convergence. In order to fully realize the benefits of using high-order methods, the mesh entities representing curved domain geometry must be curved and provide high-enough order of geometry approximation to prevent the loss of convergence due to geometric approximation errors. For high-order finite element methods, it has been demonstrated that properly curved meshes with higher-order continuity between the elements representing curved domain can achieve better solution properties. Although attaining greater than C^0 continuity is being increasingly used with tensor product representations over quadrilaterals, there is the desire to have higher than inter-element continuity on unstructured meshes where triangular finite element faces are used to represent curved domain surfaces.

This thesis presents developments of curved meshing procedures that effectively represent curved domain boundaries by using triangular surface patches of high accuracy and smoothness. A procedure has been developed to generate G^1-continuous triangular surface meshes based on positional and surface normal data sampled from the CAD model representing the problem domain. Specific parameterization approaches based on blending functions are used to define the mapping for curved element faces and volumes between local and global coordinate systems. To effectively adapt curved G^1 meshes to satisfy a desired mesh size field, a set of mesh modification operations, including topological as well as geometrical operations, have been extended to deal with the complexities risen from the high-order smooth mesh entities. The software implementation has been integrated with well established finite element solvers using high-order methods. Benefits of using adaptive curved meshing techniques are demonstrated through examples in the Computational Fluid Dynamics (CFD) applications for viscous flow analysis with curved boundary elements, as well as in Computational Electromagnetism simulations using vector finite elements.

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

Computer aided design and simulation methods are invaluable tools for scientific research and industrial development on a wide range of problems. Among them, the finite element method (FEM) is a powerful tool for solving complex engineering problems in structural analysis, fluid dynamics, electromagnetism, and many other areas. The finite element method relies on a mesh, a spatial discretization of the geometry domain into simple geometric pieces that makes numerical solution possible. Tetrahedral meshes are a popular choice for discretization of complex three-dimensional domains due to the availability of methods for their automatic generation. Accuracy of a finite element analysis depends on the spatial discretization and the polynomial order of the elements. It is well known that high-order finite element methods are capable of producing superior analysis results in terms of solution accuracy and rate of convergence compared with conventional low order FEM [6]. In order to fully realize the benefits of using high-order methods, the curved portions of the geometric domain must be properly represented by the computational mesh. An analysis based on the relation of approximation theory to the convergence of the error in the energy norm indicates that the numerical solution will converge so long as the geometric approximation of the mesh is within one order of that used in the finite element basis [7]. As a result, the mesh entities representing curved domain geometry must be curved and provide high-enough order of geometry approximation to prevent the loss of convergence due to geometric approximation errors [8, 9].

1.2 Historical Review

Early attempts to accurately represent curved domains for finite element simulations date back to the 1970s when the isoparametric element approach was introduced to solid mechanics applications [10]. The approach was quickly adopted

by applications in a wide range of other areas and became a standard technique for approximating curved domain boundary in classic FEM. Gordon and Hall [11] introduced the transfinite elements which employs blending functions to define mapping between reference and physical domains thus allows for the use of exact boundary geometry in FEM. With the rapid development of the Computer-Aided Geometric Design (CAGD) technology, researchers started to work on integration of CAD technology for geometrical representation with finite element analysis methods. Schramm and Pilkey [12] used Non-Uniform Rational B-Splines (NURBS), which is the de facto industrial standard for geometric modeling, for geometry to implement transfinite elements and applied it to shape optimization. Dey et al [7] also introduced an approach that uses NURBS geometry based on blending function based mapping. One of many issues involved in using curvilinear meshes is the ability to effieicent evaluate numerical integration since the integrands are in general not polynomials. Many researchers have proposed different ways to deal with the issue. Sherwin et al [13] proposed an approach to make use of degenerate tensor product type of mapping to directly evaluate integrand for curvilinear tet meshes. Dey et al proposed a way to interpolate the non-polynomial integrand with polynomials to an order of accuracy that is non-dominate, and carry out the numerical integrate on the polynoimals effifiently [14]. A number of different techniques have been proposed in recent years to generate curvilinear meshes based on high-order polynomial mappings and optimal nodal placement [15, 16]. Luo et al have proposed to use Bezier polynomials in their curvilinear meshing strategy and developed specific validity check and curved mesh modification procedures based on the curved Bezier geometry [9]. Persson and Peraire [17] proposed a strategy to generate curved mesh by deforming the mesh using a solid mechanics analogy. Sevilla et al [18] proposed the NURBS-enhanced FEM method which combines NURBS based elements near the CAD model boundary and standard polynomial based finite elements for the interior. Researchers are developing new analysis technique based entirely on CAD, and the Isogeometric Analysis (IGA) methods have become very popular in recent years [19]. The key idea is to use the same CAD description and basis functions to both represent the geometry and approximate the analysis solution. Due to inherent

tensor product nature of the NURBS patches, the IGA methods have been limited to applications of quad surfaces and hexahedral volume meshes.

A well designed mesh adaptation procedure provides critical capabilities that is needed to support the use of adaptive finite element simulations. As a result, it has been an area of active research interest for several decades. The majority of the research efforts on mesh adaptation have been focused on h-adaptivity dealing with low order meshes with all straight-sided elements. In order to achieve the full strength of the high-order methods, one needs to make use of hp-adaptive methods on curvilinear meshes where extra complexities may arise when the mesh involves curved entities. Details of the hp-adaptive methods will be reviewed and discussed in later chapters regarding mesh adaptation. A curved mesh adaptation procedure designed to operate for curved quadratic C^0 meshes is presented in [8]. The core procedure consists of two classes of mesh modification operations: entity geometry modification and local mesh modification for curved meshes. For the entity geometry modification, curved entity reshape operations that explicitly resolve element invalidity and improve the shape quality of curved elements are presented. The local mesh modification operations for curved meshes were extended from the operations for straight-sided meshes [20], with additional consideration and treatment of curve boundary entities and selected curved interior entities. Other mesh adaptation techniques for fully unstructured curved meshes are discussed in references [21, 22, 15, 17]. For simulations that require the numerical solutions to have extremely high accuracy, high mesh resolution is required in critical regions. Even when taking advantage of the benefits of adaptively refined meshes, element counts in the millions are common for problems with both complex physics and complex geometry. Such meshes can only be created and analyzed using large scale parallel computing systems, which requires effective parallel mesh adaptation techniques [23].

1.3 Objectives

The author's dissertation research aims to develop a novel approach for high-order curved meshing technique to effectively treat curved boundaries, which takes

advantage of the CAD technologies and uses surface patches for high-order, accurate, smooth representation of the finite element computational domain, while still maintaining the flexible and local properties of the unstructured meshing (generation, adaptation) techniques, in particular, triangular surface patches. Parallel curved mesh adaptation techniques are extended and improved to support the adaptation of high-order, higher-continuity curved meshes in order to be utilized in the context of adaptive finite element simulations.

The primary objectives of this research include the following:

- developing curved mesh representation techniques for finite element simulations using high-order methods, with specific focus on meshes with high-order geometric approximation accuracy, and high-order geometric continuity.
- developing curved mesh adaptation techniques for adaptive simulations, including mesh refinement, coarsening, and optimization.
- parallel execution of the curved meshing operations.
- integration of curved meshing procedure with existing high-order solvers and demonstration of the impact of the improved mesh geometry on the simulation solution accuracy.

1.4 Thesis Organization

The rest of the dissertation is organized as follows: Chapter 2 presents the curved mesh representation and mesh entity curving techniques. Chapter 3 discusses curved mesh adaptation and its parallel execution. Chapter 4 presents the aspects of solver integration. Chapter 5 presents demonstration cases of the integrated curved meshes with a high-order finite element solver in the application areas of fluid flow simulation. Chapter 6 summarizes the contribution of the dissertation and recommends future developments.

1.5 Nomenclature

The nomenclature used in this dissertation is defined as follows:

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.
$\S_i(\xi)$	A parametric surface patch
$\Gamma_i(\xi)$	A parametric space curve

CHAPTER 2
CURVED MESH REPRESENTATION

2.1 Introduction

This chapter presents techniques to represent curved mesh entities in ways that facilitate convenient construction of high-order curved tetrahedral meshes which approximate curved domain boundaries to the desired order. The curved mesh representation scheme is based on Bézier polynomials commonly used in the Computer Aided Geometric Design (CAGD) community. Adopting such a curved mesh representation provides convenient ways to support geometry related evaluation operations during finite element analysis processes, and it serves as a foundation for mesh curving and adaptation techniques to be presented in later chapters.

2.2 Curved Mesh Representation

Driven by the developments of high-order finite element analysis techniques and applications, for instance the p-version finite element methods, effective curvilinear mesh generation and adaptation techniques have become an important building block to construct adaptive finite element simulation loops. As a result, development of curvilinear mesh generation techniques has recently been an active area of research in the finite element meshing community.

To achieve exponential rate of convergence possible with high-order methods, mesh entities on curved domain boundaries must properly approximate the geometry of the model to the correct order. For example, the work of Sherwin et al on curvilinear mesh generation provided evidence that properly curved meshes help to increase the accuracy of the finite element and spectral element methods for fluid problems [16]. A study by Luo et al [9] based on the relation of approximation theory to the convergence of the error in the energy norm indicated that the energy norm of a finite element solution for second-order elliptic partial differential equations will converge so long as the geometric approximation of the mesh is within one order of that used in the finite element basis [9]. Ainsworth et al provided proof

of a computable error bounds for finite element solutions of Poisson's equations on non-polygonal domains [24]. In a survey article, Wang et al pointed out the importance of having high-order mesh generation capabilities in order to achieve the full potential of high-order CFD methods [25]. In the case where the finite elements are defined in terms of interpolating Lagrange polynomials, the geometric approximation requirement is met by being sure that all nodes at mesh vertices, on mesh edges and on mesh faces on curved domain boundaries are placed on the appropriate boundary with optimal placement schemes. It is well known that simple node placement method such as equal spacing nodes can lead to severely bad interpolation results [9]. Various optimal nodal placement schemes have been developed. For instance, Chen and Babuska proposed optimal interpolation points for polynomial functions based on minimization of L^2-norm of the Lebesgue constant [26]. In addition, a well known set of point distribution called Fekete points are investigated by Bos [27] and Taylor et al [28] by searching for a nodal set with small Lebesgue constant to maximize the determinant of the Vandermonde matrix. If different basis functions other than the standard Lagrange polynomials for interpolating element geometries are chosen, one has to carefully consider how to satisfy the geometric approximation requirements for these elements by the proper improvement of the mesh edge and face shapes.

2.2.1 Curvilinear Mesh Representation Based on Bézier Polynomials

Various types of methods to represent curvilinear mesh geometry have been proposed and studied over the past several decades. Historically, high-order Lagrange polynomials have been used to represent curved finite element geometry in the context of isoparametric finite element methods, which can be attributed to the pioneering work by Ahmad, Irons and Zienkiewicz on analysis of curved shell structures [29]. Despite the wide adoption of the isoparametric approach for solid mechanics applications, shortcomings are reported in applications for computational fluid dynamics (CFD) as well as computational electromagnetics (i.e. numerical analysis of Maxwell's equations). For instance, Bassi and Rebay [30] identified the origin of spurious entropy production near curved wall boundaries in the numeri-

cal solution of Euler equations of gas dynamics as caused by low-order geometric approximation of curved walls. Xue and Demkowicz [31] showed for 3D Maxwell's equations the exact geometry mapping reduces the error in solution by one order of magnitude compared to isoparametric approach. The identified drawbacks of isoparametric approach motivated several techniques aimed to incorporate the exact geometry into the finite element analysis. For instance the transfinite elements proposed by Gordon and Hall [11] and later implemented by Schramn and Pilkey in shape optimization [12]. In the meantime as the research and development of the CAGD techniques grow rapidly, more meshing techniques have been proposed that base themselves on CAD interpolation basis functions. For instance, Bézier polynomials are being increasingly used to represent curvilinear meshes thanks to a set of unique properties of the Bézier polynomials and the fact that it has been extensively researched in the CAGD community. As an effort to unify the fields of CAGD and FEM, a type of analysis methods have been proposed which are generally referred to as iso-geometric analysis (IGA), and a type of geometric modeling methods are being actively researched called analysis-aware modeling. The main drawback of the IGA methods is that a solid CAD modeler for 3D domains is needed, while in practice, most CAD modelers work with the so-called boundary representation where the 3D domains geometry is given by a set of parametric surface rather than a parametric solid. Due to the nature of the NURBS basis functions used for CAD surface parametrization, the development meshing techniques for IGA methods is focused primarily on tensor product type of meshes, e.g. quads and hexes. (See [32, 33] for example) The work presented in this thesis focuses, in stead, on providing geometry representation that supports unstructured meshes with simplex elements. In addition, it is desirable to support mesh geometry that are independent of the interpolation basis functions used for analysis such that it can be easily integrated with various finite element analysis code. In particular, the work in this thesis focuses on Bézier polynomial based meshing techniques. Therefore, both IGA and analysis-aware modeling techniques are outside the scope of this thesis. Interested readers can refer to specialized literature for details of IGA [19] and analysis-aware modeling [34].

2.2.1.1 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 [35]. 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 nth 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 Equation 2.1, $b_i^{(n)}(t)$ is the ith Bernstein basis function of degree n,

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

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

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 $\mathbf{B}(t)$ maps a one-dimensional parametric coordinate system ($t \in R$) to a 3D Cartesian coordinate system ($\mathbf{B}(t) \in R^3$), and vice verse.

Bézier polynomials have a number of properties which make them useful for representing geometry[35]. Key ones of importance to the development of curved meshing algorithms for this work are listed as follows:

1. The Convex Hull Property. The value of a Bézier polynomial evaluated within the its parametric domain is bounded by the maximum and minimum values of its corresponding control points. Figure 2.1 illustrates the convex hull of a third order Bézier polynomial in 2D.

2. The derivatives of a Bézier polynomial remain Bézier polynomials. The prod-

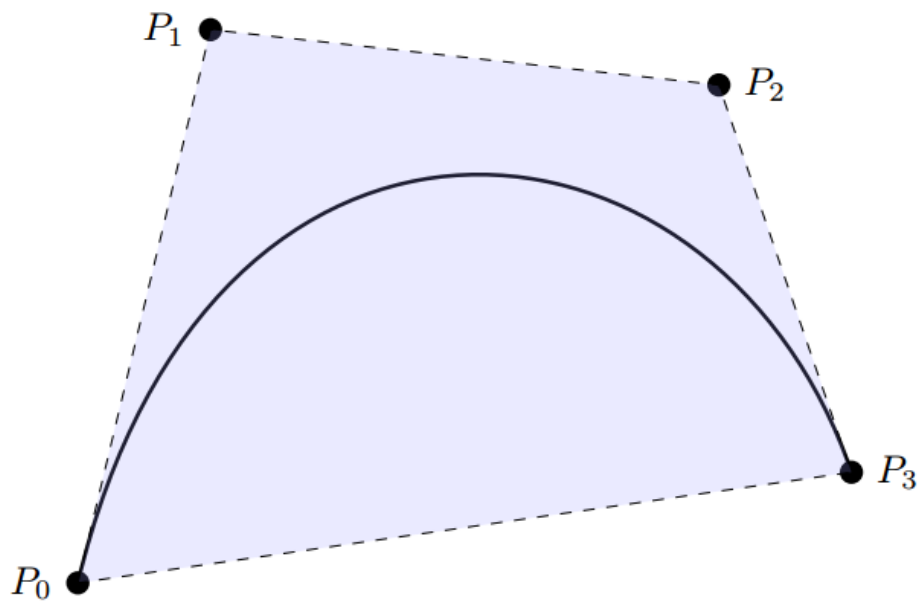


Figure 2.1: Convex hull of a third-order Bézier curve

uct of a mth order Bézier polynomial and a nth order Bézier polynomial is also a Bézier polynomial of order $(m + n)$.

3. It is straight-forward to split a Bézier curve into multiple sub-curves by subdivision or inflate the order of a given Bézier curve through degree elevation. The algorithms to perform subdivision and degree elevation are readily available. The property turns out to be useful for the h-version and p-version mesh adaptation procedures. Figure 2.2 shows an example of degree elevation and Figure 2.3 shows and example of subdivision respectively.

2.2.1.2 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 [21, 36], the nth order Bézier polynomial based volume representation of high-order tetrahedrons can be expressed by making use of the volume coordinates as follows:

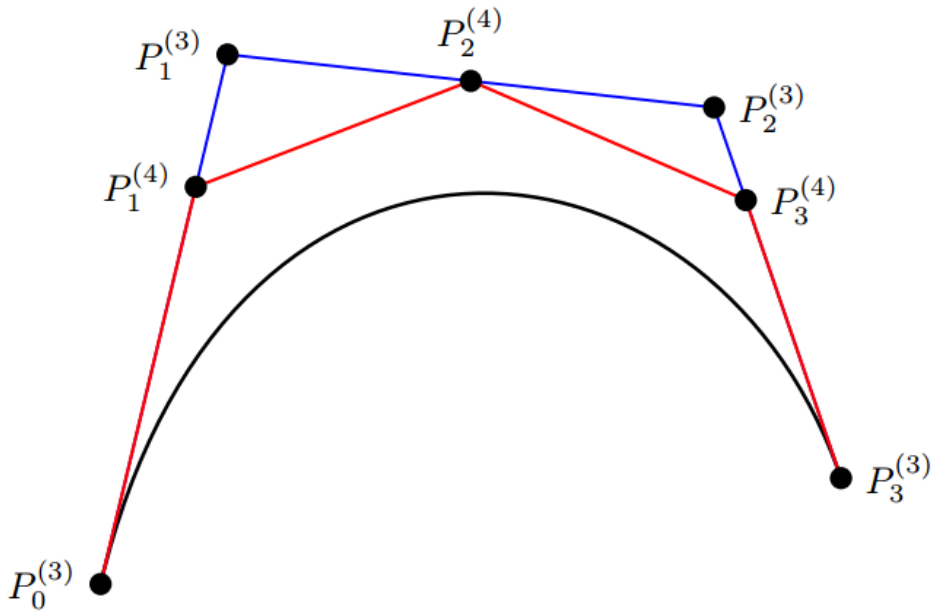


Figure 2.2: Degree elevation of a third-order Bézier curve to fourth order

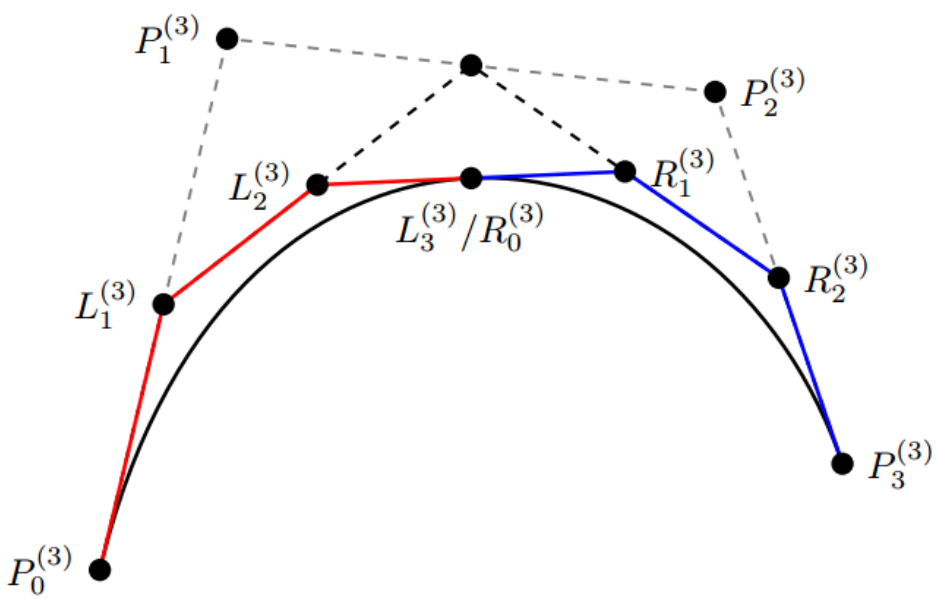


Figure 2.3: Subdivision of a third-order Bézier curve into two third-order curves

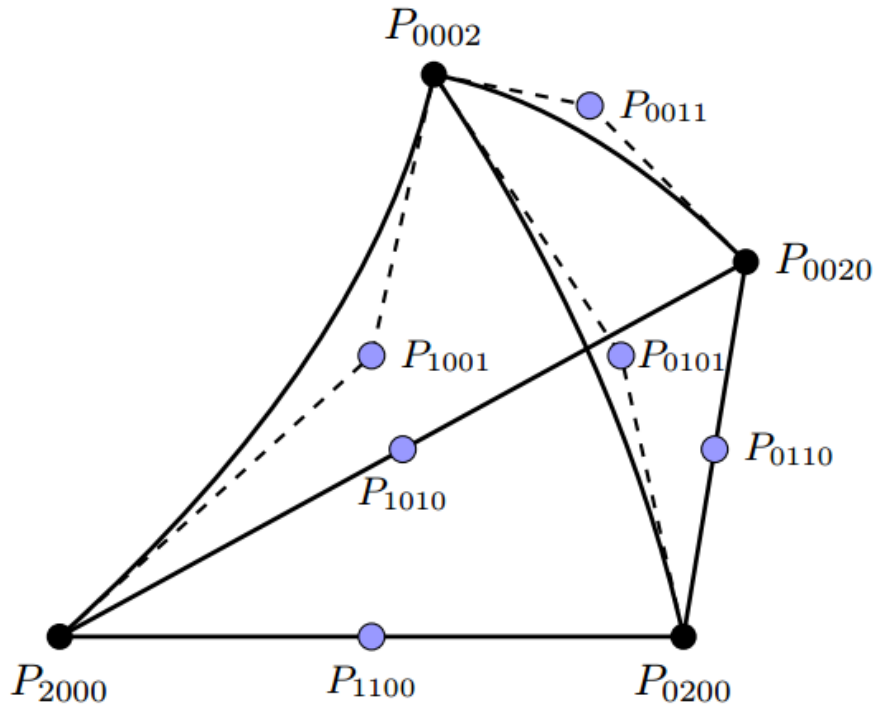


Figure 2.4: 2nd-order curved tetrahedron

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

where $b_i^{(n)}(\xi)$ are the Bernstein polynomials with $\xi = \{(\xi_1, \xi_2, \xi_3, \xi_4) | \xi_1 + \xi_2 + \xi_3 + \xi_4 = 1 \text{ and } \xi_i \in [0, 1]\}$. q is the total number of control points for a nth order Bézier polynomial in 3D. And q is determined by Eq 2.5

$$q = \sum_{i=0}^{n} \frac{(i+1)(i+2)}{2} \tag{2.5}$$

More specifically in the case of a 2nd-order curved tetrahedron (see Figure 2.4), the Bézier representation is:

$$\begin{aligned}\mathbf{B}^{(2)}(\xi_1,\xi_2,\xi_3,\xi_4) &= \sum_{i=1}^{4}\mathbf{P}_i^{(2)}(M_i^0)C_1\xi_i^2 \\ &+ \sum_{j=1}^{6}\mathbf{P}_j^{(2)}(M_j^1)C_2\xi_m\xi_n \end{aligned} \tag{2.6}$$

where $m, n = 1, 2, 3, 4$ and $m \neq n$. $\mathbf{P}_i^{(2)}(M_i^0)$ are four control points associated with the vertices, and $\mathbf{P}_j^{(2)}(M_j^1)$ are six control points associated with the edges. $C_i = \frac{2!}{i!}$ are the coefficients of the second order Bernstein polynomials. For higher-order curved tetrahedra, additional terms which are associated with face control points and volume control points will appear in Eq 2.6.

2.2.2 Curved Surface Meshes with Higher-Order Continuity

In addition to the conventional curvilinear mesh representation techniques based on continuous C^0 mesh elements, the ability to provide higher order of geometric approximation is also facilitated by the use of greater than C^0 geometric shape continuity between elements [2, 37, 38, 39]. In fact, higher order geometric continuity is being increasingly used for curved meshes with tensor product representations over quadrilaterals [40, 19]. Naturally, there is also the desire to have higher than C^0 geometric continuity between elements on unstructured meshes where curved triangular finite element faces are used. The current work aims to investigate and address the technical difficulties associated with developing curved meshing techniques for unstructured meshes where G^1 surface geometry continuity is maintained for the triangular element faces representing the curved domain surfaces.

In the following sections, several fundamental notions in the field of CAGD are briefly reviewed, such as tangent plane (G^1) continuity. The so-called vertex consistency problem is introduced as a difficult problem to solve that arises when constructing a network of G^1 joined patches around a common vertex.

2.2.2.1 Tangent Plane Continuity

To join two neighboring triangular patches smoothly, one seemingly intuitive choice is to ensure C^1 continuity in which case the two patches meet with continuous first derivatives across the common boundary. However, ensuring C^1 continuity requires a continuous parametrization of the two patches, and is not always possible for surface patches of arbitrary topology. In other words, the concept of C^1 continuity is not suitable to characterize the smoothness of a surface since a change in the parametrization of one of two adjacent patches changes the cross-boundary derivatives of that patch, thus destroying the C^1 continuity. Therefore in practice, an alternative is to construct triangular patches that meet with continuous tangent planes along the common boundary, which is referred to as G^1 continuity [41].

Let $S_{i-1}(\xi_1, \xi_2)$ and $S_i(\xi_1, \xi_2)$ be two adjacent patches share a common boundary curve $\Gamma_i(\xi)$. The two patches meet with tangent plane continuity if there exist three scalar functions ϕ, μ, ν such that [41].

$$\phi(t)\frac{\partial S_{i-1}}{\partial \xi_1}(t,0) = \mu(t)\frac{\partial S_{i-1}}{\partial \xi_2}(t,0) + \nu(t)\frac{\partial S_i}{\partial \xi_2}(0,t) \quad t \in [0,1] \tag{2.7}$$

As shown in Fig 2.5, the equation indicates that the three partial derivatives along the boundary curve C_i are always coplanar, thus ensuring tangent plane continuity of S_{i-1} and S_i.

2.2.2.2 Vertex Consistency Problem

Given the tangent plane continuity condition in Eq 2.7, it is relatively straightforward to construct two patches that meet with G^1 continuity. On the other hand, additional complexity arises when one tries to merge a closed network of G^1 joined patches incident to a vertex [42, 41]. The G^1-continuity conditions between patches lead to a system of constraints that are not always solvable for a vertex bounding an even number of edges greater than four. Details of the problem can be found in reference [41]. This problem is commonly referred to as the vertex consistency or twist compatibility problem, and every scheme aiming at constructing a tangent plane continuous surface has to find a way to cope with this problem. In the following sections, several methods are surveyed.

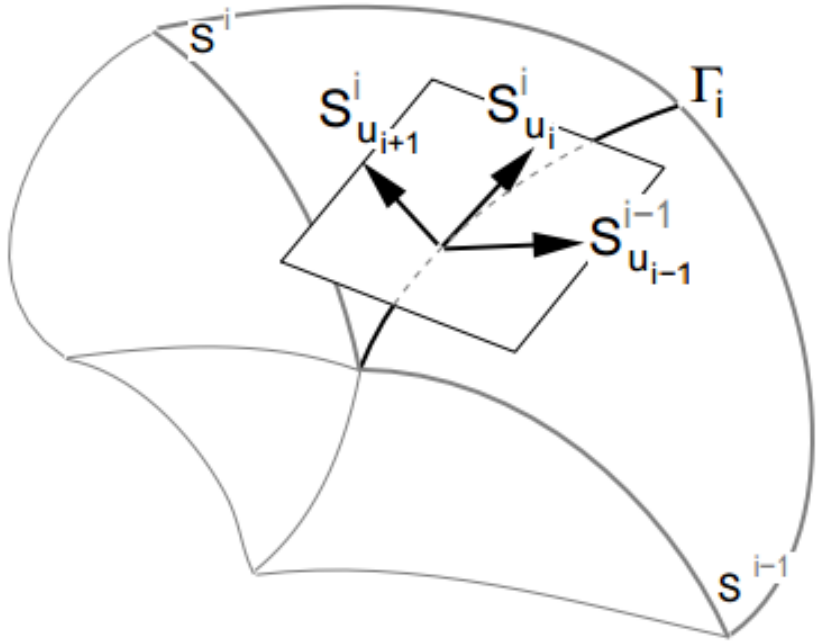


Figure 2.5: Tangent plane continuity [1]

2.2.2.3 Previous Works on Triangular Patches for G^1 Continuous Surface Interpolation

Many triangular patches and schemes have been developed to construct G^1 continuous surface interpolations [43, 2, 3, 1, 44, 45, 38, 46, 39]. In general, these methods start with constructing a network of boundary curves for the edges in the mesh. A cross-boundary tangent field is defined along the boundary curves. The triangular surface patches are then constructed to interpolate the boundary curves and the tangent field. Mann et al [47] surveyed a handful of methods up to the early 1990s. Based on how the vertex consistency problem is tackled, the methods can be categorized into two types of schemes: (1) domain-splitting schemes and (2) convex combination schemes. Boschiroli et al [43] provided a comparative study of the noticeable methods developed after Mann's survey. The methods surveyed in Boschiroli's study focus primarily on convex combination type of schemes that use the rational patches and blending technique.

2.2.2.4 Domain-Splitting Schemes

The domain-splitting scheme is first used by Clough and Tocher [48] to interpolate scalar valued data. Figure 2.6 gives an illustration of a Clough-Tocher type of 3-split scheme [2]. It is then adopted and generalized by researchers such as Piper [38], Shirman and Sequin [46], and others [2, 3, 1] in the CAD community to solve data interpolation problems for parametric surfaces.

As the name suggests, the domain-splitting methods subdivide the domain (or macro triangle) to several sub-triangles such that the constraints of the vertex consistency condition are decoupled and can be solved independently. A major difference among various domain-splitting methods is the ways the triangular domains are subdivided. For instance, Piper [38] used a 3-split scheme which introduces a new vertex at the centroid of the triangle while keeping the edges unchanged. Hahmann et al [3, 1] introduced a 4-split scheme that splits the each one of the three boundary curves into two pieces. See Figure 2.7 for an illustration of Hahmanns 4-split domain scheme [3]. Another difference among the methods has to do with the order of geometric shape functions used to interpolate the data. Piper, and Shirman, Sequin use quartic Bézier polynomials to fill the 3 sub-patches [38, 46]. Hahmanns 4-split scheme uses quintic polynomials [3, 1].

Despite the different ways to split the domain and orders of geometric shape functions being used, all the domain-splitting methods share a set of similar steps for the data interpolation process. After splitting, each of the sub-patches is used to interpolate the data alone one of the boundaries. Splitting allows the data along each boundary to be matched independently of the data on the other two boundaries. For instance, the scheme developed by Shirman and Sequin [46] starts with constructing cubic boundary curves for each edge and subsequently degree raised to quartic. A linearly varying cross-boundary tangent field is computed and the tangent plane continuity condition given by Eq 2.7 is used to ensure G^1 for the exterior boundaries of the sub-patches. After determining the set of control points that ensure the G^1 continuity of exterior boundaries, the remaining degrees of freedom are used to make the internal boundaries of the three sub-patches meet with G^1-continuity.

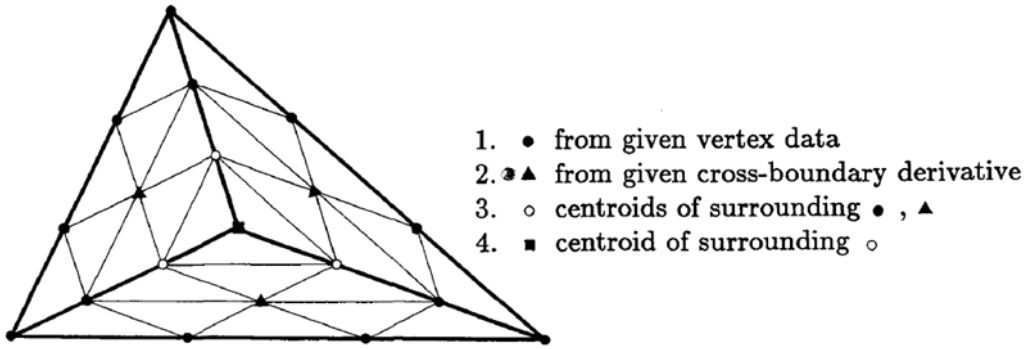


Figure 2.6: Illustration of a Clough-Tocher type of 3-split domain scheme [2]

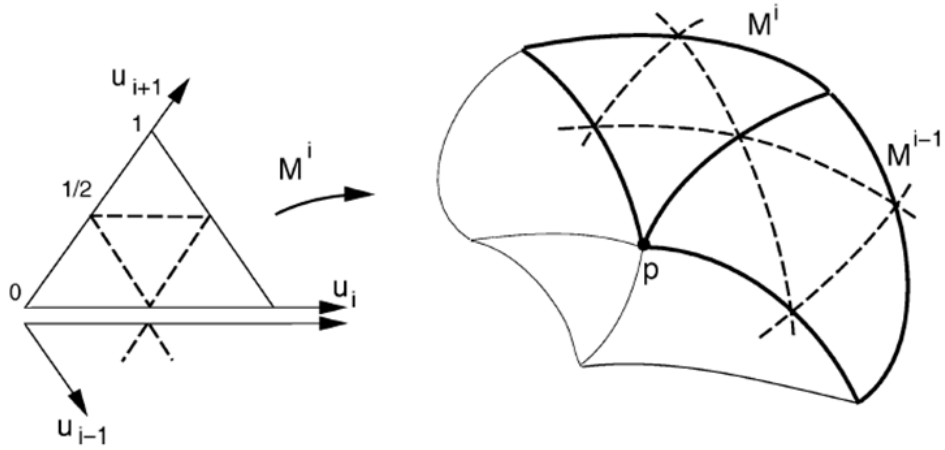


Figure 2.7: Illustration of Hahmanns 4-split domain scheme [3].

2.2.2.5 Convex Combination Schemes

Different from domain-splitting, the Convex combination schemes attempt to create a single parametric patch for each mesh face. The patches are C^2 everywhere except at the vertices [44, 45, 46, 39]. This construction avoids the vertex consistency problem by not having consistently defined mixed partial derivative terms at the patch corner. The scheme starts by first building boundary curves and across-boundary tangent plane fields along the curves. After that, three patches are created, each covering the entire triangle and interpolating the position and normal fields along each boundary curve independently. Finally, a single patch is formed by blending the three patches in a way that the resulting patch interpolates all of the

boundary data. Convex combination scheme normally leads to rational patches due to the usage of blending whereas domain-splitting schemes general creates polynomial patches. For instance, Nielson [44, 45] introduced a side-vertex method by first constructing three boundary curves corresponding to the edges of the input triangle. Three patches are created, one for each pair of edge and its opposite vertex. The interior of each patch is constructed by passing curves from points along the edge to the opposite vertex. The final patch is formed by blending the three patches together.

Let Γ_i denote the boundary curve for edge M_i^1. $\mathbf{P}_0, \mathbf{P}_1$ denote nodes at the end vertcies M_0^0 and M_1^0, and N_0, N_1 are normal vectors at those two vertices.

$$\Gamma_i[\mathbf{P}_0, \mathbf{P}_1, N_0, N_1](\xi) \tag{2.8}$$

such that $\Gamma_i(0) = \mathbf{P}_0$, $\Gamma_i(1) = \mathbf{P}_1$, $\Gamma_i'(0) \cdot N_0 = 0$, and $\Gamma_i'(1) \cdot N_1 = 0$

Assume there exists a normal field constructor C_{in} along Γ_i, together with the position and normal data of the opposite vertex $\mathbf{P}_2(M_2^0), N_2(M_2^0)$. A single surface patch S_i can be constructed that interpolates the tangent plane field of the edge M_i^1.

$$S_i[\Gamma_i, \mathbf{P}_2, C_{in}, N_2](\xi) \tag{2.9}$$

The final surface is defined as

$$S[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, N_0, N_1, N_2] = \beta_0 S_0 + \beta_1 S_1 + \beta_2 S_2 \tag{2.10}$$

where β_i are a set of blending functions. For example, in Nielson's side-vertex scheme, β_i is defined as [44]

$$\beta_i = \frac{\xi_j \xi_k}{\xi_i \xi_j + \xi_j \xi_k + \xi_i \xi_k} \tag{2.11}$$

The blending function introduced here determines that the surface patches constructed using convex combination schemes are rational patches

Another subset of the convex combination methods are based on the triangular

Gregory patches. The vertex consistency problem is addressed in the similar way by constructing boundary curves and interior control points independently for each edge. This results in an additional number of interior control points. When evaluating the patch at a given parametric location, instead of blending surface patches, pairs of interior control points are blended together to form a rational blend Bézier-like patch. For instance, Walton and Meek [39] developed a G^1 quartic Gregory patch based on this scheme. Boschiroli et al [43] proposed a cubic Gregory patch based on Walton and Meeks work. The Walton-Meek construction procedure is presented in the following section.

2.2.2.6 Quartic Triangular G^1 Patch

A fundamental step in the procedure to create G^1 curved meshes involves the scheme to construct triangular G^1 patches for boundary mesh faces that interpolate the position $x_i(\xi)$ and normal data n_j at their corner vertices. The scheme used in this work to represent the curved geometry is based on an extension of the Gregory patch originally proposed by Gregory [49] to bypass the constraint on twist compatibility for smooth surface interpolation with triangular patches. The extension to the original Gregory patch, developed by Walton et al [39], constructs boundary curve network in the way that the principal normals of each curve at the end vertices are always aligned with the prescribed surface normals. For an individual mesh face, each of the three bounding edges is assigned with a geometric representation of a cubic Bézier curve $B^{(3)}(\xi)$.

Let $\mathbf{\Gamma}_i^{(3)}(t)$ denote a cubic bezier curve representing the geometric shape of the ith boundary edge. Let $d_i = \|\mathbf{P}_{i,3} - \mathbf{P}_{i,0}\|$, $\gamma_i = \frac{\mathbf{P}_{i,3}-\mathbf{P}_{i,0}}{d_i}$, $a_i = N_i \cdot N_{i+1}$, $a_{i,0} = N_i \cdot \gamma_i$, $a_{i,1} = N_{i+1} \cdot \gamma_i$. The two edge control points of $\mathbf{\Gamma}_i^{(3)}(t)$ are determined by

$$\begin{aligned}
\mathbf{P}_{i,1} &= \mathbf{P}_{i,0} + \frac{d_i(6\gamma_i - 2\rho_i N_i + \sigma_i N_{i+1})}{18} \\
\mathbf{P}_{i,2} &= \mathbf{P}_{i,3} - \frac{d_i(6\gamma_i + \rho_i N_i - 2\sigma_i N_{i+1})}{18}
\end{aligned} \tag{2.12}$$

where

$$\rho_i = \frac{6(2a_{i,0} + a_i a_{i,1})}{4 - a_i^2} \tag{2.13}$$

and

$$\sigma_i = \frac{6(2a_{i,1} + a_i a_{i,0})}{4 - a_i^2} \tag{2.14}$$

The calculated cubic bezier curve not only interpolates the position and normal data at the end vertices M_0^0 and M_1^0, but also has its principle normal parallel to the face normal vectors at M_0^0 and M_1^0.

Next, a field of tangent planes is spanned by the curve tangent vector $\Gamma_i'(t)$ and cross boundary tangent vector $H_i(t)$, where

$$H_i(t) = \sum_{k=0}^{2} A_{i,k} b_k^{(2)}(t) \tag{2.15}$$

where

$$\begin{aligned} W_{i,k} &= \mathbf{P}_{i,k+1} - \mathbf{P}_{i,k}, \quad k = 0, 1, 2 \\ A_{i,0} &= \frac{N_i \times W_{i,0}}{\|W_{i,0}\|} \\ A_{i,2} &= \frac{N_{i,1} \times W_{i,2}}{\|W_{i,2}\|} \\ A_{i,1} &= \frac{A_{i,0} + A_{i,2}}{\|A_{i,0} + A_{i,2}\|} \end{aligned} \tag{2.16}$$

Then, the cubic bezier curve $\Gamma_i^{(3)}(t)$ is degree-elevated to quartic $\Gamma_i^{(4)}(t)$

$$\begin{aligned} \mathbf{P}_{i,j}^{(4)} = \frac{1}{4}\{j\mathbf{P}_{i,j-1}^{(3)} + (4-j)\mathbf{P}_{i,j}^{(3)}\}, \\ i = 0, 1, 2; \quad j = 0, ..., 4 \end{aligned} \tag{2.17}$$

Finally, two face control points are determined for each boundary edge.

$$\begin{aligned} G_{i,1} &= \frac{1}{2}(\mathbf{P}_{i,1}^{(4)} + \mathbf{P}_{i,2}^{(4)}) + \frac{2}{3}\lambda_{i,0} W_{i,1} + \frac{1}{3}\lambda_{i,1} W_{i,0} \\ &+ \frac{2}{3}\mu_{i,0} A_{i,1} + \frac{1}{3}\mu_{i,1} A_{i,0} \end{aligned} \tag{2.18}$$

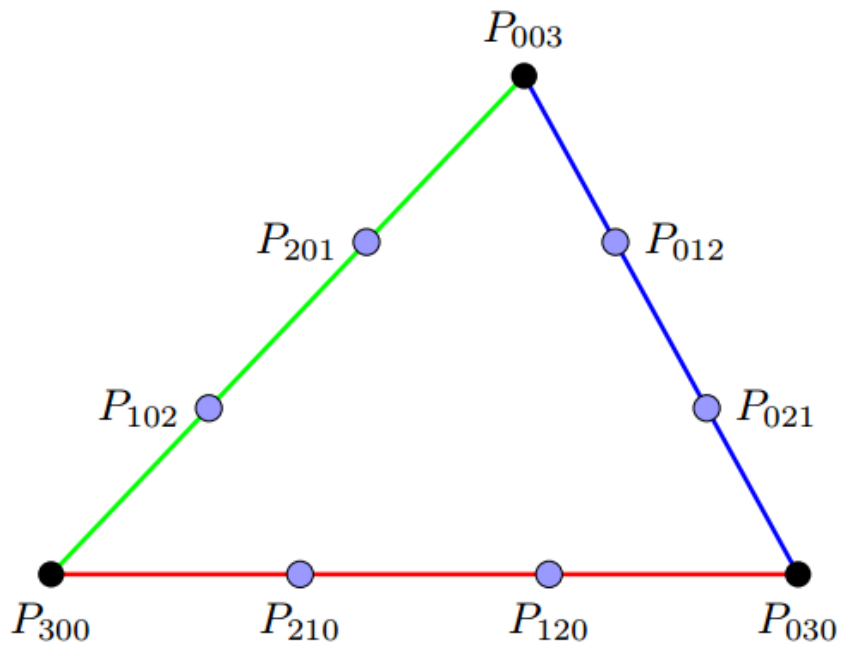


Figure 2.8: Triangular face with third order Bézier bounding curves

and

$$\begin{aligned} G_{i,2} &= \frac{1}{2}(\mathbf{P}^{(4)}_{i,2} + \mathbf{P}^{(4)}_{i,3}) + \frac{1}{3}\lambda_{i,0}W_{i,2} + \frac{2}{3}\lambda_{i,1}W_{i,1} \\ &+ \frac{1}{3}\mu_{i,0}A_{i,2} + \frac{2}{3}\mu_{i,1}A_{i,1} \end{aligned} \tag{2.19}$$

The rational blend degree-4 triangular Bézier patch is defined by

$$\mathbf{S}^{(4)}(\xi) = \sum_{1}^{15} \mathbf{P}^{(4)}_{ijk} b^{(4)}_{ijk}(\xi), \quad i,j,k \geq 1, \quad i+j+k = 4 \tag{2.20}$$

where $\mathbf{P}^{(4)}_{ijk}$ are the control points and $b^{(4)}_{ijk}(\xi)$ are 4th order Bernstein basis functions. The surface control points $\mathbf{P}_{112}, \mathbf{P}_{121}, \mathbf{P}_{211}$ are affine combinations of the split surface control points $G_{i,j}$ computed by Eq.2.18 and Eq. 2.19:

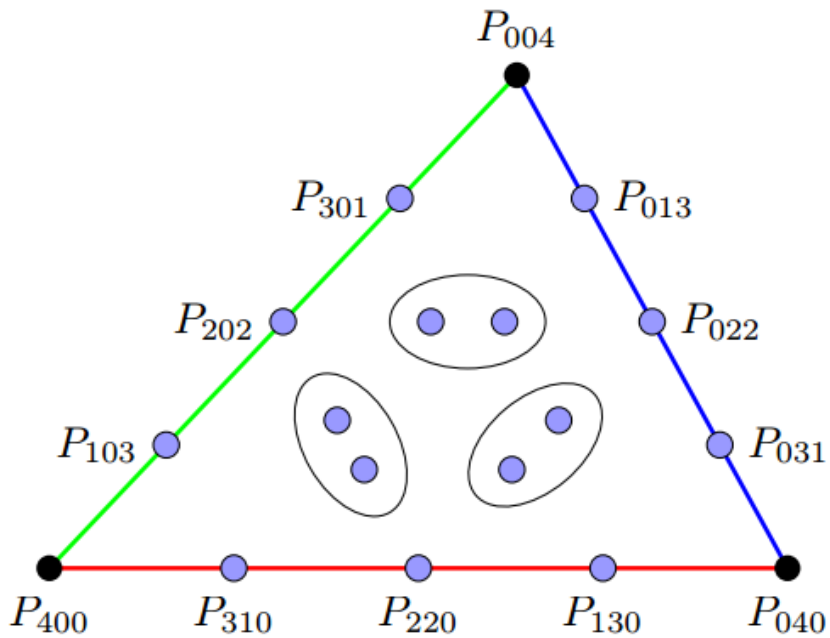


Figure 2.9: Fourth order triangular blended Bézier face

$$
\begin{aligned}
\mathbf{P}_{112} &= \frac{1}{\xi_1+\xi_2}(\xi_1 G_{2,2} + \xi_2 G_{0,1}) \\
\mathbf{P}_{121} &= \frac{1}{\xi_3+\xi_1}(\xi_3 G_{0,2} + \xi_1 G_{1,1}) \\
\mathbf{P}_{211} &= \frac{1}{\xi_2+\xi_3}(\xi_2 G_{1,2} + \xi_3 G_{2,1})
\end{aligned} \tag{2.21}
$$

Fig 2.10 shows an example patch and its control point set.

2.2.3 G1 Meshes Higher Than 4th Order

As the order of the finite element function space increases, it is desirable to provide the capability to support geometry representation higher than 4th order such that the geometric approximation accuracy of the finite element mesh is kept in accordance with the order of finite element space such that it does not become the leading source of error in the entire analysis. There is limited amount of research and published literature discussing high order polynomial patches beyond 4th in both computer graphics and CAE communities. One scheme to create quintic triangular

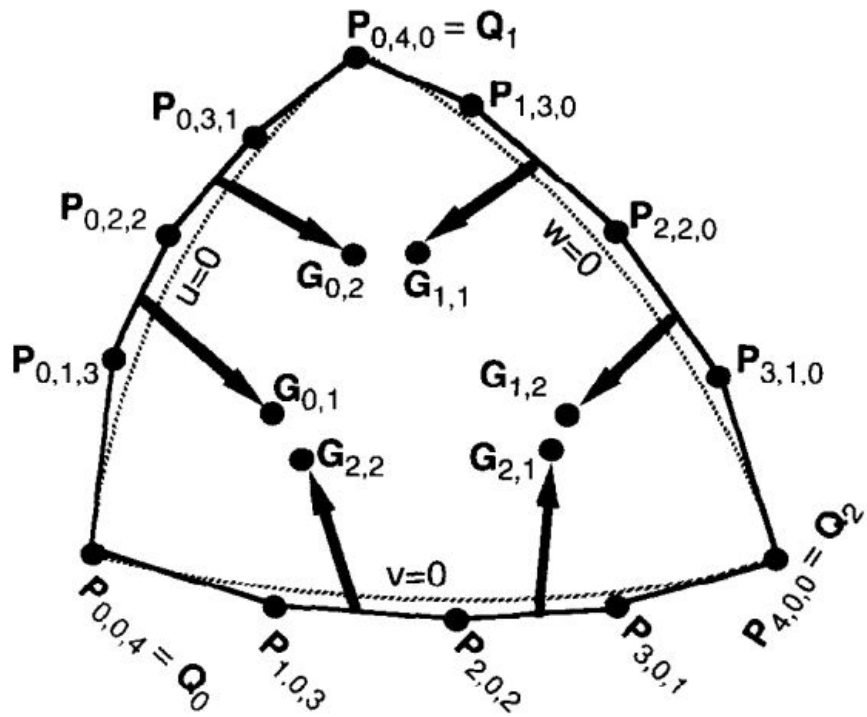


Figure 2.10: Triangular Gregory patch and its control points

patches using Bézier polynomials is developed by Farin [2]. Loop also developed a scheme which uses 6th order polynomials to form patches that are capable of interpolating the surface position and normal data [50]. However, it is also reported in [50] that enforcing interpolation in Loop's scheme leads to unwanted surface undulations due to severe constraints on the second derivatives along the boundary curves.

The scheme proposed by Farin requires six pieces of input data up to the second derivatives at each of the three boundary mesh vertices, i.e. coordinates, 1st and 2nd derivatives: $\mathbf{P}(\xi_1, \xi_2)$, $\frac{\partial \mathbf{P}}{\partial \xi_1}(\xi_1, \xi_2)$, $\frac{\partial \mathbf{P}}{\partial \xi_2}(\xi_1, \xi_2)$, $\frac{\partial^2 \mathbf{P}}{\partial \xi_1^2}(\xi_1, \xi_2)$, $\frac{\partial^2 \mathbf{P}}{\partial \xi_1 \partial \xi_2}(\xi_1, \xi_2)$, $\frac{\partial^2 \mathbf{P}}{\partial \xi_2^2}(\xi_1, \xi_2)$.

Three more pieces of information are prescribed at the mid-edge points of the three boundary edges such that the 21 coefficients of the 5th order Bézier patch can be uniquely determined. The three additional pieces of data are cross-boundary derivatives at edge mid points. For instance, the derivative at edge $M_i^1\{M_1^0, M_2^0\}$ is given by Eq 2.22:

$$\frac{\partial}{\partial \gamma} S(\frac{\xi_1 + \xi_2}{2}) \tag{2.22}$$

where γ denotes the cross-boundary direction.

With the prescribed pieces of data, the equations used to calculate the 6 control points surrounding one vertex of the patch are given as follows [2]:

$$\begin{aligned}
\mathbf{P}_{500} &= S(M_0^0) \\
\mathbf{P}_{410} &= \frac{1}{5}\frac{\partial}{\partial \xi_1} S(M_0^0) + \mathbf{P}_{500} \\
\mathbf{P}_{401} &= \frac{1}{5}\frac{\partial}{\partial \xi_2} S(M_0^0) + \mathbf{P}_{500} \\
\mathbf{P}_{320} &= \frac{1}{20}\frac{\partial^2}{\partial \xi_1^2} S(M_0^0) + 2\mathbf{P}_{410} - \mathbf{P}_{500} \\
\mathbf{P}_{302} &= \frac{1}{20}\frac{\partial^2}{\partial \xi_2^2} S(M_0^0) + 2\mathbf{P}_{401} - \mathbf{P}_{500} \\
\mathbf{P}_{311} &= \frac{1}{20}\frac{\partial^2}{\partial \xi_1 \partial \xi_2} S(M_0^0) + \mathbf{P}_{410} + \mathbf{P}_{401} - \mathbf{P}_{500}
\end{aligned} \tag{2.23}$$

The remaining 3 control points are determined by the cross-boundary tangent data given in Eq 2.22 using the Equations (4.4) and (4.5) in the reference paper by Farin [2]. The resulting surface patch has G^1 geometric continuity across patch boundaries.

Based on Farin's procedure for the quintic patch, a procedure is developed in this work to construct G^1 continuous polynomial patches that are generally applicable to 5th and higher order. The details of the proposed procedure and algorithm are presented in Chapter 3. Figure 2.11 shows an example of a triangular patch with polynomial order 6.

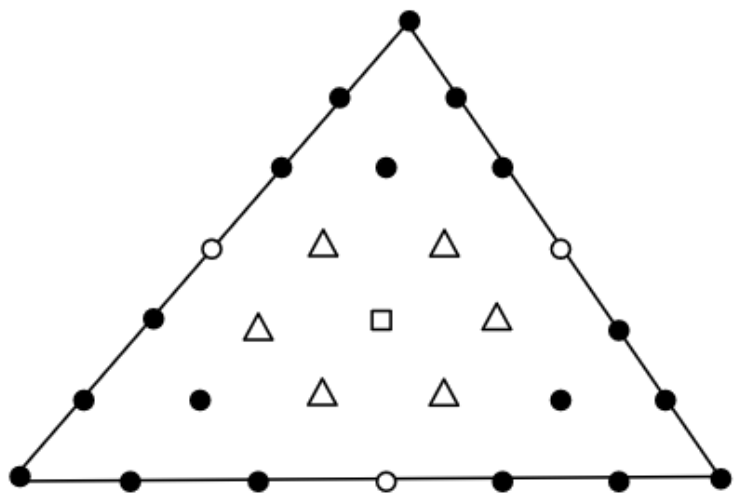


Figure 2.11: An example of a 6th order triangular Bézier patch with a total of 28 control points

CHAPTER 3
CONSTRUCTION OF HIGH-ORDER CURVED MESHES

In this chapter, techniques to construct high-order curved meshes are presented. In particular, methods to create high-order curved meshes with C^0 inter-element geometric continuity from linear straight-sided initial meshes are reviewed. A novel mesh curving technique is developed to create curved unstructured tetrahedral meshes where G^1 surface continuity is maintained for the triangular element faces representing the curved domain surfaces. A bottom-up curving approach is implemented to support geometric models with multiple surface patches where either C^0 or G^1 geometry continuity between patches is desired. Before getting into the details of curved meshing algorithms, it is important to briefly review the parallel unstructured mesh infrastructure (PUMI) which plays a critical role in supporting the desired mesh operations.

3.1 Unstructured Mesh Infrastructure

The operations to construct curved unstructured meshes are dependent on having a topology based mesh database that is able to communicate with the geometric model and answers basic queries about mesh entities regarding their topological adjacency, model entity classification as well as geometrical shape information. Such information in general is not maintained in the classical finite element data structure that stores only grid and connectivity information [51, 10]. The Parallel Unstructured Mesh Infrastructure (PUMI) [5] developed at the Scientific Computation Research Center (SCOREC) of Rensselaer Polytechnic Institute (RPI) addresses such a need by supporting the topological representation of unstructured meshes. PUMI consists of a collection of software component packages including a flexible distributed topology-based mesh database (referred to as FMDB). FMDB is a distributed mesh data management system that is capable of shaping its data structure dynamically based on the user's requested mesh representation [52, 53]. FMDB has a set of desirable features to support the mesh curving developments of

this work and the most important features are reviewed in sub-sections that follow. PUMI is being used in various scientific and engineering applications as a mesh data structure running underneath effectively supporting parallel mesh adaptive loop.

3.1.1 Topology-Based Data Structure

Different from conventional data structure for finite element analysis, the unstructured meshes stored in FMDB are effectively described using a topology based representation in which the members of the hierarchy of topological entities of regions, faces, edges and vertices are defined plus adjacency that describe how the mesh entities connect to each other [4, 52]. Figure 3.1 illustrates the supported topological mesh entities and their adjacency types for the one-level complete mesh representation. Two important features of the topology-based mesh data structure are critical in supporting the mesh curving developed in this work.

First is mesh classification against the geometric model which determines the geometric shape for the curved mesh entities. Figure 3.2 illustrates a model domain consisting of topological model entities and the corresponding mesh classification association.

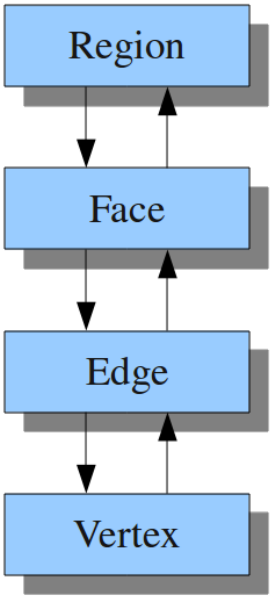


Figure 3.1: Topological mesh entity types and their adjacency [4]

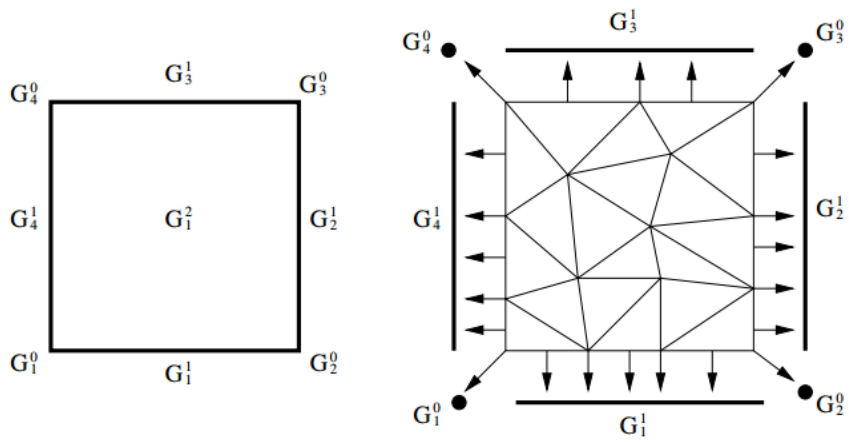


Figure 3.2: Illustration of mesh classification [5]

Second is adjacency describing how topological entities connect to each other. Having adjacency information supports effective validity checking for curved elements and determining the optimal mesh modification operation to fix a mesh invalidity. FMDB is able to efficiently support various procedures of mesh generation and adaptation such as entity creation and delete, iteration over a set of mesh entities, adjacency queries and user data attachment, etc.

3.1.2 Distributed Data Structure

FMDB is also a distributed mesh database that effectively maintains the mesh representation across multiple processors in the parallel computing environment. It has the functionality to efficiently retrieve mesh data, synchronize information shared by multiple mesh parts, redistribute the mesh by migrating groups of mesh entities from one processor to another. These are all essential to support the developments of parallel curved mesh creation and adaptation techniques. Figure 3.3 illustrates an example mesh with four partitions distributed among two processes. The dotted lines are intra-process part boundaries and thick solid lines are inter-process part boundaries between the processes representing mesh entities duplicated on multiple parts.

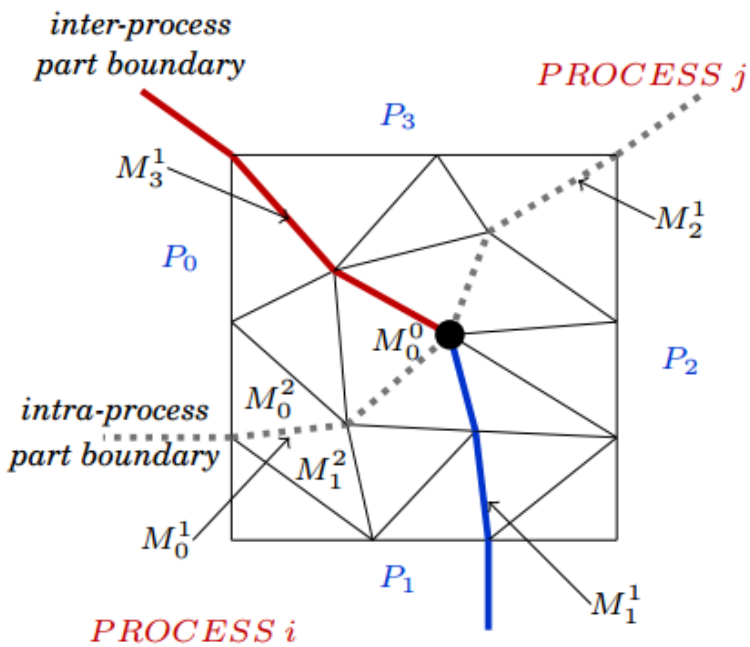


Figure 3.3: Illustration of mesh partitions [5]

3.1.3 High-Order Curved Mesh Entities in FMDB

Compared with a straight-sided mesh entity with linear geometric shape, a high-order curved mesh entity has additional data associated with its high-order geometric shape. For instance, a second-order Lagrange type of mesh edge needs one mid-edge node to define a quadratic geometry in addition to the two end vertices. Since such extra data does not affect the underlying mesh topology represented by the mesh database, the data is regarded as dynamic data components attached to the owning topological entities. The essential data members being stored are the Cartesian coordinates of a high-order node. When a high-order node is associated with a mesh entity at the geometric model boundary, its parametric coordinates with respect to the model entity is also stored by FMDB. Figure 3.4 shows an example in which two high-order nodes denoted by P and Q are attached to mesh edges M_0^1 and M_1^1. P is subsequently repositioned such as M_0^1 becomes a curved edge.

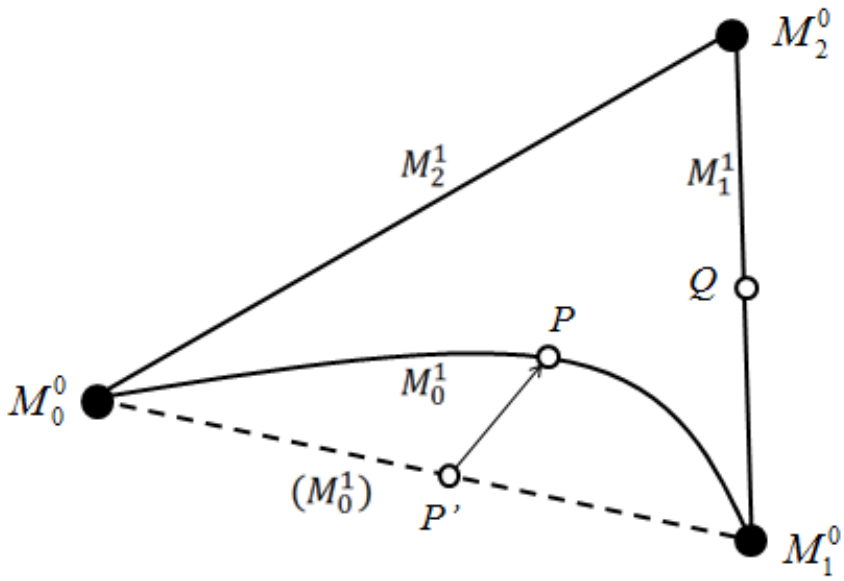


Figure 3.4: High-order nodes as attached data in FMDB

3.2 C^0 Mesh Curving

With the support of PUMI, the construction of C^0 curved meshes is achieved by developing an effective algorithm to perform local mesh modification to a selected entity, referred to as entity geometry modification. To convert an initially linear geometry mesh entity to a curved one, it is necessary to inflate the degree of the geometric shape functions by inserting high-order interpolating (or approximating) nodes that are then re-positioned to interpolate (or approximate) the geometric model boundary as desired.

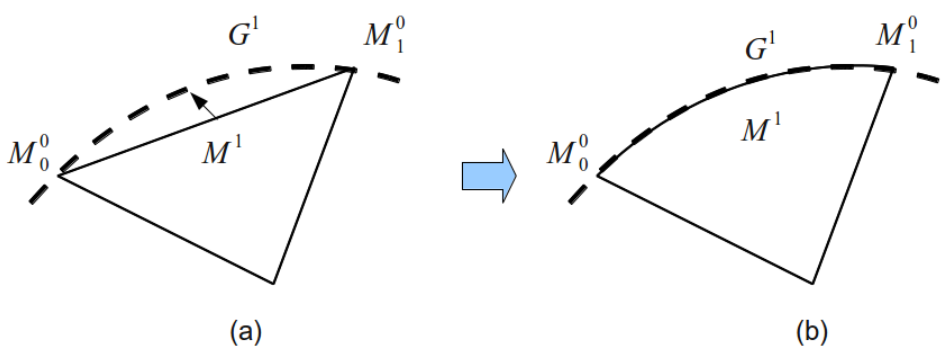


Figure 3.5: An example of curving a mesh edge to a model edge

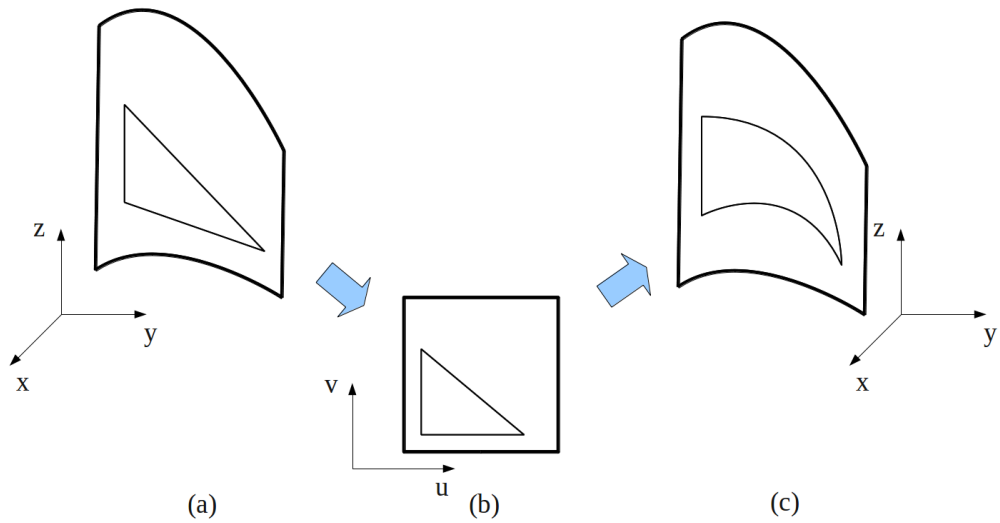


Figure 3.6: Curving boundary mesh entities by parametric interrogation. (a) is a straight-sided mesh face classified on a geometric model face in the physical space. (b) shows the mesh face in the 2D parametric space of the model face. The parametric coordinates of the edge mid-point are calculated in 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

Various types of re-positioning strategies have been proposed and a brief historical review can be found in Section 1.2. The work developed in this thesis extends upon a curving algorithm proposed by Dey et al [21] 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 re-parametrization of the boundary mesh entities using the underlying parametrization of the CAD model entities.

For example, as shown in Figure 3.5, 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, the physical position of the mid-point of the mesh edge is obtained.

$$\mathbf{x_2} = \mathbf{x}(\frac{\zeta_0 + \zeta_1}{2}) \tag{3.1}$$

Therefore a high-order node associated with the mesh edge can be introduced and placed to that location. Also see Figure 3.6 for an example of curving a planar mesh face on a curved model face.

As an implementation detail, PUMI includes a unified geometric modeling interface which maintains a high-level geometric model definition and supports interactions with two of the standard CAD system kernels – Acis [54] and Parasolid [55]. Figure 3.7 shows the inheritance diagram of an interface class GeomModel and its two concrete sub-classes that implement the interface functions. A GeomModel object is included in PUMIMesh as a data member.

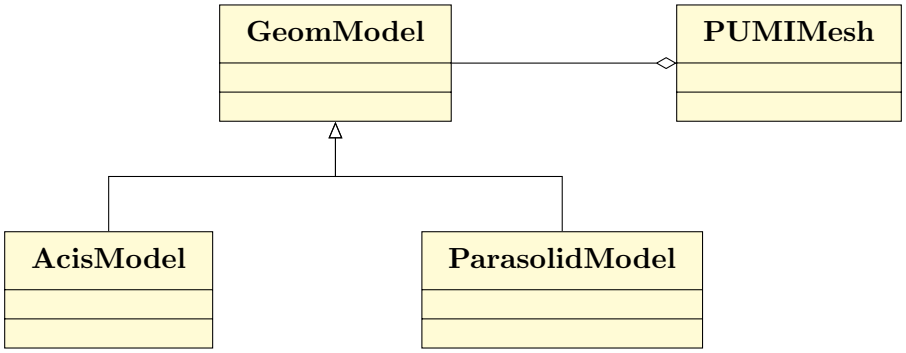


Figure 3.7: Collaboration relations among the geometric model and mesh classes

3.2.1 Nodal Interpolation Sets

When curving a mesh entity to a geometric shape represented by higher-than quadratic functions, the determination of proper nodal locations becomes an intriguing problem. The critical measure typically considered is minimization of the interpolation error between the actual function representing the shape of the curved mesh entity versus its finite dimensional interpolants. In the thesis, we focus our particular attention on the polynomial interpolation schemes.

To briefly define the interpolation problem, we give the following definition:

Consider the problem of interpolating a function $f(x)$ in the domain $\Omega = a \leq x \leq b$. Given a distinct set of points $\Pi = x_0, ..., x_m$, it is assumed that a polynomial function $g(x)$ exists such that

$$g(x_i) = f(x_i), \forall i, 0 \leq i \leq m. \tag{3.2}$$

This polynomial can be considered to be the interpolating polynomial such that

$$g(x_i) = \Xi_m f(x_i), \tag{3.3}$$

where Ξ_m is the interpolation operator. The set of points $\Pi = x_0, ..., x_m$ are referred to as the interpolation nodal set. Interpolation sets with the constraints $x_0 = -1, x_m = 1$ are called canonical interpolation sets. It is obvious that the canonical sets can be obtained from any interpolation sets by a linear transformation.

Based on the definition, the admissible interpolation nodal set for a given function is non-unique. In fact, the accuracy of the approximation is greatly influenced by the location of these nodes. Therefore, a useful way to measure a given set of nodes to determine whether its Lagrange polynomials are likely to provide good approximations is by means of the Lebesgue constant, which is formally defined as follows.

Denote a function h as the best possible polynomial function that approximates f in the maximum norm (also referred to as infinity norm), where if X is some vector

such that $X = (x_1, x_2, ..., x_n)$, then

$$\|X\|_\infty := max(|x_1|, ..., |x_n|). \tag{3.4}$$

we have

$$\|f - g\| \leq (1 + \Lambda) \|f - h\| \tag{3.5}$$

where, as mentioned earlier, g is an approximation to f found by fitting a polynomial through the m interpolation points. Λ is known as the Lebesgue constant.

Intuitively, Lebesgue constant measures how far the current nodal set deviate from the best possible set (which might not have a close form expression). As a result, the optimal interpolation nodal set can be defined as the one that has the minimal Lebesgue constant.

The convergence and interpolation accuracy of one dimensional interpolation sets have been extensively researched over the decades [27, 56]. Two factors have prominent impact on the quality of high degree polynomial interpolations: the smoothness of the function to be interpolated, and the locations of the interpolation points. Several sets are proposed and documented based on different choices of objective functions being optimized [27, 26, 56].

Runge [57] showed that polynomial interpolation on grids consisting of equally-spaced nodes may lead to unbounded and oscillatory interpolation of smooth functions as the interpolating polynomial degree increases, which is referred to as the Runge phenomenon. In fact, the Lebesgue constant of the interpolation operator for equally spaced point set increases exponentially with the degree of polynomial interpolation. Figure 3.8 shows an example of the phenomenon. The blue curve plots the Runge function defined in Eq 3.6. The green and red curves plot the polynomial interpolation of the Runge function using 5th and 9th order polynomial respectively based on equally spaced nodal points. The oscillatory behavior of high-order polynomial interpolation function can be clearly observed near the end points.

$$f(x) = \frac{1}{1 + 25x^2} \tag{3.6}$$

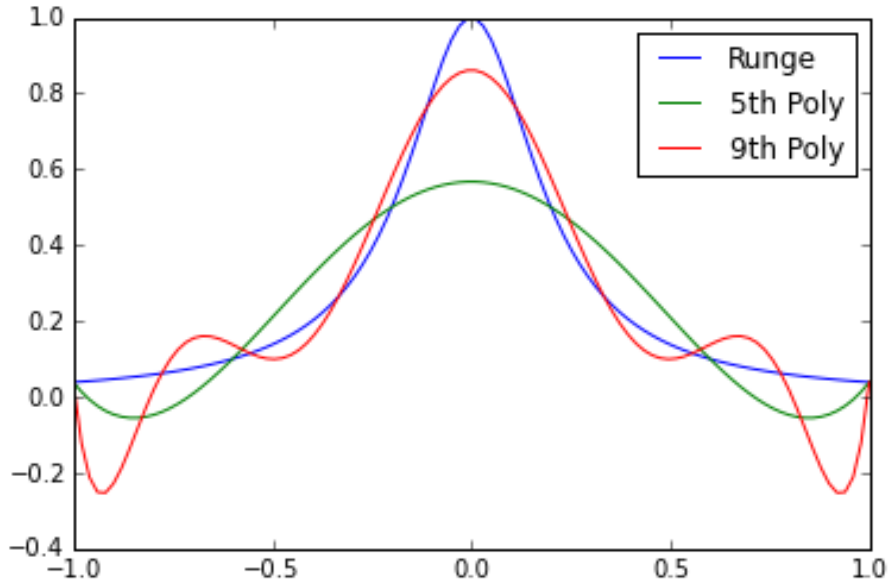


Figure 3.8: Plot of the Runge function and its polynomial interpolation functions

In one-dimension, two useful nodal sets are most commonly used based on orthogonal polynomials which eliminate the Runge phenomenon. The Gauss-Lobatto quadrature points which are determined by maximizing the determinant of the Vandermonde matrix [27, 56]. The nodal interpolation set proposed by Babuska and Chen are computed by minimizing the L_2 norm of the interpolation error [26].

Table 3.1 shows the parametric coordinates of several commonly used nodal sets compared with the equidistant point set. End points and mid-point coordinates are trivial therefore are not given in the table. Symmetric points are given only once by the positive valued coordinates.

Degree	Babuska-Chen	Gauss-Lobatto	Equal space
3	0.430664	0.447214	0.333333
4	0.636326	0.654654	0.500000
5	0.276518 0.748574	0.285232 0.765055	0.200000 0.600000

Table 3.1: Table of coordinates

Table 3.2 shows the Lebesgue constants of various commonly used nodal sets [56].

Degree	Babuska-Chen (B-C)	Gauss-Lobatto (G-L)	Equal space (EQ)
2	1.2500	1.2500	1.2500
3	1.4229	1.5000	1.6311
4	1.5595	1.6359	2.2078
5	1.6722	1.7786	3.1063
6	1.7681	1.8737	4.5493
7	1.8516	1.9724	6.9297

Table 3.2: Table of Lebesgue constants

A plot showing the growth of the Lebesgue constants as the polynomial degree increases is given in Figure 3.9. It can be observed that the B-C points have similar properties as the G-L points whereas the equally spaced points has the worst interpolation property as the Lebesgue constant grows exponentially as the polynomial degree increases. As a result, it is not given further consideration for the rest of the work.

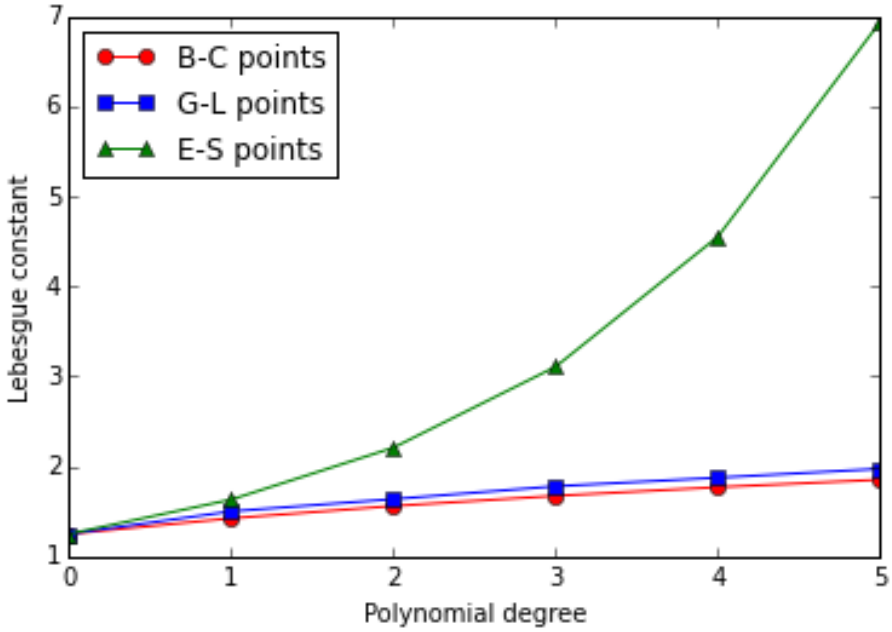


Figure 3.9: Lebesgue constants plot

The generalization of 1D sets to multi-dimensional domains are not as straightforward especially for the simplex element domains. Only a limited few sets are documented in the literature in which the points are computed in more than one dimension. The earliest is due to Bos [27], who derived Fekete points for the triangle

up to degree $d = 3$ and derived some approximate solutions up to degree 7. More recent work has been computational. Chen and Babuska [26] improved and extended Boss results to degree 13. They also computed optimal L_2-norm interpolation points and showed that these points have a lower Lebesgue constant than their approximate Fekete points. Taylor, Wingate and Vincent [28] presented Fekete points with smaller Lebesgue constant than both Fekete and minimal L_2-norm points in Chen and Babuska for degree $n > 10$. Hesthaven [56] computed a different set of near-optimal interpolation points for the triangle. For $d < 9$ these points are quite good, with a smaller Lebesgue constant than Fekete points and sometimes even the optimal L_2-norm points. This fails to be true for larger d, and for $d > 13$ these points become significantly worse than Fekete points.

Table 3.3 shows the Lebesgue constants of the nodal sets for 2D simplex [56].

Degree	Babuska-Chen (B-C)	Gauss-Lobatto (G-L)	Equal space (EQ)
2	1.6667	1.6667	1.6667
3	2.1115	2.1125	2.2698
4	2.6920	2.5878	3.4748
5	3.3010	3.1958	5.4522
6	3.7910	4.0752	8.7477
7	4.3908	4.7753	14.345

Table 3.3: Table of Lebesgue constants for 2D simplex

A plot of the Lebesgue constants of the nodal sets for 2D simplex is given in Figure 3.10.

3.2.2 Implementation of Interpolation Sets

In the work of this thesis, various interpolation nodal sets have been implemented using the object oriented programming (OOP) paradigm. OOP is a very powerful programming paradigm which allows for the design concepts such as encapsulation, abstraction, inheritance and polymorphism. In the case of our work, it allows abstract interfaces for the interpolation nodal sets to be defined in base classes so that objects can be manipulated using the base class interface, while the actual behavior of the objects of various types of nodal sets is dictated by the actual implementation of the sub-classes. This allows for a consistent definition of

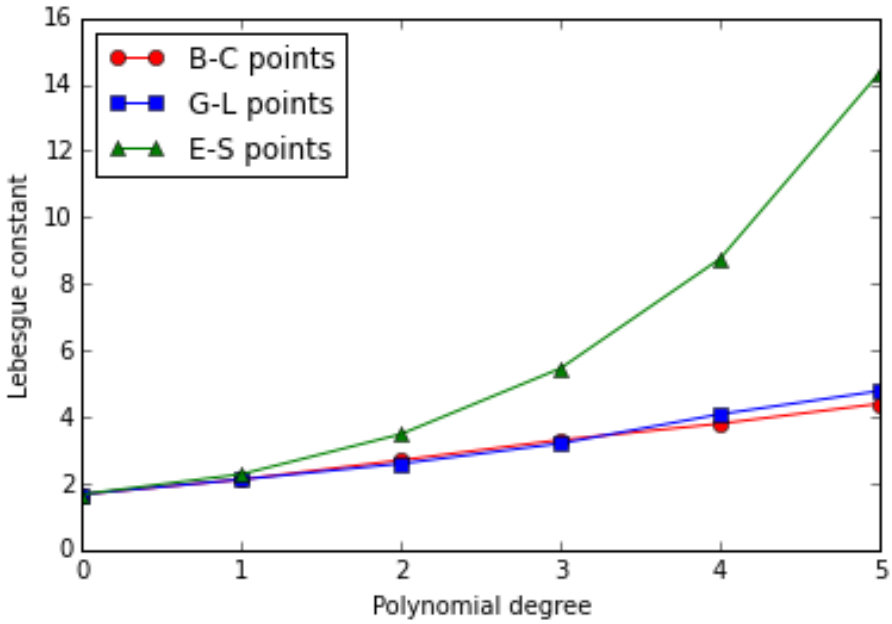


Figure 3.10: Plot of Lebesgue constants for 2D simplex

interface while making it extremely easy to extend new features. the notation used throughout this chapter to describe the object oriented design of the curved mesh classes (including class diagrams, interaction diagrams, etc) is based on the Unified Modeling Language (UML) [58].

Inheritance diagram for interpolation point package is shown in Figure 3.11. An abstract interface class is defined as IntpPt. Two sub-classes (i.e. 1D and multiD) are defined. It is desirable to separate 1D from multiD since the 1D points are well defined and can be used to define multiD points.

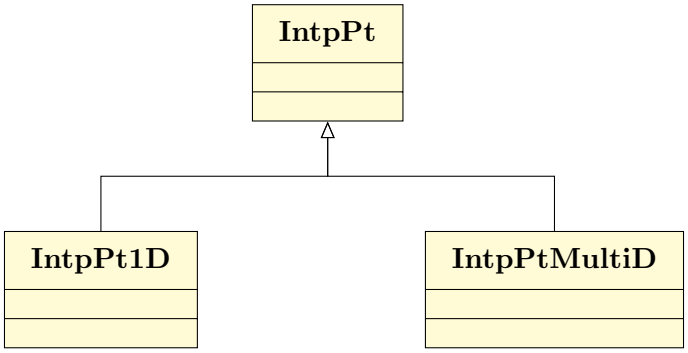


Figure 3.11: Class diagram for interpolation point classes

Various types of 1D interpolation nodal sets are implemented as sub-classes of

the IntpPt1D class as shown in Figure 3.12.

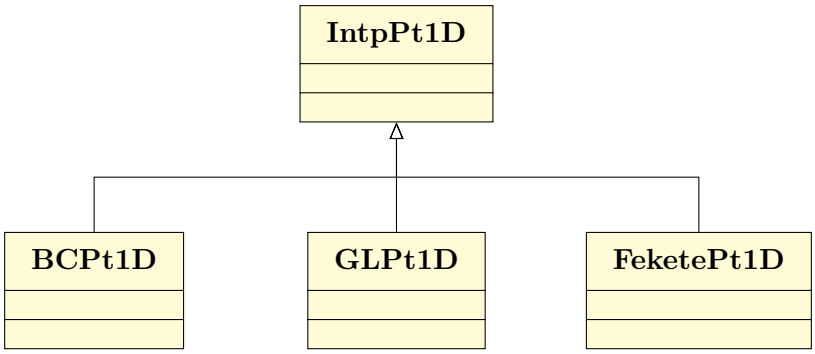


Figure 3.12: Class diagram for 1D interpolation nodal set classes

The multi-point interpolation classes are classified into specific table driven interpolation for simplex elements such as triangle and tet [26] as well as more general tensor product type of interpolation points for quad and hex elements [59]. Figure 3.13 shows the inheritance relations among various types of multi-point interpolation.

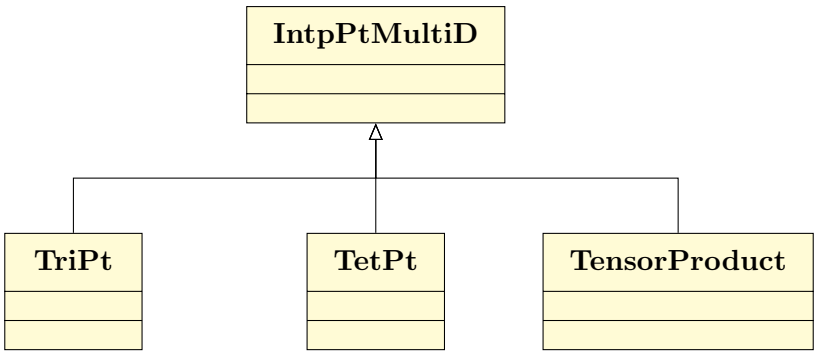


Figure 3.13: Class diagram for multi-dimensional interpolation point classes

Composition of TensorProduct and IntpPt1D classes is given in Figure 3.14. The Tensor product type of interpolation points consists one object of IntpPt1D class for each of the 2 or 3 dimensions.

3.3 G^1 Mesh Curving

In order to better support G^1 mesh curving, Bezier type of curve and triangle representation is relied on because of better control of slopes. Based on a set of nodal

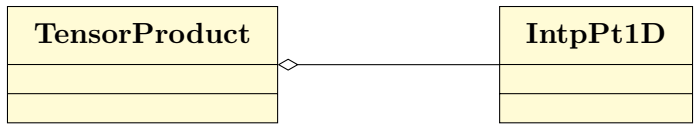


Figure 3.14: Composition diagram for 1D and multi-dimensional tensor product interpolation nodal set classes

interpolation set, the Lagrange polynomial representation is converted to Bezier representation by solving a linear equation system at specific interpolation nodes.

Recall that a Bezier curve can be represented as:

$$\mathbf{B}(t) = \sum_{i=0}^{n} b_i^{(n)}(t)\mathbf{P}_i^{(n)}, t \in [0,1] \tag{3.7}$$

For instance a 5th order Bezier curve can be expanded as

$$\begin{aligned}\mathbf{B}^{(5)}(t) = (1-t)^5\mathbf{P}_0 + 5t(1-t)^4\mathbf{P}_1 + 10t^2(1-t)^3\mathbf{P}_2 \\ + 10t^3(1-t)^2\mathbf{P}_3 + 5t^4(1-t)\mathbf{P}_4 + t^5\mathbf{P}_5\end{aligned} \tag{3.8}$$

Given a set of nodal interpolation points $\xi_i, i = 0, 1, ..., 5$ and the Bezier curve should interpolate the Lagrange polynomial function $\mathbf{L}(\xi_i)$ approximating the CAD geometry

$$\mathbf{B}^{(5)}(\xi_i) = \mathbf{L}_i, \quad i = 0, 1, ..., 5 \tag{3.9}$$

This leads to a system of $6*n$ linear equations where n is the spatial dimension.

$$\begin{bmatrix} (1-\xi_0)^5 & \cdots & \xi_0^5 \\ \vdots & \ddots & \vdots \\ (1-\xi_5)^5 & \cdots & \xi_5^5 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \vdots \\ \mathbf{P}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{L}_0 \\ \vdots \\ \mathbf{L}_5 \end{bmatrix} \tag{3.10}$$

As a result, the 6 control points $\mathbf{P}_i$ can be determined uniquely by solving the linear system. In actual implementation, the Eigen package is used to form and solve the linear system. Eigen is a C++ template library for linear algebra, matrices, vectors numerical solvers and related algorithms [60] .

3.3.1 Implementation Geometric Shapes and Curved Mesh Entities

A interface class CrvEnt is defined as an abstract class. It contains definition of a host of geometric shape related member functions that are shared by all types of curved mesh entities used in this work. An abbreviated version of the class declaration is given in Figure 3.15.

```
class CrvEnt {
 public:
  // constructer
  CrvEnt(pMeshEnt in_mesh_ent);
  // destructor
  virtual ~CrvEnt();

  /* coordinate system transformation */
  void get_xi_by_eta(Point3d in_eta, Point3d & out_xi);
  void get_eta_by_ceta(Point3d in_ceta, Point3d & out_eta);
  void get_xi_by_ceta(Point3d in_ceta, Point3d & out_xi);

  /* evaluation */
  void eval_by_xi(Point3d in, Point3d & out);
  void eval_by_eta(Point3d in, Point3d & out);
  void eval_by_coll_eta(Point3d in, Point3d & out);

  /* differentiate */
  void diff_by_xi(Point3d in, Mat3x3 & out);
  void diff_by_xi(Point3d in, int j, Point3d & out);
  void diff_by_eta(Point3d in, Mat3x3 & out);

  /* helper */
  int ent_type();

 private:
  // ordered set of mesh vertices
  VtxPtrVec m_vert_vec;
  // topological mesh entity
  pMeshEnt m_mesh_ent;
  /// the ID of the curved mesh entity
  int m_id;
};
```

Figure 3.15: CrvEnt interface class declaration

Subclasses of CrvEnt are defined to represent the mesh entity of different spatial dimensions namely 1D edges, 2D faces and 3D regions. Class diagram for curved mesh entity classes is given in Figure 3.16.

Similarly for the geometric shapes, an abstract interface class CrvGeom is

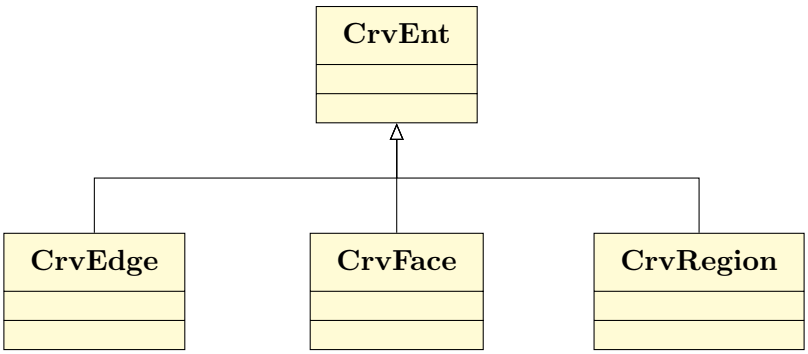


Figure 3.16: Class diagram for curved mesh entity classes

defined to represent the common geometric shape information and operations. A class diagram is given in Figure 3.17.

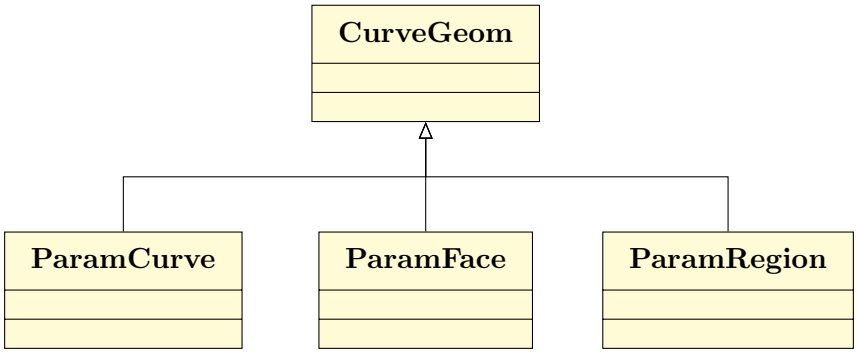


Figure 3.17: Class diagram for curved entity geometry classes

For the set of entities of given dimension, a set of sub-classes are defined which implement specific types of geometric representation. For instance, inheritance diagram for ParCurve is given in Figure 3.18. Sub-classes of ParCurve includes the Lagrange and Bezier curve types and the curve type that is defined by the CAD model representation.

As for a specific type of curve representation, for example Bezier curve, several concrete subclasses are defined and implemented to represent curves of specific orders. The inheritance diagram for BezCurve is given in Figure 3.19.

Similarly, as mentioned earlier, the parametric face representation class ParFace is a subclass of CrvGeom. And it has two sub-classes defined for different face topology, i.e. triangles and quadrilaterals. Specific concrete classes are defined and implemented for each face topology. For instance, the GrgTri class implements the

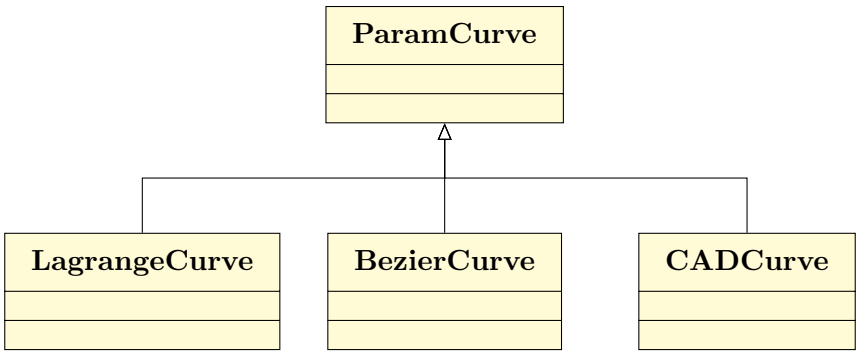


Figure 3.18: Class diagram for parametric curve classes

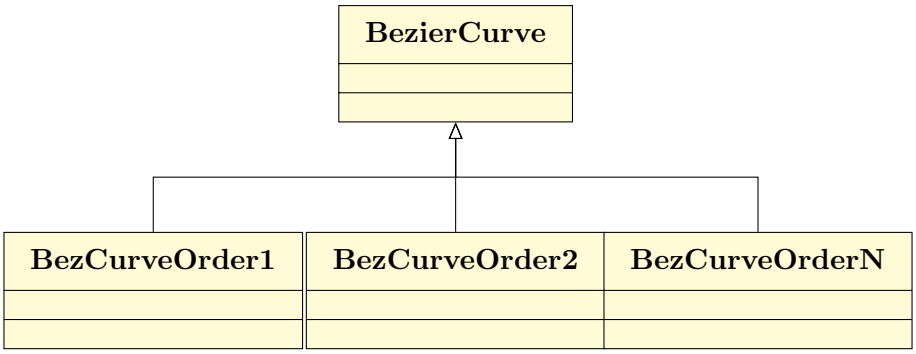


Figure 3.19: Class diagram for Bezier curve classes of various orders

triangular Gregory patch while the CnsQuad class implements the Coons patch. The inheritance diagram for ParFace is given in Figure 3.20.

More than one representation is implemented as the parametric triangular face patches. In crvMeshAdapt, triangular Lagrange, Bezier and Gregory patches are implemented. The inheritance diagram for ParTri is given in Figure 3.21.

To associate the geometric shape data with its corresponding mesh entity, each object of a curved mesh entity has a pointer to the its geometric shape object counterpart. The composition diagram of CrvTri and ParTri classes is given in Figure 3.22.

The CrvMesh class consists of a set of CrvEnt objects and a set of high level driver routines implemented to perform such procedures as mesh curving, invalidity correction and size-driven adaptation. The composition diagram of CrvMesh and CrvEnt classes is shown in Figure 3.23.

The CrvMesh class is also defined as a subclass of meshAdapt such that it

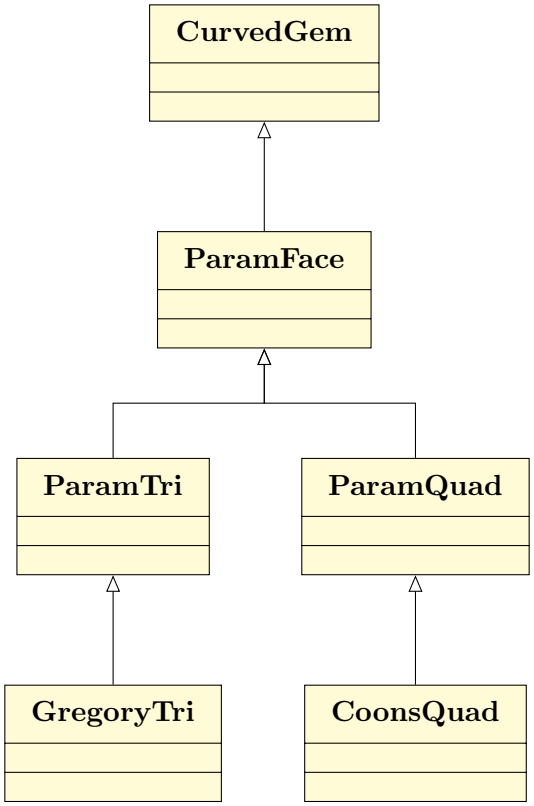


Figure 3.20: Class diagram for parametric face classes

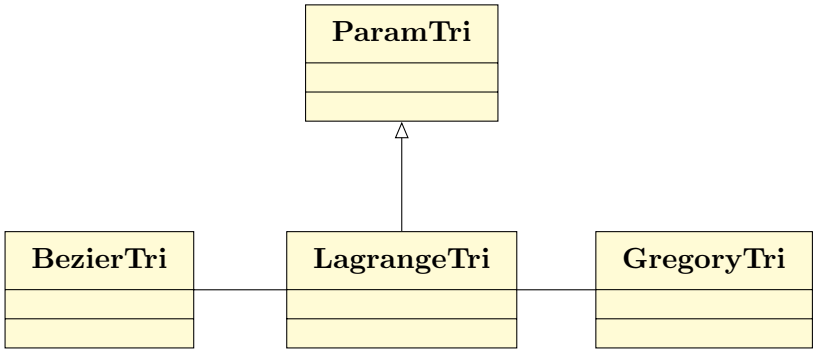


Figure 3.21: Class diagram for parametric triangular face classes

Figure 3.22: Composition diagram of CrvTri and ParTri classes

Figure 3.23: Composition diagram of CrvMesh and CrvEnt classes

inherits the existing functionality of straight-sided mesh adaptation procedure. Figure 3.24 shows the inheritance diagram.

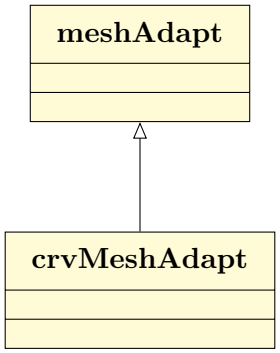


Figure 3.24: Inheritance diagram for meshAdapt and crvMeshAdapt classes

The definition of CrvMesh class is given in Figure 3.25.

3.3.2 G^1 Curving Algorithm Using Quartic Gregory Patches

In Section 2.2.2.6, the mathematical background of the Gregory patch was reviewed. The algorithmic steps to create a quartic G^1 Gregory patch is given in Algorithm 1.

```
class curveMesh : public meshAdapt{
public:
  curveMesh(pMeshMdl mesh_instance, int sizefield_type, int model_type)
    ;

  /// check the overall mesh quality
  int CMA_CheckMeshQuality();

  /// untangle a given curved mesh with a number of invalid elements
  int CMA_Untangle();

  /// optimize a given valid curved mesh
  int CMA_Optimize();

  /// make all mesh entities linear by removing edge nodes if there is
    any
  int CMA_Linearize();

  /// ceate a curved mesh with G1 surface mesh
  int CMA_CreateG1Mesh();

  /// ceate arbitrary degree C0 interpolating surface mesh
  int CMA_CreateC0Mesh(int iEdgeOrder, int iFaceOrder);

};
```

Figure 3.25: Class declaration of the driver level CrvMesh class

for *each edge* M_i^1 *of a triangle* **do**
  fit cubic Bezier curve ;
  compute tangent vector field ;
  compute cross-boundary tangent vector field ;
  degree elevate boundary curve to quartic ;
  determine the two interior face control points ;
end
for *each pair of two face control points* **do**
  apply linear blending function to get a Bezier type control point expression ;
end
write the quartic patch in Bezier form based on the 15 control points ;

Algorithm 1: Creating G^1 continuous surface triangles using Gregory patches

Concrete implementation of GrgTri class is given in Figure 3.26.

```
// Define the class that implements triangular Gregory patch of order n
class GrgTri {
  public:
    // ctor with ordered nodes defining coordinate
    // and directions
    GrgTri(std::vector<Point3d> in_ordered_nodes,
           std::vector<Point3d> in_ordered_normals);
    // dtor
    ~GrgTri();
    // Evaluate at parametric location
    int eval(double u, double v, double *retval);
    // Evaluate first derivatives
    void deriv1(Point2d in_xi,
                Point3d & out_dxyz_dxi1,
                Point3d & out_dxyz_dxi2);
    // main driver to setup the gregory patch
    void setup();
protected:
    // surface control point pairs
    Point3d m_G01_, m_G02_, m_G11_, m_G12_, m_G21_, m_G22_;
    // input data: position and normal vectors of corner control points
    Point3d m_q0_;
    Point3d m_q1_;
    Point3d m_q2_;
    Point3d m_n0_;
    Point3d m_n1_;
    Point3d m_n2_;
    // P -- edge control points
    Point3d P004;
    Point3d P013, P103;
    Point3d P022,       P202;
    Point3d P031,             P301;
    Point3d P040, P130, P220, P310, P400;
};
```

Figure 3.26: Class declaration for the triangular Gregory patch

3.3.3 Higher-Order G^1 Curving Algorithm Using Bezier Patches

Given a CAD model defined by the Acis or Parasolid CAD modeling engines, it is straight-forward to obtain data up to the second derivatives at given locations on the model face. Given the C^2 input data at mesh vertices on the model face, the first step uses the method in Farins quintic scheme to determine the 6 control points surrounding each mesh vertex as represented by black dots in Figure 3.27.

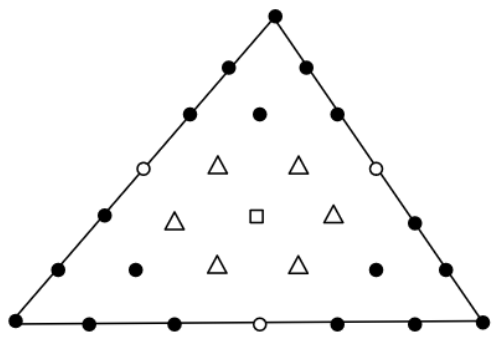


Figure 3.27: A 6th order triangular Bézier patch with a total of 28 control points

The mathematical equation to compute the control points is given in Eq 2.23 in Section 2.2.3. This step determines the position of a total of 18 control points.

The rest of the control points are set into 3 categories.

1. Edge control points that are responsible for interpolating C^0 data for each edge.

   In this 6th order patch example, these points are represented by hollow circles at the middle of the edges. They are determined by evaluating the parametrization of the mesh edge at the mid-point parametric location given by Eq 3.11.

$$\mathbf{P_i} = \mathbf{X}(\frac{\zeta_0 + \zeta_1}{2}) \tag{3.11}$$

   where P_i is the mid-edge point of the ith edge, and $\mathbf{X}(\zeta)$ is the parametric expression of the ith edge. ζ_0 and ζ_1 represent the parametric coordinates for the two end vertices.

   The tangent plane at the edge mid-points are also calculated by querying the first derivatives information of the underlying CAD face. The equations are given in Eq 2.15 and Eq 2.16 in Chapter 2. The tangent plane information is needed to set the face control points.

2. Face control points that are immediate neighbors of the edge control points (represented by triangles in Figure 3.27).

They are responsible for ensuring G1 continuity across the patch boundaries. They are determined by the tangent plane information at the edge mid-point, which has been obtained when setting the edge control points. The equations are given in Eq 2.18 and Eq 2.19 in Chapter 2.

3. The rest of the interior surface control points (represented by the square in Figure 3.27).

   They are can be determined by ensuring interpolation properties of the face parameter mapping at specific parametric location. Ref [16] proposes to solve the surface point location as an optimization problem to achieve high approximation accuracy by minimizing a surface potential energy functional.

The proposed method is capable of creating a Bézier patch of G^1 continuity to arbitrary level of polynomial order given that the under lying CAD representation of the domain can provide the needed input data which is up to the second derivatives.

3.3.4 Surface Mesh with Mixed C^0 and G^1 Continuity

The procedure introduced in section 2.2.2.6 serves the purpose of creating G^1 surface meshes for models with a single model face. In the mean time, most 3D models with challenging geometric features consist of more than one model face. Any procedure aiming to create proper surface meshes for such multi-patch models has to account for the mixture of C^0 and G^1 continuity. In this work, a bottom-up approach is adopted based on the different topological types of model entities on which a mesh entity is classified. Specifically, the mesh edges that represent the model edges where model faces join with C^0-continuity are curved first to be G^1 along the model edge direction while maintaining C^0 in the cross-edge direction. After that, the remaining surface mesh entities that represent the rest of the model boundary are curved using the procedure discussed in section 2.2.2.6. As a result, a piecewise G^1 surface mesh is created where it is G^1 within each model face as well as along the bounding model edges and C^0 in the cross-boundary direction at the model edges where model faces join together. Note that in the case where two model faces join with G^1 continuity in the first place, G^1 continuity is maintained

by curving the mesh edges representing the model edge in the same way as those representing model faces. The pseudo code for the overall procedure is given in Algorithm 2. Fig 3.28 shows an example mesh created using the algorithm. Fig 3.30 shows another example mesh based on a linear accelerator geometry.

```
for each edge M_i^1 in the mesh do
    if M_i^1 represents model edge with C^0 continuity then
        determine edge control points to interpolate model edge tangent;
    end
    if M_i^1 represents model face or edge with G^1 continuity then
        determine edge control points to interpolate model face normal;
    end
end
for each face M_i^2 in the mesh do
    if M_i^2 ⊏ G_j^2 then
        compute edge tangent vector t;
        compute cross-edge tangent vector g;
        determine face control points G_{i,j};
    end
end
```

Algorithm 2: Creating G^1 meshes for multi-patch CAD models

3.3.5 Geometric Interpolation Accuracy

To study and quantify the geometric interpolation properties of the quartic G^1 patch discussed in section 2.2.2.6, a set of numerical experiments have been conducted. A series of uniformly refined meshes are generated on a CAD model representing a cylinder. An example is shown in Fig 3.31. The distance between the mesh faces and CAD model faces is measured for each of the uniformly refined meshes. The distance is measured in terms of the Hausdorff norm which is commonly used to measure the distance between two parametric faces [61]. The definition of Hausdorff distance is given by Eq 3.12:

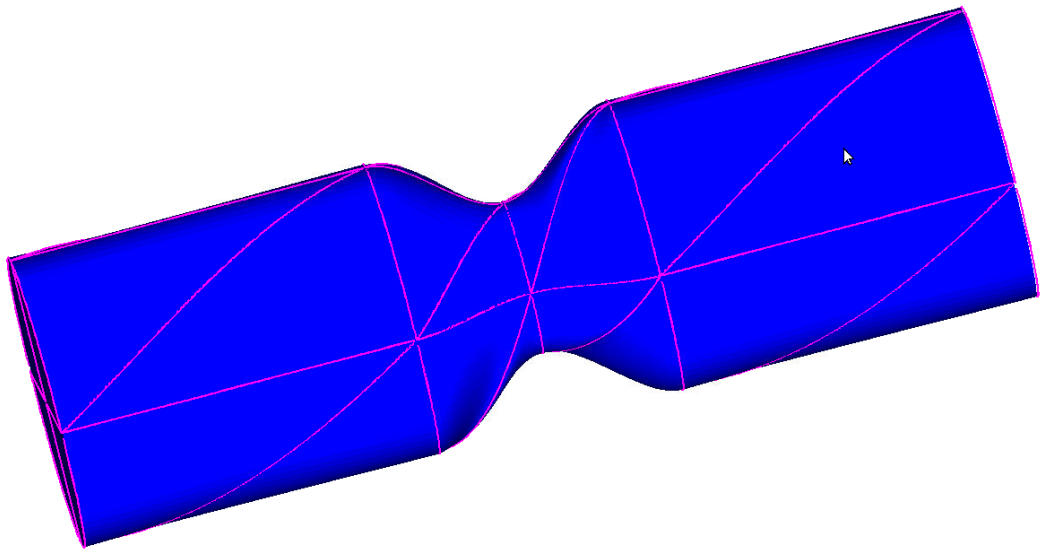


Figure 3.28: A tube model

$$d(S, S') = \max_{p \in S} \min_{p' \in S'} \|p - p'\|_2 \tag{3.12}$$

where S and S' are two sets of points representing two faces and p and p' denote points in the two sets respectively.

As a comparison, the measurement is done for both the G^1 meshes and a set of C^0 meshes using quartic Lagrange basis functions with optimal point distribution scheme proposed by Chen and Babuska [26]. Fig 3.32 shows the convergence plot generated from the distance data. For the quartic G^1 meshes, $4th$ order interpolation accuracy is observed, and for quartic C^0 meshes, it shows $5th$ order interpolation accuracy. It is a well known result in 1D that the order of accuracy for polynomial interpolation is $p+1$, where p is the highest complete polynomial order [2]. The one order difference in interpolation accuracy between the G^1 and C^0 is due to the fact that certain portion of the control points of the G^1 patch have to be constrained to ensure the higher surface continuity.

Table 3.4 gives the statistics for the cylinder model as well as the order of convergence for geometric interpolation accuracy.

Figure 3.33 shows another example which is a biological model representing a

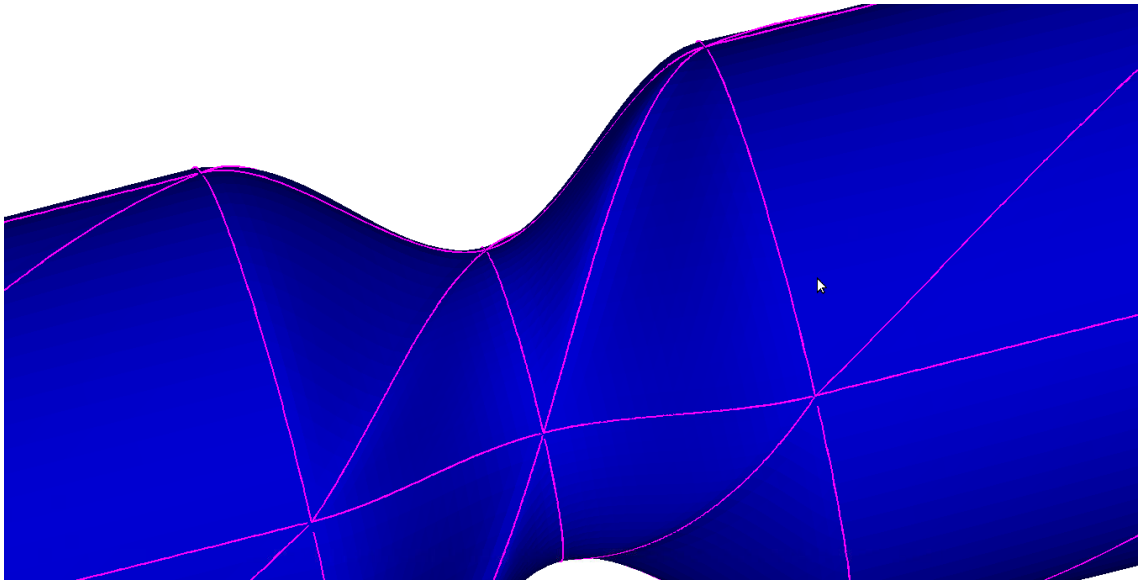


Figure 3.29: A close-up view of the tube

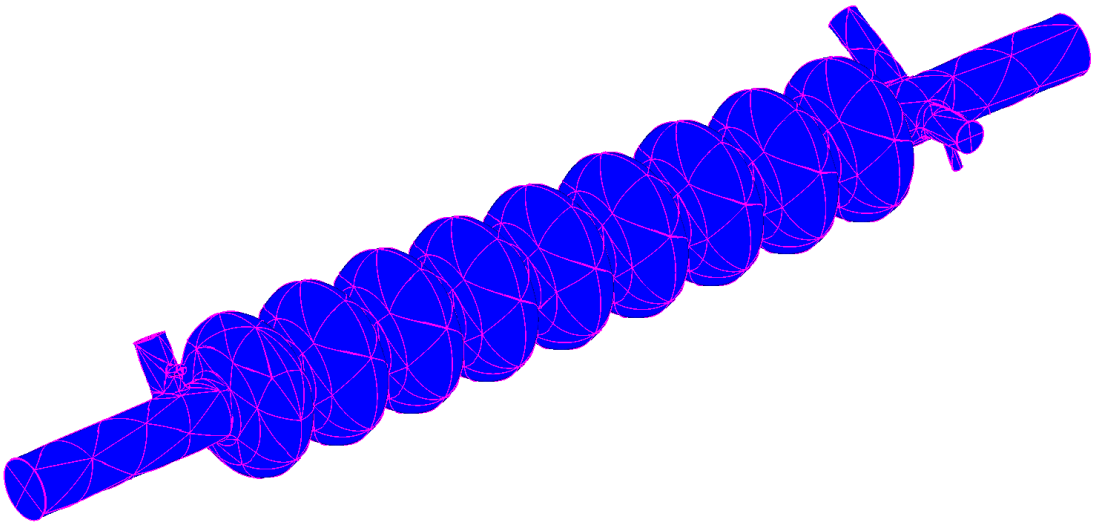


Figure 3.30: Curved G^1 mesh of a linear accelerator model

Number of element	mean distance	order
24	5.220E-3	n/a
228	3.175E-4	4.033
1840	2.056E-5	3.948
13720	1.399E-6	3.877

Table 3.4: Convergence data for meshes of the cylinder model

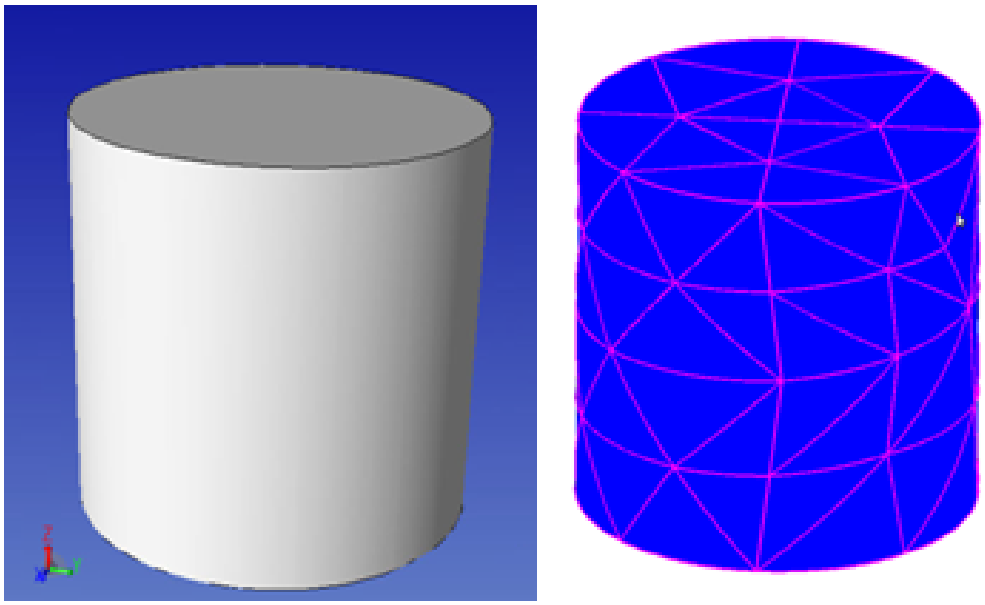


Figure 3.31: CAD model and quartic C0 mesh of a cylinder

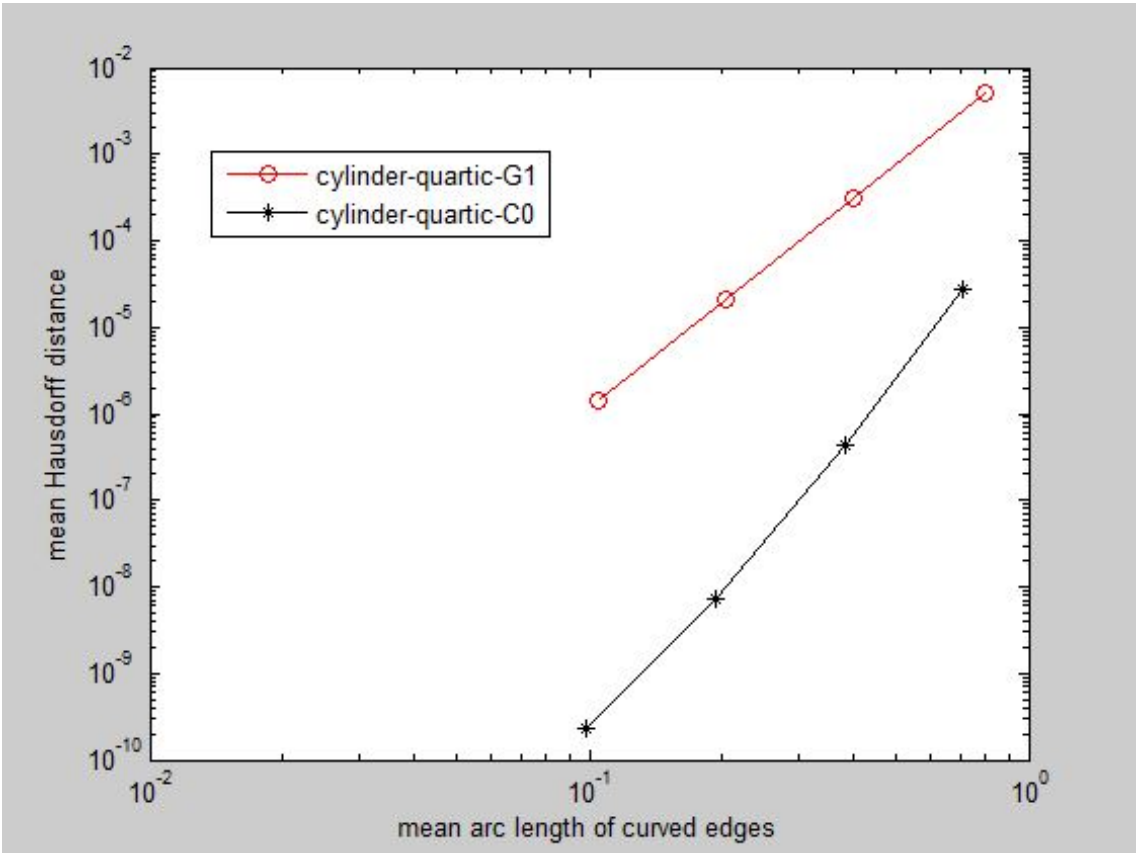


Figure 3.32: Convergence of geometric approximation error

portion of the porcine aorta.

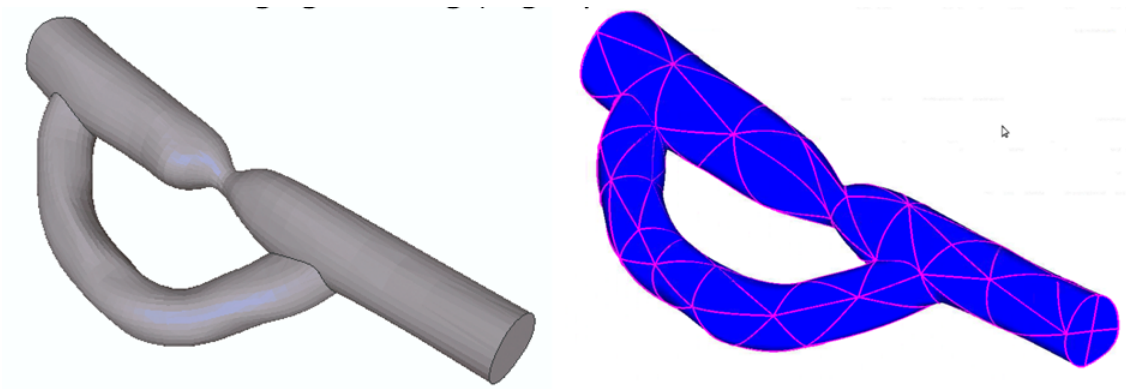


Figure 3.33: The CAD model and G^1 mesh of the porcine aorta model

Table 3.5 gives the statistics for the porcine aorta model as well as the order of convergence for geometric interpolation accuracy.

Number of element	mean distance	order
128	3.676E-1	n/a
1123	8.552E-2	2.104
9805	6.173E-3	3.809
77541	6.359E-4	3.261

Table 3.5: Convergence data for meshes of the porcine aorta model

Figure 3.34 gives the plot of distance against total number of elements used in the mesh.

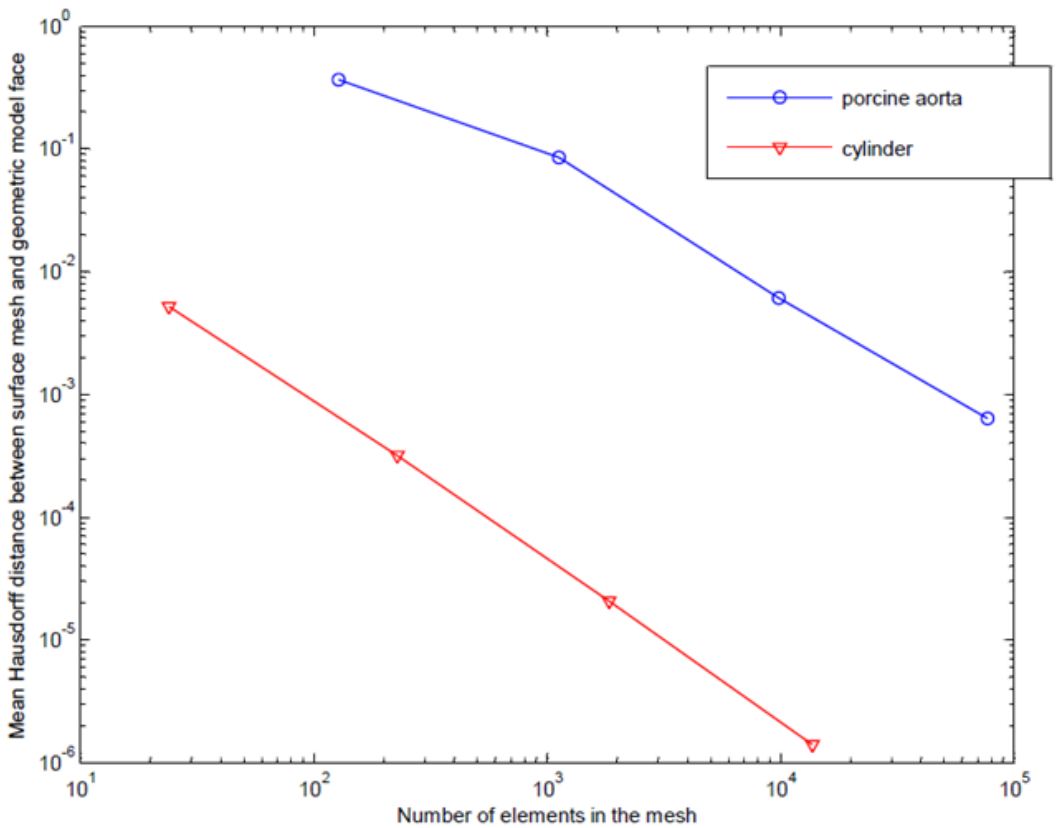


Figure 3.34: Convergence plot of number of elements v.s. distance

CHAPTER 4
MESH MODIFICATION AND ADAPTATION FOR CURVED MESHES

4.1 Introduction

A well designed mesh adaptation procedure provides the capability to modify the mesh size field while accounting for the model geometry. The majority of the research efforts on mesh adaptation have been focused on h-adaptivity dealing with low order meshes with all straight-sided elements. In order to achieve the full strength of the high-order methods, one needs to make use of hp-adaptive methods on curvilinear meshes where extra complexities arise when the mesh entities must be curved to the domain geometry. Mesh adaptation techniques for fully unstructured curved meshes are discussed in references [21, 22, 15, 17]. For simulations that require the numerical solutions to have extremely high accuracy, high mesh resolution is required in critical regions. Even when taking advantage of the benefits of adaptively refined meshes, element counts in the millions are common. Such meshes can only be created and analyzed using large scale parallel computing systems, which requires effective parallel mesh adaptation techniques [23].

A curved mesh adaptation procedure is presented in [8] which is designed for curved meshes on massively parallel computers. These procedures support curved quadratic C^0 meshes. The core procedure consists of two classes of mesh modification operations: entity geometry modification and local mesh modification for curved meshes. For the entity geometry modification, curved entity reshape operations that explicitly resolve element invalidity and improve the shape quality of curved elements are presented. The local mesh modification operations for curved meshes were extended from the operations for straight-sided meshes (see [62]) with

---

Portions of this chapter previously appeared as: Q. Lu, "Developments of Parallel Curved Meshing for High-Order Finite Element Simulations", Master's Thesis, Rensselaer Polytechnic Institute, Troy, NY, 2011.

additional consideration and treatment of curve boundary entities and selected curved interior entities. The parallel curved mesh adaptation technique is being used to support the automated adaptive accelerator simulations at SLAC National Accelerator Laboratory.

4.2 Curved Mesh Validity

A critical issue that needs to be addressed is mesh validity when curved meshes are used. The most common approach to construct curved meshes is to apply a straight-sided mesh generation procedure and then curve the mesh edges and faces on the curved domain boundaries to proper orders [63, 64]. This approach is able to take advantage of the conventional unstructured mesh generators to deal with the complexity of model geometry [65, 66]. However, the resulting meshes often become invalid because curving the straight-sided mesh entities to model boundaries can lead to negative determinant of Jacobian in the closure of curved elements [62, 36]. Effective and efficient correction of those invalid elements is critical in curvilinear mesh construction and for its usage with higher-order finite elements.

For example in the 2D case shown in Figure 4.1, 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 4.1(b)). In this case, the interior edges M_2^1 and M_3^1 can be curved (Figure 4.1(c)) to ensure element validity. The procedures for such element curving are referred to as entity geometry modification operations and the details of such operations are discussed in subsequent sections in this chapter.

A valid mesh entity requires a one-to-one mapping between the reference and physical element domains [63, 51]. This is ensured if the element's determinant of Jacobian that maps to the physical domain, $det(J)$, is strictly positive. As the mesh geometry becomes curved, the $det(J)$ is no longer a constant over the element domain as it used to be for a straight-sided simplex entity [10]. Commonly used methods, such as explicitly checking $det(J)$ at Gauss integration points, only provide necessary conditions for the validity of the mesh entity, and are computationally expensive to carry out when extremely high polynomial orders are concerned [67]. The convex hull properties of the Bezier form provides an alternative to evaluate

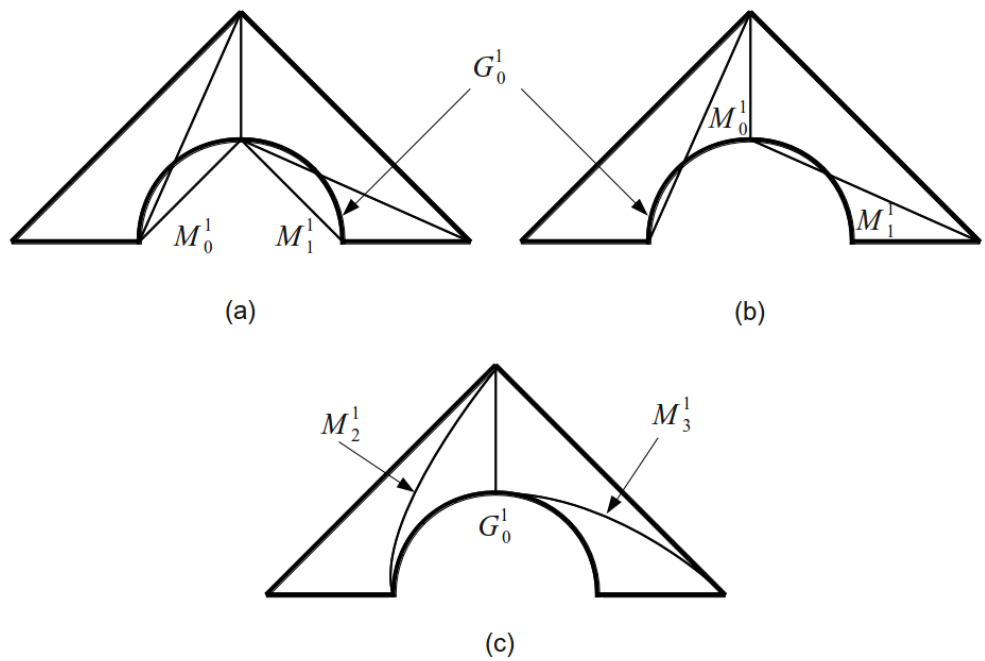


Figure 4.1: Invalid curved mesh and its correction

bounds of $det(J)$ efficiently [68, 69].

4.2.1 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 [70]:

$$r = \frac{\min_{i=1..6} l(M_i^1)}{\max_{j=1..6} l(M_j^1)}, \tag{4.1}$$

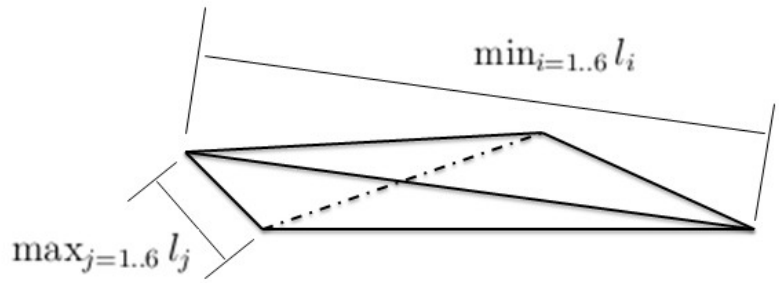


Figure 4.2: Definition of the edge ratio metric

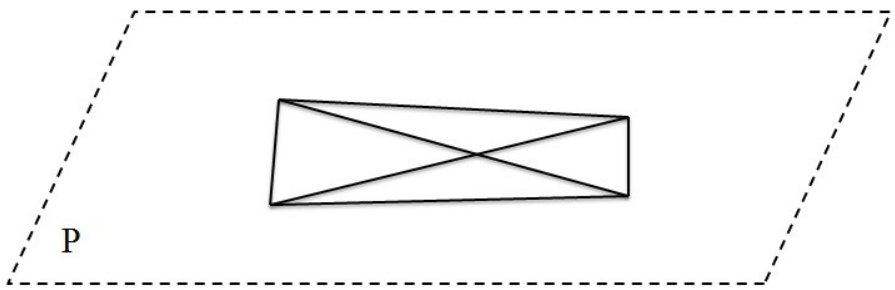


Figure 4.3: An example of a flat element in plane *P*

where l denotes the length of one of six edges. See Figure 4.2.

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 more 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 4.3. 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 [70]. It can be obtained by adding a perpendicular plane to the edge and measure the

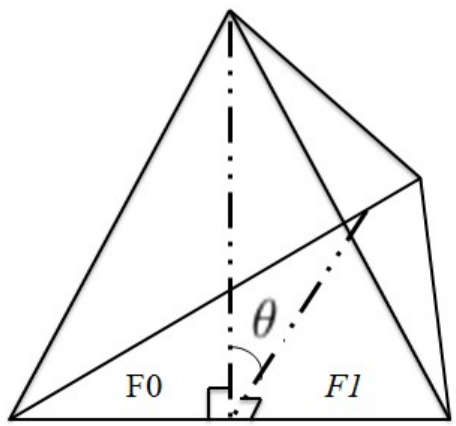


Figure 4.4: Definition of dihedral angle

angle between the two intersecting lines. See Figure 4.4.

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 [71] that non-dimensionalize the quantity by computing

$$q = \sin(\theta), \tag{4.2}$$

This metric targets at elements with small or large dihedral angles but does not detect needle-shaped tetrahedron as shown in Figure 4.5. 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.

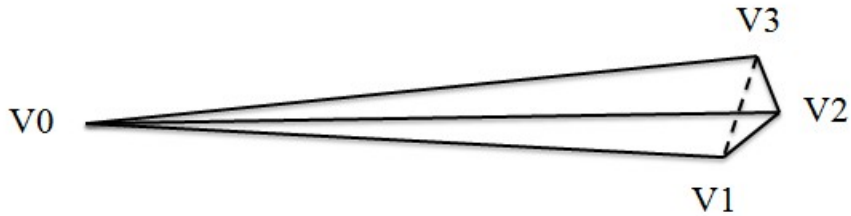


Figure 4.5: An example of needle-shaped straight-sided tetrahedron

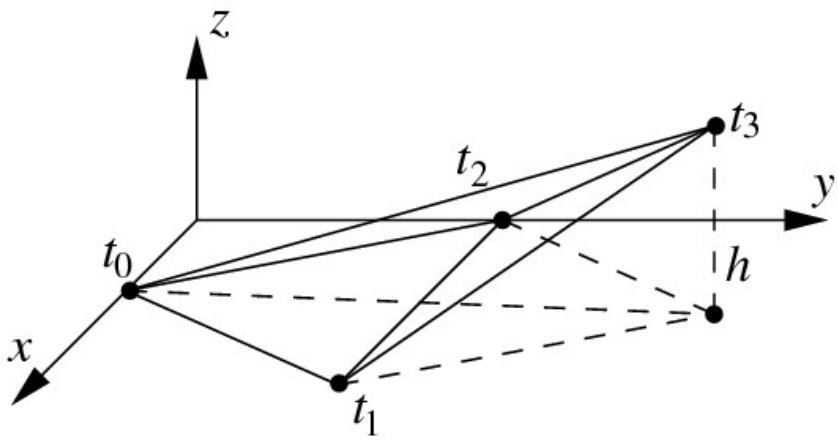


Figure 4.6: Definition of the aspect ratio metric

$$\rho = \frac{\sqrt{6}h_{min}}{2l_{max}} \tag{4.3}$$

where h_{min} is the smallest altitude, l_{max} is the longest edge length. See Figure 4.6.

Its normalized inverse form is also used as a quality metric and is discussed in [71]. 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 [72, 73, 70]. It is computed as:

$$\eta = K \frac{V(M^3)^2}{\sum_{i=1}^{6} l(M_i^1)^3} \tag{4.4}$$

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 [71].

4.2.2 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 cost of 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 [74] 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 will compromise solution accuracy and, in some cases, will cause the solver to halt [75, 76, 77]. 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 [75]. Such elements are categorized

as poorly-shaped.

Ruiz-Girones et al proposed a point-wise distortion measure for high-order curved tetrahedral elements [78]. The measure is defined as follows:

$$M(\xi) = \frac{\|J(\xi)\|^2}{n\sigma_\delta \|J(\xi)\|^{2/n}} \tag{4.5}$$

where $\|\cdot\|$ is the Frobenius norm of matrices, $\sigma = det(J)$ and σ_δ is a scaling factor given by:

$$\sigma_\delta = \frac{1}{2}(\sigma + \sqrt{\sigma^2 + 4\delta^2}) \tag{4.6}$$

It is desired to have value one when the Jacobian matrix is a rotation combined with an isotropic scaling, while the value approaches infinity when the Jacobian matrix is non-invertible. Therefore, the corresponding shape quality metric which ranges from (0, 1] is given as

$$\eta = \frac{1}{M(\xi)} \tag{4.7}$$

It is worth noting that the value of the metric is a function of the parametric coordinates of the curved element. It is not straight-forward to obtain its upper and lower bounds. A mesh optimization procedure is developed which is based on minimizing the distortion measure as well as a geometric accuracy measure based on computing L2 norm of the difference between the curved element geometry and the CAD model geometry [78].

A quality metrics for high-order curved tetrahedron which identifies poorly-shaped high-order curved elements by evaluating the scaled variations of the determinant of Jacobian over the element domain is defined as [75, 21, 79, 36]:

$$q_c = \frac{\min_{\xi \in \Omega^e} det(J(\xi))}{\max_{\xi \in \Omega^e} det(J(\xi))} \tag{4.8}$$

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.

This metric is also referred to as the scalded Jacobian metric, and has been adopted as a basis quality measure for various mesh untangling and optimization applications [79, 17, 80].

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.

4.3 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 [66, 8].

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:

$$Q_{sc} = q_s^m \times q_c^n \tag{4.9}$$

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 Eq 4.9 with respect to either of the component quality metrics,

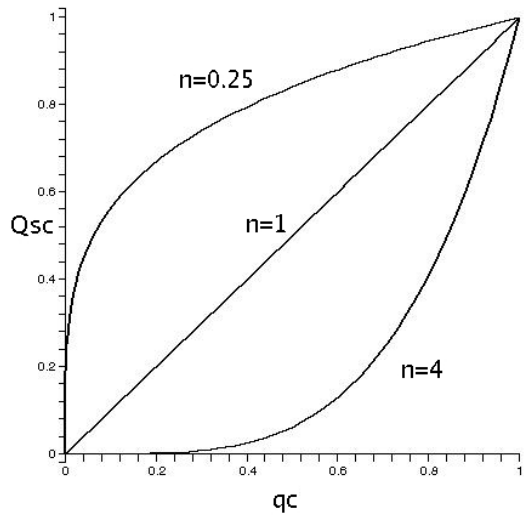


Figure 4.7: Plot of Q_{sc} with respect to q_c for different weighting constant n, assuming $q_s = 1$

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

$$Q_{,c} = \frac{\partial Q_{sc}}{\partial q_c} = C q_c^{n-1} \tag{4.10}$$

where $C = n q_s^m$.

Assuming $C > 0$, Eq 4.10 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 4.7 for examples of the three situations.

This quality metric serves as the basis to support the explicit nodal reposition-

ing procedure to identity poorly-shaped tetrahedral elements and improve element shape quality. In the present work, the parameters m and n are set to be $m = 1$ and $n = 1$, and the Mean Ratio measure is selected and implemented as the metric for straight-sided elements q_s.

4.3.1 The Validity Condition of Curved Element

To identify a valid curved tetrahedral element requires ensuring positive determinant of Jacobian throughout the domain of the element. Eq 4.11 gives the validity condition for an arbitrary order curved element in general.

$$det(J)|_{(\xi_1,\xi_2,\xi_3,\xi_4)} > 0 \quad \forall \xi \in \Omega^e \tag{4.11}$$

However, it is not feasible go through each and every point of the volume to check directly the value of determinant of Jacobian. In practice, positiveness of $det(J)$ is often checked at the set of integration points which depends upon the specific order of the element. An alternative method to ensure the element validity is 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 be represented as a Bézier polynomial of order $3(p-1)$ over the tetrahedron, where p is the order of the tetrahedral element [35]. As a result, it is bounded by the maximum and minimum values evaluated at the control points of the order $3(p-1)$ Bézier polynomial [36, 15]. In the case of a curved tetrahedron element of order p, the following inequality holds:

$$\min\{P_{|i|}^{(3(p-1))}\} \leq det(J) \leq \max\{P_{|i|}^{(3(p-1))}\} \tag{4.12}$$

where $P_{|i|}^{(3(p-1))}$ represents the value at the ith control point of the Bézier polynomial.

A sufficient condition to ensure positive determinant of Jacobian for a pth order curved tetrahedral element is that the lower bound of $det(J)$ given in Eq 4.12 is strictly positive. That is, the minimum of all control points $P_{|i|}^{(3(p-1))}$ is positive, as given in Eq 4.13.

$$\min\{P_{|i|}^{(3(p-1))}\} > 0 \tag{4.13}$$

Note that Eq 4.13 applies to any curved tetrahedral element of arbitrary order p.

4.4 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 determined by its order. In the case of cubic curved tetrahedral element, the Bézier polynomial representing $det(J)$ is of order $q = 6$. The total number of control points is 35 [69]. The computation is 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 [66, 65]. 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 $\min\{P_{|i|}^{(q)}\}$ is negative.

The level of conservativeness of the lower and upper bounds obtained by this validity check algorithm depends on the number of control points used to represent the polynomial with, 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 $\min\{P_{|i|}^{(q)}\} = \min\{det(J)\}$, while conservative if minimum is at non-interpolating points where $\min\{P_{|i|}^{(q)}\} \leq \min\{det(J)\}$. Details are discussed in Sections 4.4.1 and 4.4.2.

Note that the above stated algorithm can be generalized to apply to curved tetrahedral elements of arbitrary high-order, which still focuses at monitoring the minimum value of all control points of the corresponding Bézier polynomials.

4.4.1 $\min\{P^{(q)}_{|i|}\}$ at Interpolating Points

According to the convex hull property given by Eq 4.12, the minimum value among the control points is generally no greater than the actual minimum of the $det(J)$, i.e. $\min\{P^{(q)}_{|i|}\} \leq \min\{det(J)\}$. Also, the control points at the ends of a Bézier polynomial are interpolation points, which means $det(J) = P^{(q)}_{|i|}$ at these particular control points. This is true for any curved tetrahedral element of arbitrary order. Therefore, in the cases that $\min\{P^{(q)}_{|i|}\}$ is found at a mesh vertex control point, which is in term an interpolating point, the minimum determinant of Jacobian can be accurately obtained be to $\min\{det(J)\} = \min\{P^{(q)}_{|i|}\}$. In other words, if the value of $\min\{P^{(q)}_{|i|}\}$ 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.

4.4.2 $\min\{P^{(q)}_{|i|}\}$ at Non-interpolating Points

In the cases where $\min\{P^{(q)}_{|i|}\}$ is found at a non-interpolating point, e.g. edge/-face control point for a curved tetrahedron, the uniform validity check method becomes conservative for curved element geometry. Note that it is still accurate if the high-order tetrahedron is straight-sided in which case all the control points are interpolation points. If $\min\{P^{(q)}_{|i|}\} \geq 0$ for all control points of a particular tetrahedron, it is sufficient to determine that the element is valid according to the condition given in Eq 4.11 However in the cases where $\min\{P^{(q)}_{|i|}\} \leq 0$, one could not necessarily conclude that the tetrahedron is invalid. In fact in some applications, such cases have been reported that negative $\min\{P^{(q)}_{|i|}\}$ 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. Two adaptive refinement methods to evaluate tighter lower bounds for $det(J)$ are discussed in detail in Section 4.5.

4.5 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 [69, 35]. Thus either method can be used to obtain tighter bounds on the Jacobian evaluation. Taking advantages of this property, two approaches to

refine the uniform validity check method are proposed and studied.

It is worth mentioning that although the uniform validity check is 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 that is likely causing the invalidity.

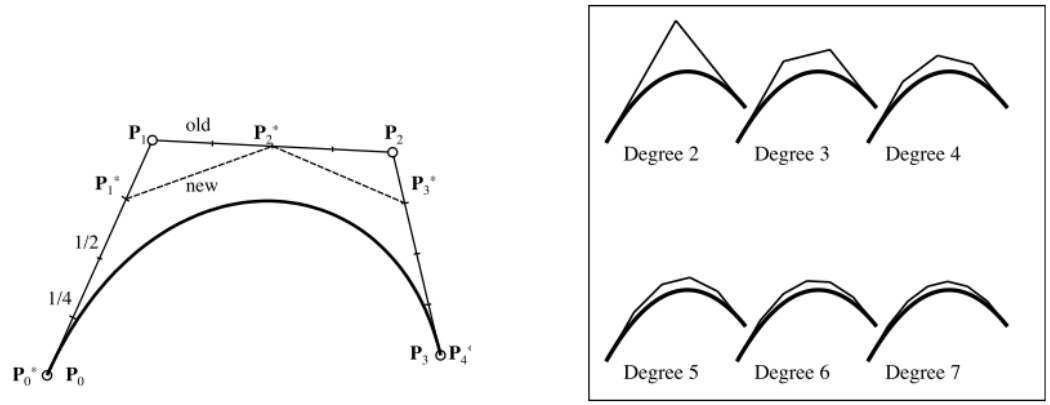
By determining the key entity, we can avoid doing degree elevation or subdivision uniformly to all the element entities. Instead, only the key entity of interest is elevated or subdivided in an appropriate manner.

4.5.1 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 tighter bounds. 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:

$$
\begin{aligned}
P_{|4000|}^{(4)} &= P_{|3000|}^{(3)} \\
P_{|3001|}^{(4)} &= \tfrac{1}{4}P_{|3000|}^{(3)} + \tfrac{3}{4}P_{|2001|}^{(3)} \\
P_{|2002|}^{(4)} &= \tfrac{2}{4}P_{|2001|}^{(3)} + \tfrac{2}{4}P_{|1002|}^{(3)} \\
P_{|1003|}^{(4)} &= \tfrac{3}{4}P_{|1002|}^{(3)} + \tfrac{1}{4}P_{|0003|}^{(3)} \\
P_{|0004|}^{(4)} &= P_{|0003|}^{(3)}
\end{aligned}
\tag{4.14}
$$

This can be generalized to obtain control points of any degree n elevated



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

Figure 4.8: Convergence of Degree Elevation

from degree $n-1$. According to [81, 82], the authors showed that the convergence rate is $O(\frac{1}{\nu})$, where ν is the polynomial order. A picture from [69] illustrates the convergence process of degree elevation to a 2D Bézier curve. See Figure 4.8.

As shown in the figure, as the polynomial degree is elevated, the number of control points increases and the control polygon becomes closer to the actual curve, and therefore gives tighter lower and upper bounds.

4.5.2 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* [35, 69]:

$$P_0^1 = \tfrac{1}{2}P^{(3)}_{|3000|} + \tfrac{1}{2}P^{(3)}_{|2001|}, \quad P_1^1 = \tfrac{1}{2}P^{(3)}_{|2001|} + \tfrac{1}{2}P^{(3)}_{|1002|}, \quad P_2^1 = \tfrac{1}{2}P^{(3)}_{|1002|} + \tfrac{1}{2}P^{(3)}_{|0003|},$$
$$P_0^2 = \tfrac{1}{2}P_0^1 + \tfrac{1}{2}P_1^1, \qquad P_1^2 = \tfrac{1}{2}P_1^1 + \tfrac{1}{2}P_2^1,$$
$$P_0^3 = \tfrac{1}{2}P_0^2 + \tfrac{1}{2}P_1^2 \tag{4.15}$$

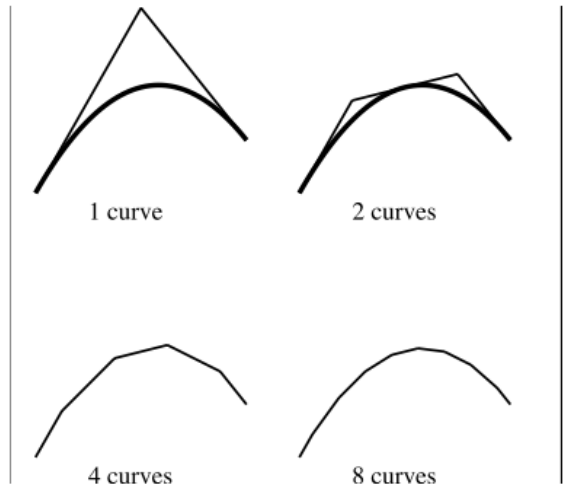


Figure 4.9: Convergence of Subdivision

The new sets of Control points for the two sub-polynomials are then: $\{P^{(3)}_{|3000|}$, P^1_0, P^2_0, $P^3_0\}$ and $\{P^3_0$, P^2_1, P^1_2, $P^{(3)}_{|0003|}\}$. 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 [69] gives an example of a 2D Bézier curve and its control polygons after several steps of subdivision. See Figure 4.9.

It is obvious that the control polygon gets closer to the curve after each step of subdivision, and according to [81, 82], it eventually converges to the curve with the rate of convergence $O(\frac{1}{2^i})$, where i is the number of subdivision steps.

4.5.3 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 $\min\{P^{(n)}_{|i|}\}$ is found to be positive, the element is valid. On the other hand, the element is invalid if negative $\min\{P^{(n)}_{|i|}\}$ is found at any 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 4.9 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 3. The input is a high-order curved tetrahedral mesh.

```
Data: A high-order 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^(n)_|i|} ;
if min{P^(n)_|i|} > 0 then
    return: the element is VALID ;
else
    if negative min{P^(n)_|i|} is at a interpolation point then
        return: the element is VALID ;
    else
        while relative increment > ϵ do
            get the mesh entity associated with the negative control point ;
            apply subdivision to the mesh entity ;
            update the new min{P^(n)_|i|} ;
            if min{P^(n)_|i|} > 0 then
                return: the element is VALID ;
            else
                if new negative min{P^(n)_|i|} is at a interpolation point then
                    return: the element is INVALID ;
                else
                    compute the relative increment of this step ;
                    if relative increment < ϵ then
                        return: the element is INVALID ;
                    else
                        update the new relative increment ;
                    end
                end
            end
        end
    end
end
```

Algorithm 3: Termination of the adaptive subdivision validity check method

4.6 Mesh Modification Operations for Curved Elements

In the section, a set of mesh modification operations designed for curved elements are presented. They can be organized into two categories, entity geometry modification operations and local mesh topology modification operations.

4.6.1 Entity Geometry Modifications

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. There has been extensive efforts devoted to develop various algorithms for the vertex repositioning operation, such as in [83, 84, 85, 67, 80, 79, 86].

One of the algorithms which is effective and inexpensive is the Constraint Laplacian smoothing method [83, 84]. It moves the target vertex to the centroid of its cavity defined as $M^0\{M^3\}$ under the constraint of the geometric approximation and improvement of the worst element shape. Given n mesh edges in a mesh cavity bounded by the vertex to be repositioned M_0^0, and the other end vertex M_0^n the target location can be evaluated as:

$$\mathbf{x_0} = \frac{\sum_{i=1}^{n} \mathbf{x_i}}{n} \tag{4.16}$$

where x_j is the Cartesian coordinates of the jth vertex M_0^j. 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.

This algorithm does not always improve the shape of some extremely poorly shaped elements [62]. In these cases an alternative vertex repositioning algorithm that guarantees a better overall mesh quality is the explicit vertex movement method. An effective design of such an algorithm has been introduced in [83]. This algorithm is computationally more demanding than the constrained Laplacian smoothing ap-

proach, and is only applied in cases where constrained Laplacian method is not able to yield desired results. The algorithm is summarized in Algorithm 4.

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 4: An explicit vertex repositioning algorithm introduced in [83]

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 [83], 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 some elements in a cavity degrade so long as the worst shaped element gets better and the overall quality get elevated.

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. Recently, the research interests of the community has been mostly in formulating the mesh curving problem in terms of a physical problem such as linear or non-linear elasticity of a deformed solid body [80], or in terms of optimization problems for some functional of given element quality or geometric accuracy measures [86, 79]. In [8], an entity reshaping algorithm has been developed for the curved mesh edges and faces to improve the element shape quality of high-order curved meshes. Pseudo code of the procedure is given in Algorithm 5.

```
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 Eq 4.9 ;
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 list for straight-sided shape improvement
      procedures ;
else
    compute curved shape quality q_c by Eq 4.8 ;
    get min{det(J(ξ))} and max{det(J(ξ))} ;
    find the mesh edges associated with min{det(J(ξ))} and
      max{det(J(ξ))} ;
    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 5: An explicit entity reshaping algorithm

The steps of the algorithm are as follows. The input to the algorithm is a list of poorly-shaped curved tetrahedrons whose shapes are evaluated by the hybrid element quality metric defined in Eq 4.9. In this present work, m and n in equation are taken to be 1.

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 Section 4.2.1. In the present work, the Mean Ratio measure is selected to compute q_s as given in Eq 4.4. 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 Eq 4.8. The explicit smoothing algorithm works independently of such choices. Since the Scaled Jacobian measure is used for calculating q_c, curving any edge or face of a given element will change the Jacobian evaluation. On the other hand, since q_s is a function of the vertex coordinates of a given element, curving any edge or face entity will not change the value of q_s as long as the vertices remain unchanged.

Step 2: Determine whether this tetrahedron should be considered for the curved entity reshaping operation. See lines 7 - 9 of Algorithm 5.

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 an appropriate candidate entity for reshaping. Based on the curved shape quality q_c defined in Eq 4.8, it is obvious that one should either increase $\min\{det(J(\xi))\}$ 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

the 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 4.10(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 of motion of the high-order node(s) to be on a straight line and consequently defines

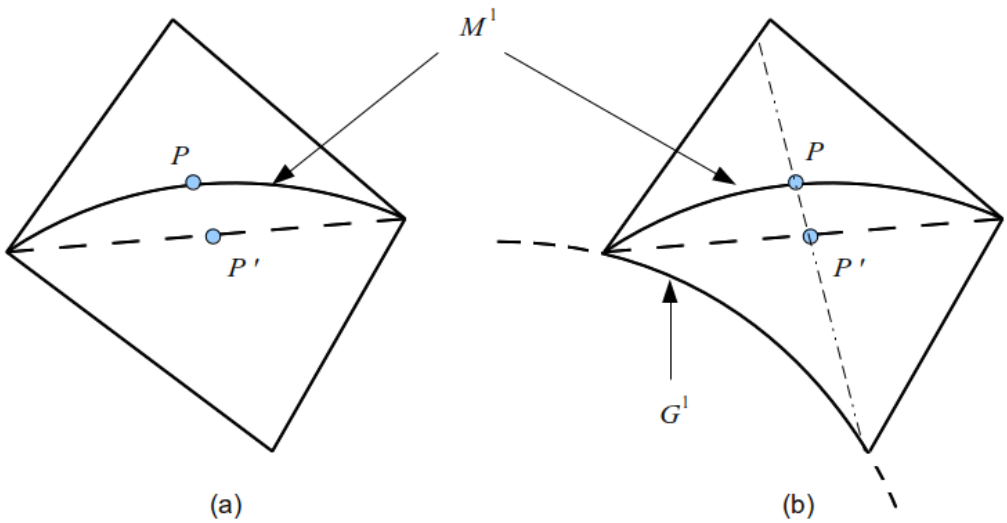


Figure 4.10: Two cases of 2D mesh edge reshaping. (a) without further geometric constraints, (b) one additional edge classified on model boundary G^1

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.

$$\mathbf{r} = \mathbf{P} + (\mathbf{P} - \mathbf{P}') \cdot t \tag{4.17}$$

See Figure 4.10(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 segment of the line of motion between intersections is considered as the interval of uncertainty for finding the local optimal position. For instance, the dash-dot line in Figure 4.10(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 search determined, one needs to find the optimal 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 [83] 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 6).

```
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 × l_0,
  P_3 = P_0 + 0.61803 × 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 > ϵ 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 × 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 × 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 6: Algorithm for the golden section search

When re-shaping curved mesh entities with higher-order than quadratic geometry, more than one high-order node may be allowed to move on a particular curved entity which increases the complexity and computational cost of solving the problem involved with determining the optimal location to be moved to. Fig 4.11 shows an example of an interior curved edge represented by a cubic Bezier geometry. While

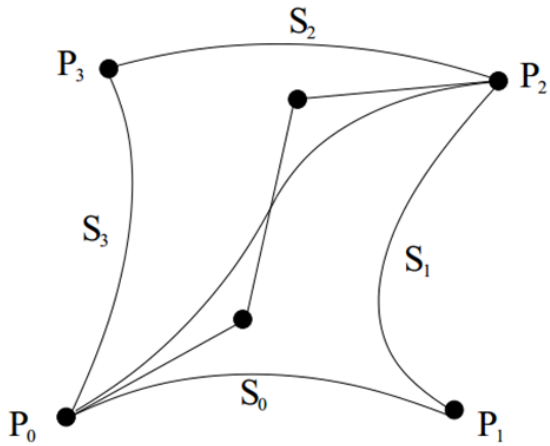


Figure 4.11: An example of an interior curved edge represented by a cubic Bezier geometry

certain heuristics [8] to determine acceptable positions for the high-order nodes can be applied to reshape element entities, it is desirable to take advantage of numerical optimization procedures that can explicitly target at improving a specific shape quality metric and obtain local optimal for the high-order nodes despite the potentially higher computational cost. One of the critical questions to be addressed is how to determine a proper objective function for the optimization problem. Since it is very likely that the chosen objective function is non-smooth, specific optimization algorithms such as the one proposed by Freitag et al [84] need to be considered.

One of the methods developed to curve high order interior mesh entities is to make use of blending based parametrization. See Fig 4.12. In this particular example, the interior edge is curved based on a Coons patch based parametrization [87, 88].

Given any 4 curves, $f(s,0)$, $f(s,1)$, $f(0,t)$, $f(1,t)$ that meet continuously at the corners, one can construct a smooth surface interpolating these curves. The mathematical expression is given in Eq 4.18.

$$C(s,t) = (1-t)f(s,0) + tf(s,1) + (1-s)f(0,t) + sf(1,t)$$
$$-(1-s)(1-t)f(0,0) - (1-s)tf(0,1) - s(1-t)f(1,0) - stf(1,1) \qquad (4.18)$$

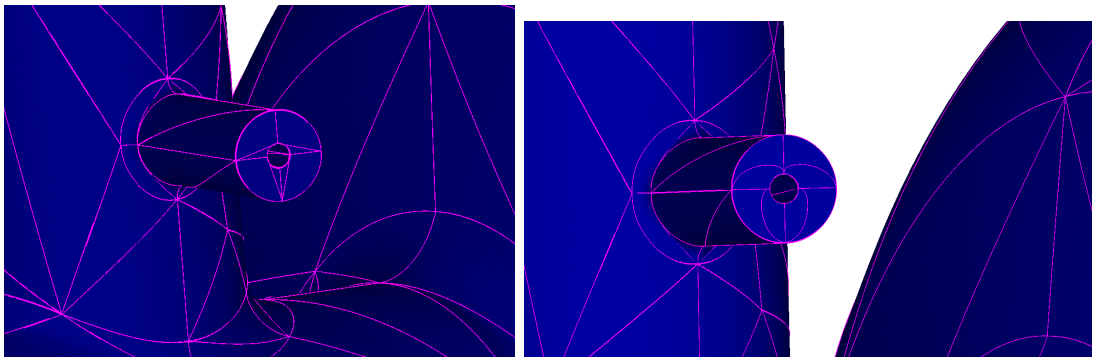


Figure 4.12: Example of curving mesh entities to fix invalid curved elements.

Figure 4.13 illustrates the boundary curves and surface rendering for a Coons patch. Algorithm 7 shows the steps to construct the Coons patch and compute the desired shape for the curved mesh entity.

Data: Candidate edge M_0^1 to be reshaped

1 get total number of faces, n, bounded by M_0^1 and are also classified on model surface G_0^1;

2 if n != 2, then non-manifold model edge, return error

3 get end vertices M_0^0 and M_1^0 that bound M_0^1

4 get the two triangle faces M_0^2 and M_1^2 bounded by M_0^1

5 for each triangle , get the third vertex M_2^0 (and M_3^0)

6 with $M_j^0, j = 0, 1, 2, 3$, and the underlying param coordinates, create Coons patch $C(\xi, \eta)$,

7 Curve edge M_0^1 using quadratic geometry: get the Cartesian coordinates of mid-edge node $P_0 = C(\frac{1}{2}, \frac{1}{2})$,

8 **if** *the element validity* **then**

9 | End of algorithm;

10 **end**

11 if not passed, curve edge M_0^1 using cubic geometry: get the Cartesian coordinates of the two high-order nodes $P_0 = C(\frac{1}{3}, \frac{1}{3}), P_1 = C(\frac{2}{3}, \frac{2}{3})$,

12 **if** *the elements are all validity* **then**

13 | End of algorithm;

14 **end**

15 **if** *there is any invalid curved elements* **then**

16 | Invoke explicit vertex smoothing to improve q_s;

17 | Repeat the algorithm from Step 1;

18 **end**

Algorithm 7: Edge curving based on Coons patch in the parametric space

A straight-sided edge M_0^1 to be curved is given as an input to the algorithm. Lines 1 - 2 check for the correct local mesh configuration. The algorithm expects one edge bounded by two triangular faces classified on a geometric model face. Non-manifold model edges are not supported by this algorithm. Lines 3 - 6 collects the four corner vertices, $M_j^0, j = 0, 1, 2, 3$, which forms a quadrilateral face in the parametric space of (ξ, η). Lines 7 - 10 try to create a curved edge with 2nd-order geometry, which is the lowest possible order. The validity of the resulting curved

elements are checked. If the elements are all valid, the algorithm then finishes. If there is any invalid element as a result of the 2nd-order edge geometry, then Lines 11 - 14 will increase the order of the geometry to cubic. By doing so, the curved edge has more degrees of freedom and is more likely to produce valid elements. If there is still any invalid curved elements as a result of the cubic edge, the algorithm will invoke explicit vertex smoothing procedure as discussed in Algorithm 4 to try to improve the straight-sided element quality of the local mesh cavity, which will in term create more space for the creation of a valid curved edge.

It is worth mentioning that while Algorithm 7 is designed for geometry modification of a curved mesh, it is still important for the overall mesh adaptation procedure to satisfy the requested mesh size field. More specifically, mesh refinements are always carried out in order to create elements as small as the requested size in certain regions, while coarsening is carried out when possible as requested in other regions. A set of curved mesh topology modification operations are designed to support the size field requirement.

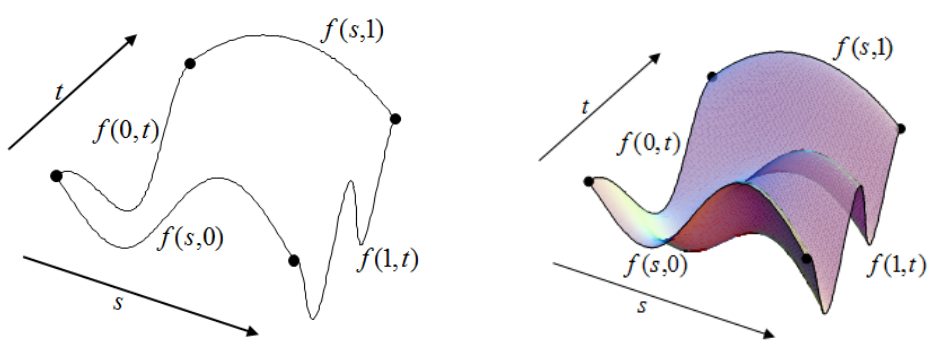


Figure 4.13: Boundary curves for a Coons patch

4.6.2 Local Mesh Topology Modification

While the entity geometry operations provide means to increase geometric approximation, local mesh topology modification operations are essential in changing mesh size to satisfy the requirement of a desired new size field. When designing curved local mesh modification operations for high-order curved meshes, a two-step process is employed. The mesh topology related modification is first processed as a

straight-sided local mesh modification followed by a curved entity reshaping operation. To generalize curved local topology modification operations, such as curved split, collapse and swap, to higher order mesh entities, the same 2-step process can be adopted. The curving step for interior entities uses the entity geometry modification operations discussed in Section 4.6.1. More specifically, the following steps are executed.

1. Test if a local mesh topology modification is applicable by applying checks to underlying straight-side element cavity.

2. If no topological violation is found, apply appropriate local mesh modification operations as if the cavity is straight-sided, otherwise end the curved mesh modification operation.

3. Perform initial curving of entities on model boundary. The curving stage leverages the techniques developed for high order mesh curving.

4. Check curved element validity

5. If an inverted element is detected due to curving, apply geometry modification operations, e.g. smoothing, to untangle. If no way to fix invalidity, revert the cavity to the status prior to applying any mesh modification.

4.7 Curved Mesh Adaptation Workflow

Given an initial curved mesh in parallel and a desired mesh size field, curved mesh adaptation in parallel produces a distributed curved mesh that satisfies the size field and preserves the geometric approximation to the right order. The procedure consists of three stages (see Algorithm 8): 1) curved mesh invalidity correction, 2) coarsening and iterative refinement, and 3) shape quality improvement. After the mesh adaptation, a load-balancing step is performed by using either Zoltan or ParMETIS. Fig 4.14 gives an illustration of the overall workflow.

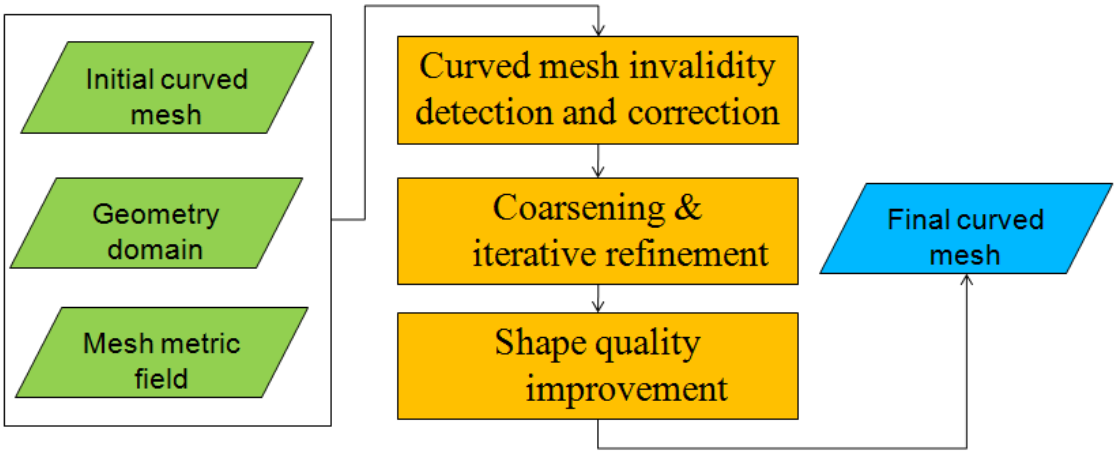


Figure 4.14: Illustration of curved mesh adaptation workflow

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 8: Overall algorithm of curved mesh adaptation

4.7.1 Invalidity Correction for Initial Curved Meshes

This stage eliminates all invalid elements prior to performing mesh modifications, since the mesh modification operations assume a valid starting mesh. The invalidity correction algorithm detects the invalid elements by the validity checks introduced in [8]. It chooses the proper local mesh modification and entity reshape operations to correct the invalidity of the curved elements. Details of the specific invalidity correction algorithm can be found in [65, 66].

4.7.2 Coarsening and Refinement

The coarsening process eliminates mesh edges that are shorter than the desired length specified by the size field [89]. It is accomplished by performing entity collapse operations on the identified short edges. Serial (on-part) curved entity collapse operation are introduced in [65]. Parallel entity collapse operation is discussed in [8]. When collapsing curved mesh entities classified on curved geometric model boundary, it is necessary to curve the new entities to conform to the boundary after collapsing. Curving the new entity may lead to intersection with another existing mesh entity which causes element invalidity. To identify such cases, curved mesh validity checks are always applied after a collapse operation. Invalidity is then resolved by the invalidity correction algorithm [65, 66] as discussed in the previous stage of this section.

After coarsening, the refinement algorithm incrementally reduces the edge lengths that are longer than the desired size field by as many iterations as needed to satisfy the local mesh metric. Entity split operations are applied to achieve the goal [89][65]. In the case that curved entities are to be refined, the new entities should be curved properly as well. The shape of the new entities are determined through parametric interrogations to the CAD modeler [21] and/or by calculation in the parametric space using Bézier parametrization of curved tetrahedrons [22][65][66]. Edge length is checked after each refinement iteration. Edge collapse operations are performed to eliminate the shorter-than-desired edges introduced by the refinement operations. In principle, some mesh edges may not be eliminated by collapsing due to local mesh invalidity that is unable to fix. In such infrequent cases, the col-

lapse operation is not performed. On the other hand, all curved mesh refinement operations are carried out in order to obtain the desired element sizes.

4.7.3 Curved Element Quality Improvement

After obtaining a mesh with the desired size field, this stage is conducted to further improve mesh quality by edge/face swap and/or curved entity reshape operations [89][65][66]. Details of the parallel swap operation is presented in [8]. The entity reshape operation for curved elements is extremely demanding in terms of computation costs due to the explicit search procedure. Thus, it is used only when swap operations fail to improve the curved element shape quality. Given a pre-specified element quality threshold Q_{th}, the elements whose shape quality $Q_{sc} < Q_{th}$ are collected to a list and are processed iteratively until either the list is empty or no further local mesh modification operations can be applied to improve the remaining elements in the list.

4.7.4 Parallel SPR Based Error Estimation

In a parallel adaptive finite element simulation, a posteriori error estimation and correction indication is an important component. A parallel error estimation procedure has been developed based on the Super-convergent Patch Recovery (SPR) scheme [90, 91]. In the error estimation procedure, a C^0 continuous gradient field is recovered from the original C^{-1} discontinuous field, and the error is defined as the difference between the recovered and original fields.

A key step of the recovery of a given nodal degree of freedom (associated with a mesh vertex, M_i^0) employs a local least-square fitting scheme over the patch of elements (mesh regions $\{M_i^0\{M^3\}\}$) surrounding the node (or the mesh vertex M_i^0). Therefore, the complete field information of the nodal patch is required. In the context of a parallel analysis based upon a distributed mesh, a set of mesh vertices is characterized as being located on mesh partition boundaries. Consequently, the corresponding nodal patch of such a vertex is distributed among several mesh partitions, thus not complete within a local part. In order to form a complete patch on a local mesh part, and in the meantime, to avoid extensive communication cost between mesh parts, the parallel SPR based error estimator has been developed to

take advantage of the ghosting functionality supported by the Flexible distributed Mesh DataBase (FMDB) as part of PUMI introduced in Chapter 3 [92].

Data: Distributed Mesh M, Solution field data S
Result: Calculate the error field, elemental error indicator and new desirable mesh size

1 Load the mesh and solution data, setup the nodal field correctly;
2 Create one layer of ghost regions with bridge vertices;
3 Conduct the on-part SPR procedure on M and S;
4 Calculate the error field, elemental error indicator and new desirable mesh size using Eq 4.19, 4.21 and 4.20;
5 Delete the ghost entities;

Algorithm 9: Parallel SPR Based Error Estimation with Ghosting

Algorithm 9 gives the steps for the error estimation procedure using ghosting. Step 1 loads the mesh and solution data and sets up the nodal field correctly. Step 2 creates one ghost layer of mesh regions on part boundaries. The field data attached to the mesh entities is also carried along with the ghosts. After the ghosting process is done, it is guaranteed that each mesh vertex on any local part (including the ones on part boundary) has a complete local patch of elements on the same mesh part. Therefore no extra inter-part communication is required to form the local nodal patch.

In Step 3, each local mesh partition including the ghosted entities is regarded as an independent serial mesh since further communication is no longer needed. For each non-ghosted mesh vertex on the local part, the SPR procedure recovers the C^0 gradient field ε^* based on the least-square fitting scheme over the complete nodal patch [93]. Note that the ghosted mesh vertices are not processed by the SPR procedure since they are essentially duplicated copies of certain non-ghosted entities on other mesh parts.

Step 4 calculates the error field, elemental error indicator and desirable mesh size. As the exact solution ϵ is unknown, the error e_ϵ can only be estimated. In this SPR based approach, the recovered solution ε^* is used to replace the exact solution. The error field computation equation is done using the following Equation 4.19.

$$e_\epsilon \approx e_\varepsilon^* = \varepsilon^* - \varepsilon^h \tag{4.19}$$

After the error field computation, the elemental error indication is carried out by integrating of the error over the element domain. The desirable mesh size is calculated through an h-adaptive procedure based on Equation 4.20 [93].

$$h_e^{new} = h_e^{current} \times r_e \tag{4.20}$$

where h_e^{new} and $h_e^{current}$ denote the new and current mesh sizes respectively. And the size scaling factor r_e is computed based on the Equation 4.21 [93].

$$r_e = \|e_\varepsilon\|_e^{\frac{2}{2p+d}} \frac{\eta^2 \|\varepsilon^*\|^2}{\sum_{i=1}^{n} \|e_\varepsilon\|_i^{\frac{2d}{2p+d}}}^{\frac{1}{2p}} \tag{4.21}$$

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

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

4.8 Examples

The Advanced Computations Department (ACD) of the SLAC National Accelerator Laboratory (SLAC) is developing a suite of high-order finite element procedures (ACE3P) for accelerator simulations that have demonstrated the ability to accurately model a variety of accelerator problems [65][68]. The level of discretization to obtain reliable predictions in ACE3P simulations often requires meshes with upwards of hundreds of millions of elements. To meet the requirements, the Scientific Computation Research Center (SCOREC) at RPI, in collaboration with Simmetrix Inc., is working with SLAC on providing the full range of parallel curved mesh generation and adaptation tools needed to work with the ACE3P simulation tools.

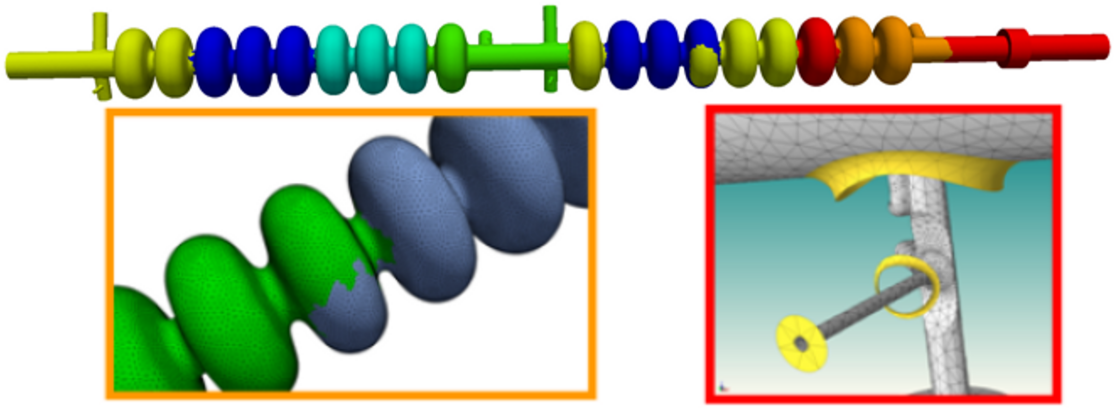


Figure 4.15: Overview and close-ups of a partitioned curved mesh of linear accelerator cavities

Fig 4.15 gives an example of a distributed curved mesh generated over a geometry of two linear accelerator cavities. The largest application so far has been a partitioned curved mesh of 180 million tetrahedral elements generated on 64 processors in less than 12 minutes (not counting I/O).

Fig 4.16 shows a small example of a parallel curved mesh refinement process with an isotropic analytic size field. The geometry is a linear accelerator cavity. The mesh to be refined on the left is of a relatively coarse global mesh size with finer mesh being generated locally at regions of large curvature. The parallel refinement focuses at the coarse mesh regions and brings the global mesh to a finer size while keeping the locally refined mesh regions unchanged.

Fig 4.17 gives an example of adapting an isotropic initial curved mesh to an anisotropic analytic size field representing a planar shock.

Fig 4.18 gives a set of pictures: (a) initial mesh of 8 parts, (b) solution field, (c) new size field and (d) 8-part adapted mesh of a pillbox model. (e) and (f) are close-up views of the region where relatively small mesh sizes are needed to get higher resolution and extensive curved mesh refinement is applied. The pictures present a complete iteration of the developed parallel adaptive loop. The initial mesh in this case has 22k elements and the mesh after adaptation has 106k elements. The wall clock time of mesh adaptation for this 8-part mesh in parallel is 6.73 seconds. The total time for the same adaptation process in serial is 22.89 seconds. Both the serial

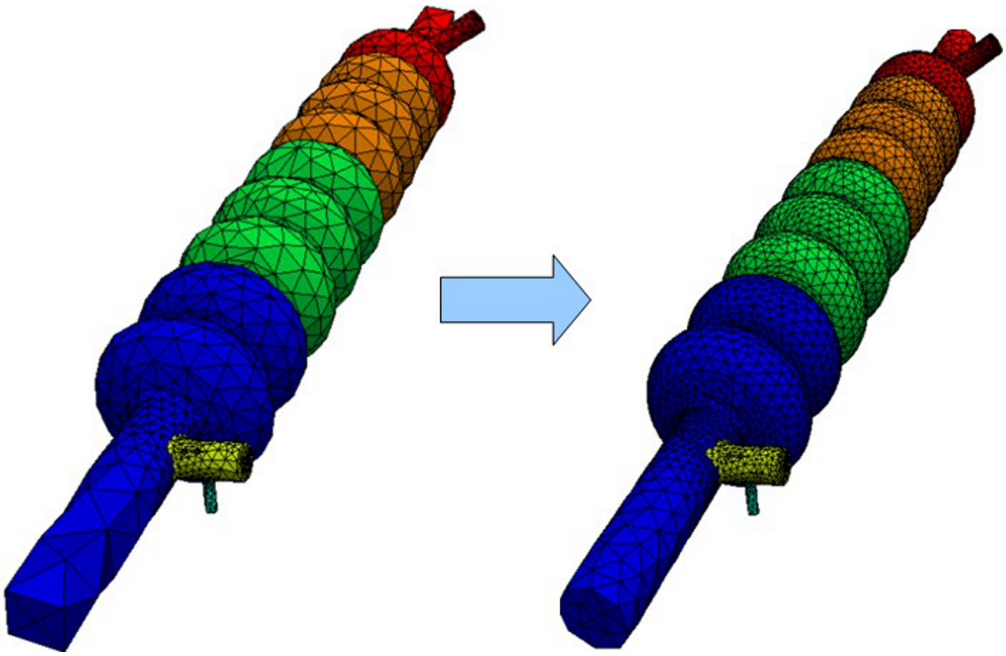


Figure 4.16: An example of parallel curved mesh refinement on a four part mesh

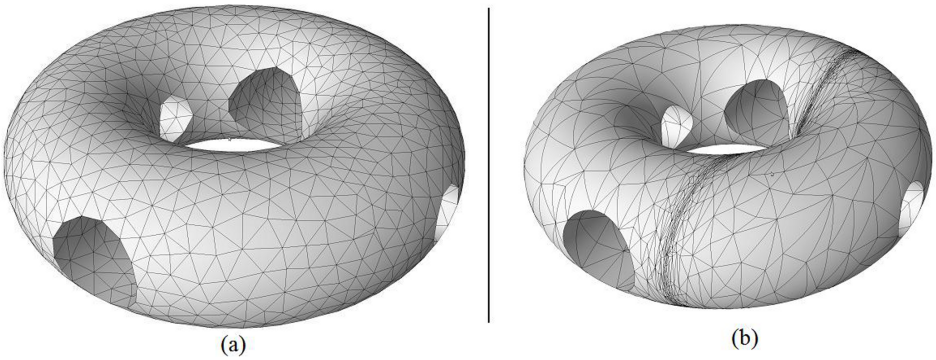


Figure 4.17: An example of curved mesh adaptation with an anisotropic size field

and parallel cases run on 2.3GHz Opteron processors.

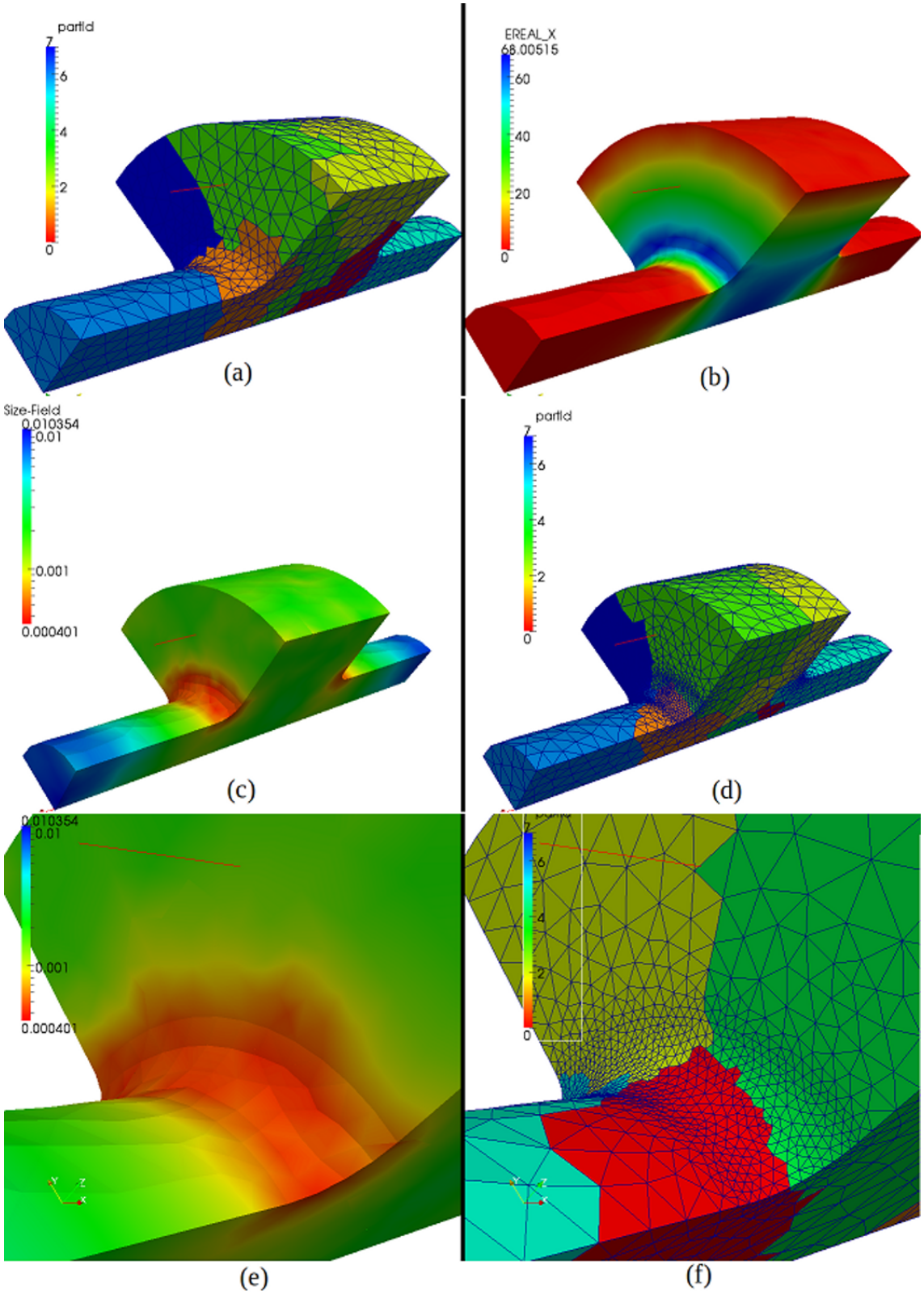


Figure 4.18: Example of one iteration of the parallel adaptive loop

CHAPTER 5
INTEGRATION WITH FINITE ELEMENT SOLVERS

5.1 Introduction

The work flow of an adaptive simulation starts with a definition of the problem domain of interest. In computer-aided design and engineering, the domain definition is typically a solid model constructed in a CAD system [35]. The analysis attributes (loads, boundary conditions, material properties) are specified with respect to the solid model. Initial mesh control attributes that guide the mesh generation process can also be specified with respect to the solid model [94]. Based on the meshing attributes, an initial mesh is generated with the required geometric approximation accuracy to the model. In the case of parallel simulations, load balancing is performed to maintain balanced distribution of workloads among the multiple processes. The finite element analysis procedure computes the solution fields of interest. To adaptively improve solution accuracy, error estimation and correction indication procedures are used to calculate a mesh size field, which is used to drive the mesh adaptation procedure to obtain an adapted mesh. After mesh adaptation parallel dynamic load balancing is applied and the finite element analysis procedure is performed, and a new set of solution fields can be obtained with improved resolution and accuracy [95]. The adaptive simulation loop continues until the desired solution accuracy is achieved as shown in Fig 5.1. Finally the results of the solution fields can be post-processed and visualized.

5.2 Geometric Mapping of Curved Finite Elements

A critical step in the finite element solution process is the evaluation of element contribution to the global matrix system, which involves the computation of geometric mapping of an element between the physical and parent domains. For the classical isoparametric elements in a C^0 mesh, the mapping between the parent space and the physical space is expressed by the same set of polynomial functions used as the basis functions for interpolating finite element solution fields. However,

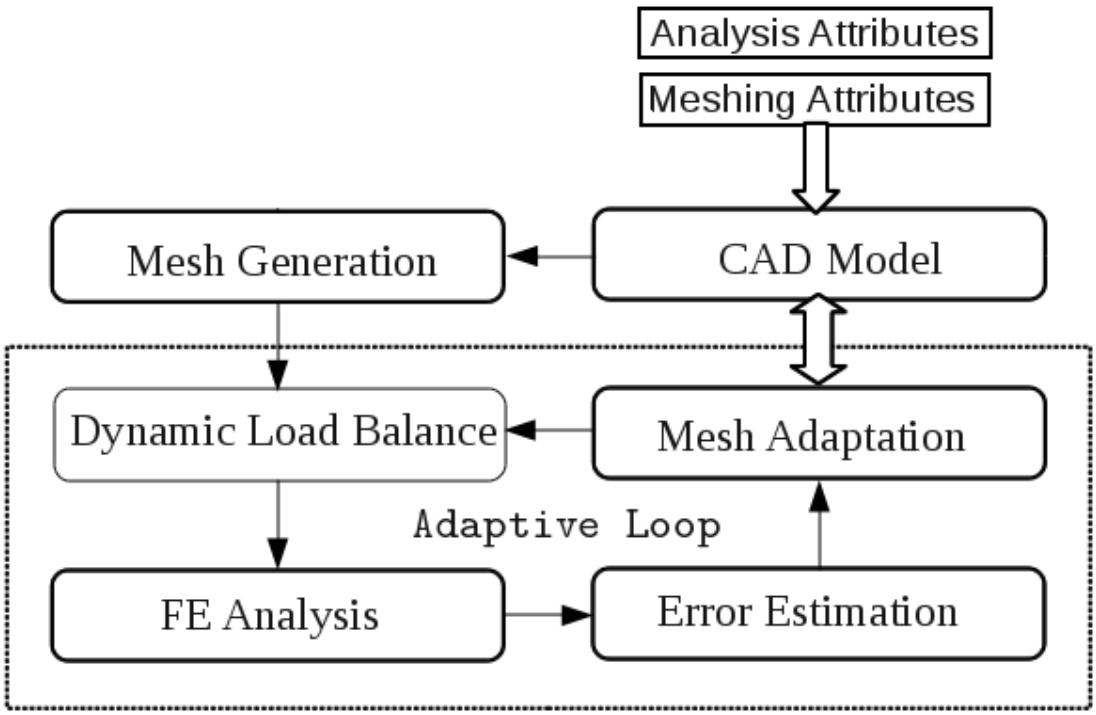


Figure 5.1: The desired workflow for an adaptive simulation

this appproach does not apply to the present work in this thesis because the basis functions used to represent the rational G^1 curved entity geometry are generally not the same as the finite element shape functions used for analysis. [1] In the present work, a general approach is adopted to construct the geometric mapping so as to account for the G^1 surface geometry based on the use of blending functions[7]. More specifically, shapes of all the lower dimensional mesh entities bounding a higher dimensional mesh entity are multiplied with linear blending functions, and the contributions are added as part of the overall mapping.

For instance, given a curved triangular face entity, its geometric mapping between the parametric domain and the physical domain is computed by blending all the vertices and edges bounding the given mesh face. The blended mapping function is given in Eq 5.1.

[1]The isogeometric analysis community aims at developing methods to use the same rational basis functions for numerical analysis, but it is outside of the scope of the thesis.

$$
\begin{aligned}
X_i(\xi_j) = \frac{1}{2}\{ & (\frac{\xi_1}{1-\xi_2})E_1(\xi_2) + (\frac{\xi_2}{1-\xi_1})E_1(1-\xi_1) \\
& +(\frac{\xi_3}{1-\xi_2})E_2(1-\xi_2) + (\frac{\xi_2}{1-\xi_3})E_2(\xi_3) \\
& (\frac{\xi_1}{1-\xi_3})E_3(1-\xi_3) + (\frac{\xi_3}{1-\xi_1})E_3(\xi_1) \\
& +\xi_1 V_1(1,0,0) + \xi_2 V_2(0,1,0) + \xi_3 V_3(0,0,1)\}
\end{aligned} \tag{5.1}
$$

where X_i denote Cartesian coordinates in the physical domain, ξ_i denote the parametric coordinates, E_i denote the mapping functions of the three edge entities respectively, and V_i denote the three vertices.

In the similar way, the geometric mapping of a tetrahedral volume entity is given in Eq 5.2.

$$
\begin{aligned}
X_i(\xi_j) = & (1-\xi_1)F_1(\xi') + (1-\xi_2)F_2(\xi') + (1-\xi_3)F_3(\xi') + (1-\xi_4)F_4(\xi') \\
& -(1-\xi_1-\xi_2)E_1(\xi') - (1-\xi_1-\xi_3)E_2(\xi') - (1-\xi_1-\xi_4)E_3(\xi') \\
& -(1-\xi_2-\xi_3)E_4(\xi') - (1-\xi_2-\xi_4)E_5(\xi') - (1-\xi_3-\xi_4)E_6(\xi') \\
& +\xi_1 V_1(1,0,0,0) + \xi_2 V_2(0,1,0,0) + +\xi_3 V_3(0,0,1,0) + \xi_4 V_4(0,0,0,1)
\end{aligned} \tag{5.2}
$$

Here, $F_j, (j = 1,2,3,4)$ denote the four face parametrization. Similarly, $E_j, (j = 1,2,3,4,5,6)$ denote edge parametrization. V_j are the vertices. ξ'_i defines the edge or face parametric coordinates normalized such that $\sum_{k=0}^{n} \xi'_k = 1$, where n is the number of mesh vertices bounding the edge or face. For example, for a face entity where $\xi_4 = 0$, the three normalized coordinates are defined as given in Eq 5.3.

$$\xi_1' = \frac{\xi_1}{\xi_1 + \xi_2 + \xi_3}$$
$$\xi_2' = \frac{\xi_2}{\xi_1 + \xi_2 + \xi_3}$$
$$\xi_3' = \frac{\xi_3}{\xi_1 + \xi_2 + \xi_3} \tag{5.3}$$

It is worth noting that the blending approach is independent of the chosen face and edge parametrization, therefore can be used with other types of parametric representations of mesh faces.

Given the blending based entity parametrization defined by Eq 5.1 and Eq 5.2, geometric mapping related calculations can be carried out in a straight-forward way. Derivatives quantities required by the Jacobian matrix $\frac{\partial x_i}{\partial \xi_j}$ can be evaluated by applying chain rule to Eq 5.2. As a result, analytic expressions of the derivatives of the blending mapping can be obtained.

For example, the partial derivative of the first term in Eq 5.2 with face F_1 is given as follows:

$$\frac{\partial}{\partial \xi_1} = -F_1(\xi_1', \xi_2') + (1 - \xi_1)(\frac{\partial F_1(\xi_1', \xi_2')}{\partial \xi_1'}\frac{\partial \xi_1'}{\partial \xi_1} + \frac{\partial F_1(\xi_1', \xi_2')}{\partial \xi_2'}\frac{\partial \xi_2'}{\partial \xi_1}) \tag{5.4}$$

Similarly, the partial derivatives of the first term that blends edge E_1 is given as follows:

$$\frac{\partial}{\partial \xi_1} = E_1(\xi_1') - (1 - \xi_1 - \xi_2)(\frac{\partial E_1(\xi_1')}{\partial \xi_1'}\frac{\partial \xi_1'}{\partial \xi_1}) \tag{5.5}$$

And the partial derivatives of the terms involving vertices are reduced to constants. Given the approach to calculate derivatives, the Jacobian matrix and its determinant can be easily evaluated. In addition to volumetric entities, Jacobian of the lower dimensional entities are also needed for carrying out integrals on the element faces/edges associated with boundary conditions.

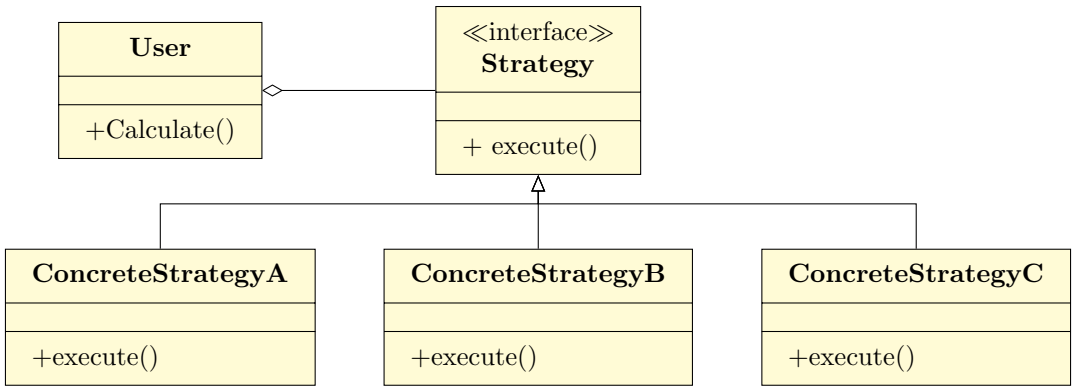


Figure 5.2: Schematic diagram of Strategy Pattern

5.3 Design of Inter-Operable Components and Interface

It is a complex task to design and implement a piece of software library which is flexible, extensible and interoperable with other finite element analysis solvers. Using proper software engineering solutions and well-known design techniques can significantly reduce the complexity and increase the quality of the program. This section introduces a few common design patterns and describes the applications of those patterns in the design and implementation of the curved element library.

5.3.1 Strategy Pattern

Strategy pattern defines a family of algorithms by encapsulating each algorithm in one separate class and making them interchangeable via a uniform interface established by their base class [96]. Figure 5.2 shows a schematic diagram of this pattern.

In this diagram the `Strategy` class declares an interface for all strategies. The client code represented by the `User` class maintains a reference to a `Strategy` object and uses the `Strategy` interface to call an algorithm. Each `ConcreteStrategy` implements the algorithm using the `Strategy` interface.

There are many places in the design of a finite element software package where this pattern can be used. For instance, linear solvers, geometries, elements, etc. In the present work, the entity class is designed by adopting the strategy pattern. Figure 5.3 shows a schematic diagram of the `Entity` class structure. Using this pattern, each curved entity type derived from the base `Entity` class encapsulates

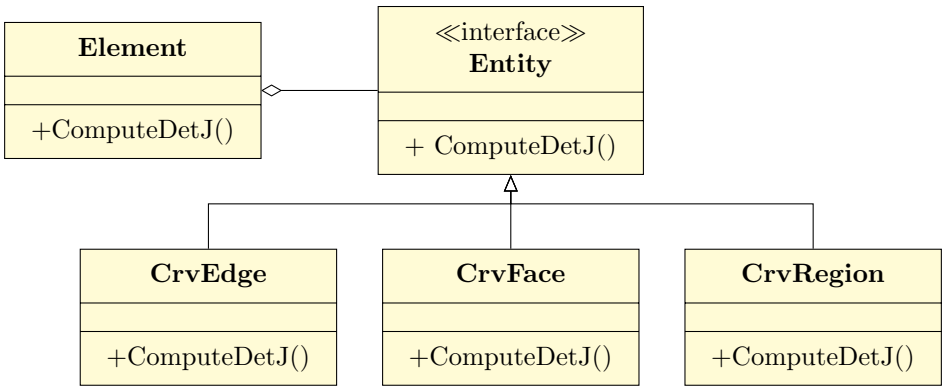


Figure 5.3: Entity class diagram using the Strategy design pattern

one algorithm to calculate the Jacobian determinant separately based on the spatial dimension of the entity. This encapsulation makes the software package easier to maintain and extend. he interface is defined by the base `Entity` class and is uniform for all derived types. The client `Element` class can be from potentially any arbitrary finite element based analysis code. T he only modification needed for the `Element` class in the user code is to maintain a pointer to the `Entity` base class, which may point to any member of the derived types of the curved entity classes. T herefore, any intrusive modification to the client code is minimized.

5.3.2 Bridge Pattern

The bridge pattern decouples the abstraction and its implementation in such a way that they can change independently [96]. Figure 5.4 shows the structure of this pattern. In this pattern Abstraction defines the interface for the user and also holds the implementor reference. AbstractionForm create a new concept and may also extend the Abstraction interface.

Implementor defines the interface for implementation part which is used by abstractions. Each ConcreteImplementor implements the implementor interface for a concrete case. Bridge pattern is useful for connecting to concepts with hierarchical structure. For instance, in a finite element program this pattern can be used to connect elements to geometries, linear solvers to their reorderer, or to connect iterative solvers to preconditioners.

In the present work, the family of curved entity classes are designed to be a

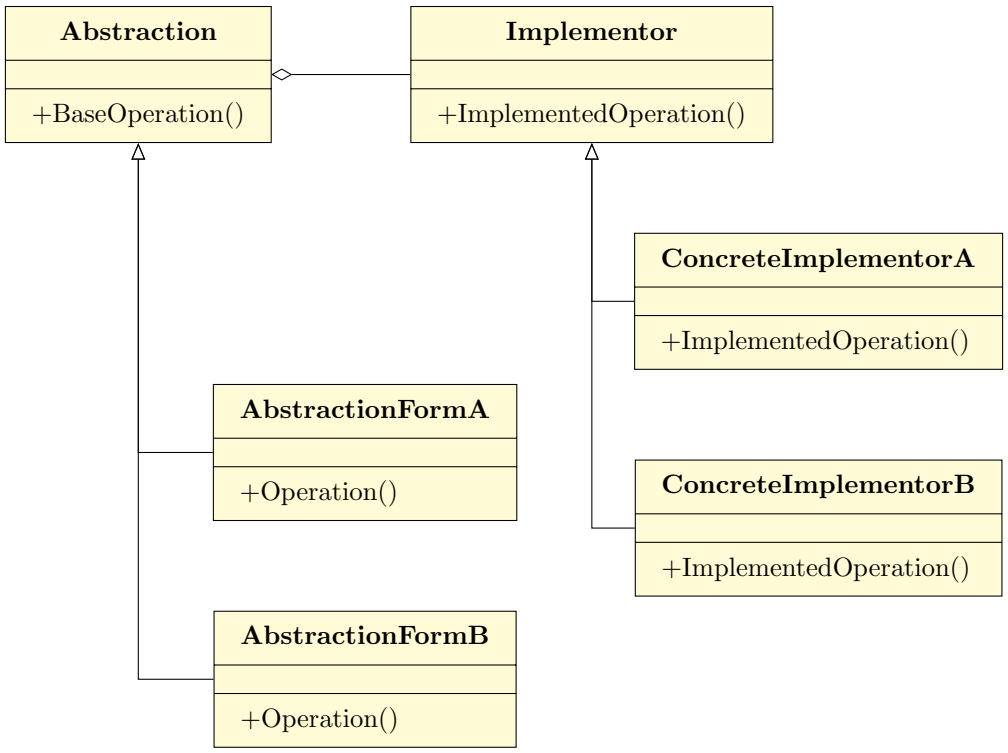


Figure 5.4: Schematic diagram of the Bridge pattern

hierarchical structure. The same is true for the geom shape classes as well. Therefore, a bridge pattern is naturally adopted to connect the curved entity family with the curved geometry family.

Figure 5.5 shows an illustration of the structure of the entity and geometric shape classes using bridge design pattern. One of the basic operations of a curved entity is to compute the Jacobian matrix and its determinant. In the present design with bridge pattern, the base (or abstract) entity class keeps a pointer to the geometry shape class, which points to any member of the actual geom shape family. The base entity class also implements the procedure to calculate Jacobian by calling the interface defined in geom shape. As a result, any derived entity type class inherits the procedure to calculate Jacobian. Therefore, any entity can be combined with any geom shape of the same spatial dimension and topology. For example, any triangular mesh entity can be combined with a triangular geometric shape such as Bezier, Gregory or other representation.

The geom shape family of classes, in the meantime, is designed and imple-

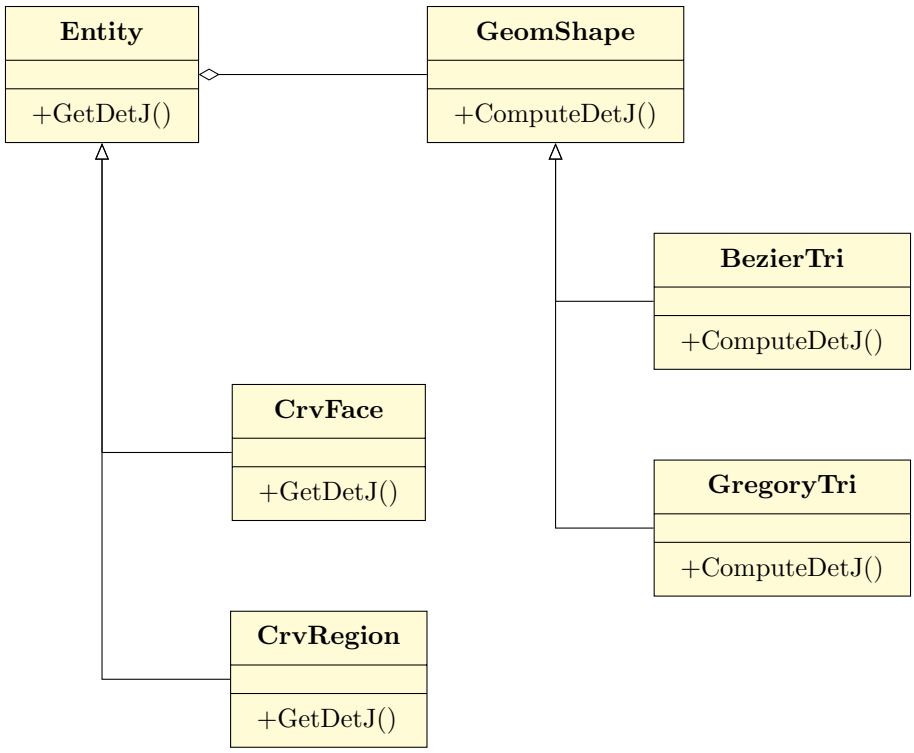


Figure 5.5: Bridge design pattern applied to mesh entity and geometric shape classes

mented with the `strategy` pattern which is similar to the design of the curved entity classes.

5.4 Solver Integration

Using curvilinear meshes and high-order finite elements in the field of computational electromagnetics has proven to produce superior analysis results than the conventional finite elements both in terms of better accuracy and faster convergence [97]. The curvilinear mesh improvement and adaptation capabilities presented in the work has been integrated with the electromagnetics solver `Omega3P` which is a part of the solver suite `ACE3P` developed by SLAC National Accelerator Laboratory (SLAC) of the Department of Energy (DOE). It is a collaboration between SCOREC and DOE to help bring high quality curvilinear meshing and adaptivity to the electromagnetics solver of interest.

The primary interest in studying the use of G^1 meshes is to examine the poten-

tial benefit of whether such meshes produce better finite element simulation results in terms of solution accuracy. It has been identified that certain types of physical applications, such as electromagnetic scattering and compressible flow applications, are extremely sensitive to the accuracy and smoothness of the computational mesh that approximates the curved domain boundaries [40, 19]. Therefore it is desirable to integrate the curved meshing techniques developed with finite element analysis solver packages that have the capability of solving some of the benchmark applications. In the present work, the finite element solver chosen for the integration and testing is are the flow solver `Nektar++` [98] developed by two collaborating research groups at the University of Utah and Imperial College London.

5.5 Nektar++

`Nektar++` is a spectral element based CFD solver package designed to allow one to construct efficient classical low polynomial order h-type elements as well as higher-order p-type elements [59].

`Nektar++` supports curvilinear meshes defined by standard Lagrange type of interpolation functions. In the solver pre-processing stage, mesh geometry data is read in by `Nektar++` in the form of spatial coordinates of vertices and high-order interpolation nodes [59]. During the analysis stage, the standard isoparametric approach is adopted to construct the geometric mapping between the parametric space and physical space. The element mapping is defined by the same set of shape functions that are used to define the finite element space and approximate the solution fields. Therefore, geometric related calculation, such as mapping evaluation, derivative evaluation, Jacobian determinant, as well as inverse Jacobian are all carried out based on the element shape functions that define the geometric mapping.

At the implementation level, the `SpatialDomains` library within the `Nektar++` package contains the most important family of classes that define the overall mesh geometry representation and provide entity level access to the geometric information needed by other analysis solver libraries within `Nektar++`. To integrate `PUMI` and the curvilinear G^1 mesh geometry representation with `Nektar++`, modifications are made to a subset of `SpatialDomains` classes to support the quartic curved

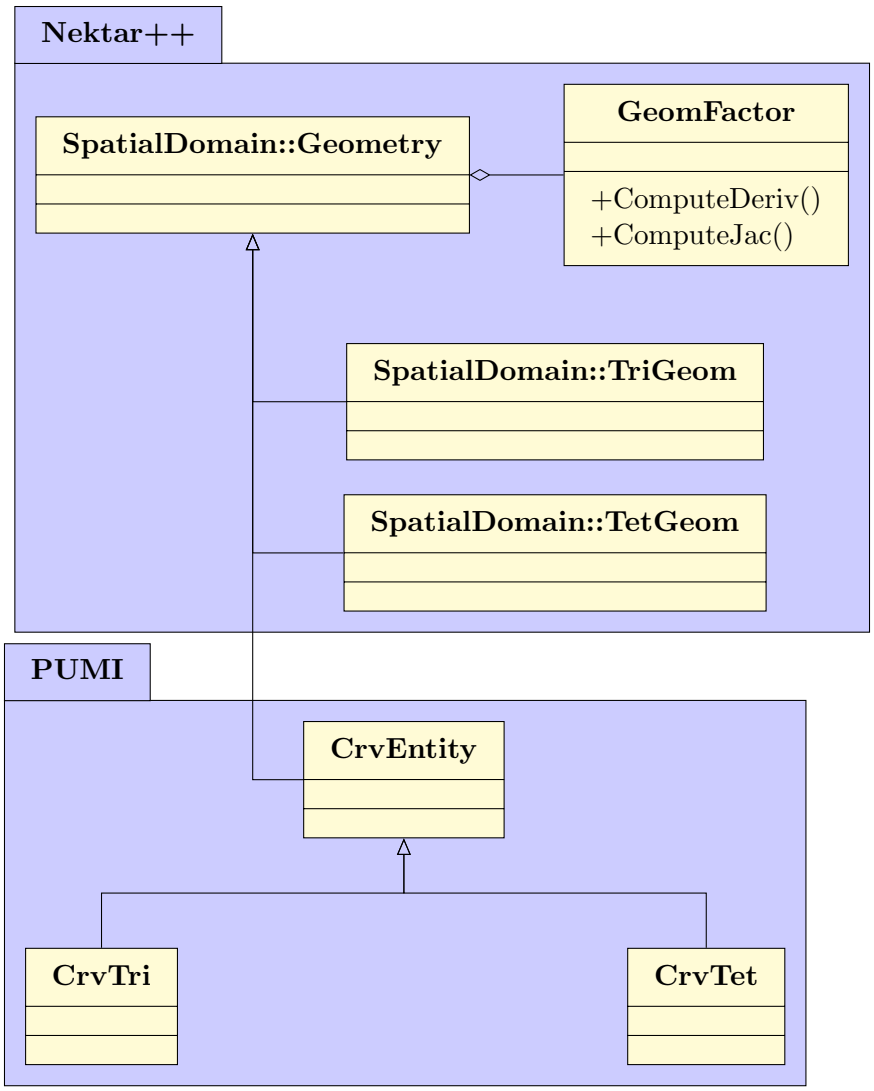


Figure 5.6: Integration of `Nektar++` solver package with `PUMI`

G^1 surface geometry and the curved volume mapping based on blending. More specifically, new classes are derived from the following existing classes of `Nektar++`: `SpatialDomains::SegGeom`; `TriGeom`; `TetGeom`, which represents the equivalent mesh edge, face and region entity in a `PUMI` mesh, to provide support for interrogations of the quartic G^1 geometric shape for mesh edge, face and region entities. The `SpatialDomains::GeomFactors` class, which originally supports Jacobian and inverse Jacobian calculation through the standard isoparametric mappings built in `Nektar++`, has been modified to perform the calculation based on the information provided by the new mesh geometry. A class diagram is given in Figure 5.6. During

runtime of the `Nektar++` solution process, an instance of `PUMI` mesh is maintained in-memory along with the `Nektar++` solver instance. Instances of the derived classes will be created for the curved mesh entities to replace the original `Nektar++` geometry class objects. Function calls originally made to compute the geometric related quantities for curved elements are re-directed to use the G^1 mesh geometry. Results obtained from test examples solved by `Nektar++` with integrated G^1 curved `MeshAdapt` are given and discussed in Chapter 6.

5.6 Omeag3P

The efforts on integration of `PUMI` and `MeshAdapt` with `Omega3P` to support linear accelerator applications have been devoted on two specific aspects: (1) extending the curved mesh adaptation capability to support specific needs of application and (2) constructing efficient in-memory adaptive loop.

5.6.1 Curved Mesh Improvement

The curved mesh improvement and adaptation procedure, in addition to being able to work with curved meshes with given CAD models, has been extended to support operations on evolving curved meshes in the absence of parametric domain definitions (such as a CAD model). The extended mesh improvement procedure includes a stage of surface mesh improvement followed by a stage of volume mesh improvement. For surface mesh improvement, an Interpolation Subdivision Surface (ISS) scheme [93] is being utilized which operates directly with nodal coordinates in the physical space, to smooth the mid-edge nodes of the deforming surface meshes. In addition to smoothing, curved local mesh modification operations (such as swap and collapse) are used to remove near-flat triangles in the surface mesh. The volume mesh improvement stage applies a set of curved mesh modification operations in a desired order to eliminate any invalid elements while keeping the improved surface mesh unchanged. High-level interface functions of the mesh improvement procedure have been implemented as part of MeshAdapt API. A version of the curved MeshAdapt has been delivered to SLAC and has been integrated with the SLAC ACDTools package. The curved mesh improvement tool is file-based. It reads in a

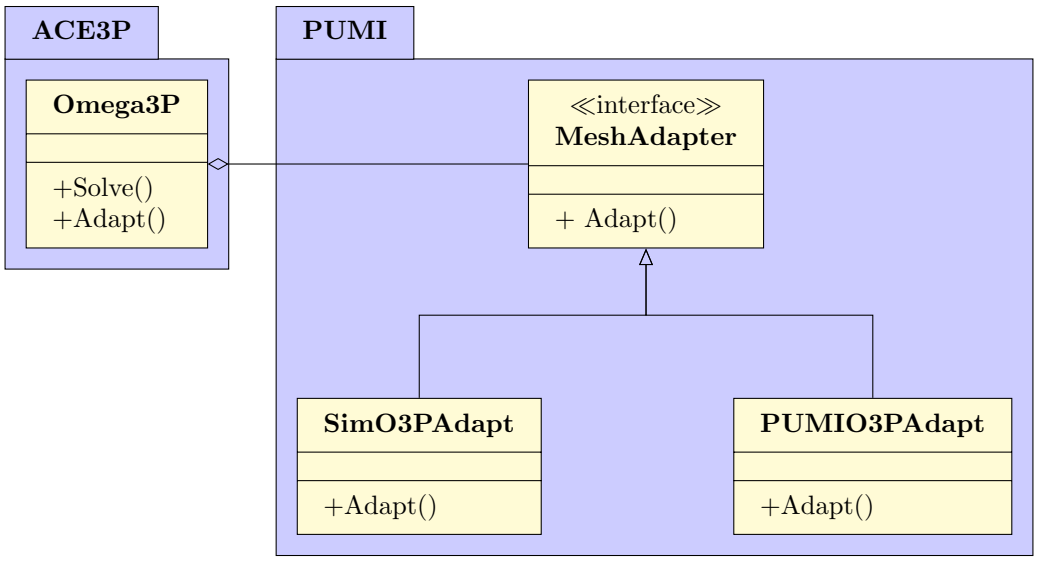


Figure 5.7: Class diagram of integration of `MeshAdapter` with `Omega3P` solver

mesh file and produces an improved mesh to a new file.

5.6.2 In-memory Adaptive Loop

A prototype in-memory adaptive loop has been designed and implemented for `Omega3P` [99]. Primary functionality of the in-memory integration includes (1) providing driver functions to run the `Omega3P` solver, SPR-based error estimation and mesh adaptation procedures and (2) providing interface-level access to the adapted mesh and solution fields for the Mesh/Field Data Transfer Functions.

`Strategy` pattern is adopted to design and implement a common interface class as well as a family of derived sub-classes implemented with specific software toolchains. A class collaboration diagram is shown in Figure 5.7.

In particular, the abstract interface class is named as `MeshAdapter`. The `MeshAdapter` class and a subset of the API functions are shown in Figure 5.8. This list of interface functions provides users with access to perform error estimation, mesh adaptation and solution transfer procedures.

One of the concrete implementation, `SimO3PAdapt` interfaces with the commercial software packages (such as GeomSim and MeshSim) from Simmetrix Inc. On the other hand, the derived class named `PUMIO3PAdapt` is implemented by using

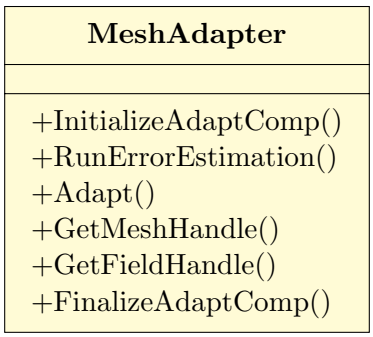


Figure 5.8: The `MeshAdapter` interface class

open-source `PUMI` and `MeshAdapt` packages from SCOREC. `PUMIO3PAdapt` manages the memory of a full representation of the `PUMI` mesh and provides access to the mesh database object for the transfer functions to transfer necessary mesh data after mesh adaptation to `Omega3P` solver. Due to the required mesh data information for `Omega3P` solver based on the classic node-connectivity data structure, specific data manipulation procedures has been implemented to perform the required manipulation of the mesh data including global vertex indexing, vertex sorting, and filling local nodal coordinate data container and element connectivity data container. The process of transferring adapted mesh data to the finite element solver is given in Algorithm 10.

```
Data: Given an adapted mesh stored in a full representation mesh
      database
Get the solver mesh handle;
Initialize the solver mesh data containers;
for each mesh vertex in the full mesh database do
    Assign a unique ID to the vertex;
    Get the coordinates of the vertex;
    Push the coordinate data to the solver mesh coordinate container;
end
for each mesh edge in the full mesh database do
    if edge has mid-point then
        Get the coordinates of the mid-point;
        Get the IDs of the end vertices of the mesh edge;
        Push the ID and coordinate data into the solver mesh edge data
          container;
    end
end
for each mesh region in the full mesh database do
    Get the IDs of the four vertices of the region;
    if region has at least one face on model boundary then
        Get the boundary condition IDs of the 4 faces of each region;
    end
    Push the ID data into the solver element container;
end
```

Algorithm 10: Transfer of adapted mesh data to finite element solver

The finite element solver class in `Omega3P` has been modified with minimal changes to be able to read in-memory mesh input data and perform the solution steps. A set of field data transfer functions have been implemented to provided in-memory access to the solution field data for `PUMIO3PAdapter` after the finite element solution process.

An adaptive loop driver is implemented in `PUMIO3PAdapter` to control the

overall workflow of the adaptive loop at the highest level through interactions with component interface API functions and mesh/field data transfer functions. Pseudo-code of the driver function is shown in Figure 5.9.

```
AdaptiveLoopDriver(Omega3PSolver & rSolver, SimO3PAdapter & rAdapter)
{
  rAdapter.ReadCADModel(inputAcisModelFile);
  rAdapter.ReadSimMeshFile(inputMeshFile);

  rAdapter.Initialize();
  rSolver.Initialize();

  for (int currentIteration = 0; currentIteration < numTotalCycles; ++
    currentIteration)
  {
    rSolver.PreProcess();
    rSolver.Solve();

    Transfer::SolveToAdapt(rSolver, rAdapter);

    rAdapter.RunErrorEstimation();
    rAdapter.DoAdapt();

    Transfer::AdaptToSolver(rAdapter, rSolver);
  }
}
```

Figure 5.9: Adaptive loop driver

CHAPTER 6
APPLICATIONS AND RESULTS

6.1 Introduction

In this chapter a set problems solved by the Omega3P and Nektar++ solvers are presented to demonstrate the capability and impact of using curved mesh creation and adaptation techniques on incompressible flow and computational electromagnetism simulations.

6.2 Incompressible Flow Applications

To test the effect of surface continuity, two problems governed by the incompressible Navier-Stokes equations are solved using Nektar++ flow solver with curved meshes of different surface continuity. The numerical results are compared with the exact analytical results to measure the impact on solution accuracy.

6.2.1 Poiseuille Flow

The first test problem is the Poiseuille Flow (also referred to as Hagen-Poiseuille Flow). It describes a fully developed laminar viscous flow through a pipe with constant circular cross-section. The governing partial differential equations for the Poiseuille flow problem are the steady-state incompressible Navier-Stokes equations written in the cylindrical coordinate system as given in Eq 6.1:

$$\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial u_z}{\partial r}) = \frac{1}{\mu}\frac{\partial p}{\partial z} \tag{6.1}$$

where μ is the viscosity of the fluid.

Closed form exact solutions exist for the pressure and z-velocity fields. The analytic expression is given in Eq 6.2. The analytical solution indicates a z-velocity profile of a parabola as shown in Figure 6.1.

$$u_z = -\frac{1}{4\mu}\frac{\partial p}{\partial z}(R^2 - r^2) \tag{6.2}$$

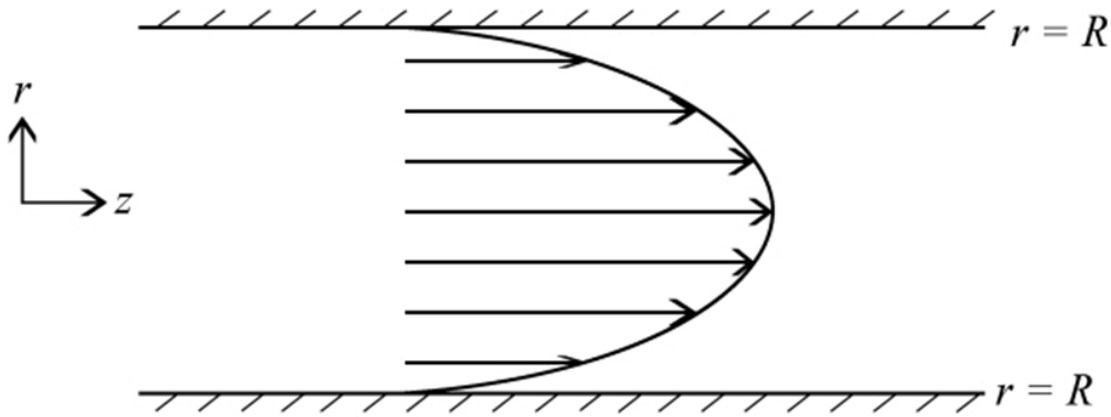


Figure 6.1: Poiseuille Flow

In the numerical test, it is of interest to solve for the fully developed velocity profile and compare the numerical solution with the exact analytical solution. A CAD model of a cylinder is constructed to represent the flow domain with radius $r = 0.5$ and length in z-direction $l = 1.0$ as shown in Figure 6.2.

The problem domain was discretized with both G^1 and C^0 meshes. No-slip condition is set for the wall. At inlet, the velocity profile is set to be fully developed as $u_z = 0.25 - r^2$. The pressure at the outlet is set to be zero. A series of simulations are performed with each type of the meshes using 4th and 5th order Legendre polynomial shape functions.

To measure the solution accuracy, L_2 and L_∞ norms of the error in velocity are used as defined in Eq 6.3.

$$\begin{aligned} L_2(e_u) &= \left(\int_{\Omega_M} (u^{h,p} - u_{exact})^2 d\Omega_M \right)^{1/2} \\ L_\infty(e_u) &= \max_{\Omega_M} |u^{h,p} - u_{exact}| \end{aligned} \tag{6.3}$$

The error of finite element solution of the z-velocity field against the exact analytic solution is measured and shown in Table 6.1. It is observed in this test case that meshes with G^1 surface continuity achieve better solution accuracy compared with C^0 meshes for the same order of shape functions. [2]

[2]Further details of the numerical tests and results have been published in Ref [100].

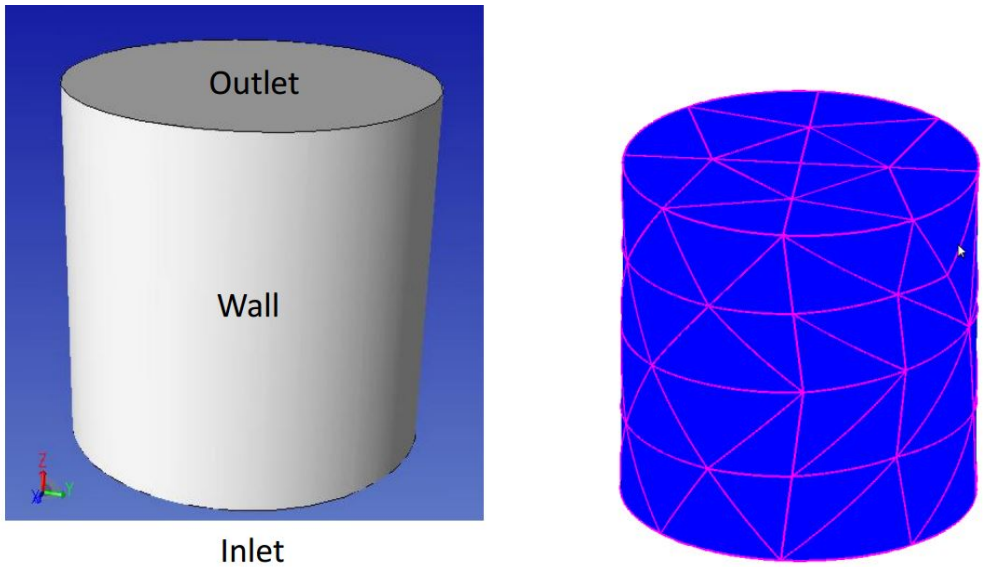


Figure 6.2: CAD model and quartic G^1 mesh for the Poiseuille flow test problem

Mesh Type	Element Order p	$L_2(e_u)$	$L_\infty(e_u)$
C^0	4	1.29e-3	4.98e-2
G^1	4	4.33e-4	4.72e-2
C^0	5	5.98e-4	2.27e-2
G^1	5	9.67e-5	7.32e-3

Table 6.1: Finite element solution error norms $L_2(u)$ and $L_\infty(u)$ in velocity for different types of curved meshes.

A similar geometric model with different dimensions was used as well [101]. The geometry is a longer cylinder with unit radius and length of 10 units in the z-direction. In addition to the error of the solution field, the first derivative error is measured in terms of L_2 norm as defined in Eq 6.4.

$$L_2(e_{\nabla u}) = \left(\int_{\Omega_M} \nabla(u^{h,p} - u_{exact}) \cdot \nabla(u^{h,p} - u_{exact}) d\Omega_M \right)^{1/2} \tag{6.4}$$

A summary of results is given in Table 6.2.

Comparing the errors in velocity, the C^0 method outperforms the G^1 method. Examining the error in pressure, for both the fifth and sixth order method, the L_2 norm is smaller for the C^0 meshes, however both the L_∞ norm of solution error and L_2 norm of the derivative error are smaller, suggesting G^1 continuity can improve

	p	$L_2(e_u)$	$L_\infty(e_u)$	$L_2(e_{\nabla u})$	$L_2(e_p)$	$L_\infty(e_p))$	$L_2(e_{\nabla p})$
C^0	5	2.14e-4	8.57e-4	6.34e-3	1.71e-3	2.40e-2	5.50e-2
G^1	5	1.22e-3	1.67e-3	1.36e-2	4.51e-3	5.25e-3	2.75e-2
C^0	6	8.12e-5	3.10e-4	2.43e-3	4.40e-4	6.85e-3	2.12e-2
G^1	6	1.21e-3	1.65e-3	1.30e-2	4.47e-3	4.61e-3	1.77e-2

Table 6.2: Finite element solution error for the Poiseuille Flow problem

the accuracy of derivatives and engineering quantities such as shear stresses [101].

6.2.2 Kovasznay Flow

The second test example studied in the present work is the Kovasznay flow problem, which describes a steady-state laminar flow behind a two-dimensional periodic array of cylinders, whose analytical solution is given by Kovasznay [102]. The governing partial differential equation is the steady-state incompressible Navier-Stokes equation as given in Eq 6.5:

$$\begin{aligned} \frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \mu \nabla^2 u + f \\ \nabla \cdot u = 0 \end{aligned} \tag{6.5}$$

The solution can be written as a function of Reynolds number Re in the form given in Eq 6.6:

$$\begin{aligned} u_x &= 1 - e^{\lambda x} \cos(2\pi y) \\ u_y &= \frac{\lambda}{2\pi} e^{\lambda x} \sin(2\pi y) \\ p &= \frac{1}{2}(1 - e^{2\lambda x}) \end{aligned} \tag{6.6}$$

where λ is a function of Reynolds number Re.

$$\lambda = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2} \tag{6.7}$$

Using the exact solution as Dirichlet boundary conditions, a steady state so-

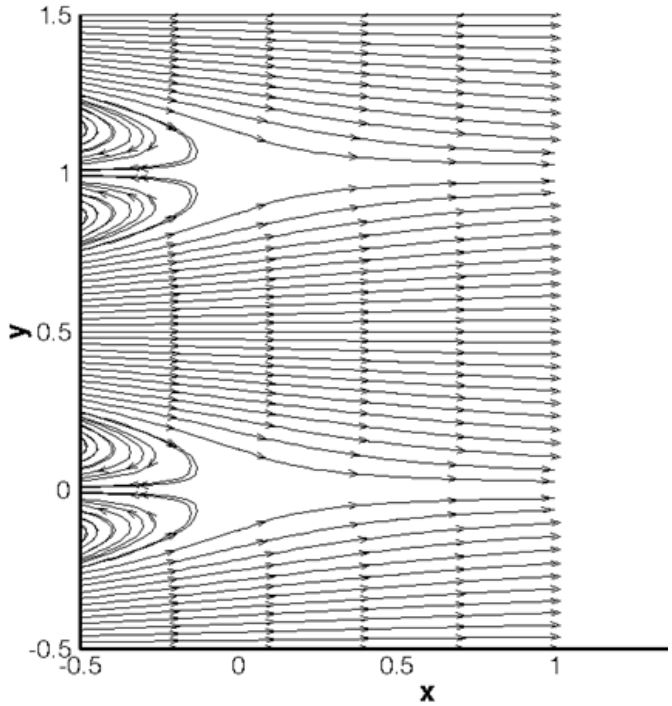


Figure 6.3: Streamline solution for 2D Kovasznay flow

lution was first solved in the 2-dimensional space.

In the classical settings, the problem was solved in a 2D square domain. In order to investigate the curved mesh impact, a curved domain boundary was introduced to the problem domain. Since the analytical solution is given, we are able to specify the exact boundary conditions on the curved geometry based on interpolating the analytical expression.

A curved mesh with 26 triangular elements is used to represent the problem domain. The mesh and solution field visualization is shown in Figure 6.4.

The study was carried out in a further step by extruding the 2D geometry in the third direction so that the domain becomes 3D. As a result, the inflow geometry is represented by a curved face which is a good candidate to test and verify the G^1 mesh curving procedure developed in the present work. Similarly, using the exact solution as Dirichlet boundary condition, a steady state solution for this 3D problem has been solved using the mesh shown in Figure 6.5.

Using the exact solution allows for the calculation of the error in the L_{inf} and

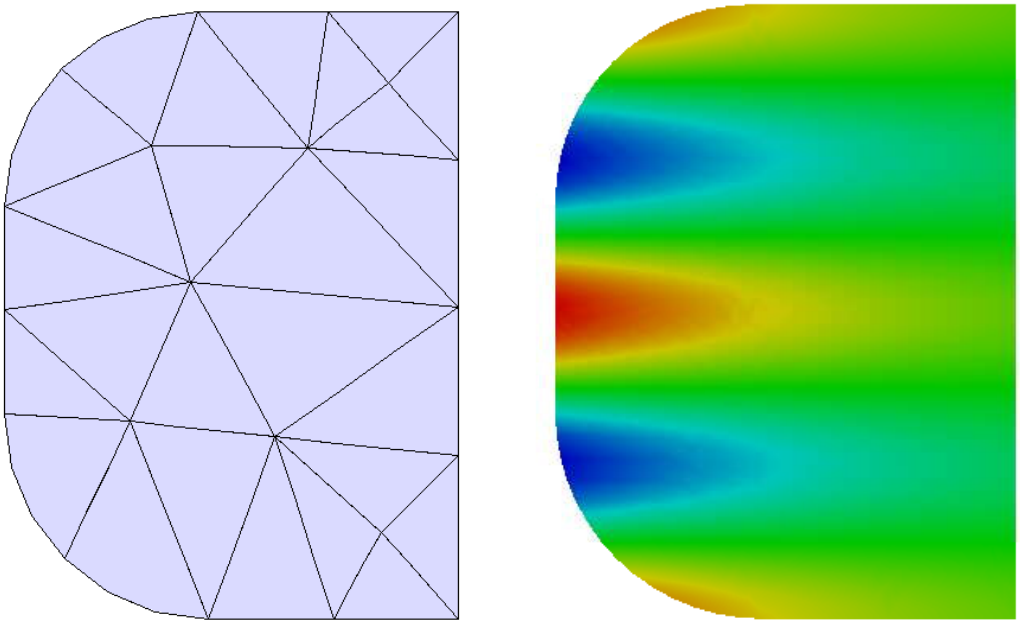


Figure 6.4: A curved mesh and the solution visualization in 2D for Kovasznay Flow

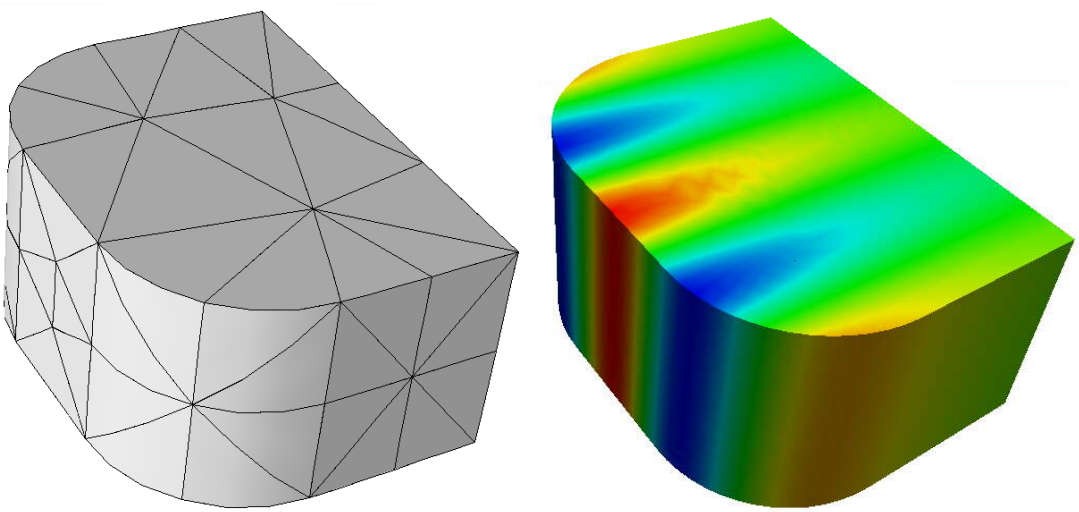


Figure 6.5: A curved mesh and the solution visualization in 3D for Kovasznay Flow

L_2 norms as defined in Eq 6.4. A summary of the simulation parameters and error statistics is given in Table 6.3.

Mesh Type	p	$L_2(e_u)$	$L_\infty(e_u)$	$L_2(e_p)$	$L_\infty(e_p))$
C^0	4	5.98e-2	1.89e-1	6.86e-2	4.95e-1
G^1	4	5.65e-2	1.88e-1	6.93e-2	5.22e-1
C^0	5	1.62e-2	1.33e-1	1.99e-2	1.35e-1
G^1	5	9.72e-3	3.00e-2	8.28e-3	5.98e-2

Table 6.3: Finite element solution errors for different types of curved meshes of the Kovasznay Flow simulations.

Comparing the errors of the velocity and pressure, the G^1 meshes outperform the C^0 meshes, which indicates potential benefits of surface continuity in the cases which the solution fields are represented by more complex analytical functions than simple polynomials as in the Poiseuille Flow case.

6.3 Computational Electromagnetism Application

As discussed in Section 5.6, the curved mesh adaptation techniques have been integrated with Omega3P to support computational electromagnetism simulations in the field linear accelerator design. Figure 4.18 in Section 4.8 gives an example that demonstrates a single-step adaptive loop on a relatively simple geometric model.

In addition, tests of the parallel adaptive loop are carried out with real world accelerator models as well. Fig 6.6 gives a 32-part mesh of the linear electron-positron collider TESLA, which is based on 9-cell superconducting niobium cavities. The eigenmode solver of Omega3P is used to compute the nominal dipole properties of the cavities. An initial mesh of 178k elements with 32 partitions is used to obtain a set of solution data. Based on the solution field, parallel error estimation and mesh adaptation processes are performed. The adapted mesh is used as input for Omega3P to re-run the computation and get more accurate solution fields. The adaptive loop iterations continue until the solution converges within tolerance. Fig 6.7 shows the solution of electric field of Omega3P analysis with a 32-part mesh of 5.14 million elements. Table 6.4 shows the series of eigenvalue solutions from the adaptive loop iterations that start from less than 200k degrees of freedom (DOFs)

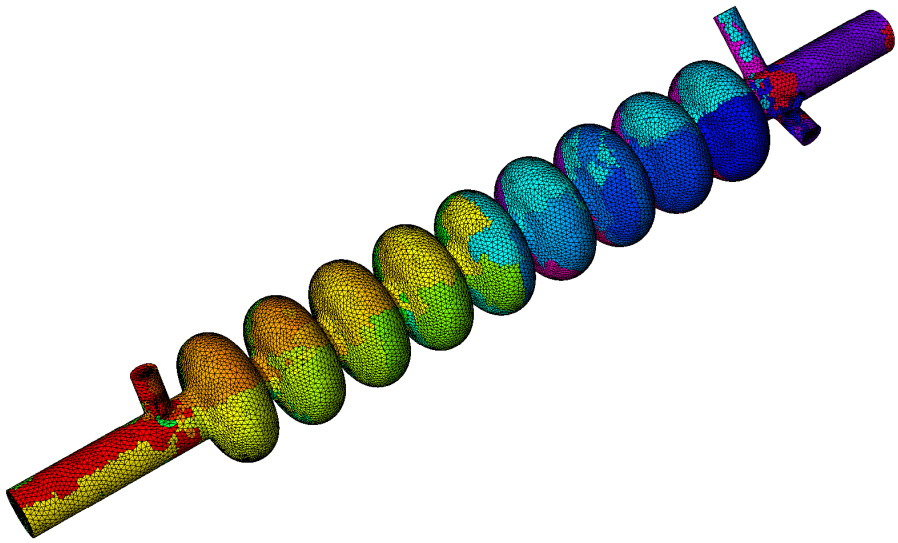


Figure 6.6: 32-part mesh of the Tesla accelerator cavity model

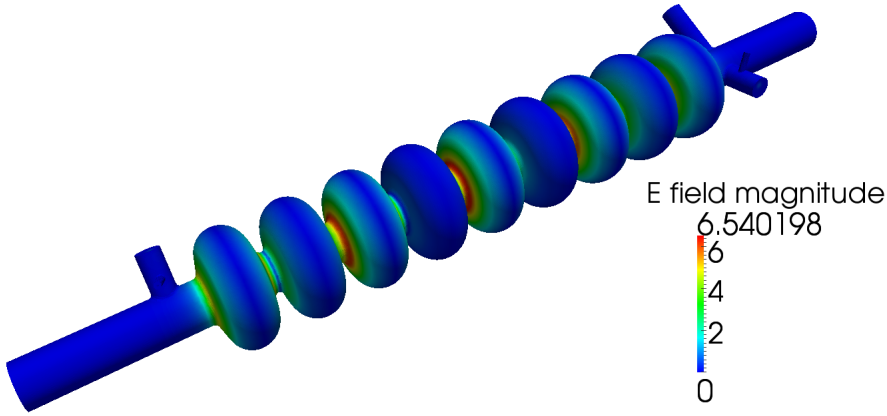


Figure 6.7: Solution field on the 32-part mesh

to more than 5.82 million DOFs. As a comparison with the error-based adaptation loop, another series of meshes adapted by uniform refinement are studied for the same problem. Solutions are obtained and listed in Table 6.5.

Fig 6.8 shows the convergence plot of the solutions obtained by error-based adaptation loops and uniform refinement meshes respectively. One can observe that the solution fields converges much faster with relatively smaller number of DOFs using the error-based mesh adaptation procedure. Note that the parallel mesh adaptation procedure is able to adapt the mesh to have number of DOFs in the order of hundreds of millions. But, it requires substantially larger amount of

No. of elements	No. of DOFs	Eigenvalue (e+09)
178k	191k	1.29377
429k	468k	1.29214
755k	834k	1.29146
1.65m	1.84m	1.29108
5.14m	5.82m	1.29088

Table 6.4: Solutions obtained by error based mesh adaptation

No. of elements	No. of DOFs	Eigenvalue (e+09)
68k	72k	1.29663
544k	558k	1.29270
4.35m	4.44m	1.29191

Table 6.5: Solutions obtained by uniformly refined meshes

resource for the Omega3P solver to run the analysis. Therefore, the results of the adaptive loop shown here are with meshes only up to several million elements.

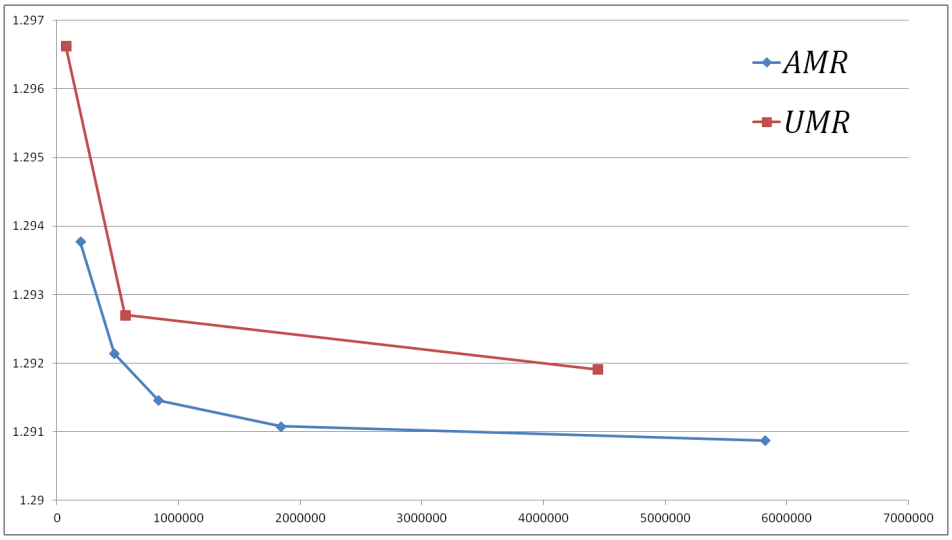


Figure 6.8: Convergence of the solution under error-based adaptation and uniform refinement

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Contributions

This dissertation presented novel procedures for high-order curved meshing and adaptation technique to effectively support high-order finite element simulations. The mesh curving procedures take advantage of the CAD technologies and make use of higher-continuity surface patches for high-order, accurate, smooth representation of the finite element computational domain, while still maintaining the flexible and local properties of the unstructured meshing (generation, adaptation) techniques. Curved mesh adaptation procedures are extended and improved to support the adaptation of high-order, higher-continuity curved meshes in parallel in order to be utilized in the context of adaptive finite element simulations. A method to create G^1-continuous surface patches is introduced and an approach to integrate a G^1 mesh with existing finite element solver is presented. Numerical test results show the advantage of using G^1 continuous meshes, compared with conventional C^0 meshes, in terms of finite element solution accuracy of specific engineering norms.

As a summary, the major contributions of the work presented in this dissertation include:

- development of curved mesh representation techniques with high-order geometric approximation accuracy and high-order geometric continuity.
- development of curved mesh adaptation techniques for parallel adaptive simulations.
- extension the mesh curving and modification operations to higher-order surface continuity.
- integration of curved meshing procedure with high-order finite element analysis solvers of interest.

- demonstration the impact of the improved mesh geometry on the simulation solution accuracy.

7.2 Future Work

Additional studies to examine the influence of G^1 continuity on more problems and for other solution norms must be carried out. There is particular interest in examining solution parameters more local to the surface. For future developments, it is of interest to study other types of high-order surface patches. Furthermore, the capability of using the CAD model surface parametrization to define exact geometric mapping is to be developed. The definition of curved mesh quality metric needs to be extended to account for mesh entities bounded by high-order Gregory patches.

It is also desirable to extend and apply the high-order surface patches to the boundary layer meshes in the application of viscous flow simulations in order to study the impact of the higher-order surface continuity.

In addition to computational fluid dynamics applications, it is also of interest to perform numerical experiments on computational electromagnetism applications with G^1 meshes as it is reported that those applications are sensitive to the accuracy and smoothness of the computational meshes [31].

In order to fully support parallel adaptive simulations, extensions of the existing mesh modification operations and curved mesh adaptation procedure is needed to account for high-order curved meshes in parallel.

REFERENCES

[1] S. Hahmann and G.-P. Bonneau, "Polynomial surfaces interpolating arbitrary triangulations." *IEEE Trans. on Visualization and Comput. Graph.*, vol. 9, no. 1, pp. 99–109, Jan-March 2003.

[2] G. E. Farin, "Triangular Bernstein-Bezier patches," *Comput. Aided Geometric Des..*, vol. 3, pp. 83–127, 1986.

[3] S. Hahmann and G.-P. Bonneau, "Triangular G1 interpolation by 4-splitting domain triangles," *Comput. Aided Geometric Des.*, vol. 17, no. 8, pp. 731 – 757, 2000.

[4] M. W. Beall and M. S. Shephard, "A general topology-based mesh data structure," *Int. J. Numerical Methods in Eng.*, vol. 40, pp. 1573–1596, 1997.

[5] S. Seol, C. Smith, D. Ibanez, and M. Shephard, "A parallel unstructured mesh infrastructure," in *High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, UT, USA, 2012, pp. 1124–1132.

[6] I. Babuska, B. A. Szabo, and I. N. Katz, "The p-version of the finite element method," *SIAM J. on Numerical Anal.*, vol. 18, no. 3, pp. 515–545, Jun. 1981.

[7] S. Dey, M. S. Shephard, and J. E. Flaherty, "Geometry representation issues associated with p-version finite element computations," *Comput. Methods in Appl. Mech. and Eng.*, vol. 150, no. 14, pp. 39 – 55, 1997.

[8] Q. Lu, M. S. Shephard, S. Tendulkar, and M. W. Beall, "Parallel mesh adaptation for high-order finite element methods with curved element geometry," *Eng. with Comput.*, vol. 30, no. 2, pp. 271–286, 2014.

[9] X. Luo, M. S. Shephard, J.-F. Remacle, R. M. O'bara, M. W. Beall, B. Szabo, and R. Actis, "P-version mesh generation issues," in *Proc. of the 11th Meshing Roundtable.*, Ithaca, NY, USA, 2002, pp. 343–354.

[10] O. C. Zienkiewicz, *The Finite Element Method in Engineering Science*, 2nd ed. London, UK: McGraw-Hill, 1971.

[11] W. J. Gordon and C. A. Hall., "Transfinite element methods: Blending-funtion interpolation over arbitrary curved element domains." *Numerische Mathematik.*, vol. 21, no. 2, pp. 109–129, 1973.

[12] U. Schramm and W. Pilkey, "The coupling of geometric descriptions and finite elements using NURBS – a study in shape optimization," *Finite Elements in Anal. and Des.*., vol. 15, pp. 11–34, 1993.

[13] S. J. Sherwin and G. E. Karniadakis, "A triangular spectral element method; applications to the incompressible navier-stokes equations," *Comput. Methods in Appl. Mech. and Eng.*, vol. 123, no. Issues 1-4, pp. 189–229, June 1995.

[14] S. Dey, J. E. Flaherty, T. K. Ohsumi, and M. S. Shephard, "Integration by table look-up for p-version finite elements on curved tetrahedra," *Comput. Methods in Appl. Mech. and Eng.*, vol. 195, pp. 4532–4543, 2006.

[15] A. Johnen, J.-F. Remacle, and C. Geuzaine, "Geometrical validity of curvilinear finite elements," in *Proc. of the 20th Int. Meshing Roundtable*, Paris, France, 2011.

[16] S. J. Sherwin and J. Peiro, "Mesh generation in curvilinear domains using high-order elements," *Int. J. Numerical Methods in Eng.*, vol. 53, pp. 207–223, 2002.

[17] P.-O. Persson and J. Peraire, "Curved mesh generation and mesh refinement using lagrangian solid mechanics," in *Proc. of the 47th AIAA Aerospace Sci. Meeting and Exhibit.*, Orlando, FL, USA, January 2009.

[18] R. Sevilla, S. Fernandez-Mendez, and A. Huerta, "3D NURBS-enhanced finite element method (NEFEM)," *Int. J. Numerical Methods in Eng.*, vol. 88, pp. 103–125, 2011.

[19] T. J. R. Hughes, J. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement." *Comput. Methods in Appl. Mech. and Eng.*., vol. 194, no. 39-41, pp. 4135–4195, 2005.

[20] X. Li, "Mesh modification procedures for general 3D non-manifold domains," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY., 2003.

[21] S. Dey, R. M. O'Bara, and M. S. Shephard., "Curvilinear mesh generation in 3D." *Comput.-Aided Des.*., vol. 33, pp. 199–209, 2001.

[22] P. L. George and H. Borouchaki, "Construction of tetrahedral meshes of degree two," *Int. J. Numerical Methods in Eng.*, vol. 90, pp. 1156–1182, 2012.

[23] H. L. de Cougny and M. S. Shephard., "Parallel refinement and coarsening of tetrahedral meshes." *Int. J. Numerical Methods in Eng.*., vol. 46, no. 7, pp. 1101–1125, 1999.

[24] M. Ainsworth and R. Rankin, "Guaranteed computable bounds on quantities of interest in finite element computations." *Int. J. Numerical Methods in Eng.*, vol. 89, no. 13, pp. 1605–1634, 2012.

[25] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, and H. D. et al., "High-order cfd methods: current status and perspective." *Int. J. for Numerical Methods in Fluids*, vol. 72, no. 8, pp. 811–845, 2013.

[26] Q. Chen and I. Babuška, "Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle," *Comput. Methods in Appl. Mech. and Eng.*, vol. 128, no. 3, pp. 405–417, 1995.

[27] L. P. Bos, "Bounding the lebesgue function for lagrange interpolation in a simplex," *J. of Approximation Theory*, vol. 38, no. 1, pp. 43–59, 1983.

[28] M. A. Taylor, B. A. Wingate, and R. E. Vincent, "An algorithm for computing fekete points in the triangle," *SIAM J. Numer. Anal.*, vol. 38, no. 5, pp. 1707–1720, 2006.

[29] S. Ahmad, B. M. Irons, and O. C. Zienkiewicz, "Analysis of thick and thin shell sstructure by curved finite elements," *Int. J. Numerical Methods in Eng.*, vol. 2, no. 3, pp. 419–451, 1970.

[30] F. Bassi and S. Rebay, "High-order accurate discontinuous finite element solution of the 2d euler equations," *J. of Computational Physics*, vol. 138, pp. 251–285, 1997.

[31] D. Xue and L. Demkowicz, "Control of geometry induced error in hp finite element (fe) simulations. i. evaluation of fe error for curvilinear geometries," *Int. J. Numer. Anal. Model*, vol. 2, no. 3, pp. 283–300, 2005.

[32] Y. Zhang, C. Bajaj, and G. Xu, "Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow," *Commun. in Numerical Methods in Eng.*, vol. 25, pp. 1–18, 2009.

[33] M. J. Borden, M. A. Scott, J. A. Evans, and T. J. R. Hughes, "Isogeometric finite element data structures based on bézier extraction of NURBS," *Int. J. Numerical Methods in Eng.*, vol. 87, pp. 15–47, 2011.

[34] E. Cohen, T. Martin, R. M. Kirby, R. F. Riesenfeld, and T. Lyche, "Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis," *Comput. Methods Appl. Mech. Engrg.*, vol. 199, pp. 334–356, 2010.

[35] G. E. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, 3rd ed. Waltham, MA, USA: Academic, 1992.

[36] 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," in *Proc. of the 17th Int. Meshing Roundtable*, Pittsburgh, PA, USA, 2008, pp. 585–602.

[37] J. Peters, "Biquartic C1-surface splines over irregular meshes," *Comput.-Aided Des.*, vol. 27, no. 12, pp. 895 – 903, 1995.

[38] B. R. Piper, "Visually smooth interpolation with triangular Bezier patches," in *Geometric Modeling: Algorithms and new Trends*, G. Farin, Ed. Philadelphia, PA, USA: SIAM, 1987, pp. 221–234.

[39] D. Walton and D. Meek, "A triangular G1 patch from boundary curves," *Comput.-Aided Des.*, vol. 28, no. 2, pp. 113 – 123, 1996.

[40] L. Demkowicz, P. Gatto, W. Qiu, and A. Joplin, "G1-interpolation and geometry reconstruction for higher order finite elements," *Comput. Methods in Appl. Mech. and Eng.*, vol. 198, no. 13, pp. 1198–1212, 2009.

[41] J. Peters, "Smooth interpolation of a mesh of curves," *Constructive Approximation*, vol. 7, no. 1, pp. 221–246, 1991.

[42] M. Watkins, "Problems in geometric continuity," *Comput.-Aided Des.*, vol. 20, no. 8, pp. 499–502, Oct. 1988.

[43] M. Boschiroli, C. Funfzig, L. Romani, and G. Albrecht, "G1 rational blend interpolatory schemes: A comparative study," *Graphical Models*, vol. 74, no. 1, pp. 29 – 49, 2012.

[44] G. Nielson, "A transfinite, visually continuous, triangular interpolant,," in *Geometric Modeling: Algorithms and new Trends*, G. E. Farin, Ed. Philadelphia, PA, USA: SIAM, 1987, pp. 235–246.

[45] ——, "Interactive surface design using triangular network splines," in *Proc. 3rd Int. Conf. on Eng. Graph. and Descriptive Geometry,*, vol. 2, Vienna, Austria, 1988, pp. 70–77.

[46] L. A. Shirman and C. H. Sequin, "Local surface interpolation with bezier patches," *Comput. Aided Geometric Des.*, vol. 4, no. 4, pp. 279–295, Dec. 1987.

[47] S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan, "A survey of parametric scattered data fitting using triangular interpolants," in *Curve and Surface Design*, H. Hagen, Ed. Philadelphia, PA, USA: SIAM, 1992, pp. 145–172.

[48] R. W. Clough and J. L. Tocher, "Finite element stiffness matrices for analysis of plates in bending," in *Proc. of conf. on matrix methods in structural anal.*, 1965, pp. 515–545.

[49] J. A. Gregory, "Smooth interpolation witwith twist constraints," in *Comput. Aided Geometric Des.*, R. E. Barnhill and R. F. Riesenfeld, Eds. New York, NY, USA: Academic, 1975.

[50] C. Loop, "A G1 triangular spline surface of arbitrary topological type," *Comput. Aided Geometric Des.*, vol. 11, pp. 303–330, 1994.

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

[52] E. S. Seol, "Fmdb : Flexible distributed mesh database." Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY., 2005.

[53] E. S. Seol and M. S. Shephard, "Efficient distributed mesh data structure for parallel automated adaptive analysis." *Eng. with Comput.*, vol. 22, no. 3, pp. 197–213, 2006.

[54] "3D ACIS Modeler," 2015. [Online]. Available: http://www.spatial.com/products/3d-acis-modeling, Accessed on: Apr. 5, 2017.

[55] "Parasolid: 3D Geometric Modeling Engine." 2015. [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/open/parasolid, Accessed on: Apr. 5, 2017.

[56] J. S. Hesthaven, "From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex." *SIAM J. on Numerical Anal.*, vol. 35, no. 2, pp. 655–676, 1998.

[57] C. Runge, "Uberempirische functionen und die interpolation zwischen aqui-distanten ordinaten." *Zeitschrift fur Mathematik und Physik*, vol. 46, pp. 224–243, 1901.

[58] "Unified Modeling Language Resource Page," 2017. [Online]. Available: http://www.uml.org, Accessed on: Apr. 5, 2017.

[59] G. Karniadakis and S. Sherwin, *Spectral/hp element methods for computational fluid dynamics.* Oxford, UK: Oxford University Press, 1999.

[60] G. Guennebaud, B. Jacob *et al.*, "Eigen Version 3.0," 2010. [Online]. Available: http://eigen.tuxfamily.org, Accessed on: Apr. 5, 2017.

[61] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance," in *IEEE Int. Conf. on Multimedia and Expo*, vol. 1, 2002, pp. 705–708.

[62] X. Li, M. S. Shephard, and M. W. Beall, "Accounting for curved domains in mesh adaptation," *Int. J. Numerical Methods in Eng.*, vol. 58, no. 2, pp. 247–276, 2003.

[63] S. Dey, "Geometry-based three dimensional hp finite element modeling and computations." Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY., 1997.

[64] X. Luo, M. S. Shephard, L.-Q. Lee, C. Ng, and L. Ge., "Curved mesh correction and adaptation tool to improve compass electromagnetic analyses," *J. of Physics*, vol. 125, no. 1, pp. 1–5, 2008.

[65] X. Luo, M. S. Shephard, L.-Q. Lee, L. Ge, and C. Ng, "Moving curved mesh adaptation for higher-order finite element simulations," *Eng. with Comput.*, vol. 27, no. 1, pp. 41–50, 2010.

[66] Q. Lu, "Developments of parallel curved meshing for high-order finite element simulations," Master's thesis, Rensselaer Polytechnic Institute., Troy, NY, December 2011.

[67] P. M. Knupp, "Introducing the target-matrix paradigm for mesh optimization via node-movement." in *Proc. of the 19th Int. Meshing Roundtable*, Chattanooga, TN, USA, 2010, pp. 67–84.

[68] X. Luo, M. S. Shephard, L.-Z. Yin, R. M. O'Bara, R. Nastasi, and M. W. Beall, "Construction of near optimal meshes for 3D curved domains with thin sections and singularities for p-version method," *Eng. with Comput.*, vol. 22, no. 1, pp. 41–50, 2010.

[69] T. W. Sederberg, "Computer Aided Geometric Design," 2011. [Online]. Available: http://tom.cs.byu.edu/ 557/text/cagd.pdf, Accessed on: March 31, 2017.

[70] J. Dompierre, P. Labbé, F. Guibault, and R. Camarero, "Benchmarks for 3D unstructured tetrahedral mesh optimization." in *Proc. of the 7th Int. Meshing Roundtable*, Dearborn, MI, USA, 1998.

[71] S. Gosselin and C. Ollivier-Gooch, "Tetrahedral mesh generation using delaunay refinement with non-standard quality measures." *Int. J. Numerical Methods in Eng..*, vol. 87, pp. 795–820, 2011.

[72] A. Liu and B. Joe., "On the shape of tetrahedra from bisection." *Mathematics of Computation.*, vol. 63, pp. 141–154, 1994.

[73] ——, "Relationship between tetrahedron shape measures." *BIT Numerical Mathematics*, vol. 34, pp. 268–287, 1994.

[74] P. M. Knupp, "Algebraic mesh quality metrics." *SIAM J. on Scientific Computing.*, vol. 23, no. 1, pp. 193–218, 2010.

[75] M. S. Shephard, S. Dey, and M. K. George, "Automatic meshing of curved three-dimensional domains: Curving finite elements and curvature-based mesh control." in *Modeling, Mesh Generation, and Adaptive Numerical Method for Partial Differential Equations.*, I. Babuska, J. Flaherty, W. Henshaw, J. Hopcroft, J. Oliger, and T. Tezduyar, Eds. Berlin, Germany: Springer, 1994, pp. 67–96.

[76] X. Luo, "An automatic adaptive directional variable p-version method in 3D curved domains," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY., Troy, NY., 2005.

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

[78] E. Ruiz-Girons, J. Sarrate, and X. Roca, "Generation of curved high-order meshes with optimal quality and geometric accuracy," *Procedia Eng.*, vol. 163, pp. 315 – 327, 2016.

[79] A. Johnen, C. Geuzaine, T. Toulorge, and J.-F. Remacle, "Efficient computation of the minimum of shape quality measures on curvilinear finite elements," in *Proc. of the 25th Int. Meshing Roundtable*, Washington, DC, USA, 2016.

[80] M. Turner, J. Peiro, and D. Moxey, "A variational framework for high-order mesh generation," in *Proc. of the 25th Int. Meshing Roundtable*, Washington, DC, USA, 2016.

[81] H. Prautzsch and L. Kobbelt, "Convergence of subdivision and degree elevation." *Advances in Computational Mathematics*, vol. 2, pp. 143–154, 1994.

[82] G. Morin and R. Goldman, "On the smooth convergence of subdivision and degree elevation for bezier curves." *Comput. Aided Geometric Des.*., vol. 18, pp. 657–666, 2001.

[83] H. L. de Cougny, M. S. Shephard, and M. K. Georges., "Explicit node point mesh smoothing within the octree mesh generator." Scientfic Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY, USA, Tech. Rep., 1990.

[84] L. A. Freitag, M. T. Jones, and P. E. Plassmann, "An efficient parallel algorithm for mesh smoothing." in *Proc. of the 4th Int. Meshing Roundtable*, Albuquerque, NM, USA, 1995, pp. 47–58.

[85] L. A. Freitag and P. M. Knupp., "Tetrahedral element shape optimization via the jacobian determinant and condition number." in *Proceeding of the 8th Int. Meshing Roundtable*, South Lake Tahoe, CA, USA, 1999, pp. 247–258.

[86] E. Ruiz-Girones, J. Sarrate, and X. Roca, "Generation of curved high-order meshes with optimal quality and geometric accuracy," in *Proc. of the 25th Int. Meshing Roundtable*, Washington, DC, USA, 2016.

[87] S. Coons, "Surfaces for computer aided design." Massachusetts Institute of Technology, Cambridge, MA, USA, Tech. Rep., 1964.

[88] ——, "Surface patches and b-spline curves," in *Comput. Aided Geometric Des.*, R. E. Barnhill and R. F. Riesenfeld, Eds. New York, NY, USA: Academic, 1974.

[89] X. Li, M. S. Shephard, and M. W. Beall, "3D anisotropic mesh adaptation by mesh modification." *Comput. Methods in Appl. Mech. and Eng.*, vol. 194, pp. 4915–4950, 2005.

[90] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. part 1. the recovery technique." *Int. J. Numerical Methods in Eng..*, vol. 33, pp. 1331–1361, 1992.

[91] ——, "The superconvergent patch recovery and a posteriori error estimates. part 2. error estimates and adaptivity." *Int. J. Numerical Methods in Eng..*, vol. 33, pp. 1365–1382, 1992.

[92] M. Mubarak, S. Seol, Q. Lu, and M. S. Shephard, "A parallel ghosting algorithm for the flexible distributed mesh database," *Submitted to Scientific Programming*, 2012.

[93] J. Wan, "An automatic adaptive procedure for 3D metal forming simulations." Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY., 2006.

[94] M. W. Beall and M. S. Shephard, "An object-oriented framework for reliable numerical simulations." *Eng. with Comput.*, vol. 15, no. 1, pp. 61–72, 1999.

[95] F. Alauzet, X. Li, E. S. Seol, and M. S. Shephard, "Parallel anisotropic 3D mesh adaptation by mesh modification," *Eng. with Comput.*, vol. 21, pp. 247–258, 2006.

[96] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[97] 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., Tech. Rep., February 2009.

[98] C. D. Cantwell, S. Yakovlev, R. M. Kirby, N. S. Peters, and S. J. Sherwin, "High-order spectral/hp element discretisation for reaction-diffusion problems on surfaces: Application to cardiac electrophysiology," *J. of Computational Physics*, vol. 257, pp. 813 – 829, 2014.

[99] C. Smith, B. Orecchio, O. Sahni, and M. Shephard, "Building effective parallel unstructured adaptive simulations by in-memory integration of existing software components," Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY, USA, Tech. Rep. 2, 2012.

[100] Q. Lu and M. S. Shephard, "Development of unstructured curved meshes with G1 surface continuity for high-order finite element simulations," in *Int. Conf. on Spectral and High Order Methods*, Salt Lake City, Utah, June 2014.

[101] D. W. Zaide, Q. Lu, and M. S. Shephard, "A comparison of C0 and G1 continuous curved tetrahedral meshes for high-order finite element simulations," in *Proc. of the 24th Int. Meshing Roundtable*, Austin, TX, USA, 2015.

[102] L. I. G. Kovasznay, "Laminar flow behind a two-dimensional grid," *Mathematical Proc. of the Cambridge Philosophical Society*, vol. 44, pp. 58–62, 1 1948.