Spring 1-1-2018

Geometrically Exact and Analysis Suitable Mesh Generation Using Rational Bernstein–Bezier Elements

Luke H. Engvall
University of Colorado at Boulder, luke.engvall@gmail.com

Geometrically Exact and Analysis Suitable Mesh Generation using Rational Bernstein–Bézier Elements

by

Luke Houlden Engvall

B.S., University of New Mexico, 2012

M.S., University of Colorado, 2015

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Mechanical Engineering

2018

This thesis entitled:
Geometrically Exact and Analysis Suitable Mesh Generation using Rational Bernstein–Bézier Elements
written by Luke Houlden Engvall
has been approved for the Department of Mechanical Engineering

Assistant Prof. John A. Evans

Assistant Prof. Peter Hamlington

Prof. Kenneth Jansen

Prof. Daven Henze

Prof. Franck Vernerey

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Engvall, Luke Houlden (Ph.D., Mechanical Engineering)

Geometrically Exact and Analysis Suitable Mesh Generation using Rational Bernstein–Bézier Elements

Thesis directed by Assistant Prof. John A. Evans

This dissertation presents two novel contributions to the fields of isogeometric analysis and p-version finite elements. First, we present a framework for geometrically exact volumetric mesh generation. By leveraging ideas from both traditional mesh generation as well as isogeometric analysis, we develop a framework for volumetric mesh generation using rational Bernstein–Bézier discretizations. Within this framework, we provide a set of easily verifiable sufficient conditions for guaranteeing that a mesh will be geometrically exact. Second, we develop a complete theory of mesh quality for these rational Bernstein–Bézier elements. From this, we derive a set of easily computable mesh quality metrics for verifying that a rational Bernstein–Bézier discretization will be analysis suitable.

Dedication

To my parents. You taught me to be curious, ask questions, and never stop learning.

Acknowledgements

This work would not have been possible without my advisor, Professor John A. Evans. His mentorship has been invaluable throughout my graduate studies. I am also grateful to my fellow graduate students and the members of the CMGLab for their camaraderie and collaboration over the past five years. I would like to thank Joe Benzaken in particular for his insights regarding mesh quality metrics. Finally, I owe an immense debt of gratitude to Jordan Marks. Her support and encouragement has been indispensable while writing this dissertation, and her edits to this dissertation greatly improved the finished work.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1144083.

Contents

Tables

Figures

Chapter 1

Introduction

Computer aided engineering (CAE) has become an indispensable and ubiquitous tool for practicing engineers. The concept of CAE is incredibly broad, and includes any computational tools used in the engineering workflow, such as computer aided design (CAD), finite element analysis (FEA), and computer aided manufacturing (CAM). The widespread adoption of CAE has been largely spurred by the increasingly powerful computational resources available to scientists and engineers. More powerful computers have not only allowed for increasingly complex designs in the CAD environment, but have also enabled larger and more complex simulations using FEA. However, as engineers rely increasingly on the CAE workflow, there is a greater need to ensure that the numerics that underpin these software packages are robust, accurate, and efficient.

To motivate the importance of these numerical methods, it is illuminating to first consider the current state of the CAE workflow, with a particular focus on CAD and FEA, the topics most relevant to this work. When a product enters the CAE workflow, a design is first modeled in computer aided design (CAD) software. In CAD software, products are typically modeled using boundary representations (B-Reps), which represent solid objects as a collection of their bounding surfaces. These surfaces are typically parametrized by trimmed non-uniform rational B-spline (NURBS) surfaces, which give a mathematical description of the outer surface of the product. We provide a more detailed description of both NURBS surface and B-Reps in Chapter 2.

Once a design has been established, engineering analysis is typically performed by solving the partial differential equations that govern the system by some numerical method such as the finite difference method, finite volume method or finite element method. Of these methods, one of the most widely used is the finite element method (FEM), also referred to as FEA. In the finite element method, the geometry is first discretized into a collection of finite elements, or mesh. Typically, this mesh is constructed by an automated mesh generation program, with varying levels of user oversight to ensure that the resulting mesh will be adequate for FEA. Once a mesh is obtained, analysts run a simulation over the mesh, which informs design change. The CAD model is then updated, and the process begins anew.

Ideally, engineers should be able to easily leverage both CAD and FEA in tandem to converge on optimal designs. However, in the traditional CAD to FEA paradigm, this is not the case, due to the difficulty associated with created finite element meshes from CAD geometries. In an internal study at Sandia National Laboratories, Hardwick and Clay identified 10 distinct steps in their design and analysis cycle [29]. They found that approximately:

- 4% of the total design cycle time was spent on creating the CAD model.
- 72% of the time was spent converting this CAD model to an analysis suitable mesh. This includes the geometry de-featuring, mesh generation, and mesh manipulation, among other steps.
- 24% of the time was spent on running the analysis. This includes simulation setup, run time, and post processing.

This disconnect between CAD and FEA leads to what has come to be known as the **design-to-analysis bottleneck**.

From this case study, it is immediately obvious that the design-to-analysis bottleneck is a major hinderance to the engineering design process. Moreover, we note that the meshes generated for analysis are most often comprised of linear (straight-sided) elements. These linear meshes are completely naíve to the geometry of the original CAD model, and the analysis model is, at best, a linear approximation of the true geometry of interest. The increased reliance on automated

CAE processes motivates the need for numerical methods that mitigate the effects of the design-to-analysis bottleneck and allow for tighter coupling between CAD and FEA. Two fields of research that have seen increased attention in recent years are p-version finite elements and isogeometric analysis (IGA).

In p-version FEA, the mesh is generally composed of curvilinear simplicial or tensor product elements, which give a piecewise polynomial parameterization of the domain. The solution is then typically approximated by piecewise polynomial interpolation over each element in the mesh. By discretizing the domain using curvilinear elements, the geometry can be approximated more accurately while simultaneously using fewer elements than a corresponding linear discretization, allowing for increased accuracy for a given computational cost. However, generating good quality, geometrically accurate curvilinear meshes over complex domains in an automated and robust manner is non-trivial, and is still an area of active research [28, 34, 39, 45, 53, 56, 63, 64, 71, 76].

Isogeometric analysis was introduced in 2005 as an attractive alternative to p-version finite elements [31]. In IGA, rather than discretizing the domain using a mesh, the NURBS objects from CAD are used directly as the parametrization for analysis, and the solution is approximated by the NURBS basis functions. This approach affords several advantages over traditional p-version finite elements, including higher-order continuity of the basis and circumvention of the meshing process, which results in a tight coupling between the CAD and FEA models. While IGA promises to alleviate the need for traditional meshes, in many cases some amount of preprocessing of the CAD model is still needed before the analysis step. In CAD software, B-Reps are composed of **trimmed** NURBS surfaces. That is, wherever two bounding surfaces intersect, the surfaces are trimmed, and the resulting surfaces are joined at the trimming curve. Unfortunately these trimming curves result in **implicit** parameterizations of the surface, whereas an **explicit** parameterization is required for analysis with IGA[1] . However, even if we can obtain an explicit surface parameterization, we still often require an explicit volumetric parameterization for engineering analysis. To make IGA

[1] An explicit parameterization is required for classical isoparametric finite element methods. While there have been a number of meshless and immersed methods proposed for both FEA and IGA [32, 35, 49, 51, 59], the classic isoparametric paradigm is still the most widely used.

a truly automated design-through-analysis workflow, robust and efficient methods are required to transform a B-Rep from CAD into an explicit parameterization that can be used for analysis. This problem has been termed **surface-to-volume parameterization** and, as with mesh generation for p-version FEA, is an area of active research [1, 3, 24, 25, 42, 43, 46, 72, 77, 80].

It is clear that there exists a need for robust, efficient, and automatic geometric preprocessing algorithms for both traditional p-version finite elements and IGA. Furthermore, it is not enough to simply create any explicit parameterization, but rather the resulting parameterization must also be **analysis suitable**[2] . When practitioners of FEA or IGA use automated geometric preprocessing software, they must be able to easily verify if the resulting discretization can be expected to yield accurate results when used for analysis. As such, a set of easily computable **element quality metrics** is integral to any geometric preprocessing algorithm. Indeed, as discussed in Section 5.2.4, a suite of quality metrics for curvilinear discretizations already exist in both the FEA and IGA communities [24, 26, 37, 54, 68, 78, 82]. However, to the best of our knowledge, the existing metrics **are not sufficient to guarantee the analysis suitability of curvilinear meshes!** This is troubling, as the metrics currently in use can admit non analysis–suitable discretizations without any indication to the end user.

This dissertations presents two novel contributions that address these open problems in the fields of IGA and p-version finite elements.

(1) **Geometrically Exact Volumetric Mesh Generation**

By leveraging ideas from both traditional mesh generation as well as IGA, we develop a framework for geometrically exact mesh generation using rational Bernstein–Bézier discretizations. Within this framework, we provide a verifiable set of conditions for guaranteeing that a mesh will be geometrically exact.

[2] We say a parameterization is analysis suitable if it is both **invertible** and **well-conditioned**. A parameterization is invertible if the element-wise Jacobian determinant is bounded from both above and below. A parameterization is well-conditioned if the parameterization does not adversely affect the approximation accuracy of the finite element method.

(2) **Sufficient and Computable Mesh Quality Metrics**

We develope a complete theory of mesh quality for these rational Bernstein–Bézier elements. From this, we derive a set of easily computable mesh quality metrics for verifying the analysis suitability of these rational Bernstein–Bézier discretizations.

With these two contributions in mind, the remainder of this dissertation is organized as follows. Thus far, we have given only a cursory overview of CAD objects, mesh generation, and IGA. We begin by providing an in-depth literature review in Chapter 2. Then, as rational Bernstein–Bézier elements are central to this work, we dedicate the entirety of Chapter 3 to reviewing these elements. With the necessary preliminaries established, we present original work of this dissertation. In Chapter 4, we present an algorithm for geometrically exact mesh generation using mixed element Bernstein–Bézier discretizations, and in Chapter 5, we present element quality metrics for these Bernstein–Bézier discretizations. Finally, we provide some concluding remarks in Section 6 regarding the work done in this dissertation, its potential impact on the field, as well as unanswered questions and directions for future research.

Chapter 2

Review of the Literature

To provide necessary background for later chapters, we first review the state of the art of the three fields of research directly related to this dissertaion. First, Section 2.1 reviews the mathematical descriptions used to represent three dimensional objects in CAD software. Then, Section 2.2 reviews the field of mesh generation, which provides much of the inspiration for this disseration. Finally, Section 2.3 reviews isogeometric analysis, with a particular focus on the problem of surface-to-volume parameterization. This last section reviews previously attempted solutions to surface-to-volume parameterization, as well as motivates the need for further work in the area.

2.1 Computer Aided Design

We begin this section by discussing the construction of NURBS curves and surfaces. Because these objects share largely the same construction, we will collectively refer to them as **NURBS objects**, and will make the explicit differentiation between curves and surfaces where necessary. We then present an introduction to B-Reps, which are one of the de-facto standards for representing complex geometries in modern CAD packages[1] . Finally, we discuss some of the limitations of these representations, and discuss T-splines as an attractive alternative.

[1] We note that constructive solid geometry (CSG) representations are also commonly encountered in CAD packages, but as we are specifically interested in the problem of surface-to-volume parameterization, we limit our discussion to B-Reps.

2.1.1 NURBS Objects

NURBS (Non-Uniform Rational B-spline) objects lie at the heart of modern CAD packages due to their flexibility in representing a wide variety of geometries. The simplest NURBS object is a B-spline curve. To define a B-spline curve of polynomial degree p, we require:

(1) A set of basis functions defined in d_r-dimensional **parameter space**[2]. This is given by a set of n univariate ($d_r = 1$) B-spline basis functions of degree p, $\{N_i^p\}_{i=1}^n$.

(2) A control net in d_s-dimensional **physical space**[2]. This is given by a set of n control points, $\{\mathbf{P}_i\}_{i=1}^n$, that lie in $\mathbb{R}^{d_s}$.

To define the B-spline basis in parameter space, we first define a **knot vector**, a set of non-decreasing coordinates in parameter space, $\boldsymbol{\Xi} = \{\Xi_1 \ \Xi_2 \ ... \ \Xi_{n+p+1}\}$. The B-spline basis functions of degree p are then defined via the Cox-deBoor recursion formula as:

$$N_i^0(\xi) := \begin{cases} 1 \text{ if } \Xi_i \leq \xi < \Xi_{i+1} \\ 0 \text{ otherwise} \end{cases}$$

$$N_i^p(\xi) := \frac{\xi - \Xi_i}{\Xi_{i+p} - \Xi_i} N_i^{p-1}(\xi) + \frac{\Xi_{i+p+1} - \xi}{\Xi_{i+p+1} - \Xi_{i+1}} N_{i+1}^{p-1}(\xi) \qquad \forall \ \xi \in [\Xi_1, \Xi_{n+p+1}]$$

The desired B-spline curve is then defined as the sum of the B-spline control points multiplied with their respective basis functions, viz.:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_i^p(\xi) \mathbf{P}_i$$

As their name implies, NURBS are a generalization of B-splines, with the key difference being that each NURBS control point has an associated weight, and the NURBS basis is a rational basis.

[2] Here, the subscript r in d_r stands for **reference**, and the subscript s in d_s stands for **spatial**.

To define a NURBS curve, we require three things:

(1) A set of n B-spline basis functions $\{N_i\}_{i=1}^{n}$, in $\mathbb{R}^{d_r}$.

(2) A set of n control points, $\{\mathbf{P}_i\}_{i=1}^{n}$, in $\mathbb{R}^{d_s}$.

(3) A set of n control weights, $\{w_i\}_{i=1}^{n}$.

Physically, a NURBS curve is the **projective mapping** of a B-spline curve in $\mathbb{R}^{d_s+1}$ to $\mathbb{R}^{d_s}$. The control points of the B-spline curve in $\mathbb{R}^{d_s+1}$ are the **projective** or **homogenous** NURBS control points, $\widetilde{\mathbf{P}}_i$. The homogenous NURBS control points are found as:

$$\begin{aligned} (\widetilde{\mathbf{P}}_i)_j = (\mathbf{P}_i)_j w_i \qquad i \in \{1, ..., n\},\ j \in \{1, ..., d_s\} \\ (\widetilde{\mathbf{P}}_i)_{d_s+1} = w_i \end{aligned} \tag{2.1}$$

Numerically, a NURBS curve is defined as a linear combination of the product of the NURBS control points with their respective basis functions, viz.:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} R_i^p(\xi)\mathbf{P}_i = \sum_{i=1}^{n} \frac{N_i^p(\xi) w_i}{\sum\limits_{j=1}^{n} N_j^p(\xi) w_j}\mathbf{P}_i$$

where the NURBS basis, $\{R_i\}_{i=1}^{n}$, is defined from the B-spline basis and control point weights via the relation:

$$R_i^p(\xi) = \frac{N_i^p(\xi) w_i}{\sum\limits_{j=1}^{n} N_j^p(\xi) w_j}$$

Note that when all the control point weights are identically unity, the NURBS basis functions reduce to the B-spline basis functions, and the corresponding NURBS curve reduces to a B-spline curve. In this sense, the B-spline basis functions and B-spline curves are a restricted subset of the NURBS basis and NURBS curves, respectively. From an analysis point of view, the NURBS basis exhibits several beneficial properties, including pointwise non-negativity, partition of unity, linear independence, and compact support. From a geometric modeling point of view, NURBS curves exhibit the beneficial properties of affine covariance and a strong convex hull.

Fig. 2.1 and Fig. 2.2 provide graphical representations of B-spline and NURBS curves, respectively. Fig. 2.1 shows an example quadratic B-spline basis for the knot vector $\mathbf{\Xi} = [0\ 0\ 0\ 1\ 1\ 2\ 3\ 4\ 4\ 4]$

and a corresponding B-spline curve. Fig. 2.2 shows the NURBS basis functions for the same knot vector $\boldsymbol{\Xi} = [0\ 0\ 0\ 1\ 1\ 2\ 3\ 4\ 4\ 4]$. Note that the control net for the NURBS curve in Fig. 2.2b is the same as for the B-spline in Fig. 2.1a. However, a weight of 3 is applied to the third control point, leading to a change in both the basis and shape of the curve. Extending our discussion to

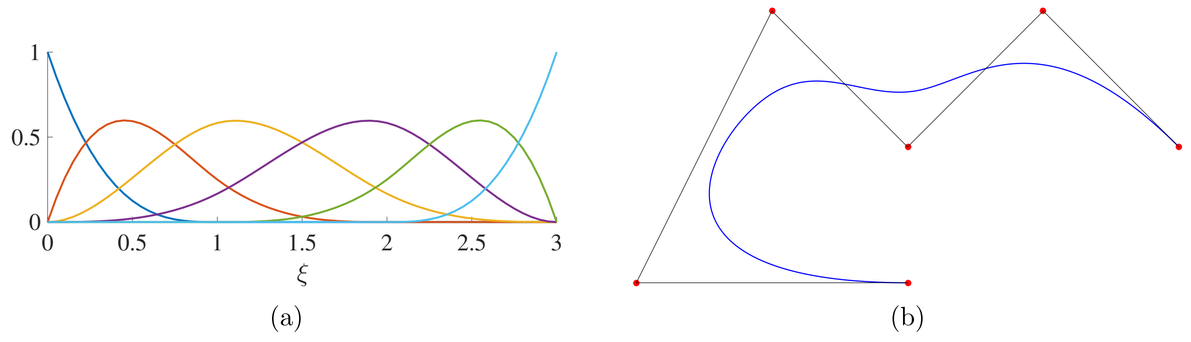


Figure 2.1: (a) The B-spline basis functions and (b) the corresponding B-spline curve.

NURBS surfaces is relatively straight forward, as the bivariate NURBS basis is simply the tensor product of two sets of univariate NURBS basis functions of degrees p_1 and p_2, respectively. That is, we define the bivariate ($d_r = 2$) NURBS basis functions as:

$$R_{\{i_1,i_2\}}(\boldsymbol{\xi}) = \frac{N_{i_1}^{p_1}(\xi_1)N_{i_2}^{p_2}(\xi_2)w_{\{i_1,i_2\}}}{\sum\limits_{j_1=1}^{n_1}\sum\limits_{j_2=1}^{n_2} N_{j_1}^{p_1}(\xi_1)N_{j_2}^{p_2}(\xi_2)w_{\{j_1,j_2\}}}$$

and we can then define a NURBS surface as:

$$\boldsymbol{S}(\boldsymbol{\xi}) = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2} R_{\mathbf{i}}(\boldsymbol{\xi})\mathbf{P}_{\mathbf{i}} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{N_{i_1}^{p_1}(\xi_1)N_{i_2}^{p_2}(\xi_2)w_{\mathbf{i}}}{\sum\limits_{j_1=1}^{n_1}\sum\limits_{j_2=1}^{n_2} N_{j_1}^{p_1}(\xi_1)N_{j_2}^{p_2}(\xi_2)w_{\mathbf{j}}}\mathbf{P}_{\mathbf{i}}$$

wherein $\mathbf{i} = \{i_1, i_2\}$ and $\mathbf{j} = \{j_1, j_2\}$ are a multi-indices over the NURBS basis functions. Fig. 2.3a shows a representative bivariate NURBS basis function, and Fig. 2.3b shows the corresponding control net and resulting NURBS surface.

2.1.2 B-Reps

As we have already discussed, solid objects in CAD packages are visually represented as a collection of surfaces. These representations of solid objects by a collection of surfaces are called

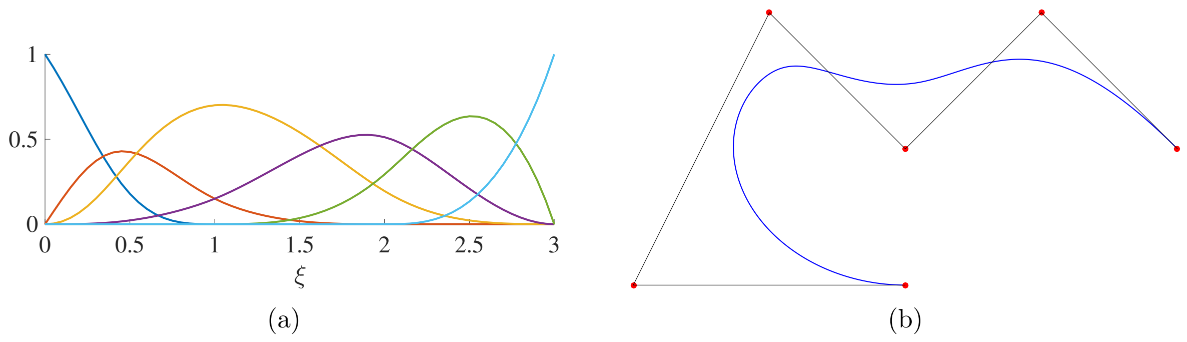


Figure 2.2: (a) The NURBS basis functions and (b) the corresponding NURBS curve.

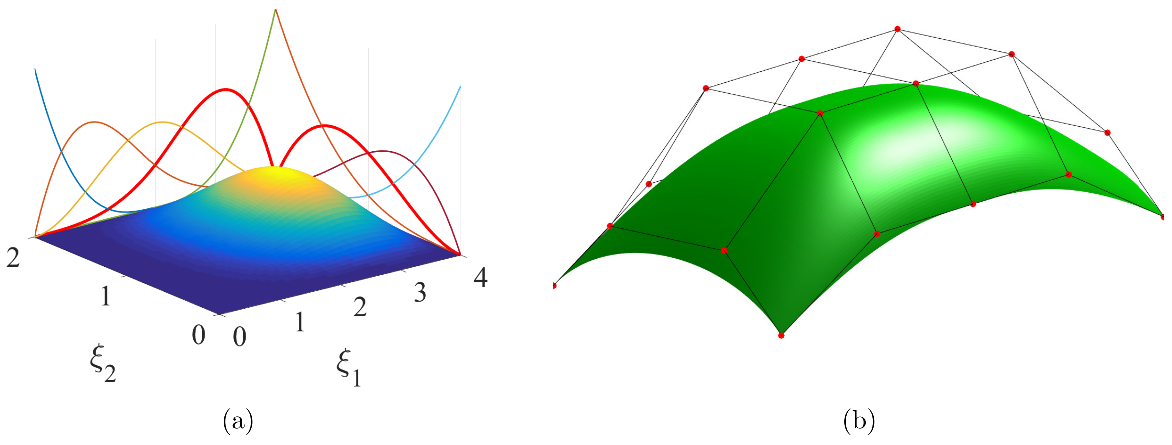


Figure 2.3: (a) The NURBS basis function $R_{\{4,3\}}$. (b) Corresponding NURBS control net and resultant NURBS surface.

boundary representations, or B-Reps. Along with NURBS, B-Reps have long been the standard in CAD packages, due to their relative ease of implementation and flexibility in representing a wide array of complex geometries.

When constructing B-Reps, NURBS surfaces are the predominant geometric entities used to represent the bounding surfaces. The collection of NURBS surfaces comprising a B-Rep is known as a multi-patch NURBS surface. With multipatch NURBS surfaces, we are able to represent a wide array of geometries. Fig. 2.4 shows some simple examples of multipatch geometries, with the individual NURBS patches shown in different colors. Although many geometries can be represented using multi-patch NURBS surfaces, each NURBS patch must be mappable to a rectangle in parametric space. As such, surfaces that cannot be mapped to a rectangle cannot be represented in this manner.

Traditionally, this limitation has been overcome by the introduction of trimming curves. Complex objects are first modeled using a collection of NURBS surfaces. In the locations where these surfaces intersect, the modeler introduces trimming curves to truncate the respective surfaces. The trimmed surfaces can then be connected along this trimming curve. We provide context for our discussion of B-Reps and trimming curves by considering the simple (yet commonly encountered) example of modeling the T-junctions of two cylinders, shown in Fig. 2.5. First, the NURBS surfaces defining the surfaces of the cylinder are created (Fig. 2.5a). Next, a trimming curve is created along the intersection of the two cylinders, and the overlapping segments are removed. Figs. 2.5b and c show this process, with the trimmed sections highlighted in red, and the resulting surface is shown in Fig. 2.5d. Next, the ends of the cylinders, are capped with trimmed NURBS surfaces (Fig. 2.5e). The final geometry is then defined by the volume bounded by the union of the trimmed surfaces, shown in Fig. 2.5f.

While NURBS and B-Reps are undoubtably powerful tools for CAD, as evidenced by their widespread use, there are some notable disadvantages to these representations. First, whereas a NURBS parameterization is **explicit**, the introduction of trimming curves causes the underlying parameterization to be **implicit**, which is undesirable from a numerical analysis standpoint. Ad-

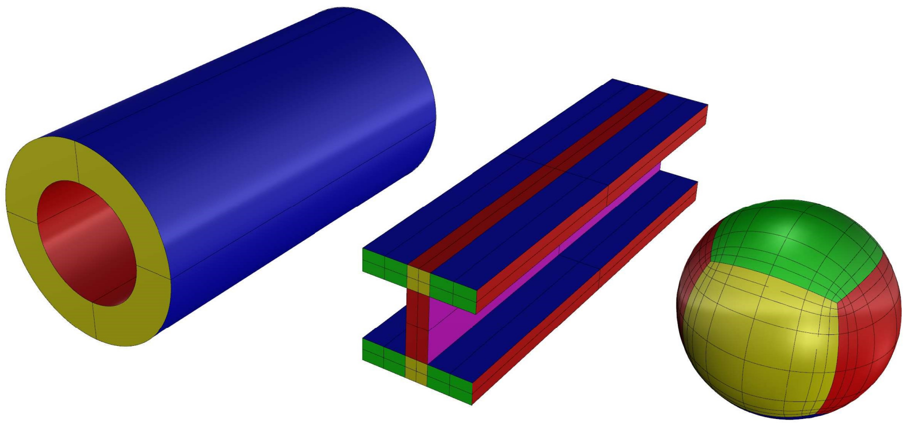


Figure 2.4: Examples of multipatch NURBS surfaces. Each NURBS patch is shown in a different color.

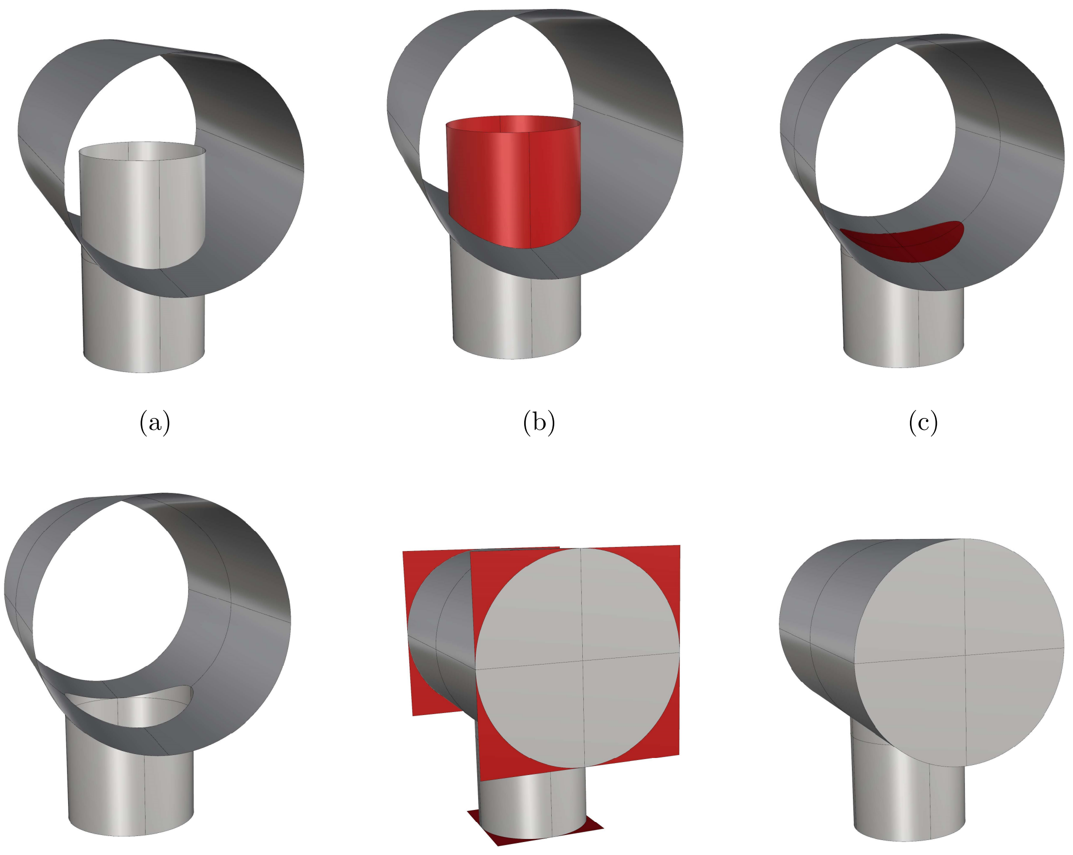


Figure 2.5: Creating a junctions of two cylinders using trimming curves.

ditionally, multi-patch geometries often have gaps and overlaps, and we therefore say they are not watertight. While these discontinuities in the geometry may not be large, and thus not readily apparent in CAD, they pose serious problems in the analysis framework. For instance, gaps and overlaps in geometry result in crack-tip stress concentrations in structural analysis and leakage in fluid flow simulations. Finally, it is impossible to perform local refinement of a NURBS surface. Because of the tensor product nature of the NURBS basis, any refinement must propagate across an entire NURBS patch. It is these shortcomings that motivated the advent of T-splines as an alternative to NURBS based B-Reps in CAD packages [62].

2.1.3 T-splines

T-splines are a generalization of NURBS that allow for T-junctions in the parameterization [7, 62]. To define a NURBS basis, one must simply define the knot vectors in each parametric direction that define the basis. Because the knot span is then defined as the tensor product of these knot vectors, NURBS are inherently a structured basis. T-splines on the other hand allow for more geometric flexibility by defining a set of **local** knot vectors for each basis function. Each local set is made up of d_r knot vectors, one in each parametric direction, with each knot vector defining a single basis function, N^l_α. Once we have defined the local knot vectors and their corresponding basis functions, we can then define a single d_r-variate basis function for each control point, $\mathbf{P}_\alpha$, viz:

$$B_\alpha(\boldsymbol{\xi}) = \prod_{l=1}^{d_r} N^l_\alpha(\xi^l)$$

Consider, for example, the point $\mathbf{P}_{\{3,4\}}$ with local knot vectors $\boldsymbol{\Xi}_1 = [1\ 2\ 3\ 4\ 5]$ and $\boldsymbol{\Xi}_2 = [2\ 3\ 4\ 5\ 6]$ shown in Fig. 2.6a. We can define the cubic basis functions N^1_3 and N^2_4 over their respective local knot vectors (Fig. 2.6b). The bivariate basis function $B_{\{3,4\}}$ is then defined as the tensor product of the two univariate basis functions, $B_{\{3,4\}} = N^1_3 \otimes N^2_4$ (Fig. 2.6c).

Thus, much like with NURBS surfaces, each control point $\mathbf{P}_\alpha$ in physical space will have a corresponding basis function defined in parametric space, B_α. Then, to define a T-spline surface, we need a set of T-spline control points, as well as a way to relate these control points to each other.

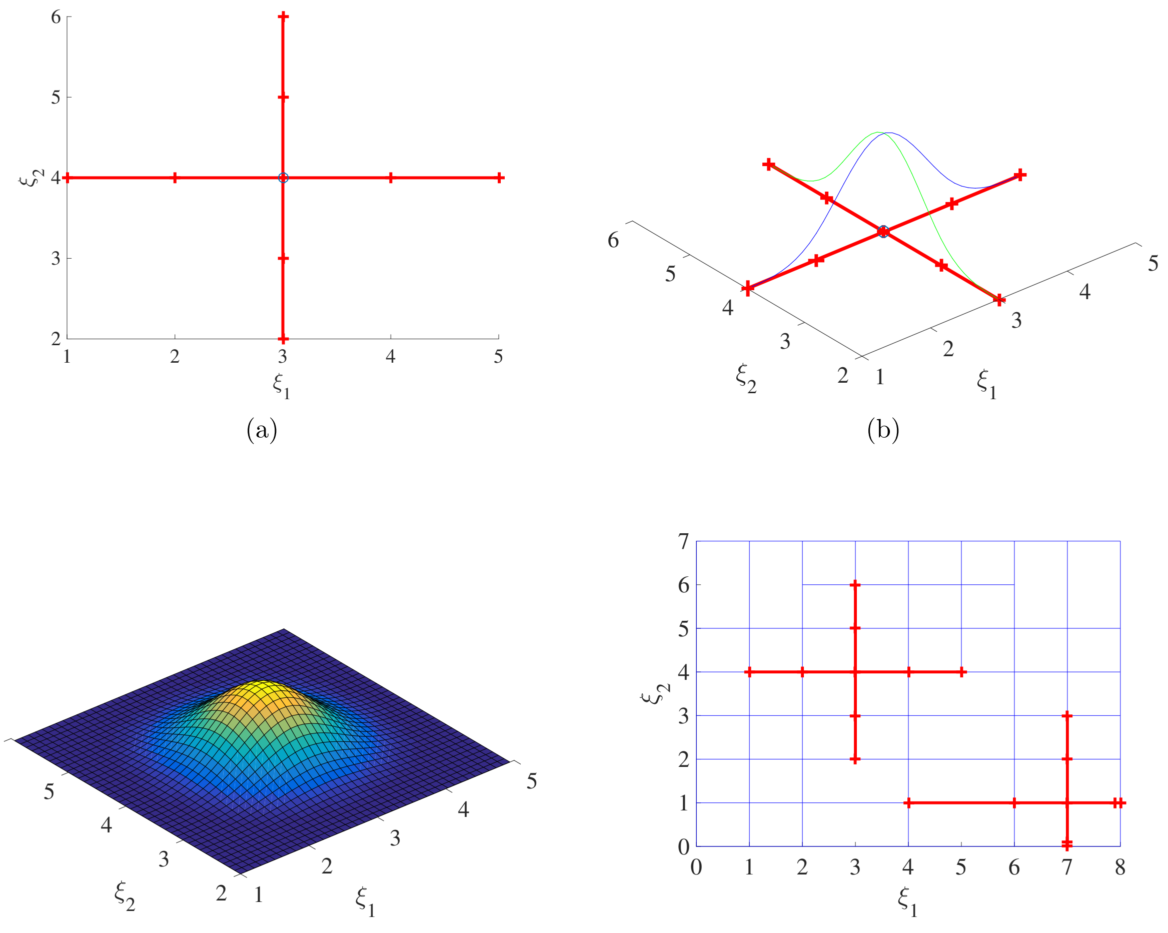


Figure 2.6: (a) The local knot vectors. (b) Local univariate basis functions defined over the knot vectors. (c) The resulting local bivariate basis function. (d) A T-mesh with the representative local knot vectors shown in red.

This is done via a T-mesh, which defines the topology and parameterization for the corresponding T-spline surface. A representative T-mesh for a cubic T-spline is shown in Fig. 2.6d. At this point, the theory for T-splines diverges slightly. For odd polynomial degree T-splines, each vertex in the T-mesh defines an anchor, s_α, and there is a direct mapping from every anchor in the T-mesh to every control point $\mathbf{P}_\alpha$ in physical space. For T-splines of even polynomial degree, the anchors are instead defined at the center of each face in the T-mesh. However, bi-cubic T-splines have come to be the de-facto standard in the CAD community, so we limit our discussion of T-splines to this case. Also shown in Fig. 2.6 are representative local knot vectors for the anchors $s_{\{3,4\}}$, and $s_{\{7,1\}}$, where the hash marks indicate the location of a knot. Observe that for a cubic T-spline, the local knot vectors are formed by extending a line in each parametric direction, until the line intersects with two orthogonal T-mesh lines in each direction. For the anchor $s_{\{3,4\}}$ the local knot vectors are $\boldsymbol{\Xi}_1 = [1\ 2\ 3\ 4\ 5]$ and $\boldsymbol{\Xi}_2 = [2\ 3\ 4\ 5\ 6]$. In the case of the anchor $s_{\{7,1\}}$, the local knot vectors are $\boldsymbol{\Xi}_1 = [4\ 6\ 7\ 8\ 8]$ and $\boldsymbol{\Xi}_2 = [0\ 0\ 1\ 2\ 3]$, where we have repeated knots because of the intersection with the T-mesh boundary.

Finally, as with NURBS, each T-spline control point has an associated control weight. Thus, for each anchor in the T-mesh, we define a set of T-spline blending functions:

$$R_\alpha(\xi) = \frac{w_\alpha B_\alpha(\xi)}{\sum\limits_{\beta \in A} w_\beta B_\beta(\xi)}$$

With these blending functions defined, we can then define our T-spline in physical space as:

$$S(\boldsymbol{\xi}) = \sum_{\alpha = A} \mathbf{P}_\alpha R_\alpha(\boldsymbol{\xi})$$

Fig. 2.7b shows a T-spline surface, and Fig. 2.7a shows the corresponding T-mesh. Note the direct mapping of anchors in the T-mesh to control points of the T-spline. Finally, it is worth emphasizing that NURBS are simply a restricted subset of T-splines that do not permit T-junctions. A substantial amount of work has been done in recent years on the applications of T-splines, both in the context of design and analysis. In this dissertation, we use T-splines only from the design point of view, as mathematical representations of surfaces from CAD. In this context, T-spline surfaces

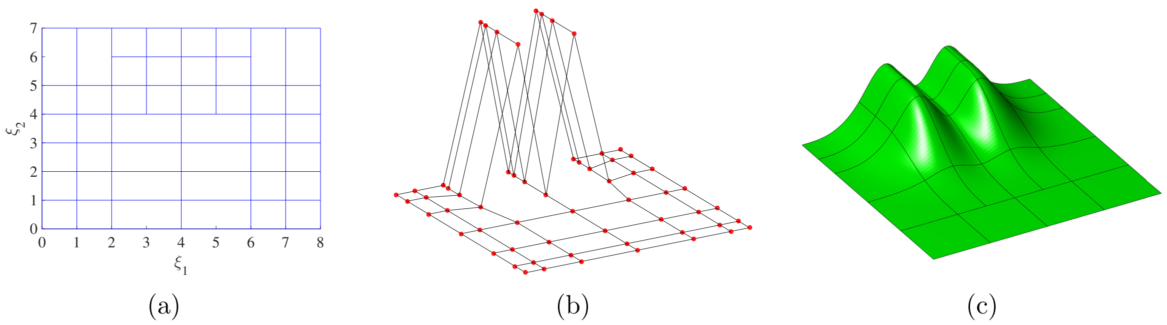


Figure 2.7: (a) The T-mesh in parametric space. (b) The T-spline control net. (c) The corresponding T-spline surface.

exhibit many of the desireable properties of NURBS surfaces such as affine covariance and a strong convex hull. However, is it worth mentioning that using T-splines as a basis for isogeometric analysis is an area of active research. As they are presented here, T-spline blending functions are not necessarily linearly independent. However, it has been shown that we can define a restricted subset of T-splines, called **analysis-suitable T-splines**, which are guaranteed to exhibit linear independence, point-wise non-negativity and partition of unity [40, 61]. Finally, as we will discuss at length in Section 2.3, much work has been done to solve the surface-to-volume parameterization problem using T-splines. While there have been many proposed methods for creating trivariate T-spline discretizations for the use in isogeometric analysis, this is still largely an unsolved problem. Because of this, we limit our treatment of T-splines in this dissertation to the bi-variate case, and focus on their use as purely a design technology for representing surfaces in CAD.

2.2 Mesh Generation

As stated previously, the standard practice in the current engineering workflow is to start with a design model from CAD. If some form of analysis needs to be performed on the design, the design model must first be discretized into a **mesh**. Generally speaking, a mesh is a collection of polygons (2D) or polyhedrons (3D) whose union provides an approximation to the area or volume of interest, respectively. Meshes can be broken into two broad classes: **structured** and **unstructured**.

Structured meshes are characterized by the fact that each vertex (with the exception of boundary vertices) in the mesh must have the same number of connecting edges. In two dimensions, these meshes are almost exclusively composed of quadrilaterals, where each vertex has exactly four connecting edges. Similarly, in three dimensions, structured meshes are almost always composed of hexahedra, with six edges connecting at each vertex. Structured meshes often are able to provide better accuracy per degree-of-freedom when compared to unstructured meshes. However, a severely limiting factor is that to create a structured mesh, a 2D domain must be mappable to a square, while a 3D domain must be mappable to a cube. As a result, this precludes the use of structured grids for many engineering applications that involve complex geometries.

Unstructured meshes, on the other hand, allow for any number of edges to be connected to any given vertex. More often than not, unstructured meshes are composed of triangular elements in 2D and tetrahedral elements in 3D. This is due to the fact that triangles and tetrahedra are the simplest geometric objects in each respective space, and therefore offer the most flexibility in the meshing process. Unstructured triangular and tetrahedral mesh generation is a well established field with a large body of literature from which we may draw inspiration. Generally speaking, unstructured mesh triangular and tetrahedral mesh generation techniques can be further classified into two subclasses: **mesh first** and **geometry first** methods.

In mesh first methods, unsurprisingly, the mesh is generated first. A mesh of triangles (2D) or tetrahedra (3D) is created that completely covers the domain of interest. This is usually using some form of quadtree decomposition in 2D [5, 58] or octree decomposition in 3D [79]. Then, the elements that intersect the boundary of the domain (defined by curves in 2D or surfaces in 3D) are manipulated so that their vertices lie on the boundary. The mesh first approach generally produces high quality elements throughout the interior of the mesh, but quality can suffer at the boundaries as a result of the mesh manipulation step.

In the geometry first approach, the geometry is discretized from the lowest dimensional geometric entities upwards. In the case of 2D mesh generation, this simply means boundary curves are first discretized into a collection of lines, and areas are then discretized into a collection of triangles. In 3D mesh generation, curves are first discretized into lines, then surfaces into triangles, and finally volumes into tetrahedra. Discretizing the domain into a collection of triangles or tetrahedra is most often done using either Delaunay techniques [8, 14] or advancing front techniques [44, 48]. Geometry first techniques are particularly useful when user control over mesh parameters is desired near domain boundaries, such as when creating a boundary layer mesh for fluid flow simulation.

In recent years, more and more research has been devoted to advanced meshing techniques, which extend mesh generation beyond meshes composed of only linear triangles or tetrahedra. In particular, mixed element mesh generation has been a research topic of great interest recently, with a focus on generating hexahedral dominant meshes [9, 67], as well as generating boundary layer meshes for fluid flow simulations, where hexahedral elements are used for the boundary layer, and tetrahedra are used for the free stream [30, 47, 57]. Additionally, curvilinear mesh generation is hardly limited to the purview of isogeometric analysis, and there has been a significant amount of previous work on curvilinear triangular and tetrahedral mesh generation using other methods [28, 34, 39, 45, 53, 56, 63, 64, 71, 76]. Despite this large volume of research into curvilinear mesh generation, there are two drawbacks to existing techniques that should be highlighted. First, existing curvilinear mesh generation techniques generally use polynomial elements, and are therefor unable to capture conic sections exactly. Secondly, most existing mesh generation approaches employ point interpolation in order to fit curvilinear elements to CAD geometries. This can lead to stability issues, including the presence of spurious surface oscillations, that may result from a poor choice of interpolation nodes [73].

Finally, we note that there exists a plethora of open-source meshing packages for both 2D and 3D mesh generation, such as MESH2D [23], Triangle [65], gmsh [28], and tetgen [66]. Additionally, we note that by no means are the works cited above the full extent of the field of mesh generation, nor is the listed software an exhaustive list of the available open source meshing software. Mesh generation has been an incredibly active area of research for many years now, and a complete review of the state of the art would be untenable. Instead, we have tried to list a few of the most impactful and relevant papers, and direct the interested reader to the references above for more information.

2.3 Isogeometric Analysis and Surface-to-Volume Parameterization

As previously stated, isogeometric analysis was originally conceived as a means to mitigate the design-to-analysis bottleneck inherent in the traditional engineering design cycle. The underlying concept is surprisingly simple. Rather than start with a design model from CAD and then discretize the model into a mesh for analysis, the analysis is performed directly on the CAD geometry (NURBS and T-spline surfaces). Furthermore, in FEA, the solution of interest is represented by a linear combination of basis function. Traditionally, when using linear triangular or tetrahedral meshes, the solution was likewise represented as a linear combination of linear triangular or tetrahedral basis functions. In isogeometric analysis, the solution is instead represented as a linear combination of NURBS or T-spline basis functions. Isogeometric analysis offers several distinct advantages over tradition FEA, namely:

(1) **Exact Geometry:** The analysis is performed on the exact geometric model from CAD, not a linear approximation in the form of a mesh.

(2) **NURBS as the Basis for Analysis:** NURBS have several desirable properties that make them advantageous as a basis for analysis, including linear independence, partition of unity, pointwise non-negativity and higher-order continuity.

(3) **A Tight Link with CAD:** There is a direct link with the original CAD model. This can be advantagous for several reasons:

   (a) If the analysis suggests that the design should be changed, only one model needs to be changed, rather than changing the CAD model and re-meshing.

   (b) If adaptive refinement is needed during analysis, the analysis does not need to query the CAD model.

   (c) This tight mathematical link between design and analysis eases the implementation of automatic shape optimization.

Recall that in CAD packages solids objects are usually represented by a collection of bounding surfaces, commonly referred to as boundary representations or B-reps. While these representations are adequate for providing a visual representation of objects, they do not provide an explicit parameterization of the interior volume. Thus arises the aforementioned problem of surface-to-volume parameterization. For our purposes, we define the goal of surface-to-volume parameterization as follows:

Given a NURBS or T-spline surface from CAD, automatically parameterize the corresponding volume such that:

(1) *The parameterization is explicit.*

(2) *The parameterization exactly matches the geometry.*

(3) *The parameterization is analysis suitable.*

(4) *The parameterization is defined by piecewise polynomial or rational basis functions that satisfy linear independence, partition of unity, positivity, and higher-order continuity.*

The four properties listed above are necessary for a parameterization to be suitable for isogeometric analysis. The surface-to-volume problem is non-trivial, and there exists a large body of work on attempts to solve this specific problem using trivariate B-spline, NURBS, and T-spline parameterizations. However, it is the opinion of this author that none of the previous work successfully addresses all of the goals listed above. Presented below is a short summary of the state of the art, as well as a short discussion of the advantages and disadvantages of each of the methods with respect to the goals listed above.

Early work into the problem of surface-to-volume parameterization focused on generating trivariate B-spline and NURBS discretizations. One approach, proposed by Aigner et. al., is to use swept volumes to create trivariate NURBS parameterizations, which has been shown to work well for a suite of relatively simple geometries, such as low-genus topologies, and geometries with small variations in cross-sectional profiles [1]. However, this method requires sweeping curves to be defined, and is therefore not automatic. Additionally, there is no study of analysis suitability of the parameterization, and the method is limited to the restrictive class of geometries described above. Another method, proposed by Martin et. al. [46], attempts to fit trivariate B-splines to tetrahedral meshes using harmonic functions. However, this method is not fully automatic, and is also not truly isogeometric as it uses trivariate B-splines to parameterize the volume, and therefore cannot exactly represent certain geometries of engineering interest, such as conic sections. Finally, any method for volume parameterization using trivariate NURBS or B-splines suffers from the same limitations as structured hexahedral mesh generation for classical FEM. As such, it is often impossible automatically generate analysis suitable meshes of complex geometries using NURBS or B-spline volumes.

Work has also been done to develop volumetric parameterizations using T-splines, which allow for local refinement via T-junctions in the parameterization. A notable example is the meccano method proposed by Escobar et. al. [24]. This method shows considerable promise, but the authors acknowledge further work needs to be done to enable automatic mesh generation of arbitrary genus solids, as well as to ensure analysis suitability of meshes.

Many methods for volume parameterization have also been proposed by the Zhang research group. Wang et. al. proposed a method for T-spline parameterizations from boundary triangulations [72]. This method has been successful in automatically parameterizing a wide variety of complex volumes, but still suffers from poor mesh quality in some cases. Liu et. al. proposed a method for T-spline construction using Boolean operations [42], but it has been shown that while this method can automatically parameterize many complex volumes, it is unable to generate parameterizations for certain classes of geometries, such as tetrahedra and cones. Finally, Liu et. al. recently proposed a method for T-mesh construction using skeleton-based polycubes [43], and while this method has been shown to produce analysis suitable parameterizations for a variety of geometries, user interaction is required in the preprocessing step.

A third area of research is to use Bernstein triangles and tetrahedra to generate volume parameterizations. Zeng et. al. have developed a hybrid meshing technology that creates a boundary layer mesh of trivariate B-splines and an internal mesh of Bézier tetrahedra [80]. George et.

al. proposed a method for generating tetrahedral meshes using 10-node Bézier tetrahedra [27]. However, both of these methods fail to be truly isogeometric in nature as neither use rational Bézier representations and therefore cannot exactly represent volumes defined by NURBS surfaces. Indeed, the ability to exactly represent conic sections is paramount for engineering applications, where geometries such as cylinders and spheres are commonplace.

We include here a table that briefly compares each method discussed above. Specifically, we evaluate each method based on how well it accomplishes the goals of surface-to-volume parameterization as outlined above, based on four criteria:

(1) **Automatic:** *Is the method capable of automatically creating a volumetric parameterization from a CAD surface without user interaction?*

(2) **Complex Geometries:** *Is the method capable of parameterizing complex geometries, such as arbitrary genus objects, and objects with large variations in length scales?*

(3) **Exact Geometry:** *Is the method capable of creating a geometrically exact parameterization of engineering geometries, including conic sections?*

(4) **Analysis Suitable:** *Is it demonstrated that the method produces analysis suitable geometries?*

For each method and each criteria, we say that the method either (a) fulfills the criteria (✓),(b) does not fulfill the criteria ($\times$), or (c) fulfills the criteria with some caveats ($\sim$). Of course, this ternary classification system obscures many of the nuances of each individual method. Additionally, terminology such as “automatic” and “complex geometry” is somewhat vague, and can vary from application to application. As such, Table 2.1 should be used as a quick visual reference to compare various methods, and interested readers are directed to the original works for more information.

In this work, we propose to address the problem of surface-to-volume parameterization through the use of unstructured Bernstein–Bézier discretizations. By employing rational Bernstein–Bézier geometric primitives, our approach combines the robustness of unstructured mesh generation with the advantageous properties of IGA, namely geometric exactness and a tight link between

Table 2.1: Comparison of existing surface-to-volume approaches.

	Automatic	Complex Geometries	Exact Geometry	Analysis Suitable
Swept Volumes [1]	×	×	✓	✓
NURBS from B-Reps [3]	✓	∼	×	∼
Trivariate B-splines [46]	∼	×	×	✓
Multi-Block Domain [77]	∼	×	×	✓
Hybrid B-spline/Tetrahedral Discretizations [80]	✓	✓	×	✓
Meccano Method [24, 25]	∼	∼	✓	∼
T-splines from Boundary Triangulations [72]	✓	×	✓	✓
T-splines From Boolean Operators [42]	✓	×	✓	✓
Skeleton Based Polycubes [43]	∼	∼	✓	✓

CAD and FEA. Furthermore, we note that the analysis suitability of mixed-element tetrahedral-hexahedral-pyramidal Bernstein-Bézier discretizations has already been demonstrated from a theoretical point of view, and it has also been shown that these discretizations benefit from good computational efficiency [2].

Many researchers have performed preliminary work on surface-to-volume parameterizations by first considering the two dimensional analog: curve-to-area parameterizations. Speleers et. al. have used Powell-Sabin splines over triangular meshes to solve advection-diffusion-reaction problems [68, 69]. While this paper presented a novel idea in the field of isogeometric analysis, the analysis was limited to very simple geometries. Jaxon et. al. presented similar work, using Powell-Sabin splines for linear elasticity and advection-diffusion problems, while also presenting the boundary replacement method for elevating linear triangular meshes to higher order Bernstein-Bézier discretizations [33, 75]. There has also been preliminary work into the use of rational Bernstein–Bézier tetrahedra for surface-to-volume parameterization. Recently, Xia and Qian have succeeded in creating geometrically exact, C^r-continuous discretizations using Bernstein-Bézier tetrahedra [74]. It should be noted, though, that C^1-continuous tetrahedral discretizations require the use of either nonic tetrahedra [81], or sextic tetrahedral macro-elements [36]. Moreover, achieving C^2-and-higher continuity requires the use of even higher order bases.

Our work on unstructured Bernstein–Bézier discretizations draws much of its inspiration from the related previous work, but it diverges in a few key areas. The previous work has focused on using Powell-Sabin or related splines to preserve higher order continuity of the basis over unstructured triangular or tetrahedral meshes. We choose instead to focus on generating mixed-element meshes composed of C^0-continuous Bernstein-Bézier elements. For many problems of practical interest, we believe the loss of C^k continuity is an acceptable tradeoff for the geometric flexibility and ease of implementation of C^0 Bernstein–Bézier elements. Moreover, we note that it is possible to combine our approach to surface-to-volume parameterization with the C^r-continuous tetrahedra of Xia and Qian to recover higher continuity, though this requires the use of relatively high order elements, and is limited to purely tetrahedral meshes.

Chapter 3

Bernstein–Bézier Elements

In this chapter, we present the rational Bernstein–Bézier elements used throughout the remained of this work. Before presenting the definitions for Bernstein–Bézier elements, we first review multi-index notation in Section 3.1. With this notation established, we present the general Bernstein–Bézier form in Section 3.2, the present the explicit definitions for various elements in Sections 3.3–3.6.

3.1 Multi-Index Notation

Throughout this dissertation, we make heavy use of multi-index notation in order to simplify the equations presented in the sections that follow. So that the meaning of the equations is unambiguous, we review this notation here. Let $\mathbf{n}$ and $\mathbf{k}$ denote $1 \times d$ multi-indices of natural numbers, $\mathbf{n} = \{n_1, ..., n_d\}$ and $\mathbf{k} = \{k_1, ..., k_d\}$, and let $\boldsymbol{x}$ denote a $1 \times d$ vector of real numbers. Then we use the following notation to denote several common vector operations.

$$\mathbf{n}! = \prod_{i=1}^{d} n_i!$$

$$\boldsymbol{x}^{\mathbf{n}} = \prod_{i=1}^{d} x_i^{n_i}$$

$$x^{\mathbf{n}} = \prod_{i=1}^{d} x^{n_i}$$

$$|\mathbf{n}| = \sum_{i=1}^{d} n_i$$

$$\binom{\mathbf{n}}{\mathbf{k}} = \prod_{i=1}^{d} \binom{n_i}{k_i}$$

It is also beneficial to introduce a barycentric multi-index, $\Bbbk$, which is a multi-index of length $d+1$ that satisfies the property $|\Bbbk| = n$, where n is a positive integer. The conversion from $\mathbf{k}$ to $\Bbbk$ is given by the equation:

$$\Bbbk = \{\mathbf{k}\}_{\Bbbk}^{n} = \{\mathbf{k}, n - |\mathbf{k}|\}$$

When both $\mathbf{k}$ and $\Bbbk$ are used in the same equation this conversion is implied. For a barycentric multi-index, we can define the multinomial coefficient $\binom{n}{\Bbbk}$:

$$\binom{n}{\Bbbk} = \frac{n!}{\Bbbk!}$$

3.2 The Bernstein–Bézier Form

Generally speaking, to define an arbitrary Bézier element Ω^e, we need three items:

(1) Bernstein basis functions defined over a reference element in parametric space $\{B_{\mathbf{i}}(\boldsymbol{\xi})\}_{\mathbf{i}\in I}$,

(2) A control net defining the element in physical space $\{\mathbf{P}_{\mathbf{i}}^{b}\}_{\mathbf{i}\in I}$, and

(3) Corresponding weights for each control point in the control net $\{w_{\mathbf{i}}\}_{\mathbf{i}\in I}$.

Here, I is an index set defining the ordering of the control points on the reference geometry. Then, a Bernstein polynomial is defined as:

$$b(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} B_{\mathbf{i}}(\boldsymbol{\xi})\, \beta_{\mathbf{i}} \quad \forall\ \boldsymbol{\xi} \in \hat{\Omega}$$

Now, let us define a set of control points $\{\mathbf{P}_{\mathbf{i}}^{b}\}_{\mathbf{i}\in I}$ in $\mathbb{R}^{d_s}$. Then, a Bézier element is simply defined through the mapping:

$$\mathbf{x}^{e}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} B_{\mathbf{i}}(\boldsymbol{\xi})\, \mathbf{P}_{\mathbf{i}}^{b}$$

Thus a Bézier element is defined via a polynomial pushforward mapping from parametric space, $\mathbb{R}^{d_r}$ to physical space, $\mathbb{R}^{d_s}$. We note that in general, this mapping holds for any $d_r \leq d_s$. When $d_r = 1$ we have a curve, when $d_r = 2$ we have triangles or quadrilaterals, and when $d_r = 3$ we have tetrahedra, hexahedra, wedges, and pyramids.

In addition to the Bernstein basis functions and control points, let $\{w_{\mathbf{i}}\}_{\mathbf{i}\in I}$ denote a set of **control weights** corresponding to $\{B_{\mathbf{i}}(\boldsymbol{\xi})\}_{\mathbf{i}\in I}$. Then we can define a set of corresponding **rational** Bernstein basis functions as:

$$R_{\mathbf{i}}(\boldsymbol{\xi}) = \frac{B_{\mathbf{i}}(\boldsymbol{\xi})w_{\mathbf{i}}}{\sum\limits_{\mathbf{j}\in I} B_{\mathbf{j}}(\boldsymbol{\xi})w_{\mathbf{j}}} = \frac{B_{\mathbf{i}}(\boldsymbol{\xi})w_{\mathbf{i}}}{w(\boldsymbol{\xi})}$$

where $w(\boldsymbol{\xi}) = \sum\limits_{\mathbf{j}\in I} B_{\mathbf{j}}(\boldsymbol{\xi})w_{\mathbf{j}}$ denotes the **weighting function** defined over the domain $\hat{\Omega}$. Then, a rational Bernstein-Bézier element is defined by the mapping:

$$\mathbf{x}^e(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} R_{\mathbf{i}}(\boldsymbol{\xi})\mathbf{P}^b_{\mathbf{i}} = \frac{B_{\mathbf{i}}(\boldsymbol{\xi})\mathbf{P}^b_{\mathbf{i}}w_{\mathbf{i}}}{\sum\limits_{\mathbf{j}\in I} B_{\mathbf{j}}(\boldsymbol{\xi})w_{\mathbf{j}}}$$

When working with rational Bézier elements, it is often convenient to consider the corresponding polynomial element in **projective space**, $\widetilde{\Omega}^e \in \mathbb{R}^{d+1}$. The projective element is defined by the mapping:

$$\widetilde{\mathbf{x}}^e(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} B_{\mathbf{i}}(\boldsymbol{\xi})\widetilde{\mathbf{P}}^b_{\mathbf{i}}$$

wherein $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I}$ is the set of projective control points, defined as:

$$\begin{aligned} \left(\widetilde{\mathbf{P}}^b_{\mathbf{i}}\right)_j &= w_{\mathbf{i}}\left(\mathbf{P}^b_{\mathbf{i}}\right)_j \qquad j \in [1, d] \\ \left(\widetilde{\mathbf{P}}^b_{\mathbf{i}}\right)_{d+1} &= w_{\mathbf{i}} \end{aligned}$$

This allows us to write a rational Bézier element in $\mathbb{R}^d$ as the projective transformation of a polynomial Bézier element in $\mathbb{R}^{d+1}$, viz:

$$\mathbf{x}^e(\boldsymbol{\xi}) = \frac{[\widetilde{\mathbf{x}}^e(\boldsymbol{\xi})]_d}{w(\boldsymbol{\xi})}$$

wherein, $[\widetilde{\mathbf{x}}^e(\boldsymbol{\xi})]_d$ denotes the first d components of $\widetilde{\mathbf{x}}^e(\boldsymbol{\xi})$. We illustrate the control points for a rational Bernstein–Bézier element and the corresponding element in projective space in Fig. 3.1.

The Bernstein polynomials, and the resulting rational Bernstein–Bézier basis functions, exhibit several desirable properties that make them suitable as a set of basis functions for finite element analysis. These include partition of unity, point-wise non-negativity, linear independence and compact support. From our general definition of a Bézier element in Eq. (3.2), together with

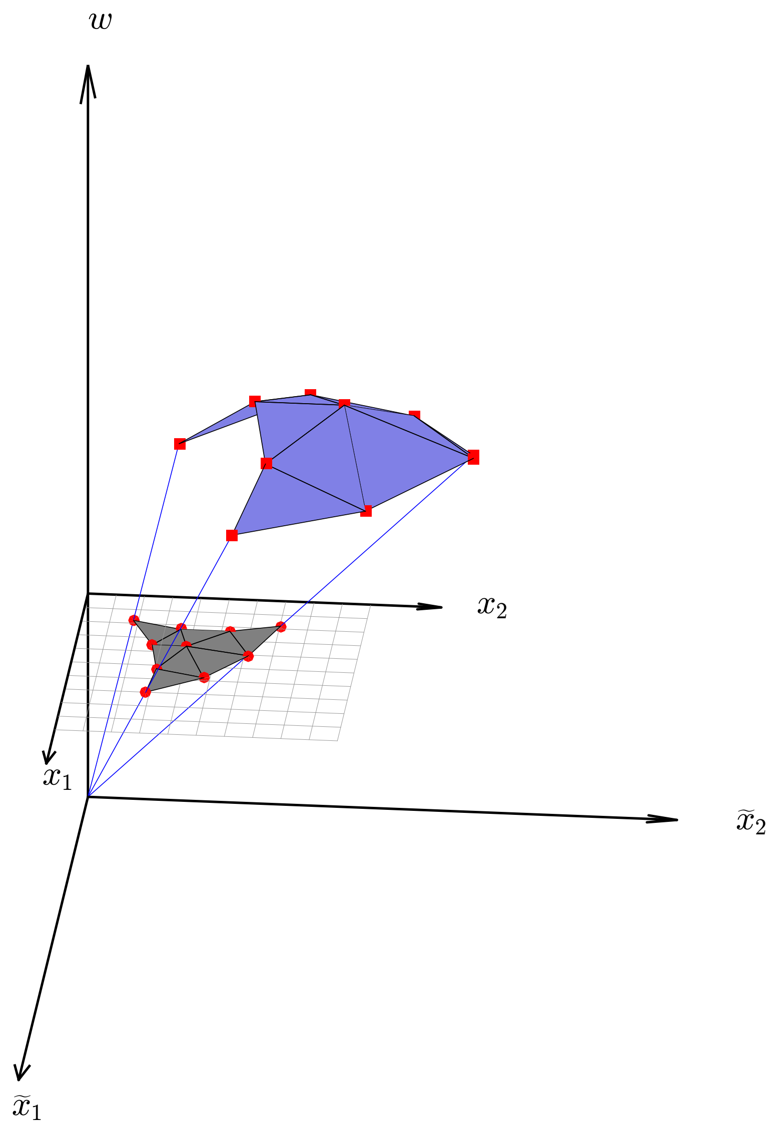


Figure 3.1: Projective transformation from the control net for a projective element $\widetilde{\Omega}^e \in \mathbb{R}^{d+1}$ (shown in blue), to the physical element $\Omega^e \in \mathbb{R}^d$ (shown in grey). Points on the projective element are given in projective coordinates $\{\widetilde{x}_1, ..., \widetilde{x}_d, w\}$. The physical element is embedded in the $w = 1$ plane (gray grid), and points on the physical element are given in terms of the physical coordinates $\{x_1, ..., x_d\}$.

the properties of the Bernstein polynomials, it can be shown that any Bézier element will exhibit affine covariance, and a strong convex hull. Most importantly, note that in order to define an arbitrary Bézier element, we need only to be able to define the Bernstein polynomials over a corresponding parametric reference geometry. This also illuminates another important property of all Bézier elements: the boundary of a Bézier element is simply another Bézier element. That is, the faces of Bézier hexahedra are simply Bézier quadrilaterals, and the faces of Bézier tetrahedra are simply Bézier triangles. Likewise, the edges of Bézier quadrilaterals and triangles are simply Bézier curves.

At this juncture, it is useful to make a few remarks regarding terminology. We use the clarifier "Bernstein" when referring to the Bernstein polynomial basis and the clarifier "Bézier" when referring to an element in physical space or the control net defining said element. We use the clarifier "Bernstein–Bézier" when referring to the rational basis or the parametrization of the element as these involve both the Bernstein basis and the Bézier control net.

3.3 Simplicial Elements

The simplest Bernstein-Bézier elements are the simplicial elements: curves (1-simplices), triangles (2-simplices), and tetrahedra (3-simplices). When working with simplicial elements, it is most common to work in barycentric coordinates. However, as we are particularly interested in the derivatives of Bernstein polynomials with respect to the unit Cartesian reference triangle, it is also convenient to work explicitly in terms of Cartesian coordinates. As such, we present both forms, and switch between forms as necessary.

3.3.1 Cartesian Coordinates

We define the reference domain for a d-simplex as:

$$\hat{\Omega} = \left\{ \boldsymbol{\xi} \in \mathbb{R}^d : 0 \leq \xi_j \leq 1, \sum_{j=1}^{d} \xi_j \leq 1 \right\}$$

and define the index set for the simplicial Bernstein polynomials of degree p as:

$$I^p := \{\mathbf{i} = \{i_1, ..., i_d\} : 0 \leq i_j \leq p, |\mathbf{i}| \leq p\}$$

Then, we can define the simplicial Bernstein basis polynomials as:

$$B^p_{\mathbf{i}}(\boldsymbol{\xi}) = p! \prod_{j=1}^{d} \frac{(\xi_j)^{i_j}}{i_j} \frac{(1 - |\boldsymbol{\xi}|)^{p-|\mathbf{i}|}}{p - |\mathbf{i}|}$$

3.3.2 Barycentric Coordinates

For a d-simplex, we define the set of $d+1$ barycentric coordinates $\{\lambda_j\}_{j=1}^{d+1}$ to be:

$$\lambda_j = \xi_j \quad j \in [0, d]$$
$$\lambda_{d+1} = 1 - \sum_{j=1}^{d} \lambda_j$$

Then the reference domain is defined as:

$$\hat{\Omega} = \{\mathbb{\lambda} \in \mathbb{R}^{d+1} : 0 \leq \lambda_j \leq 1, |\mathbb{\lambda}| = 1\}$$

and the index set for the simplicial Bernstein polynomials of degree p is defined as:

$$I^p_{\mathbb{i}} := \{\mathbb{i} = \{i_1, ..., i_{d+1}\} : 0 \leq i_j \leq p, |\mathbb{i}| = p\}.$$

This allows us to write the simplicial Bernstain polynomials incredibly compactly as:

$$B^p_{\mathbb{i}}(\mathbb{\lambda}) = \binom{p}{\mathbb{i}} \mathbb{\lambda}^{\mathbb{i}}$$

For simplicial elements, we differentiate between the Cartesian and barycentric forms by either explicitly specifying the dependence on $\boldsymbol{\xi}$ or $\mathbb{\lambda}$, or implicitly by the index $\mathbf{i}$ or $\mathbb{i}$. The index $\mathbf{i}$ is taken to always be a multi-index of length d, whereas we understand $\mathbb{i}$ to denote a multi-index of length $d+1$ such that $|\mathbb{i}| = p$. Figure 3.2 shows examples of two simplicial Bernstein–Bézier elements, a Bernstein–Bézier triangle and a Bernstein–Bézier tetrahedron.

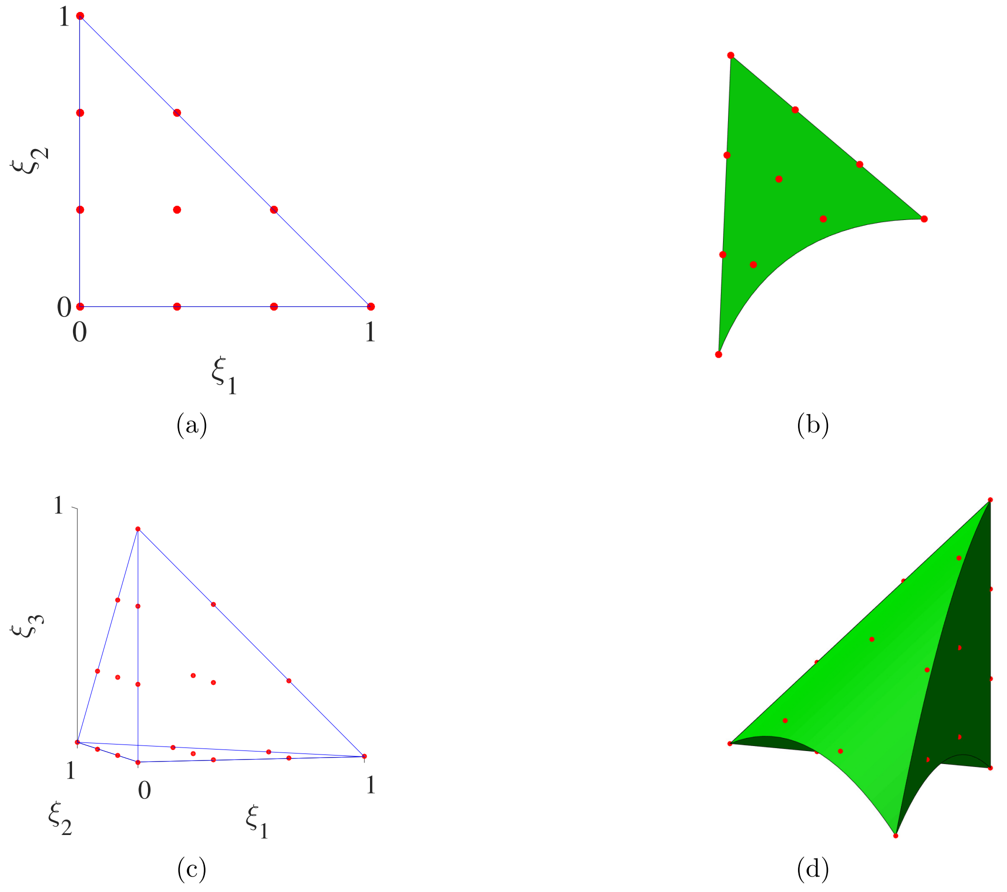


Figure 3.2: Simplicial Bézier element gallery. (a) The cubic triangular control net in parametric space. (b) A cubic Bézier triangle in physical space. (c) The cubic tetrahedral control net in parametric space. (d) A cubic Bézier tetrahedra in physical space.

3.4 Tensor Product Elements

We define Bernstein–Bézier quadrilaterals in $\mathbb{R}^2$ and hexahedra in $\mathbb{R}^3$ using a tensor product construction. The reference domain is given by:

$$\hat{\Omega} = (0,1)^d$$

and the index set for a tensor product construction is given by:

$$I^{\mathbf{p}} := \{\mathbf{i} = \{i_1, ..., i_d\} \mid 0 \le i_j \le p_j\}.$$

Then, the Bernstein basis polynomials are defined as:

$$B_{\mathbf{i}}^{\mathbf{p}}(\boldsymbol{\xi}) = \prod_{j=1}^{d} B_{i_j}^{p_j}(\xi_j)$$

where $B_{i_j}^{p_j}$ are simply the univariate Bernstein polynomials from Section 3.3. We distinguish between simplicial and tensor product constructions implicitly by using a scalar superscript p for the simplicial basis functions $B_{\mathbf{i}}^{p}$, and a vector super script $\mathbf{p}$ for the tensor product basis functions $B_{\mathbf{i}}^{\mathbf{p}}$. Figure 3.3 shows examples of the two tensor product Bernstein–Bézier elements, a Bernstein–Bézier quadrilateral and a Bernstein–Bézier hexahedron.

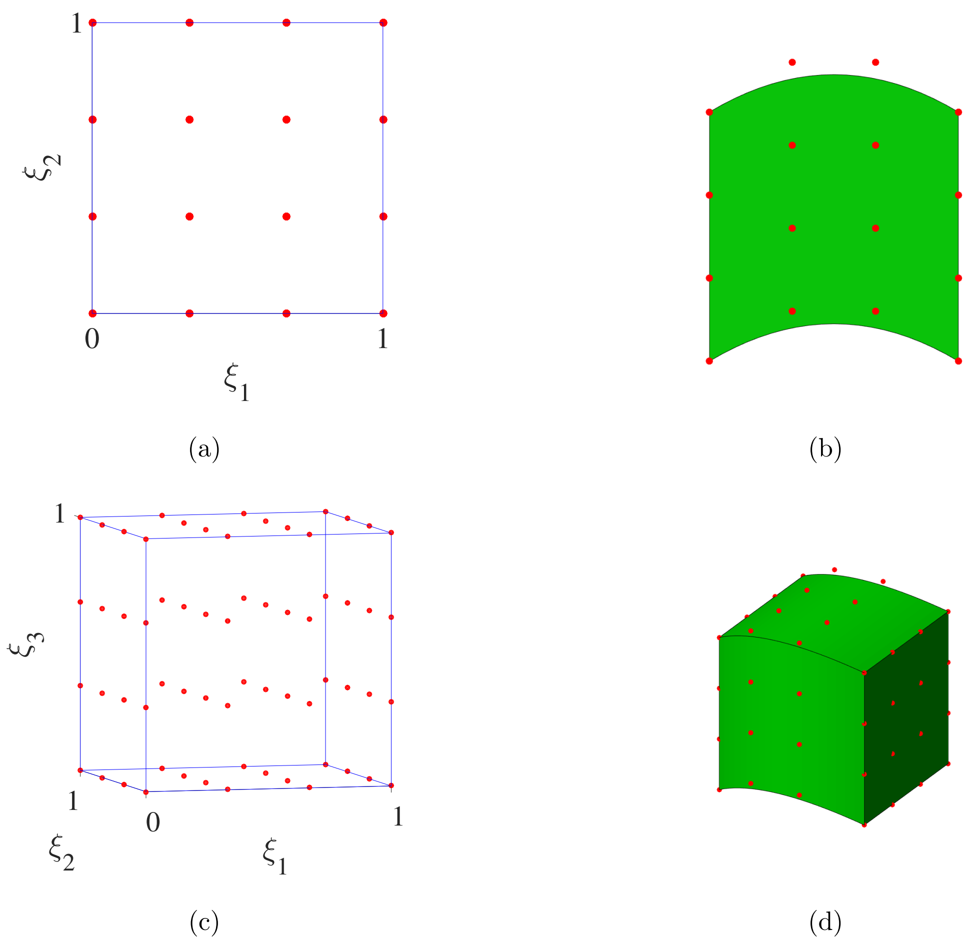


Figure 3.3: Tensor product Bézier elements. (a) The bicubic quadrilateral control net in parametric space. (b) A bicubic Bézier quadrilateral in physical space. (c) The tricubic hexahedral control net in parametric space. (d) A tricubic Bézier hexahedra in physical space.

3.5 Wedges

A Bézier wedge is defined from the tensor product of Bernstein polynomials over a triangle with the univariate Bernstein polynomials. Thus, we define the reference wedge in parameter space as:

$$\hat{\Omega} = \{(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3 : \xi_1, \xi_2, \xi_3 \geq 0, \xi_1 + \xi_2 \leq 1, \xi_3 \leq 1\}$$

The index set over the Bernstein basis polynomials is given by:

$$I^{\{p_{1,2}, p_3\}} := \{\mathbf{i} = \{i_1, ..., i_d\} \mid 0 \leq i_1 + i_2 \leq p_{1,2}, 0 \leq i_3 \leq p_3\}.$$

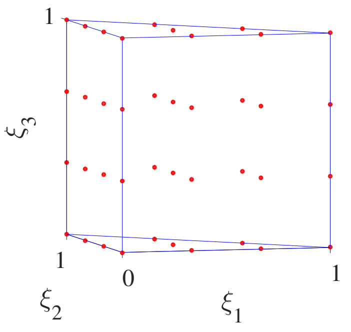
and the Bézier wedge of degree p in physical space is defined by the mapping:

$$\mathbf{x}^e(\boldsymbol{\xi}) = \sum_{\mathbf{i} \in I^{\{p_{1,2}, p_3\}}} R_{\mathbf{i}}^{wedge}(\boldsymbol{\xi}) \mathbf{P}_{\mathbf{i}}^b \quad \forall \boldsymbol{\xi} \in \hat{\Omega}$$

where the basis functions $\left\{ R_{\mathbf{i}}^{wedge}(\boldsymbol{\xi}) \right\}_{\mathbf{i} \in I^{\{p_{1,2}, p_3\}}}$ are defined from the tensor product of triangular Bernstein basis functions and univariate Bernstein basis functions, viz.:

$$R_{\mathbf{i}}^{wedge}(\boldsymbol{\xi}) = \frac{B_{\{i_1,i_2\}}^{p_{1,2}}(\xi_1, \xi_2) B_{i_3}^{p_3}(\xi_3) w_{\{i_1,i_2\}} w_{i_3}}{\sum_{\mathbf{j} \in I^{\{p_{1,2}, p_3\}}} B_{\{j_1,j_2\}}^{tri}(\xi_1, \xi_2) B_{j_3}(\xi_3) w_{\{j_1,j_2\}} w_{j_3}}$$

Figure 3.4a shows the control net for the cubic Bézier wedge in parametric space, and Figure 3.4b shows a corresponding cubic Bézier wedge in physical space.



(a)

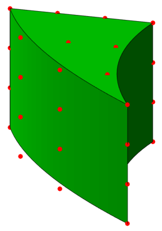

(b)

Figure 3.4: A cubic Bézier wedge in (a) parametric space and (b) physical space.

3.6 Pyramids

Bézier pyramids take more care to define than the other primitives considered in this dissertation. First, let us define the unit reference pyramid as:

$$\hat{\Omega}^{pyr} = \{(\xi_1, \xi_2, \xi_3) \in \mathbb{R}^3 : \xi_1, \xi_2, \xi_3 \geq 0, \xi_1 + \xi_3 \leq 1, \xi_2 + \xi_3 \leq 1, \xi_3 \leq 1\}$$

However, unlike the other primitives considered in the dissertation, we do not have an immediate definition of Bernstein polynomials over the reference pyramid $\hat{\Omega}^{pyr}$. Rather, Chan et. al. [13]

recently showed that the Bernstein basis of degree p for a pyramid can first be defined over the unit cube as:

$$B_{\mathbf{i}}(\boldsymbol{a}) = B_{i_1}^{p-i_1}(a)B_{i_2}^{p-i_2}(b)B_{i_3}^{p}(c) \qquad \forall \boldsymbol{a} = (a,b,c) \in (0,1)^3$$

We can then map the basis to the reference pyramid via the Duffy transform. That is, the coordinates (a, b, c) are found from (ξ_1, ξ_2, ξ_3) as:

$$a = \frac{\xi_1}{1-\xi_3}$$
$$b = \frac{\xi_2}{1-\xi_3}$$
$$c = \xi_3$$

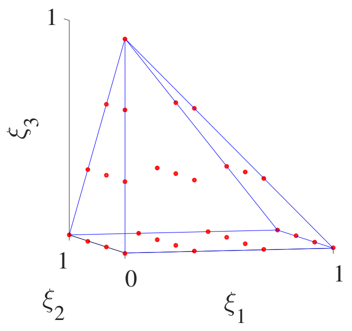
and we can then define the rational Bernstein-Bézier basis over the pyramid as:

$$R_{\mathbf{i}}^{pyr}(\boldsymbol{a}) = \frac{B_{i_1}^{p-i_3}(a)B_{i_2}^{p-i_3}(b)B_{i_3}^{p}(c)w_{\mathbf{i}}}{\sum\limits_{j_3=0}^{p}\sum\limits_{j_1=0}^{p-j_3}\sum\limits_{j_2=0}^{p-j_3} B_{j_1}^{p-j_3}(a)B_{j_2}^{p-j_3}(b)B_{j_3}^{p}(c)w_{\mathbf{j}}}$$

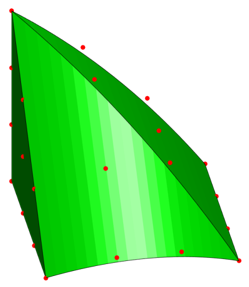
We can finally define a Bernstein–Bézier pyramid of degree p using the mapping:

$$\mathbf{x}^e(\boldsymbol{\xi}) = \sum_{i_3=0}^{p}\sum_{i_2=0}^{p-i_3}\sum_{i_1=0}^{p-i_3} R_{\mathbf{i}}^{pyr}(\boldsymbol{a}(\boldsymbol{\xi}))\mathbf{P}_{\mathbf{i}}^{b} \qquad \forall \boldsymbol{\xi} \in \hat{\Omega}^{pyr}$$

The Duffy transform can also be employed to define the Bernstein basis for a triangle by collapsing the coordinate system for a quadrilateral. Under the Duffy transform, the polynomial basis over the unit cube is transformed to a rational basis over the reference pyramid. Nevertheless, it can be shown that this rational basis is complete in that it contains all polynomials of degree p. Moreover, it can be shown that over each triangular face of the pyramid, the trace of the rational basis is spanned by the triangular Bernstein basis of degree p, and over the rectangular face of the pyramid, the trace is spanned by the tensor-product Bernstein basis of degree (p, p). It can also be demonstrated that the Bernstein basis over a pyramid maintains all the beneficial properties of the Bernstein polynomials, such as partition of unity, point-wise non-negativity, and linear independence. One very important property of the Bézier pyramid is that it serves as a transition element between Bézier hexahedra and Bézier tetrahedra. Figure 3.5a shows the control net for the cubic Bézier wedge in parametric space, and Figure 3.5b shows a corresponding cubic Bézier wedge in physical space.



(a)



(b)

Figure 3.5: A cubic Bézier pyramid in (a) parametric space and (b) physical space.

Chapter 4

Three Dimensional Mesh Generation

In this chapter, we present the first novel contribution of this dissertation, geometrically exact volumetric mesh generation. We begin by discussing Bézier extraction and Bézier projection, two numerical techniques central to this work. Then, we introduce the notion of geometric polynomial complexity, which is imperative for ensuring that a mesh will be geometrically exact. Next, we present the main contribution of this chapter: a framework for geometrically exact mesh generation and mesh refinement. Finally, we conclude this section with some numerical examples demonstrating the analysis suitability of these meshes.

4.1 Bézier Extraction and Bézier Projection

Thus far, we have presented NURBS and T-spline surfaces as methods for representing objects in CAD, and we have presented various Bézier primitives used to create analysis suitable meshes. However, it remains to establish a relationship between these global NURBS and T-spline representations, and the local Bézier elements. Specifically, we would like to establish a relationship between NURBS curves and Bézier curves, and a relationship between NURBS or T-spline surfaces and Bézier surfaces. Because the faces of Bézier hexahedra, tetrahedra, wedges and pyramids are themselves Bézier quadrilaterals or triangles, this will then give us a relationship between CAD surfaces (NURBS/T-splines) and the faces of the Bézier meshing primitives presented above. The motivation for establishing this relationship will become clear in Section 4.3.3, where we use face swapping on Bézier elements to reconstruct the surface mesh to match the CAD geometry. For now, we introduce two tools necessary to do this task: Bézier extraction and Bézier projection.

In this section, and throughout the rest of this dissertation, it will be useful to introduce some additional notation. When working with sets of points, we often find it useful to arrange the members of the set into a matrix. We denote this matrix representation by dropping the index on the variable. That is, for a set of Bézier points, we write the matrix representation as:

$$\mathbf{P}^b = \begin{bmatrix} \left(\mathbf{P}_1^b\right)^T \\ \left(\mathbf{P}_2^b\right)^T \\ . \\ . \\ \left(\mathbf{P}_n^b\right)^T \end{bmatrix} = \begin{bmatrix} \left(\mathbf{P}_1^b\right)_1 \left(\mathbf{P}_1^b\right)_2 \cdots \left(\mathbf{P}_1^b\right)_{d_s} \\ \left(\mathbf{P}_1^b\right)_2 \left(\mathbf{P}_2^b\right)_2 \cdots \left(\mathbf{P}_2^b\right)_{d_s} \\ . \\ . \\ \left(\mathbf{P}_n^b\right)_1 \left(\mathbf{P}_n^b\right)_2 \cdots \left(\mathbf{P}_n^b\right)_{d_s} \end{bmatrix}$$

4.1.1 Bézier Extraction

Bézier extraction was introduced by Borden et al. [10] as a means to express the relationship between the B-spline basis and the Bernstein basis via an element-wise extraction operator, $\mathbf{C}^e$. Consider the set of univariate B-spline basis functions $\{N_i\}_{i=1}^n$, and corresponding control points $\{\mathbf{P}_i\}_{i=1}^n$. Over any given knot interval $\xi \in [\xi_m, \xi_{m+1})$, there exist a set of $p+1$ non-zero basis functions:

$$\{N_i^e\}_{i=0}^p := \{N_i | N_i(\xi) > 0 \; \forall \; \xi \in [\xi_m, \xi_{m+1})\}$$

We can then define these local B-spline basis functions $\{N_i^e\}_{i=0}^p$ in terms of the Bernstein basis via the element extraction operator:

$$N_i^e = \sum_j \mathbf{C}_{ij}^e B_j$$

Now let us also denote the set of $p+1$ local control points that correspond to to the element B-spline basis functions as $\{\mathbf{P}_i^e\}_{i=0}^p$. We can then define the relationship between the element B-spline control points and the Bézier control points for the current element, $\{\mathbf{P}_i^{b,e}\}_{i=0}^p$ by the equation:

$$\mathbf{P}^{b,e} = (\mathbf{C}^e)^T \mathbf{P}^e$$

When performing Bézier extraction on NURBS curves, the process is the same, but we perform extraction on the homogenous element NURBS control points $\{\widetilde{\mathbf{P}}_i^e\}_{i=0}^p$ to arrive at the homogenous Bézier control points $\{\widetilde{\mathbf{P}}_i^{b,e}\}_{i=0}^p$, viz:

$$\widetilde{\mathbf{P}}^{b,e} = (\mathbf{C}^e)^T \widetilde{\mathbf{P}}^e \tag{4.1}$$

where we can find the homogenous NURBS control points via Equation (4.3). Finally, the Bézier control points $\{\mathbf{P}_i^{b,e}\}_{0=1}^p$ and weights $\{w_i^{b,e}\}_{i=0}^p$, are recovered by:

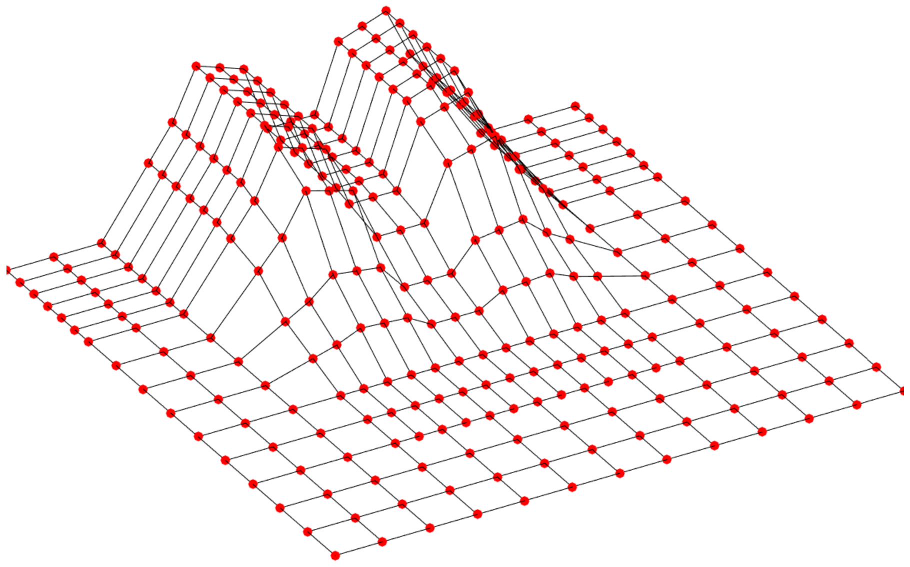
$$(\mathbf{P}_i^{b,e})_j = \frac{(\widetilde{\mathbf{P}}_i^{b,e})_j}{w_i^{b,e}} \qquad i \in \{0, ..., p\}, j \in \{1, ..., d_s\}$$

$$w_i^b = (\widetilde{\mathbf{P}}_i^{b,e})_{d_s+1}$$

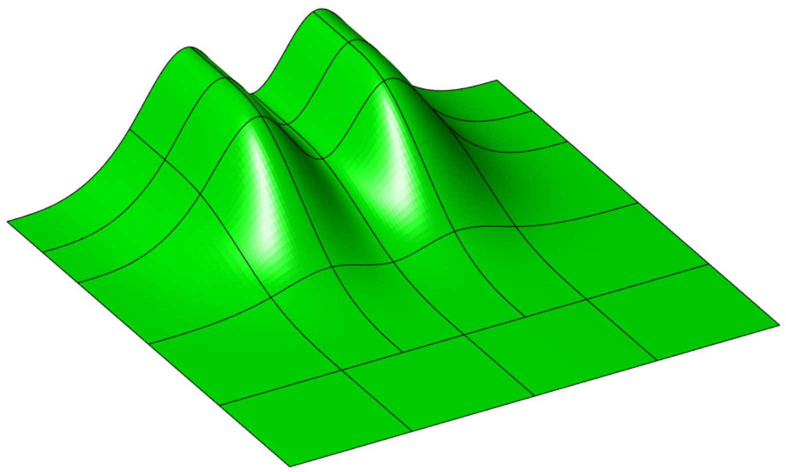
A nice property of Bézier extraction is that it extends directly to the multivariate case from the univariate case. In fact, in order to perform Bézier extraction of NURBS or T-spline surfaces, Equation (4.1) still holds, and we must simply define the appropriate element extraction operators. This is powerful as it allows any NURBS or T-spline geometry to be defined by a collection of Bézier elements, provided an extraction operator can be computed. Figure 4.1 shows the result of performing Bézier extraction on the T-spline surface from Figure 2.7. Note that the T-junctions remain in the Bézier extracted form of the T-spline surface. Interested readers are directed to Borden et al. [10] for details on computing extraction operators for NURBS curves and surfaces, and Scott et at. [60] for details on computing extraction operators for T-splines.

4.1.2 Bézier Projection

As stated earlier, we wish to define some relation between a global NURBS or T-spline surface, and triangular and quadrilateral faces of the Bézier elements that lie on the global boundary of the mesh. Bézier projection, proposed by Thomas et al. [70], is a element-based local projection framework for splines that allows us to do exactly this. In Bézier extraction, we perform a **local** projection of the spline basis onto the Bernstein basis via an extraction operator, $\mathbf{C}^e$. In Bézier projection, we instead project **any** function onto a spline basis, and as a result, we can approximate any geometry as a spline object. More specifically, we are interested in projecting the global NURBS



(a)



(b)

Figure 4.1: Bézier extraction on a T-spline surface. (a) The control net for the extracted Bézier elements. (b) The Bézier elements.

or T-spline surface onto a collection of piecewise Bézier elements. In this section, we introduce Bézier projection for the univariate case by example. In Section 4.3.3, we will demonstrate how we extend this to the bivariate case for the purposes of surface reconstruction.

First let us define what we mean by Bézier projection. Let $\hat{\Omega} \subseteq \mathcal{R}^{d_r}$ denote a given domain, and let $f : \hat{\Omega} \to \mathbb{R}^{d_s}$ be a given target function to be approximated. For the case of interest, f is the parameterization of the global NURBS or T-spline surface, and $\hat{\Omega}$ is the parametric domain for the surface. We aim then to approximate f using a spline function f^h within a set of candidate spline function $\mathcal{T}$. There exist many means of doing so, including point collocation and L^2 projection. Generally speaking, we aim to define a projection operator $\Pi : (L^2(\hat{\Omega})) \to \mathcal{T}$ such that (1) Π is a linear operator, and (2) Π is spline perserving in that $\Pi f = f$ for all $f \in \mathcal{T}$. That is:

$$f^h = \Pi(f) = \sum_A \mathbf{P}_A(f) N_A$$

where $\mathbf{P}_A$ are the control points of the spline approximation and N_A are the spline basis functions $N_A \in \mathcal{T}$. We note that if $f \in \mathcal{T}$, then Bézier projection is exact.

It remains to describe precisely how the spline control points are found. With Bézier preojection, we define the spline control points via a three step process, which we demonstrate below in the univariate setting. Consider the function $f(\xi) = [\xi, \sin(\xi) + \xi], \xi \in [0, 3\pi]$ shown in Figure 4.2a. We desire to project the spline onto the B-spline curve defined over the knot vector $\Xi = \{0\ 0\ 0\ 0\ \pi\ \pi\ \pi\ 2\pi\ 2\pi\ 2\pi\ 3\pi\ 3\pi\ 3\pi\ 3\pi\}$. The process for Bézier projection is then carried out as follows:

(1) **Local projection onto the Bernstein Basis:** First, the domain $\hat{\Omega}$ is partitioned into N elements, $\hat{\Omega}^e \subseteq \hat{\Omega}$, where the domain $\hat{\Omega}^e$ corresponds to the domain of a local Bézier curve. For each subdomain, there exists a mapping $\phi^e : [0\ 1] \to \hat{\Omega}^e$ that maps the unit interval to the element e. Then, we find the local Bézier control points, $\mathbf{P}^{b,e}$ by solving the system:

$$\sum_{j=0}^{p} (B_i, B_j) \mathbf{P}_j^{b,e}(f) = (B_i, f \circ \phi^e) \quad i = \{0, ..., p\}$$

where (f, g) denotes the L_2 inner product over the unit interval, defined as:

$$(f, g) = \int_0^1 f(\xi) g(\xi) d\xi$$

Note that this yields the L^2 projection of f onto the space of Bernstein polynomials. Thus the local projection of f onto the Bézier curve is given by:

$$\Pi^e(f) = \sum_{i=0}^{p} \mathbf{P}_i^{b,e}(f) B_i$$

Figure 4.2b shows the result of local projection of f onto three Bézier curves.

(2) **Projection of local Bézier values onto local NURBS values:** Once we have determined the local Bézier control points for each element, we must project these values onto the NURBS control points. To do so, we employ the element reconstruction operator, which maps the Bernstein basis into the global spline basis for a given element. Recall that in Bézier extraction, the element extraction operator $\mathbf{C}^e$ maps the NURBS control points to the Bézier control points. It holds then that $\mathbf{R}^e$, can be defined as the inverse of the element extraction operator, $\mathbf{R}^e = (\mathbf{C}^e)^{-1}$. Then, to exactly preserve the function $\Pi^e(f)$ in terms of the spline basis, it is then sufficient to define the local spline control points as:

$$\mathbf{P}^e(f) = (\mathbf{R}^e)^T \mathbf{P}^{b,e}(f)$$

(3) **Blending of local values to find global values:** Finally, if $f \notin \mathcal{T}$, the reconstructed NURBS control points will not coincide from element to element, as is readily seen in Figure 4.2b. It remains then to somehow select a global control point from the collection of element control points. We note that there exists a wide variety of possible choices for selecting, averaging, or recombining these control points, and of the possibilities, none have been shown to be provably optimal. However, Thomas et al. propose a weighted averaging scheme based on element size, that has been demonstrated to yield good results. The global NURBS control points are found from the local NURBS control points via the relation:

$$\mathbf{P}_A(f) = \sum \omega_A^e \mathbf{P}_A^e(f)$$

where the weights, ω_A^e, are defined as:

$$\omega_A^e = \frac{\int_{\Omega^e} N_A d\Omega}{\sum_{e'=E_A} \int_{\Omega^{e'}} N_A d\Omega}$$

Figure 4.2c shows the resulting global NURBS curve approximation of f.

With our description of the Bézier projection process concluded, we take a moment to comment briefly on a few more practical considerations. First, we note that much like with Bézier extraction, when working with a rational spline basis, it is required to first transform the spline to projective space, and perform projection there. Also, as with Bézier extraction, Bézier projection extends readily to the multivariate case. Finally, we note that for the example considered here, and the applications considered in this dissertation, we desire to simply project onto the piecewise Bernstein basis. As such, the projection operation on the local Bézier control points simply has the effect of performing a direct mapping from local Bézier control points to local spline control points. Indeed, it can be verified that for the knot vector considered in this example, the element extraction operators, and therefore the element reconstruction operators, are all simply the identity matrix.

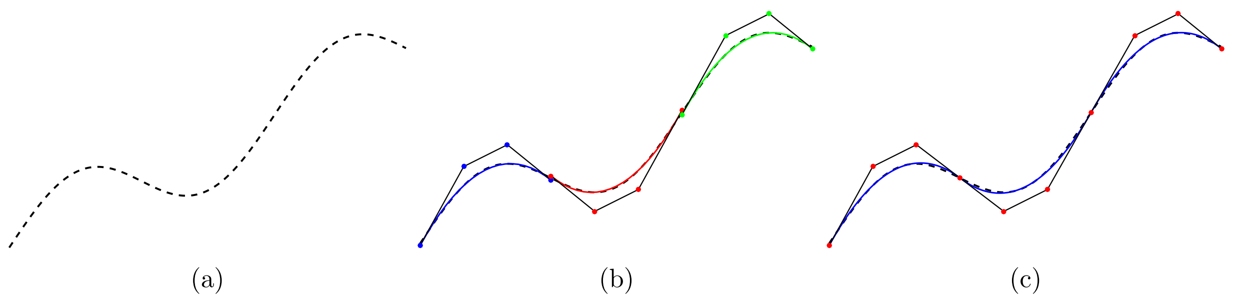


Figure 4.2: (a) The target function, f. (b) Local projection of f onto three Bézier curves. The three Bézier curves are shown in different colors along with their control nets. (c) The NURBS approximation to f.

4.2 Geometric Polynomial Complexity

Before proceeding forward to a discussion on mesh generation, we introduce one more fundamental concept. When dealing with higher-order surfaces, it is useful to define some notion of the total (rational) polynomial complexity of the surface. That is, we would like to define an upper bound on the polynomial degree needed to exactly represent the surface. Recall from Section 4.1.2 that Bézier projection is exact for a target function f if $f \in \mathcal{T}$. Thus, if we can determine the polynomial complexity of the surface, we can determine what polynomial degree elements are needed to create a geometrically exact mesh of the object. With this in mind, we introduce the following two definitions.

Definition 4.2.1. P_1 Complexity

A geometry $\Omega \subset \mathbb{R}^{d_s}$ is said to have polynomial complexity, or P_1-complexity, of degree p if the following hold:

(1) There exists a set of non-overlapping open sets, Ω^e, such that $\overline{\Omega} = \overline{\bigcup_{e=1}^{n} \Omega^e}$

(2) For each open set Ω^e, there exists a bijective polynomial or rational[1] map of degree p, $\mathbf{x}^e : \hat{\Omega}^e \to \Omega^e$, where $\hat{\Omega}^e$ is an open set $\hat{\Omega}^e \subset \mathbb{R}^{d_r}$.

(3) For each pair Ω^i and Ω^j, where $i \neq j$, and $\partial\Omega^i \cap \partial\Omega^j \neq \emptyset$, the transition mapping, $\vec{t}_{ij} : \hat{\Gamma}_{ij} \to \hat{\Gamma}_{ji}$, defined by:

- $\vec{t}_{ij} := \mathbf{x}_j^{-1} \circ \mathbf{x}_i$
- $\hat{\Gamma}_{ij} := \mathbf{x}_i^{-1}(\partial\Omega^i \cap \partial\Omega^j)$
- $\hat{\Gamma}_{ji} := \mathbf{x}_j^{-1}(\partial\Omega^i \cap \partial\Omega^j)$

is affine.

Definition 4.2.2. P_2 Complexity

A geometry $\Omega \subset \mathbb{R}^{d_s}$ is said to have polygonal polynomial complexity, or P_2-complexity, of degree p if the above hold and the open sets $\hat{\Omega}^e$ are polygons.

[1] We identify the polynomial degree of a rational map as the maximum of the degrees of the numerator and denominator.

It should be noted that our definitions for P_1 and P_2 complexity accommodate for rational maps in addition to polynomial maps. This is important as many geometric objects of engineering interest including conic sections cannot be represented using polynomial maps of finite degree. To illustrate the notion of polynomial complexity, we have included a collection of surfaces with different P_2 complexity in Figure 4.3. It remains to establish a simple computational test to determine the P_2 complexity of a surface. As it turns out, such a test is not readily available. However, recall that for any given NURBS or T-Spline surface, we can define the surface as a collection of Bézier quadrilaterals obtained through Bézier extraction. Each Bézier quadrilateral is represented in terms of a parameterization $\mathbf{x}^{quad} : \hat{\Omega}^{quad} \rightarrow \mathbb{R}^3$. While we cannot directly measure the geometric complexity of the **surface**, we can in fact design a test to measure the complexity of the **parameterization**. To design such a test, we first consider the transformation from the bivariate Bernstein basis to the power basis, which is defined as the complete set of polynomials of degree p. For example, the bi-cubic power basis is defined as:

$$P = [1,\ \xi_1,\ \xi_2,\ \xi_1^2,\ \xi_2^2,\ \xi_1\xi_2,\ \xi_1^3,\ \xi_2^3,\ \xi_1^2\xi_2,\ \xi_1\xi_2^2,\ \xi_1^3\xi_2,\ \xi_1\xi_2^3,\ \xi_1^2\xi_2^2,\ \xi_1^3\xi_2^2,\ \xi_1^2\xi_2^3,\ \xi_1^3\xi_2^3]$$

We then define the transformation matrix $\mathbf{T}$, which defines the transformation from the power basis to the Bernstein basis through the relation:

$$B_i = \sum_j \mathbf{T}_{ij} P_j$$

This allows us to define the set of homogeneous power basis coefficients $\{\widetilde{\mathbf{P}}_i^p\}_{i=1}^n$ from the homogeneous Bézier control points $\{\widetilde{\mathbf{P}}_i^b\}_{i=1}^n$, via the relation:

$$\widetilde{\mathbf{P}}^p = (\mathbf{T})^T \widetilde{\mathbf{P}}^b \tag{4.2}$$

The power basis coefficients $\{\mathbf{P}_i^p\}_{i=1}^n$, and their corresponding weights $\{w_i^p\}_{i=1}^n$ can be found as:

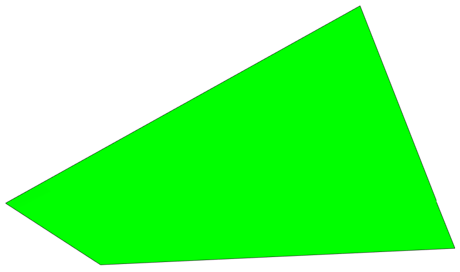
$$\begin{aligned} (\mathbf{P}_i^p)_j &= (\widetilde{\mathbf{P}}_i^p)_j / (\widetilde{\mathbf{P}}_i^p)_{d_s+1} \qquad i \in \{1, ..., n\},\ j \in \{1, ..., d_s\} \\ w_i^p &= (\widetilde{\mathbf{P}}_i^p)_{d_s+1} \end{aligned} \tag{4.3}$$

It can be verified that the Bézier surface can then be defined in terms of the power basis and corresponding coefficients in much the same way as it can using the Bernstein basis and Bézier control points. That is:

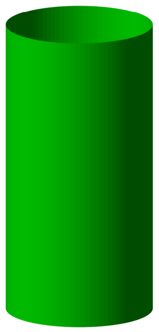
$$\mathbf{x}^{equad}(\boldsymbol{\xi}) = \frac{\sum\limits_{i} P_i(\boldsymbol{\xi})\mathbf{P}_i^p}{\sum\limits_{j} P_j(\boldsymbol{\xi})w_j^p} \tag{4.4}$$

Of course, the power basis coefficients give very little intuition into the shape of the surface, which is why they are not used in the geometric modeling context. However these coefficients do give insight into the polynomial degree of the parameterization. It can be readily seen from Equation (4.4) that any polynomial degree that is not needed to parameterize the surface will have a corresponding power basis coefficient $\mathbf{P}_i^p = [0\ 0\ 0]$. In this way, we establish a simple, computationally efficient test for the P_2 complexity of a parameterization. The transformation matrix $\mathbf{T}$ is defined only as a function of the two bases, and needs only be computed once. Then, for each Bézier element, the power basis coefficients, and therefore the P_2 complexity of the parameterization, can be found via Equation (4.2).

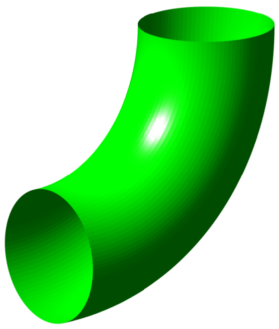
Finally, we must clarify a somewhat subtle aspect of P_2 complexity, namely the difference between the P_2 complexity of the **geometry**, and the P_2 complexity of the **parameterization**. The test for P_2 complexity outlined in Equation (4.2) will yield the P_2 complexity of the parameterization, which is not necessarily equal to the P_2 complexity of the geometry. For example, consider that it is possible to parametrize a circle by a collection of C^0-continuous quadratic Bézier curves, or by a single C^1-continuous quartic rational Bézier curve [16]. In both cases, the P_2 complexity of the geometry is $P_2 = 2$. However, the P_2 complexity of the C^0-continuous quadratic parameterization is $P_2 = 2$, while the polynomial complexity of the C^1-continuous quartic parameterization is $P_2 = 4$. It is useful to note that for any given geometry, the P_2 complexity of the parameterization serves as an upper bound for the P_2 complexity of the geometry. While it is beyond the scope of this work, it could be both interesting and useful in future work to develop a direct test of the P_2 complexity of the **geometry**, as well as developing an algorithm to re-parameterize the geometry to the lowest complexity parameterization.



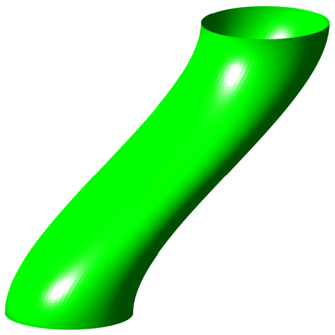
(a)



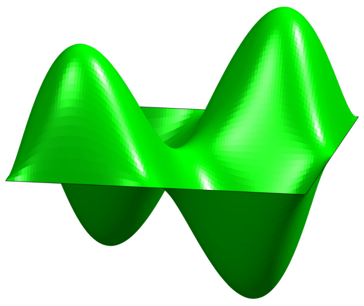
(b)



(c)



(d)



(e)

Figure 4.3: A variety of surfaces with different P_2 complexity: (a) Bilinear surface, $P_2 = 2$, (b) Cylinder, $P_2 = 3$, (c) Pipe Elbow, $P_2 = 4$, (d) S-Duct, $P_2 = 5$, (e) Bi-cubic surface, $P_2 = 6$.

4.3 Mesh Generation Procedure

With the relevant background introduced, we now turn our attention to the main focus of this work: construction of higher-order, geometrically exact meshes. Generally speaking the procedure is broken into four steps:

Step 1: *Generation of a compatible linear mesh.*

Step 2: *Degree elevation of the linear mesh.*

Step 3: *Surface reconstruction.*

Step 4: *Smoothing of control points and weights.*

The rest of this section will provide detailed descriptions of each of the steps listed above.

4.3.1 Generation of a Compatible Linear Mesh

The focus of this work is not on linear mesh generation technologies, and we do not propose any new methods for mesh generation in this dissertation. Instead, we rely heavily on existing mesh generation technologies, and this section serves to outline our requirements for the initial linear mesh. In traditional linear mesh generation, the meshing process begins by first creating a polygonal surface mesh, and a polyhedral volumetric mesh is created based off of this surface mesh. Since CAD geometries are encoded by surfaces, we are specifically interested in ensuring that we generate suitable linear surface meshes such that we are able to reconstruct the geometry in a later step. To ensure that this is possible, we impose several constraints on both the CAD surface and resulting linear surface mesh.

First, let the CAD surface be comprised of any number of non-intersecting, orientable, closed manifolds without boundary. For each manifold, $\mathcal{S}$, we require that the manifold be explicitly parameterized by a watertight[2] NURBS or T-spline surface. Thus for each manifold, we can define a set of Bézier elements Ω^e through Bézier extraction such that:

$$\overline{\mathcal{S}} = \overline{\bigcup_{e=1}^{n} \Omega^e}$$

[2] By a water tight, we mean that the surface is free from gaps and self-intersections.

and for each Bézier element there exists a rational Bernstein-Bézier mapping $\mathbf{x}^e : \hat{\Omega}^{quad} \to \Omega^e$. Thus the extracted elements provide an explicit, bijective, watertight parameterization of $\mathcal{S}$.

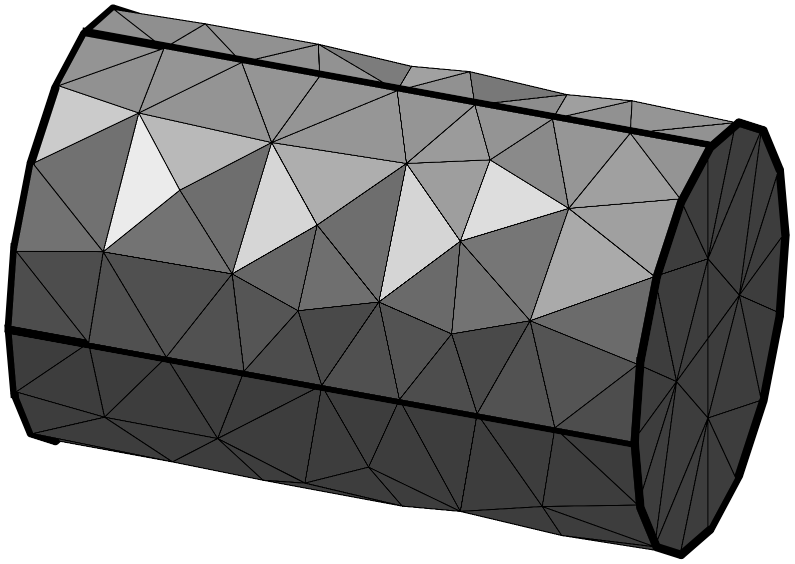
Once we have ensured that the CAD surface is valid, we must generate a suitable linear mesh of the surface. We call such a surface mesh **compatible**, and the requirements for a compatible mesh are roughly stated as follows:

Requirement 1: *Each vertex in the mesh is a point on the surface $\mathcal{S}$.*

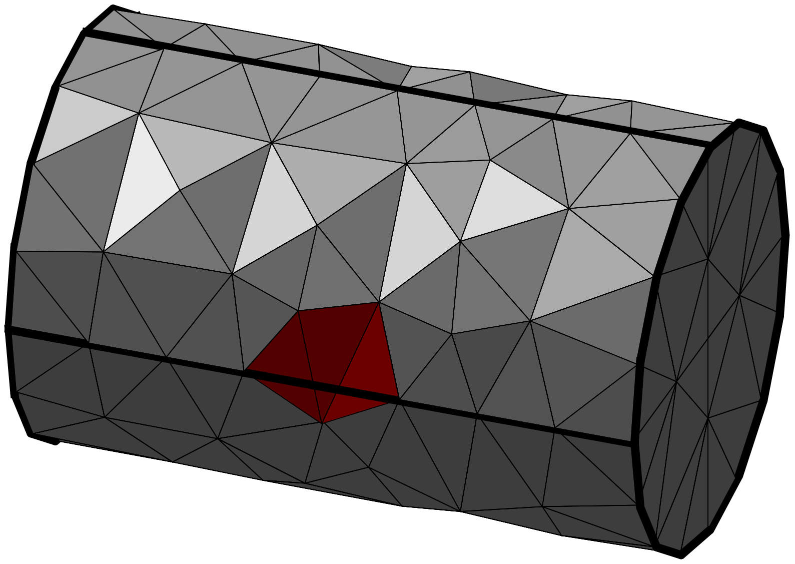
Requirement 2: *Each polygon in the mesh belongs to a unique Bézier element.*

To make these requirements precise, we denote a linear surface mesh as $\mathcal{M} = \{\mathcal{V}, \mathcal{P}\}$, where $\mathcal{V}$ is the set of vertices in the mesh, and $\mathcal{P}$ is a polygonization of the vertices in $\mathcal{V}$. Requirement #1, mathematically speaking, requires that $V \in \mathcal{S}$ for every vertex $V \in \mathcal{V}$. Requirement #2, on the other hand, requires that each vertex for a given polygon $p_k^e \in \mathcal{P}$ must lie on a unique Bézier element $\Omega^e \subset \mathcal{S}$. To illustrate this second requirement, Figure 4.4a shows an example of a compatible mesh, while Figure 4.4b shows an incompatible mesh, with the incompatible polygons highlighted in red. The motivation for this second requirement will become clear in Section 4.3.3, as it ensures that we will be able to perform Bézier projection in order to recover the exact geometry. Again, we note that there exists a wide variety of well established linear mesh generation algorithms [15, 28, 38]. Any one of these algorithms may be used produce the preliminary linear mesh, providing the above conditions are met.

We must mention that there is one caveat on the precise requirements on the CAD surfaces that we have listed above. Consider the cylindrical surface shown in Figure 4.5a. As it is, this surface does not satisfy our criteria because it is open at the ends, and is therefore not a manifold without boundary. However, for practical implementation, we permit the closure of open surfaces using a trimmed surface, **provided that the trimmed surface is planar.** Figure 4.5b shows the cylinder closed by a trimmed planar surface. The reasoning behind this caveat comes directly from the surface reconstruction constraints. If a surface is planar, we are not required to perform geometry reconstruction, and therefore do not require that the surface be explicitly parameterized



(a)



(b)

Figure 4.4: a) A valid and (b) an invalid linear mesh. Bézier element boundaries are shown in bold, and invalid triangles are shown in red.

by Bézier elements. With a suitable surface mesh generated, it then remains to generate a linear mesh of the volume. Figure 4.6 shows a cut view of the volume mesh of the cylinder from Figure 4.5c. The elements intersected by the cutting plane are highlighted in yellow.

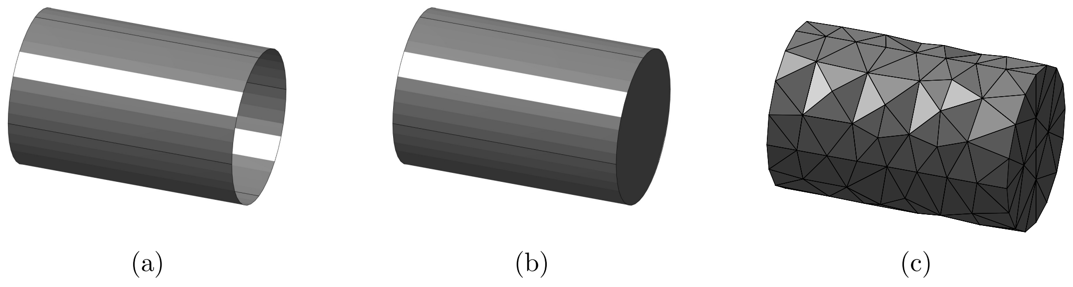


Figure 4.5: Triangular surface mesh generation. (a) The initial open cylindrical surface. (b) Closing the surface with a trimmed plane. (c) Generation of an unstructured surface mesh.

4.3.2 Degree Elevation of the Linear Mesh

Once we have generated a suitable linear mesh, we next degree elevate the mesh by locally degree elevating the linear elements to create the control nets for our higher-order Bézier elements. That is, we would like to generate some higher-order Bernstein–Bézier representation of the elements in the linear mesh. To achieve this, let us first denote the set of n_{el} polyhedrons in the linear mesh as $\{\Omega^{poly,E}\}_{E=1}^{n_{el}}$. We then define a mapping $\mathbf{x}^{poly,E} : \hat{\Omega} \to \Omega^{poly,E}$, that maps the reference polyhedron $\hat{\Omega}$ to the physical polyhedron $\Omega^{poly,E}$ for each polyhedron in the mesh. Let us also define the equispaced control net over the reference polyhedron $\{\boldsymbol{\xi}_i\}_{i\in I} \in \hat{\Omega}$. Examples of such reference control nets are visualized for the cubic geometric primitives in the figures presented in Chapter 3. Then, for each polyhedron in the linear mesh, we can define the higher order Bézier control net via the mapping:

$$\mathbf{P}_i^{b,E} = \mathbf{x}^{poly,E}(\boldsymbol{\xi}_i)$$

We note that the mapping $\mathbf{x}^{poly,E}$ is affine for tetrahedra, bilinear for wedges, and trilinear for hexahedra and pyramids. Finally, during this step, we additionally set all of the control point weights to unity. The result of this process is shown in Figure 4.7, which displays the degree

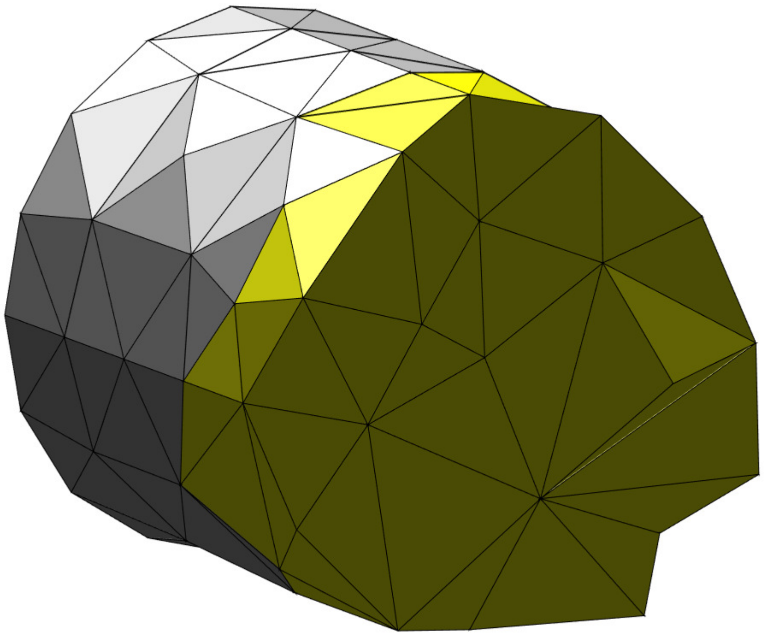


Figure 4.6: Cut view of the linear cylinder mesh.

elevated linear mesh of the cylinder from Figure 4.6 to a cubic tetrahedral mesh. With this step complete, we have obtained a higher-order mesh, but it remains to encode the geometry from the surface in the mesh. This is achieved through surface reconstruction procedure described in the next section.

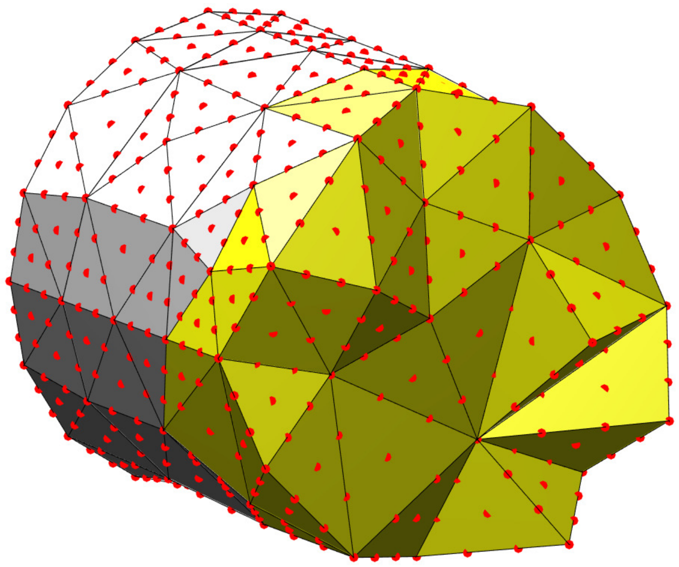


Figure 4.7: Degree elevated cylindrical mesh.

4.3.3 Surface Reconstruction

The next step is to update the newly generated control nets so that the mesh surface exactly matches the CAD surface. Recall from Section 4.1.2 that Bézier projection allows us to project any function onto a NURBS basis. In the context of surface reconstruction, we desire to project the Bézier elements from our CAD surface onto a collection of Bézier quadrilaterals and triangles. However, before we describe the exact process for performing surface reconstruction, we first introduce some useful notation. Figure 4.8 serves to illustrate the notation introduced here.

Recall from Section 4.3.1 that each polygon in the surface mesh belongs to a unique Bézier element. As a result, we can associate a unique set of polygons, which we denote as $\{p_k^e\}_{k=1}^n$, with each Bézier element Ω^e. Then, for a given polygon p_k^e, we define $\boldsymbol{\psi}_k^e : \hat{\Omega}^k \rightarrow p_k^e$ to be the unique mapping such that:

$$\boldsymbol{\psi}_k^e((\mathbf{x}^e)^{-1}(V)) = V$$

for every vertex V of the polygon p_k^e. This mapping is affine for triangles and bilinear for quadrilaterals. Next, let us define the corresponding parametric polygon $\hat{p}_k^e$ as:

$$\hat{p}_k^e = (\boldsymbol{\psi}_k^e)^{-1}(p_k^e)$$

Note that the set of parametric polygons $\{\hat{p}_k^e\}_{k=1}^n$ forms a non-overlapping cover of $\hat{\Omega}^{quad}$, meaning:

$$\overline{\hat{\Omega}^{quad}} = \overline{\cup \overline{\hat{p}_k^e}}$$

$$\cap \hat{p}_k^e = \emptyset$$

Consequently, $\{\hat{p}_k^e\}_{k=1}^n$ forms a watertight polygonization of the reference quadrilateral.

Now, let us introduce a few more mappings. Namely, let $\boldsymbol{\phi}_k^e : \hat{\Omega}^k \rightarrow \hat{p}_k^e$ be the unique mapping from the reference polygon $\hat{\Omega}^k$ to the parametric polygon $\hat{p}_k^e$. As with $\boldsymbol{\psi}_k^e$, this mapping is affine for triangles and bilinear for quadrilaterals. With $\boldsymbol{\phi}_k^e$ defined, we can define the composite mapping:

$$\mathbf{x}_k^e := \mathbf{x}^e \circ \boldsymbol{\phi}_k^e$$

where $\mathbf{x}_k^e$ is a bijective mapping between $\hat{\Omega}^k$ and the physical entity $\mathbf{x}^e(\hat{p}_k^e) = \Omega_k^e \subseteq \Omega^e$. The set of physical entities $\{\Omega_k^e\}_{k=1}^n$ form a non-overlapping cover of Ω^e, and thus our goal is to construct a Bernstein–Bézier representation of each mapping $\mathbf{x}_k^e : \hat{\Omega}^k \rightarrow \Omega_k^e$. That is, for each entity Ω_k^e, we would like to find a representation of the form $\Omega_k^e = \sum\limits_{i \in I} R_i(\boldsymbol{\xi})\mathbf{P}_i^{b,k}$, where $\{\mathbf{P}_i^{b,k}\}_{i \in I}$ are the set of Bézier control points for Ω_k^e.

Thus, it remains to provide a method to find the Bézier control points for the surface elements. Let us denote the control points of the Bézier element from the CAD surface as $\mathbf{P}_j^{b,e}$. Let us also denote the Bernstein basis functions defined over the unit reference polygon as $\{B_i^k\}_{i \in I}$ and the basis functions defined over the unit reference quadrilateral as $\{B_i^{quad}\}_{i \in I}$.Then, following Bézier projection, we can then find the control points $\{\mathbf{P}_i^{b,k}\}_{i \in I}$ by the relation:

$$\mathbf{P}^{b,k} = \mathbf{M}^{-1}\mathbf{T}\mathbf{P}^{b,e} \tag{4.5}$$

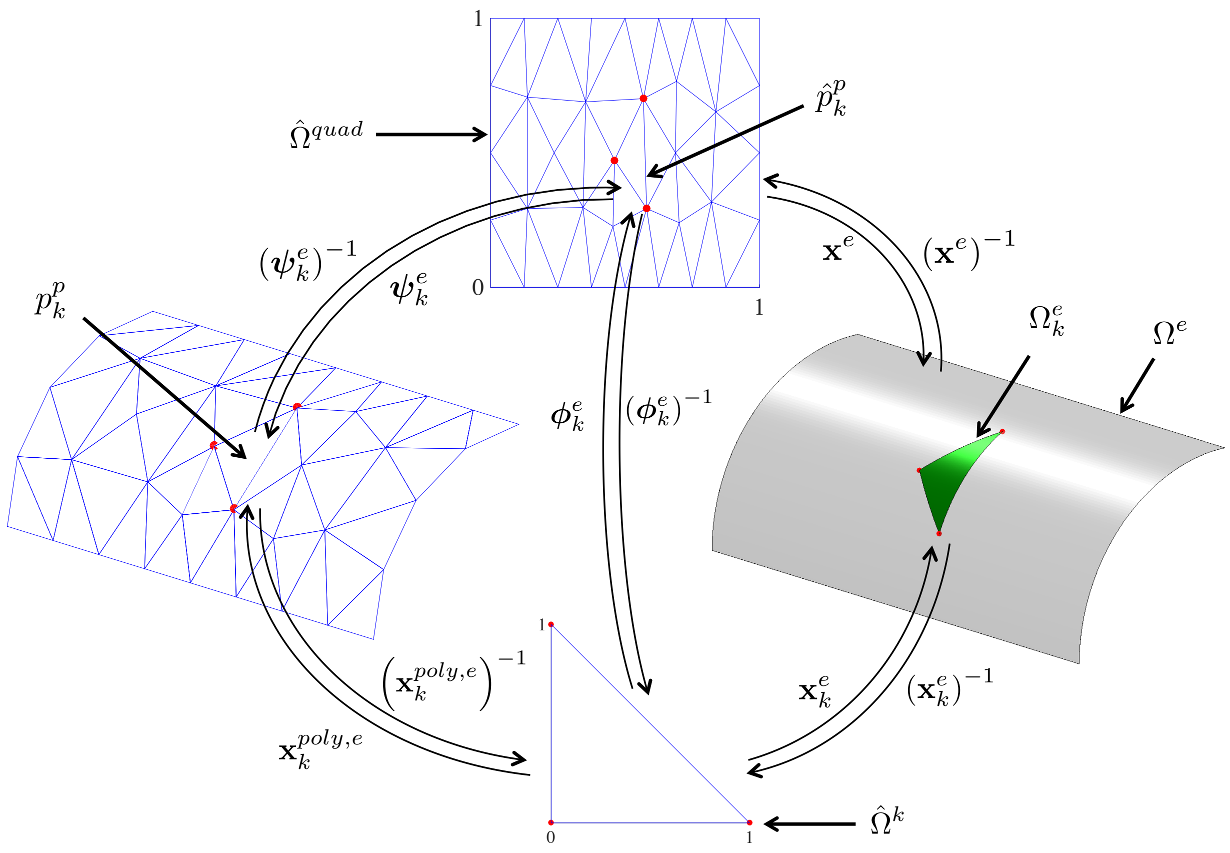


Figure 4.8: Visualization of Bézier projection of a Bézier element onto a collection of Bézier polygons. This illustrates the notation used in the surface reconstruction step presented in Section 4.3.3.

where $\mathbf{M}_{ij}$ is defined as:

$$\mathbf{M}_{ij} = \int_{\hat{\Omega}^k} B_i B_j d\hat{\Omega}^k$$

and $\mathbf{T}_{ij}$ is defined as:

$$\mathbf{T}_{ij} = \int_{\hat{\Omega}^k} B_i (B_j^{quad} \circ \boldsymbol{\phi}_k^e) d\hat{\Omega}^k$$

In the case when the Bézier element Ω^e is rational, we proceed similarly, but enforce Equation (4.5) on the homogenous control points, viz.:

$$\widetilde{\mathbf{P}}^{b,k} = \mathbf{M}^{-1}\mathbf{T}\widetilde{\mathbf{P}}^{b,e}$$

Once we have successfully reconstructed the surface, we simply replace the control points that lie on the surface of the degree elevated linear mesh with the control points obtained through surface reconstruction. We likewise update the corresponding control point weights with the weights obtained through surface reconstruction. Figure 4.9 shows the mesh of the cylinder displayed in Figure 4.7 after surface replacement.

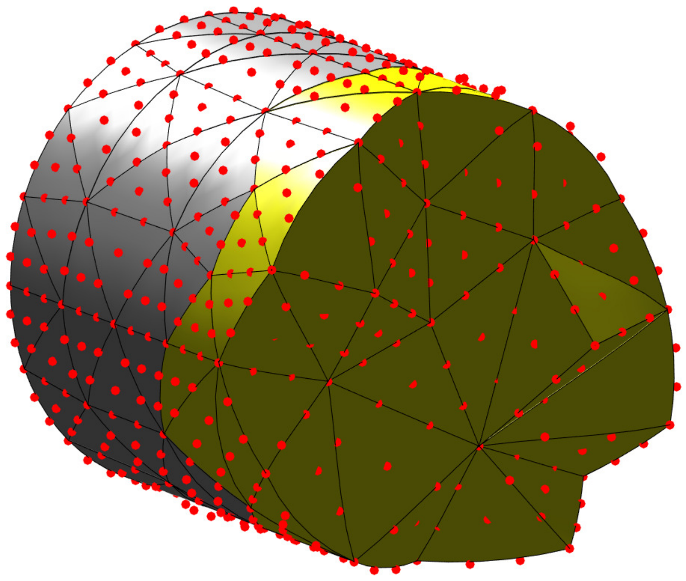


Figure 4.9: Surface reconstructed cylindrical mesh.

4.3.4 Smoothing of Control Points and Weights

At this point we have succeeded in our original goal of performing isogeometric degree elevation of our mesh. However, we note that the process of surface reconstruction can often lead to large distortion of certain elements. Chapter 5 provides a more detailed discussion of mesh quality metrics, but for now we present a method for smoothing the internal control point locations and weights in the reconstructed mesh. We have found that in many cases, this final smoothing step can help to improve mesh quality, so we present it here as a useful tool in the meshing process.

When the control point weights are not all identically unity, we begin by solving a Laplace smoothing problem to smooth the control point weights on the interior of the mesh. Let Ω denote the degree-elevated, non-reconstructed mesh, and let w_s denote the weighting function associated with the surface obtained during the surface reconstruction process. Then, let $\mathcal{V}^h$ be the set of candidate weighting functions that satisfy $w^h|_\Gamma = w_s$. We can then smooth the interior control point weights w^h by solving the minimization problem:

$$\min_{w^h \in \mathcal{V}^h} \frac{1}{2} \int\limits_\Omega |\nabla w^h|^2 d\Omega$$

We can similarly smooth the internal control point locations by solving a linear elasticity problem. Again, let Ω denote the degree-elevated, non-reconstructed mesh, and let $\vec{d_s}$, denote the displacement of the surface nodes under the surface reconstruction process. We define $\mathcal{V}^h$ to be the set of candidate displacement functions that satisfy $\vec{d^h}|_\Gamma = \vec{d_s}$. Note that $\mathcal{V}^h$ is a set of rational functions, with respect to the control point weights obtained in the weight smoothing step. We can then smooth the interior control point locations by solving the following linear elasticity problem for the control point displacement, $\vec{d^h}$:

$$\min_{\vec{d^h} \in \vec{\mathcal{V}}^h} \frac{1}{2} \int\limits_\Omega \boldsymbol{\epsilon}(\vec{d^h}) \mathbf{D} \boldsymbol{\epsilon}(\vec{d^h}) d\Omega$$

where $\mathbf{D}$ and $\boldsymbol{\epsilon}$ are the elasticity tensor and strain vector, respectively.

4.3.5 Applications to Higher-Order FEA

We take this opportunity to discuss the nature of exact geometry. Recall from section 4.1.2 that Bézier projection is exact if and only if $\mathbf{x}_k^e \in \mathcal{T}$. Thus in order to guarantee that our surface reconstruction process is exact, we must ensure that the polynomial degree of our mesh is at least as large as the P_2 complexity of the parameterization given by $\mathbf{x}^e$. That is, if our parameterization has a P_2 complexity of $P_2 = 6$, we require sextic triangles on the surface to exactly match the geometry. Additionally, generally speaking, the mapping $\boldsymbol{\phi}_k^e$ for a quadrilateral is bilinear. As a result the composition of the mappings $\mathbf{x}_k^e = \boldsymbol{\phi}_k^e \circ \mathbf{x}^e$ has twice the polynomial degree of the mapping $\mathbf{x}^e$. Thus we would require dodecic quadrilaterals to exactly match the geometry!

It would appear at this point that we have reached an impasse, as dodecic, or even sextic elements are often impractical for simulation applications. However, we briefly suggest two alternatives. First, we recall that even in the case that Bézier projection is not exact, it will still yield a curve or surface of best fit. Thus, it is still possible to use Bézier projection to perform surface reconstruction as we have presented it in Section 4.3.3. While this surface reconstruction might not be truly isogeometric, it does provide a best fit to the CAD surface, and it is certainly more accurate than the linear mesh. Additionally, despite not being exact, we still recover an unstructured Bernstein–Bézier representation of the volume. We can then use these lower-order meshes to perform an isoparametric FEA simulation on the geometry of interest. Another possible method for reducing the computational complexity is to use a superparametric representation as is sometimes employed in traditional FEA. In a superparametric approach, the geometry is represented by a basis that is of a higher polynomial degree than the basis used to represent the simulation variables. For example, one could employ a discretization of sextic Bernstein-Bézier elements to represent the geometry but use cubic Bernstein polynomials as the basis for FEA.

There are varying advantages and disadvantages of the approaches listed above. Lower order mesh elevation can be computationally less intensive, but suffers from one of the main drawbacks of traditional mesh generation: the initial mesh is still just an approximation, and does not encode the exact geometry. As a result, the geometric approximation error is “set” at the coarsest level of mesh generation. Under mesh refinement, the geometric accuracy cannot be increased without querying the CAD model. The superparametric approach on the other hand does not suffer from this drawback. Because the initial mesh is exact, it can be refined without subsequent queries to CAD. Additionally, a superparametric simulation is less computationally intensive than a corresponding isogeometric simulation but is more computationally intensive than a lower-order isoparametric simulation. In either case, it is most important to note that one of the main advantages to our approach is that it allows for a great deal of flexibility in both mesh generation and simulation. We demonstrate this in Section 4.6.3 by providing numerical examples of purely isogeometric simulations, as well as superparametric and lower-order isoparametric simulations.

4.3.6 Construction of Structured Surface Meshes

Before concluding this section, we make a quick aside regarding structured surface meshes. While surface meshes of this type form a very restricted subset of all surface meshes, they arise frequently when considering applications such as boundary layer meshing or hex-dominant meshing. As such, it is useful to briefly make special mention of this subset of meshes. Consider the surface mesh of the cylinder shown in Figure 4.10. The cylindrical surface shown in Figure 4.10a is refined via knot insertion, resulting in the surface shown in Figure 4.10b. The surface can then be degree reduced to form a surface mesh comprised of linear quadrilaterals, as shown in Figure 4.10c.

When the surface mesh over a Bézier element is derived in this manner, the surface reconstruction step is much simpler. Since the linear mesh is derived from a refined surface, for each quadrilateral on the surface, there exists an extraction operator $\mathbf{C}^k$ such that:

$$\widetilde{\mathbf{P}}^{b,k} = (\mathbf{C}^k)^T \widetilde{\mathbf{P}}^{b,e}$$

It is therefore a simple task to find the reconstructed surface control points for each element in the structured surface mesh, and reconstruct the original surface. We note that in the case of extraction based structured surface meshes, the reconstructed surface is guaranteed to be geometrically exact.

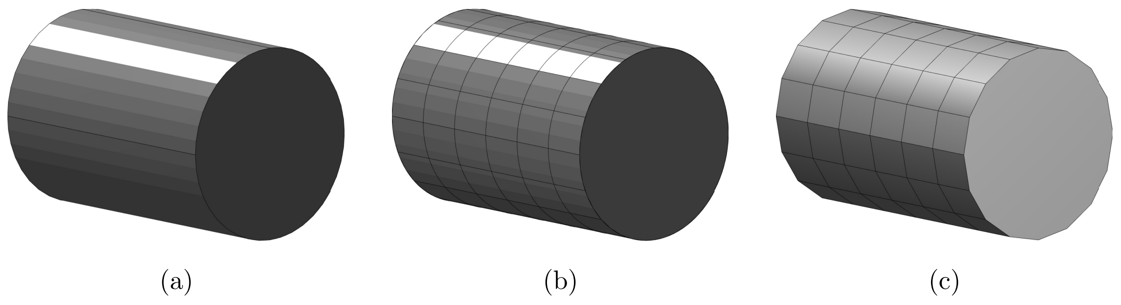


Figure 4.10: Structured surface mesh generation. (a) Start with a closed surface. (b) Refine the Bézier surface using Bézier extraction. (c) Degree reduce to arrive at a structured quadrilateral surface mesh.

This gives us some confidence in the ability to generate geometrically exact meshes without the use of dodecic elements.

This highlights one of the key advantages of Bézier pyramids: they allow us to transition from a structured surface mesh to an unstructured volume mesh, and preserve the exact geometry in the process. However, while mixed element mesh generation may seem like a panacea, offering both geometric exactness and the benefits of unstructured meshing at relatively low polynomial degree, we note that there are some distinct drawbacks to this approach. First, mixed element mesh generation is still a relatively new field, and it will be some time before it reaches the maturity and robustness of tetrahedral mesh generation. Second, refinement, and especially anisotropic refinement, of mixed element meshes is a challenging task and an ongoing topic of research interest. Largely due to these reasons, the use of all-tetrahedral meshes is still much more popular than mixed-element meshes in a number of applications, including computational fluid dynamics. As such, we have presented both tetrahedral and mixed-element approaches in this dissertation in hopes to provide a flexible framework capable of handling a wide array of simulation challenges.

Finally, much like Bézier extraction is a special case of Bézier projection, surface reconstruction of structured meshes is simply a special case of the more general unstructured reconstruction procedure. To wit, we will obtain exactly the same results if we use the Bézeir projection method in lieu of the Bézier extraction method to reconstruct a structured surface mesh. However, if we know *a priori* that we are working with a structured surface mesh, it is desirable to reconstruct the surface using the extraction method, as it is simpler.

4.4 Element/Mesh Refinement

Before considering some numerical examples, it remains to briefly touch upon the topic of mesh refinement. Indeed, mesh refinement is a topic as rich and complex as mesh generation itself. There exist a broad array of algorithms for performing both local and global refinement, as well as more advanced refinement techniques, such as adaptive mesh refinement for applications such as shock capturing. We note that it is always possible to simply refine the linear mesh and degree elevate the resulting refined linear mesh, provided that the refined linear mesh meets the requirements outlined in Section 4.3.1. However, while this approach eases implementation, it can be costly, as we must repeat the degree elevation process each time we refine. Additionally, this method does not guarantee that the parameterization is exactly preserved, which is often undesirable. In light of this, we present in this section a detailed method for performing refinement via uniform subdivision of the coarse mesh. However, we are secure in the knowledge that we are not restricted to only this type of refinement.

Consider a parent unit polyhedron in parametric space $\hat{\Omega} \subset \mathbb{R}^{d_r}$, and the corresponding Bézier polyhedra in physical space defined by m Bézier control points $\mathbf{P}_i^b \subset \mathbb{R}^{d_s} \quad i = 1, ..., m$. Then, for each subdivision polyhedra defined in parametric space $\hat{\Omega}^k \subset \hat{\Omega}$, we seek a corresponding control net in physical space defined by n Bézier control points, $\mathbf{P}_j^{b,k} \subset \mathbb{R}^{d_s} \quad j = 1, ..., n$. To compute these control points, we first define two sets of collocation points. Let $\{\boldsymbol{\xi}_l\}_{l=1}^n \in \hat{\Omega}$ denote the set of collocation points in the reference polyhedra, and let $\{\boldsymbol{\xi}_l^k\}_{l=1}^n \in \hat{\Omega}^k$ denote the corresponding set of collocation points on the k^{th} subdivision polyhedron. Then, we can find the control points $\mathbf{P}_j^{b,k}$ by solving the system of equations:

$$\sum_{j=1}^{n} \mathbf{P}_j^{b,k} B_j(\boldsymbol{\xi}_l^k) = \sum_{i=1}^{n} \mathbf{P}_i^b B_i(\boldsymbol{\xi}_l) \qquad l = 1, ..., n$$

where B_j and B_j^k are the Bernstein basis functions defined over the parent unit polyhedra and subdivision polyhedra, respectively. Rearranging, we can define a more explicit relationship of the form:

$$\mathbf{P}^{b,k} = \left(\mathbf{M}^k\right)^{-1}\mathbf{S}\mathbf{P}^b \tag{4.6}$$

where $\mathbf{M}_{ij}^k = B_j(\boldsymbol{\xi}_i^k)$ and $\mathbf{S}_{ij} = B_j(\boldsymbol{\xi}_i)$. The astute reader will recognize that with this operation, we are simply projecting the parent polyhedra onto the subdivision polyhedra.

When dealing with a rational Bézier polyhedra, we simply evaluate Equation (4.6) on the homogenous control points, viz.:

$$\widetilde{\mathbf{P}}^{b,k} = \left(\mathbf{M}^k\right)^{-1}\mathbf{S}\widetilde{\mathbf{P}}^b$$

For brevity and ease of implementation, we can define a projection matrix for each subdivision element of the form $\mathbf{T}^k = (\mathbf{M}^k)^{-1}\mathbf{S}$. Thus, to perform uniform subdivision on any given Bézier element, we must simply define the projection operators. This in turn simply requires defining collocation points at which to evaluate $\mathbf{M}_{ij}^k$ and $\mathbf{S}_{ij}$. Figures 4.11 through 4.14 shows the collocation points for calculating $\mathbf{S}_{ij}$ and $\mathbf{M}_{ij}^k$, for the four geometric primitives discussed in this dissertation and for the case of cubic Bernstein-Bézier discetizations. Note that the choice of the collocation points is somewhat arbitrary, but the selection of equispaced control points as shown in Figures 4.11–4.14 is both simple and yields suitable results. Finally, it is worth recognizing that with this approach, we are not required to subdivide a Bézier element into like elements. For example, uniform subdivision of a pyramid will result in 6 subdivision pyramids, and 4 subdivision tetrahedra. Figure 4.15 shows the result of performing uniform subdivision on some representative elements.

Finally, we make one last observation on our refinement process. Consider the uniform subdivisions on hexahedra, tetrahedra, and wedges as presented above. The subdivisions of the parent elements will result in a collection of child subdivision elements. Also note that the faces of the parent elements are subdivided into a number of child subdivision faces. The key observation is this: subdivision of the parent mesh element yields the same subdivision on the face as would result from a subdivision of that face in isolation. The key consequence of this is that the coincident faces of two neighboring elements will remain coincident after subdivision. This means that we can **locally** subdivide each element in a mesh, and the resulting mesh will be valid.

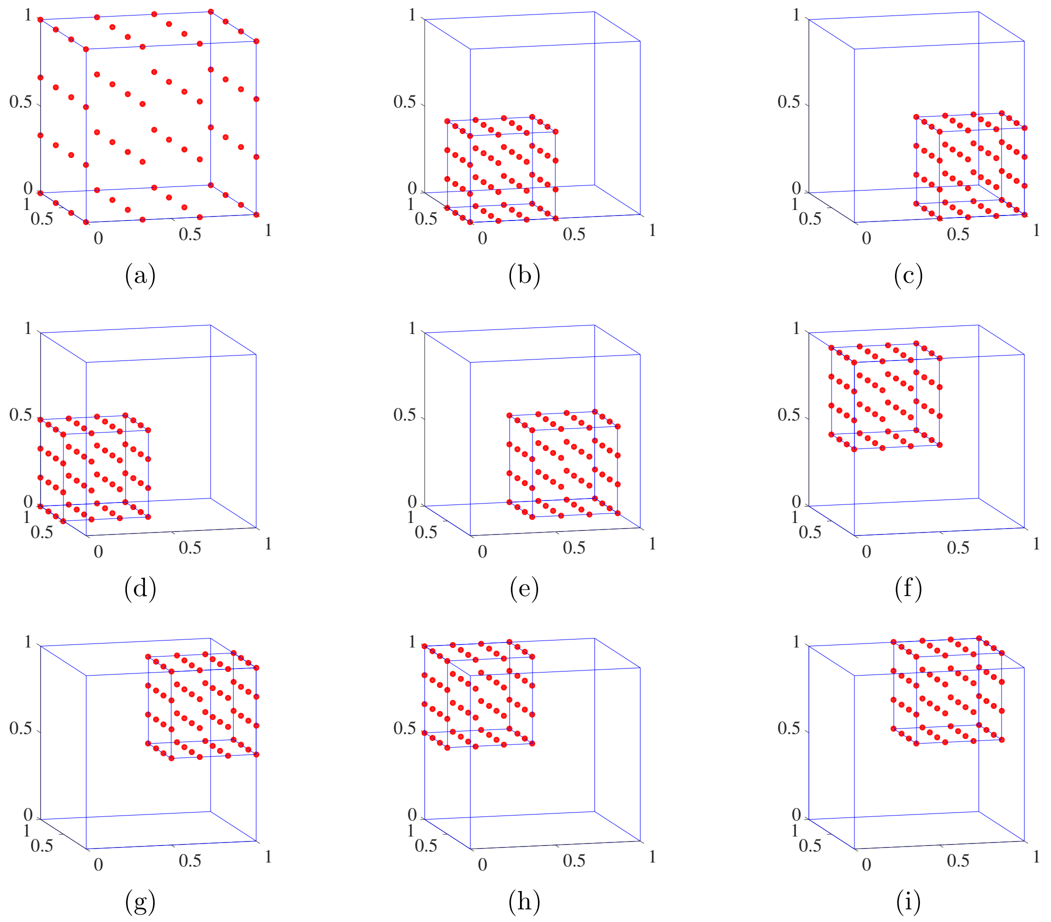
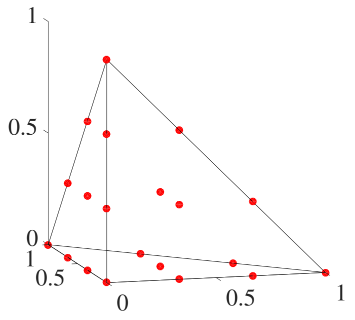
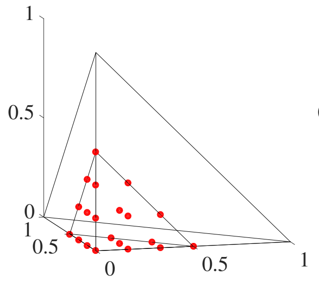


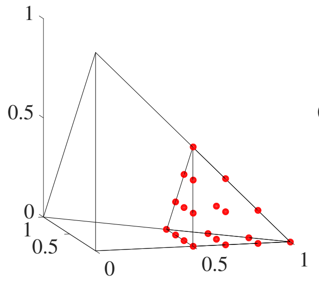
Figure 4.11: Refinement of a tricubic Bézier hexahedron. (a) Collocation points for calculating $\mathbf{S}_{ij}$. (b-i) Collocation points for calculating $\mathbf{T}_{ij}^{k}$.



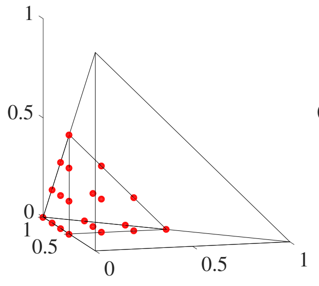
(a)



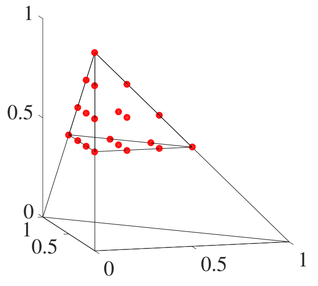
(b)



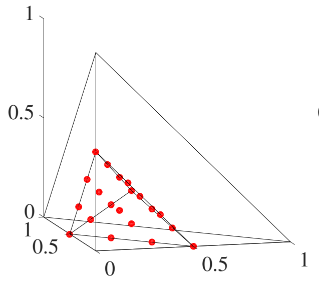
(c)



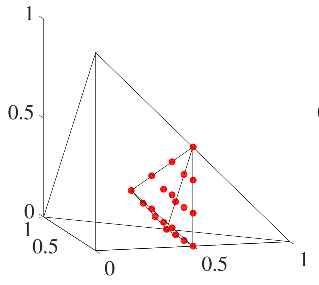
(d)



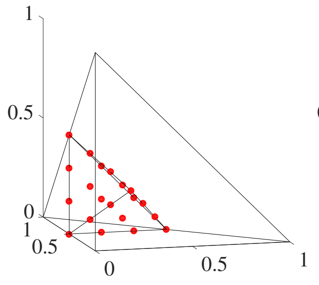
(e)



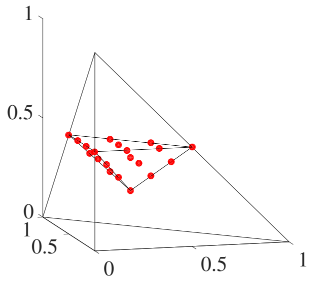
(f)



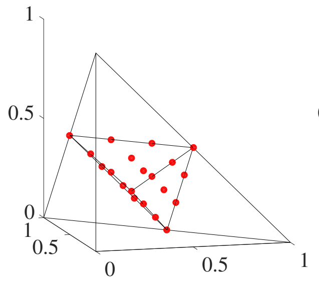
(g)



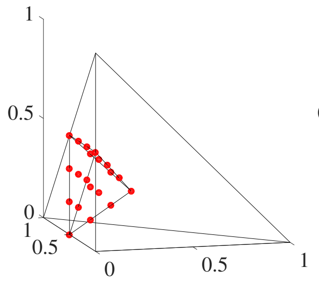
(h)



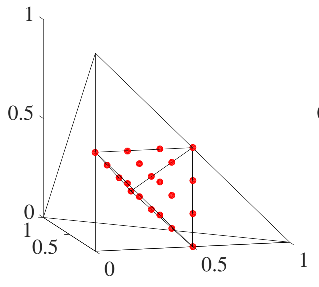
(i)



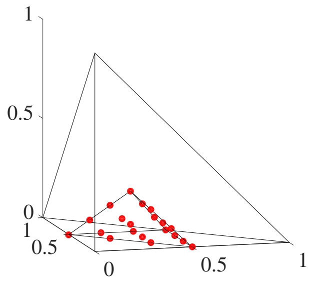
(j)



(k)



(l)



(m)

Figure 4.12: Refinement of a cubic Bézier tetrahedron. (a) Collocation points for calculating $\mathbf{S}_{ij}$. (b-m) Collocation points for calculating $\mathbf{T}_{ij}^{k}$. We note that uniform subdivision of a tetrahedron results in 4 tetrahedra and an a octahedron in the center. We choose to split this octahedron into 8 tetrahedra as shown above, but other methods of splitting are readily implemented.

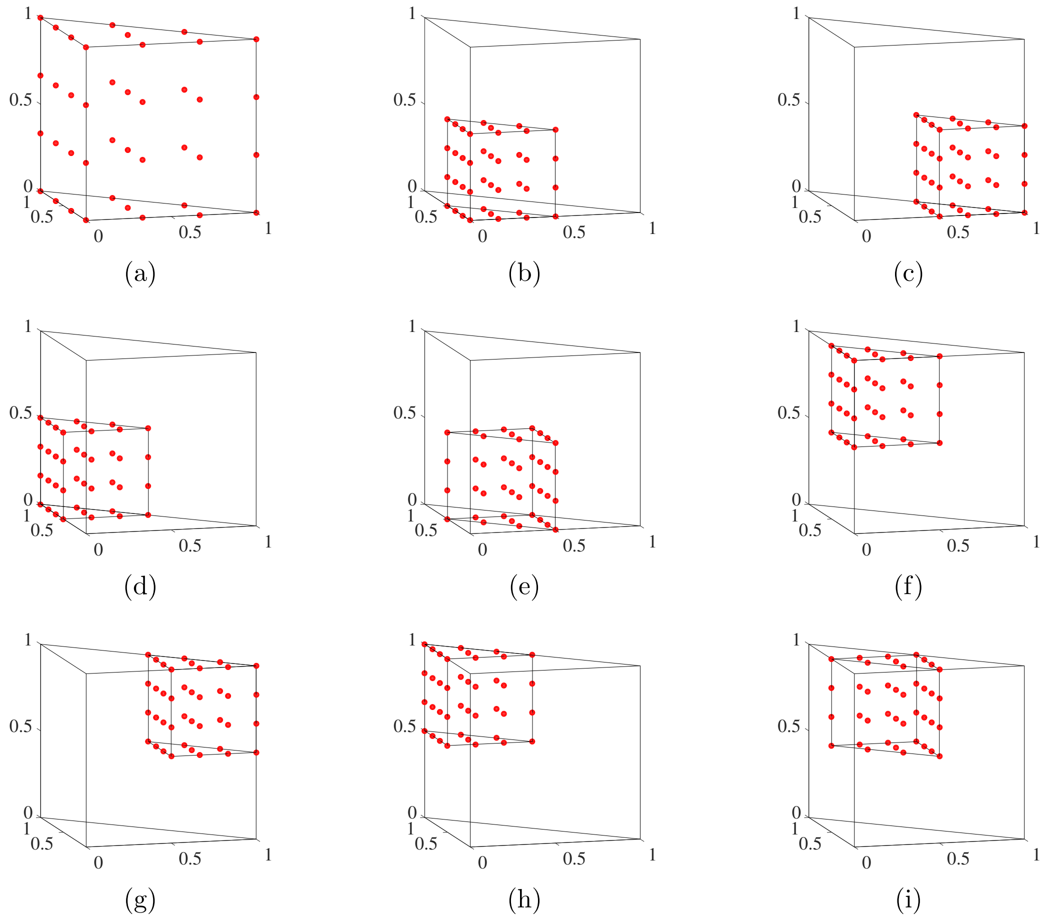


Figure 4.13: Refinement of a cubic Bézier wedge. (a) Collocation points for calculating $\mathbf{S}_{ij}$. (b-i) Collocation points for calculating $\mathbf{T}_{ij}^{k}$.

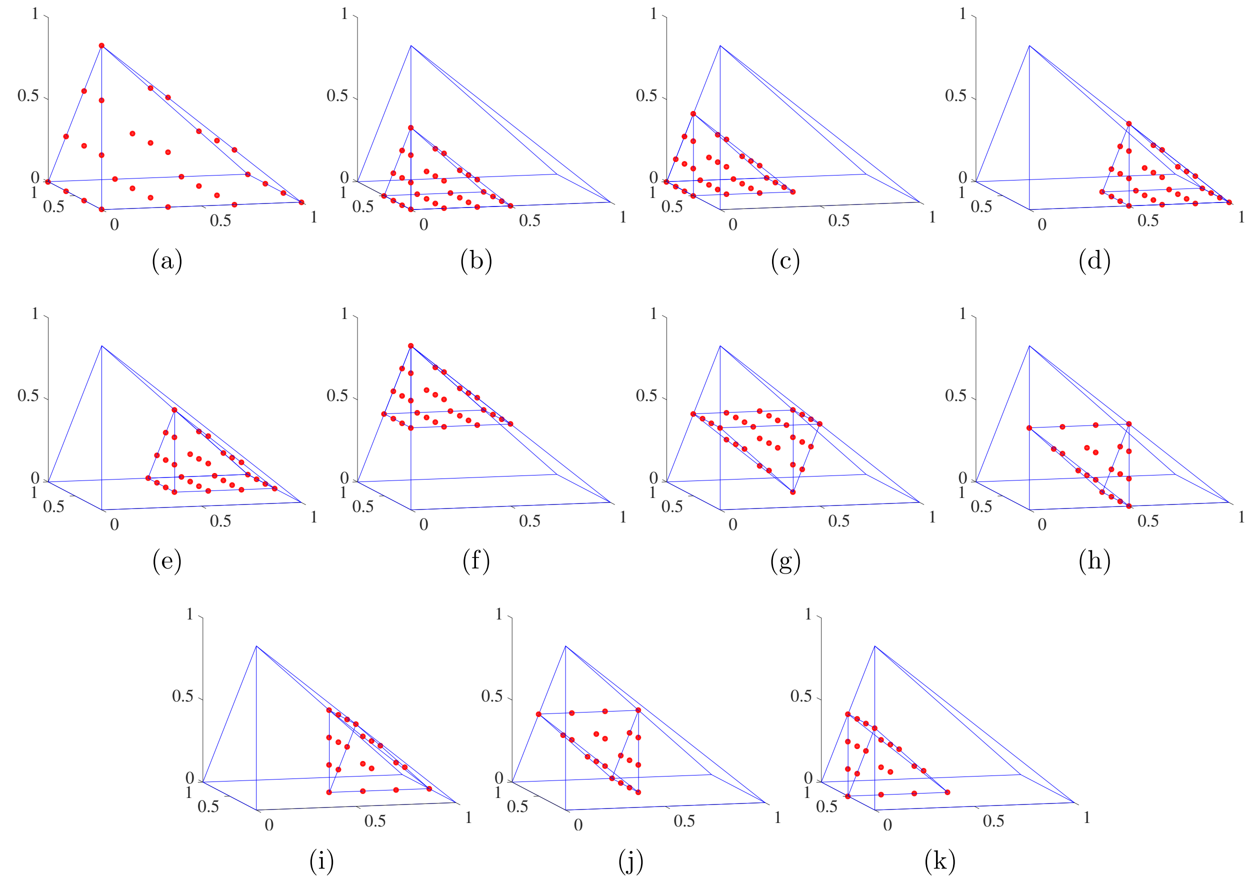


Figure 4.14: Refinement of a cubic Bézier pyramid. (a) Collocation points for calculating $\mathbf{S}_{ij}$. (b-k) Collocation points for calculating $\mathbf{T}^k_{ij}$.

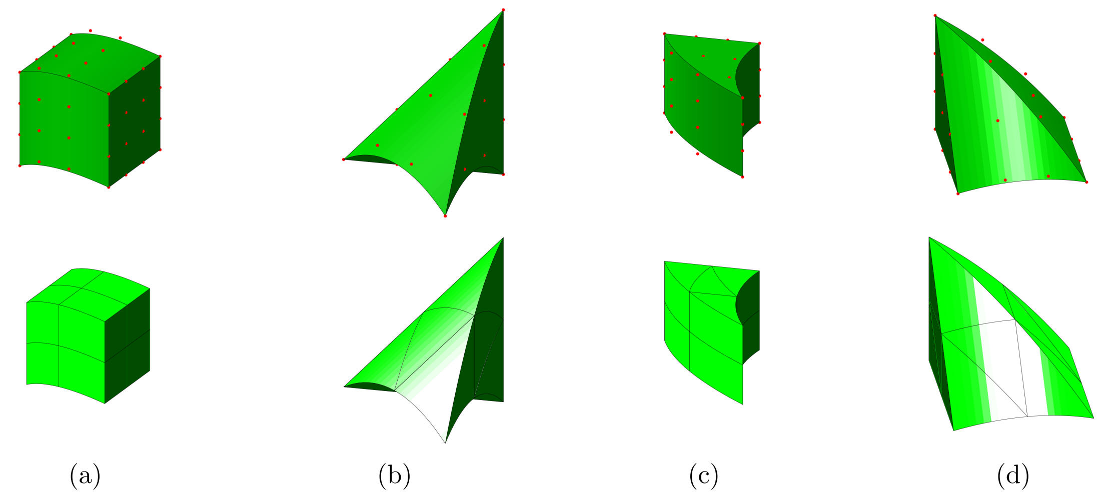


Figure 4.15: Uniform subdivision of the meshing primitives.

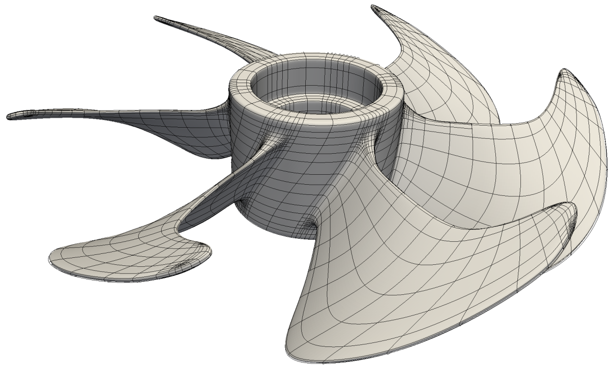
4.5 Mesh Gallery

With all of the technical details regarding mesh generation covered, we take this opportunity to present a few example meshes generated using the methodology presented in this dissertation. Figure 4.16 shows the mesh of a marine propeller, composed of sextic Bernstein-Bézier tetrahedra. The original T-spline surface is shown in Figure 4.16a, followed by the surface mesh in Figure 4.16b. Figure 4.16c shows a detailed view of the surface mesh with the boundaries of Bézier elements highlighted in bold, and Figure 4.16d shows a cut cell view of the volumetric mesh. To create the mesh of the propeller, a compatible linear mesh was first created using Tetgen [66]. Then, using our test for P_2 complexity, we determined the P_2 complexity of the propeller T-spline surface to be $P_2 = 6$. Finally, we created a geometrically exact mesh composed of sextic Bernstein–Bézier tetrahedra through degree elevation and surface reconstruction.

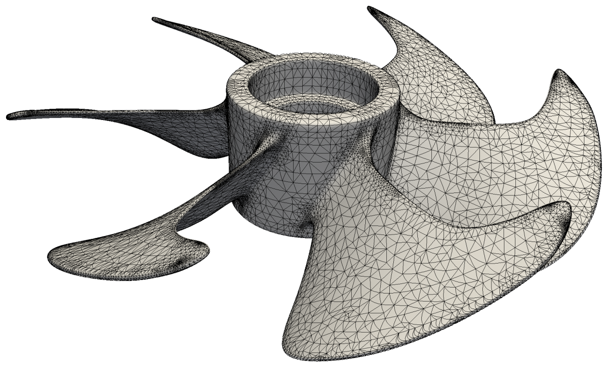
Figure 4.17 shows the mesh of a bike frame, composed of cubic Bernstein-Bézier tetrahedra. Again, the T-spline surface is shown in Figure 4.17a, followed by the surface mesh in Figure 4.17b, a detailed view in Figure 4.17c, and a cut cell view in Figure 4.17d. As with the propeller, a compatible linear mesh was created using Tetgen, and we found the P_2 complexity of the surface to be $P_2 = 6$. However, to demonstrate the flexibility of our method, we elect instead to create a lower-order mesh of the bike. This is achieved by simply performing degree elevation and surface reconstruction using Bernstein-Bézier tetrahedra of polynomial degree $p = 3$ rather than $p = 6$.

Finally, Figure 4.18 shows a mixed-element boundary layer mesh around an aircraft wing, composed of cubic Bernstein-Bézier elements. In this example, a boundary layer mesh of hexahedra is created around the wing surface, but transitions to tetrahedra via a layer of pyramids in the far field. Of course, a boundary layer mesh would normally consist of many, much thinner, hexahedral elements, but we just show a single, thicker, layer here for illustration purposes. For this mesh we begin with a NURBS representation of the aircraft wing shown in Figure 4.18a. Then, the hexahedral and pyramidal boundary layer elements were created by our own simple boundary layer mesh tool (Figure 4.18b). Triangle [65] was then used to create a quality linear surface mesh of the bounding box, and Tetgen was then used to create the linear volumetric mesh (Figure 4.18c). Figure 4.18d shows a detailed view of the boundary layer mesh transitioning to the tetrahedral mesh.

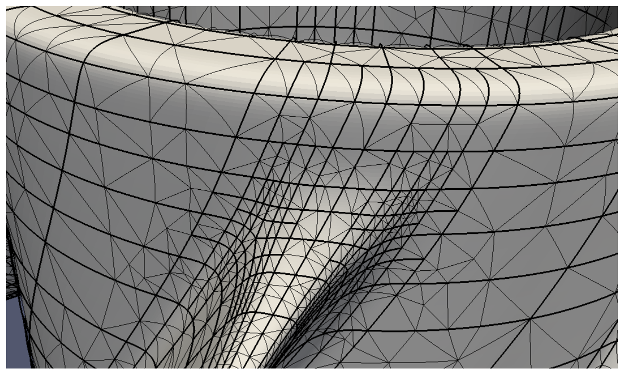
These three examples highlight several key aspects of our meshing procedure. First, the detailed views shown in Figure 4.16c and Figure 4.17c illustrate an important property of unstructured surface meshes. Namely, the length scale of the surface mesh is constrained by the length scale of the local Bézier elements as a result of the criteria imposed on the surface mesh outlined in Section 4.3.1. Second, the bike frame and the propeller serve to show how our method can easily generate either geometrically exact meshes or lower-order approximations. It is also worth noting that although the bike mesh is not geometrically exact, the lower-order mesh approximates the exact geometry with a great deal of accuracy. Finally, the boundary mesh around the airfoil illustrates one of the attractive advantages of mixed-element meshes, namely the ability to create geometrically exact volumetric meshes at relatively low polynomial degree.



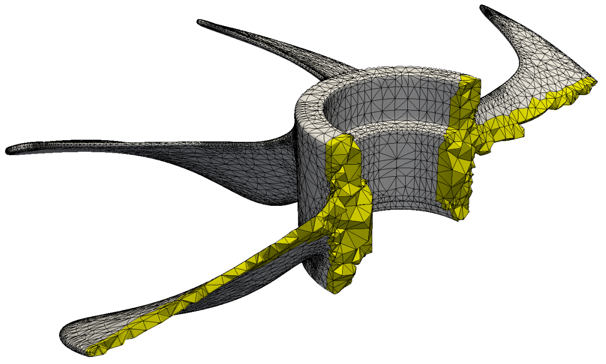
(a)



(b)



(c)



(d)

Figure 4.16: Geometrically exact mesh of a marine propeller using sextic Bézier tetrahedra. (a) T-spline surface. (b) Surface mesh. (c) Surface mesh detail. (d) Cut view of interior mesh.

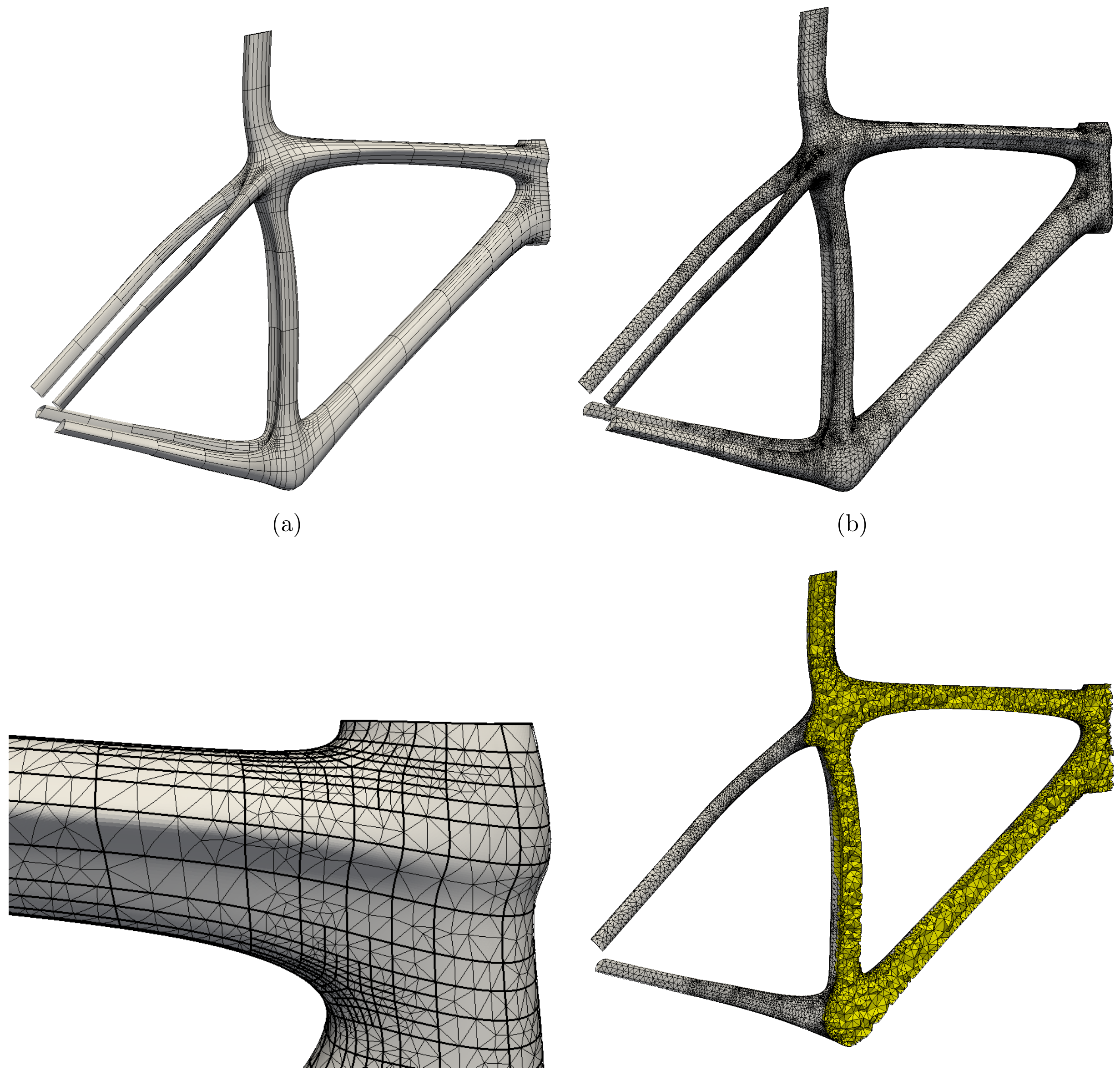
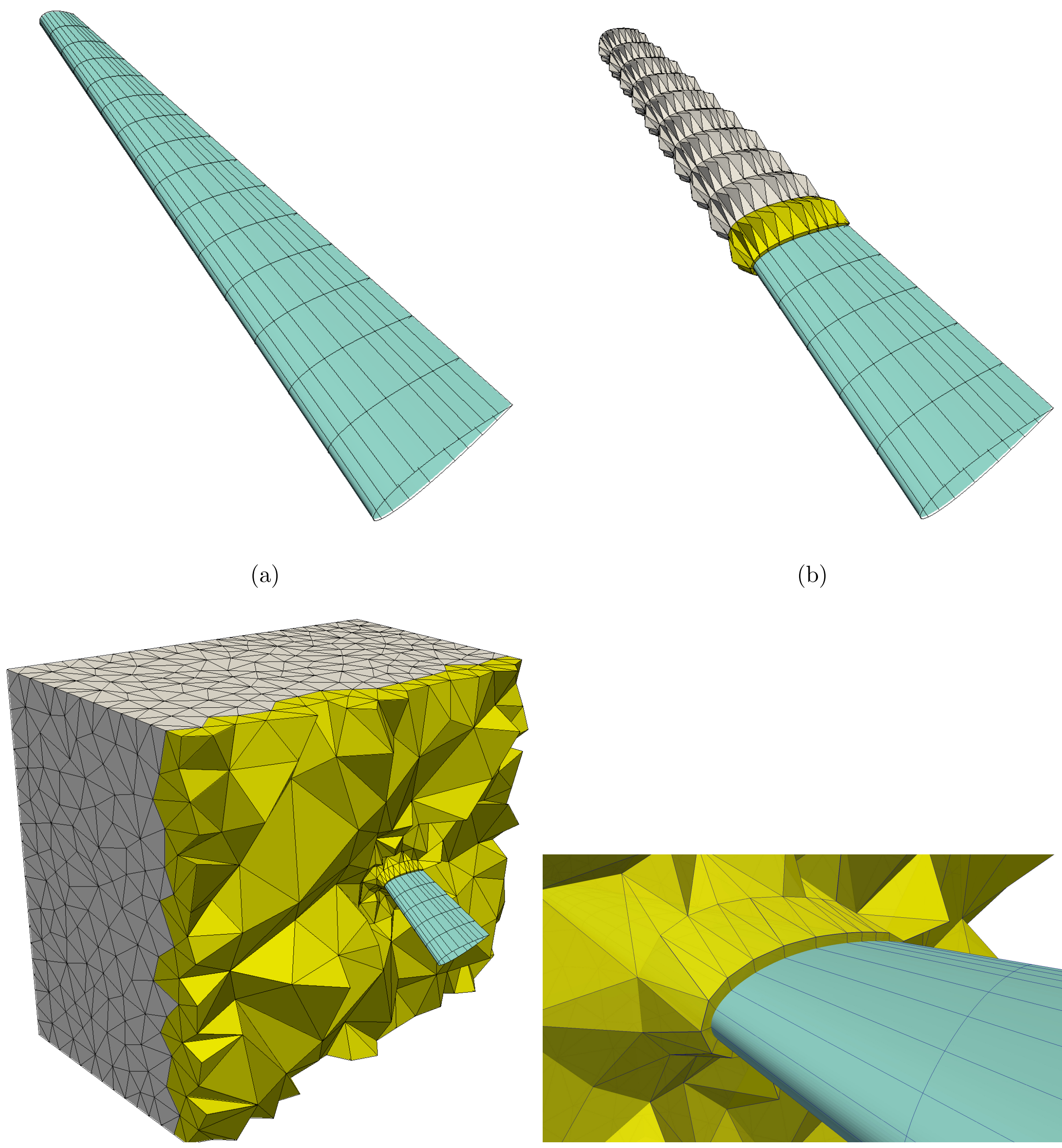


Figure 4.17: Lower-order mesh of a bike frame using cubic Bézier tetrahedra. (a) T-spline surface. (b) Surface mesh. (c) Surface mesh detail. (d) Cut view of interior mesh.



(a) (b) (c) (d)

Figure 4.18: Mesh of the volume around an aircraft wing. (a) NURBS surface of the aircraft wing. (b) Boundary layer mesh of hexahedra and pyramids. (c) Cut view of volume mesh. (d) Volume mesh detail.

4.6 Numerical Examples

We now turn our attention from the problem of mesh generation to that of the analysis suitability of these discretizations. Specifically, we perform three numerical tests. First, a patch test is performed to demonstrate that the elements presented here are capable of exactly representing linear solutions, even under large deformation. Second, the method of manufactured solutions is used to verify that optimal convergence rates are observed. Finally, we present the solution of a linear elasticity problem over the bike frame from Figure 4.17 to demonstrate the analysis suitability of a more complicated geometry. For brevity, we consider only cubic Bernstein-Bézier discretizations in what follows.

4.6.1 Patch Test

Following Lipton et al. [41], we would like to show that these proposed higher order elements are capable of exactly representing certain solutions, even under severe mesh distortion. Since the behavior of hexahedra have already been extensively studied in the previous literature, we will limit our tests to tetrahedra, pyramids and wedges. For each class of element, we create a discretization of the unit cube using only that element type. in each case, the mesh is created so that each element has at least one face on the global boundary of the cube, and all the elements share a single vertex in the center. Figure 4.19 shows the meshes for each of the three geometric primitives. Then for each mesh, we perform two patch tests to study the behavior of the elements under severe distortion.

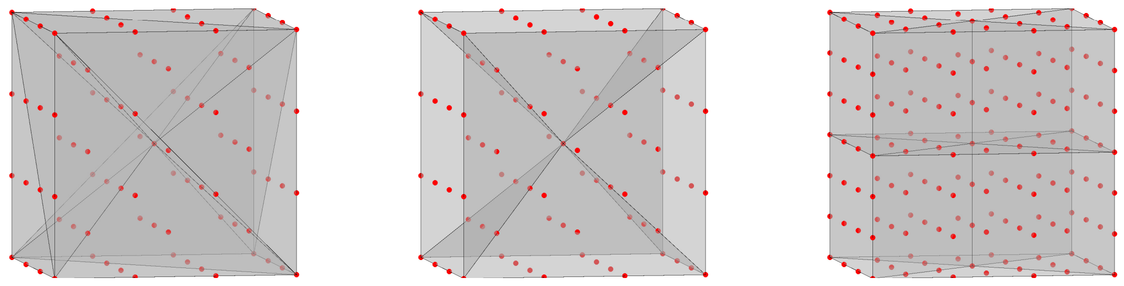
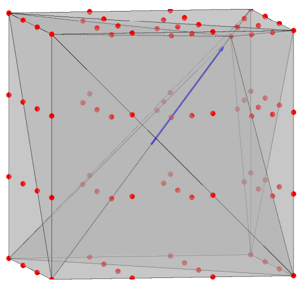


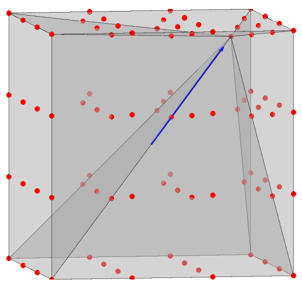
Figure 4.19: Undeformed meshes for the patch test. (a) Tetrahedral mesh. (b) Pyramidal mesh. (c) Wedge mesh.

Test 1: In the first test, the center node that is shared by all elements is moved to one of the corners of the cube, causing the mesh to become skewed as is shown in Figure 4.20. However, the distribution of internal nodes still remains linear across the element. That is, the linear quality degrades significantly with increasing distortion, but higher order quality remains the same.

Test 2: In the second test, the center node is left in the center, but internal nodes of the Bézier elements are collapsed to the center point as shown in Figure 4.21. In this way, the linear quality remains unchanged, but higher order quality degrades as the element is distorted.

We then solve two linear elasticity problems over each mesh, a constant tension, and constant shear Dirichlet problem, as illustrated in Figure 4.22. In each case, the bottom nodes are held fixed. In the constant tension case, we apply a uniform displacement of $d = 0.1$ in the z direction to the top nodes. In the constant shear case, we apply a uniform displacement of $d = 0.1$ in both the x and y direction to the top nodes. We also enforce zero z displacement on the sides of the cube. For each test, we gradually increase the mesh deformation from 0 (no deformation) to 1 (fully deformed), and evaluate the solution for each patch test at the control points on each element. If the computed solution agrees with the theoretical solution to within 13 decimal places for both the constant tension and constant shear case, we say that the patch test is passed. Table 4.1 shows the maximum deformation achieved for each test and element type before the patch test was no longer passed.


(a)


(b)

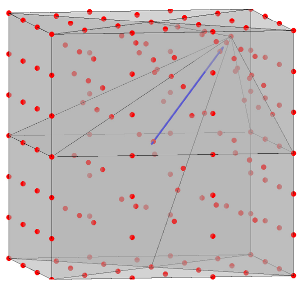
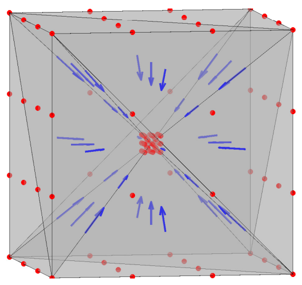
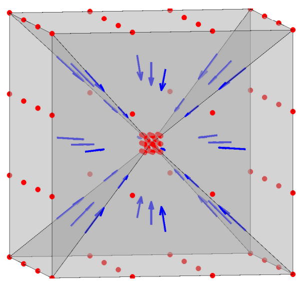

(c)

Figure 4.20: Deformations for the first patch test. (a)Tetrahedral mesh. (b) Pyramidal mesh. (c) Wedge mesh.


(a)


(b)

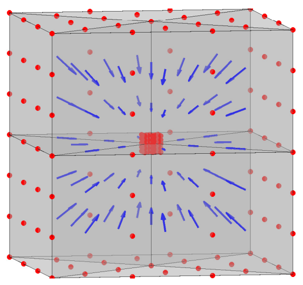
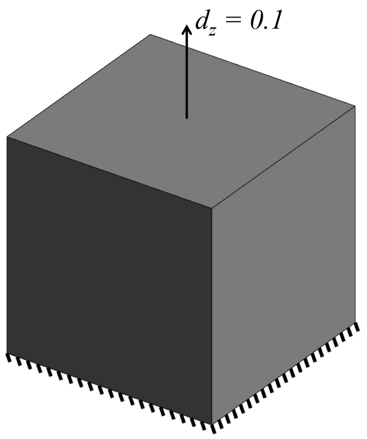
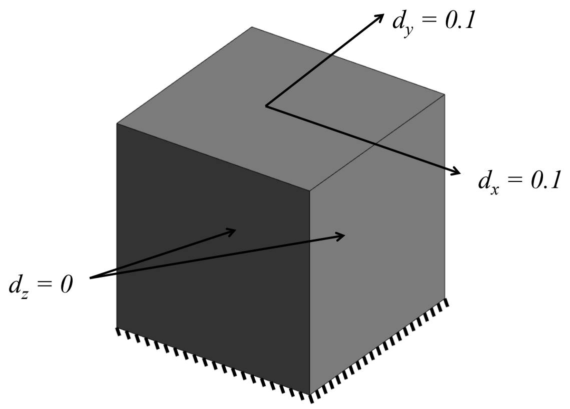

(c)

Figure 4.21: Deformations for the second patch test. (a)Tetrahedral mesh. (b) Pyramidal mesh. (c) Wedge mesh.



(a)



(b)

Figure 4.22: Boundary conditions for (a) the constant tension, and (b) the constant shear patch tests.

Table 4.1: Patch Test Results

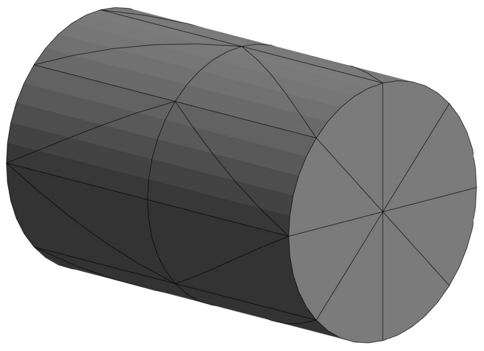
	Test 1	Test 2
Tetrahedra	0.99	0.99
Pyramids	0.98	0.99
Wedges	0.5	0.99

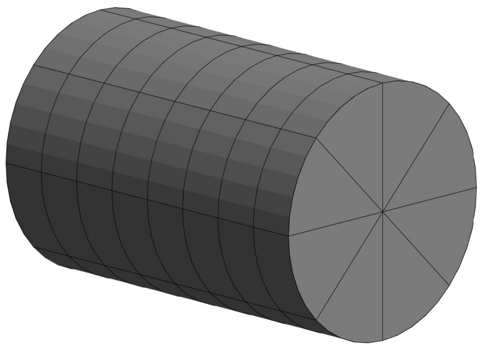
4.6.2 Method of Manufactured Solutions

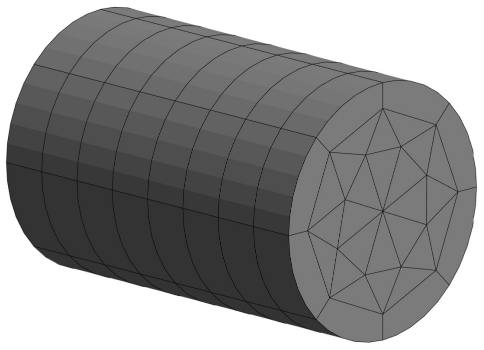
We also demonstrate higher order convergence of our method using the method of manufactured solutions. For the test, we consider three different meshes of a cylinder, including (a) a pure tetrahedral mesh, (b) a pure wedge mesh and (c) a mixed element mesh consisting of hexahedra, pyramids, and and tetrahedra. The three meshes are shown below in Figure 4.23. For each mesh, we define a temperature field over the cylinder defined by the function:

$$\phi(\boldsymbol{x}) = (x_1/L)^2 x_2^2 x_3^2$$

The temperature field is shown in Figure 4.23d. We then enforce Dirichlet boundary conditions on the faces of the cylinder, and solve Poisson's equation for the temperature field over the domain under successive levels of refinement. The convergence plots of the error in the L^2 and H^1 norms versus the non-dimensional mesh size for each mesh are shown in Figure 4.24. As is readily seen from Figures 4.24a-b, the tetrahedral meshes exhibit optimal convergence rates after several levels of refinement. Additionally, both the wedge mesh and the mixed element mesh approach optimal convergence rates under successive refinement, as can be seen in Figures 4.24c-d and Figures 4.24e-f, respectively.


(a)


(b)


(c)

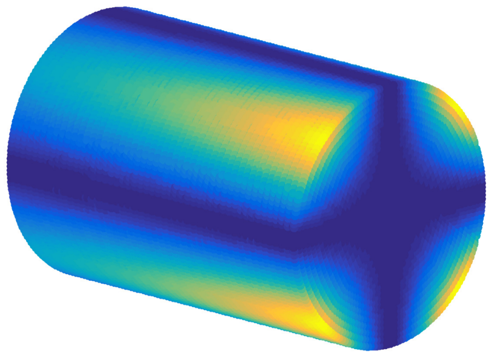
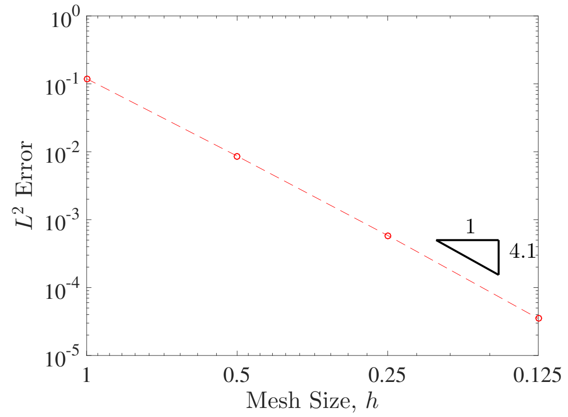
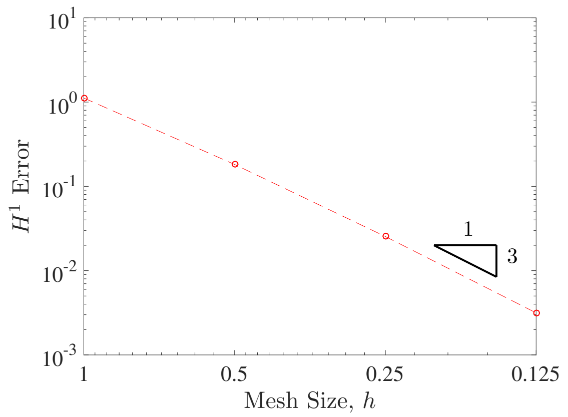

(d)

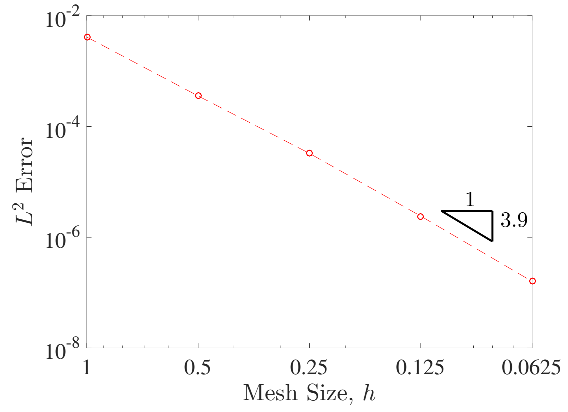
Figure 4.23: Meshes used with the method of manufactured solutions. (a) Tetrahedral mesh (b) Wedge mesh. (c) Mixed element hexahedral-pyramidal-tetrahedral mesh. (d) The prescribed temperature field.



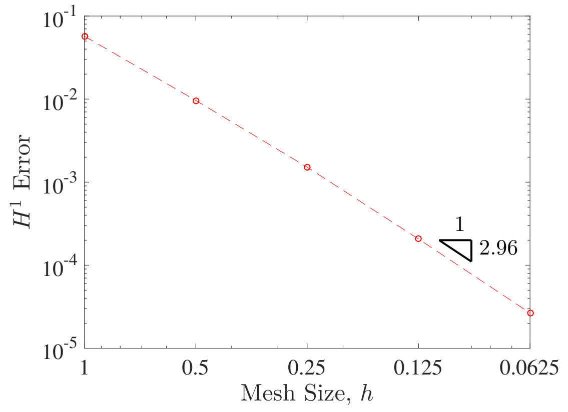
(a)



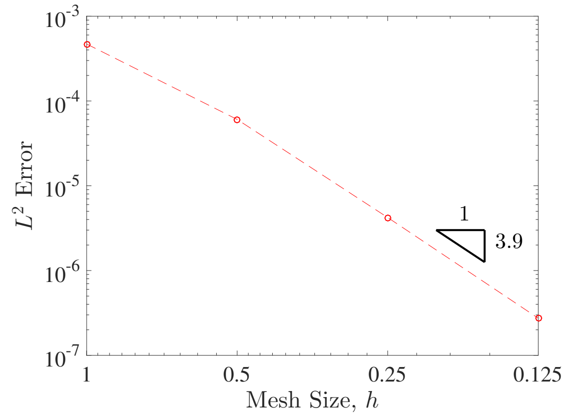
(b)



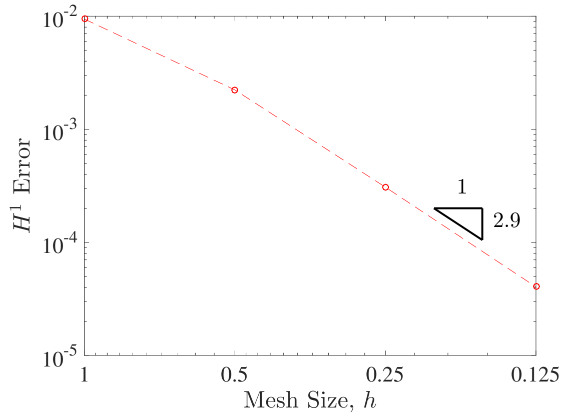
(c)



(d)



(e)



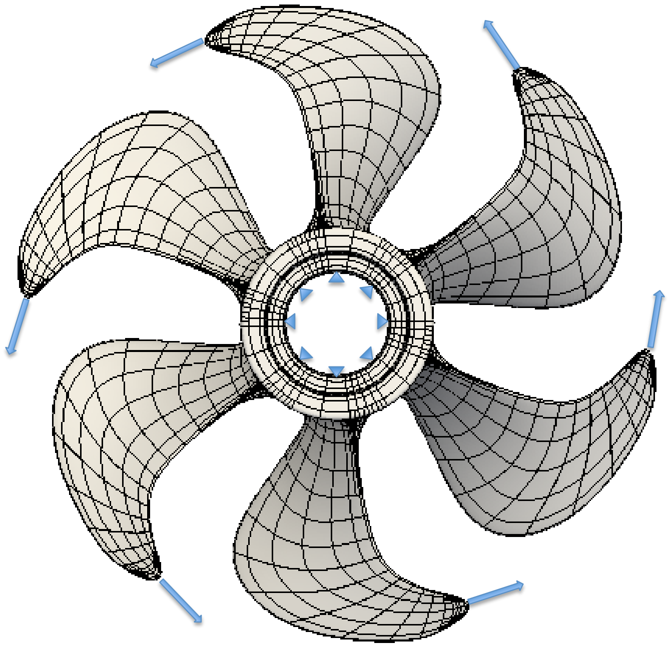
(f)

Figure 4.24: Convergence plots for three different meshes (a,b) L^2 and H^1 convergence plots for the tetrahedral mesh. (c,d) L^2 and H^1 convergence plots for the wedge mesh. (e,f) L^2 and H^1 convergence plots for the mixed element mesh.

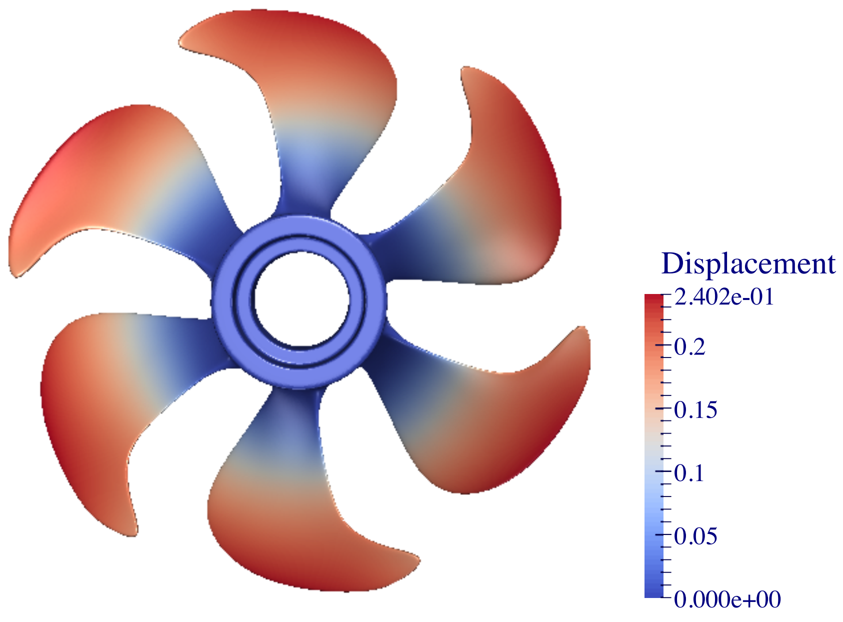
4.6.3 Practical Examples

Finally, we present some analysis results for some slightly more interesting geometries. First, we simulate the effect of torsion on the propeller from Figure 4.16 by solving a linear elasticity problem using a superparametric approach. We prescribe zero displacement boundary conditions on the interior radius of the propeller, and prescribe set displacements for the propeller tips, as shown in Figure 4.25a. We then solve the linear elasticity equations over the mesh of sextic tetrahedra, while using cubic Bernstein polynomials as the basis for analysis. Figure 4.25b shows the resulting displacement magnitude.

Next, we solve a linear elasticity problem over the bike frame from Figure 4.17. The head tube and rear stays are held fixed as shown in Figure 4.26a, and a downward force is applied on the top of the seat post. For the bike simulation, we employed an isoparametric approach, using cubic Bernstein polynomials as the basis for a simulation over the cubic tetrahedral mesh. Figure 4.26b illustrates the magnitude of the displacement of the bike under the applied load. For both examples, the results seem to be accurate, at least from an intuitive sense, and, coupled with the other numerical tests in this section, give us confidence in the analysis suitability of these discretizations.

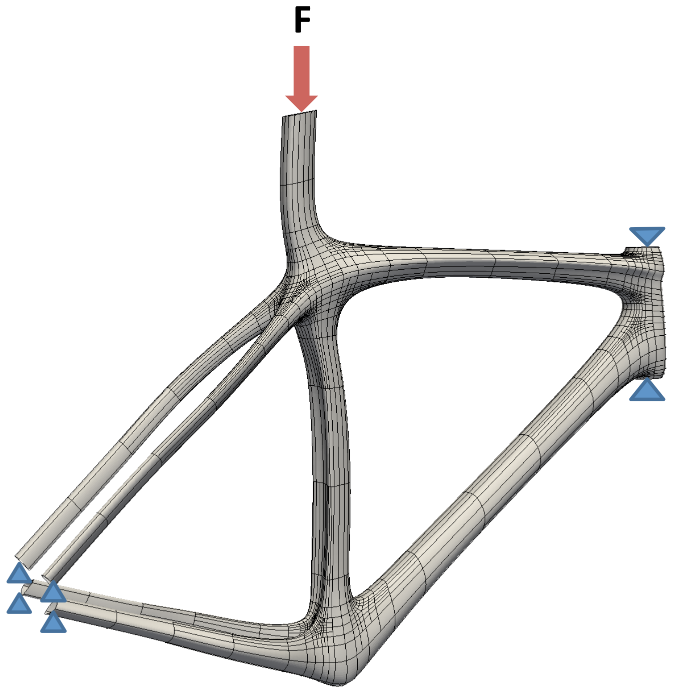


(a)

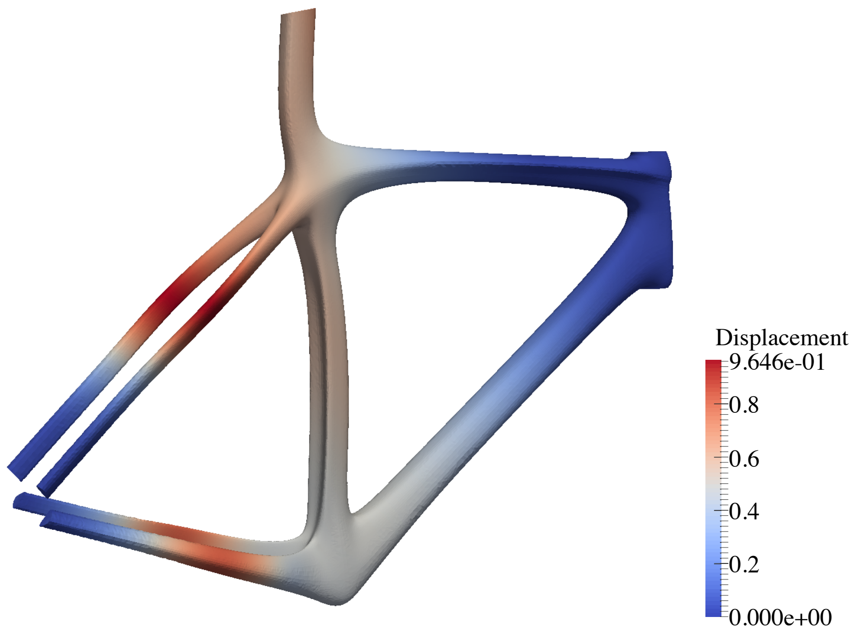


(b)

Figure 4.25: Superparametric simulation of linear elasticity on the propeller. (a) Boundary conditions. (b) Displacement magnitude.



(a)



(b)

Figure 4.26: Lower-order isoparametric simulation of linear elasticity on a bike frame. (a) Boundary conditions. (b) Displacement magnitude.

Chapter 5

Mesh Quality

With a framework for geometrically exact mesh generation established, it remains to provide a means to verify that the resulting meshes are analysis suitable. In this chapter, we present sufficient and computable mesh quality metrics for rational Bernstein–Bézier elements. I begin by introducing some notation not yet used in this dissertation, and then provide a review of the literature regarding finite element interpolation theory. I then present asymptotic error bounds on interpolation error over rational Bernstein–Bézier elements. Using these error bounds, we derive sufficient conditions for guaranteeing that a Bernstein–Bézier element is analysis suitable. Then, we develop easily computable element quality metrics for verifying if these conditions hold. Finally, we conclude this chapter with numerical examples demonstrating the use of these element quality metrics.

5.1 Notation and Preliminaries

5.1.1 Bernstein–Bézier Elements

Throughout this chapter, we make heavy use of the notation for Bernstein–Bézier element established in Chapter 3. In this chapter, we consider **only** simplicial and tensor product elements, but the ideas presented here can be readily extended to wedge elements, as they are simply the tensor product of a Bernstein–Bézier 1-simplex and a Bernstein–Bézier 2-simplex. Bernstein–Bézier pyramids on the other hand are slightly more complicated, and as of yet, we have not extended our theory to this case. Nonetheless, the results presented here are incredibly useful, as simplicial and tensor product elements are by far the most commonly used elements for analysis. Finally, we note that for in this chapter, we consider only the case wherein the element is a d manifold with codimension zero. That is, we assume that for triangular and quadrilateral elements $d = d_r = d_s = 2$, and for tetrahedral and hexahedral elements, we recognize that $d = d_r = d_s = 3$.

5.1.2 Derivative Notation

In order to write derivatives compactly, we denote the partial derivative operator of order $|\boldsymbol{\alpha}|$ with respect to the variables $\boldsymbol{\xi}$ as:

$$D^{\boldsymbol{\alpha}}_{\boldsymbol{\xi}} = \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial \xi_1^{r_1} \partial \xi_2^{r_2} ... \partial \xi_d^{r_d}}$$

The derivative of a vector–valued function $\boldsymbol{f}$ of length m is understood to result in a $m \times 1$ column vector, viz.:

$$D^{\boldsymbol{\alpha}}_{\boldsymbol{\xi}} \boldsymbol{f} = \begin{bmatrix} D^{\boldsymbol{\alpha}}_{\boldsymbol{\xi}} f_1 \\ \vdots \\ D^{\boldsymbol{\alpha}}_{\boldsymbol{\xi}} f_m \end{bmatrix}$$

and we denote the set of all partial derivatives of order $k = |\boldsymbol{\alpha}|$ as the matrix:

$$\boldsymbol{\nabla}^k_{\boldsymbol{\xi}} \boldsymbol{f} = \begin{bmatrix} D^{\boldsymbol{\alpha}_1}_{\boldsymbol{\xi}} \boldsymbol{f} & D^{\boldsymbol{\alpha}_2}_{\boldsymbol{\xi}} \boldsymbol{f} & \dots & D^{\boldsymbol{\alpha}_n}_{\boldsymbol{\xi}} \boldsymbol{f} \end{bmatrix} \quad \forall \ \boldsymbol{\alpha}_i : |\boldsymbol{\alpha}_i| = k$$

We note that $\boldsymbol{\nabla}^1_{\boldsymbol{\xi}} = \boldsymbol{\nabla}_{\boldsymbol{\xi}}$ is the standard gradient operator, and that $\boldsymbol{\nabla}^{\mathbf{0}}_{\boldsymbol{\xi}}$ is understood to be the identity operator. Additionally, to shorten certain equations, we at times omit the subscript denoting the independent variable. That is, we write $\boldsymbol{\nabla f} = \boldsymbol{\nabla}_{\boldsymbol{\xi}} \boldsymbol{f}$ when the choice of the independent variable is unambiguous.

5.2 Review of Finite Element Interpolation Theory

Before proceeding to the novel contributions of this work, it is critical to understand the existing theory regarding error bounds for finite elements, and how these error bounds motivate the need for element quality metrics. First, we briefly review the isoparametric concept as it applies to finite elements (Section 5.2.1). We then review the fundamental interpolation theory for both linear (Section 5.2.2) and curvilinear (Section 5.2.3) finite elements. Finally, we present the current state of the art with regards to element quality metrics for curvilinear finite elements (Section 5.2.4).

5.2.1 The Isoparametric Concept

Simply put, the isoparamtric concept allows us to define an element in physical space in terms of a mapping from a reference element in parametric space. Let us denote a unit reference element in parametric space $\hat{\Omega}$. Then we denote the element in physical space Ω^e, and denote a mapping $\mathbf{x}^e$ that maps points on the parametric element to points on the physical element. In the case of p-version finite elements, this is a higher-order mapping. Finally, we also consider the element $\overline{\Omega}^e$ which is the **purely linear** physical element. That is, $\overline{\Omega}^e$ is defined by an affine mapping $\overline{\mathbf{x}}^e$. We assume that the mapping $\mathbf{x}^e$ is known. Then we define the linear mapping $\overline{\mathbf{x}}^e$ as:

$$\overline{\mathbf{x}}^e(\boldsymbol{\xi}) = \mathbf{x}^e(\mathbf{0}) + \sum_{i=1}^{d} \left(\mathbf{x}^e(\mathbf{e}_i) - \mathbf{x}^e(\mathbf{0})\right) \xi_i$$

wherein $\mathbf{e}_i$ is the unit vector in the i^{th} parametric direction, and $\mathbf{0}$ is the $d \times 1$ vector of zeros.

These mappings are illustrated in Fig. 5.1. We note that for simplicial elements, the linear physical element is simply defined as the linear interpolant of the corners of the curvilinear physical element. However, for tensor product elements (i.e. quadrilaterals or hexahedra), the affine element will not necessarily interpolate every corner of the curvilinear element, as seen in Fig. 5.1. This is due to the fact that the tensor product admits bilinear mappings for quadrilaterals and trilinear mappings for hexahedra.

5.2.2 Finite Element Interpolation Theory: Linear Elements

The rigorous study of the mathematical foundations of the finite element method began in earnest in the 1960's and 1970's [4, 11, 12, 17, 18, 83, 84]. Of particular note are the pioneering works of Bramble and Hilbert [11, 12], and Ciarlet and Raviart [17, 18], which led to the **interpolation theory for finite elements**. As this fundamental theory has close bearing on our current work, we

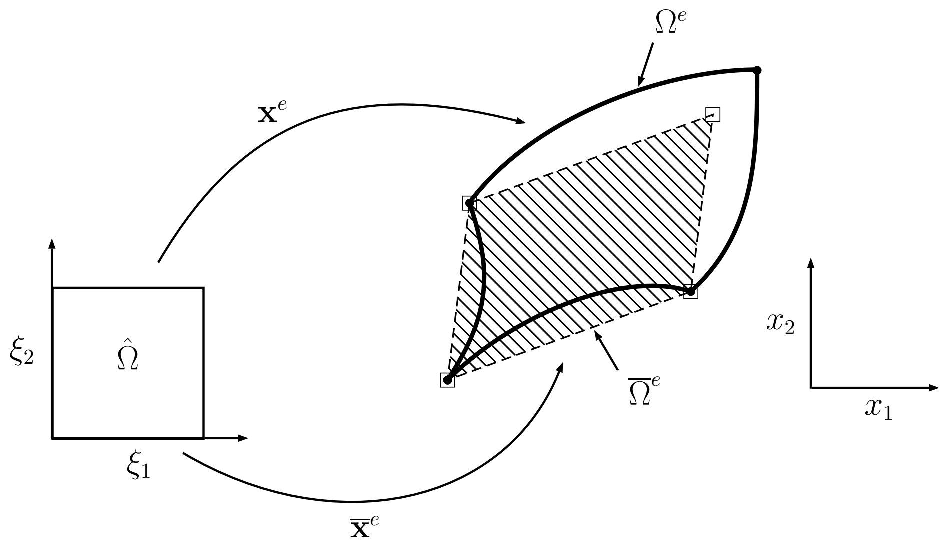


Figure 5.1: Isoparametric mappings from a reference element $\hat{\Omega}$ in parametric space to physical space. The element Ω^e (shown by the bold line) is defined by the higher-order mapping $\mathbf{x}^e$. The corresponding affine element $\overline{\Omega}^e$ (shown by the dashed line) is defined by a purely affine mapping $\overline{\mathbf{x}}^e$.

take this opportunity to briefly review the theory here, introducing notation to be used throughout the remainder of this dissertation.

In the finite element method, we approximate a domain Ω using a set of finite elements, $\{\Omega^e\}_{e=1}^{E}$, where each element $\Omega^e \in \mathbb{R}^d$ is an open simply connected set, with simply connected boundary. Together, this collection of elements forms a finite element discretization or mesh, which we denote as:

$$\mathcal{M} = \overline{\bigcup_{e=1}^{E} \Omega^e}$$

Traditionally, a finite element mesh is composed of linear triangular or quadrilateral elements in $\mathbb{R}^2$, and linear tetrahedral or hexahedral elements in $\mathbb{R}^3$. The study of mesh quality, then, concerns itself with how the shapes and sizes of these elements affect the accuracy of the finite element method. In order to begin this study, it is first useful to introduce some mesh measures. Let us denote the diameter of an element as h_e, where we measure the diameter as the largest distance between any two vertices of the element. Next, we denote the diameter of the incircle (in $\mathbb{R}^2$) or insphere (in $\mathbb{R}^3$) of the element as ρ_e. These two metrics are visualized for a quadrilateral element in Fig. 5.2.

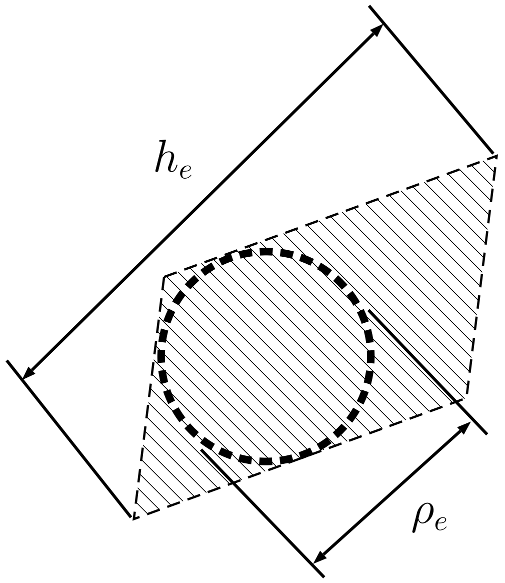


Figure 5.2: Element measures ρ_e and h_e for the linear quadrilateral element $\overline{\Omega}^e$ from Fig. 5.1.

Given these element-wise measures, we can then define the corresponding global mesh measures as:

$$h = \max_{1 \leq e \leq E} h_e$$

$$\rho = \min_{1 \leq e \leq E} \rho_e$$

These mesh measures are useful, as they allow us to put certain classifications on our meshes. Specifically, it allows us to introduce the notion of **shape regularity**. For a linear element $\overline{\Omega}^e$, the **element** shape regularity is given by:

$$\sigma_e = \frac{h_e}{\rho_e}$$

and for a mesh of linear elements, the **global** shape regularity is given by:

$$\sigma = \frac{h}{\rho}$$

The notion of shape regularity is important, as it is allows us to compare the shapes of elements independent of their size. This is motivated by the fact that we are interested in the effect of element shape on approximation error under mesh refinement, that is as $h \to 0$.

Let us consider a set of M increasingly refined meshes $\{\mathcal{M}\}_{i=1}^{M}$, with corresponding metrics $\{h_i\}_{i=1}^{M}$ and $\{\rho_i\}_{i=1}^{M}$ where $h_1 > h_2 > \ldots > h_M$. We say a refinement is **uniform** if each element is simply split into a collection of similarly shaped sub-elements, thereby preserving the existing geometrical structure of the parent mesh. Due to the ease of implementation, and constant element shape, uniform refinements are frequently used with the finite element method. However, uniform refinements are constrained by the choice of the initial mesh, and it is not always practical or even possible to perform uniform subdivision on a given mesh. It is useful then to introduce the concept of **quasi-uniform** refinement. Consider a series of refined meshes $\{\mathcal{M}\}_{i=1}^{M}$ with corresponding global shape regularity metrics $\{\sigma_i\}_{i=1}^{M}$. We say a series of refinements is quasi-uniform if we can bound σ_i by some constant σ_0, viz.:

$$\sigma_i \leq \sigma_0 \quad i = 1, 2, ..., M$$

In the case of both uniform and quasi-uniform refinements, we say that the set of all elements in the set of meshes belong to a **regular family** of elements. If a series of refinements is not uniform or quasi-uniform, we say that the refinements are **irregular**, and the elements do not belong to a regular family. These three classes of refinements are visualized for a simple mesh in Table 5.1.

With the necessary notation established, we can proceed to state the interpolation theory for finite elements. Suppose we have a sequence of meshes $\{\mathcal{M}_i\}_{i=1}^{M}$. Given some boundary value problem defined over the domain, we desire to approximate the true solution u using the finite element method. That is, we desire to approximate u by some approximation function u_h (a piecewise polynomial of degree p) for each mesh in the series. Then, following Raviart and Ciarlet [17], it can be shown that the error over a mesh with mesh size h is bounded by:

$$||u - u_h||_{H^m(\Omega)} \leq C\frac{h^{p+1}}{\rho^m}|u|_{H^{p+1}(\Omega)}$$

wherein C is a constant independent of the mesh size h, and $||\cdot||_{H^m(\Omega)}$ denotes the norm:

$$||f||_{H^m(\Omega)} = \left(\sum_{j=0}^{m}|f|^2_{H^j(\Omega)}\right)^{1/2}$$

and $|\cdot|_{H^j(\Omega)}$ denotes the seminorm:

$$|f|_{H^j(\Omega)} = \left(\int\limits_{\Omega}\sum_{|\boldsymbol{\alpha}|=j}|D^{\boldsymbol{\alpha}}f|^2\,d\Omega\right)^{1/2}$$

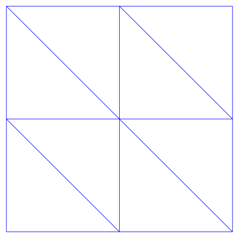
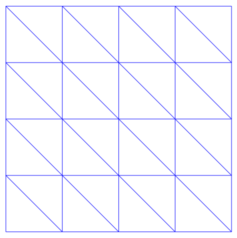
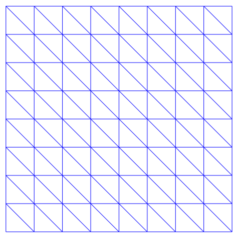
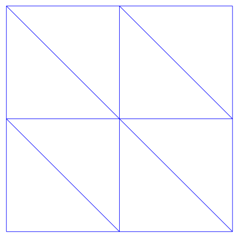
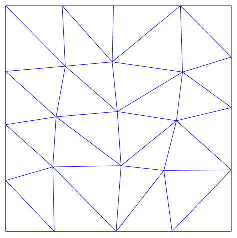
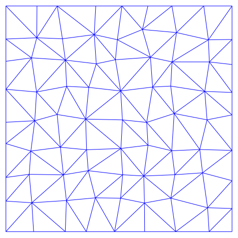
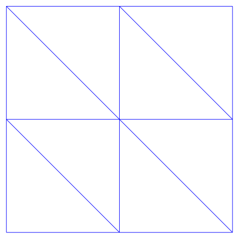
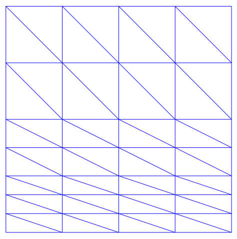
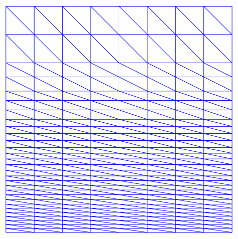
Furthermore, we notice that if every mesh in series belongs to a regular family, this bound simplifies to:

$$||u - u_h||_{H^m(\Omega)} \leq Ch^{p+1-m}|u|_{H^{p+1}(\Omega)}$$

5.2.3 Finite Element Interpolation Theory: Curvilinear Elements

Similar bounds have also been established for interpolation over curvilinear elements [18, 50]. In curvilinear mesh generation, the mesh is most often generated by first constructing a mesh of straight sided elements, and then manipulating element nodes to curve the elements to better match the geometry (see Fig. 5.3). Thus with curvilinear finite elements, there is the notion of both the

Table 5.1: Various types of mesh refinement. In uniform refinement, the structure of the original mesh is preserved in each level of refinement. In quasi-uniform refinement, the structure is not preserved, but all elements belong to a regular family. In irregular refinement, the bottom elements become increasingly thin, and therefore do no belong to a regular family.

	h = 0.5	h = 0.25	h = 0.125
Uniform Refinements			
Quasi-Uniform Refinements			
Irregular Refinements			

curvilinear mesh $\mathcal{M}$, as well as the underlying linear mesh $\overline{\mathcal{M}}$. Now, let us denote a mapping $\mathbf{x}^e$ that maps a linear master element $\hat{\Omega}$ to the physical curvilinear element Ω^e. Then, let us assume that the following conditions hold for every element in the mesh.

Cond. (5.2.3.1) The underlying linear elements $\overline{\Omega}^e$ belong to a regular family.

Cond. (5.2.3.2) The mapping $\mathbf{x}^e$ is invertible. That is:

$$\mathbf{x}^e(\boldsymbol{\xi}) = \boldsymbol{x} \Leftrightarrow \left(\mathbf{x}^e\right)^{-1}(\boldsymbol{x}) = \boldsymbol{\xi} \quad \forall\, \boldsymbol{\xi} \in \hat{\Omega}$$

Cond. (5.2.3.3) The derivatives of the mapping $\mathbf{x}^e$ are bounded as follows:

$$\sup_{\boldsymbol{\xi}\in\hat{\Omega}} \max_{|\boldsymbol{\alpha}|=k} \left|D^{\boldsymbol{\alpha}}_{\boldsymbol{\xi}} \mathbf{x}^e\right| \leq c_k h^k \quad 1 \leq k \leq p+1$$

$$\sup_{\boldsymbol{x}\in\Omega^e} \max_{|\boldsymbol{\alpha}|=1} \left|D^{\boldsymbol{\alpha}}_{\boldsymbol{x}} \left(\left(\mathbf{x}^e\right)^{-1}\right)\right| \leq c_0 h^{-1}$$

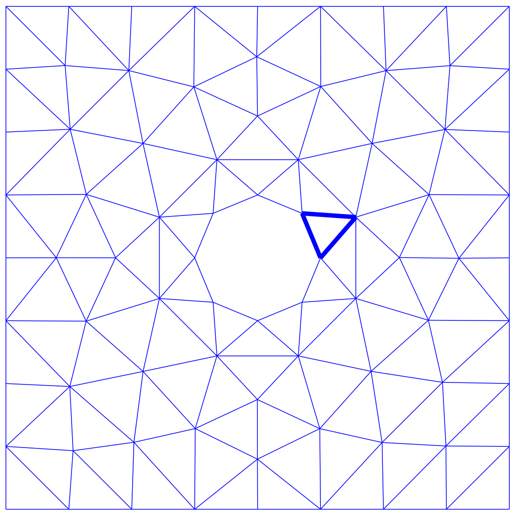
Then, for a series of meshes belonging to a regular family, we have the error bound:

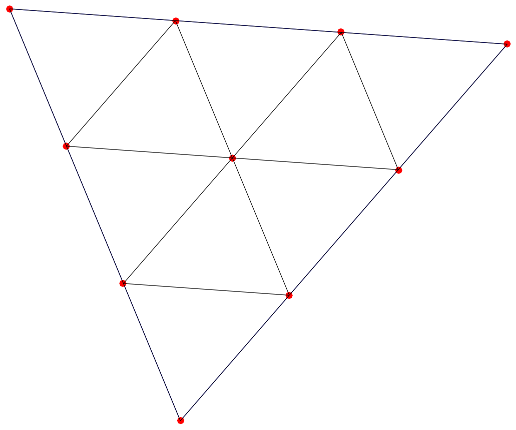
$$||u - u_h||_{H^m(\Omega)} \leq C \frac{\sup\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}(\boldsymbol{\xi})|}{\inf\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}(\boldsymbol{\xi})|} h^{p+1-m} ||u||_{H^{p+1}(\Omega)} \tag{5.1}$$

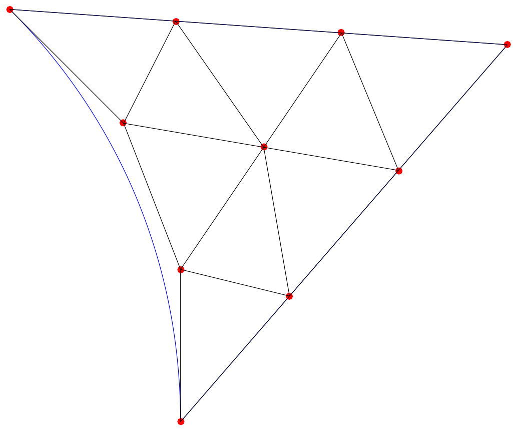
From this, we see that for a curvilinear mesh to exhibit similar convergence rates to a linear mesh, several criteria must hold. First, as before, the underlying linear elements must be shape-regular (Cond. 5.2.3.1). However, we must also ensure that the higher-order mapping is invertible (Cond. 5.2.3.2), and that its derivatives are bounded (Cond. 5.2.3.3). Put simply, we must ensure that the curvilinear elements are not **too** curvilinear.

5.2.4 Element Distortion and Quality Metrics

The results of the previous section illuminate some important considerations regarding the effect of element shape on the convergence rates of p-version finite element methods. From Cond. 5.2.3.2 and Cond. 5.2.3.3, there is a clear need to quantify the magnitude of element distortion, as there is a direct effect on element quality. This need has led to the development of so called **element distortion metrics** and **element quality metrics**. The precise definition of these terms is often


(a)


(b)


(c)

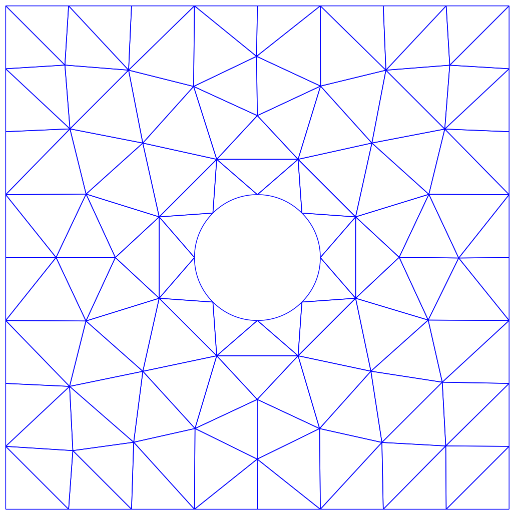

(d)

Figure 5.3: Steps for basic curvilinear mesh generation on a plate with a hole. (a) Create an initial linear mesh. A representative boundary element is highlighted in bold. (b) Degree elevate the linear element by inserting higher-order control points. (c) Curve the element to match the boundary by updating control point location. (d) Repeat for each element int he mesh to yield the final curvilinear mesh.

quite nebulous, and often varies from application to application. However, for the purposes of this dissertation we say element distortion metrics quantify the difference between an arbitrary element Ω^e, and some ideal element Ω_{ideal}. Element quality metrics, then, are are simply taken as the inverse of element distortion metrics. That is, element quality will increase as distortion decreases, and vice-versa.

In general, since element distortion metics and element quality metrics are closely related, we will refer to both with the umbrella term **element metrics**. To motivate the need for the novel element metrics presented in this dissertation, we review the existing curvilinear element metrics currently in use in the literature. We then argue that none of the existing element metrics are sufficient for guaranteeing optimal convergence rates of the p-version finite element method over curvilinear elements.

We begin our study of curvilinear element metrics by recognizing that Eq. (5.1) contains the term:

$$1 \leq \frac{\sup\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}|}{\inf\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}|} < \infty$$

If the mapping $\mathbf{x}^e$ becomes singular, then $\inf\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}| = 0$, and the error bounds in Eq. (5.1) will tend towards infinity. It is perhaps due to this observation that the overwhelming majority of curvilinear quality metrics are based on some measure of the Jacobian matrix. Of these Jacobian based quality metrics, perhaps the most commonly used is the **scaled Jacobian** [20, 52], defined as:

$$J_S = \frac{\inf\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}|}{\sup\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}|} \tag{5.2}$$

From Eq. (5.2), it is readily apparent that $0 \leq J_S \leq 1$, with element quality increasing as $J_S \to 1$. For an affine element, the Jacobian is constant across the element, and the metric is identically unity. For a singular or inverted element, $\inf\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det \mathbf{J}| = 0$, and the metric is zero.

Besides the scaled Jacobian, there have been other proposed higher order quality metrics all based on some measure of the Jacobian matrix, both in traditional p-version finite elements [26, 37, 54], and in IGA [24, 68, 78, 82]. Despite the wide array of metrics currently in use, we are not aware of any work relating bounds on these metrics to bounds on higher order derivatives. Thus, to our knowledge, none of the existing quality metrics are sufficient for guaranteeing that Cond. 5.2.3.3 holds. Because of this, we argue that existing curvilinear element metrics are insufficient for guaranteeing that an arbitrary curvilinear element is analysis suitable. To illustrate a particularly egregious example, consider the element shown in Fig. 5.4, which has a scaled Jacobian of $J_S = 1$. While not all Jacobian based quality metrics will indicate that this element is of good quality, it is troubling that the most commonly used quality metric for curvilinear elements cannot distinguish between this highly skewed element and a purely linear triangle. We hope this example motivates the need for further study of curvilinear element distortion metrics, and we identify two key challenges to be addressed by the present work in Section 5.5.

(1) **Bounds on the Jacobian Determinant of Rational Elements**

In general, the Jacobian determinant of a polynomial mapping of degree p is itself a polynomial of degree $p' = d(p-1)$. As such, bounding the Jacobian determinant from above and below is difficult for higher-order polynomial elements, and the task is even more difficult for elements defined by a rational mapping. We note that computable bounds have been establish for polynomial elements [34], but are unaware of any analogous bounds for rational elements.

(2) **Bounds on the Higher Order Derivatives of the Parametric Mapping**

We are not aware of any element metrics that quantify the magnitude of higher–order partial derivatives of the parametric mapping $\mathbf{x}^e : \hat{\Omega} \rightarrow \Omega^e$. Furthermore, we are not aware of any attempts to show that bounds on existing metrics imply bounds on higher–order derivatives[1] .

[1] In their original paper, Ciarlet and Raviart do provide conditions to ensure boundedness of the higher–order derivatives for certain classes of elements [18]. However, these conditions are too restrictive to be used effectively with modern automated meshing algorithms.

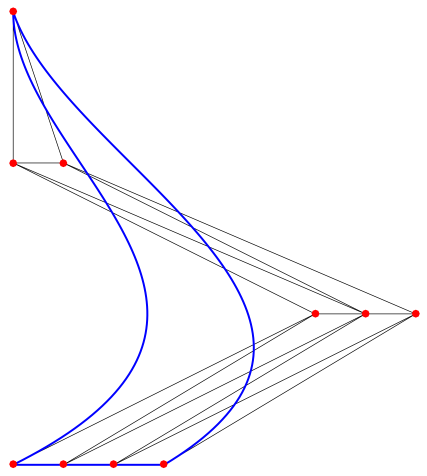


Figure 5.4: Highly distorted triangular element with a scaled Jacobian of $J_S = 1$.

5.3 Interpolation Theory for Rational Bernstein–Bézier Elements

With the necessary preliminaries established, we now turn our attention to the novel contributions of the present work. In this section, we present error estimates for rational Bernstein–Bézier elements of simplicial or tensor product construction. The analysis follows closely the work of Bazilevs et. al. [6], wherein interpolation error bounds were derived for IGA using NURBS.

Let us consider a rational Bernstein–Bézier element Ω^e with corresponding reference element in parametric space $\hat{\Omega}$. For simplicial elements, we denote the space of approximation functions of degree p over the physical curvilinear element as:

$$\mathcal{S}_p^h(\Omega^e) := \left\{ u_h \in L^2(\Omega^e) : w(u_h \circ \mathbf{x}^e) \in \mathscr{P}^p\left(\hat{\Omega}\right) \right\}$$

where $\mathscr{P}^p\left(\hat{\Omega}\right)$ denotes the space of polynomials of degree p, and $w \in \mathscr{P}^p\left(\hat{\Omega}\right)$ is the weighting function defined over the reference element. For tensor product elements, we denote the space of tensor product approximation functions of degree $\mathbf{p} = \{p, ..., p\}$ over the physical curvilinear element as:

$$\mathcal{S}_{\mathbf{p}}^h(\Omega^e) := \left\{ u_h \in L^2(\Omega^e) : w(u_h \circ \mathbf{x}^e) \in \mathscr{Q}^{\mathbf{p}}\left(\hat{\Omega}\right) \right\}$$

where $\mathscr{Q}^{\mathbf{p}}\left(\hat{\Omega}\right)$ denotes the space of tensor product polynomials of degree $\mathbf{p}$, and $w \in \mathscr{Q}^{\mathbf{p}}\left(\hat{\Omega}\right)$ is the weighting function defined over the reference element.

Furthermore, we note that the physical element has corresponding linear element $\overline{\Omega}^e$, defined by an affine mapping $\overline{\mathbf{x}}^e : \hat{\Omega} \rightarrow \overline{\Omega}^e$. We then define a warping function $\mathbf{F}$ that maps the linear element to the curvilinear element, $\mathbf{F} : \overline{\Omega}^e \rightarrow \Omega^e$, and we note that the mapping $\mathbf{x}^e$ is simply the compositions of these two mappings, $\mathbf{x}^e = \mathbf{F} \circ \overline{\mathbf{x}}^e$. We can then derive error bounds for the element Ω_e in terms of the mapping $\mathbf{F}$.

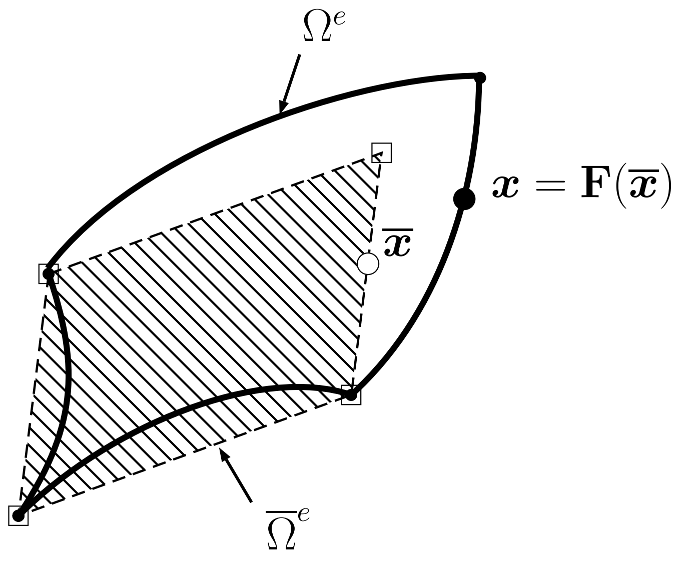

Ω^e
$\boldsymbol{x} = \mathbf{F}(\overline{\boldsymbol{x}})$
$\overline{\boldsymbol{x}}$
$\overline{\Omega}^e$

Figure 5.5: Mapping $\mathbf{F}$ from the linear element $\overline{\Omega}^e$ to the curved element Ω^e.

Theorem 5.3.1. *Over the physical element Ω^e, there exists a constant c_s independent of the mesh size h, the warping function $\mathbf{F}$, and the weighting function w, such that for all $u \in H^{p+1}(\Omega^e)$, there exists an approximation function $u_h \in \mathcal{S}_p^h$ satisfying the estimate:*

$$||u - u_h||_{L^2(\Omega^e)} \leq c_s h_e^{p+1} c_d(\mathbf{F}) c_v(\mathbf{F}, w, u)$$

where:

$$c_d(\mathbf{F}) = \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{1/2}\right|\right|_{L^\infty\left(\overline{\Omega}^e\right)} \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{-1/2}\right|\right|_{L^\infty(\Omega^e)}$$

and:

$$c_v(\mathbf{F}, w, u) = \sum_{k=0}^{p+1} \sum_{j=0}^{k} c_\alpha(\mathbf{F}, j, k) \left|\left|\frac{1}{w}\right|\right|_{L^\infty\left(\hat{\Omega}\right)} \left|\left|\nabla^{p+1-k} w\right|\right|_{L^\infty\left(\overline{\Omega}^e\right)} |u|_{H^j(\Omega^e)}$$

wherein:

$$c_\alpha(\mathbf{F}, j, k) = \sum_{\substack{i_1 + i_2 + \ldots + i_k = j \\ i_1 + 2i_2 + \ldots + k i_k = k}} \left(||\boldsymbol{\nabla}\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)}\right)^{i_1} \left(||\boldsymbol{\nabla}^2\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)}\right)^{i_2} \ldots \left(||\boldsymbol{\nabla}^k\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)}\right)^{i_k}$$

Proof. By definition, we have:

$$\int_{\Omega^e} u^2 d\boldsymbol{x} = \int_{\overline{\Omega}^e} (u \circ \mathbf{F})^2 \det \boldsymbol{\nabla}\mathbf{F} d\overline{\boldsymbol{x}}$$

Therefore, $||u||_{L^2(\Omega^e)} = \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{1/2} u \circ \mathbf{F}\right|\right|_{L^2\left(\overline{\Omega}^e\right)}$, and as a consequence, we can bound:

$$||u||_{L^2(\Omega^e)} \leq \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{1/2}\right|\right|_{L^\infty\left(\overline{\Omega}^e\right)} ||u \circ \mathbf{F}||_{L^2\left(\overline{\Omega}^e\right)}$$

Consequently for each approximation function $u_h \in \mathcal{S}_p^h$, it follows that:

$$||u - u_h||_{L^2(\Omega^e)} \leq \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{1/2}\right|\right|_{L^\infty\left(\overline{\Omega}^e\right)} ||u \circ \mathbf{F} - u_h \circ \mathbf{F}||_{L^2\left(\overline{\Omega}^e\right)}$$

Since the weighting function is bounded from above and below, it further follows that

$$||u - u_h||_{L^2(\Omega^e)} \leq \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{1/2}\right|\right|_{L^\infty\left(\overline{\Omega}^e\right)} \left|\left|\frac{1}{w}\right|\right|_{L^\infty\left(\hat{\Omega}\right)} ||w(u \circ \mathbf{F}) - w(u_h \circ \mathbf{F})||_{L^2\left(\overline{\Omega}^e\right)}$$

Now, we note that by construction of u_h, the function $w(u_h \circ \mathbf{x}^e)$ is an arbitrary polynomial of degree p. Furthermore, we recognize that $w(u_h \circ \mathbf{x}^e) = w(u_h \circ \mathbf{F} \circ \overline{\mathbf{x}}^e)$. Thus, because the mapping $\overline{\mathbf{x}}^e$ is purely affine, the function $w\left(u_h \circ \mathbf{F}\right)$ is similarly an arbitrary polynomial of degree p. Therefore, by the classical Bramble–Hilbert lemma, we may select u_h such that the following inquality holds:

$$||w\left(u \circ \mathbf{F}\right) - w\left(u_h \circ \mathbf{F}\right)||_{L^2\left(\overline{\Omega}^e\right)} \leq c_1 h^{p+1} \left|w\left(u \circ \mathbf{F}\right)\right|_{H^{p+1}\left(\overline{\Omega}^e\right)}$$

where $c_1 = c_1\left(p, \sigma_e\right)$ is a constant that depends only on the polynomial degree p and shape regularity σ_e of the linear element $\overline{\Omega}^e$. We must now bound the seminorm $\left|w\left(u \circ \mathbf{F}\right)\right|_{H^{p+1}\left(\overline{\Omega}^e\right)}$ appearing in the above estimate by an analagous norm over the physical element Ω^e. To do so, we first recognize that:

$$\left|w\left(u \circ \mathbf{F}\right)\right|_{H^{p+1}\left(\overline{\Omega}^e\right)} \leq \sum_{k=0}^{p+1} \left|\left|\boldsymbol{\nabla}^{p+1-k} w\right|\right|_{L^\infty\left(\overline{\Omega}^e\right)} \left|u \circ \mathbf{F}\right|_{H^k\left(\overline{\Omega}^e\right)}$$

It remains to bound the seminorms $\left|u \circ \mathbf{F}\right|_{H^k\left(\overline{\Omega}^e\right)}$. We may easily obtain control of the H^1-seminorm using the estimate:

$$\begin{aligned}
\left|u \circ \mathbf{F}\right|_{H^1\left(\overline{\Omega}^e\right)} &= \left(\int_{\overline{\Omega}^e} \boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\left(u \circ \mathbf{F}\right) \cdot \boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\left(u \circ \mathbf{F}\right) d\overline{\boldsymbol{x}}\right)^{1/2} \\
&= \left(\int_{\Omega^e} \boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\mathbf{F}\boldsymbol{\nabla}_{\boldsymbol{x}} u \cdot \boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\mathbf{F}\boldsymbol{\nabla}_{\boldsymbol{x}} u \det \boldsymbol{\nabla}\mathbf{F}^{-1} d\boldsymbol{x}\right)^{1/2} \\
&\leq ||\boldsymbol{\nabla}\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)} \left|\left|\det \boldsymbol{\nabla}\mathbf{F}^{-1/2}\right|\right|_{L^\infty\left(\Omega^e\right)} |u|_{H^1\left(\Omega^e\right)}
\end{aligned}$$

To obtain control of the higher–order seminorms, we simply recurse on the previous estimate, as is done in [6], resulting in the estimate:

$$\left|u \circ \mathbf{F}\right|_{H^k\left(\overline{\Omega}^e\right)} \leq c_2 ||\boldsymbol{\nabla}\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)} \sum_{j=0}^{k} c_\alpha\left(j, k\right)\left(\boldsymbol{\nabla}\mathbf{F}\right) |u|_{H^j\left(\Omega^e\right)}$$

where $c_2 = c_2\left(p, \sigma_e\right)$ is again a constant that depends only on the polynomial degree p of the basis and the shape regularity of $\overline{\Omega}^e$. Thus, letting $c_s = c_1 c_2$, we arrive at the bound presented in Theorem 5.3.1. $\square$

Theorem 5.3.1 gives valuable theoretical insight into the convergence behavior of rational curvilinear Bernstein–Bézier elements, as it clearly delineates the effect of the linear shape quality (c_s) and curvilinear shape quality (c_d and c_v) on the interpolation error bounds. However, its utility is somewhat limited as the bound is given in terms of the warping function $\mathbf{F}$, whereas Bernstein–Bézier elements are defined by the mapping $\mathbf{x}^e$. Since the mapping $\mathbf{x}^e$ is given explicitly by the control points and weights, it is desirable to derive sufficient conditions based on this mapping instead. We begin by deriving bounds on the gradients $||\boldsymbol{\nabla}^k_{\overline{\boldsymbol{x}}}\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)}$ in terms of the gradients $||\boldsymbol{\nabla}^k_{\boldsymbol{\xi}}\mathbf{x}^e||_{L^\infty\left(\hat{\Omega}\right)}$ (Theorem 5.3.2), as well as bounds on the gradients $||\boldsymbol{\nabla}^k_{\overline{\boldsymbol{x}}}w||_{L^\infty\left(\overline{\Omega}^e\right)}$ in terms of the gradients $||\boldsymbol{\nabla}^k_{\boldsymbol{\xi}}w||_{L^\infty\left(\hat{\Omega}\right)}$ (Theorem 5.3.3). We then use the results of these theorems to prove Theorem I.

Theorem 5.3.2. *Given the mappings* $\mathbf{F}$ *and* $\mathbf{x}^e$*, there exist some constant* $c_3 = c_s\left(\sigma_e\right)$*, dependent only on the shape regularity of* $\overline{\Omega}^e$*, such that:*

$$||\boldsymbol{\nabla}^k\mathbf{F}||_{L^\infty\left(\overline{\Omega}^e\right)} \leq c_3^k\left|\left|\boldsymbol{\nabla}^k\mathbf{x}^e\right|\right|_{L^\infty\left(\hat{\Omega}\right)}\left(\frac{1}{h_e}\right)^k$$

Proof. We first recognize that the warping function $\mathbf{F}$ can be written as a composition of $\mathbf{x}^e$ and $\overline{\mathbf{x}}^{e-1}$, viz:

$$\mathbf{F}\left(\overline{\boldsymbol{x}}\right) = \mathbf{x}^e\left(\overline{\mathbf{x}}^{e-1}\left(\overline{\boldsymbol{x}}\right)\right)$$

Thus, the gradient of $\mathbf{F}$ can be written:

$$\boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\mathbf{F} = \left[\boldsymbol{\nabla}_{\boldsymbol{\xi}}\mathbf{x}^e\right]\left[\boldsymbol{\nabla}_{\boldsymbol{\xi}}\overline{\mathbf{x}}^e\right]^{-T} \tag{5.3}$$

We note that since $\overline{\Omega}^e$ is a linear element, the gradient $\boldsymbol{\nabla}_{\boldsymbol{\xi}}\overline{\mathbf{x}}^e$ is constant and $||\boldsymbol{\nabla}_{\boldsymbol{\xi}}\overline{\mathbf{x}}^e||_{L^\infty\left(\hat{\Omega}\right)}$ is bounded from below by the radius of the element incircle, ρ_e, viz.:

$$\rho_e \leq ||\boldsymbol{\nabla}_{\boldsymbol{\xi}}\overline{\mathbf{x}}^e||_{L^\infty\left(\hat{\Omega}\right)} \tag{5.4}$$

and we can similarly bound the norm of the inverse mapping by:

$$||\boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\left(\overline{\mathbf{x}}^{e-1}\right)||_{L^\infty\left(\hat{\Omega}\right)} \leq \frac{1}{\rho_e} \tag{5.5}$$

Now, let us define a constant $c_3 = c_3(\sigma_e)$, which is independent of mesh size h_e, but is dependent on the shape regularity of $\overline{\Omega}^e$. Furthermore, let c_3 satisfy the inequality:

$$c_3 \geq \sigma_e = \frac{h_e}{\rho_e}$$

Then, we can rewrite Eq. (5.5) as:

$$\left|\left|\boldsymbol{\nabla}_{\overline{\boldsymbol{x}}}\left(\overline{\mathbf{x}}^{e-1}\right)\right|\right|_{L^{\infty}(\hat{\Omega})} \leq c_3 \frac{1}{h_e}$$

Then, from Eq. (5.3) and Eq. (5.4), we can bound the norm of $\boldsymbol{\nabla}\mathbf{F}$ by:

$$||\boldsymbol{\nabla}\mathbf{F}||_{L^{\infty}\left(\overline{\Omega}^{e}\right)} \leq c_3 \, ||\boldsymbol{\nabla}_{\boldsymbol{\xi}}\mathbf{x}^{e}||_{L^{\infty}(\hat{\Omega})} \left(\frac{1}{h_e}\right)$$

Recursing on this process, we can bound the magnitude of the k^{th} total derivative by:

$$\left|\left|\boldsymbol{\nabla}^{k}\mathbf{F}\right|\right|_{L^{\infty}\left(\overline{\Omega}^{e}\right)} \leq c_3^{k} \left|\left|\boldsymbol{\nabla}^{k}\mathbf{x}^{e}\right|\right|_{L^{\infty}(\hat{\Omega})} \left(\frac{1}{h_e}\right)^{k}$$

□

Theorem 5.3.3. *Given the weighting function w, there exist some constant $c_3 = c_3\left(\sigma_e\right)$, dependent only on the shape regularity of $\overline{\Omega}^{e}$, such that:*

$$\left|\left|\boldsymbol{\nabla}^{k}_{\overline{\boldsymbol{x}}} w\right|\right|_{L^{\infty}\left(\overline{\Omega}^{e}\right)} \leq c_3^{k} \left|\left|\boldsymbol{\nabla}^{k}_{\boldsymbol{\xi}} w\right|\right|_{L^{\infty}(\hat{\Omega})} \left(\frac{1}{h_e}\right)^{k}$$

Proof. The proof of Theorem 5.3.3 is identical to the proof for Theorem 5.3.2. □

Theorem I. *Over the physical element Ω^e, there exists a constant C_{shape} independent of the mesh size h, the mapping $\mathbf{x}^e$, and the weighting function w, such that for all $u \in H^{p+1}(\Omega^e)$, there exists an approximation function $u_h \in \mathcal{S}_p^h$ satisfying the estimate:*

$$||u - u_h||_{L^2(\Omega^e)} \leq C_{shape} h_e^{p+1} C_{det}(\mathbf{x}^e) C_{var}(\mathbf{x}^e, w, u)$$

where:

$$C_{det}(\mathbf{x}^e) = \left|\left|\det \boldsymbol{\nabla}\mathbf{x}^{e1/2}\right|\right|_{L^\infty(\hat{\Omega})} \left|\left|\det \boldsymbol{\nabla}\mathbf{x}^{e-1/2}\right|\right|_{L^\infty(\Omega^e)}$$

and:

$$C_{var}(\mathbf{x}^e, w, u) = \sum_{k=0}^{p+1} \sum_{j=0}^{k} \alpha_{j,k}(\mathbf{x}^e) \left|\left|\frac{1}{w}\right|\right|_{L^\infty(\hat{\Omega})} \left(\frac{\left|\left|\nabla_{\boldsymbol{\xi}}^{p+1-k} w\right|\right|_{L^\infty(\hat{\Omega})}}{h_e^{p+1-k}} \right) |u|_{H^j(\Omega^e)}$$

wherein:

$$\alpha_{j,k}(\mathbf{x}^e) \leq \sum_{\substack{i_1+i_2+\ldots+i_k=j \\ i_1+2i_2+\ldots+ki_k=k}} \left(\frac{||\boldsymbol{\nabla}\mathbf{x}^e||_{L^\infty(\hat{\Omega})}}{h_e} \right)^{i_1} \left(\frac{||\boldsymbol{\nabla}^2\mathbf{x}^e||_{L^\infty(\hat{\Omega})}}{h_e^2} \right)^{i_2} \cdots \left(\frac{||\boldsymbol{\nabla}^k\mathbf{x}^e||_{L^\infty(\hat{\Omega})}}{h_e^k} \right)^{i_k}$$

Proof. Proving Theorem I amounts to simply bounding the constants c_d and c_v from Theorem 5.3.1 by analagous constants in terms of the mapping $\mathbf{x}^e$. We begin by recognizing that $\mathbf{x}^e = \mathbf{F} \circ \overline{\mathbf{x}}^e$, and by extension $\mathbf{F} = \mathbf{x}^e \circ \overline{\mathbf{x}}^{e-1}$. Thus, we can write the gradient of the mapping $\mathbf{F}$ as $\boldsymbol{\nabla}\mathbf{F} = [\boldsymbol{\nabla}\mathbf{x}^e] [\boldsymbol{\nabla}\overline{\mathbf{x}}^e]^{-T}$ and as a result, the determinant of $\boldsymbol{\nabla}\mathbf{F}$ can be written as:

$$\det \boldsymbol{\nabla}\mathbf{F} = \frac{\det \boldsymbol{\nabla}\mathbf{x}^e}{\det \boldsymbol{\nabla}\overline{\mathbf{x}}^e}$$

allowing us to rewrite $c_d(\mathbf{F})$ in terms of $\overline{\mathbf{x}}^e$ and $\mathbf{x}^e$ as:

$$c_d(\mathbf{F}) = C_{det}(\overline{\mathbf{x}}^e, \mathbf{x}^e) = \left|\left| \left(\frac{\det \boldsymbol{\nabla}\mathbf{x}^e}{\det \boldsymbol{\nabla}\overline{\mathbf{x}}^e} \right)^{1/2} \right|\right|_{L^\infty(\hat{\Omega})} \left|\left| \left(\frac{\det \boldsymbol{\nabla}\overline{\mathbf{x}}^e}{\det \boldsymbol{\nabla}\mathbf{x}^e} \right)^{1/2} \right|\right|_{L^\infty(\hat{\Omega})}$$

Then, recognizing that $\det \boldsymbol{\nabla}\overline{\mathbf{x}}^e$ is a constant, the above equations simplifies immediately to:

$$c_d(\mathbf{F}) = C_{det}(\mathbf{x}^e) = \left|\left|\det \boldsymbol{\nabla}_{\boldsymbol{\xi}}\mathbf{x}^{e1/2}\right|\right|_{L^\infty(\hat{\Omega})} \left|\left|\det \boldsymbol{\nabla}_{\boldsymbol{x}}\mathbf{x}^{e-1/2}\right|\right|_{L^\infty(\Omega^e)}$$

It remains then to bound the constant c_v by a bound in terms of $\mathbf{x}^e$. We begin by substituting the results of Theorem 5.3.2 into the expression $c_\alpha(\mathbf{F}, j, k)$, which yields.

$$c_\alpha(\mathbf{F}, j, k) \leq \sum_{\substack{i_1+i_2+...+i_k=j \\ i_1+2i_2+...+ki_k=k}} \left(c_3 \frac{||\boldsymbol{\nabla}\mathbf{x}^e||_{L^\infty(\hat{\Omega})}}{h_e} \right)^{i_1} \left(c_3^2 \frac{||\boldsymbol{\nabla}^2\mathbf{x}^e||_{L^\infty(\hat{\Omega})}}{h_e^2} \right)^{i_2} ...$$
$$... \left(c_3^k \frac{||\boldsymbol{\nabla}^k\mathbf{x}^e||_{L^\infty(\hat{\Omega})}}{h_e^k} \right)^{i_k}$$

Then, recognizing that $c_3^{i_1} c_3^{2i_2} ... c_3^{ki_k} = c_3^k$, we can factor out the constant c_3, to arrive at the bound:

$$c_\alpha(\mathbf{F}, j, k) \leq c_3^k \alpha_{j,k}(\mathbf{x}^e) \tag{5.6}$$

Finally, employing the results of Eq. (5.6) along with Theorem 5.3.3, we can bound $c_v(\mathbf{F}, w, u)$ by:

$$c_v(\mathbf{F}, w, u) \leq \sum_{k=0}^{p+1} \sum_{j=0}^{k} c_3^k \alpha_{j,k}(\mathbf{x}^e) \left|\left|\frac{1}{w}\right|\right|_{L^\infty(\hat{\Omega})} c_3^{p+1-k} \left(\frac{\left|\left|\nabla_{\boldsymbol{\xi}}^{p+1-k} w\right|\right|_{L^\infty(\hat{\Omega})}}{h_e^{p+1-k}} \right) |u|_{H^j(\Omega^e)}$$

which we can simplify to:

$$c_v(\mathbf{F}, w, u) \leq c_3^{p+1} C_{var}(\mathbf{x}^e, w, u)$$

Thus, we arrive at the results of Theorem I, where $C_{shape} = c_s c_3^{p+1}$. □

The astute reader will recognize that the above results can be simplified even further. Indeed, we have left Theorem I in its current form intentionally, as it clearly demonstrates the dependence on h_e in each part of the error bound. This clear dependence on h_e will prove useful in the next section. However, we similarly note that at times it is useful develop size invariant error bounds. We further simplify the results of Theorem I to achieve error bounds that are independent of the element diameter h_e.

Corollary 5.3.4. *Over the physical element Ω^e, there exists a constant C_{shape} independent of the mesh size h, the mapping $\mathbf{x}^e$, and the weighting function w, such that for all $u \in H^{p+1}(\Omega^e)$, there exists an approximation function $u_h \in \mathcal{S}_p^h$ satisfying the estimate:*

$$||u - u_h||_{L^2(\Omega^e)} \leq C_{shape} C_{det}(\mathbf{x}^e) C'_{var}(\mathbf{x}^e, w, u)$$

where:

$$C_{det}\left(\mathbf{x}^e\right) = \left|\left|\det \boldsymbol{\nabla}\mathbf{x}^{e1/2}\right|\right|_{L^\infty(\hat{\Omega})}\left|\left|\det \boldsymbol{\nabla}\mathbf{x}^{e-1/2}\right|\right|_{L^\infty(\Omega^e)}$$

and:

$$C'_{var}(\mathbf{x}^e, w, u) = \sum_{k=0}^{p+1}\sum_{j=0}^{k}\alpha'_{j,k}(\mathbf{x}^e)\left|\left|\frac{1}{w}\right|\right|_{L^\infty(\hat{\Omega})}\left|\left|\nabla_{\boldsymbol{\xi}}^{p+1-k}w\right|\right|_{L^\infty(\hat{\Omega})}|u|_{H^j(\Omega^e)}$$

wherein:

$$\alpha'_{j,k}(\mathbf{x}^e) = \sum_{\substack{i_1+i_2+\ldots+i_k=j \\ i_1+2i_2+\ldots+ki_k=k}}\left(||\boldsymbol{\nabla}\mathbf{x}^e||_{L^\infty(\hat{\Omega})}\right)^{i_1}\left(||\boldsymbol{\nabla}^2\mathbf{x}^e||_{L^\infty(\hat{\Omega})}\right)^{i_2}\ldots\left(\left|\left|\boldsymbol{\nabla}^k\mathbf{x}^e\right|\right|_{L^\infty(\hat{\Omega})}\right)^{i_k}$$

5.4 Regular Families of Curvilinear Elements

The results of Section 5.3 are convenient, as they clearly delineate the effect of element shape on interpolation error bounds. Namely, C_{shape} quantifies the contribution of **linear** element shape regularity, while C_{det} and C_{var} quantify the contribution of the higher–order mapping $\mathbf{x}^e$ and weighting function w. By writing the error bounds in terms of the mapping $\mathbf{x}^e$ we may work directly with the Bernstein basis functions defined over the reference element $\hat{\Omega}$, and the Bézier control points in physical space. Furthermore, the results of Theorem I clearly delineate the dependence of the interpolation error bounds on the higher–order derivatives of the mapping in terms of the element size h_e. In light of this, we desire to develop sufficient conditions for guaranteeing that rational Bernstein–Bézier elements will preserve optimal convergence rates under refinement. That is, we desire to develop sufficient conditions for guaranteeing that:

$$||u - u_h||_{L^2(\Omega^e)} < Ch_e^{p+1}$$

where C is some constant independent of element size h_e and the mapping $\mathbf{x}^e$. Indeed, as discussed in Section 5.2, such sufficient conditions have been developed for finite elements based on **polynomial** mappings. In this section, we develop analogous sufficient conditions for **rational** Bernstein–Bézier elements.

Theorem 5.4.1. *Let there exist some constant, c_w, dependent on p, but not on h_e such that:*

$$\left|\left|D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} w\right|\right|_{L^{\infty}(\hat{\Omega})} \leq c_w h_e^{|\boldsymbol{\alpha}|} \quad \forall \boldsymbol{\alpha} : |\boldsymbol{\alpha}| \leq p+1 \tag{5.7}$$

Then, there exists a constant, c_{inv}, such that:

$$\left|\left|D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\left(\frac{1}{w}\right)\right|\right|_{L^{\infty}(\hat{\Omega})} \leq c_{inv} h_e^{|\boldsymbol{\alpha}|}$$

Proof. By application of the multivariate Faà di Bruno's formula [19], we write the derivative of the inverse of the weighting function as:

$$D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\left(\frac{1}{w}\right) = \sum_{1 \leq |\mathbf{s}| \leq |\boldsymbol{\alpha}|} (-1)^{|\mathbf{s}|} \frac{|\mathbf{s}|!}{w^{1+|\mathbf{s}|}} \sum_{p(\boldsymbol{\alpha},\mathbf{s})} \boldsymbol{\alpha}! \prod_{j=1}^{|\boldsymbol{\alpha}|} \frac{\left(D_{\boldsymbol{\xi}}^{\boldsymbol{\ell}_j} w\right)^{k_j}}{k_j! \left(\boldsymbol{\ell}_j!\right)^{k_j}}$$

where $p(\boldsymbol{\alpha}, \mathbf{s})$ denotes the set:

$$p(\boldsymbol{\alpha}, \mathbf{s}) = \bigcup_{s=1}^{|\boldsymbol{\alpha}|} p_s(\boldsymbol{\alpha}, \mathbf{s})$$

$$p_s(\boldsymbol{\alpha}, \mathbf{s}) = \Big\{ \left(k_1, ..., k_{|\boldsymbol{\alpha}|}; \boldsymbol{\ell}_1, ..., \boldsymbol{\ell}_{|\boldsymbol{\alpha}|}\right) : \\ 0 < k_i, \mathbf{0} < \boldsymbol{\ell}_1 < ... < \boldsymbol{\ell}_s, \\ \sum_{i=1}^{s} k_i = \mathbf{s} \text{ and } \sum_{i=1}^{s} k_i \boldsymbol{\ell}_i = \boldsymbol{\alpha} \Big\}$$

Then, by Eq. (5.7), we have:

$$\left|\left|D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\left(\frac{1}{w}\right)\right|\right|_{L^{\infty}(\Omega)} \leq \sum_{1 \leq |\mathbf{s}| \leq |\boldsymbol{\alpha}|} \frac{|\mathbf{s}|!}{w^{1+|\mathbf{s}|}} \sum_{p(\boldsymbol{\alpha},\mathbf{s})} \boldsymbol{\alpha}! \prod_{j=1}^{|\boldsymbol{\alpha}|} \frac{\left(c_w h_e^{|\boldsymbol{\ell}_j|}\right)^{k_j}}{k_j! \left(\boldsymbol{\ell}_j!\right)^{k_j}}$$

Rearranging, and factoring out terms not dependent on h_e yields:

$$\left|\left|D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \frac{1}{w}\right|\right|_{L^{\infty}(\Omega)} \leq \sum_{1 \leq |\mathbf{s}| \leq |\boldsymbol{\alpha}|} \sum_{p(\boldsymbol{\alpha},\mathbf{s})} c_1(\boldsymbol{\alpha}, \mathbf{s}) \prod_{j=1}^{|\boldsymbol{\alpha}|} \left(h_e^{|\boldsymbol{\ell}_{\mathbf{j}}|}\right)^{k_j} \tag{5.8}$$

where:

$$c_1(\boldsymbol{\alpha}, \mathbf{s}) = \frac{|\mathbf{s}|! \boldsymbol{\alpha}!}{w^{|\mathbf{s}|+1}} \prod_{j=1}^{|\boldsymbol{\alpha}|} \frac{c_w^{k_j}}{k_j! \left(\boldsymbol{\ell}_l!\right)^{k_j}}$$

Finally, noting that:

$$\prod_{j=1}^{|\boldsymbol{\alpha}|} \left(h_e^{|\boldsymbol{\ell}_j|}\right)^{k_j} = h_e^{\left(\sum_{j=1}^{|\boldsymbol{\alpha}|} k_j |\boldsymbol{\ell}_j|\right)}$$

and that by construction of $p(\boldsymbol{\alpha}, \mathbf{s})$:

$$\sum_{j=1}^{|\boldsymbol{\alpha}|} k_j |\boldsymbol{\ell}_j| = |\boldsymbol{\alpha}|$$

Inequality (5.8) becomes:

$$\left|\left| D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \frac{1}{w} \right|\right|_{L^\infty(\hat{\Omega})} \leq \sum_{1 \leq |\mathbf{s}| \leq |\boldsymbol{\alpha}|} \sum_{p(\boldsymbol{\alpha}, \mathbf{s})} c_1(\boldsymbol{\alpha}, \mathbf{s}) \, h_e^{|\boldsymbol{\alpha}|}$$

and the results of Lemma 5.4.1 follows directly, where $c_{inv} = \sum\limits_{1 \leq |\mathbf{s}| \leq |\boldsymbol{\alpha}|} \sum\limits_{p(\boldsymbol{\alpha}, \mathbf{s})} c_1(\boldsymbol{\alpha}, \mathbf{s})$. $\square$

Theorem 5.4.2. *Let us assume there exists a constant C_{proj} not dependent on h_e such that:*

$$\left|\left| D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \widetilde{\mathbf{x}}^e \right|\right|_{L^\infty(\hat{\Omega})} \leq C_{proj} h_e^{|\boldsymbol{\alpha}|} \quad \forall \boldsymbol{\alpha} : |\boldsymbol{\alpha}| = k, \quad k \leq p+1 \tag{5.9}$$

Then there exists a constant C_{grad}, such that:

$$\left|\left| \boldsymbol{\nabla}^k \mathbf{x}^e \right|\right|_{L^\infty(\hat{\Omega})} \leq C_{grad} h_e^k$$

$$\left|\left| \boldsymbol{\nabla}^k w \right|\right|_{L^\infty(\hat{\Omega})} \leq C_{grad} h_e^k$$

Proof. We immediately recognize that since $w = (\widetilde{\mathbf{x}}^e)_{d+1}$, the inequality

$$\left|\left| D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} w \right|\right|_{L^\infty(\hat{\Omega})} \leq C_{proj} h_e^k \tag{5.10}$$

holds by definition. It remains then to show that the bound on the derivatives of the projective mapping $\widetilde{\mathbf{x}}^e$ imply a bound on the derivatives of the physical mapping $\mathbf{x}^e$. We begin by writing the derivative $D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \mathbf{x}^e$ by the multi-variate product rule as:

$$D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \mathbf{x}^e = D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \left(\frac{\widetilde{\mathbf{x}}^e}{w} \right) = \sum_{\mathbf{k} \in I^{\boldsymbol{\alpha}}} \binom{\boldsymbol{\alpha}}{\mathbf{k}} D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha} - \mathbf{k}} \left(\widetilde{\mathbf{x}}^e \right) D_{\boldsymbol{\xi}}^{\mathbf{k}} \left(\frac{1}{w} \right) \tag{5.11}$$

Then, from Eq. (5.10) and Theorem 5.4.1, it is readily seen that:

$$\left|\left| D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \left(\frac{1}{w} \right) \right|\right|_{L^\infty(\hat{\Omega})} \leq c_{inv} h_e^{|\boldsymbol{\alpha}|} \quad \forall \; \boldsymbol{\alpha} : |\boldsymbol{\alpha}| = k$$

Taking this result, along with Eq. (5.9), and substituting into Eq. (5.11) yields:

$$\left|\left| D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}} \mathbf{x}^e \right|\right|_{L^\infty(\hat{\Omega})} \leq \sum_{\mathbf{k} \in I^{\boldsymbol{\alpha}}} \binom{\boldsymbol{\alpha}}{\mathbf{k}} C_{proj} h_e^{|\boldsymbol{\alpha}| - |\mathbf{k}|} c_{inv} h_e^{|\mathbf{k}|}$$

which reduces immediately to:

$$\left|\left|D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\mathbf{x}^e\right|\right|_{L^\infty(\hat{\Omega})} \leq C_{proj}c_{inv}h_e^{|\boldsymbol{\alpha}|} \quad \forall\ \boldsymbol{\alpha} : |\boldsymbol{\alpha}| = k \tag{5.12}$$

Finally, we recognize that if Eq. (5.10) and Eq. (5.12) hold for every derivative of order k, then there exists some $C_{grad} = C_{grad}\left(C_{proj}, c_{inv}\right)$ such that:

$$\left|\left|\boldsymbol{\nabla}_{\boldsymbol{\xi}}^k w\right|\right|_{L^\infty(\hat{\Omega})} \leq C_{grad}h_e^k$$

$$\left|\left|\boldsymbol{\nabla}_{\boldsymbol{\xi}}^k \mathbf{x}^e\right|\right|_{L^\infty(\hat{\Omega})} \leq C_{grad}h_e^k$$

which are exactly the results of Theorem 5.4.2 that we set out to prove. □

Theorem II. *For a rational Bernstein–Bézier element Ω^e of degree p with corresponding projective element $\widetilde{\Omega}^e$, let the following conditions hold for the mappings $\mathbf{x}^e$ and $\widetilde{\mathbf{x}}^e$:*

Cond. (II.1) *There exists a constant C_{max} independent of the element size h_e, such that:*

$$\left|\left|\det \boldsymbol{\nabla}\mathbf{x}^{e1/2}\right|\right|_{L^\infty(\hat{\Omega})}\left|\left|\det \boldsymbol{\nabla}\mathbf{x}^{e-1/2}\right|\right|_{L^\infty(\Omega^e)} \leq C_{max}$$

Cond. (II.2) *There exists a constant C_{proj}, independent of the element size h_e, such that:*

$$\left|\left|D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e\right|\right|_{L^\infty(\hat{\Omega})} \leq C_{proj}h_e^{|\boldsymbol{\alpha}|} \quad \forall\boldsymbol{\alpha} : |\boldsymbol{\alpha}| = k,\ k \leq p+1$$

Then, there exist a constant C independent of the element size h_e such that:

$$||u - u_h||_{L^2(\Omega^e)} \leq Ch_e^{p+1}$$

We further say the rational Bernstein–Bézier element belongs to a ***regular family of curvilinear elements****, and will exhibit similar convergence rates to traditional affine finite elements.*

Proof. From Theorem I, we immediately see that if Cond. (II.1) holds, then $C_{det} \leq C_{max}$. Next, if Cond. (II.2) holds, we can bound C_{var} using the results of Theorem 5.4.2 by:

$$C_{var}(\mathbf{x}^e, w, u) \leq \sum_{k=0}^{p+1} \sum_{j=0}^{k} \alpha_{j,k}(\mathbf{x}^e) \left|\left| \frac{1}{w} \right|\right|_{L^\infty(\hat{\Omega})} \left(\frac{C_{grad} h_e^{p+1-k}}{h_e^{p+1-k}} \right) |u|_{H^j(\Omega^e)}$$

wherein:

$$\alpha_{j,k}(\mathbf{x}^e) \leq \sum_{\substack{i_1+i_2+...+i_k=j \\ i_1+2i_2+...+ki_k=k}} \left(\frac{C_{grad} h_e}{h_e} \right)^{i_1} \left(\frac{C_{grad} h_e^2}{h_e^2} \right)^{i_2} \cdots \left(\frac{C_{grad} h_e^k}{h_e^k} \right)^{i_k}$$

which reduces immediately to:

$$C_{var}(\mathbf{x}^e, w, u) \leq C_{grad} \left|\left| \frac{1}{w} \right|\right|_{L^\infty(\hat{\Omega})} \sum_{k=0}^{p+1} \sum_{j=0}^{k} \sum_{\substack{i_1+i_2+...+i_k=j \\ i_1+2i_2+...+ki_k=k}} C_{grad}^j |u|_{H^j(\Omega^e)}$$

Thus, we arrive at the results of Theorem II, wherein $C = C(C_{shape}, C_{max}, C_{grad}, w, u)$ is a constant independent of h_e. □

With Theorem II, we have derived a set of two sufficient conditions for guaranteeing that a rational Bernstein–Bézier element will exhibit convergence rates similar to traditional affine elements. Along with this, we have introduced the notion of a **regular family of curvilinear elements**. Specifically, we say that if Cond. II.1 and Cond. II.2 hold, then a rational Bernstein–Bézier element belongs to a regular family of curvilinear elements.

5.5 Computable Distortion Metrics for Rational Bernstein–Bézier Elements

With sufficient conditions for analysis suitability of rational Bernstein–Bézier discretizations established, it remains to develop easily computable distortion metrics that are sufficient to show that these conditions hold. In general, explicitly calculating distortion metrics for every element in a mesh is computationally expensive, particularly when the elements are of high polynomial degree p. As such, in this section, we present easily computable **bounds** on the element distortion. We begin by considering easily computable upper and lower bounds on the Jacobian determinant that may be used to bound the magnitude of C_{max}. We then present easily computable upper bounds on the derivatives of the mapping $\mathbf{x}^e$ so that we may bound the magnitude of c_{proj}.

5.5.1 Computable Bounds on the Jacobian Determinant

From Cond. II.1, we see that the ratio of the maximum and minimum values of the Jacobian determinant must be bounded from above. As noted in Section 5.2.4, this is equivalent to bounding the scaled Jacobian from below. However, efficiently computing bounds on the Jacobian determinant is not a straightforward task. As previously mentioned, efficient algorithms have been proposed for bounding the Jacobian determinant of polynomial elements [34], but no such bounds have been proposed for rational elements. We do note, however, that for every rational element $\Omega^e \in \mathbb{R}^d$, there is a corresponding projective element $\widetilde{\Omega}^e \in \mathbb{R}^{d+1}$ that is defined by a polynomial mapping. Naturally then, we seek a way to compute the Jacobian determinant of the physical element in terms of the projective element.

For an element in $\mathbb{R}^d$ the differential d-form ω is given by the d^{th} external product of the directional derivatives of $\boldsymbol{\nabla}\mathbf{x}^e$. That is:

$$\omega = \frac{\partial \mathbf{x}^e}{\partial \xi_1} \wedge ... \wedge \frac{\partial \mathbf{x}^e}{\partial \xi_d}$$

where $\wedge$ denotes the wedge product. For elements in $\mathbb{R}^2$ this yields a 2-form, which is a differential area dA, and for elements in $\mathbb{R}^3$ this yields a 3-form, which is a differential volume element dV. The Jacobian determinant then, is simply the Hodge dual of ω, viz:

$$\det\left[\mathbf{x}^e\right] = *\left(\omega\right)$$

where $*(\cdot)$ is the Hodge star operator. For a vector space $V \in \mathbb{R}^d$, the Hodge star operator denotes the duality between k-forms and $(d-k)$-forms. For elements in $\mathbb{R}^d$, ω is a d-form, and as such, $*(\omega)$ is a 0-form, which is a scalar. Geometrically, the Jacobian determinant gives the area of 2-forms in $\mathbb{R}^2$, and the volume of 3-forms in $\mathbb{R}^3$.

For projective elements in $\mathbb{R}^{d+1}$, we can define the differential d-form $\widetilde{\omega}$ as:

$$\widetilde{\omega} = \frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1} \wedge ... \wedge \frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_d}$$

As with the physical element, this yields area elements dA when $d = 2$ and volume elements dV when $d = 3$. For projective elements however, the d-form is defined in the $d+1$ dimensional vector space $V \in \mathbb{R}^{d+1}$. As such, the Hodge dual of d-forms in $\mathbb{R}^{d+1}$ are 1-forms, which are simply vectors. We illustrate these concepts for a rational Bernstein–Bézier triangle, shown in Fig. 5.6. The 2-form for the physical element is visualized by the blue parallelogram, and the Hodge dual of the 2-form gives the area of the parallelogram. The 2-form for the projective element is shown by the yellow parallelogram, and the Hodge dual of the 2-form is the corresponding normal vector, and this relation is denoted:

$$\mathbf{N} = * \left(\frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1} \wedge \frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_2} \right) = \frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1} \times \frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_2}$$

We take care to note that this vector $\mathbf{N}$ is **not** a unit normal. Rather, the magnitude of $\mathbf{N}$ is equal to the area of the corresponding 2-form.

Visualizing the differential forms for volumetric elements becomes untenable, as the projective elements are embedded in $\mathbb{R}^4$. However, for an arbitrary element in $\mathbb{R}^{d+1}$, we can write the vector $\mathbf{N}$ as:

$$\mathbf{N}(\boldsymbol{\xi}) = \sum_{i=1}^{d+1} (-1)^{d+1+i} M_i \mathbf{e}_i \tag{5.13}$$

wherein the minor M_i is the determinant of the $d \times d$ matrix formed by deleting the i^{th} row of $\boldsymbol{\nabla} \widetilde{\mathbf{x}}^e$. Additionally, let $\boldsymbol{x} = \mathbf{x}^e(\boldsymbol{\xi})$ denote a coordinate on the physical element, and let $\widetilde{\boldsymbol{x}} = \widetilde{\mathbf{x}}^e(\boldsymbol{\xi})$ denote the corresponding coordinate on the projective element. With this nomenclature established, we can present bounds on $\det[\boldsymbol{\nabla}\mathbf{x}^e]$ in terms of the polynomial mapping $\widetilde{\mathbf{x}}^e$.

Theorem 5.5.1. *For any rational Bernstein–Bézier element, the Jacobian determinant can be calculated by:*

$$\det[\boldsymbol{\nabla}\mathbf{x}^e] = \frac{\mathbf{N}(\boldsymbol{\xi}) \cdot \widetilde{\mathbf{x}}^e(\boldsymbol{\xi})}{w^{d+1}}$$

Proof. First, let us denote the first d components of the mapping $\widetilde{\mathbf{x}}^e$ as $\left[\widetilde{\mathbf{x}}^e\right]_d$. We then recognize that we desire to calculate the determinant of the matrix:

$$\det[\boldsymbol{\nabla}\mathbf{x}^e] = \det\left[\boldsymbol{\nabla}_{\boldsymbol{\xi}} \left(\frac{\left[\widetilde{\mathbf{x}}^e\right]_d}{w} \right)\right]$$

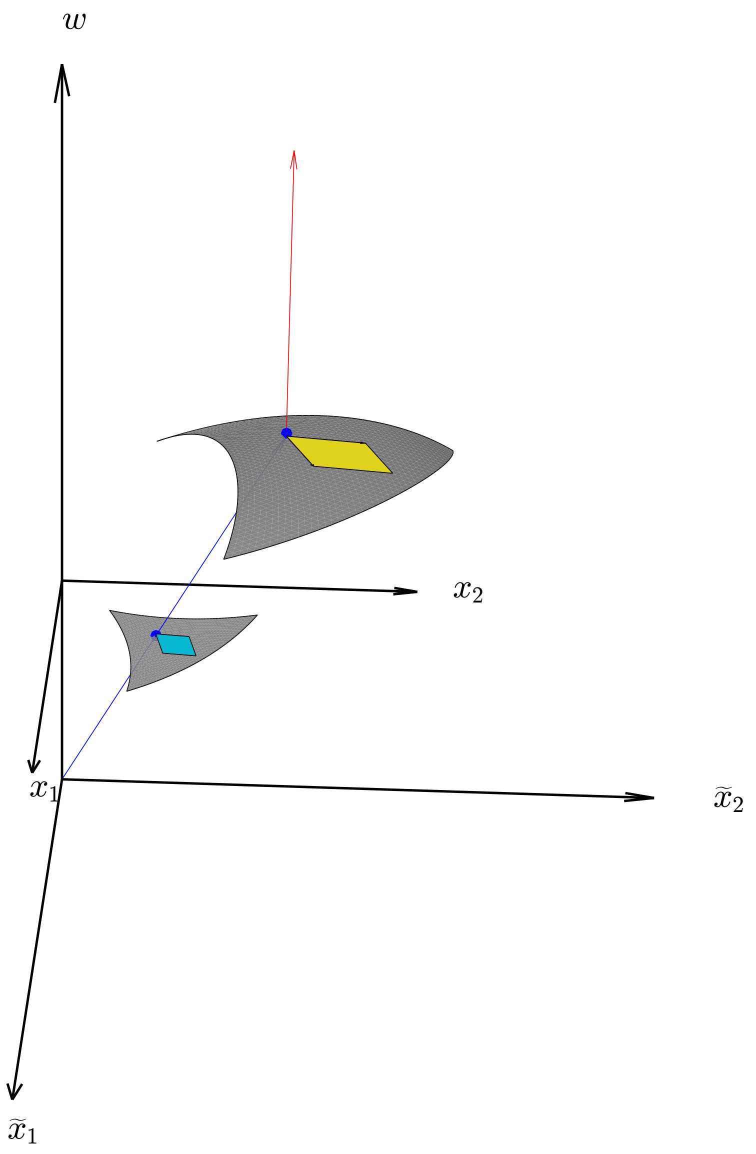


Figure 5.6: Differential 2-forms on a physical element and a projective element.

By the quotient rule we have:

$$\det\left[\boldsymbol{\nabla}\mathbf{x}^e\right]\det\left[\frac{1}{w}\left[\boldsymbol{\nabla}_{\boldsymbol{\xi}}\left[\widetilde{\mathbf{x}}^e\right]_d - \boldsymbol{x}[\boldsymbol{\nabla}_{\boldsymbol{\xi}}w]^T\right]\right]$$

and, because for any given $\boldsymbol{\xi} \in \hat{\Omega}$, w is some positive constant, we can factor out the weighting function to write:

$$\det\left[\boldsymbol{\nabla}\mathbf{x}^e\right] = \frac{1}{w^d}\det\left[\boldsymbol{\nabla}_{\boldsymbol{\xi}}\left[\widetilde{\mathbf{x}}^e\right]_d - \boldsymbol{x}[\boldsymbol{\nabla}_{\boldsymbol{\xi}}w]^T\right] \tag{5.14}$$

Then, by Eq. (5.13) we can write the vector $\mathbf{N}$ as:

$$\mathbf{N}\left(\boldsymbol{\xi}\right) = \sum_{i=1}^{d+1}(-1)^{d+1+i}\,M_i\mathbf{e}_i$$

and as a result, we can write the dot product $\mathbf{N}\cdot\boldsymbol{x}$ as:

$$\mathbf{N}\cdot\boldsymbol{x} = \sum_{i=1}^{d+1}(-1)^{d+1+i}\,M_i x_i = \det\begin{bmatrix}\boldsymbol{\nabla}_{\boldsymbol{\xi}}\left[\widetilde{\mathbf{x}}^e\right]_d & \boldsymbol{x}\\ & \\ \boldsymbol{\nabla}_{\boldsymbol{\xi}}w & 1\end{bmatrix}$$

which in turn can be written as:

$$\mathbf{N}\cdot\boldsymbol{x} = \det\begin{bmatrix}\boldsymbol{\nabla}_{\boldsymbol{\xi}}\left[\widetilde{\mathbf{x}}^e\right]_d & \boldsymbol{x}\\ & \\ \boldsymbol{\nabla}_{\boldsymbol{\xi}}w & 1\end{bmatrix} = \det\left[\boldsymbol{\nabla}_{\boldsymbol{\xi}}\left[\widetilde{\mathbf{x}}^e\right]_d - \boldsymbol{x}[\boldsymbol{\nabla}_{\boldsymbol{\xi}}w]^T\right] \tag{5.15}$$

Then, recognizing that $\mathbf{N}\cdot\boldsymbol{x} = \dfrac{\mathbf{N}\cdot\widetilde{\boldsymbol{x}}}{w}$, and substituting the results of Eq. (5.15) into Eq. (5.14), we get

$$\det\left[\boldsymbol{\nabla}\mathbf{x}^e\right] = \frac{1}{w^d}\frac{\mathbf{N}\cdot\widetilde{\boldsymbol{x}}}{w}$$

from which the results of Theorem 5.5.1 follow immediately. □

Conceptually, Theorem 5.5.1 can be thought of as projecting the vector $\mathbf{N}$ onto the $w = 1$ plane along the vector $\widetilde{\boldsymbol{x}}$, scaled by w^{d+1}. Alternatively, $\det\left[\boldsymbol{\nabla}\mathbf{x}^e\right]$ can be thought of as the apparent magnitude of $\widetilde{\omega}$ as seen by an observer at the origin. Either way, we recognize that we can compute the Jacobian determinant as the dot product of two vectors, normalized by the weighting function.

We recognize that for a rational Bernstein–Bézier element of degree p in $\mathbb{R}^d$, both $\widetilde{\boldsymbol{x}}$ and $\mathbf{N}$ will be vectors in $\mathbb{R}^{d+1}$. It is readily seen that the vector $\widetilde{\boldsymbol{x}}$ can be written in Bernstein–Bézier form, as the control points $\widetilde{\mathbf{P}}^b_{\mathbf{i}}$ are known. However, we note that the equation for the surface normal $\mathbf{N} = \mathbf{N}(\boldsymbol{\xi})$ can also be written in Bernstein–Bézier form. It remains to present a method for calculating the Bézier coefficients for the surface normal. We present formulas for these coefficients for simplicial Bernstein–Bézier elements in Theorem 5.5.2 and for tensor product elements in Theorem 5.5.3.

We begin by considering simplicial elements. Let $\{B^{p'}_{\mathbf{k}}\}_{\mathbf{k}\in I^{p'}}$ denote the set of simplicial Bernstein polynomials of degree $p' = d\,(p-1)$, and let $\{\mathbf{N}_{\mathbf{k}}\}_{\mathbf{k}\in I^{p'}}$ denote the set of Bézier coefficients for the vector $\mathbf{N}$. Then, for a simplicial element in projective space we can write $\mathbf{N}\,(\boldsymbol{\xi})$ in Bernstein–Bézier form as:

$$\mathbf{N}\,(\boldsymbol{\xi}) = \sum_{\mathbf{k}\in I^{p'}} B^{p'}_{\mathbf{k}} \mathbf{N}_{\mathbf{k}} \quad \forall\ \boldsymbol{\xi} \in \hat{\Omega}$$

wherein $I^{p'}$ denotes the index set over the simplicial Bernstein polynomials of degree p'. Then, let us denote a d-tuple of multi-indices as $\mathbf{I} = \{\mathbf{i}_1, ..., \mathbf{i}_d\}$, and let us we define the set $\mathcal{I}^{p'}_{\mathbf{k}}$ as:

$$\mathcal{I}^{p'}_{\mathbf{k}} := \left\{ \mathbf{I} = \{\mathbf{i}_j\}_{j=1}^{d} \ :\ \mathbf{i}_j \in I^{p-1},\ \sum_{j=1}^{d} \mathbf{i}_j = \mathbf{k} \right\}$$

Finally, let us denote the set of difference vectors in the ξ_j direction as $\{\Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_j}\}_{\mathbf{i}_j\in I^{\mathbf{p}-\mathbf{e}_j}}$, where we define $\Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_j}$ as:

$$\Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_j} = \widetilde{\mathbf{P}}^b_{\mathbf{i}+\mathbf{e}_j} - \widetilde{\mathbf{P}}^b_{\mathbf{i}}$$

Theorem 5.5.2. *For a simplicial Bernstein–Bézier element of degree p in projective space, with projective control points $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I^p}$, the Bézier coefficients $\{\mathbf{N}_{\mathbf{k}}\}_{\mathbf{k}\in I'}$ for the vector $\mathbf{N}$ can be calculated as:*

$$\mathbf{N}_{\mathbf{k}} = * \left(\sum_{\mathbf{I}\in\mathcal{I}^{p'}_{\mathbf{k}}} \eta_{\mathbb{k}}\,(\mathbf{I}) \left(\Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_1} \wedge ... \wedge \Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_d} \right) \right) \tag{5.16}$$

wherein the coefficient $\eta_{\mathbb{k}}\,(\mathbf{I})$ is defined to be:

$$\eta_{\mathbf{k}}\,(\mathbf{I}) = \frac{p^d \binom{p-1}{\mathbb{i}_1} \cdots \binom{p-1}{\mathbb{i}_d}}{\binom{p'}{\mathbb{k}}}$$

Proof. We first recognize that a projective Bézier element is a d-manifold with codimension 1, and as such the surface normal can be found as the Hodge dual of the wedge product of the parametric derivatives, viz:

$$\mathbf{N} = * \left(\frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1} \wedge \ldots \wedge \frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_d} \right) \tag{5.17}$$

Note that when $d = 2$, this is simply the cross product, but the above notation holds for arbitrary d. Now, recognizing that the partial derivative with respect to the j^{th} parametric coordinate can be found as:

$$\frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_j} = p \sum_{\mathbf{i}_j \in I^{p-1}} B^{p-1}_{\mathbf{i}_j} \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_j}$$

we rewrite Eq. (5.17) as:

$$\mathbf{N} = * \left(p \sum_{\mathbf{i}_1 \in I^{p-1}} B^{p-1}_{\mathbf{i}_1} \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_1} \wedge \ldots \wedge p \sum_{\mathbf{i}_d \in I^{p-1}} B^{p-1}_{\mathbf{i}_d} \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_d} \right)$$

and since the distributive property holds, we can rearrange to yield:

$$\mathbf{N} = * \left(p^d \sum_{\mathbf{i}_1 \in I^{p-1}} \ldots \sum_{\mathbf{i}_d \in I^{p-1}} B^{p-1}_{\mathbf{i}_1} \ldots B^{p-1}_{\mathbf{i}_d} \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_1} \wedge \ldots \wedge \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_d} \right)$$

Now, recognizing that we can write the product of the Bernstein basis functions as:

$$B^{p-1}_{\mathbf{i}_1} \ldots B^{p-1}_{\mathbf{i}_d} = \frac{\binom{p-1}{\mathbb{i}_1} \ldots \binom{p-1}{\mathbb{i}_d}}{\binom{p'}{\mathbb{k}}} B^{p'}_{\mathbf{k}}$$

we arrive at:

$$\mathbf{N} = * \left(p^d \sum_{\mathbf{i}_1 \in I^{p-1}} \ldots \sum_{\mathbf{i}_d \in I^{p-1}} \frac{\binom{p-1}{\mathbb{i}_1} \ldots \binom{p-1}{\mathbb{i}_d}}{\binom{p'}{\mathbb{k}}} B^{p'}_{\mathbf{k}} \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_1} \wedge \ldots \wedge \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_d} \right)$$

Finally, rearranging the order of summation, we get:

$$\mathbf{N} = * \left(\sum_{\mathbf{k} \in I^{p'}} B^{p'}_{\mathbf{k}} \sum_{\mathbf{I} \in \mathcal{I}^{p'}_{\mathbf{k}}} \frac{p^d \binom{p-1}{\mathbb{i}_1} \ldots \binom{p-1}{\mathbb{i}_d}}{\binom{p'}{\mathbb{k}}} \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_1} \wedge \ldots \wedge \Delta \widetilde{\mathbf{P}}^b_{\mathbf{i}_d} \right)$$

From which Eq. (5.16) immediately follows. $\square$

We can now derive similar results for tensor product elements. Let $\{B^{\mathbf{p}'}_{\mathbf{k}}\}_{\mathbf{k} \in I^{\mathbf{p}'}}$ denote the set of tensor product Bernstein polynomials of degree $\mathbf{p}' = d\mathbf{p} - \mathbf{1}$, and let $\{\mathbf{N}_{\mathbf{k}}\}_{\mathbf{k} \in I^{\mathbf{p}'}}$ denote the set of Bézier coefficients for the vector $\mathbf{N}$. Then, for a tensor product element in projective space we can write $\mathbf{N}\left(\boldsymbol{\xi}\right)$ in Bernstein–Bézier form as:

$$\mathbf{N}(\boldsymbol{\xi}) = \sum_{\mathbf{k} \in I^{\mathbf{p}'}} B^{\mathbf{p}'}_{\mathbf{k}} \mathbf{N}_{\mathbf{k}} \;\; \forall \, \boldsymbol{\xi} \in \hat{\Omega}$$

Now, let us denote a d-tuple of multi-indices as $\mathbf{I} = \{\mathbf{i}_1, ..., \mathbf{i}_d\}$. Then, we define the set of d-tuples $\mathcal{I}^{\mathbf{p}'}_{\mathbf{k}}$ as:

$$\mathcal{I}^{\mathbf{p}'}_{\mathbf{k}} := \left\{ \mathbf{I} = \{\mathbf{i}_j\}_{j=1}^{d} \;:\; \mathbf{i}_j \in I^{\mathbf{p}-\mathbf{e}_j}, \; \sum_{j=1}^{d} \mathbf{i}_j = \mathbf{k} \right\}$$

wherein $I^{\mathbf{p}'}$ denotes the index set over the tensor product Bernstein polynomials of degree $\mathbf{p}'$. As before, let $\{\Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_j}\}_{\mathbf{i}_j \in I^{\mathbf{p}-\mathbf{e}_j}}$ denote the set of difference vectors in the ξ_j direction.

Theorem 5.5.3. *For a tensor product Bernstein–Bézier element of degree* $\mathbf{p}$ *in projective space, with projective control points* $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i} \in I^{\mathbf{p}}}$*, the Bézier coefficients* $\{\mathbf{N}_{\mathbf{k}}\}_{\mathbf{k} \in I'}$ *for the vector* $\mathbf{N}$ *can be calculated as:*

$$\mathbf{N}_{\mathbf{k}} = * \left(\sum_{\mathbf{I} \in \mathcal{I}^{\mathbf{p}'}_{\mathbf{k}}} \eta_{\mathbf{k}}\left(\mathbf{I}\right) \left(\Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_1} \wedge ... \wedge \Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_d} \right) \right) \tag{5.18}$$

wherein the coefficient $\eta_{\mathbf{k}}\left(\mathbf{I}\right)$ *is defined to be:*

$$\eta_{\mathbf{k}}\left(\mathbf{I}\right) = \frac{p_1 \binom{\mathbf{p}-e_1}{\mathbf{i}_1} ... p_d \binom{\mathbf{p}-e_d}{\mathbf{i}_d}}{\binom{\mathbf{p}'}{\mathbf{k}}}$$

Proof. The proof follows the proof for simplicial elements almost exactly. We simply recognize that j^{th} directional derivative for a tensor product element is:

$$\frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_j} = p_j \sum_{\mathbf{i}_j \in I^{\mathbf{p}-\mathbf{e}_j}} B^{\mathbf{p}-\mathbf{e}_j}_{\mathbf{i}_j} \Delta\widetilde{\mathbf{P}}^b_{\mathbf{i}_j}$$

and that the product of tensor product Bernstein polynomials can be written as:

$$B^{\mathbf{p}-\mathbf{e}_1}_{\mathbf{i}_1} ... B^{\mathbf{p}-\mathbf{e}_d}_{\mathbf{i}_d} = \frac{\binom{\mathbf{p}-\mathbf{e}_1}{\mathbf{i}_1} ... \binom{\mathbf{p}-\mathbf{e}_d}{\mathbf{i}_d}}{\binom{\mathbf{p}'}{\mathbf{k}}} B^{\mathbf{p}'}_{\mathbf{k}}$$

Then, using these identities along with Eq. (5.17), Eq. (5.18) can be readily obtained. □

Theorem III. *Let $\Omega^e \in \mathbb{R}^d$ be a rational Bernstein–Bézier element in physical space with corresponding rational element in projective space $\widetilde{\Omega}^e \in \mathbb{R}^{d+1}$. Then, letting $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I}$ denote the projective control points, and letting $\{\mathbf{N}_{\mathbf{k}}\}_{\mathbf{k}\in I'}$ denote the Bézier coefficients for the normal vector $\mathbf{N}$, the constant C_{det} is bounded from above by:*

$$C_{det} \leq \left(\frac{\max\limits_{j\in I} w_j}{\min\limits_{j\in I} w_j} \right)^{d+1} \left(\frac{\max\limits_{\substack{\mathbf{i}\in I \\ \mathbf{k}\in I'}} \mathbf{N}_{\mathbf{k}} \cdot \widetilde{\mathbf{P}}^b_{\mathbf{i}}}{\min\limits_{\substack{\mathbf{i}\in I \\ \mathbf{k}\in I'}} \mathbf{N}_{\mathbf{k}} \cdot \widetilde{\mathbf{P}}^b_{\mathbf{i}}} \right)$$

Proof. From Theorem 5.5.1, we recognize that $\det[\boldsymbol{\nabla}\mathbf{x}^e]$ is given by:

$$\det[\boldsymbol{\nabla}\mathbf{x}^e] = \frac{\mathbf{N} \cdot \widetilde{\boldsymbol{x}}}{w^{d+1}}$$

From this, we can rewrite $\det[\boldsymbol{\nabla}\mathbf{x}^e]$ explicitly in terms of the Bernstein basis polynomials as:

$$\det[\boldsymbol{\nabla}\mathbf{x}^e] = \frac{1}{\left(\sum\limits_{\mathbf{j}\in I} B_{\mathbf{j}} w_{\mathbf{j}} \right)^{d+1}} \sum_{\mathbf{i}\in I} \sum_{\mathbf{k}\in I'} B_{\mathbf{k}} B_{\mathbf{i}} \mathbf{N}_{\mathbf{k}} \cdot \widetilde{\mathbf{P}}^b_{\mathbf{i}}$$

Then, because the Bernstein basis polynomials satisfy positivity and partition of unity, we can bound the magnitude of the Jacobian determinant by:

$$\frac{\min\limits_{\substack{\mathbf{i}\in I \\ \mathbf{k}\in I'}} \mathbf{N}_{\mathbf{k}} \cdot \widetilde{\mathbf{P}}^b_{\mathbf{i}}}{\left(\max\limits_{\mathbf{i}\in I} w_{\mathbf{i}} \right)^{d+1}} \leq |\det[\boldsymbol{\nabla}\mathbf{x}^e]| \leq \frac{\max\limits_{\substack{\mathbf{i}\in I \\ \mathbf{k}\in I'}} \mathbf{N}_{\mathbf{k}} \cdot \widetilde{\mathbf{P}}^b_{\mathbf{i}}}{\left(\min\limits_{\mathbf{i}\in I} w_{\mathbf{i}} \right)^{d+1}} \tag{5.19}$$

Finally, recognizing that the equation for C_{det} can be equivalently written:

$$C_{det} = \left(\frac{\sup\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det[\boldsymbol{\nabla}\mathbf{x}^e]|}{\inf\limits_{\boldsymbol{\xi}\in\hat{\Omega}} |\det[\boldsymbol{\nabla}\mathbf{x}^e]|} \right)^{1/2}$$

and we use the results of Eq. (5.19) to arrive at the results of Theorem III. □

5.5.2 Computable Bounds on Derivatives of the Mapping $\mathbf{x}^e$

With a method for calculating bounds on the Jacobian determinant established, we turn our attention to computing bounds for higher order derivatives. Compared to bounds on the Jacobian determinant, bounds on the higher order derivatives are relatively easy to derive. These bounds are presented below in the proof for Theorem IV. With these bounds established, we have succeeded in establishing a set of sufficient and computable validity metrics for rational Bernstein–Bézier metrics.

Theorem IVa. *Let us denote the projective control points of a simplicial Bernstein–Bézier element of degree p as* $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I^p}$. *Then, the* $\boldsymbol{\alpha}^{th}$ *partial derivative of the mapping* $\widetilde{\mathbf{x}}^e$ *is bounded by:*

$$||D^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e||_{L^\infty(\hat{\Omega})} \leq \frac{p!}{(p-|\boldsymbol{\alpha}|)!} \max_{\mathbf{i}\in I^{p-|\boldsymbol{\alpha}|}} \left| \sum_{\mathbf{j}\in I^{\boldsymbol{\alpha}}} (-1)^{\boldsymbol{\alpha}+\mathbf{j}} \binom{\boldsymbol{\alpha}}{\mathbf{j}} \widetilde{\mathbf{P}}^b_{\mathbf{i}+\mathbf{j}} \right|$$

Proof. Consider a simplicial Bernstein–Bézier element in projective space, $\widetilde{\Omega}^e \in \mathbb{R}^{d+1}$ defined by control points $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I^p}$. We recognize that the derivatives of Bernstein polynomials are themselves Bernstein polynomials of a lower degree [55]. Thus, we can recursively take the derivative of the mapping $\widetilde{\mathbf{x}}^e$, which yields the following equation for the $\boldsymbol{\alpha}^{th}$ partial derivative:

$$D^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e = \frac{p!}{(p-|\boldsymbol{\alpha}|)!} \sum_{\mathbf{i}\in I^{p-|\boldsymbol{\alpha}|}} \left[B^{p-|\boldsymbol{\alpha}|}_{\mathbf{i}}(\boldsymbol{\xi}) \sum_{\mathbf{j}\in I^{\boldsymbol{\alpha}}} (-1)^{\boldsymbol{\alpha}+\mathbf{j}} \binom{\boldsymbol{\alpha}}{\mathbf{j}} \widetilde{\mathbf{P}}^b_{\mathbf{i}+\mathbf{j}} \right]$$

Note, we take care to emphasize that the sum $\sum\limits_{\mathbf{j}\in I^{\boldsymbol{\alpha}}}$ is a sum over a tensor product index set. This is a consequence of the fact that the partial derivative $D^{\boldsymbol{\alpha}}$ has an inherently tensor product nature. Then, because the Bernstein polynomials satisfy positivity and partition of unity, the bounds of Theorem IVa are obtained. □

Theorem IVb. *Let us denote the projective control points of a tensor product Bernstein–Bézier element of degree* $\mathbf{p}$ *as* $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I^{\mathbf{p}}}$*. Then, the* $\boldsymbol{\alpha}^{th}$ *partial derivative of the mapping* $\widetilde{\mathbf{x}}^e$ *is bounded by:*

$$||D^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e||_{L^\infty(\hat{\Omega})} \leq \frac{\mathbf{p}!}{(\mathbf{p}-\boldsymbol{\alpha})!}\max_{\mathbf{i}\in I^{\mathbf{p}-\boldsymbol{\alpha}}}\left|\sum_{\mathbf{j}\in I^{\boldsymbol{\alpha}}}(-1)^{\boldsymbol{\alpha}+\mathbf{j}}\binom{\boldsymbol{\alpha}}{\mathbf{j}}\widetilde{\mathbf{P}}^b_{\mathbf{i}+\mathbf{j}}\right|$$

Proof. Consider a tensor product Bernstein–Bézier element in projective space, $\widetilde{\Omega}^e \in \mathbb{R}^{d+1}$ defined by control points $\{\widetilde{\mathbf{P}}^b_{\mathbf{i}}\}_{\mathbf{i}\in I^{\mathbf{p}}}$. As before, we write the derivatives of the Bernstein polynomials as Bernstein polynomials of lower degree. This yields the following equation for the $\boldsymbol{\alpha}^{th}$ partial derivative of the mapping $\widetilde{\mathbf{x}}^e$:

$$D^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e = \frac{\mathbf{p}!}{(\mathbf{p}-\boldsymbol{\alpha})!}\sum_{\mathbf{i}\in I^{\mathbf{p}-\boldsymbol{\alpha}}}\left[B^{\mathbf{p}-\boldsymbol{\alpha}}_{\mathbf{i}}(\boldsymbol{\xi})\sum_{\mathbf{j}\in I^{\boldsymbol{\alpha}}}(-1)^{\boldsymbol{\alpha}+\mathbf{j}}\binom{\boldsymbol{\alpha}}{\mathbf{j}}\widetilde{\mathbf{P}}^b_{\mathbf{i}+\mathbf{j}}\right]$$

Then, because the Bernstein polynomials satisfy positivity and partition of unity, the desired bounds are obtained. □

With the relevant theory established, we now demonstrate a particularly convenient property of the bounds presented in Theorem IVa and Theorem IVb. First, let us consider the case of finding the higher order derivatives of a cubic Bernstein–Bézier triangle. Table 5.2 shows the bounding expressions for several derivatives of the mapping $\mathbf{x}^e$. We see that the bounds on the first derivative can be found by evaluating the expression:

$$\left|3\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{1,0\}} - 3\widetilde{\mathbf{P}}^b_{\mathbf{i}}\right|$$

on the sub-triangle containing points $\widetilde{\mathbf{P}}^b_{\mathbf{i}} = \{\widetilde{\mathbf{P}}^b_{\{0,0\}}, \widetilde{\mathbf{P}}^b_{\{1,0\}}, \widetilde{\mathbf{P}}^b_{\{2,0\}}, \widetilde{\mathbf{P}}^b_{\{0,1\}}, \widetilde{\mathbf{P}}^b_{\{1,1\}}, \widetilde{\mathbf{P}}^b_{\{0,2\}}\}$. Similarly, the bound on the second derivative is found by evaluating

$$\left|6\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{2,0\}} - 12\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{1,0\}} + 6\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{0,0\}}\right|$$

at the points $\widetilde{\mathbf{P}}^b_{\mathbf{i}} = \{\widetilde{\mathbf{P}}^b_{\{1,0\}}, \widetilde{\mathbf{P}}^b_{\{1,0\}}, \widetilde{\mathbf{P}}^b_{\{0,1\}}\}$. Then, we note that for a simplicial element of degree p, all partial derivatives of order $|\boldsymbol{\alpha}| = p$ will be a constant across the element. As such, the third derivative for a cubic Bernstein–Bézier triangle can be calculated analytically by the expression:

$$\left|6\widetilde{\mathbf{P}}^b_{\{3,0\}} - 18\widetilde{\mathbf{P}}^b_{\{2,0\}} + 18\widetilde{\mathbf{P}}^b_{\{1,0\}} - 6\widetilde{\mathbf{P}}^b_{\{0,0\}}\right|$$

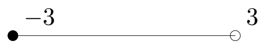
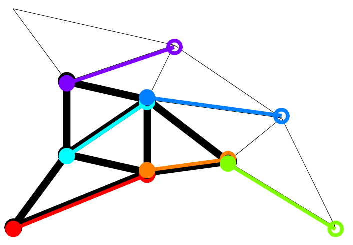
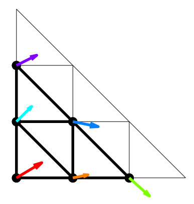
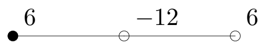
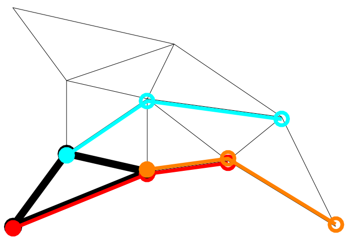
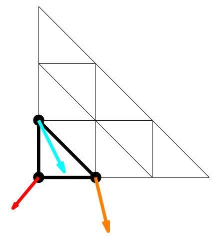
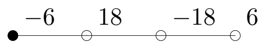
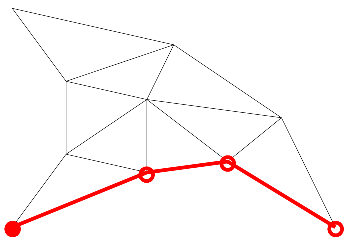
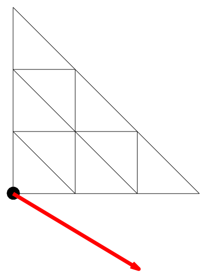
From this, it is apparent that bounds on the derivatives of Bernstein–Bézier elements can be calculated using what is effectively a weighted finite difference method. To illustrate this notion, Table 5.3 shows finite difference stencils for the each of the derivatives shown in Table 5.2. We then show this stencil applied to the control points of the element, as well as the resulting Bézier coefficients for the derivative, shown as vectors on the reference element.

For clarity, we have shown a non-rational cubic Bézier triangle in $\mathbb{R}^2$, but the concepts extends readily to elements in projective space. We have also included the explicitly calculated stencils for a variety of elements in Appendix A. This is not an exhaustive list, but we note note that the results of Section 5.5 can be used to calculate stencils for **any** simplicial or tensor product Bernstein–Bézier element.

Table 5.2: Bounds on the derivatives of a cubic Bernstein–Bézier triangle.

Derivative	Bound
$\dfrac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1}$	$\leq \max\limits_{\mathbf{i} \in I^{p-1}} \left|3\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{1,0\}} - 3\widetilde{\mathbf{P}}^b_{\mathbf{i}}\right|$
$\dfrac{\partial^2 \widetilde{\mathbf{x}}^e}{\partial \xi_1^2}$	$\leq \max\limits_{\mathbf{i} \in I^{p-2}} \left|6\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{2,0\}} - 12\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{1,0\}} + 6\widetilde{\mathbf{P}}^b_{\mathbf{i}+\{0,0\}}\right|$
$\dfrac{\partial^3 \widetilde{\mathbf{x}}^e}{\partial \xi_1^3}$	$= \left|6\widetilde{\mathbf{P}}^b_{\{3,0\}} - 18\widetilde{\mathbf{P}}^b_{\{2,0\}} + 18\widetilde{\mathbf{P}}^b_{\{1,0\}} - 6\widetilde{\mathbf{P}}^b_{\{0,0\}}\right|$

Table 5.3: Example of finding the Bézier coefficients for several different derivatives using a stencil.

Derivative	Stencil	Stencils Applied to the Physical Triangle	Bézier Coefficients of the Derivative
$\frac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1}$			
$\frac{\partial^2 \widetilde{\mathbf{x}}^e}{\partial \xi_1^2}$			
$\frac{\partial^3 \widetilde{\mathbf{x}}^e}{\partial \xi_1^3}$			

5.6 Numerical Examples

In this section, we present several numerical examples to demonstrate how our element distortion metrics may be used in practice. Our goals are threefold. First, we desire to confirm our approximation results for shape regular refinements. That is, we desire to demonstrate that if the sufficient conditions of Theorem I hold, then we will observe optimal convergence rates over rational Bernstein–Bézier elements. Second, we demonstrate that our element distortion metrics are capable of predicting whether a family of refinements is shape regular. Finally, we demonstrate how the element distortion metrics presented here can be used for mesh optimization.

We provide four examples to benchmark our methods. First, we consider a simple rectangular plate, meshed with distorted **polynomial** elements, to study the effect of control point distortion under h-refinement. Next, we consider a plate with a hole, meshed with distorted **rational** elements, to examine the effect of weighting function distortion under h-refinement. We then consider a quarter annulus, meshed with rational elements, under p-refinement, and conclude with an example of how our metrics may be used for mesh optimization.

The examples considered here are relatively simple, but they still demonstrate that poorly shaped elements can have appreciable impacts on solution accuracy. We also note that the examples shown here are constrained to the two dimensional case, as this allows for clear and easy visualization of element shape. However, the implications of these 2D results extend immediately to elements in three dimensions.

5.6.1 Mesh of a Rectangular Plate

To demonstrate the use of our validity metrics, we begin by considering several different meshes of a rectangular plate. To account for both tensor product and simplicial elements, we consider both quadrilateral and triangular meshes. The triangular meshes are formed by simply bisecting each element in the quadrilateral mesh. For both types of elements, we construct an initial mesh, and then create three families of refined meshes.

The initial curvilinear mesh is created by first creating a linear quadrilateral mesh, and degree elevating to **non-rational** bi-cubic Bézier quadrilaterals. Then, for each element, we horizontally perturb the middle two rows of control points, $\left\{\mathbf{P}^b_{\{i_1,i_2\}}\right\}_{i_1=\{0,1,2,3\},i_2=\{1,2\}}$ by some distance:

$$d\mathbf{P}^b_{\mathbf{i}} = (-1)^{i_2}\,\frac{2a}{(4m)^{7/4}}\,\frac{a-\left|\left(\mathbf{P}^b_{\mathbf{i}}\right)_1\right|}{a} \tag{5.20}$$

wherein m denotes the m^{th} mesh in the family, with $m = 1$ being the first mesh.

Then, for both the quadrilateral and triangular mesh, we create the three families of refined meshes as follows. The first family of meshes, shown in Table 5.4, is created by simple uniform subdivision of the original mesh. To create the second family of meshes, shown in Table 5.5, we first perform uniform subdivision on the original *linear* mesh. We then create the m^{th} curvilinear mesh in the family by again perturbing the interior control points by Eq. (5.20). The final family of meshes, shown in Table 5.6, is created analogously to the second family, but the perturbation distance is instead given by the equation,

$$d\mathbf{P}^b_{\mathbf{i}} = (-1)^{i_2}\,\frac{8a}{(4m)^{3}}\,\frac{a-\left|\left(\mathbf{P}^b_{\mathbf{i}}\right)_1\right|}{a}$$

With these three families of meshes established for both the quadrilateral and triangular case, we

Table 5.4: Mesh Family 1.

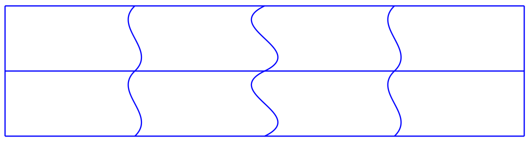
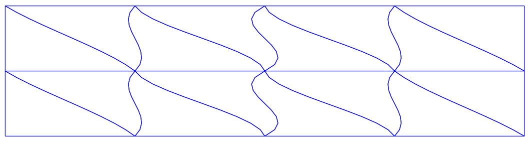
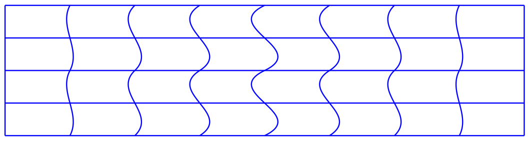
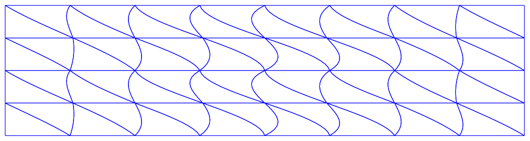
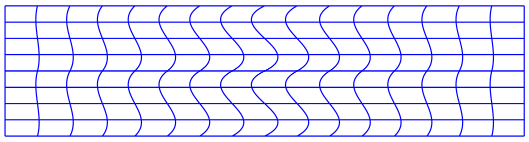
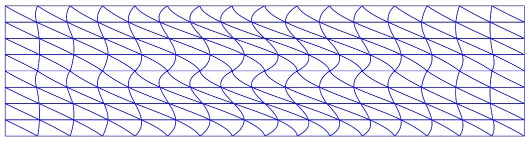
m	Quad Family 1	Tri Family 1
1		
2		
3		

Table 5.5: Mesh Family 2.

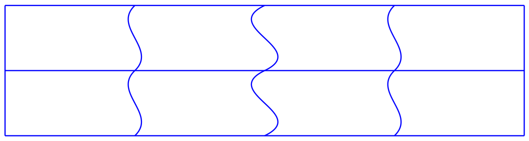
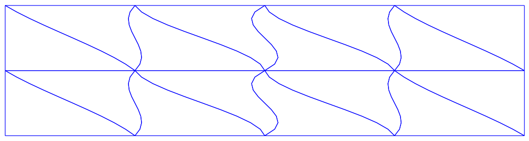
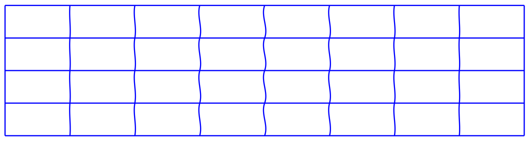
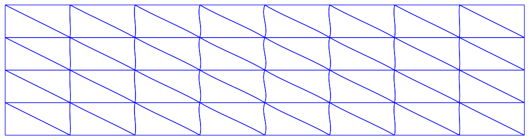
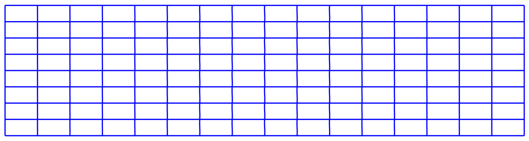
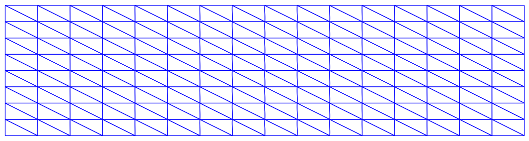
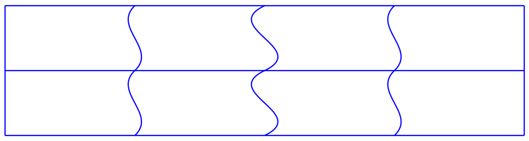
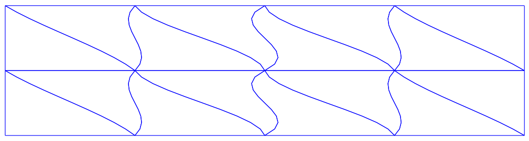
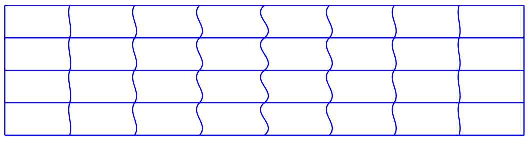
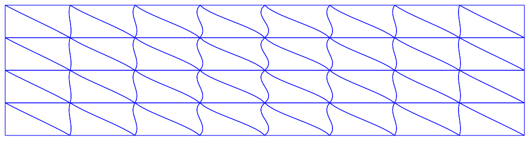
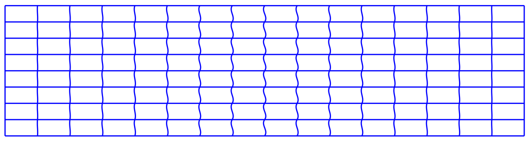
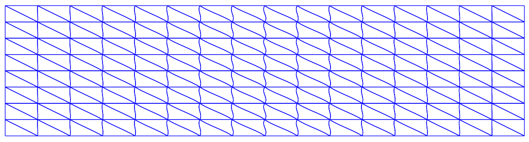
m	Quad Family 2	Tri Family 2
1		
2		
3		

Table 5.6: Mesh Family 3.

m	Quad Family 3	Tri Family 3
1		
2		
3		

use the method of manufactured solutions to study approximation error in each family of meshes. However, when solving PDEs using finite elements, error can be introduced not only by the element shape, but also by the choice of finite element method (e.g. Galerkin's method). As such, over each family of meshes, we solve two problems, an L^2 projection and the Poisson problem. We solve the L^2 projection over the mesh so that we may isolate the effect of element shape on approximation error. We then consider the Poisson problem so as to consider an example with practical engineering applications. Given the domain Ω, let $\mathcal{f}$ denote a forcing function and let $\hbar$ denote a flux across the boundary $\Gamma_{\hbar}$. Then, letting $\mathcal{V}$ denote the space of trial solutions, we desire to find a solution u_h such that for all $v \in \mathcal{V}$:

$$\int\limits_{\Omega} \boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} u_h d\Omega = \int\limits_{\Omega} v \mathcal{f} d\Omega + \int\limits_{\Gamma_{\hbar}} v \hbar d\Gamma$$

For both cases, we attempt to approximate the manufactured solution

$$u\left(\boldsymbol{x}\right) = (x_1 - a)(x_2 - b)\cos\left(\frac{x_2}{2a}\pi\right)\sin\left(\frac{x_2}{b}\pi\right)$$

wherein a and b are the half-width and half-height of the plate centered at the origin. To study the approximation error, we examine the convergence rate of the error $||u - u_h||$ in the L^2 norm for both problems over each family of meshes. Fig. 5.7 shows error convergence plots for the three families of quadrilateral meshes, and Fig. 5.8 shows convergence plots for the three families of triangular meshes. From the convergence plots, we see that in all cases the solution error for problems solved over the first and second family of meshes are converging as expected. However, the error for the problems over the third family of meshes is converging at a less than optimal rate for both the quadrilateral and triangular meshes. To gain insight into this, we look to the distortion metrics for each mesh family. Fig. 5.9 shows the mesh distortion metrics for each family of quadrilateral meshes, and Fig. 5.10 the mesh distortion metrics for each family of triangular meshes.

From Fig. 5.9a, we see that the minimum scaled Jacobian is bounded from below for every quadrilateral mesh family, and that in each case $J_S \to 1$ under mesh refinement. Furthermore, we note that J_S is larger for the third family of meshes (irregular refinements) than it is for the

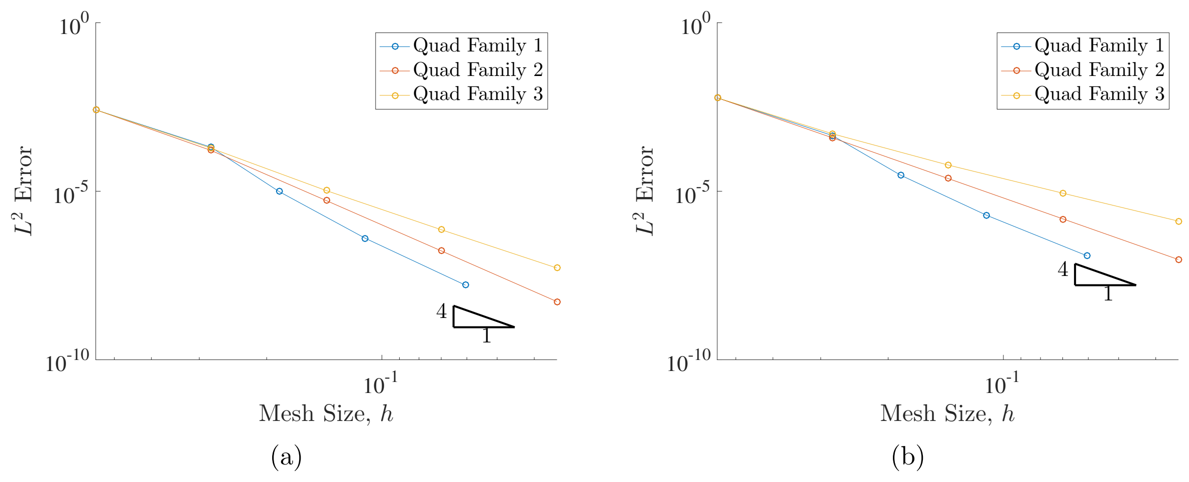


Figure 5.7: Convergence plots for the quadrilateral meshes. (a) L^2 norm of the error for the L^2 projection problem. (b) L^2 norm of the error for Poisson's problem.

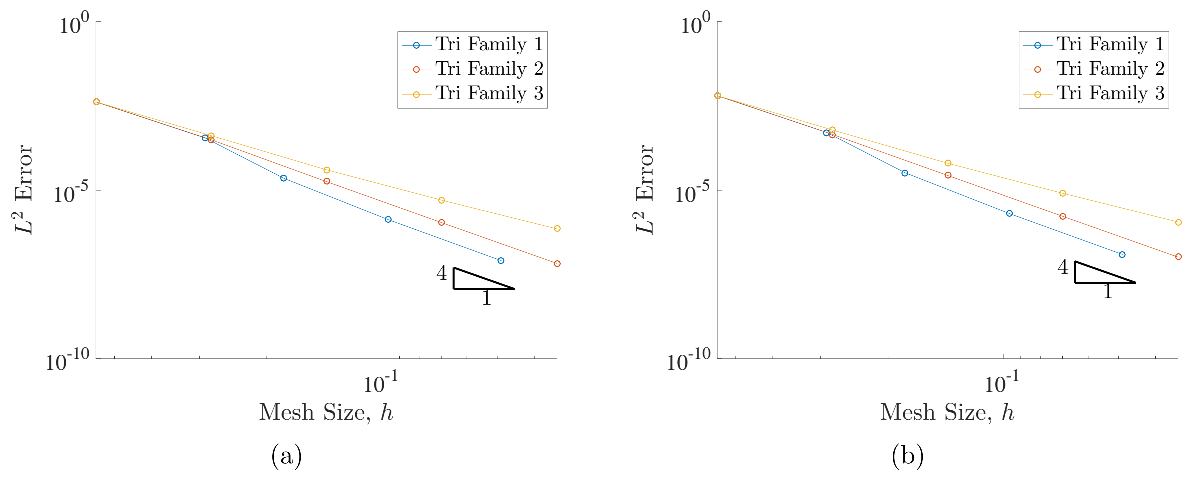


Figure 5.8: Convergence plots for the triangular meshes. (a) L^2 norm of the error for the L^2 projection problem. (b) L^2 norm of the error for Poisson's problem.

first family (uniform refinements). Similar behavior is also observed for the scaled Jacobian of the triangular meshes, shown in Fig. 5.10a. At first blush, these observations seems contradictory, as in both cases, the first family converges as expected, while the third does not.

The cause of the slowed convergence rates can be explained by instead looking at the norms of the higher order derivatives, shown in Fig. 5.9b-d for the quadrilateral case, and Fig. 5.10b-d for the triangular case. For each plot, we show the global upper bound on the derivatives $D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e$ of order $|\boldsymbol{\alpha}| = k$ across the entire mesh. That is, for each $k \leq p$, we plot the value of $\boldsymbol{\nabla}_{max}^k$, where

$$\boldsymbol{\nabla}_{max}^k := \max_{e=1,\ldots,nel} \max_{|\boldsymbol{\alpha}|=k} \sup_{\boldsymbol{\xi}\in\hat{\Omega}} \left| D_{\boldsymbol{\xi}}^{\boldsymbol{\alpha}}\widetilde{\mathbf{x}}^e \right|$$

From Fig. 5.9d, it is readily seen that the cause of the slowed convergence for the third family of quadrilateral meshes is the fact that $||\boldsymbol{\nabla}^3\mathbf{x}^e||_{L^\infty(\hat{\Omega})}$ is converging at approximately $O(h^2)$, whereas Cond. (I.2) requires that it converge at $O(h^3)$.

These results highlight how sensitive convergence rates for higher order elements can be, and motivate the utility of the validity metrics developed in this work. Indeed, from the Jacobian metrics shown in Fig. 5.9a and Fig. 5.10a, as well as visual inspection of the meshes, one might be tempted to draw the conclusion that all three families should preserve optimal convergence rates, even though we have observed that this is clearly not the case.

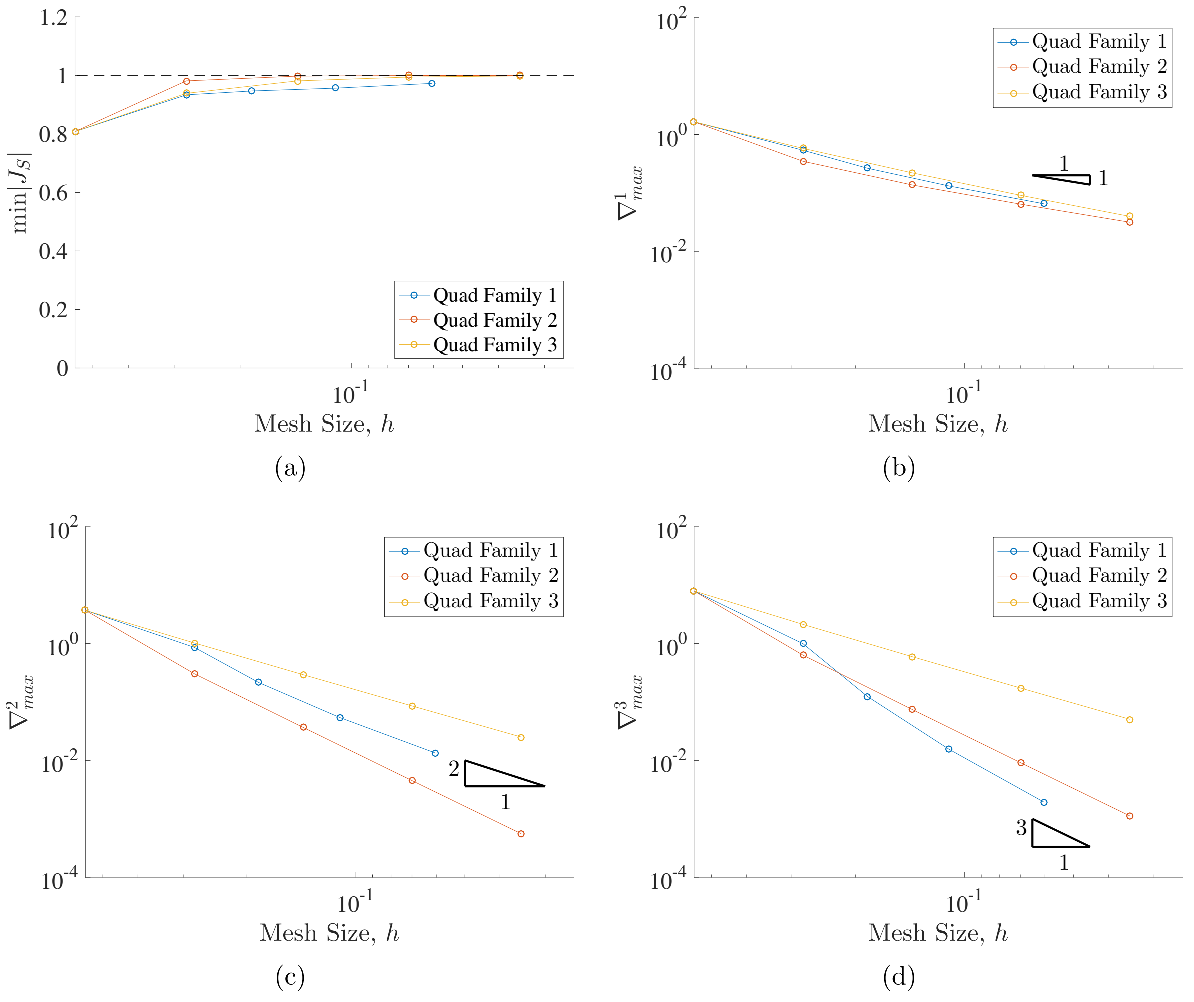


Figure 5.9: Mesh distortion metrics for the quadrilateral meshes of the plate. (a) Lower bound on the scaled Jacobian. (b) Upper bound on the first derivative. (c) Upper bound on the second derivative. (d) Upper bound on the third derivative.

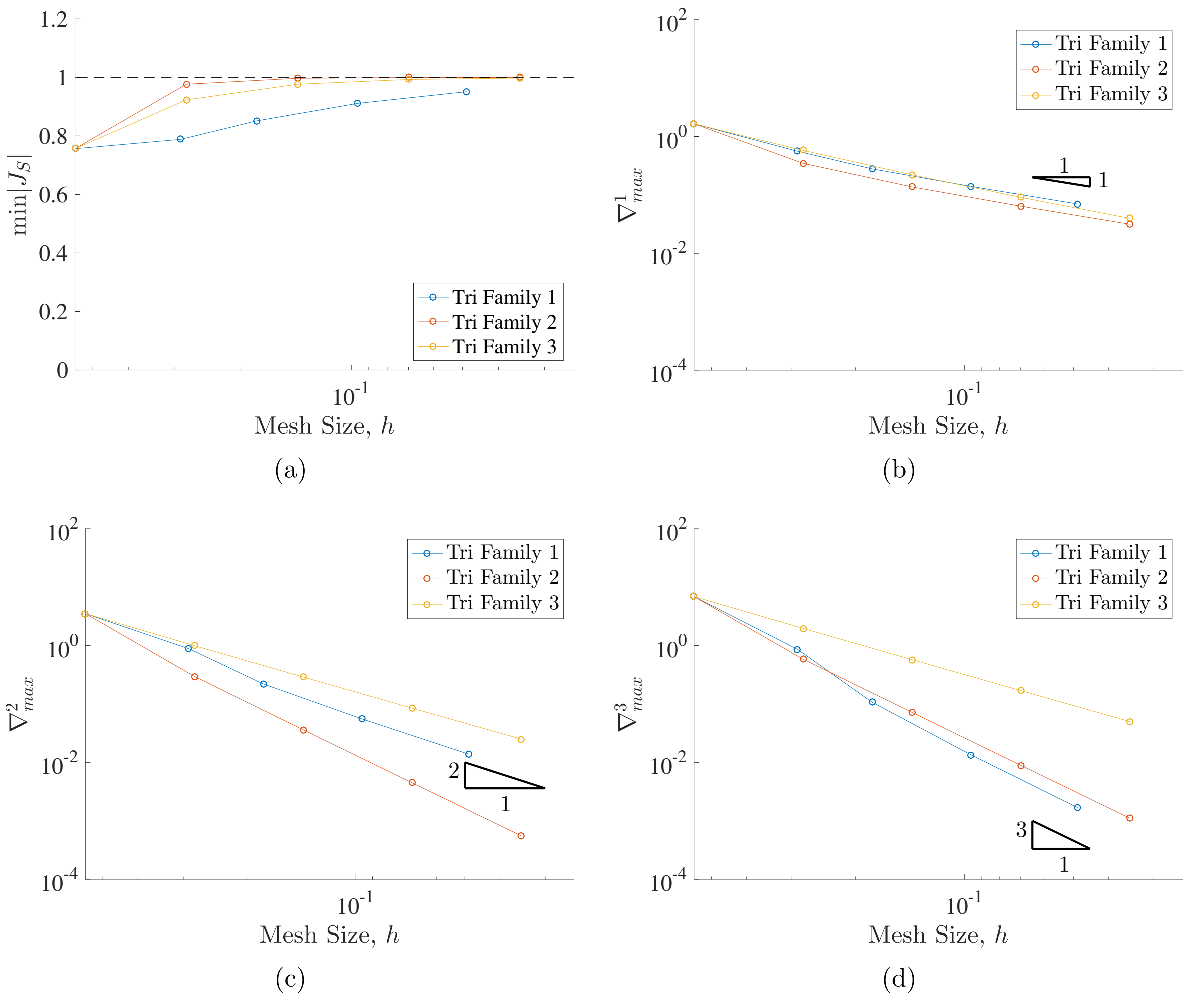
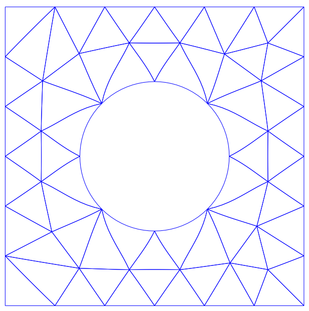
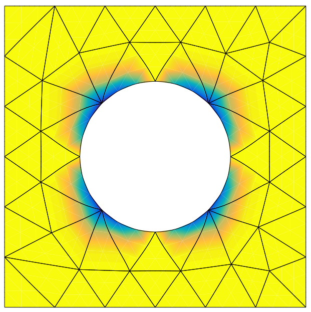
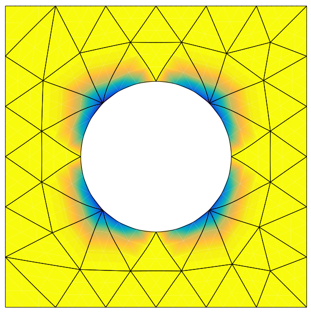
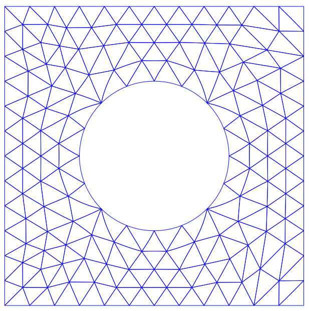
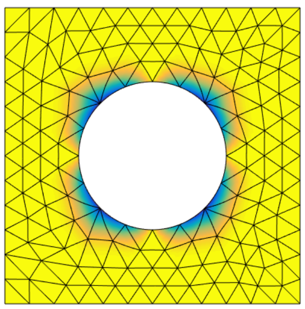
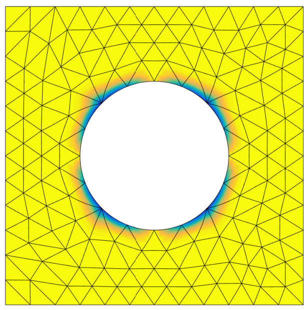
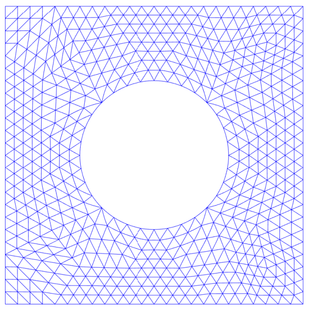
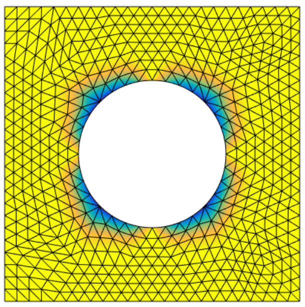
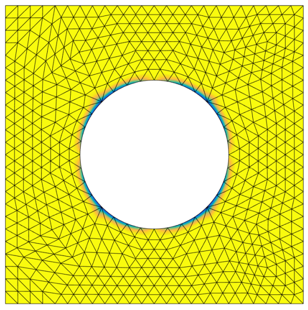


Figure 5.10: Mesh distortion metrics for the triangular meshes of the plate. (a) Lower bound on the scaled Jacobian. (b) Upper bound on the first derivative. (c) Upper bound on the second derivative. (d) Upper bound on the third derivative.

5.6.2 Plate with a Hole

In the previous example, we examined the effect of element shape distortion on approximation error by perturbing control points. In this example, we examine the effect of weighting function distortion on the approximation error by perturbing control weights. We consider a mesh of a plate with a hole, composed of cubic rational Bernstein–Bézier triangles, shown in Table 5.7. We note that since the hole in the plate is circular, we must use rational elements to capture the geometry exactly, and as a result, the control weights corresponding to the points on the boundary will be non unity. However, it remains to set the control weights for the interior points in the mesh. We consider two possible methods of setting control weights for a series of meshes. The first method is to simply set the weights on the circular boundary, to the appropriate values, and set all other weights to one. Then, under mesh refinement, we perform uniform subdivision on both the control points, and control weights As a results, the weighting function remains the same under mesh refinement. The second option we consider here is to perform uniform subdivision on the control points, but not the control weights. Instead, only control weights corresponding to points on the boundary are updated during each refinement step, and all other control weights are set to one. Both of these families of refined weighting functions are shown in Table 5.7.

Table 5.7: Meshes and two families of weighting functions for the plate with a hole.

m	Meshes	Weight Family 1	Weight Family 2
1			
2			
3			

With these two families establish, we study the approximation accuracy using the method of manufacture solutions. As before, we solve Poisson's problem, with the manufactured solution

$$u\left(x_1, x_2\right) = \left(x_1 - a\right)\left(x_1 + a\right)\left(x_2 - a\right)\left(x_2 + a\right)\left(r - \sqrt{x_1^2 + x_2^2}\right)$$

wherein a is the half-width of the square plate and r is the radius of the hole, and the plate is centered at the origin. Figure 5.11 shows the convergence rates of the L^2 norm of the approximation error over both mesh families.

From our results, we see that the first family of meshes converge as expected, while the second does not. Again, we examine the distortion metrics, shown in Fig. 5.12, to gain insight into the cause of the stalled convergence for the second family of meshes. We immediately see that the higher-order derivatives are not converging for the second family of meshes. The reason for this can be observed from the plots of the weighting functions in Table 5.7. Since only the weights lying on the circular boundary are being updated, the gradient of the weighting function becomes increasingly sharp under refinement.

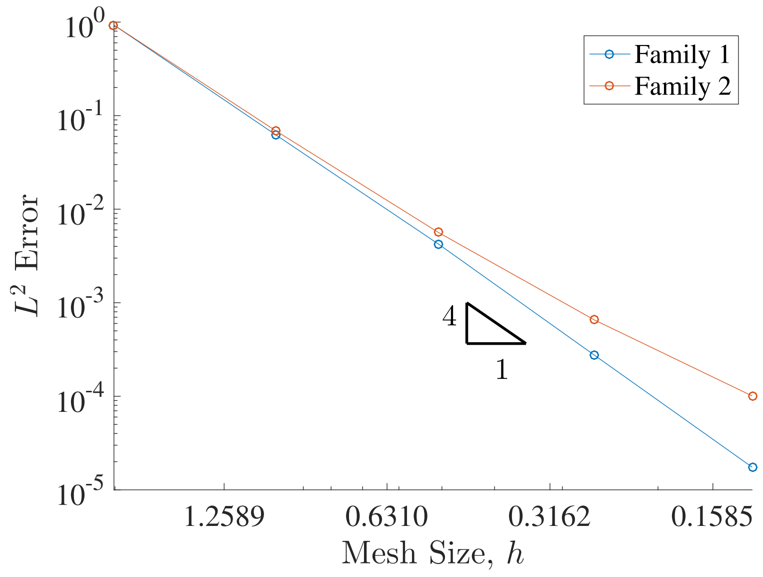


Figure 5.11: Convergence rate of the error in the L^2-norm.

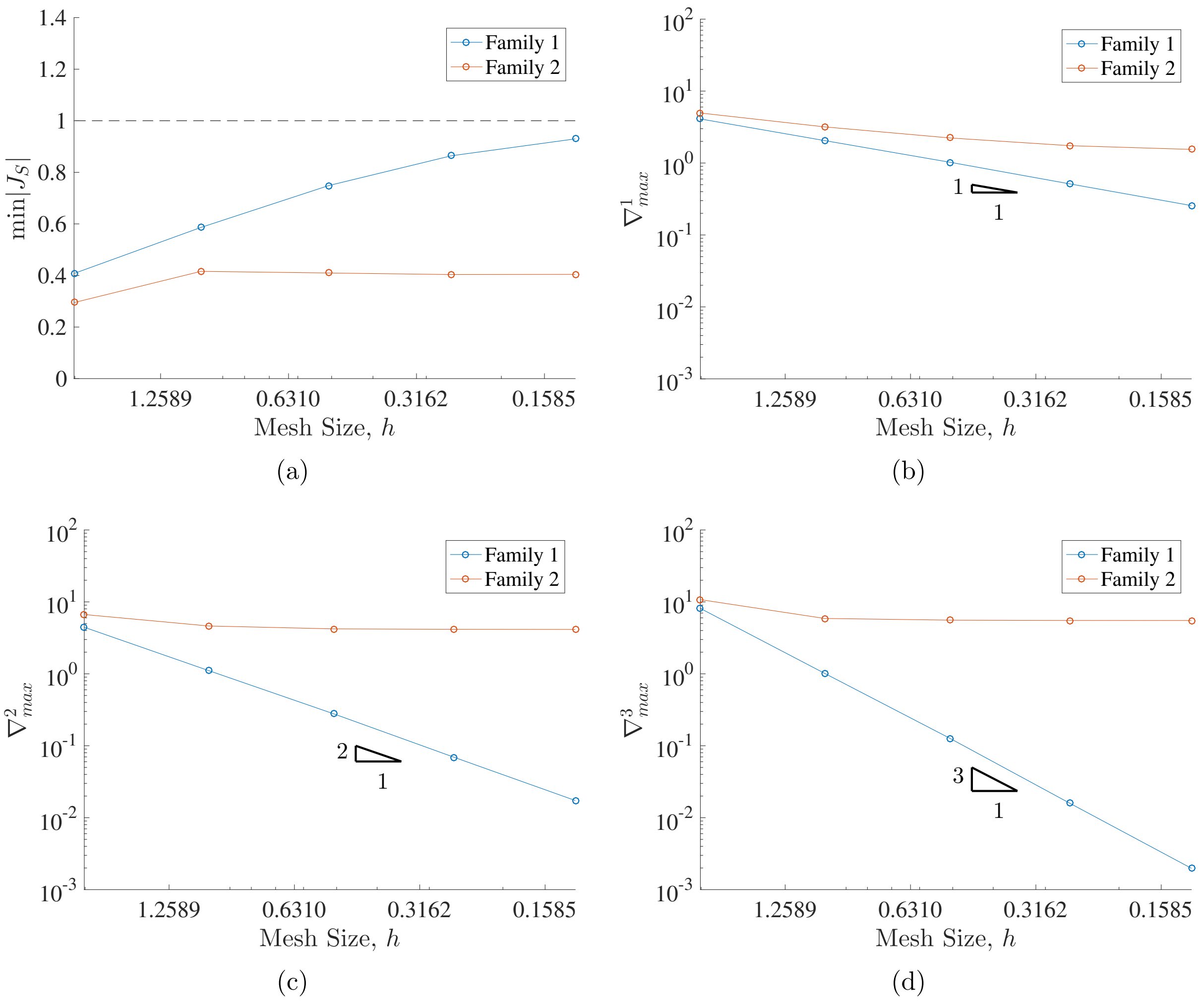
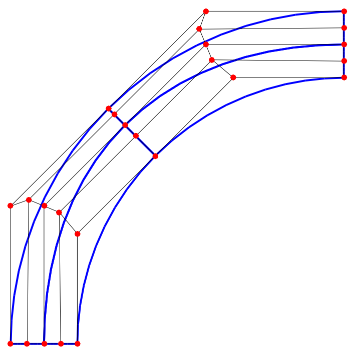
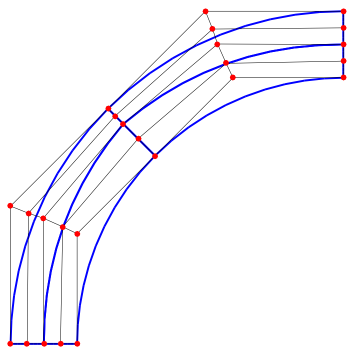
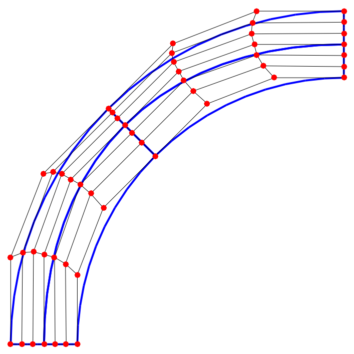
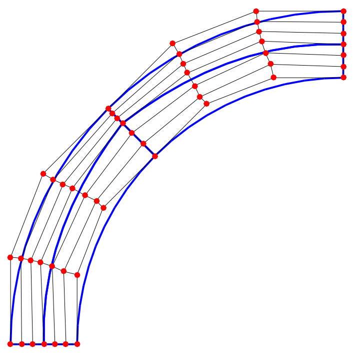
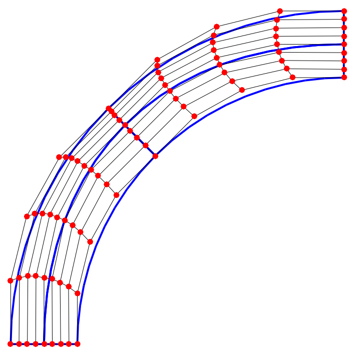
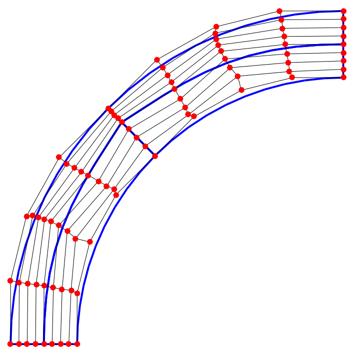


Figure 5.12: Mesh distortion metrics for the triangular meshes of the plate. (a) Lower bound on the scaled Jacobian. (b) Upper bound on the first derivative. (c) Upper bound on the second derivative. (d) Upper bound on the third derivative.

5.6.3 Convergence Under p-refinement

Thus far, we have considered the effect of both control point distortion and control weight distortion on approximation error for cubic Bézier elements. We now consider the effect of mesh shape on convergence under p-refinement. We consider the simple case of a quarter annulus meshes with four rational bi-quadratic Bernstein–Bézier quadrilaterals, shown in Table 5.8. For this initial mesh, we consider two series of p-refined meshes. The first series is created by simple degree elevation of the bi-quadratic mesh. The second series is created using a linear elastic analogy, how higher-order meshes are created in practice. For each level of refinement, we degree elevate the underlying linear mesh to order p, and the geometry is recovered via edge replacement. Then, we solve a linear elasticity problem to update the positions of the interior nodes. The series of p-refined meshes with the Bézier control nets are shown in Table 5.8.

Table 5.8: Families of p-refined meshes of the quarter annulus.

p	Family 1	Family 2
2		
3		
4		

As before, we solve Poisson's problem, with the manufactured solution

$$u\left(x_1, x_2\right) = \frac{70 \ln\left(\sqrt{x_1^2 + x_2^2}/r_i\right) - 200 \ln\left(\sqrt{x_1^2 + x_2^2}/r_o\right)}{\ln\left(r_o/r_i\right)}$$

wherein r_i and r_o are the inner and outer radii of the quarter annulus. The convergence plots of the L^2 error with respect to the polynomial degree p are shown in Fig. 5.14.

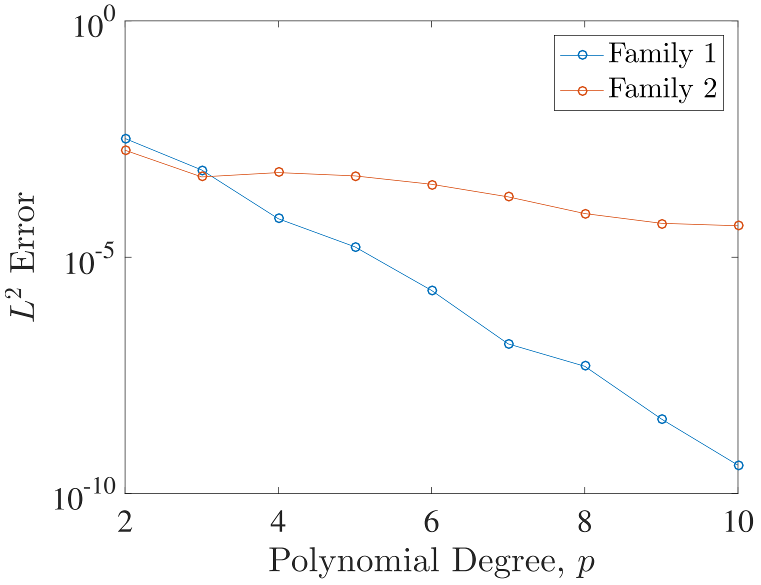


Figure 5.13: Convergence in the L^2 error under p-refinement.

We notice immediately that the first family of meshes exhibits exponential convergence, as is expected. The second family, however, stagnates. The cause of this can be determined by observing the plots of the higher order derivatives of the elements with respect to p, shown in Fig. 5.15. For the first family of meshes (Fig. 5.15a), we see that every derivative up through order $k = 10$ is bounded from above. For the second family of meshes (Fig. 5.15b), we see that for every level of p refinement, the magnitude of **every** derivative of order $k \leq p$ increases.

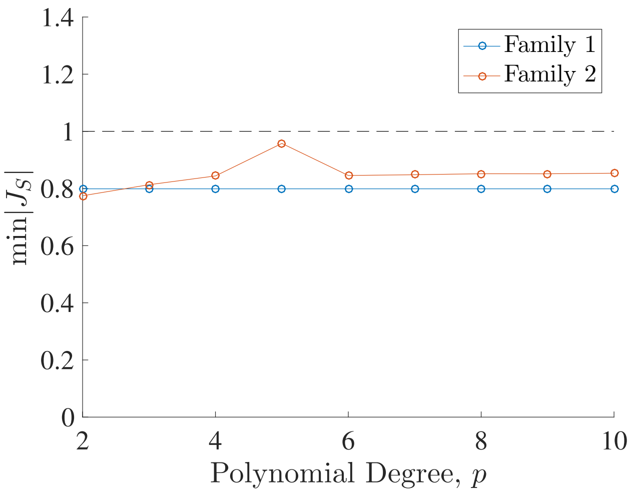


Figure 5.14: Scaled Jacobian metrics for the p-refined meshes of the quarter annulus.

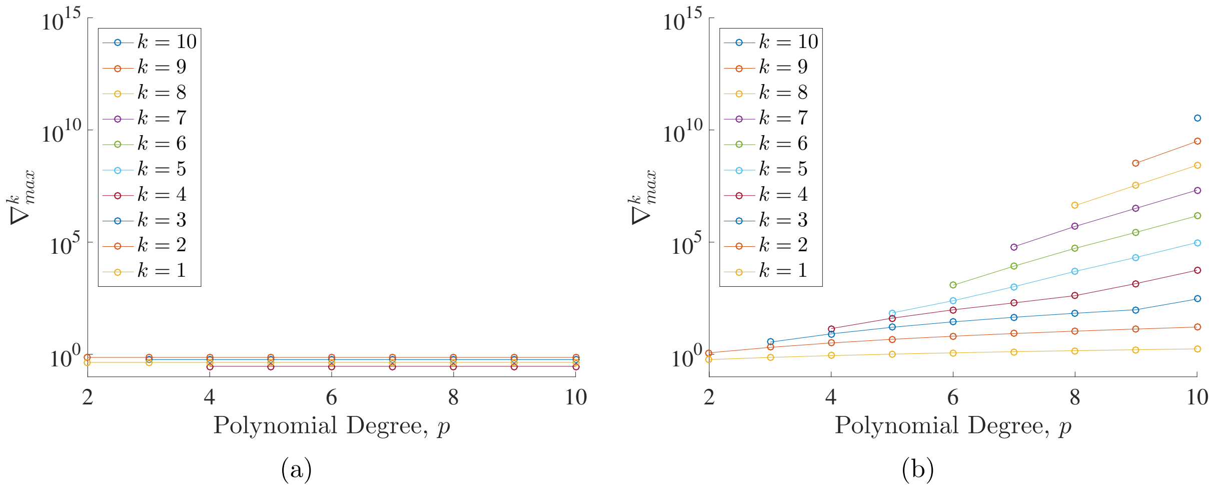


Figure 5.15: Upper bounds on the higher order derivatives for the p-refined meshes of the quarter annulus. The metrics for Family 1 are shown in (a) and the metrics for Family 2 are shown in (b).

5.6.4 Mesh Optimization

Thus far, we have used our mesh distortion metrics to explain sub-optimal convergence rates *a posteriori*. However, we recognize that since these metrics relate mesh distortion to error bounds, we should be able to use these metrics for *a priori* mesh optimization. We consider again a plate with a hole, but now with small chamfers at the corners of the plate.

We consider three families of meshes. In Family 1, we consider an initial coarse mesh, refined by uniform subdivision. In Family 2, we create a new linear mesh at each refinement level m, enforcing a maximum edge length h_m of

$$h_m = \frac{h_0}{2^{m-1}}$$

We then degree elevate, and recover the geometry through edge replacement. The mesh is then smoothed using a linear elastic analogy. In Family 3, we begin the same linear mesh at each refinement step as with Family 2. However, rather than solve a linear elasticity problem to smooth internal nodes, we seek to minimize the cost function

$$c\left(\mathbf{x}^e\right) = ||\boldsymbol{\nabla}\mathbf{x}^e||_{L^\infty(\Omega)} + 2\left|\left|\boldsymbol{\nabla}^2\mathbf{x}^e\right|\right|_{L^\infty(\Omega)} + 4\left|\left|\boldsymbol{\nabla}^3\mathbf{x}^e\right|\right|_{L^\infty(\Omega)}$$

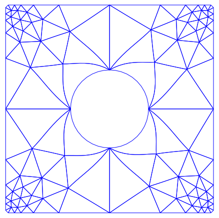
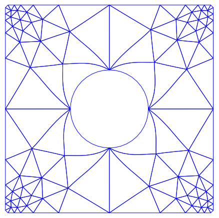
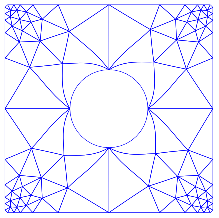
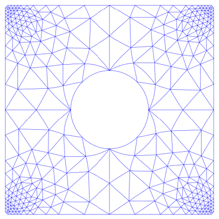
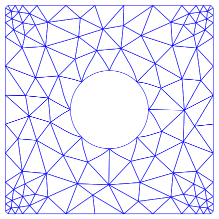
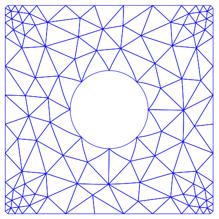
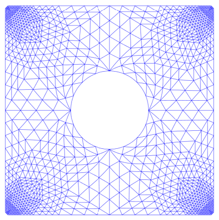
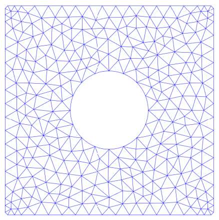
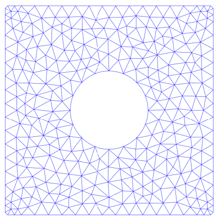
These three families of meshes are shown in Table 5.9. We note that visually, there is little difference between the meshes of Family 2 and Family 3.

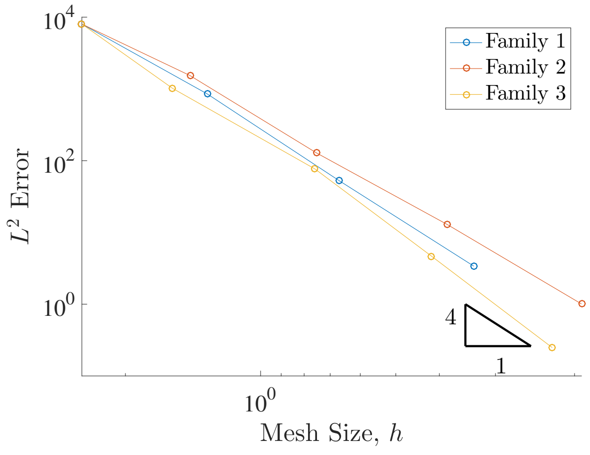
To study the effect of the mesh optimization on the accuracy, we solve Poisson's problem over each mesh family, with the manufactured solution

$$\begin{aligned} u\left(x_1, x_2\right) = \left(2a - c - x_1 - x_2\right)\left(2a - c - x_1 + x_2\right)\left(2a - c + x_1 - x_2\right)\left(2a - c + x_1 + x_2\right)... \\ ...\left(x_1 - a\right)\left(x_1 + a\right)\left(x_2 - a\right)\left(x_2 + a\right)\left(r - \sqrt{x_1^2 + x_2^2}\right) \end{aligned}$$

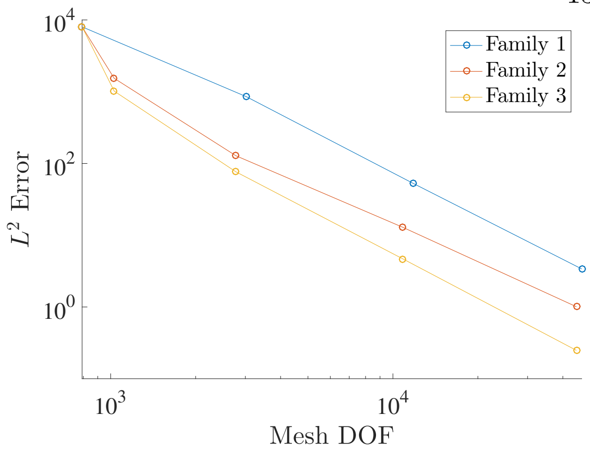
wherein a is the plate half width, r is the radius of the hole, and c is the length of the chamfer. We plot the convergence rate of the error in Fig. 5.16.

Table 5.9: Three mesh families for a plate with a hole and chamfered corners.

m	Family 1	Family 2	Family 3
1			
2			
3			



(a)



(b)

Figure 5.16: Convergence plots of the error in the L^2 norm. (a) Error versus mesh size. (b) Error versus nodal degrees of freedom in the mesh.

From Fig. 5.16a, we see that our proposed smoothing method (Family 3) outperforms both uniform subdivision (Family 1) and linear elastic smoothing (Family 2) in terms of convergence with respect to h. Furthermore, we note that the initial coarse mesh has small elements as a result of the chamfers at the corners of the plate. As such, the meshes in Family 1 become overly refined at the plate corners under uniform subdivision. If we instead plot solution error with respect to system degrees of freedom, as done in Fig. 5.16b, we see that the advantages of our proposed smoothing technique are even more dramatic. This is compelling, as we recognize that we may use our element distortion metrics to optimize higher order meshes. We plot element distortion metrics for the three families in Fig. 5.17.

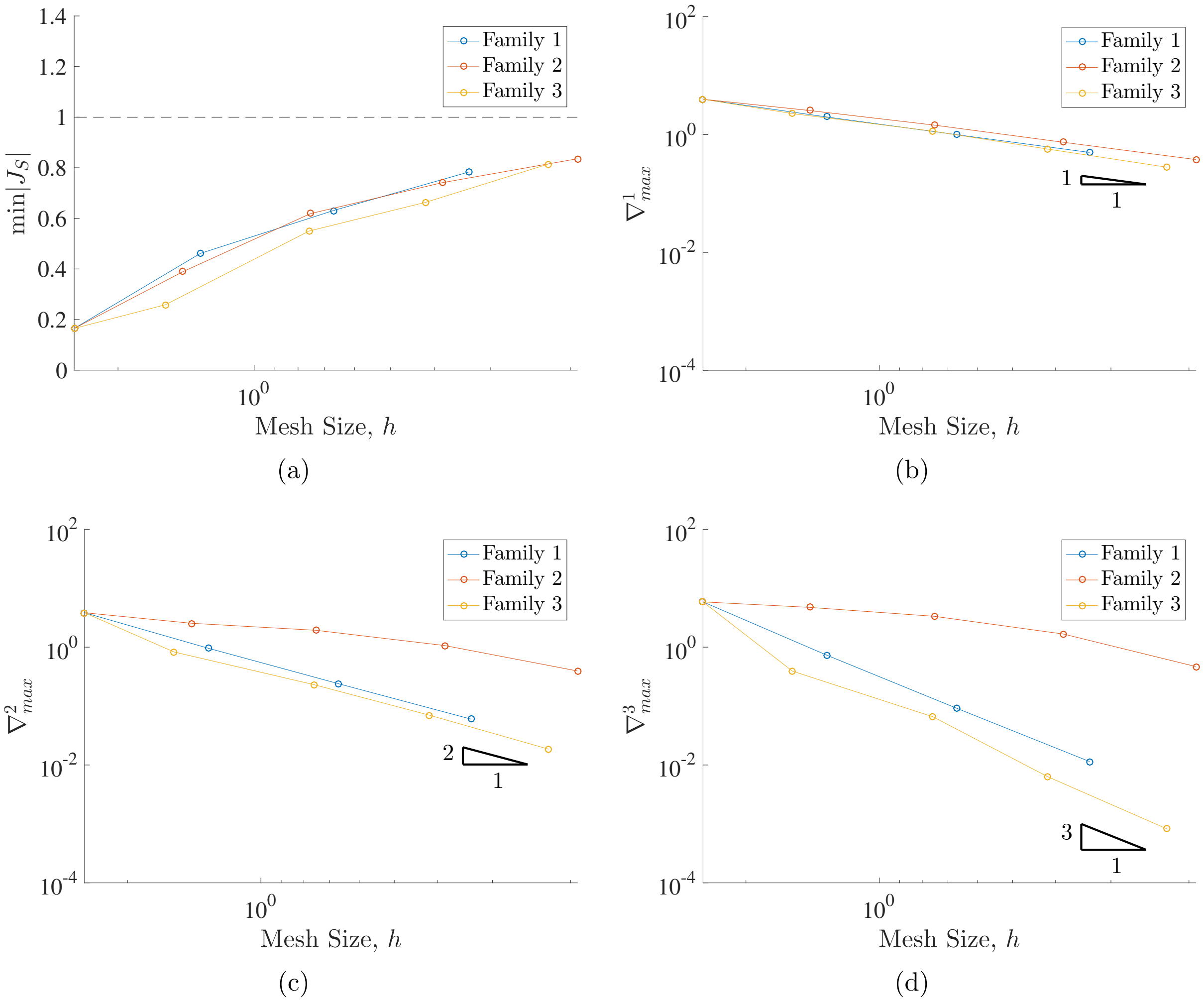


Figure 5.17: Mesh distortion metrics for the triangular meshes of the plate with a hole and chamfers. (a) Lower bound on the scaled Jacobian. (b) Upper bound on the first derivative. (c) Upper bound on the second derivative. (d) Upper bound on the third derivative.

Chapter 6

Conclusions

The goal of this dissertation was to address two outstanding problems in the fields of IGA and p-version finite elements. First, in an effort to mitigate the design-to-analysis bottleneck, we presented a framework for geometrically exact mesh generation. Second, we presented sufficient and computable element quality metrics that can be used to verify if curvilinear Bernstein–Bézier elements are analysis suitable. Generally, this work was successful in addressing both of these problems, at least from a theoretical point of view. The work on geometrically exact mesh generation resulted in academic publications on two-dimensional mesh generation using Bernstein–Bézier triangles [21] and volumetric mesh generation using mixed–element Bernstein–Bézier discretizations [22]. The work on element quality metrics is the subject of a forthcoming publication.

Moving forward, there are a variety of logical extensions to this work to integrate these theoretical results into software tools used in practice. In regard to the meshing framework presented here, the next logical step is to consider trimmed B-Reps. As stated in Section 4.3.1, we have limited our discussion to CAD objects that do not contain trimming curves. This is because trimming curves are implicitly defined, and cannot be represented exactly using Bézier curves. However, most geometries of engineering interest contain trimming curves, so a method that cannot handle geometries containing trimming curves proves to be rather limiting. To truly be able to handle trimmed B-Reps, the mesh generation framework presented here will need to be integrated into a fully featured commercial software.

Concerning element quality metrics, we note that the error bounds presented in Theorem I hold in the limit of mesh refinement. As a result, these can be rather loose upper bounds on the error, particularly over coarse meshes. As a further consequence of this, the conditions presented in Theorem II are sufficient but not necessary conditions for analysis suitability of rational Bernstein–Bézier elements. Thus, the element distortion metrics presented here are not **always** a good indicator of the effect of element distortion on solution accuracy. As an example, it has been observed that for boundary layer meshes, highly distorted elements yield better results per degree of freedom than their affine counterparts. We are curious to see if the error bounds here can be sharpened, particularly if something is known *a priori* about the PDE to be solved. We have also presented easily computable bounds on element distortion, and demonstrated their use in a few simple examples. However, the implementation of these metrics has not yet optimized for computational efficiency, and the metrics have not been benchmarked on large problems. We are curious to see how these metrics will perform when scaled to high performance computing and benchmarked against existing distortion metrics.

Despite these unanswered questions, this work has important implications for anyone working with curvilinear finite elements. Hopefully, this dissertation has successfully illuminated some of the challenges and subtleties of curvilinear mesh generation, and provided theoretical insights that will be prove useful in practical applications.

Bibliography

[1] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, and A.-V. Vuong, Swept Volume Parameterization for Isogeometric Analysis, in Mathematics of Surfaces XIII, E. R. Hancock, R. R. Martin, and M. A. Sabin, eds., no. 5654 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 19–44.

[2] M. Ainsworth, O. Davydov, and L. L. Schumaker, Bernstein-Bézier finite elements on tetrahedral–hexahedral–pyramidal partitions, Computer Methods in Applied Mechanics and Engineering, 304 (2016), pp. 140–170.

[3] H. Al Akhras, T. Elguedj, A. Gravouil, and M. Rochette, Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models, Computer Methods in Applied Mechanics and Engineering, 307 (2016), pp. 256–274.

[4] I. Babuška and A. Aziz, On the angle condition in the finite element method, SIAM Journal on Numerical Analysis, 13 (1976), pp. 214–226.

[5] P. L. Baehmann, S. L. Wittchen, M. S. Shephard, K. R. Grice, and M. A. Yerry, Robust, geometrically based, automatic two-dimensional mesh generation, International Journal for Numerical Methods in Engineering, 24 (1987), pp. 1043–1078.

[6] Y. Bazilevs, L. Beirão Da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli, Isogeometric Analysis: Approximation, stability and error estimates for h-refined meshes, Mathematical Models and Methods in Applied Sciences, 16 (2006), pp. 1031–1090.

[7] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering, 199 (2010), pp. 229–263.

[8] M. Bern, D. Eppstein, and J. Gilbert, Provably good mesh generation, in , 31st Annual Symposium on Foundations of Computer Science, 1990. Proceedings, Oct. 1990, pp. 231–241 vol.1.

[9] P. E. Bernard, J. F. Remacle, N. Kowalski, and C. Geuzaine, Frame field smoothness-based approach for hex-dominant meshing, Computer-Aided Design, 72 (2016), pp. 78–86.

[10] 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, International Journal for Numerical Methods in Engineering, 87 (2011), pp. 15–47.

[11] J. Bramble and S. Hilbert, Estimation of linear functionals on Sobolev spaces with application to Fourier transforms and spline interpolation, SIAM Journal on Numerical Analysis, 7 (1970), pp. 112–124.

[12] J. H. Bramble and S. R. Hilbert, Bounds for a class of linear functionals with applications to Hermite interpolation, Numerische Mathematik, 16 (1971), pp. 362–369.

[13] J. Chan and T. Warburton, A short note on a Bernstein-Bézier basis for the pyramid, arXiv:1508.05609 [math], (2015). arXiv: 1508.05609.

[14] L. P. Chew, Constrained delaunay triangulations, Algorithmica, 4 (1989), pp. 97–108.

[15] ——, Guaranteed-quality Mesh Generation for Curved Surfaces, in Proceedings of the Ninth Annual Symposium on Computational Geometry, SCG '93, New York, NY, USA, 1993, ACM, pp. 274–280.

[16] J. J. Chou, Higher order Bézier circles, Computer-Aided Design, 27 (1995), pp. 303–309.

[17] P. G. Ciarlet and P. A. Raviart, General Lagrange and Hermite interpolation in R^n with applications to finite element methods, Archive for Rational Mechanics and Analysis, 46 (1972), pp. 177–199.

[18] P. G. Ciarlet and P. A. Raviart, Interpolation theory over curved elements, with applications to finite element methods, Computer Methods in Applied Mechanics and Engineering, 1 (1972), pp. 217–249.

[19] G. M. Constantine and T. H. Savits, A multivariate Faà di Bruno formula with applications, Transactions of the American Mathematical Society, 348 (1996), pp. 503–520.

[20] S. Dey, R. M. O'Bara, and M. S. Shephard, Curvilinear mesh generation in 3D, in In Proceedings of the Eighth International Meshing Roundtable, John Wiley & Sons, 1999, pp. 407–417.

[21] L. Engvall and J. A. Evans, Isogeometric triangular Bernstein–Bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis, Computer Methods in Applied Mechanics and Engineering, 304 (2016), pp. 378–407.

[22] ——, Isogeometric unstructured tetrahedral and mixed-element Bernstein–Bézier discretizations, Computer Methods in Applied Mechanics and Engineering, 319 (2017), pp. 83–123.

[23] D. Engwirda, MESH2D.

[24] J. M. Escobar, J. M. Cascón, E. Rodríguez, and R. Montenegro, A new approach to solid modeling with trivariate T-splines based on mesh optimization, Computer Methods in Applied Mechanics and Engineering, 200 (2011), pp. 3210–3222.

[25] J. M. Escobar, R. Montenegro, E. Rodríguez, and J. M. Cascón, The meccano method for isogeometric solid modeling and applications, Engineering with Computers, 30 (2012), pp. 331–343.

[26] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate, Distortion and quality measures for validating and generating high-order tetrahedral meshes, Engineering with Computers, 31 (2015), pp. 423–437.

[27] P. George and H. Borouchaki, Construction of tetrahedral meshes of degree two, International Journal for Numerical Methods in Engineering, 90 (2012), pp. 1156–1182.

[28] C. Geuzaine and J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering, 79 (2009), pp. 1309–1331.

[29] M. F. Hardwick and R. L. Clay, Dart system analysis, tech. rep., Sandia National Laboratories, 2005.

[30] A. Haselbacher and J. Blazek, Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids, AIAA Journal, 38 (2000), pp. 2094–2102.

[31] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 4135–4195.

[32] S. R. Idelsohn, E. Oate, N. Calvo, and F. D. Pin, The meshless finite element method, International Journal for Numerical Methods in Engineering, 58 (2003), pp. 893–912.

[33] N. Jaxon and X. Qian, Isogeometric analysis on triangulations, Computer-Aided Design, 46 (2014), pp. 45–57.

[34] A. Johnen, J. F. Remacle, and C. Geuzaine, Geometrical validity of curvilinear finite elements, Journal of Computational Physics, 233 (2013), pp. 359–372.

[35] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes, An immersogeometric variational framework for fluidstructure interaction: Application to bioprosthetic heart valves, Computer Methods in Applied Mechanics and Engineering, 284 (2015), pp. 1005–1053.

[36] M.-J. Lai and L. L. Schumaker, Spline Functions on Triangulations, Cambridge University Press, Apr. 2007. Google-Books-ID: 6hvqGgbBmEoC.

[37] P. Lamata, I. Roy, B. Blazevic, A. Crozier, S. Land, S. A. Niederer, D. R. Hose, and N. P. Smith, Quality metrics for high order meshes : Analysis of the mechanical simulation of the heart beat, IEEE Transactions on Medical Imaging, 32 (2013), pp. 130–138.

[38] T. S. Lan and S. H. Lo, Finite element mesh generation over analytical curved surfaces, Computers & Structures, 59 (1996), pp. 301–309.

[39] X. Li, M. S. Shephard, and M. W. Beall, Accounting for curved domains in mesh adaptation, International Journal for Numerical Methods in Engineering, 58 (2003), pp. 247–276.

[40] X. Li, J. Zheng, T. W. Sederberg, T. J. R. Hughes, and M. A. Scott, On linear independence of T-spline blending functions, Computer Aided Geometric Design, 29 (2012), pp. 63–76.

[41] S. Lipton, J. A. Evans, Y. Bazilevs, T. Elguedj, and T. J. R. Hughes, Robustness of isogeometric structural discretizations under severe mesh distortion, Computer Methods in Applied Mechanics and Engineering, 199 (2010), pp. 357–373.

[42] L. Liu, Y. Zhang, T. J. R. Hughes, M. A. Scott, and T. W. Sederberg, Volumetric T-spline construction using Boolean operations, Engineering with Computers, 30 (2013), pp. 425–439.

[43] L. Liu, Y. Zhang, Y. Liu, and W. Wang, Feature-preserving T-mesh construction using skeleton-based polycubes, Computer-Aided Design, 58 (2015), pp. 162–172.

[44] R. Löhner and P. Parikh, Generation of three-dimensional unstructured grids by the advancing-front method, International Journal for Numerical Methods in Fluids, 8 (1988), pp. 1135–1149.

[45] X.-J. Luo, M. S. Shephard, R. M. O'Bara, R. Nastasia, and M. W. Beall, Automatic p-version mesh generation for curved domains, Engineering with Computers, 20 (2004), pp. 273–285.

[46] T. Martin, E. Cohen, and M. Kirby, Volumetric Parameterization and Trivariate B-spline Fitting Using Harmonic Functions, in Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, SPM '08, New York, NY, USA, 2008, ACM, pp. 269–280.

[47] D. J. Mavriplis and V. Venkatakrishnan, A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes, International Journal of Computational Fluid Dynamics, 8 (1997), pp. 247–263.

[48] P. Möller and P. Hansbo, On advancing front mesh generation in three dimensions, International Journal for Numerical Methods in Engineering, 38 (1995), pp. 3551–3569.

[49] N. Mos, J. Dolbow, and T. Belytschko, A finite element method for crack growth without remeshing, International Journal for Numerical Methods in Engineering, 46 (1999), pp. 131–150.

[50] J. T. Oden and J. N. Reddy, An Introduction to the Mathematical Theory of Finite Elements, Courier Corporation, May 2012. Google-Books-ID: 2ZOGwDyB1IgC.

[51] J. Parvizian, A. Dster, and E. Rank, Finite cell method, Computational Mechanics, 41 (2007), pp. 121–133.

[52] P.-O. Persson and J. Peraire, Curved mesh generation and mesh refinement using Lagrangian solid mechanics, 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, (2008).

[53] P.-O. Persson and J. Peraire, Curved mesh generation and mesh refinement using lagrangian solid mechanics, in Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit, vol. 204, 2009.

[54] R. Poya, R. Sevilla, and A. J. Gil, A unified approach for *a posteriori* high-order curved mesh generation using solid mechanics, Computational Mechanics, 58 (2016), pp. 457–490.

[55] H. Prautzsch, W. Boehm, and M. Paluszny, Bézier and B-Spline Techniques, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[56] J.-F. Remacle, J. Lambrechts, C. Geuzaine, and T. Toulorge, Optimizing the Geometrical Accuracy of 2d Curvilinear Meshes, Procedia Engineering, 82 (2014), pp. 228–239.

[57] O. Sahni, K. E. Jansen, M. S. Shephard, C. A. Taylor, and M. W. Beall, Adaptive boundary layer meshing for viscous flow simulations, Engineering with Computers, 24 (2008), pp. 267–285.

[58] H. Samet, The Quadtree and Related Hierarchical Data Structures, ACM Comput. Surv., 16 (1984), pp. 187–260.

[59] D. Schillinger, L. Ded, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, and T. J. R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Computer Methods in Applied Mechanics and Engineering, 249-252 (2012), pp. 116–150.

[60] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes, Isogeometric finite element data structures based on Bézier extraction of T-splines, International Journal for Numerical Methods in Engineering, 88 (2011), pp. 126–156.

[61] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes, Local refinement of analysis-suitable T-splines, Computer Methods in Applied Mechanics and Engineering, 213–216 (2012), pp. 206–222.

[62] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, T-splines and T-NURCCs, in ACM SIGGRAPH 2003 Papers, SIGGRAPH '03, New York, NY, USA, 2003, ACM, pp. 477–484.

[63] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, J.-F. Remacle, M. W. Beall, and R. M. O'Bara, ADAPT '03: Conference on Adaptive Methods for Partial Differential Equations and Large-Scale ComputationAdaptive mesh generation for curved domains, Applied Numerical Mathematics, 52 (2005), pp. 251–271.

[64] S. J. Sherwin and J. Peiró, Mesh generation in curvilinear domains using high-order elements, International Journal for Numerical Methods in Engineering, 53 (2002), pp. 207–223.

[65] J. R. Shewchuk, Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator, in Applied Computational Geometry Towards Geometric Engineering, M. C. Lin and D. Manocha, eds., no. 1148 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1996, pp. 203–222.

[66] H. Si, TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator, ACM Trans. Math. Softw., 41 (2015), pp. 11:1–11:36.

[67] D. Sokolov, N. Ray, L. Untereiner, and B. Lévy, Hexahedral-dominant meshing, Oct. 2015.

[68] H. Speleers and C. Manni, Optimizing domain parameterization in isogeometric analysis based on Powell-Sabin splines, Journal of Computational and Applied Mathematics, 289 (2015), pp. 68–86.

[69] H. Speleers, C. Manni, F. Pelosi, and M. L. Sampoli, Isogeometric analysis with Powell-Sabin splines for advection–diffusion–reaction problems, Computer Methods in Applied Mechanics and Engineering, 221–222 (2012), pp. 132–148.

[70] D. C. Thomas, M. A. Scott, J. A. Evans, K. Tew, and E. J. Evans, Bézier projection: A unified approach for local projection and quadrature-free refinement and coarsening of NURBS and T-splines with particular application to isogeometric design and analysis, Computer Methods in Applied Mechanics and Engineering, 284 (2015), pp. 55–105.

[71] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts, Robust untangling of curvilinear meshes, Journal of Computational Physics, 254 (2013), pp. 8–26.

[72] W. Wang, Y. Zhang, L. Liu, and T. J. R. Hughes, Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology, Computer-Aided Design, 45 (2013), pp. 351–360.

[73] T. Warburton, An explicit construction of interpolation nodes on the simplex, Journal of Engineering Mathematics, 56 (2006), pp. 247–262.

[74] S. Xia and X. Qian, Isogeometric analysis with Bézier tetrahedra, Computer Methods in Applied Mechanics and Engineering.

[75] S. Xia, X. Wang, and X. Qian, Continuity and convergence in rational triangular Bézier spline based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering, 297 (2015), pp. 292–324.

[76] Z. Q. Xie, R. Sevilla, O. Hassan, and K. Morgan, The generation of arbitrary order curved meshes for 3d finite element analysis, Computational Mechanics, 51 (2012), pp. 361–374.

[77] G. Xu, B. Mourrain, R. Duvigneau, and A. Galligo, Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications, Computer-Aided Design, 45 (2013), pp. 395–404.

[78] G. Xu, B. Mourrain, A. Galligo, and T. Rabczuk, High-quality construction of analysis-suitable trivariate NURBS solids by reparameterization methods, Computational Mechanics, 54 (2014), pp. 1303–1313.

[79] M. A. Yerry and M. S. Shephard, A Modified Quadtree Approach To Finite Element Mesh Generation, IEEE Computer Graphics and Applications, 3 (1983), pp. 39–46.

[80] S. Zeng and E. Cohen, Hybrid volume completion with higher-order Bézier elements, Computer Aided Geometric Design, 35–36 (2015), pp. 180–191.

[81] A. Ženíšek, Polynomial approximation on tetrahedrons in the finite element method, Journal of Approximation Theory, 7 (1973), pp. 334–351.

[82] Y. Zhang, W. Wang, and T. J. R. Hughes, Solid T-spline construction from boundary representations for genus-zero geometry, Computer Methods in Applied Mechanics and Engineering, 249 (2012), pp. 185–197.

[83] M. Zlámal, Curved elements in the finite element method. I, SIAM Journal on Numerical Analysis, 10 (1973), pp. 229–240.

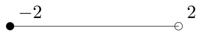
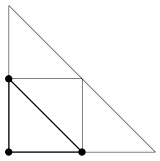
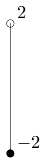
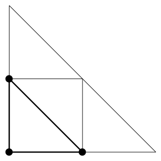
[84] ——, Curved elements in the finite element method. II, SIAM Journal on Numerical Analysis, 11 (1974), pp. 347–362.

Appendix A

Derivative Stencils

This appendix includes lookup tables for several common 2D Bernstein–Bézier elements. Each table contains stencils for every non-zero derivative over the respective element. Furthermore, the nodes at which to apply the given stencil are shown on a reference element. We do not include explicit stencils for 3D elements, as they are hard to visualize, but stencils for any 2D or 3D simplicial or tensor product element can be derived using the equations derived previously in this dissertation.

Table A.1: Partial derivative stencils for quadratic Bernstein–Bézier triangles.

First Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 1$					
Deriv.	Stencil	Evaluation Triangle	Deriv.	Stencil	Evaluation Triangle
$\dfrac{\partial\widetilde{\mathbf{x}}^e}{\partial\xi_1}$			$\dfrac{\partial\widetilde{\mathbf{x}}^e}{\partial\xi_2}$		

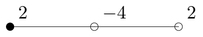
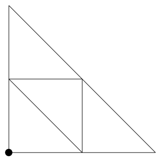
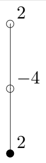
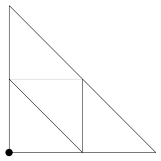
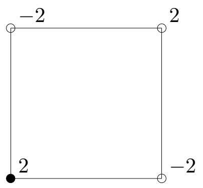
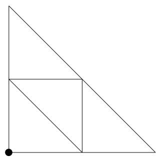
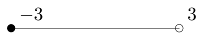
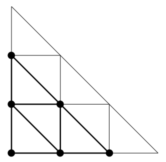
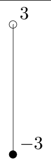
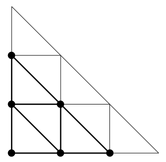
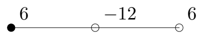
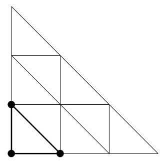
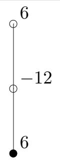
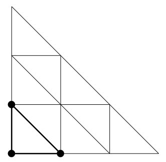
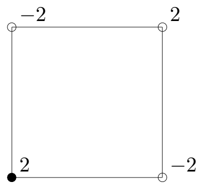
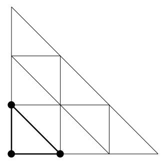
Second Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 2$					
Deriv.	Stencil	Evaluation Triangle	Deriv.	Stencil	Evaluation Triangle
$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1^2}$			$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_2^2}$		
$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1\partial\xi_2}$					

Table A.2: Partial derivative stencils for cubic Bernstein–Bézier triangles.

First Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 1$					
Deriv.	Stencil	Evaluation Triangle	Deriv.	Stencil	Evaluation Triangle
$\dfrac{\partial\widetilde{\mathbf{x}}^e}{\partial\xi_1}$			$\dfrac{\partial\widetilde{\mathbf{x}}^e}{\partial\xi_2}$		

Second Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 2$					
Deriv.	Stencil	Evaluation Triangle	Deriv.	Stencil	Evaluation Triangle
$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1^2}$			$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_2^2}$		
$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1\partial\xi_2}$					

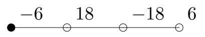
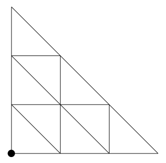
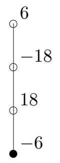
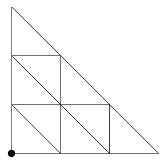
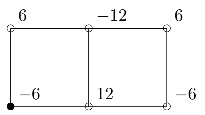
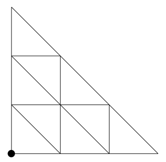
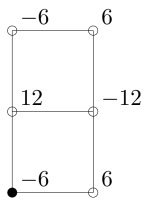
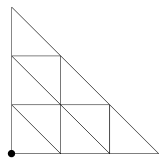
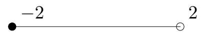
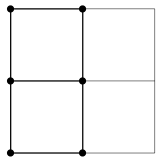
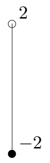
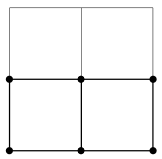
Third Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 3$					
Deriv.	Stencil	Evaluation Triangle	Deriv.	Stencil	Evaluation Triangle
$\dfrac{\partial^3\widetilde{\mathbf{x}}^e}{\partial\xi_1^3}$			$\dfrac{\partial^3\widetilde{\mathbf{x}}^e}{\partial\xi_2^3}$		
$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1^2\partial\xi_2}$			$\dfrac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1\partial\xi_2^2}$		

Table A.3: Partial derivative stencils for bi-quadratic Bernstein–Bézier quadrilaterals.

First Order Derivatives, $\|\boldsymbol{\alpha}\| = 1$					
Deriv.	Stencil	Evaluation Quadrilateral	Deriv.	Stencil	Evaluation Quadrilateral
$\dfrac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_1}$			$\dfrac{\partial \widetilde{\mathbf{x}}^e}{\partial \xi_2}$		

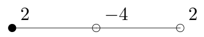
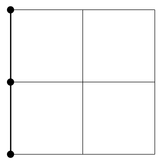
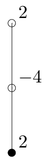
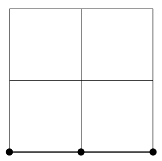
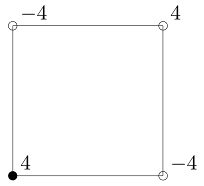
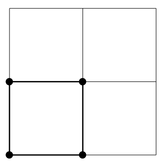
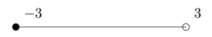
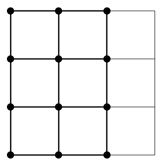
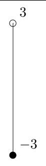
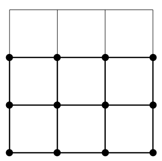
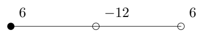
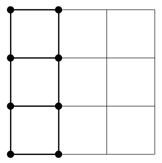
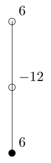
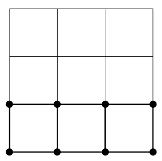
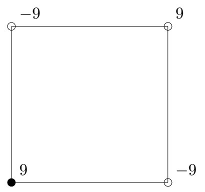
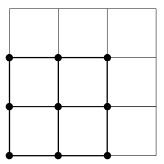
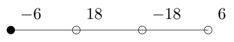
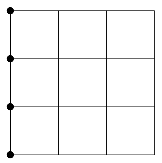
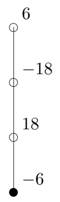
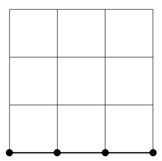
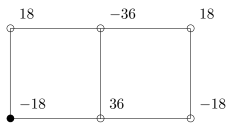
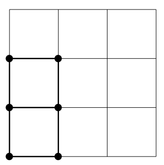
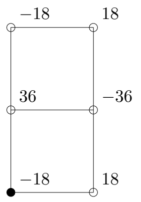
Second Order Derivatives, $\|\boldsymbol{\alpha}\| = 2$					
Deriv.	Stencil	Evaluation Quadrilateral	Deriv.	Stencil	Evaluation Quadrilateral
$\dfrac{\partial^2 \widetilde{\mathbf{x}}^e}{\partial \xi_1^2}$			$\dfrac{\partial^2 \widetilde{\mathbf{x}}^e}{\partial \xi_2^2}$		
$\dfrac{\partial^2 \widetilde{\mathbf{x}}^e}{\partial \xi_1 \partial \xi_2}$					

Table A.4: Partial derivative stencils for bi-cubic Bernstein–Bézier quadrilaterals.

First Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 1$					
Deriv.	Stencil	Evaluation Quadrilateral	Deriv.	Stencil	Evaluation Quadrilateral
$\frac{\partial\widetilde{\mathbf{x}}^e}{\partial\xi_1}$			$\frac{\partial\widetilde{\mathbf{x}}^e}{\partial\xi_2}$		

Second Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 2$					
Deriv.	Stencil	Evaluation Quadrilateral	Deriv.	Stencil	Evaluation Quadrilateral
$\frac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1^2}$			$\frac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_2^2}$		
$\frac{\partial^2\widetilde{\mathbf{x}}^e}{\partial\xi_1\partial\xi_2}$					

Third Order Derivatives, $\lvert\boldsymbol{\alpha}\rvert = 3$					
Deriv.	Stencil	Evaluation Quadrilateral	Deriv.	Stencil	Evaluation Quadrilateral
$\frac{\partial^3\widetilde{\mathbf{x}}^e}{\partial\xi_1^3}$			$\frac{\partial^3\widetilde{\mathbf{x}}^e}{\partial\xi_2^3}$		
$\frac{\partial^3\widetilde{\mathbf{x}}^e}{\partial\xi_1^2\partial\xi_2}$			$\frac{\partial^3\widetilde{\mathbf{x}}^e}{\partial\xi_1\partial\xi_2^2}$	 −18 18 36 −36 −18 18