Isogeometric Triangular Bernstein-Bézier Discretizations: Automatic Mesh Generation and Geometrically Exact Finite Element Analysis

Luke Engvall, John A. Evans

Aerospace Engineering Sciences, University of Colorado at Boulder

February 23, 2016

Abstract

Isogeometric analysis (IGA) was introduced as a way to bypass the design-to-analysis bottleneck inherent in the traditional Computer Aided Design (CAD) through Finite Element Analysis (FEA) paradigm. However, an outstanding problem in the field of IGA is that of surface-to-volume parameterization. In CAD packages, solid objects are represented by a collection of NURBS or T-spline bounding surfaces, but to perform engineering analysis on real world problems, we must find a way to parameterize the volumes of these objects as well. This has proven to be difficult using traditional IGA, as the tensor-product nature of trivariate NURBS and T-splines limit their ability to create analysis suitable parameterizations of arbitrarily complex volumes.

To overcome the limitations of trivariate NURBS and T-splines, we propose the use of rational Bernstein-Bézier tetrahedra to create analysis suitable volumetric parameterizations for isogeometric analysis. In this paper, which is part one of a two part series, we present the methodology for discretizing two dimensional geometries using rational Bernstein-Bézier triangles. In addition to presenting finite element analysis methodologies based on rational Bernstein-Bézier triangles, we also introduce two new mesh generation strategies for automatically creating high quality, geometrically exact curvilinear meshes. We assess the quality of our mesh generation schemes using a suite of challenging two-dimensional geometries, and we verify the accuracy of our proposed numerical discretization approach using the method of manufactured solutions.

Contents

1	Introduction	3
2	B-splines, NURBS and Bézier Extraction	6
2.1	B-splines	6
2.2	NURBS	7
2.3	Bernstein Polynomials and Bézier Extraction	8
3	Rational Bézier Triangles	10
4	Two-Dimensional Mesh Generation	11
4.1	Lifted Surface	12
4.2	Dynamic Quadtree	14
4.3	Comparison of Techniques	19
4.4	Control Point Weights	20
4.5	Assurance of Mesh Quality	20
5	Mesh Refinement	23
6	Numerical Examples	27
6.1	Plate with a Hole	27
6.2	Quarter Annulus	27
6.3	Potential Flow over an Airfoil	29
7	Conclusions	31
8	Acknowledgements	31
A	Appendix: Flowcharts and Algorithms	32
A	Lifted Surface Flowchart	32
B	Dynamic Quadtree Flowchart	33
C	Algorithms	35
D	Impact of Mesh Quality on Approximation	37

1 Introduction

In the product design cycle, designers and engineers will often iterate over many design permutations. A product is designed, some form of engineering analysis is performed, and the analysis in turn informs the next design iteration. With the ever increasing computational resources available to the modern engineer, Computer Aided Design (CAD) and Finite Element Analysis (FEA) are becoming more and more prevalent in this design cycle.

Ideally, engineers should be able to easily leverage these tools in tandem to converge on optimal designs. That is, there should be easy and seamless transfer of information between CAD and FEA. However, in the traditional CAD to FEA paradigm, this is not the case. This disconnect between CAD and FEA leads to what has been termed the Design-to-Analysis bottleneck.

Isogeometric Analysis (IGA) was conceived in 2005 as a solution to this very problem [15]. By using the functions native to CAD packages (NURBS) as the basis for finite element analysis, the tight link between CAD and FEA is established. IGA has been applied to many problems of engineering interest with promising results. However, traditional isogeometric analysis still suffers limitations in representing arbitrary volumes. This is because CAD packages represent solid objects using a collection of NURBS and/or T-spline bounding surfaces, but there is no numerical parameterization of the interior domain. Indeed, this proves to be a major drawback to isogeometric analysis, as its ability to remove the design-to-analysis bottleneck is inhibited for many problems of real engineering interest, including problems arising in acoustics, electromagnetics, and fluid dynamics.

In the field of isogeometric analysis, this aforementioned issue is referred to as “surface-to-volume parameterization”. Roughly speaking, the goals of surface-to-volume parameterization are as follows:

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

1. *The parameterization exactly matches the geometry.*
2. *The parameterization is analysis suitable.*¹
3. *The parameterization is defined by piecewise polynomial or rational basis functions that satisfy linear independence, partition of unity, positivity, and higher-order continuity.*

The three properties listed above are necessary for a parameterization to be suitable for isogeometric analysis. The first property is obviously necessary to ensure that the exact geometry is employed in analysis, while the final two properties ensure that an analysis framework built on the parameterization is well-posed [19], robust [20] and exhibits optimal convergence rates [5].

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 these authors that none of the previous work successfully addresses all of the goals listed above. Presented below is a short summary

¹We say a parameterization is analysis suitable if it is both invertible and well-conditioned. A parameterization is invertible if the element-wise Jacobian is bounded from both above and below. A parameterization is well-conditioned if it preserves optimal convergence rates. See Appendix D for more details.

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 [2]. 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. [27], 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, it is worth noting that any method for volume parameterization using trivariate NURBS or B-splines suffers from the same limitations as structured hexahedral mesh generation for classical FEM, and are often unable to automatically generate analysis suitable meshes of complex geometries.

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. [9]. 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 [36]. This method has been successful in automatically parameterizing a wide variety of complex volumes, however still suffers from poor mesh quality in some cases. Liu et. al. proposed a method for T-spline construction using Boolean operations [22], 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 [23], 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.

Yet 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 [39]. George et. al. proposed a method for generating tetrahedral meshes using 10-node Bézier tetrahedra [11]. 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.

From this cursory review of the state of the art, it is evident that there is still room for active research in the realm of surface-to-volume parameterization. While this paper only considers two dimensional problems, the clear motivation is to solve the problem of surface-

to-volume parameterization in three dimensions. In this spirit, this paper will focus on solving the analogous two-dimensional problem of curve-to-area parameterizations, which will provide the background necessary for moving to the three-dimensional problem in part two of this series.

We propose to solve the surface-to-volume parameterization problem by generalizing classical isogeometric methods to unstructured triangular and tetrahedral meshes. This paper outlines our approach for generating two dimensional meshes using rational Bernstein-Bézier triangles, while part two will address the surface-to-volume parameterization problem using meshes composed of rational Bernstein-Bézier tetrahedra. Much like NURBS surfaces, rational triangular Bézier surfaces are a mature and established technology in the CAD community [10, 13, 21, 14]. However, in the CAD community, Bézier triangles are most commonly used to create a three dimensional surface mesh from an existing three-dimensional NURBS surface, or to fit a surface mesh to discrete data points [35, 25, 24, 4]. The main emphasis is placed on the visual representation of the surfaces of objects, and as a result, volume parameterization and analysis suitability are not well studied problems in the CAD community. As with traditional isogeometric analysis, our work draws inspiration from these established technologies, but with the added focus of creating analysis suitable discretizations for both surfaces and volumes.

Using triangular meshes for isogeometric analysis is by no means a new idea. Speleers et. al. have used Powell-Sabin splines over triangular meshes to solve advection-diffusion-reaction problems [33, 32]. 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 [16]. Recent work by the same group has demonstrated that optimal convergence rates are achieved when solving Poisson and linear elasticity problems on discretizations composed of these rational Triangular Bézier Splines [37].

While our work draws much of its inspiration from the related previous work, 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 triangular meshes, while we choose instead to focus on generating meshes of C^0 continuous Bernstein-Bézier triangles. The advantage to this approach is simplicity in both implementation and automation, as well as it being readily generalizable to three dimensional meshing. In a sense, our approach can be viewed as geometrically exact degree elevation of existing linear meshes, with a main implication being the ability to leverage a wide array of existing meshing technologies. Using this approach, we demonstrate that we are able to produce analysis suitable meshes for a broad range of challenging geometries. Additionally, we present implementation details within the current work and we have provided a functioning open source code in a GitHub repository at: <https://github.com/CMGLab/TriGA>.

Unstructured triangular mesh generation is a well established field, which is advantageous for generalizing isogeometric analysis to triangular meshes, as there is a large body of literature to draw inspiration from. This work leverages existing meshing software as much as possible, and there exist a plethora of open-source meshing packages such as mesh2d [1] and Triangle [31]. Some key algorithms that underlie these meshers, and that we use in the

present work, are quadtree decompositions [38, 30, 3], and Delaunay triangulation [8, 6]. Additionally, it is worth mentioning that 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 [12, 17, 18, 26, 29, 34]. By no means are the works cited above the extent of the field of triangular mesh generation, but a complete review of the state of the art would be untenable. Instead, we have listed a few impactful and relevant papers, and direct the interested reader to the references above for more information.

The remainder of this paper will be organized as follows. First, mathematical preliminaries regarding B-splines and NURBS will be presented, as well as the concept of Bézier extraction and the resulting Bernstein polynomial basis. Then, we will provide the mathematical description of rational Bernstein-Bézier triangles that will compose our two dimensional meshes. Next, we will provide an in-depth description of our meshing processes as well as our method for performing mesh refinement. We will also present a study of mesh quality for a suite of challenging two-dimensional geometries. Finally, we will present some preliminary numerical results and convergence studies.

2 B-splines, NURBS and Bézier Extraction

2.1 B-splines

B-splines, or Basis-splines, are basis functions that, as their name implies, form a basis for representing any piecewise polynomial of equal degree and smoothness. The B-spline basis functions are defined by their knot vector, a set of non-decreasing coordinates in parameter space, $\Xi = [\xi_1 \xi_2 \dots \xi_k]$. The length of the knot vector is calculated as $k = n + p + 1$, where p is the polynomial degree of the basis and n is the number of basis functions. For a given knot vector, B-spline basis functions of degree p are defined via the Cox-deBoor recursion formula as:

$$N_{i,0}(\xi) := \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$N_{i,p}(\xi) := \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \quad \forall \xi \in [\xi_1, \xi_k] \quad (2)$$

B-splines have several favorable properties that make them an ideal basis for the finite element method [15]. These include partition of unity, point-wise non-negativity, linear independence, compact support, and control of continuity.

A B-spline curve in $\mathbb{R}^d$, where d is the number of spatial dimensions, is defined as a linear combination of B-splines basis functions multiplied by control points, $\mathbf{P}_i$, in $\mathbb{R}^d$:

$$\mathbf{C}(\xi) = \sum_{i=1}^k N_{i,p}(\xi) \mathbf{P}_i \quad (3)$$

Two important properties of B-spline curves that make them well suited for representing geometry are affine covariance and a strong convex hull. Figure 1 shows an example quadratic

B-spline basis for the knot vector $\Xi = [0 \ 0 \ 0 \ 1 \ 1 \ 2 \ 3 \ 4 \ 4 \ 4]$ and a corresponding B-spline curve. Multi-variate B-splines are simply defined as the tensor product of univariate B-splines. In this way, a B-spline surface is defined as:

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^k \sum_{j=1}^l N_{i,p}(\xi) N_{j,q}(\eta) \mathbf{P}_{ij} \quad (4)$$

Figure 1: (a) The B-spline basis functions and (b) the corresponding NURBS curve.

2.2 NURBS

One drawback of B-splines is that they are unable to exactly represent certain geometries of engineering interest, such as conic sections (circles, ellipses, parabolas, and hyperbolas). NURBS, or Non-Uniform Rational B-splines, are a generalization of B-splines that address this issue, and are the de-facto standard in computer graphics and CAD packages. A NURBS curve in $\mathbb{R}^d$ is constructed from a projective mapping of a B-spline curve in $\mathbb{R}^{d+1}$ to $\mathbb{R}^d$. A B-spline is first constructed in $\mathbb{R}^{d+1}$, or projective space, with the control points $\mathbf{P}_i^w$. The NURBS control points in physical space, $\mathbf{P}_i$, and their corresponding weights, w_i , are then calculated as:

$$(\mathbf{P}_i)_j = \frac{(\mathbf{P}_i^w)_j}{w_i} \quad (5)$$

$$w_i = (\mathbf{P}_i^w)_{d+1} \quad (6)$$

NURBS basis functions are then defined as the projective transformation of B-spline basis functions, viz.

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi)w_i}{\sum_{j=1}^k N_{j,p}(\xi)w_j} \quad (7)$$

while NURBS curves are defined as the projective transformation of B-spline curves, viz.

$$\mathbf{C}(\xi) = \sum_{i=1}^k R_{i,p}(\xi) \mathbf{P}_i \quad (8)$$

The NURBS basis inherits all the nice properties of the B-spline basis, and NURBS curves inherit two important properties of B-spline curves, namely: affine covariance and a strong convex hull (provided that the weights are non-negative).

Figure 2 shows the NURBS Basis functions for $\Xi = [0 \ 0 \ 0 \ 2 \ 2 \ 3 \ 4 \ 5 \ 5 \ 5]$. The control net for the NURBS curve in Figure 2 is the same as for the B-spline in Figure 1. However, a weight of 3 is applied to the 6th control point, leading to a change in both the basis and shape of the curve.

Figure 2: (a) The NURBS basis functions and (b) the corresponding NURBS curve.

2.3 Bernstein Polynomials and Bézier Extraction

The special case of evaluating the B-spline basis functions over the knot vector $[\underbrace{0 \dots 0}_{p+1 \text{ times}} \underbrace{1 \dots 1}_{p+1 \text{ times}}]$ yields the Bernstein polynomials:

$$B_{i,p}(x) = \binom{p}{i} x^i (1-x)^{p-i} \quad (9)$$

The corresponding B-spline curve defined over the knot span $[0 \ 1]$ is known as a Bezier Curve. In the case of evaluating a NURBS curve on the knot vector $[\underbrace{0 \dots 0}_{p+1 \text{ times}} \underbrace{1 \dots 1}_{p+1 \text{ times}}]$, this yields a rational Bézier curve.

Bézier extraction is a powerful tool that allows NURBS curves to be expressed as a collection of C^0 -continuous rational Bézier curves. Following Borden et. al. [7], we can define the relationship between the NURBS basis and the Bernstein basis via an extraction operator, $\mathbf{C}$. We can then likewise define the relationship between the NURBS control points, $\mathbf{P}$, and the Bézier control points, $\mathbf{P}^b$, as:

$$\mathbf{P}^b = (\mathbf{W}^b)^{-1} \mathbf{C}^T \mathbf{W} \mathbf{P} \quad (10)$$

where $\mathbf{W}$ and $\mathbf{W}^b$ are diagonal matrices of the form:

$$\mathbf{W} = \begin{bmatrix} w_1 & & & & \\ & w_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & w_n \end{bmatrix} \quad (11)$$

$$\mathbf{W}^b = \begin{bmatrix} w_1^b & & & & \\ & w_2^b & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & w_n^b \end{bmatrix} \quad (12)$$

where $\mathbf{w} = (w_1, \dots, w_n)$ are the weights of the NURBS control points, and $\mathbf{w}^b = (w_1^b, \dots, w_n^b)$ are the weights of the Bézier curves. These $\mathbf{w}^b$ are found by the equation:

$$\mathbf{w}^b = \mathbf{C}^T \mathbf{w} \quad (13)$$

Figure 3 shows the results of performing Bézier extraction on the NURBS Curve from Figure 2. The NURBS curve is refined to a collection of four rational Bézier curves. The NURBS basis is also refined, and is made up of the rational Bernstein-Bézier polynomials over each knot span.

Figure 3: (a) The extracted rational Bernstein basis functions, and (b) the extracted rational Bézier curves.

The preceding discussion illustrates Bézier extraction from a global standpoint, but extraction can also be performed on a local element level. See Borden et. al. for more details [7]. Bézier extraction and the resulting Bernstein basis are extremely powerful in the context of finite element analysis. Through extraction, we are able to achieve completely local representations of both our geometry and our basis from the global NURBS curves and basis functions that define the geometry. This is a powerful tool for finite element methods, as the basis functions are evaluated locally, and then assembled to form the global system of equations.

3 Rational Bézier Triangles

In two dimensions, we solve the problem of curve-to-area parameterization by creating an unstructured mesh of rational Bézier triangles. A rational Bézier triangle is defined as a linear combination of Bernstein basis functions multiplied by control points that form a triangular control net. Namely, we express a rational Bézier triangle as:

$$T(\boldsymbol{\lambda}) = \sum_{i+j+k=p} R_{ijk}(\boldsymbol{\lambda}) \mathbf{P}_{ijk} \quad (14)$$

where the rational Bézier basis functions are defined as:

$$R_{ijk}(\boldsymbol{\lambda}) = \frac{B_{ijk}(\boldsymbol{\lambda}) w_{ijk}}{\sum_{r+s+t=p} B_{rst}(\boldsymbol{\lambda}) w_{rst}} \quad (15)$$

and the Bernstein polynomials over a triangle are defined as:

$$\begin{aligned} B_{ijk}(\boldsymbol{\lambda}) &= \frac{p!}{i!j!k!} \lambda_1^i \lambda_2^j \lambda_3^k \\ i + j + k &= p \\ i, j, k &\in [0, p] \end{aligned} \quad (16)$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]$ are the barycentric coordinates. Note that for clarity, we have dropped the polynomial degree index p from the basis function definitions B_{ijk} and R_{ijk} . For a cubic Bézier triangle, the control net is defined by 10 control points and their respective weights. The control net for the cubic unit triangle is shown in Figure 4, as well as the control net for a representative curvilinear triangle. In the context of finite elements, these are the parent and physical elements, respectively.

Figure 4: Control nets in (a) parameter space and (b) physical space.

Note that the edges of the cubic Bézier Triangles are simply cubic Bézier curves. Because of this, we can easily construct the perimeter of a cubic Bézier triangle from three cubic Bézier curves, provided that the convex hulls of the curves do not intersect. Then, the full control net of the triangle is completed by picking interior control points that yield a well conditioned triangle.

The cubic triangular Bernstein polynomials defined over the two triangles are also shown in Table 1 as 2-D heatmaps. Note that the axes have been omitted for clarity.

Table 1: The cubic triangular Bernstein polynomials basis.

n	Parent	Physical	n	Parent	Physical
1			6
2			7
3			8
4			9
5			10

4 Two-Dimensional Mesh Generation

With the necessary mathematical background established, we now turn to the problem of mesh generation. Recall from the introduction that the end goal of this work is to automatically generate volumetric parameterizations of complex volumes from a set of bounding surfaces. These parameterizations must be analysis suitable and exactly match the problem geometry. As a stepping stone towards volumetric parameterizations, this paper considers

the problem of creating a 2D parameterization of an area, based off a set of NURBS bounding curves. Like the 3D case, this parameterization must be analysis suitable and exactly match the problem geometry.

Unstructured triangular meshes offer several distinct advantages over tensor-product meshes. First, triangular meshes can be used to parameterize domains of arbitrary topology. To do this with NURBS surfaces requires using multiple patches, while T-spline representations of arbitrary topologies often require hanging nodes, or “T-junctions”. Additionally, with triangular meshes it is possible to generate locally isotropic meshes, even when the domain itself is globally anisotropic. This can often be impossible with NURBS surfaces, and again requires the use of hanging nodes when using T-splines. Finally, and most importantly, automatic mesh generation using triangles is a mature field, with many robust meshing algorithms. As of yet, such robust and automatic algorithms do not exist for automatically generating NURBS or T-spline parameterizations. Additionally, it is worth noting that it is relatively straight-forward to extend the concepts developed here to generating tetrahedral meshes in three dimensions, as will be presented in part two of this series.

We have developed two new algorithms for generating unstructured triangular meshes in 2D. Both algorithms are presented here, and their relative strengths and weaknesses are discussed. For the examples presented here, we have considered cubic Bernstein-Bezier triangles for illustrative purposes, but the methods presented here are easily extended to arbitrary polynomial degree. In addition to the textual descriptions provided below, we have also included several key flowcharts and algorithms in the appendix. Flowcharts are provided to present a general overview of the two algorithms presented below, and we have also included MATLAB style pseudocode to provide a more in-depth explanation where necessary. Note that the algorithms included are limited to the new contributions presented here. For example, we do not provide algorithms for Bézier extraction, as this can be found in the previous literature. In conjunction with this work, we have also started the TriGA project (for Triangular IGA). An open git repository is available for download and contains a beta version of our dynamic quadtree decomposition method.

4.1 Lifted Surface

The method of lifted surfaces works by creating a layer of curved rational Bernstein-Bézier triangles around the NURBS curves that define the geometry, and then using existing meshing technologies to create a straight sided mesh of the remaining area. In its most basic form, the lifted surface method can be broken into 3 steps:

1. For each NURBS curve, create a layer of curved triangles around the curve, and connect their vertices to create a polygon enclosing the curve.
2. Feed the polygons from Step 1 to a linear mesh generator to generate a mesh of the rest of the domain.
3. Elevate the linear mesh from Step 2 to mesh of degree p .

We illustrate this method below on the simple example geometry of a plate with a hole. Figure 5 shows the NURBS curves that define the problem geometry. For this example, and in the pseudocode included in the appendix, we create a cubic rational Bézier mesh, but the methodology is readily modified to create a mesh of arbitrary polynomial degree. We

chose to demonstrate our meshing capabilities on cubic curves, as cubic curves are standard in CAD packages, and most engineering geometries can be represented with a collection of cubic curves.

Step 1: The meshing process begins by subdividing the given NURBS curves into a user specified number of cubic rational Bézier Curves. For this example, we choose to divide the circle into 8 cubic Rational Bézier curves. This is done by performing knot insertion at 8 equi-spaced knot locations around the circumference of the circle, and performing degree elevation to obtain cubic curves. Figure 5 shows the control points of the resulting curves.

Figure 5: (a) The curves defining the plate with a hole. (b) The control net for the original NURBS curve defining the hole. (c) The control net for the hole after Bézier extraction and degree elevation.

Once the given curves have been discretized, we create a layer of rational Bézier triangles around our curve by defining a set of points lifted off of the surface (see `Create.Triangle` in Appendix C for details). Figure 6 shows these lifted points and the resulting control nets over each of the triangles defined by these lifted points and their respective Bézier curves. Connecting the lifted vertices defines a Planar Straight Line Graph (PSLG), which can be easily passed to existing meshing technologies.

Figure 6: The three step process for generating a layer of lifted triangles. (a) Find the lifting direction. (b) Generate the lifted points. (c) Generate the control nets.

Step 2: The next step is to simply call a 3rd party mesher (we used mesh2D by Darren Engwirda [1]) to create a mesh of the rest of the domain. Figure 7 shows the PSLG and the resulting mesh from mesh2D.

Step 3: Finally, we create cubic Bézier triangles from the resulting linear mesh by laying a control net of 10 control points down over each triangle (see Elevate_Triangle in Appendix C). It remains to specify the weights for the interior control points. One valid possibility is to enforce that the weights of all of the interior control points be one, but we have found that this method inhibits convergence [28]. An alternative methodology that yields better convergence characteristics will be discussed after our presentation of the dynamic quadtree method.

Figure 7: (a) The Planar Straight Line Graph generated by connecting the lifted points. (b) The final mesh.

The lifted surface technique has a variety of attractive advantages, such as precise control of the surface mesh. However, this method is not fully automated, and can fail if the user incorrectly specifies the size of the surface mesh. Correctly specifying the size of the surface mesh can be particularly difficult for complex geometries, and for domains that are highly anisotropic. Figure 8 shows some examples of these geometries, such as the nozzle, which is a highly anisotropic domain, or the flange, which has complex topology.

4.2 Dynamic Quadtree

The dynamic quadtree decomposition method provides a more robust and automatic meshing capability over the lifted surface method. For example, the nozzle geometry in Figure 8 requires extensive local refinement around the tight constriction to accurately capture the geometry and simultaneously preserve mesh quality. However, much coarser

triangles can be used near the inflow and outflow to limit mesh complexity. The dynamic quadtree method accounts for this by borrowing the idea of a quadtree decomposition from traditional mesh generation [38, 30, 3].

Figure 8: More challenging geometries. (a) A nozzle with a tight constriction. (b) A flange.

The dynamic quadtree method consists of the following steps:

1. Construction of a sufficiently accurate polygon approximation of the geometry.
2. Construction of a quadtree background mesh use to further refine the polygon approximation to capture small geometry features.
3. Construction of a linear triangular mesh of the polygon approximation.
4. Conversion of the linear triangular mesh to a geometrically exact triangular Bernstein-Bézier mesh.

These steps will be discussed in further detail below. Additionally, Appendix B contains a flowchart outlining the method. Note that both the flowchart and the textual description are segmented to align with the steps outlined above.

Step 1: Given an arbitrary collection of NURBS curves, knots are inserted at the midpoint of each knot span, and a new knot vector is created. Then, the physical coordinates on the NURBS curve corresponding to each of the knots in the newly formed knot vector are calculated, as well as a connectivity array that defines a polygon that approximates the NURBS curve. Then, the mesher calculates the length of each section of the NURBS curve defined by each knot span, as well as the length of the edge of the polygon that approximates it. If the percentage difference in lengths is above some threshold (default of 1%), a knot is again inserted at the midpoint of that knot span. This process continues until a polygon that matches the exact geometry sufficiently closely is generated. Figure 9 shows how this method iterates until a adequate polygon is generated. The exact geometry is in blue and the polygon is in red.

Step 2: Once we have a polygon that approximates the geometry sufficiently well, we generate a quadtree background mesh of the polygonal approximation using the `quadtree.m`

Figure 9: Two iterations of generating a polygonal approximation. (a) The original polygonal approximation. (b) The once-refined approximation used to generate the quadtree decomposition.

function in mesh2d. With this mesh generated, the sizing function of each side of the polygon is calculated. In our method, we use the linear sizing function used in mesh2d, but a variety of different sizing functions could be chosen. If the ratio of the size function to the length of the side is above a threshold of 1.5, a new knot is inserted at the midpoint of the corresponding knot vector, the control point corresponding to that knot is calculated, and a new polygon is formed. The sizing functions are once again calculated for all edges of the new polygon. This process repeats until the sizing functions of each side of the polygon are within acceptable limits. Figure 10 shows a few iterates of this process.

Step 3: Finally, with a polygon that is sufficiently refined to mesh, a mesh of the area is generated using meshpoly.m, which is part of mesh2d. Figure 11 shows the resulting linear mesh. While we use mesh2d for our linear meshing, it is worth noting that any linear mesher can be used for this step, with slight modification. As an aside, we note that generating a linear mesh from a curved domain as we have done in Steps 1-3 is a standard capability in many existing commercial and open source mesh generation softwares, and we do not claim that the above steps are an original contribution. We have simply described the process that we use in full in order to illuminate our entire meshing procedure.

Step 4: The last step is to convert our linear mesh generated by meshpoly.m to a higher order triangular mesh that inherits its geometry from the NURBS curve. This is done by first degree elevating all of the linear triangles in the mesh through linearly interpolation of their three vertices. This defines the locations of seven more control points to make a 10 point control net defining a cubic Bézier triangle. This process is described algorithmically by Elevate_Triangle in Appendix C, and shown graphically in Figure 12. Once the linear mesh has been degree elevated, knot insertion is performed on the bounding NURBS curve until the knots corresponding to each vertex of the bounding polygon have multiplicity p . Now each knot span of the NURBS curve defining the geometry boundary has 4 control points, corresponding to the side of one of the triangles in the mesh. We now simply replace the control points on that edge generated by linear interpolation with the control points generated from knot insertion and degree elevation of the NURBS curve, and we have generated a higher order triangular mesh that exactly matches the geometry. Figure 13 illustrates this edge replacement procedure on an example triangle, and Figure 14 shows the resulting curvilinear nozzle mesh, in blue, laid over the linear mesh (red).

Figure 10: The quadtree refinement process. (a) The polygonal approximation and the corresponding quadtree decomposition. (b) Detail of the polygon and quadtree decomposition. (c-e) Three successive levels of refinement.

Figure 11: The linear mesh of the nozzle.

Figure 12: The degree elevation process for a straight sided triangle. (a) A linear triangle. (b) Find two nodes on each side by linear interpolation. (c) Find the middle node at the midpoint of the line connecting node 6 and node 9.

Figure 13: The edge replacement process. (a) The straight-sided, higher-order triangle. (b) The cubic Bézier curve corresponding to the bottom edge of the triangle. (c) The curvilinear triangle resulting from edge replacement.

Figure 14: (a) The final curvilinear nozzle mesh, and (b) a detailed view.

4.3 Comparison of Techniques

As mentioned above, the two meshing algorithms outlined in this paper have distinct advantages and disadvantages, which are outlined in Table 2, with Figure 15 showing a comparison of meshes generated by each method. Note that the lifted surface method succeeds in generating a much sparser mesh than the dynamic quadtree method. However, we have noted in the table that the lifted surface meshing technique is not automatic. This is because the surface mesh resolution must be specified by the user, which is done automatically in the dynamic quadtree method.

It is worth noting that efforts could be made to blend these methods to leverage their relative strengths and weaknesses. Indeed, mesh generation in general is a rich field and there are almost limitless opportunities for developing new methodologies, and adding new functionality to meshing software. However, in the interest of limiting the scope of this paper, the remainder of our discussion will focus predominantly on our dynamic quadtree method, while future work could focus on adding more functionality to one or both of these methods.

Figure 15: A comparison of meshes generated by (a) the lifted surface method, and (b) the dynamic quadtree method.

Table 2: Comparison of meshing techniques.

	Lifted Surface	Dynamic Quadtree
Exact Geometry	x	x
Analysis Suitable	x	x
Automatic		x
User Control of Boundary Mesh	x	

4.4 Control Point Weights

As previously mentioned, care should be taken in selecting control point weights so as to not inhibit convergence. In fact, it has been shown analytically that the variation of control points across the domain should be minimized to achieve quality numerical results [28, 5]. To guarantee that this variation is as small as possible in our meshes, we first create a mesh of the domain using either of the two meshing algorithms described above, but do not calculate the weights at the interior control points. We then solve the Laplace problem for the weights over the non-weighted domain, using the weights of the control points on curvilinear edges as Dirichlet boundary conditions. Figure 16 shows the results of this smoothing of control point weights over a unit circle. It should be noted that this is not mesh smoothing in the traditional sense: we smooth the control point weights, but are not moving the control points in physical space. The goal of this is to make the mapping as polynomial as possible; we allow for variation of control point weights on curvilinear boundaries, where it is necessary to accurately capture geometry, but simultaneously try to limit the variation in weights on the interior of the domain.

Figure 16: Smoothed weights over the unit circle.

4.5 Assurance of Mesh Quality

No paper describing a new meshing technique would be complete without a discussion of mesh quality. As the methods outlined in this paper rely heavily on existing linear mesh generation technologies, metrics of linear mesh quality are a good early indicator of higher order mesh quality. However, linear mesh quality is a necessary, but not sufficient, condition for higher order mesh quality. We consider two additional metrics of mesh quality:

$$J = \min_{\Delta^e \in \Delta_M} J^e \quad (17)$$

$$S = \max_{\Delta^e \in \Delta_M} S^e \quad (18)$$

Where the element-wise metrics, J^e and S^e , are defined as:

$$S^e = \frac{\sup_{\vec{\xi} \in \Delta^e} |\nabla F|}{\sup_{\vec{\xi} \in \Delta^e} |\det(\nabla F)|^{1/2}} \quad (19)$$

$$J^e = \frac{\inf_{\vec{\xi} \in \Delta^e} |\det(\nabla F)|}{\sup_{\vec{\xi} \in \Delta^e} |\det(\nabla F)|} \quad (20)$$

In the equations above, F is the mapping from parametric space to physical space, $\vec{\xi}$ are the quadrature points on the parent triangle, Δ^e is the domain of the e^{th} triangle in the mesh, and Δ_M is the domain of the triangulation. These four equations give convenient quality metrics for both individual elements in the mesh, and the mesh as a whole. The following section will first discuss the nature of the element-wise quality metrics, J^e and S^e , and we will then provide global mesh quality metrics for select meshes presented in this paper.

S^e is a metric of the skewness of the element, with possible values lying in the domain $S^e \in [1, \infty)$. Intuitively speaking, a value of $S^e = 2$ for a linear triangle would indicate that the triangle is roughly twice as long as it is wide. For curvilinear triangles this metric is similar, but is a local quantity, and can change depending on where it is evaluated on the triangular domain. We therefore choose the “worst case” as the value for S^e over a triangle. Figure 17c shows a triangle that is highly skewed ($S^e = 1.73$).

J^e is a metric of the variation in the Jacobian determinant of the mapping from physical space to parametric space, and has domain $J^e \in [0, 1)$. For a linear triangle, J^e is always unity, as the determinant is constant across the triangle. However, for a higher order triangle, the mapping can vary across the triangle. Figure 17b shows an example of this. Note how the control points at the top of the triangle are much more closely spaced than those at the bottom, leading to a poorly conditioned triangle ($J^e = 0.14$). An ideal isotropic element would maximize J^e and minimize S^e , that is $J^e = S^e = 1$. A triangle of good quality in both of these metrics will have both an evenly distributed control net in physical space, as well as a consistent mapping from physical space to parametric space, such as the triangle in Figure 17a.

Figure 17: Examples of triangles of varying quality. (a) A well-conditioned, isotropic triangle. (b) A triangle with an ill-conditioned Jacobian. (c) A highly skewed triangle.

Table 3: Mesh quality metrics for different geometries.

	J	Mean J^e	S	Mean S^e
Airfoil [Fig. 26]	0.8408	0.9957	1.7197	1.1071
Quarter Annulus [Fig. 24]	0.7320	0.8483	1.5331	1.1604
Plate with Hole (Lifted Surface) [Fig. 15a]	0.7115	0.9730	1.3621	1.0327
Plate with Hole (Dynamic Quadtree)[Fig. 15b]	0.8925	0.9767	1.5886	1.1117
Nozzle [Fig. 14]	0.5381	0.8481	1.8926	1.1163

Table 3 presents global mesh quality metrics, J and S , for some of the geometries shown in this paper, as well as the averages of the element-wise metrics. Unless otherwise noted, every mesh in the table was generated using our dynamic quadtree method. The values present in Table 3 demonstrate that our meshing procedure succeeds in producing high quality meshes for a wide variety of two-dimensional geometries. In all cases, the average J^e and S^e values are very close to their ideal value of 1. The global mesh quality values are similarly well behaved. Take for example the mesh of the quarter annulus ($J = 0.73, S = 1.53$). These values are still relatively close to the ideal value of 1, and as shown in Section 6, these “worst-case” elements are still well conditioned enough to allow for optimal convergence rates.

To further illuminate the meaning of the metric J^e , we considered two numerical experiments. First, we consider the control net for a unit triangle with all weights equal to unity. We then calculate the value for J^e as we vary the weight of the center control point, w_{10} , from zero to ten. Second, we again consider the unit triangle, and evaluate the value of J^e as we move the center control point, $\mathbf{P}_{10}$, ever closer to the origin ($d_{10} \rightarrow 0$), while keeping all other control points fixed. These two experiments have the effect of studying (1) the effect of control point weight distribution on J^e and (2) the effect of spatial control point distribution on J^e . Figure 18 shows the results of these experiments. Note that as the value of w_{10} moves away from unity, J^e quickly approaches zero and the element becomes ill-conditioned. Likewise, as d_{10} moves closer to zero, J^e similarly approaches zero.

As mentioned above, both the control points and weights play an important role in the performance of numerical methods employing rational Bernstein-Bézier triangles. In fact, it can be shown that if the weights are not properly chosen, suboptimal convergence rates may be observed. We have developed rich mathematical theory regarding the effects of control points and weights on analysis, but such theory is beyond the scope of the current paper. However, a key finding of our work is that a critical and necessary ingredient for optimal convergence rates is that our meshes are asymptotically affine, that is: $J \rightarrow 1$ as $h \rightarrow 0$. We will demonstrate later that our mesh refinement strategy satisfies this property. In order to arrive at an asymptotically affine series of meshes, our meshes must satisfy the following property: for each rational Bernstein-Bézier triangle of radius r in $\mathbb{R}^2$, the control points in projective space, $\mathbb{R}^3$, lie within a circumsphere of radius r . This indicates that the control weights for each element should be as uniform as possible, a characteristic enforced by our smoothing procedure.

Figure 18: (a) The effect of control point weight distribution on J^e . (b) The effect of spatial control point distribution on J^e .

5 Mesh Refinement

We can perform mesh refinement by subdividing triangles into any number of subtriangles. It remains, then, to find the control points and weights associated with the subtriangles. We first consider the polynomial setting (where the weights are unity) before turning to the rational case. Let $\Delta \subset \mathbb{R}^2$ be the parent triangle, and let $\Delta^k \subset \Delta$ denote the k^{th} subtriangle.

Figure 19: Mapping a parent triangle (Δ) and subdivision triangle (Δ^k) from parent to physical space. The collocation points within Δ^k are shown in red.

Let $\{\vec{x}_i\}_{i=1}^n$ denote the control points associated with Δ . We wish to find $\{\vec{x}_j^k\}_{j=1}^n$, the control points associated with Δ^k , that yield:

$$\sum_{j=1}^n \vec{x}_j^k B_j^k(\vec{\xi}) = \sum_{i=1}^n \vec{x}_i B_i(\vec{\xi}) \quad \forall \vec{\xi} \in \Delta^k \quad (21)$$

where $\{B_i\}_{i=1}^n$ and $\{B_j^k\}_{j=1}^n$ are the Bernstein polynomials associated with Δ and Δ^k , respectively. To obtain these control points, we enforce Equation (21) at select collocation points $\{\vec{\xi}_l\}_{l=1}^n$ within Δ^k , viz:

$$\sum_{j=1}^n \vec{x}_j^k B_j^k(\vec{\xi}_l) = \sum_{i=1}^n \vec{x}_i B_i(\vec{\xi}_l) \quad l = 1, \dots, n \quad (22)$$

We have found that an equispaced selection of collocation points yields suitable results, but the choice of collocation points is by no means unique. For the rational case, we repeat the above steps for the projective control points $\{\vec{x}_i^w\}_{i=1}^n$ and $\{\vec{x}_j^{k,w}\}_{j=1}^n$. This method will preserve both the geometric mapping and the weighting function, which is necessary for obtaining optimal convergence rates [5].

As an example of the refinement process described above, we illustrate the case of uniform subdivision of a triangular element. Figures 20a,b show the parent and physical triangles, respectively, and Figures 20c,d show the respective triangles under uniform subdivision. Figure 21 shows an exploded view of the subdivided triangles. It is worth reiterating that this refinement method is not limited to uniform subdivision. The formulation above allows for refinement to be done by arbitrary subdivision of the parent triangle in parametric space, which is then mapped to the physical triangle. This is an incredibly powerful tool in the context of mesh refinement. For example, we can perform local mesh refinement by bisecting triangles to create transition elements.

Finally, we make one last observation about mesh quality under successive levels of refinement. Recall from the previous section that we introduced the metric J^e as a measure of how linear a triangle is. In the context of mesh refinement, we note that optimal convergence rates under refinement are obtained if the mesh is asymptotically affine. In the next section, we use the method of manufactured solutions to demonstrate convergence under uniform subdivision. Table 4 shows the values of J under uniform subdivision for the meshes considered in Sections 6.1 and 6.2. The table shows that the values of J trend towards unity under refinement, which is to say the mesh becomes more and more linear, and overall mesh quality improves.

Table 4: Mesh quality metrics for different geometries.

Refinement Level	Plate and Hole	Quarter Annulus
0	0.7320	0.6249
1	0.7417	0.7417
2	0.7885	0.7885

Figure 20: Uniform subdivision. (a) The parent triangle in parametric space. (b) The parent triangle in physical space. (c) The subdivided triangles in parametric space. (b) The subdivided triangles in physical space.

Figure 21: An exploded view of the subdivision triangles from Figure 20.

6 Numerical Examples

To test that our method does indeed produce analysis suitable meshes, we solve a simple 2D steady heat conduction problem, and use the method of manufactured solutions to verify that our solutions converge as expected under uniform subdivision of the mesh. We consider two canonical examples: a plate with a hole and a quarter annulus. We also present results for potential flow over a three piece airfoil to demonstrate simulation capabilities over more complex geometries.

6.1 Plate with a Hole

We first consider the canonical example of a plate with a circular hole. We define a manufactured solution over the geometry of the form:

$$\phi(x, y) = -(1 - \sqrt{x^2 + y^2})(x + 4)(x - 4)(y + 4)(y - 4) \quad (23)$$

The problem was set to have homogeneous Dirichlet boundary conditions on every global boundary, and the internal heat generation over the domain was calculated as $f(x, y) = -\nabla^2 \phi$. Figure 22 shows the problem mesh under several levels of refinement. Figure 23 shows that the problem converges as expected in both the L_2 and H_1 norms.

Figure 22: Uniformly refined meshes of the plate and hole.

6.2 Quarter Annulus

We next consider a quarter annulus, with the manufactured solution:

$$\phi(r) = \frac{u_o \ln(r/r_i) - u_i \ln(r/r_o)}{\ln(r_o/r_i)} \quad (24)$$

The problem was set to have a constant flux boundary condition on the inner radius, no-flux boundary conditions at then ends of the annulus and homogeneous Dirichlet boundary conditions on the outer radius. Figure 24 shows the problem mesh under several levels of refinement and Figure 25 shows that the problem converges as expected in both the L_2 and H_1 norms.

Figure 23: Method manufactured solutions convergence plots for the plate and hole. Rational Bernstein polynomials of degree $p = 3$ were used for analysis.

Figure 24: Uniformly refined meshes of the quarter annulus.

Figure 25: Method manufactured solutions convergence plots for the quarter annulus. Rational Bernstein polynomials of degree $p = 3$ were used for analysis.

6.3 Potential Flow over an Airfoil

As a final demonstration of the suitability of these meshes for analysis, we consider the problem of potential flow over a slotted airfoil, and solve for the velocity potential field using the same solver as in the previous examples. The problem was set to have constant velocity inflow on the left of the domain, no flow boundary conditions on the top and bottom of the domain as well as on the airfoil and slot surface, and zero potential at the outflow. No circulation conditions about the airfoil and slots were also enforced. Figure 26 shows the mesh used for the problem and Figure 27 shows the solution visualized as streamlines.

Figure 26: Mesh around the airfoil.

Figure 27: Streamlines around the airfoil.

7 Conclusions

We have presented the problem of surface-to-volume parameterization in the context of isogeometric analysis. In part one of this two part series, we have presented a method for blending isogeometric analysis with traditional meshing techniques to create geometrically exact 2-dimensional meshes. We have shown that these meshes derive all their characteristics directly from their bounding NURBS curves, and that they are indeed analysis suitable. The one departure of our approach from traditional isogeometric analysis is that our basis is not higher-order smooth across element boundaries. By using rational Bernstein-Bézier triangles as our finite elements, we have limited our basis to be C^0 -continuous across element boundaries. However, we consider this to be an acceptable trade-off for the flexibility and robustness offered by triangular mesh generation technology. In the second paper in this series, we will extend the work presented here to solving the surface-to-volume parameterization problem, using three-dimensional meshing techniques and rational Bernstein-Bézier tetrahedra.

8 Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1144083, as well as work supported by the Air Force Office of Scientific Research under Grant No. FA9550-14-1-0113. The authors would additionally like to thank the anonymous reviewers for their comments, which helped to improve the quality of this paper.

A Appendix: Flowcharts and Algorithms

A Lifted Surface Flowchart

Figure 28: Flowchart for the lifted surface method. The flowchart is segmented to align with the textual description provided in Section 4.1.

B Dynamic Quadtree Flowchart

Figure 29: Flowchart for the dynamic quadtree method. The flowchart is segmented to align with the textual description provided in Section 4.2.

C Algorithms

Algorithms for triangular control net creation and degree elevation. These algorithms are referenced by name in Sections 4.1 and 4.2 as well as Appendices A and B.

Algorithm 1 Create_Triangle

Input: $\mathbf{P}^b$, the control points of a Bézier curve.
Output: T, the control net of the Bézier triangle.
// For the current Bezier Curve, find the first and last control points.
x1 = $\mathbf{P}^b(1+p*hh,:)$
x2 = $\mathbf{P}^b(1+p*(hh-1),:)$
// Based off of the first and last control points, find the lifted point.
dx = x2-x1
x3 = (x2+x1)/2 + [dx(2) -dx(1)]
// x1, x2 and x3 are the vertices of the Bezier triangle
T(1,:) = x1
T(2,:) = x2
T(3,:) = x3
// The nodes on the curved side are taken from the Bezier Curve.
T(4,:) = B(4,:)
T(5,:) = B(3,:)
// Find the other side nodes by linear interpolation
T(6,:) = B(2,:)+ (B(3,:)-B(2,:))/3
T(7,:) = B(2,:)+ 2*(B(3,:)-B(2,:))/3
T(8,:) = B(3,:)+ (B(1,:)-B(3,:))/3
T(9,:) = B(3,:)+ 2*(B(1,:)-B(3,:))/3
// Find the center node as the location that minimizes the distance to all the other nodes.
A = [ones(9,1) zeros(9,1);zeros(9,1) ones(9,1)];
b = [T(1:9,1); T(1:9,2)];
T(10,:) = pinv(A)*b;

Algorithm 2 Elevate_Triangle

Input: vert, the vertices of the linear triangle.

Output: T, the control net of the Bézier triangle.

T(1,:) = vert(1,:)

T(2,:) = vert(2,:)

T(3,:) = vert(3,:)

// Find the side nodes by linear interpolation

T(4,:) = T(1,:) + (T(2,:) - T(1:))/3

T(5,:) = T(1,:) + 2*(T(2,:) - T(1:))/3

T(6,:) = T(2,:) + (T(3,:) - T(2:))/3

T(7,:) = T(2,:) + 2*(T(3,:) - T(2:))/3

T(8,:) = T(3,:) + (T(1,:) - T(3:))/3

T(9,:) = T(3,:) + 2*(T(1,:) - T(3:))/3

// Find the center node as the location thats between node 6 and node 9.

T(10,:) = T(6,:) + (T(9,:) - T(6:))/2

Figure 30: Here we show the warping function $\mathbf{F}$ that takes points $\mathbf{v}$ on a linear physical element $\tilde{\Omega}_e$ to points $\mathbf{F}(\mathbf{v})$ on a curvilinear physical element Ω_e .

D Impact of Mesh Quality on Approximation

In this appendix, we examine the impact of mesh quality on approximation through the use of functional analysis. More specifically, we present an *a priori* approximation estimate for the spaces utilized in an isogeometric triangular Bernstein-Bézier discretization in which all dependencies on shape and the weighting function are precisely specified. This estimate extends a previously obtained result for the blended isogeometric discontinuous Galerkin (BIDG) method [28], and it extends previously obtained results in the isogeometric community to triangles [5]. Moreover, our estimate may be seen as a practical nonlinear extension of the Bramble-Hilbert lemma for linear triangular elements as it clearly delineates the effects of linear mesh regularity and the nonlinear mapping.

In this direction, let us introduce the notion of a warping function $\mathbf{F} : \tilde{\Omega}_e \rightarrow \Omega_e$ that characterizes the map from a physical linear triangle $\tilde{\Omega}_e$ defined by the three vertex control points $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ to the physical coordinate triangle Ω_e . Such a warping function is graphically depicted in Figure 30. With the definition of a warping function established, the approximation space of degree p over Ω_e associated with an isogeometric triangular Bernstein-Bézier discretization is defined as follows:

$$\mathcal{S}_p^h(\Omega_e) := \left\{ u^h \in L^2(\Omega_e) : w(u^h \circ \mathbf{F}) \in \mathcal{P}^p(\tilde{\Omega}_e) \right\}$$

where $\mathcal{P}^p(\tilde{\Omega}_e)$ is the space of polynomials of degree p defined over the triangle $\tilde{\Omega}_e$ and $w \in \mathcal{P}^p(\tilde{\Omega}_e)$ is the weighting function defined via the relation:

$$w(\mathbf{v}) = \sum_{r+s+t=p} B_{rst}(\mathbf{v}) w_{rst}$$

wherein $\{B_{rst}\}_{r+s+t=p}$ are the Bernstein polynomials of degree p over the triangle $\tilde{\Omega}_e$ and $\{w_{rst}\}_{r+s+t=p}$ are the Bézier weights. By construction, the Bernstein polynomials are point-wise non-negative and form a partition of unity and the Bézier weights are positive and bounded above by one. Consequently, the weighting function is bounded below by a positive number and above by one. In what follows, we shall also refer to the mesh size h which is defined as the diameter of the physical linear triangle $\tilde{\Omega}_e$.

With the aforementioned terminology established, we have the following result.

Theorem A.1. *Over the physical element Ω_e , there exists a constant C_{shape} 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 $u^h \in \mathcal{S}_p^h(\Omega_e)$ satisfying the estimate:*

$$\|u - u_h\|_{L^2(\Omega_e)} \leq C_{shape} h^{p+1} C_{det}(\nabla \mathbf{F}) C_{var}(\nabla \mathbf{F}, w, u)$$

where:

$$C_{det}(\nabla \mathbf{F}) = \|\det \nabla \mathbf{F}^{1/2}\|_{L^\infty(\tilde{\Omega}_e)} \|\det \nabla \mathbf{F}^{-1/2}\|_{L^\infty(\Omega_e)}$$

and:

$$C_{var}(\nabla \mathbf{F}, w, u) = \sum_{k=0}^{p+1} \sum_{j=0}^k \|\nabla^{p+1-k} w\|_{L^\infty(\tilde{\Omega}_e)} \alpha_{j,k}(\nabla \mathbf{F}) |u|_{H^j(\Omega_e)}$$

wherein:

$$\alpha_{j,k}(\nabla \mathbf{F}) = \sum_{\substack{i_1+i_2+\dots+i_k=j \\ i_1+2i_2+\dots+ki_k=k}} \|\nabla \mathbf{F}\|_{L^\infty(\tilde{\Omega}_e)}^{i_1} \|\nabla^2 \mathbf{F}\|_{L^\infty(\tilde{\Omega}_e)}^{i_2} \cdots \|\nabla^k \mathbf{F}\|_{L^\infty(\tilde{\Omega}_e)}^{i_k}$$

Proof. By definition we have,

$$\int_{\Omega_e} u^2 d\mathbf{x} = \int_{\tilde{\Omega}_e} (u \circ \mathbf{F})^2 \det \nabla \mathbf{F} d\mathbf{v}.$$

Therefore, $\|u\|_{L^2(\Omega_e)} = \|\det \nabla \mathbf{F}^{1/2} u \circ \mathbf{F}\|_{L^2(\tilde{\Omega}_e)}$, and as a consequence:

$$\|u\|_{L^2(\Omega_e)} \leq \|\det \nabla \mathbf{F}^{1/2}\|_{L^\infty(\tilde{\Omega}_e)} \|u \circ \mathbf{F}\|_{L^2(\tilde{\Omega}_e)}$$

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

$$\|u - u_h\|_{L^2(\Omega_e)} \leq \|\det \nabla \mathbf{F}^{1/2}\|_{L^\infty(\tilde{\Omega}_e)} \|u \circ \mathbf{F} - u_h \circ \mathbf{F}\|_{L^2(\tilde{\Omega}_e)}$$

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

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

Note that, by construction, $w(u^h \circ \mathbf{F})$ is an arbitrary polynomial of degree p . Therefore, by the classical Bramble-Hilbert lemma, we may select u^h such that the following inequality holds:

$$\|w(u \circ \mathbf{F}) - w(u_h \circ \mathbf{F})\|_{L^2(\tilde{\Omega}_e)} \leq C_1 h^{p+1} |w(u \circ \mathbf{F})|_{H^{p+1}(\tilde{\Omega}_e)}$$

where C_1 is a constant that only depends on the polynomial degree p and the shape regularity of $\tilde{\Omega}_e$ (i.e., the constant depends on the shape but not the size of $\tilde{\Omega}_e$). We must now bound the seminorm $|w(u \circ \mathbf{F})|_{H^{p+1}(\tilde{\Omega}_e)}$ appearing in the above estimate by an analogous norm over the physical coordinate triangle Ω_e . To do so, we first recognize that:

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

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

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

To obtain control of higher-order seminorms, we simply recurse on the previous estimate, as is done in [5], resulting in the estimate:

$$|u \circ \mathbf{F}|_{H^k(\tilde{\Omega}_e)} \leq C_2 \|\det \nabla \mathbf{F}^{-1/2}\|_{L^\infty(\Omega_e)} \sum_{j=0}^k \alpha_{j,k}(\nabla \mathbf{F}) |u|_{H^j(\Omega_e)}$$

where C_2 again is a constant that only depends on the polynomial degree p and the shape regularity of $\tilde{\Omega}_e$. We thus arrive with the desired bound where $C_{shape} = C_1 C_2$. $\square$

The above results indicates that if the following three conditions hold, then an isogeometric triangular Bernstein-Bézier discretization exhibits a similar convergence rate to that seen for affine finite elements:

- The determinant Jacobian (i.e., $\det \nabla \mathbf{F}$) is uniformly bounded from above and below.
- The condition $\|\nabla^k \mathbf{F}\|_{L^\infty(\tilde{\Omega}_e)} = O(1)$ holds for $k = 0, \dots, p+1$.
- The condition $\|\nabla^k w\|_{L^\infty(\tilde{\Omega}_e)} = O(1)$ holds for $k = 0, \dots, p+1$.

It is easily shown that if nested refinements are performed, then $\nabla \mathbf{F} \rightarrow \mathbb{I}$ and $w \rightarrow \text{constant}$ element-wise in the limit of mesh refinement, and hence one may expect to see optimal convergence provided that the corresponding linear triangular mesh is shape regular. Indeed, this is verified numerically within the main text of the current work. That being said, it is instructive to examine the effect of mesh warping in the pre-asymptotic limit as well as in the setting of non-nested refinements. Such an analysis is beyond the scope of the current work, but it is shown in [28] that $\|\nabla \mathbf{F}\|_{L^\infty(\tilde{\Omega}_e)} = O(1)$ if projective control points $\mathcal{P}_{jkl} = \{w_{jkl} \mathbf{P}_{jkl}, w_{jkl}\}^T$ for a particular Bernstein-Bézier triangle lie in a convex polyhedron of radius $O(h)$. Provided that the Jacobian is also bounded from above and below, this condition will guarantee that an isogeometric triangular Bernstein-Bézier discretization will exhibit first-order convergence. However, it does not guarantee that such a method will converge with optimal rate. This requires more restrictive bounds on the projective control points, and such bounds will be the focus of a forthcoming publication.

References

- [1] MESH2d - Automatic Mesh Generation - File Exchange - MATLAB Central.
- [2] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, and A.-V. Vuong. Swept Volume Parameterization for Isogeometric Analysis. In Edwin R. Hancock, Ralph R. Martin, and Malcolm A. Sabin, editors, *Mathematics of Surfaces XIII*, number 5654 in Lecture Notes in Computer Science, pages 19–44. Springer Berlin Heidelberg, 2009.
- [3] 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(6):1043–1078, June 1987.
- [4] C. L. Bajaj, F. Bernardini, and G. Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3d Scans. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 109–118, New York, NY, USA, 1995. ACM.
- [5] 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(07):1031–1090, July 2006.
- [6] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. In , *31st Annual Symposium on Foundations of Computer Science, 1990. Proceedings*, pages 231–241 vol.1, October 1990.
- [7] 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(1-5):15–47, July 2011.
- [8] L. P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1-4):97–108, June 1989.
- [9] 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(45–46):3210–3222, October 2011.
- [10] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, August 1986.
- [11] P.I. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 90(9):1156–1182, June 2012.
- [12] 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(11):1309–1331, September 2009.

- [13] R. N. Goldman. Subdivision algorithms for Bézier triangles. *Computer-Aided Design*, 15(3):159–166, May 1983.
- [14] R. N. Goldman and D. J. Filip. Conversion from Bézier rectangles to Bézier triangles. *Computer-Aided Design*, 19(1):25–27, January 1987.
- [15] 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(39–41):4135–4195, October 2005.
- [16] N. Jaxon and X. Qian. Isogeometric analysis on triangulations. *Computer-Aided Design*, 46:45–57, January 2014.
- [17] A. Johnen, J. F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 233:359–372, January 2013.
- [18] X. Li, M. S. Shephard, and M. W. Beall. Accounting for curved domains in mesh adaptation. *International Journal for Numerical Methods in Engineering*, 58(2):247–276, September 2003.
- [19] 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(1):63–76, January 2012.
- [20] 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(5–8):357–373, January 2010.
- [21] D. Liu and J. Hoschek. G1 continuity conditions between adjacent rectangular and triangular Bézier surface patches. *Computer-Aided Design*, 21(4):194–200, May 1989.
- [22] L. Liu, Y. Zhang, Thomas J. R. Hughes, M. A. Scott, and T. W. Sederberg. Volumetric T-spline construction using Boolean operations. *Engineering with Computers*, 30(4):425–439, November 2013.
- [23] L. Liu, Y. Zhang, Y. Liu, and W. Wang. Feature-preserving T-mesh construction using skeleton-based polycubes. *Computer-Aided Design*, 58:162–172, January 2015.
- [24] C. Loop. A G1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11(3):303–330, June 1994.
- [25] C. Loop. Smooth Spline Surfaces over Irregular Meshes. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’94, pages 303–310, New York, NY, USA, 1994. ACM.
- [26] 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(3):273–285, July 2004.

- [27] 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, pages 269–280, New York, NY, USA, 2008. ACM.
- [28] C. Michoski, J. Chan, L. Engvall, and J.A. Evans. Foundations of the Blended Isogeometric Discontinuous Galerkin (BIDG) Method. *In review*, 2015.
- [29] J.-F. Remacle, J. Lambrechts, C. Geuzaine, and T. Toulorge. Optimizing the Geometrical Accuracy of 2d Curvilinear Meshes. *Procedia Engineering*, 82:228–239, 2014.
- [30] Hanan Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.*, 16(2):187–260, June 1984.
- [31] J.R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry Towards Geometric Engineering*, number 1148 in Lecture Notes in Computer Science, pages 203–222. Springer Berlin Heidelberg, 1996.
- [32] H. Speleers and C. Manni. Optimizing domain parameterization in isogeometric analysis based on Powell–Sabin splines. *Journal of Computational and Applied Mathematics*, 289:68–86, December 2015.
- [33] 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*, 221–222:132–148, May 2012.
- [34] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, December 2013.
- [35] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell. Curved PN Triangles. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pages 159–166, New York, NY, USA, 2001. ACM.
- [36] 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(2):351–360, February 2013.
- [37] 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*, 297:292–324, December 2015.
- [38] M. Yerry and M. Shephard. A Modified Quadtree Approach To Finite Element Mesh Generation. *IEEE Computer Graphics and Applications*, 3(1):39–46, January 1983.
- [39] S. Zeng and E. Cohen. Hybrid volume completion with higher-order Bézier elements. *Computer Aided Geometric Design*, 35–36:180–191, May 2015.