2009-08-12

Isogeometric Finite Element Analysis Using T-Splines

Jingang Li
Brigham Young University - Provo

ISOGEOMETRIC FINITE ELEMENT ANALYSIS USING T-SPLINES

by

Jingang Li

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Civil and Environmental Engineering

Brigham Young University

December 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Jingang Li

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Richard J. Balling, Chair
Date	Steven E. Benzley
Date	David W. Jensen

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Jingang Li in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Richard J. Balling
Chair, Graduate Committee

Accepted for the Department

E. James Nelson
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

ISOGEOMETRIC FINITE ELEMENT ANALYSIS USING T-SPLINES

Jingang Li

Department of Civil and Environmental Engineering

Master of Science

Non-uniform rational B-splines (NURBS) methodology is presented, on which the isogeometric analysis is based. T-splines are also introduced as a surface design methodology, which are a generalization of NURBS and permit local refinement. Isogeometric analysis using NURBS and T-splines are applied separately to a structural mechanics problem. The results are compared with the closed-form solution. The desirable performance of isogeometric analysis using T-splines on engineering analysis is demonstrated.

ACKNOWLEDGMENTS

I would like to gratefully thank my advisor Dr. Richard J. Balling for his counsel and guidance, along with my other committee members, Dr. Steven E. Benzley and Dr. David W. Jensen for their input and suggestions in my research. I would also like to thank Dr. Thomas W. Sederberg for his insight on T-splines. Finally, I would like to thank my beautiful wife for her incredible patience, understanding, encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Finite element analysis (FEA) uses shape functions and nodes, while Computer-aided design (CAD) employs basis functions and control points. The typical situation in engineering practice is that designs are created in CAD systems, meshes are generated from CAD data, and FEA is executed as shown in Figure 1-1.

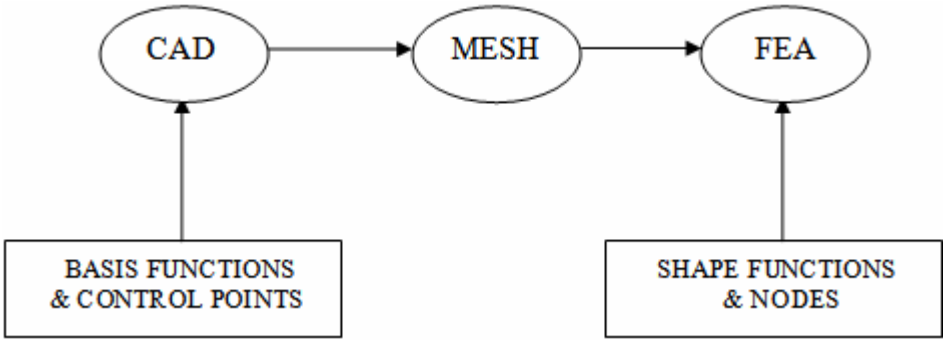


Figure 1-1: Relationship between CAD and FEA in traditional analysis.

Tremendous effort has been made to integrate CAD and FEA. Recently, Dr. Thomas J.R. Hughes introduced the concept of isogeometric analysis to make viable a seamless interaction between CAD and FEA [1]. The name of isogeometric analysis signifies that the same basis functions can be used in both CAD and FEA. Figure 1-2 shows relationship between CAD and FEA in isogeometric analysis.

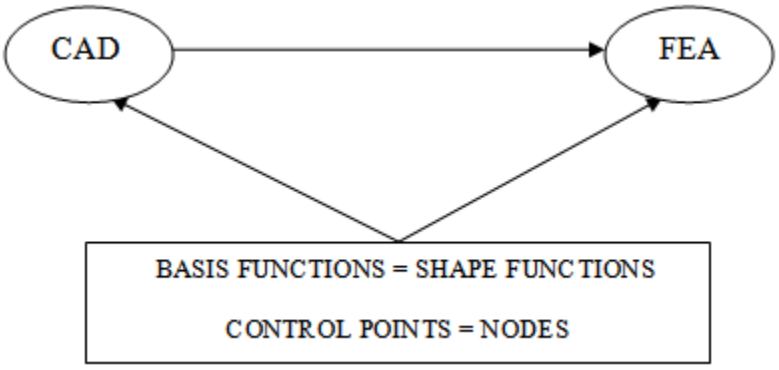


Figure 1-2: Relationship between CAD and FEA in isogeometric analysis.

Isogeometric analysis possesses many desirable features. In many cases, mesh generation is unnecessary. It is deemed costly and time consuming. In addition, accuracy problems can be avoided, since geometries may be exactly represented using isogeometric analysis.

The most extensively used computational geometry technology in isogeometric analysis is Non-uniform rational B-splines (NURBS). Although NURBS are ubiquitous in the CAD industry, NURBS possesses some deficiencies. First, NURBS do not allow for local refinement. In order to refine a local area, a global refinement is required because the B-spline control grid traverses the entire domain. As a result of global refinement, many superfluous control points are created, which is inefficient. In Figure 1-3.a, the red NURBS control points are superfluous [12]. Second, gaps and overlaps are inevitable at intersections of NURBS-based surfaces. Figure 1-4 illustrates a hand model comprised of seven B-spline surfaces with gap at the intersection of the hand and arm [11].

T-splines were recently proposed by Dr. Thomas Sederberg as a generalization of NURBS technology that is capable of substantially reducing the number of superfluous control points [11]. In terms of applications in CAD industry, T-splines preserve all of the desirable properties of NURBS. In addition, T-splines permit local refinement by using T-junctions. T-splines allow a row of control points to terminate. The final control point in a partial row is called a T-junction [11]. All the purple points shown in Figure 1-3.b are T-junctions [12]. Control points can be inserted into the control grid without propagating an entire row or column of superfluous control points. T-splines are also capable of closing gaps at intersections of geometric model shapes. Figure 1-5 shows a gap between two B-spline surfaces, which is fixed with T-splines [11].

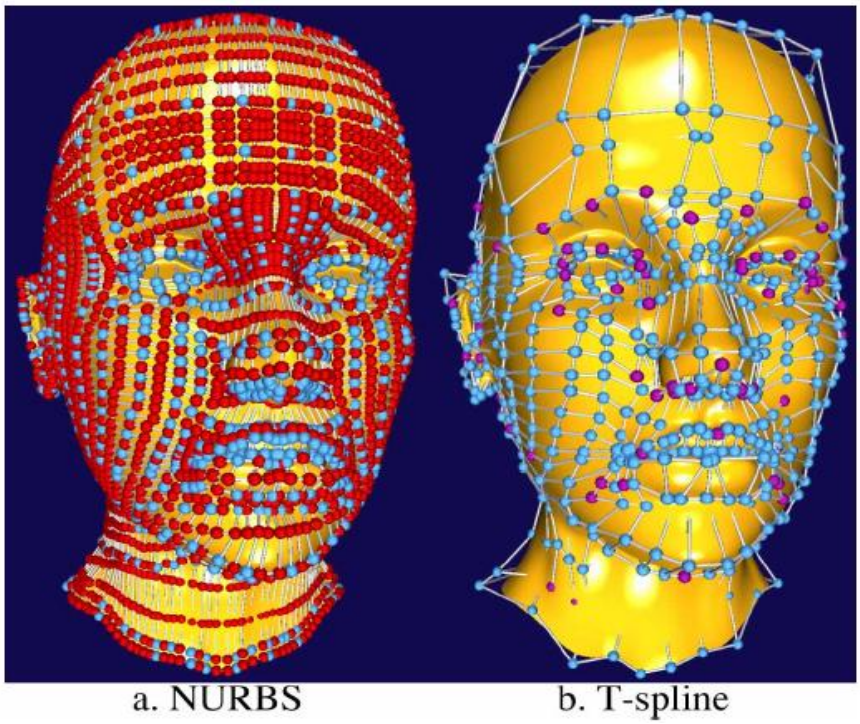


Figure 1-3: Head modeled (a) as a NURBS and (b) as a T-spline [12].

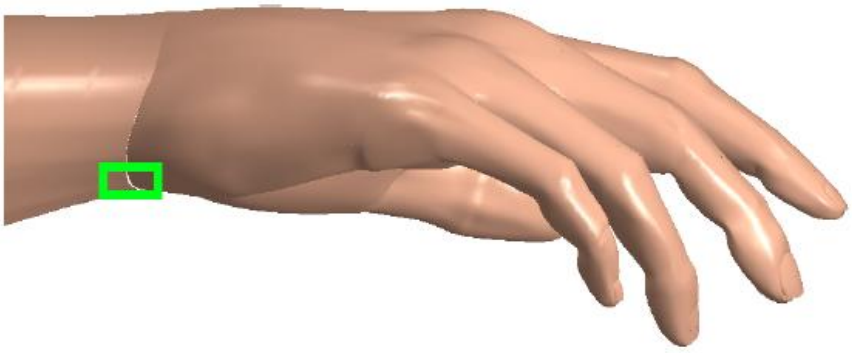


Figure 1-4: Model of a hand comprised of B-spline surfaces [11].

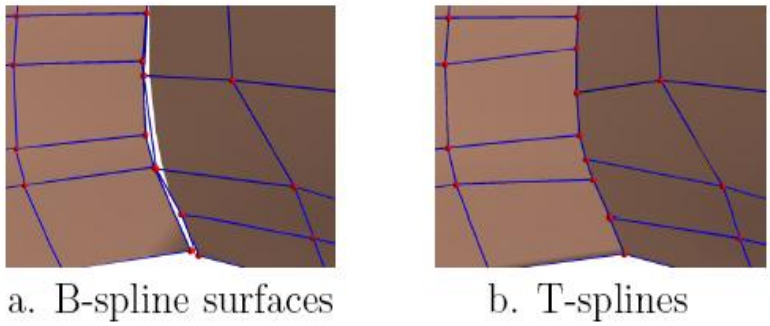


Figure 1-5 A gap at the intersection of B-spline surfaces, fixed with a T-spline [11].

The goal of this thesis is to verify that the same features making T-splines desirable for CAD make it desirable for analysis. In this thesis, isogeometric analysis using T-splines is introduced. The isogeometric analyses of a structural mechanics problem are performed using B-splines and using T-splines, respectively. It is anticipated that isogeometric analyses using both B-splines and T-splines can produce results with the same level of accuracy, while T-splines use significantly fewer control points than B-splines, and is therefore more computationally efficient.

2 Literature Review

2.1 Isogeometric Analysis

Hughes [1] introduced the concept of isogeometric analysis using NURBS to explore the new generation of computational mechanics procedures based on modern developments in computational geometry.

Bazilevs and Hughes [2] explored the mathematical study of isogeometric analysis based on NURBS. They investigated approximation and stability properties in the context of h-refinement. Furthermore, they developed approximation estimates based on a new Bramble-Hilbert lemma in so-called "bent" Sobolev spaces appropriate for NURBS approximations and established inverse estimates similar to ones for finite elements.

Cottrell and Hughes [3] investigated the effects of smoothness of basis functions on solution accuracy within the isogeometric analysis framework using NURBS. They also developed a local refinement strategy that can be utilized in one of the shell analyses.

Hughes, Reali and Sangalli [4] initiated a study of efficient quadrature rules for NURBS-based isogeometric analysis. They developed efficient rules for spaces arising in the calculation of mass, stiffness, and advection matrices.

Auricchio, Beirão de Veiga et al. [5] investigated plane incompressible elastic problems by means of a "stream function" formulation and developed the numerical scheme within the framework of NURBS-based isogeometric analysis. They also proposed a discontinuous Galerking approach to deal with multiple mapped, possibly multiply connected domains.

Cottrell, Reali et al. [6, 7] initiated the study of the isogeometric analysis in the field of structural vibration analysis. They emphasized the concept of k-refinement, a higher-order procedure employing smooth basis functions, which was used repeatedly in the vibration calculations. They applied isogeometric analysis to some simple model problems of structural vibration. The k-method was shown to provide more robust and accurate frequency spectra than typical higher-order finite elements (i.e., the p-method).

Bazilevs, Calo et al. [8] developed a fully-coupled isogeometric monolithic formulation of the fluid-structure interaction of an incompressible fluid on a moving domain with a nonlinear hyperelastic solid.

Bazilevs, Calo et al. [9] developed a NURBS based isogeometric fluid-structure interaction formulation, coupling incompressible fluids with nonlinear elastic solids, and allowing for large structural displacements. They applied this methodology to problems of arterial blood flow modeling.

Zhang, Bazilevs et al. [10] introduced an approach to construct hexahedral solid NURBS meshes and applied the meshes to patient-specific vascular geometric models from imaging data for use in isogeometric analysis.

2.2 T-splines

Sederberg, Zheng et al. [11] proposed a generalization of non-uniform B-spline surfaces called T-splines to allow for local refinements. They also introduced a locally renewable subdivision surface called T-NURCCs (Non-uniform rational Catmull-Clark surfaces with T-junctions).

Sederberg, Cardon et al. [12] introduced a T-spline simplification algorithm to eliminate superfluous control points. They also presented a new T-spline local refinement algorithm

Li, Ray, and Lévy [13] introduced an algorithm for the automatic generation of a control mesh driven by the anisotropy of the shape. Their algorithm made it possible an automatic conversion from a mesh of arbitrary topology to a T-spline surface.

Wang, Zheng, and Seah [14] introduced two local knot insertion based algorithms to resolve the issue of conversion back and forth of a surface between the T-spline and hierarchical B-spline representations.

He, Wang et al. [15] developed the manifold T-splines to extend the currently available algorithms of the popular planar tensor-product NURBS and T-splines to arbitrary manifold domain of any topological type.

Zheng, Wang, and Seah [16] introduced an automatic algorithm to develop smooth parametric surfaces using T-splines from z-map data. The algorithm starts with a rough surface approximation and then progressively refines it in the regions where the approximation accuracy does not meet the requirement.

Bazilevs, Calo et al. [17] explored T-splines as a basis for isogeometric analysis. They applied T-splines to some basic problems of computational fluid and structural mechanics and attained desirable results in all cases.

Yang, Fuchs et al. [18] investigated the evolution of T-spline level sets. They avoided extra branches and singularities of the T-spline level sets without having to use re-initialization steps by incorporating the distance field constraints.

Yang and Jüttler [19] introduced a method for 3D shape metamorphosis based on the evolution of T-spline level sets. They verified that the morphing process of T-spline level sets can be formulated as least squares problems. They also developed a fully automatic algorithm to produce metamorphosis between shapes of any topology.

Deng, Chen and Feng [20] introduced polynomial spline functions over T-meshes. They forced the spline function on every cell to be a tensor-product polynomial, and to achieve the specified smoothness across the common edges. They also demonstrated several advantages of splines over T-meshes overt T-splines.

3 Bézier Curves, B-splines and T-splines

3.1 Bézier Curves

Bézier curves are named after Dr. Pierre Bézier who was an engineer with the Renault car company. He started to create a method known as the Bézier curve formulation in the 1960's, which would be easy and intuitive enough to allow drafters to develop curves without a background in the corresponding mathematics fields. A degree n Bézier curve has a few characteristics as follows:

A Bézier curve has a corresponding control polygon.

A control polygon has n+1 control points numbered from 0 to n.

A control polygon is comprised of straight lines connecting the control points.

A Bézier curve passes through the first and last control points.

A Bézier curve is tangent to the control polygon at the end control points.

Figure 3-1 illustrates two different Bézier curves associated with their corresponding control polygons.

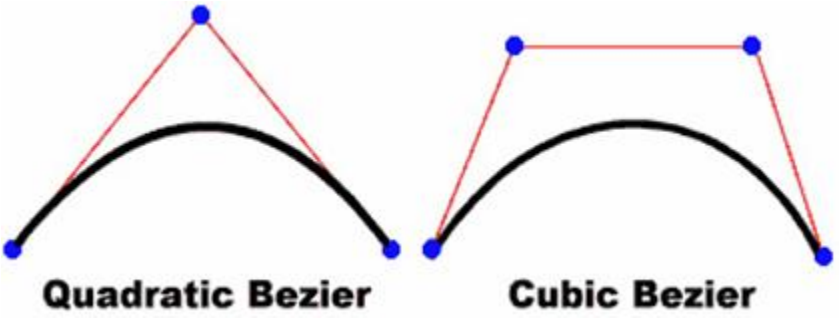


Figure 3-1: Illustration of Bézier curves.

3.1.1 The Equation of a Bézier Curve

The equation of a Bézier curve is given in Equation 3-1.

$$P(t) = \sum_{i=0}^{n} B_i^n(t) P_i \qquad \textbf{(3-1)}$$

Where t is a parameter ranging from 0 to 1, P(t) is either the X or Y coordinate of a point on the curve corresponding to t. P_i is either the X or Y coordinate of a control point i. $B_i^n(t)$ is the basis function for the control point i. For cubic Bézier curves (n=3), the basis functions are given in Equation 3-2.

.

$$B_0^3(t) = (1-t)^3,\ B_1^3(t) = 3t(1-t)^2,\ B_2^3(t) = 3t^2(1-t),\ B_3^3(t) = t^3 \qquad \textbf{(3-2)}$$

The cubic Bézier mass functions are plotted in Figure 3-2 [21].

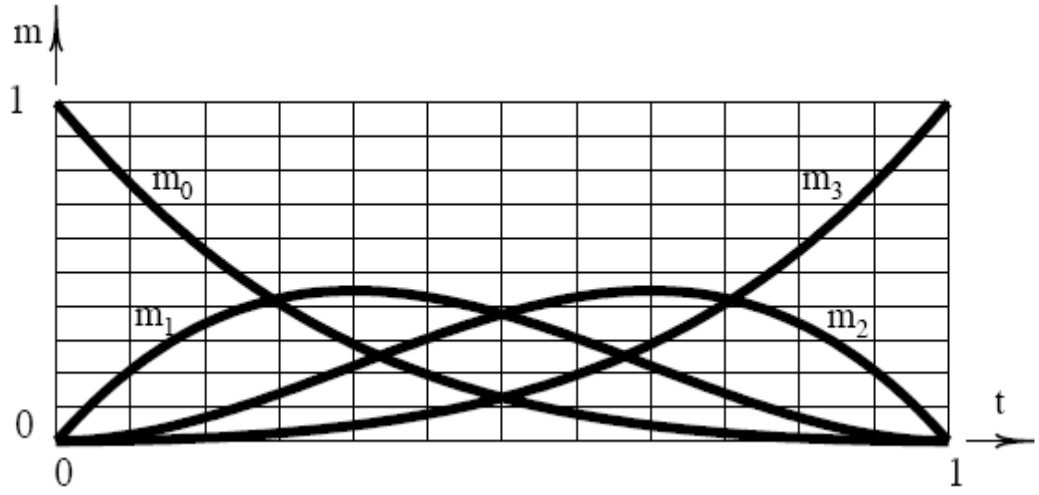


Figure 3-2: Cubic Bézier mass functions [21].

Note that for a given value of t, the basis functions sum to one. This is true for any degree n. Although cubic Bézier curves are extensively used in all kinds of industry, other degree Bézier curve may be used in some cases. Therefore, it's desirable to generalize the basis function for a degree n Bézier curve. Equation 3-3 provides a formula for degree n Bézier curve blending functions denoted $B_i^n(t)$ where the parameter i = 0, 1, …, n.

$$B_i^n(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!}(1-t)^{n-i}t^i \qquad \textbf{(3-3)}$$

3.1.2 Bézier Curves over Arbitrary Parameter Intervals

In Equation 3-3, the parameter t ranges from zero to one. It is desirable to have the parameter t vary over an arbitrary parameter interval (t_0, t_1). The Equation 3-4 provides a new formula for the basis function with the parameter t ranging over an arbitrary parameter interval (t_0, t_1).

$$B_i^n(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^{i} \quad \textbf{(3-4)}$$

3.1.3 Rational Bézier Curves

If each control point of a Bézier curve P_i is assigned a scalar weight, it will become the rational Bézier curve. The equation of a rational Bézier curve is calculated by Equation 3-5.

$$P(t) = \sum_{i=1}^{n+1} R_i^n(t) P_i \quad \textbf{(3-5)}$$

The Equation 3-6 provides a formula for the rational Bézier basis function.

$$R_i^n(t) = \frac{w_i B_i^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)} \quad \textbf{(3-6)}$$

$B_i^n(t)$ represents the Bézier basis function and W_i indicates the weight. When the scale weight of a control point is changed, the shape of the Bézier curve will be changed accordingly. Figure 3-3 illustrates how a set of Bézier curves is developed by changing a control point weight. The reason why this type of Bézier curve is called a rational Bézier curve is because the blending functions are rational polynomials, or the ratio of two polynomial functions of the same variable t. If each control point is assigned the same scalar weight, a rational Bézier curve will be exactly the same as a standard Bézier curve.

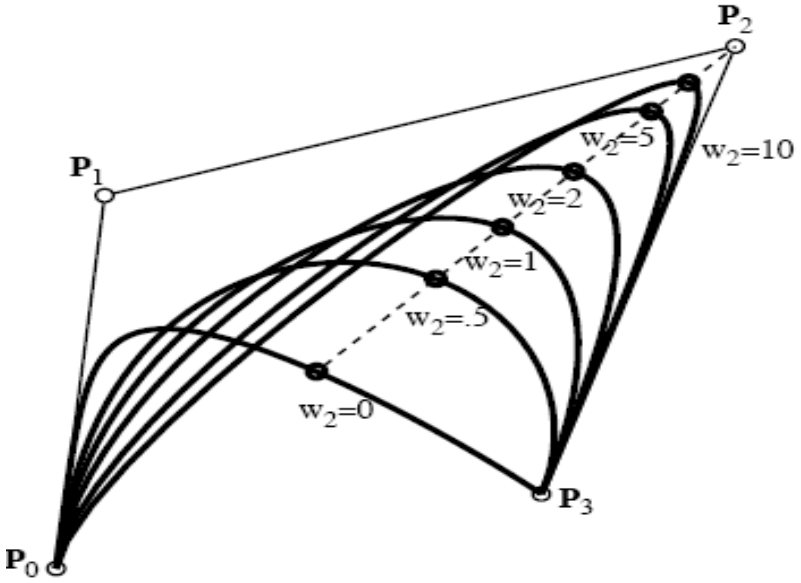


Figure 3-3: Example of rational Bézier curve with one varying scalar weight [21].

3.1.4 Rational Bézier Curve Representation of Circular Arcs

Rational Bézier curves can represent circular arcs exactly. Figure 3-4 illustrates a circular arc represented both by a degree 2 and a degree 3 rational Bézier curve [21].

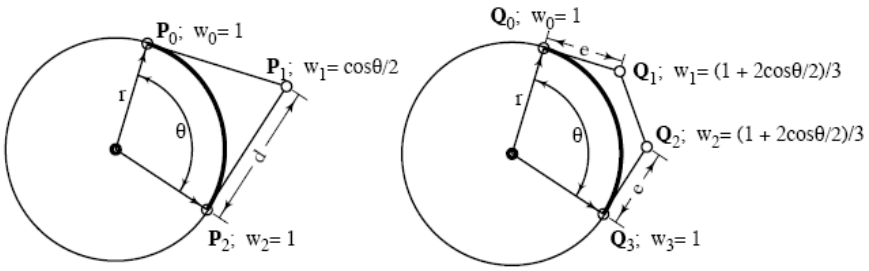


Figure 3-4: Rational Bézier curve representations of circular arcs [21].

The length e shown in Figure 3-4 is calculated in Equation 3-7.

$$e = \frac{2\sin\frac{\theta}{2}}{1+2\cos\frac{\theta}{2}}r \qquad \textbf{(3-7)}$$

3.2 B-spline Curves

The word “spline” stems from the ship building industry, where it is originally referred to a thin strip. Drafts men needed to plot a line through a set of points. The solution was to place metal weights (called ducks) at the control points, and bend a thin strip through the ducks. Figure 3-5 shows a spline and ducks [21]. The B in B-spline stands for basis function. B-spline curves are a series of Bézier curves connected end to end.

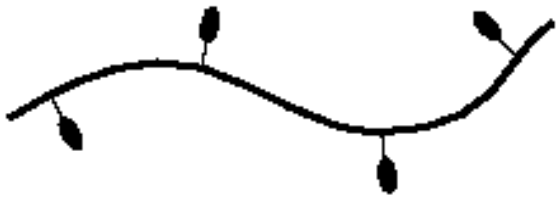


Figure 3-5: Spline and ducks [21].

The physics behind B-spline is that a duck has the greatest control over the shape of the curve at the point of contact, and the influence decreases gradually further along the spline. The draftsmen could insert more metal weights into a certain part of the spline where more control over the shape would be gained.

3.2.1 Polar Form

The traditional approach to define B-spline was mainly focused on basis functions and recurrence relations which required the background in the underlying mathematics. The alternative method of developing a Bézier curve or B-spline curve is denoted Polar Form, which was introduced by Dr. Lyle Ramshaw. The polar form eliminates the necessity of delving into the corresponding intricate mathematical theories to allow people to grasp the concept of a B-spline curve.

The following three rules are essential to develop Bézier and B-spline curves.

1. For a degree n Bézier curve $P_{[a,b]}(t)$, the control points are relabeled $P_i = P(u_1, u_2, \ldots u_n)$ where $u_j = a$ if $j <= n - i$ and otherwise $u_j = b$. Figure 3-6 shows that two degree three Bézier curves marked with polar values are connected together [21]. The parameter interval for the first curve is [0, 2] and the parameter t of the second curve

ranges from 2 to 3. Note that the polar value P(t, t, …, t) corresponds to the point located on a Bézier curve with parameter value t.

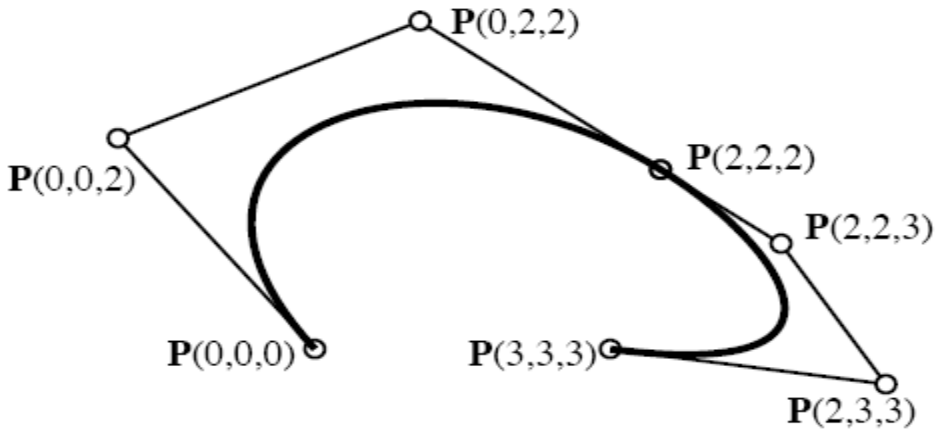


Figure 3-6: Polar form representation of Bézier curves [21].

2. A polar value is symmetric in its arguments. This signifies that the order of arguments has no influence on the polar value. In other words, a polar value can not be changed by changing the order of the arguments.

3. Given (u_1, u_2, …, u_{n-1}, a) and P(u_1, u_2, …, u_{n-1}, b), Equation (3-8 provides a formula to calculate P(u_1, u_2, …, u_{n-1}, c). $P(u_1,u_2,...,u_{u-1},c)$ can be treated as the affine combination of (u_1, u_2, …, u_{n-1}, a) and (u_1, u_2, …, u_{n-1}, b)

$$P(u_1,u_2,...,u_{u-1},c)=\frac{(b-c)P(u_1,u_2,...,u_{u-1},a)+(c-a)P(u_1,u_2,...,u_{u-1},b)}{(b-a)} \qquad \textbf{(3-8)}$$

Equation 3-8 has the geometrical significance which indicates that if one parameter of a polar value varies when the others remain constant, the polar value will form a straight line as illustrated in Figure 3-7.

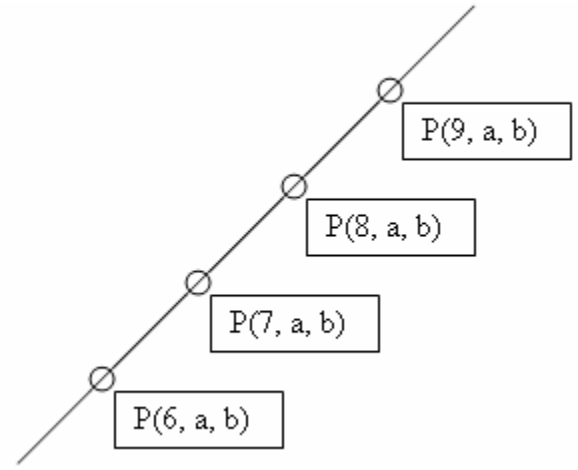


Figure 3-7: Affine map property of polar form.

3.2.2 Knot Vectors

A knot vector is comprised of a list of parameter values denoted knots, which specify the parameter intervals for all the Bézier curves connected together to form a B-spline. Assume that a cubic B-spline is comprised of three Bézier curves whose parameter intervals are [1, 3], [3, 5] and [5, 8]. Thus, the knot vector of this B-spline would be [t_0, t_1, 1, 3, 5, 8, t_6, t_7]. Note that there are two additional knots listed at the beginning and end of the knot vector. These end knots have control over the end conditions of the B-spline curve.

Due to historical reasons, the knot vector of a degree n B-spline seems to need n knots attached to its front and back. In practice, the first and the last knots are usually ignored because they have no impact on the shape of the curve. Therefore, n-1 end-condition knots would be prepended and appended to the knot vector. Given the knot vector for a degree n B-spline $[t_1, t_2, t_3, \ldots]$, the parameters of any polar value are comprised of n adjacent knots extracted from the knot vector and hence the i^{th} polar value can be denoted $P_i(t_i, \ldots, t_{i+n-1})$.

Apparently, a knot vector consists of non-decreasing series of real numbers. If any knot value is placed more than once at the same coordinate in the knot vector, it is referred to as a multiple knot and the corresponding Bézier curve can be thought of as a zero length curve. If the knot vector of a B-spline curve is evenly spaced, the curve is uniform B-spline. Otherwise, it is a non-uniform B-spline. Figure 3-8 shows the polar form representation and the knot vector of a B-spline curve [21].

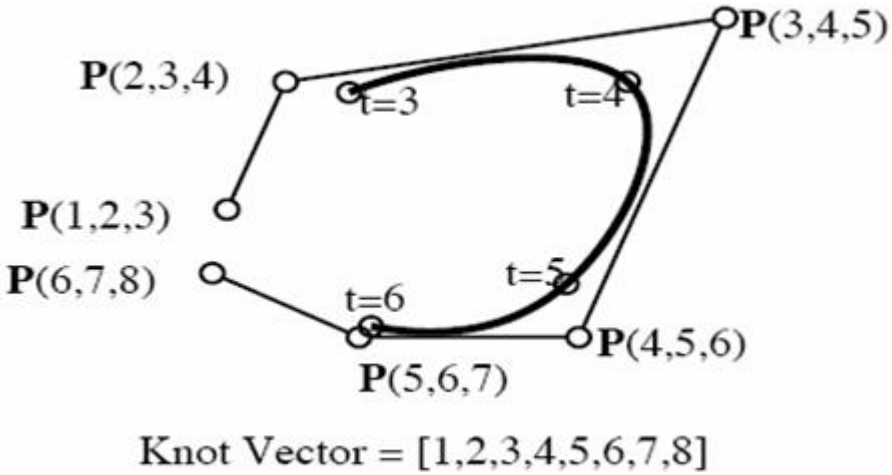


Figure 3-8: Polar form representation and knot vector of B-spline [21].

As indicated earlier, there are n-1 additional knots at the beginning and end of a knot vector. All these extra knots do not represent the parameter intervals of the Bézier curves that constitute the B-spline. However, they play an important role in determining the shape of the B-spline at its ends. For an open B-spline, the conventional practice is to place n identical knots at each end of the knot vector. This makes a B-spline assume a Bézier behavior at its ends. In other words, the B-spline passes through its end control points. Furthermore, the B-spline is tangent to the control polygon at its end control points just as a Bézier curve does. Figure 3-9 illustrates how the Bézier curve end conditions are imposed on the ends of a B-spline [21].

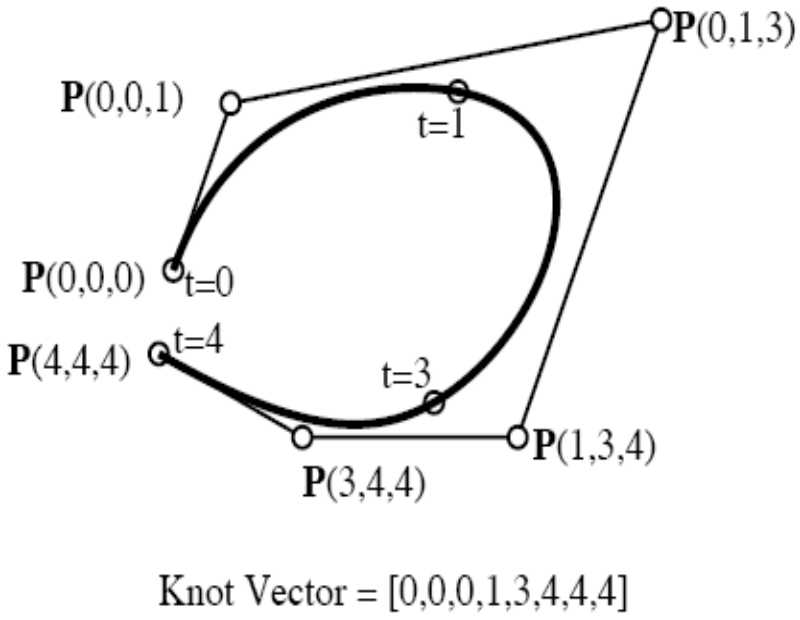


Figure 3-9: Imposition of Bézier curve end conditions on B-spline [21].

3.2.3 Knot Insertion

One of the standard operations of B-spline is knot insertion. The definition of knot insertion is to insert one knot into the existing knot vector without changing the shape of the curve. In addition, some of the existing control points are deleted and replaced with new ones due to the inserted knot. Knot insertion is extensively used to provide more local control by only modifying the part of the curve of interest. The operation of knot insertion can be easily performed with the symmetry and affine properties of polar values.

3.3 T-splines

This subchapter is extracted from reference [11]. T-splines are a generalization of NURBS surfaces that allow for local refinement. Figure 3-10 illustrates an example of a T-mesh in parameters s and t [11]. The s_i represents s coordinates, the t_i represents t coordinates, and the d_i and e_i represent knot intervals.

The control grid of T-splines is called T-mesh, which is simply a rectangular grid permitting T-junctions. Each edge in a T-mesh is a line segment of constant s or of constant t. A T-junction is a vertex shared by one s-edge and two t-edges, or by one t-edge and two s-edges. Each edge of a T-mesh has a knot interval, which is governed by the following rules.

Rule 1: The sum of knot intervals on one edge must equal the sum of knot intervals on the opposing edge.

Rule 2: If T-junctions on opposing edges of a face can be connected with rule 1 being held valid, that edge mush be deemed as part of the T-mesh.

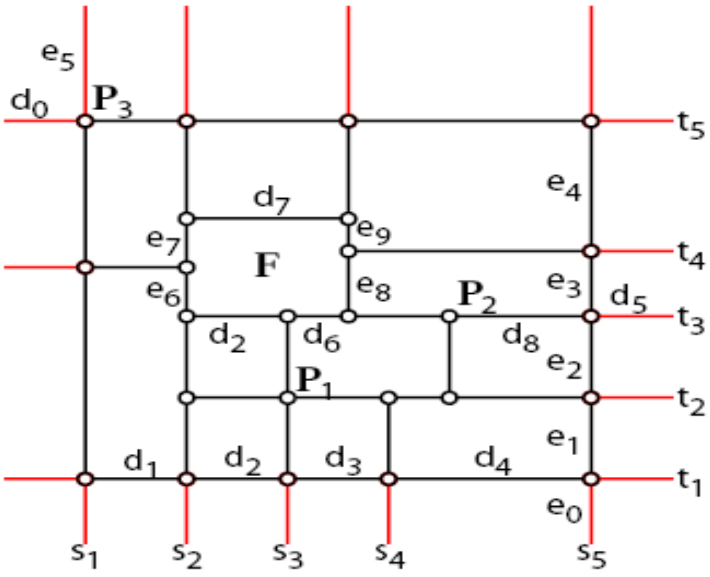


Figure 3-10: Example of T-mesh [11].

3.3.1 Control Point Insertion

The general practice of control point insertion using B-splines is that additional control points are inserted where needed, and the shape of the B-splines will be changed globally.

One of the advantages of T-splines over B-splines is that T-splines allow for local knot insertion, which is the procedure of inserting a single control point into a T-mesh without altering the shape of T-splines.

Local knot insertion requires executing knot insertion into all of the basis functions whose knot vectors will be altered by the insertion of the new control point. In Figure 3-11, only the basis functions of control points P_1, P_2, P_4, and P_5 are altered due to the presence of the new control point $P_{3'}$ [11].

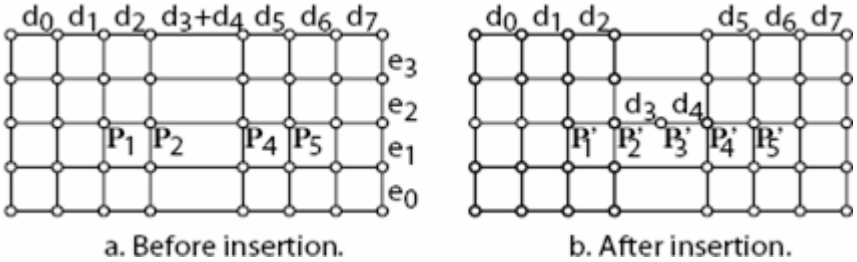


Figure 3-11: T-mesh knot insertion [11].

4 Finite Element Program

A simple finite element program for the analysis of membrane (plane stress) problems was developed. This program uses quadrilateral Lagrangian elements. The order of the shape functions is the square root of the number of nodes per element, minus one. Thus, the shape functions for the 4-node Lagrangian quad element are linear, the shape functions for the 9-node Lagrangian quad element are quadratic, etc. There is one shape function corresponding to each node in an element, and the i_{th} shape function has a value of one at node i and a value of zero at all other nodes $j \neq i$ in the element.

A single global coordinate system exists for the entire system of elements with the global x-coordinate in the horizontal direction and the global y-coordinate in the vertical direction. A natural coordinate system exists for each element. Two opposite sides of each quadrilateral element are arbitrarily designated as the "left" and "right" sides, while the other two opposite sides are designated as the "top" and "bottom" sides. The natural s-coordinate ranges from a value of -1 on the left side to +1 on the right side, and the natural t-coordinate ranges from a value of -1 on the bottom side to +1 on the top side. Opposite sides of an element need not be parallel, and in fact, the sides may be curved if there are more than two nodes on a side. The shape functions are functions of the natural coordinates s and t.

4.1 Input and Output

The geometry input to the finite element program includes the following scalars:

nnode = number of nodes in the system

nelem = number of elements in the system

nodel = number of nodes per element

nsupport = number of supports

E = Young's modulus of elasticity

ν = Poisson's ratio

H = membrane thickness

The geometry input to the finite element program includes the following arrays:

xsys[nnode][2] = global x and y coordinates of each node in system

conn[nelem][nodel] = list of node numbers connected to each element

support[nsupport][2] = node number and direction number for each support

The value of the direction number is either one, representing support restraint in the x direction, or two, representing support restraint in the y direction.

The program considers nodal loads and boundary loads. Nodal loads are point forces applied to unsupported nodes. Boundary loads are stress traction vectors applied on the sides of elements. Boundary loads may vary from element to element and from point to point along a side. The user must supply a function that evaluates the stress traction vector at a given point on the side of an element. The load input to the finite element program includes the following:

nnodeload = number of nodal loads

nodeload[nnodeload][2] = node number and direction number of each nodal load

point[nnodeload] = value of each nodal load

nboundload = number of boundary loads

boundload[nboundload][2] = element number and side number for each boundary load

npar = number of parameters needed for user function

boundpar[nboundload][npar] = boundary load parameters for each boundary load

The four sides of an element are ordered as follows: 1 = bottom, 2 = top, 3 = left, and 4 = right. The user-supplied function for boundary loads is named UserBoundLoad, and is given the boundary load parameters, the x and y coordinates of a point on the boundary, and the x and y components of an unit outward normal vector at the point. This function returns the x and y components of the stress traction vector at the point.

The program outputs global coordinates, displacements u_x and u_y, and stresses σ_{xx}, σ_{yy}, and σ_{xy} at each output point. The user must specify:

noutput = number of output points

eoutput[noutput] = element number of each output point

coutput[noutput][2] = natural s and t coordinates of each output point

4.2 Program Organization

The program is organized into five portions: 1) preprocessing, 2) stiffness matrix, 3) force vector, 4) equation solving, and 5) postprocessing. The preprocessing portion reads input data, allocates memory space for internal arrays, and numbers the degrees of

freedom (dof's). Degrees of freedom correspond to the x and y displacements at the unsupported nodes. The total number of dof's is ndof.

The stiffness matrix portion of the program assembles the system stiffness matrix, ksys[ndof][ndof]. The ksys matrix is assembled by looping through the elements and calling two functions. The first function, ElementStiff, constructs the element stiffness matrix, kelem[2*nodel][2*nodel]. The second function, ElemToSysStiff, adds the kelem matrix into the ksys matrix according to the connectivity of the elements to nodes and the numbering of the dof's.

The force vector portion of the program assembles the system force vector, fsys[ndof]. First, the nodal loads are added to the fsys vector according to the degree of freedom numbering for the nodes. Second, the boundary loads are added to fsys by looping through the elements and calling two functions. The first function, ElementForce, constructs the element force vector, felem[2*nodel]. The second function, ElemToSysForce, adds the felem vector into the fsys vector according to the connectivity of the elements to nodes and the numbering of the dof's.

The equation solving portion of the program solves the linear system of equations ksys*usys = fsys for the system displacement vector, usys[ndof]. The system is normally solved by triangularizing the symmetric system stiffness matrix, ksys, into the product of a lower triangular matrix, a diagonal matrix, and an upper triangular matrix. Then backsubstitution is performed to solve the lower triangular system, the diagonal system, and the upper triangular system of equations.

The postprocessing portion of the program evaluates and outputs the global coordinates and global components of displacement and stress at each output point. This

is accomplished by looping through the output points, obtaining the element number and s and t coordinates from the eoutput and coutput arrays, and calling two functions. The first function, SysToElemDisp, extracts the element displacement vector, uelem[2*nodel], from the usys vector according to the connectivity of the elements to nodes and the numbering of the dof's. The second function, ElementOutput, evaluates and outputs the coordinates, displacements, and stresses at the output point.

4.3 Element Functions

The ElementStiff function (Equation 4-1) evaluates the element stiffness matrix by evaluating the following integral by Gauss quadrature. The D matrix in Equation 4-1 is a 3 x 3 stiffness matrix involving Young's modulus and Poisson's ratio.

$$\text{kelem} = H\int_{-1}^{1}\int_{-1}^{1} B^T D B |J| ds dt \qquad \textbf{(4-1)}$$

$$D = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0 \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 \\ 0 & 0 & \frac{E}{2(1+\nu)} \end{bmatrix} \qquad \textbf{(4-2)}$$

The B matrix in Equation 4-1 is a 3 x 2*nodel matrix involving derivatives of the shape functions $N^{(1)}, N^{(2)}, \ldots, N^{(nodel)}$ with respect to global x and y coordinates.

$$B = \begin{bmatrix} \frac{\partial N^{(1)}}{\partial x} & 0 & \frac{\partial N^{(2)}}{\partial x} & 0 & \dots & 0 \\ 0 & \frac{\partial N^{(1)}}{\partial y} & 0 & \frac{\partial N^{(2)}}{\partial y} & \dots & \frac{\partial N^{(nodel)}}{\partial y} \\ \frac{\partial N^{(1)}}{\partial y} & \frac{\partial N^{(1)}}{\partial x} & \frac{\partial N^{(2)}}{\partial y} & \frac{\partial N^{(2)}}{\partial x} & \dots & \frac{\partial N^{(nodel)}}{\partial x} \end{bmatrix} \quad \textbf{(4-3)}$$

The shape functions are constructed in terms of the local s and t coordinates in Equation 4-4.

$$N^{(i)}(s,t) = S^{(k)}(s)T^{(m)}(t) \quad \textbf{(4-4)}$$

where:

$$n = \sqrt{nodel} \qquad k = 1+\text{remainder}\left(\frac{i-1}{n}\right) \qquad m = 1+\text{truncate}\left(\frac{i-1}{n}\right)$$

$$S^{(k)}(s) = \prod_{\substack{j=1 \\ j\neq k}}^{n} \frac{s - s^{(j)}}{s^{(k)} - s^{(j)}} \qquad T^{(m)}(t) = \prod_{\substack{j=1 \\ j\neq m}}^{n} \frac{t - t^{(j)}}{t^{(m)} - t^{(j)}}$$

$$s^{(j)} = 2\left(\frac{j-1}{n-1}\right) - 1 \qquad s^{(k)} = 2\left(\frac{k-1}{n-1}\right) - 1$$

$$t^{(j)} = 2\left(\frac{j-1}{n-1}\right) - 1 \qquad t^{(m)} = 2\left(\frac{m-1}{n-1}\right) - 1$$

The derivatives of the shape functions with respect to global x and y coordinates are obtained from the derivatives of the shape functions with respect to the natural s and t coordinates via the inverse Jacobian matrix (Equation 4-5)

$$\begin{bmatrix} \frac{\partial N^{(1)}}{\partial x} & \frac{\partial N^{(2)}}{\partial x} & \cdots & \frac{\partial N^{(nod)}}{\partial x} \\ \frac{\partial N^{(1)}}{\partial y} & \frac{\partial N^{(2)}}{\partial y} & \cdots & \frac{\partial N^{(nodel)}}{\partial y} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial N^{(1)}}{\partial s} & \frac{\partial N^{(2)}}{\partial s} & \cdots & \frac{\partial N^{(nodel)}}{\partial s} \\ \frac{\partial N^{(1)}}{\partial t} & \frac{\partial N^{(2)}}{\partial t} & \cdots & \frac{\partial N^{(nodel)}}{\partial t} \end{bmatrix} \quad \textbf{(4-5)}$$

The Jacobian matrix whose determinant is in Equation 4-1, and whose inverse is in Equation 4-5 is calculated from the natural derivatives of the shape functions and the global coordinates of the nodes (Equation 4-6).

$$J = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial N^{(1)}}{\partial s} & \frac{\partial N^{(2)}}{\partial s} & \cdots & \frac{\partial N^{(nodel)}}{\partial s} \\ \frac{\partial N^{(1)}}{\partial t} & \frac{\partial N^{(2)}}{\partial t} & \cdots & \frac{\partial N^{(nodel)}}{\partial t} \end{bmatrix} \begin{bmatrix} x^{(1)} & y^{(1)} \\ x^{(2)} & y^{(2)} \\ . & . \\ . & . \\ . & . \\ x^{(nodel)} & y^{(nodel)} \end{bmatrix} \quad \textbf{(4-6)}$$

In summary, the ElementStiff function loops through each Gauss point, whose natural coordinates are given, and:

1) Calls the Lagrange2D function that evaluates the shape functions and their natural derivatives at the Gauss point according to Equation 4-4. 2) Calls the Isoparametric2D function that evaluates Equations 4-6 and 4-5 at the Gauss point. 3)

Evaluates Equations 4-2 and 4-3 to get the D and B matrices at the Gauss point. 4) Computes the product $HB^T DB|J|W$ at the Gauss point where W is the Gauss weight, and adds this product to kelem.

The ElementForce function (Equation 4-7) evaluates the element force vector by evaluating the following integral by Gauss quadrature on the sides of the element that have boundary loads.

$$\text{felem} = H \int_{-1}^{1} A^T T \|J\| (d_s \text{ or } d_t) \qquad \textbf{(4-7)}$$

Equation 4-7 is integrated over d_s for the top and bottom sides with t held constant at -1 for the bottom side and +1 for the top side. Equation 4-7 is integrated over d_t for the left and right sides with s held constant at -1 for the left side and +1 for the right side. The A matrix in Equation (4-7) is a 2 x 2*nodel matrix involving shape functions N(1), N(2), ... N(nodel).

$$A = \begin{bmatrix} N^{(1)} & 0 & N^{(2)} & 0 & \ldots & 0 \\ 0 & N^{(1)} & 0 & N^{(2)} & \ldots & N^{(\text{nodel})} \end{bmatrix} \qquad \textbf{(4-8)}$$

The shape functions are constructed in terms of the local s and t coordinates according to Equation 4-4 given previously. The Jacobian matrix is constructed

according to Equation 4-6 given previously. The norm of the Jacobian in Equation 4-7 is determined from the elements of the Jacobian matrix:

$$\text{Top and bottom sides: } \|J\| = \sqrt{J_{11}^2 + J_{12}^2} \quad \text{Left and right sides: } \|J\| = \sqrt{J_{21}^2 + J_{22}^2} \qquad \textbf{(4-9)}$$

The T vector in Equation 4-7 is the stress traction vector, which is evaluated on the boundary by calling the user-supplied function, UserBoundLoad. This function is given the x and y coordinates of a Gauss point, which are determined from the natural coordinates of the Gauss point by evaluating the shape functions at the Gauss point and pre-multiplying by the global coordinates of the nodes.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(nodel)} \\ y^{(1)} & y^{(2)} & \dots & y^{(nodel)} \end{bmatrix} \begin{bmatrix} N^{(1)} \\ N^{(2)} \\ . \\ . \\ . \\ N^{(nodel)} \end{bmatrix} \qquad \textbf{(4-10)}$$

The UserBoundLoad function is also given the x and y components of a unit outward normal vector at the Gauss point. These components are different for each side of the element and are determined from the elements of the Jacobian matrix and the scalars in Equation 4-9.

Bottom side: $\begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} J_{12}/\|J\| \\ -J_{11}/\|J\| \end{bmatrix}$ Top side: $\begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} -J_{12}/\|J\| \\ J_{11}/\|J\| \end{bmatrix}$

Left side: $\begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} -J_{22}/\|J\| \\ J_{21}/\|J\| \end{bmatrix}$ Right side: $\begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} J_{22}/\|J\| \\ -J_{21}/\|J\| \end{bmatrix}$ **(4-11)**

In summary, the ElementForce function loops through each Gauss point on the loaded side of an element and:

- Calls the Lagrange2D function that evaluates the shape functions and their natural derivatives at the Gauss point according to Equation 4-4.
- Calls the Isoparametric2D function that evaluates Equations 4-6, 4-9, 4-10, and 4-11 at the Gauss point.
- Evaluates Equation 4-8 to get the A matrix at the Guass point, and calls the UserBoundLoad function to get the stress traction vector at the Gauss point.
- Computes the product $HA^T T\|J\|W$ at the Gauss point where W is the Gauss weight, and adds this product to felem.

The ElementOutput function evaluates the coordinates, displacements, and stresses at an output point by calling the Lagrange2D function to evaluate the shape functions and their natural derivatives according to Equation 4-4, and then calling the Isoparametric2D function to evaluate Equations 4-6, 4-5, and 4-10. Equation 4-10 gives the coordinates of the output point. Equation 4-8 is evaluated to get the a matrix, and the displacements at the output point are given in Equation 4-12.

Equations 4-2 and 4-3 are evaluated to get the D and B matrices, and the stresses at the output point are given in Equation 4-13.

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = A\ uelem \tag{4-12}$$

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \text{D B uelem} \tag{4-13}$$

4.4 Finite Element Program Using T-splines

The finite element program described previously was modified to use T-spline shape functions rather than Lagrange shape functions. To do so, the Lagrange2D function was replaced with the Tspline2D function. All other functions remain unchanged. In addition, the program input was modified slightly.

4.4.1 Modification to Program Input

The nodes in the finite element program become the control points in the T-spline program. The shape functions in the finite element program become the basis functions in the T-spline program. There is one basis function for each control point. The T-spline basis functions are functions of natural coordinates s and t, but there is only one natural coordinate system for the entire system of elements, rather than a separate natural

coordinate system for each element. Thus, the natural coordinates do not always range from -1 to +1 in each element as they do for Lagrange shape functions. In fact, the user must specify the range of s and t for each element as part of the program input:

srange[nelem][2] = minimum and maximum s value for each element

trange[nelem][2] = minimum and maximum t value for each element

The only Lagrange shape functions that are nonzero (i.e. that are supported) on the domain of an element are those corresponding to the nodes within the element domain. This is not true for T-spline basis functions. There are T-spline basis functions corresponding to control points lying outside the domain of an element that may be supported on the domain of the element. If nodel is the number of T-spline basis functions supported on the domain of an element, it's possible that nodel may vary from element to element. In the T-spline program, the conservative assumption is made to set nodel equal to the total number of nodes in the system, and the connectivity matrix conn[nelem][nodel] is set to be the same for every element, consecutively listing all control points in the system. Thus, the scalar nodel and the conn matrix are not input by the user in the T-spline program.

To calculate the value of the ith T-spline basis function, it is necessary to know the s and t values at the ith control point and at two adjacent lines of control points on either side of the ith control point. In addition, it is necessary to know the weighting value corresponding to the ith control point. This information must be input by the user:

sval[nnode][5] = s values of five control point lines for each control point

tval[nnode][5] = t values of five control point lines for each control point

wval[nnode] = weight value for each control point

The five control point lines include two preceding control point lines, the control point iteself, and two succeeding control point lines.

4.4.2 The Tspline2D Function

The Tspline2D function is given the sval, tval, and wval arrays that were input by the user. It is also given the s and t coordinates of a Gauss point. It returns the values of all basis functions evaluated at the Gauss point as well as the values of the derivatives of all basis functions with respect to s and t evaluated at the Gauss point. This function computes the value of the ith basis function and its derivatives, where i ranges from one to nnode, as follows:

$$N^{(i)}(s,t) = \frac{W^{(i)}S^{(i)}(s)T^{(i)}(t)}{\sum_{j=1}^{nnode} W^{(j)}S^{(j)}(s)T^{(j)}(t)} \tag{4-14}$$

$$\frac{\partial N^{(i)}}{\partial s}(s,t) = \frac{W^{(i)}\frac{dS^{(i)}}{ds}(s)T^{(i)}(t) - N^{(i)}(s,t)\sum_{j=1}^{nnode} W^{(j)}\frac{dS^{(j)}}{ds}(s)T^{(j)}(t)}{\sum_{j=1}^{nnode} W^{(j)}S^{(j)}(s)T^{(j)}(t)} \tag{4-15}$$

$$\frac{\partial N^{(i)}}{\partial t}(s,t) = \frac{W^{(i)}S^{(i)}(s)\frac{dT^{(i)}}{dt}(t) - N^{(i)}(s,t)\sum_{j=1}^{nnode} W^{(j)}S^{(j)}(s)\frac{dT^{(j)}}{dt}(t)}{\sum_{j=1}^{nnode} W^{(j)}S^{(j)}(s)T^{(j)}(t)} \tag{4-16}$$

In Equations 4-14, 4-15 and 4-16, W(i) = wval[i]. The values of S(i)(s) and $\frac{dS^{(i)}}{ds}(s)$ are calculated by sending the s-coordinate of the Gauss point as well as sval[i][1], sval[i][2], sval[i][3], sval[i][4], and sval[i][5] to the TsplineDeBoor function. The values of T(i)(t) and $\frac{dT^{(i)}}{dt}(t)$ are calculated by sending the t-coordinate of the Gauss point as well as tval[i][1], tval[i][2], tval[i][3], tval[i][4], and tval[i][5] to the TsplineDeBoor function. Equations 4-14, 4-15 and 4-16 are only evaluated for the i_{th} basis function if the s and t coordinates of the Gauss point fall in the range:

sval[i][1] ≤ s ≤ sval[i][5]

tval[i][1] ≤ t ≤ tval[i][5]

Otherwise, $N^{(i)}(s,t)$ and its derivatives are set to zero.

4.4.3 The TsplineDeBoor Function

The TsplineDeBoor function is called with either s values and t values by the Tspline2D function. It is given the coordinate, s (or t), of a Gauss point, and it is given five values, val[1] through val[5], representing control point lines of s (or t). It returns the value of a function, S (or T), and its derivative, dS (or dT), evaluated at s (or t).

The TsplineDeBoor function begins by determining the interval into which the Gauss point falls:

if (val[1] ≤ s ≤ val[2] and val[1] ≠ val[2])

k = 1 f_2 = s−val[1]

else if (val[2] ≤ s ≤ val[3] and val[2] ≠ val[3])

k = 2 $f_1 = s - val[1]$ $f_2 = val[5] - s$

else if ($val[3] \le s \le val[4]$ and $val[3] \ne val[4]$)

k = 3 $f_1 = val[5] - s$ $f_2 = s - val[1]$

else if ($val[4] \le s \le val[5]$ and $val[4] \ne val[5]$)

k = 4 $f_2 = val[5] - s$

else

return S = dS = 0

Then function values are calculated by the following DeBoor algorithm:

if $(2 \le k \le 3)$ $h_1 = \dfrac{f_1}{val[k+2] - val[k-1]}$ else $h_1 = 0$

if $(1 \le k \le 2)$ $h_2 = \dfrac{f_2}{val[k+3] - val[k]}$ else $h_2 = 0$

if $(3 \le k \le 4)$ $h_3 = \dfrac{f_2}{val[k+1] - val[k-2]}$ else $h_3 = 0$

if $(1 \le k \le 3)$ $h_4 = \dfrac{h_1(val[k+2] - s) + h_2(s - val[k])}{val[k+2] - val[k]}$ else $h_4 = 0$

if $(2 \le k \le 4)$ $h_5 = \dfrac{h_1(s - val[k-1]) + h_3(val[k+1] - s)}{val[k+1] - val[k-1]}$ else $h_5 = 0$

$$S = \frac{h_4(s - val[k]) + h_5(val[k+1] - s)}{val[k+1] - val[k]}$$

$$dS = \frac{3(h_4 - h_5)}{val[k+1] - val[k]}$$

5 Test Example

A membrane is evaluated by the finite element program with T-spline shape functions. The stresses at the points of interest on the membrane using both T-splines and B-splines are obtained. Both results are compared to the closed-form solution.

The membrane is an infinite plate with circular hole under constant in-plane tension. Specifically, the membrane is subjected to tensile stress p; the radius of the circular hole α is equal to 5 in; the width of the membrane is equal to 40 in; the thickness of the membrane is equal to 0.1 in; the Poisson's ratio ν is equal to 0.3 and the Young's modulus E equals 10000 ksi. For simplicity and analysis reasons, the infinite membrane is reduced to a finite quarter plate. Figure 5-1 and Figure 5-2 show respectively the original infinite membrane and the simplified finite quarter plate.

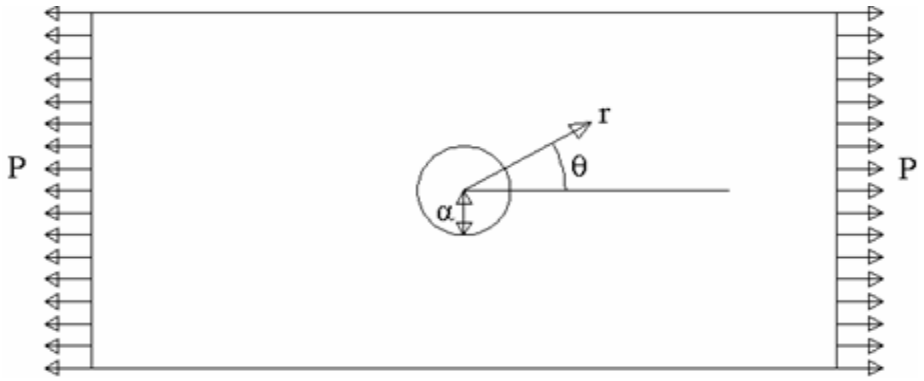


Figure 5-1: Membrane with a circular hole.

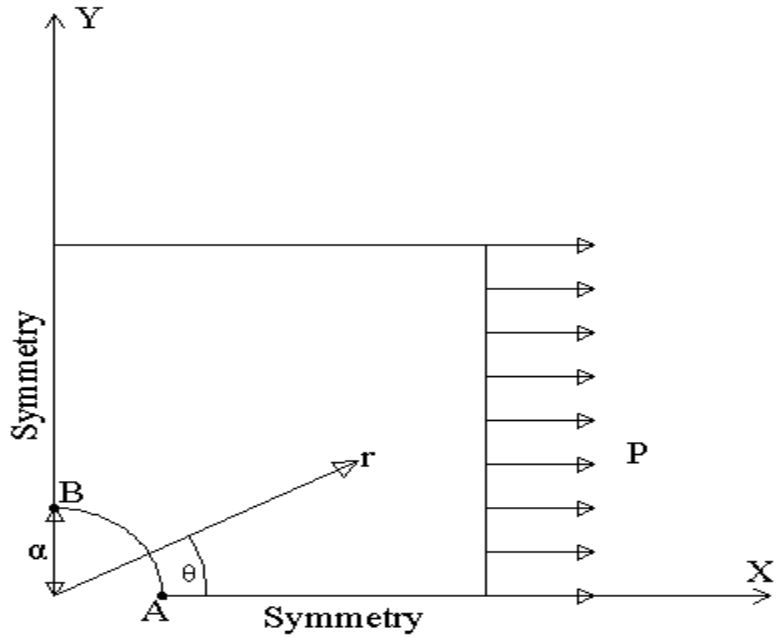


Figure 5-2: Finite quarter membrane with a circular hole.

5.1 B-spline Model (B.1)

The initial B-spline model (B.1) includes 16 control points. The bottom and left sides of the model are restrained by the nodal supports. There are four nodal supports on each side. The bottom side of the model is restrained in Y direction. The left side of the model is restrained in X direction. Only the right side of the system is subject to the uniform load of 100 psi and there are no nodal loads involved in this model. During the process of the determination of the stiffness matrix, Gaussian quadrature method is adopted to evaluate the integral of the stiffness matrix formula. 64 gauss points are required to integrate the polynomial T-spline basis functions. The whole model has only one region. S and t coordinates of that region range from 0 to 1 and 0 to 2, respectively.

There are three points of interest, whose displacements, stresses and norms are evaluated. Figure 5-3 illustrates the B-spline model (B.1) with three output points (point 1, 4 and the middle point of arc 14, which are designated as point A, C and B, respectively.

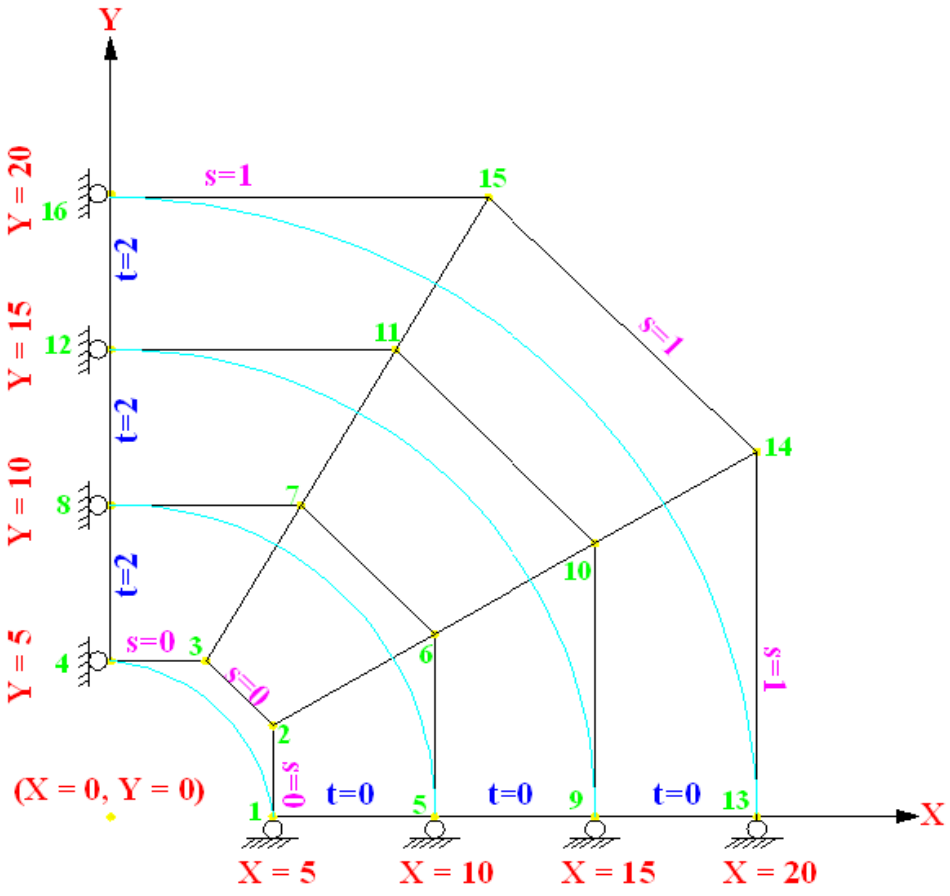


Figure 5-3: B-spline model (B.1).

Each control point corresponds to two knot vectors. The knot coordinates of a control point P_i are designated as (s_{i2}, t_{i2}). Thus, the knot vectors of P_{si} are denoted as $s_i = (s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4})$ and $t_i = (t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4})$. The knots t_{i3}, t_{i4} can be determined by imposing a ray in the parameter space $R(\alpha) = (s_{i2}, t_{i2} + \alpha)$. Thus, t_{i3}, t_{i4} become the t

coordinates of the first t-edges intersected by the ray excluding the initial (s_{i2}, t_{i2}). The knot vectors for B-spline Model (B.1) are listed in the Table 5-1.

Table 5-1: Knot vectors for B-spline model (B.1).

Control Point	***knot vectors in terms of s coordinate***				
1	*0*	*0*	*0*	*0*	*1*
2	*0*	*0*	*0*	*0*	*1*
3	*0*	*0*	*0*	*0*	*1*
4	*0*	*0*	*0*	*0*	*1*
5	*0*	*0*	*0*	*1*	*1*
6	*0*	*0*	*0*	*1*	*1*
7	*0*	*0*	*0*	*1*	*1*
8	*0*	*0*	*0*	*1*	*1*
9	*0*	*0*	*1*	*1*	*1*
10	*0*	*0*	*1*	*1*	*1*
11	*0*	*0*	*1*	*1*	*1*
12	*0*	*0*	*1*	*1*	*1*
13	*0*	*1*	*1*	*1*	*1*
14	*0*	*1*	*1*	*1*	*1*
15	*0*	*1*	*1*	*1*	*1*
16	*0*	*1*	*1*	*1*	*1*
Control Point	***knot vectors in terms of t coordinate***				
1	*0*	*0*	*0*	*0*	*2*
2	*0*	*0*	*0*	*2*	*2*
3	*0*	*0*	*2*	*2*	*2*
4	*0*	*2*	*2*	*2*	*2*
5	*0*	*0*	*0*	*0*	*2*
6	*0*	*0*	*0*	*2*	*2*
7	*0*	*0*	*2*	*2*	*2*
8	*0*	*2*	*2*	*2*	*2*
9	*0*	*0*	*0*	*0*	*2*
10	*0*	*0*	*0*	*2*	*2*
11	*0*	*0*	*2*	*2*	*2*
12	*0*	*2*	*2*	*2*	*2*
13	*0*	*0*	*0*	*0*	*2*
14	*0*	*0*	*0*	*2*	*2*
15	*0*	*0*	*2*	*2*	*2*
16	*0*	*2*	*2*	*2*	*2*

The weights of all control points are equal to one except for the ones of control points 2 and 3. According to the equations shown in Figure 3-4, the weights of control points 2 and 3 are evaluated as follows, w = (1 + 2cos(90°)/2)/3 = 0.804738.

Equation 3-7 provides a formula to determine Y coordinate of control point 2 as follows, $y = e = \frac{2\sin\frac{\theta}{2}}{1+2\cos\frac{\theta}{2}} r = \frac{2\sin\frac{90°}{2}}{1+2\cos\frac{90°}{2}} \times 5 =$ 2.928932. X and Y coordinates of other control points can be determined in the same manner. Table 5-2 shows the X and Y coordinates of all control points for B-spline Model (B.1).

Table 5-2: X and Y coordinates of all control points for B-spline model (B.1).

Control Point	***X***	***Y***
1	*5*	*0*
2	*5*	*2.928932*
3	*2.928932*	*5*
4	*0*	*5*
5	*10*	*0*
6	*10*	*5.857864*
7	*5.857864*	*10*
8	*0*	*10*
9	*15*	*0*
10	*15*	*8.786797*
11	*8.786797*	*15*
12	*0*	*15*
13	*20*	*0*
14	*20*	*11.71573*
15	*11.71573*	*20*
16	*0*	*20*

5.2 B-spline Model (B.2)

Based on the B-spline model (B.1), a B-spline model (B.2) is created by inserting four new control points (s = 0.5) on all four radial control grids of B.1 as illustrated in Figure 5-4. The knot vectors of all control points for B-spline Model (B.2) are listed in Table 5-3.

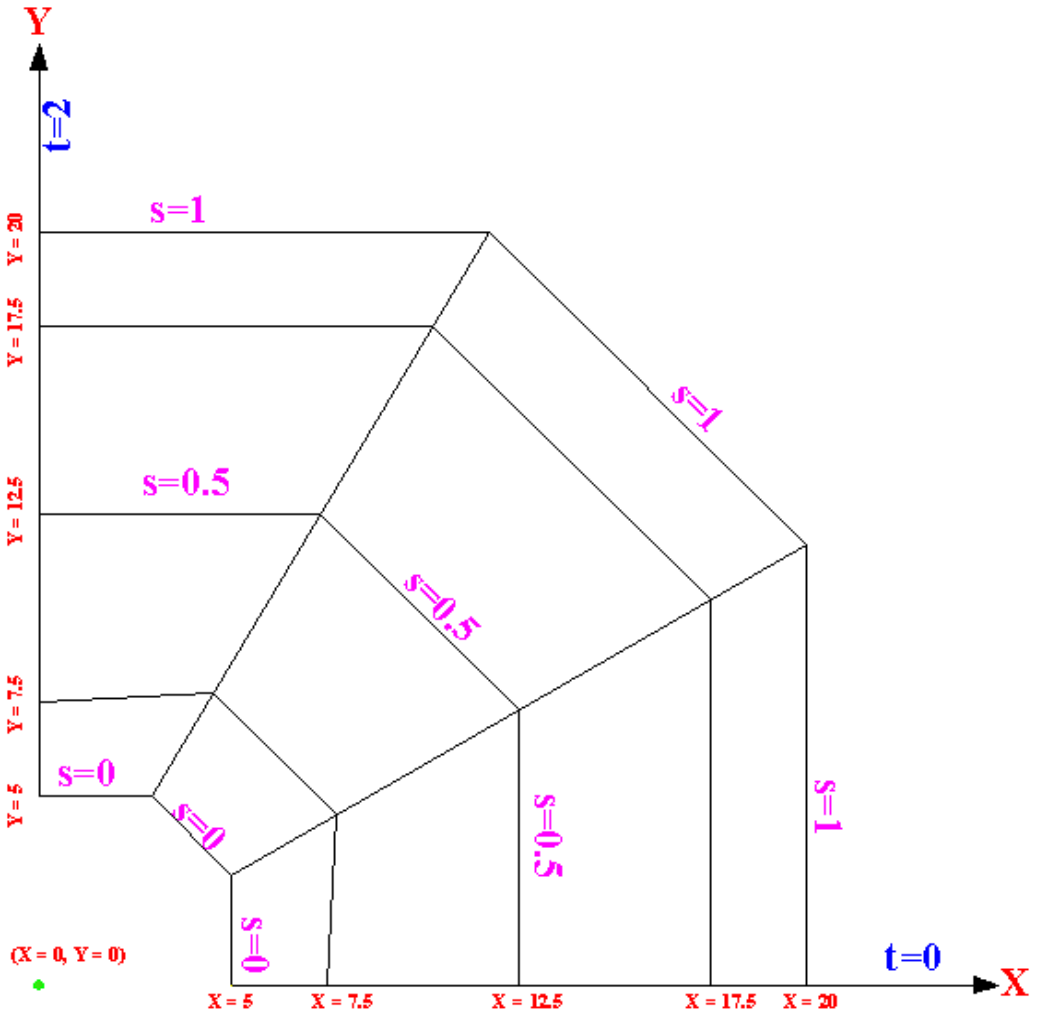


Figure 5-4: B-spline model (B.2).

Table 5-3: Knot vectors of all control points for B-spline model (B.2).

Control Point	***knot vectors in terms of s coordinate***				
1	0	0	0	0	0.5
2	0	0	0	0	0.5
3	0	0	0	0	0.5
4	0	0	0	0	0.5
5	0	0	0	0.5	1
6	0	0	0	0.5	1
7	0	0	0	0.5	1
8	0	0	0	0.5	1
9	0	0	0.5	1	1
10	0	0	0.5	1	1
11	0	0	0.5	1	1
12	0	0	0.5	1	1
13	0	0.5	1	1	1
14	0	0.5	1	1	1
15	0	0.5	1	1	1
16	0	0.5	1	1	1
17	0.5	1	1	1	1
18	0.5	1	1	1	1
19	0.5	1	1	1	1
20	0.5	1	1	1	1
Control Point	***knot vectors in terms of t coordinate***				
1	0	0	0	0	2
2	0	0	0	2	2
3	0	0	2	2	2
4	0	2	2	2	2
5	0	0	0	0	2
6	0	0	0	2	2
7	0	0	2	2	2
8	0	2	2	2	2
9	0	0	0	0	2
10	0	0	0	2	2
11	0	0	2	2	2
12	0	2	2	2	2
13	0	0	0	0	2
14	0	0	0	2	2
15	0	0	2	2	2
16	0	2	2	2	2
17	0	0	0	0	2
18	0	0	0	2	2
19	0	0	2	2	2
20	0	2	2	2	2

Table 5-4: X and Y coordinates of all control points for B-spline model (B.2).

Control Point	*X*	*Y*
1	5	0
2	5	2.928932
3	2.928932	5
4	0	5
5	7.5	0
6	7.770485	4.551845
7	4.551845	7.770485
8	0	7.5
9	12.5	0
10	12.5	7.32233
11	7.32233	12.5
12	0	12.5
13	17.5	0
14	17.5	10.25126
15	10.25126	17.5
16	0	17.5
17	20	0
18	20	11.71573
19	11.71573	20
20	0	20

Due to the insertion of the new control point 10 shown in Figure 5-4, the locations of the existing adjacent control points 6 and 10 shown in Figure 5.3 are altered. Those existing adjacent control points become the control points 6 and 14 of the B-spline model (B.2). Control points at the ends of the control grids remain fixed in this case. In addition, the weights of the existing adjacent control points are also affected. The weights of control points 6, 10 and 14 of the B-spline model (B.2) are calculated in Equation 5-1.

$$w_{(\text{Control Point } 6, B.2)} = \frac{(1-0.5)\times w_{(2,B.1)} + (0.5-0)\times w_{(6,B.1)}}{(1-0)} = 0.902369$$

$$w_{(\text{Control Point } 14, B.2)} = \frac{(1-0.5)\times w_{(10,B.1)} + (0.5-0)\times w_{(14,B.1)}}{(1-0)} = 1$$

$$w_{(\text{Control Point } 10, B.2)} = \frac{(1-0.5)\times w_{(6,B.1)} + (0.5-0)\times w_{(10,B.1)}}{(1-0)} = 1 \qquad \textbf{(5-1)}$$

Weights of all the other control points can be determined in like manner.

Since the B-spline curves involved in this case are non-uniform rational degree three B-spline curves. The coordinates of the control point 6 of the B-spline model (B.2) are evaluated using Equation 5-2 as follows:

$$X_{(6,B.2)} = \frac{(1-0.5)\times w_{(2,B.1)}\times X_{(2,B.1)} + (0.5-0)\times w_{(6,B.1)}\times X_{(6,B.1)}}{(1-0)\times w_{(6,B..2)}} = 7.77$$

$$Y_{(6,B.2)} = \frac{(1-0.5)\times w_{(2,B.1)}\times Y_{(2,B.1)} + (0.5-0)\times w_{(6,B.1)}\times Y_{(6,B.1)}}{(1-0)\times w_{(6,B..2)}} = 4.55 \qquad \textbf{(5-2)}$$

The X and Y coordinates of all the other control points are determined in like manner.

5.3 B-spline Model (B.3)

Based on the B-spline model (B.2), a B-spline model (B.3) is created by inserting four new control points (s = 0.25) on all four radial control grids of B.2 as illustrated in Figure 5-5.

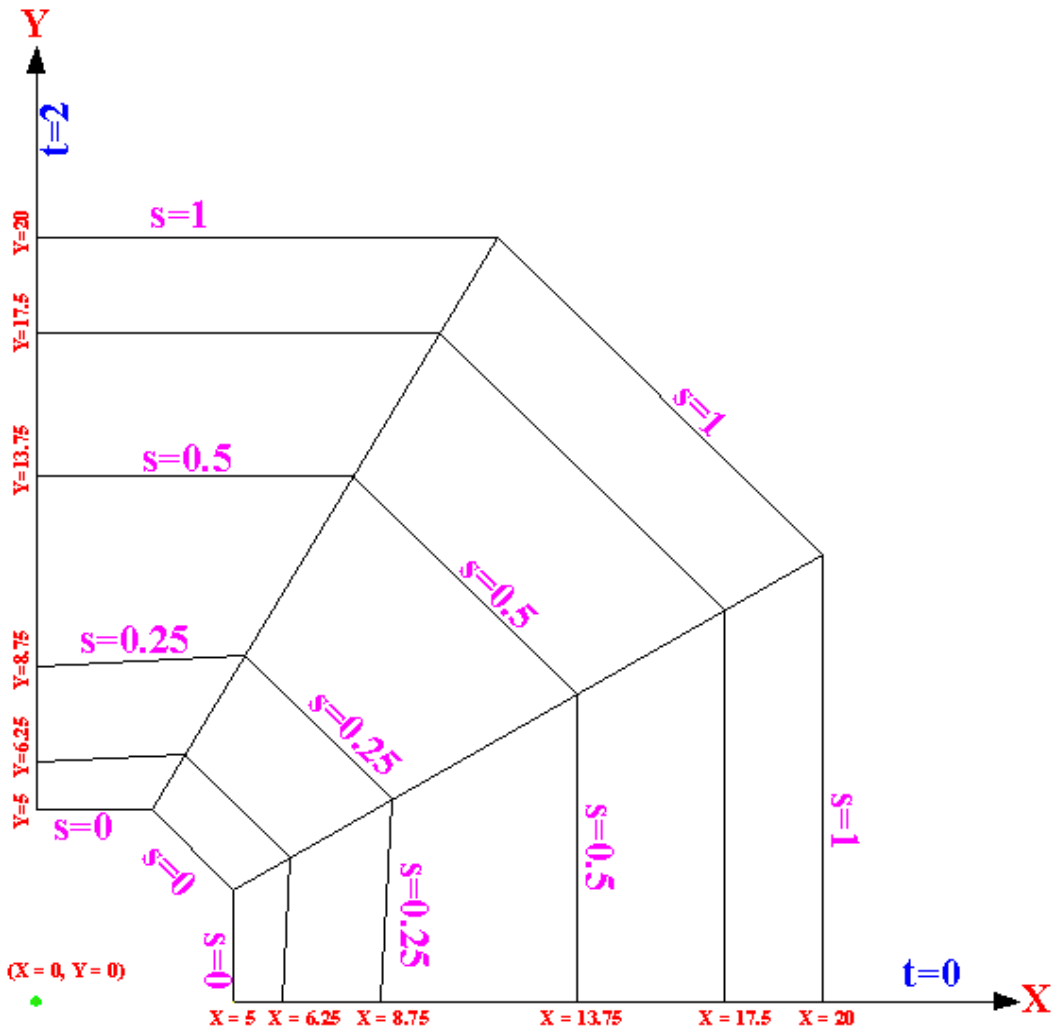


Figure 5-5: B-spline model (B.3).

5.4 B-spline Model (B.4)

Based on the B-spline model (B.3), a B-spline model (B.4) is created by inserting four new control points (s = 0.125) on all four radial control grids of B.3 as illustrated in Figure 5-6.

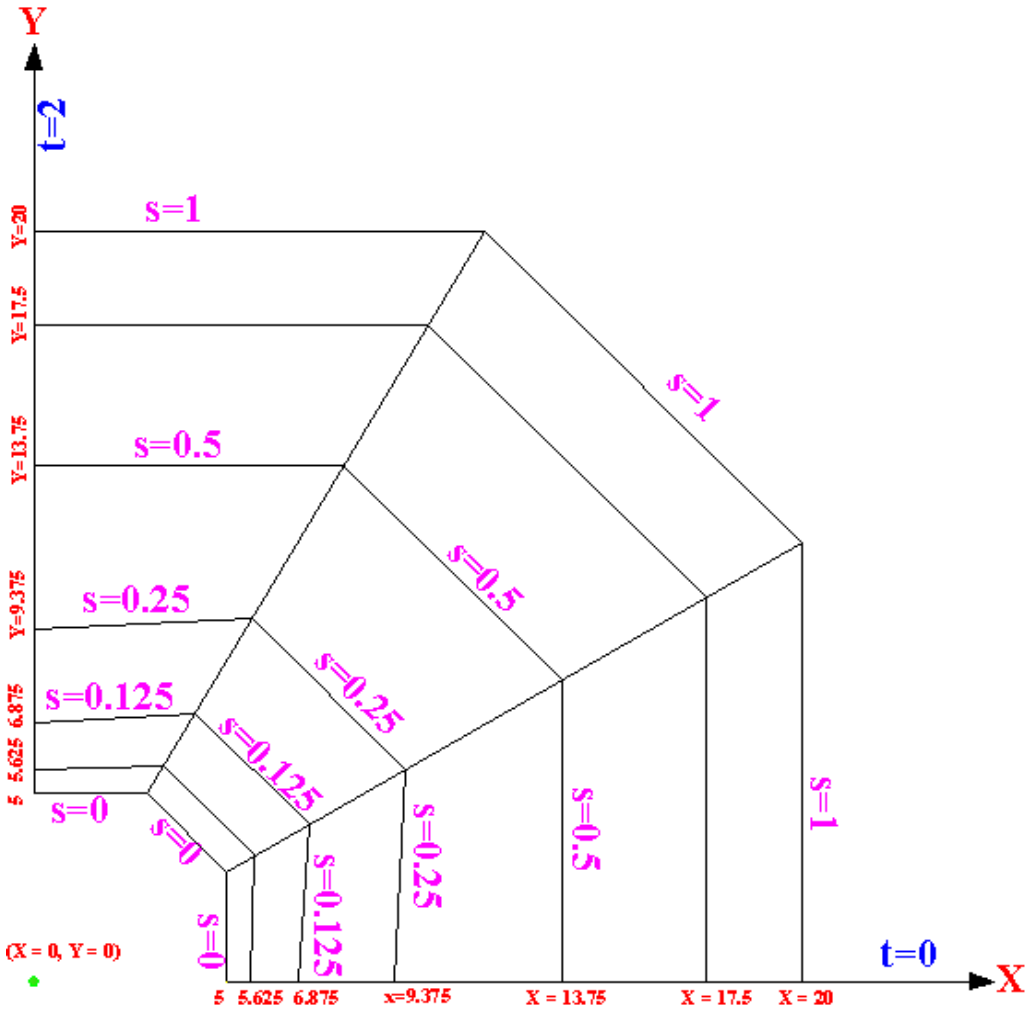


Figure 5-6: B-spline model (B.4).

5.5 B-spline Model (B.5)

From subchapter 5.1 to 5.4, circumferential refinements have been performed. Starting form this model, radial refinements are conducted. Based on B.4, B.5 is generated by adding one additional radial control line (t = 1) propagating the entire control girds as shown in Figure 5-7.

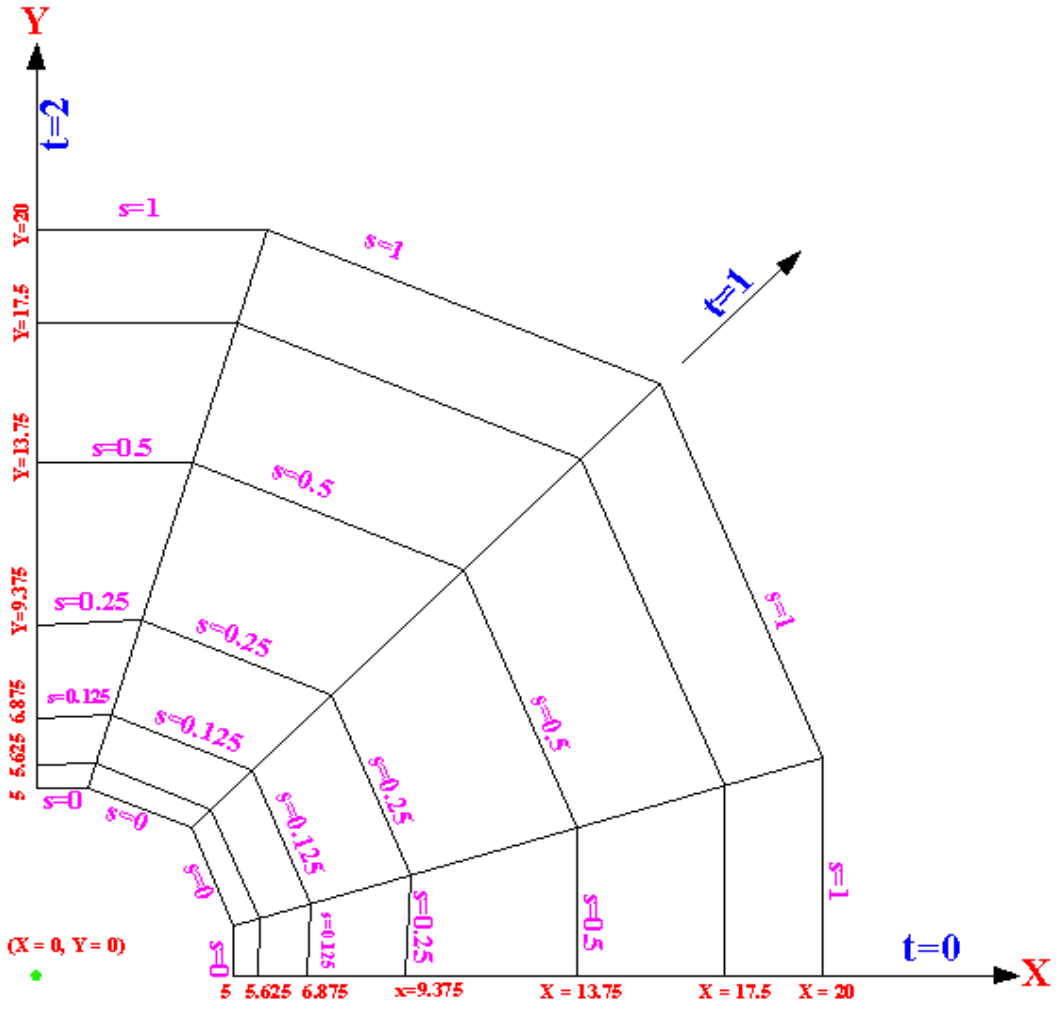


Figure 5-7: B-spline model (B.5).

5.6 B-spline Model (B.6)

Based on B.5, B.6 is generated by adding two additional radial control lines (t = 0.5 and t = 1.5) propagating the entire control girds as shown in Figure 5-8.

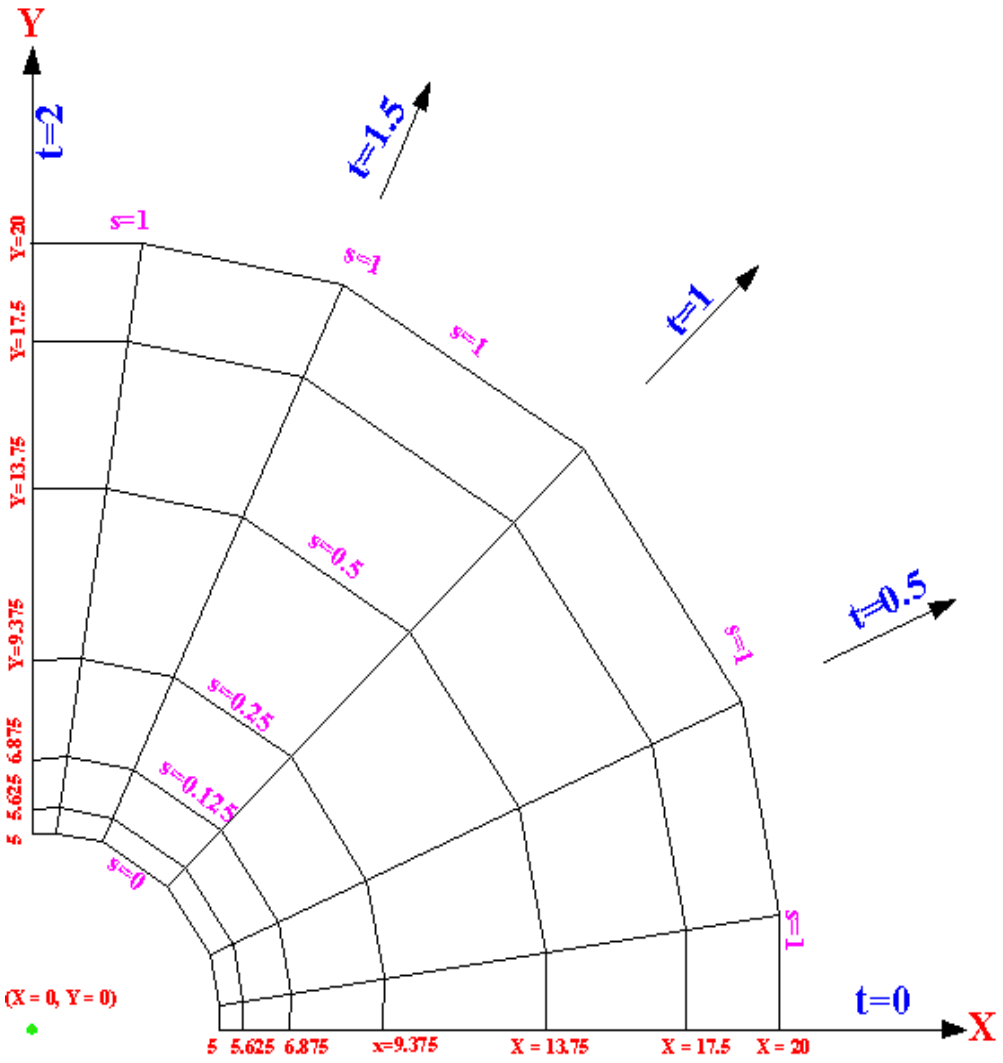


Figure 5-8: B-spline model (B.6).

5.7 T-spline Model (T.5)

In this section, local refinement is performed by inserting partial control line into the control grid. Based on B.4, a T-spline model (T.5) is generated by adding a radial control line (t = 1), which terminates at the circumferential control line (s = 0.125) as shown in Figure 5-9.

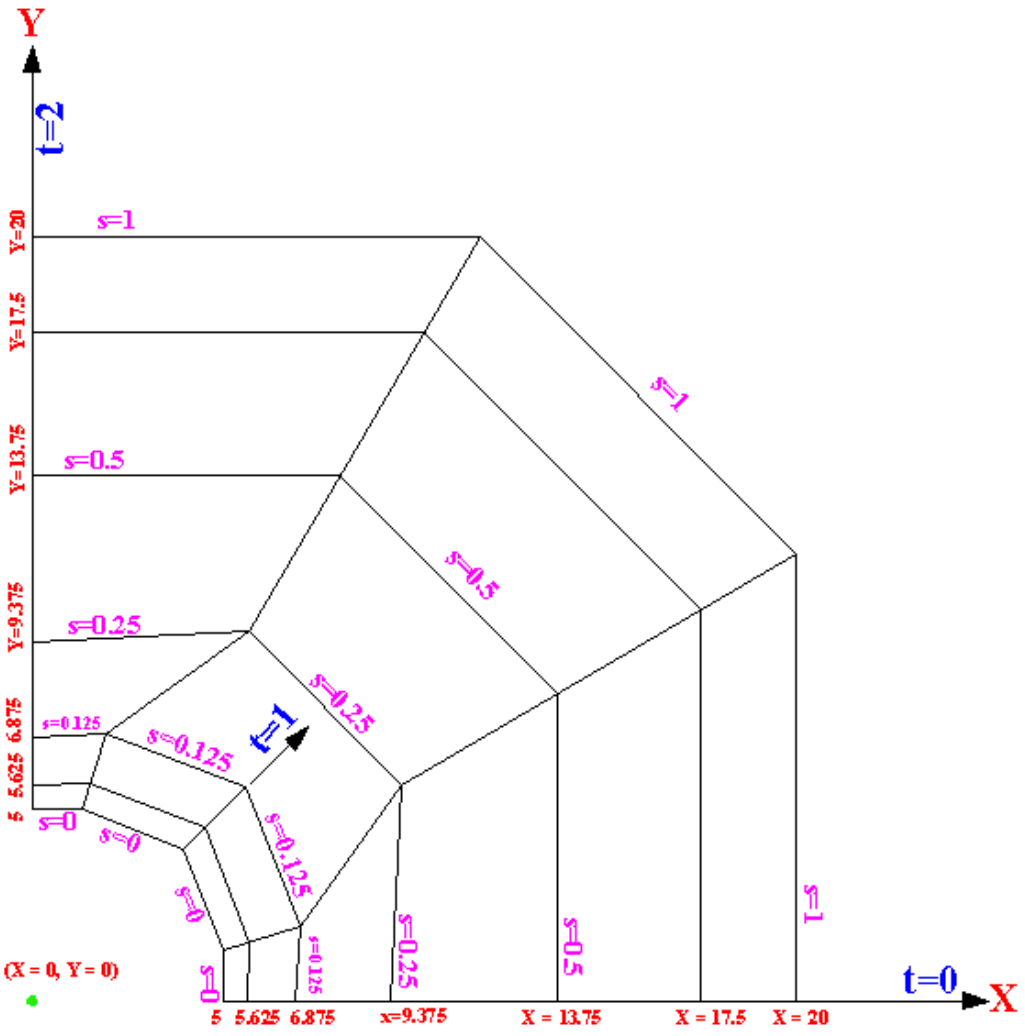


Figure 5-9: T-spline model (T.5).

5.8 T-spline Model (T.6)

Based on T.5, a T-spline model (T.6) is generated by adding two partial radial control lines (t = 0.5 and t =1.5), which terminate at the circumferential control line (s = 0.125) as shown in Figure 5-10.

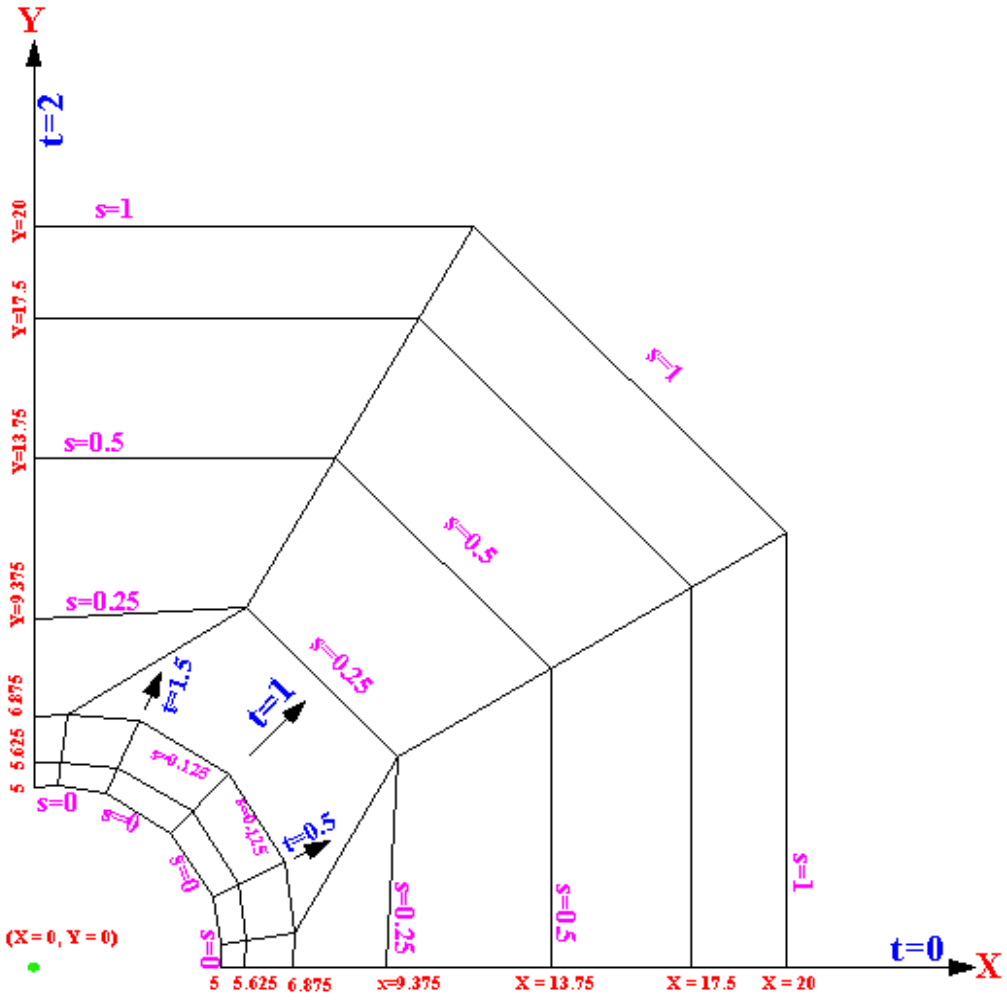


Figure 5-10: T-spline model (T.6).

6 Results

All the models were constructed in the previous chapter. The required information for the input files of the finite element program using T-splines were obtained. In this chapter, the results are determined by performing the finite element program and are compared with the closed-form solution.

6.1 Closed Form Solution

The closed-form solution using a cylindrical coordinate system to the membrane problem was solved by Kirsch. Equations 6-1, 6-2 and 6-3 provide formulas for the stresses $\sigma_{\theta\theta}$, σ_{rr} and $\sigma_{r\theta}$ of a point in the membrane [22].

$$\sigma_{rr} = \frac{p}{2}\left(1-\frac{\alpha^2}{r^2}+\left(1-\frac{4\alpha^2}{r^2}+\frac{3\alpha^4}{r^4}\right)\cos 2\theta\right) \quad \textbf{(6-1)}$$

$$\sigma_{\theta\theta} = \frac{p}{2}\left(1+\frac{\alpha^2}{r^2}-\left(1+\frac{3\alpha^4}{r^4}\right)\cos 2\theta\right) \quad \textbf{(6-2)}$$

$$\sigma_{r\theta} = -\frac{p}{2}\left(1+\frac{2\alpha^2}{r^2}-\frac{3\alpha^4}{r^4}\right)\sin 2\theta \quad \textbf{(6-3)}$$

The transformation of the stress matrix from cylindrical to rectangular coordinate system is given in Equation 6-4 [22].

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_{rr} \\ \sigma_{\theta\theta} \\ \sigma_{r\theta} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(6-4)}$$

Equation 6-4 is then evaluated to determine the closed-form solution using a rectangular coordinate system to the membrane problem at three points of interest (point A, C, and B), which is indicated in Table 6-1. The three points of interest (point A, C and B) are points 1, 4 and the middle point of arc 14 shown in Figure 5-3.

Table 6-1: Closed form solution using rectangular coordinate system

	$\sigma_{\theta\theta}$	σ_{rr}	$\sigma_{r\theta}$
Point A	0	-100	0
Point B	50	50	-50
Point C	300	0	0

6.2 Output Results

Table 6-2 is the output file of the finite element program for the B-spline model (B.1), which includes the stresses and displacements at the points of interest and the corresponding norms. Tables 6-3, 6-4, 6-5, 6-6, 6-7, 6-8, 6-9 show the output results for models B.2, B.3, B.4, B.5 , B.6, T.5 and T.6, respectively.

Table 6-2: Output results of B-spline model (B.1).

B-spline model (B.1)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-15.4208	-84.6847	-10.7170
Point B	111.0062	11.6209	-41.7564
Point C	299.8741	57.6585	11.6407
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.143884	0	
Point B	0.098318	-0.02812	
Point C	0	-0.04594	
	Square Root Norm		96.48783
	Maximum Norm		61.00617

Table 6-3: Output results of B-spline model (B.2).

B-spline Model (B.2)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-21.1879	-101.5465	-11.2417
Point B	93.4618	20.8630	-44.9467
Point C	312.3041	45.8567	11.6653
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.146534	0	
Point B	0.101156	-0.03041	
Point C	0	-0.04739	
	Square Root Norm		75.70623
	Maximum Norm		45.85675

Table 6-4: Output results of B-spline model (B.3).

B-spline Model (B.3)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-10.7025	-109.628	-11.2842
Point B	71.98185	33.7959	-47.7359
Point C	315.6899	21.28265	11.77187
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.14808	0	
Point B	0.103031	-0.03213	
Point C	0	-0.04855	
	Square Root Norm		43.85399
	Maximum Norm		21.98185

Table 6-5: Output results of B-spline model (B.4).

B-spline Model (B.4)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	4.274096	**-107.297**	-11.2553
Point B	60.31171	41.52561	-48.8883
Point C	**311.7068**	1.657165	11.80785
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.148352	0	
Point B	0.103372	-0.03246	
Point C	0	-0.04879	
	Square Root Norm		25.628
	Maximum Norm		11.80785

Table 6-6: Output results of B-spline model (B.5).

B-spline Model (B.5)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-11.2603	-107.843	-1.9467
Point B	55.43312	45.71679	-49.5358
Point C	308.6477	13.61196	1.707825
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.150206	0	
Point B	0.105766	-0.03499	
Point C	0	-0.05026	
	Square Root Norm		22.43121
	Maximum Norm		13.61196

Table 6-7: Output results of B-spline model (B.6).

B-spline Model (B.6)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-4.78673	-101.382	0.198392
Point B	55.68806	45.81124	-49.5799
Point C	301.8389	6.217114	-0.22238
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.149984	0	
Point B	0.106015	-0.0353	
Point C	0	-0.04999	
	Square Root Norm		10.81766
	Maximum Norm		6.217114

Table 6-8: Output results of T-spline model (T.5).

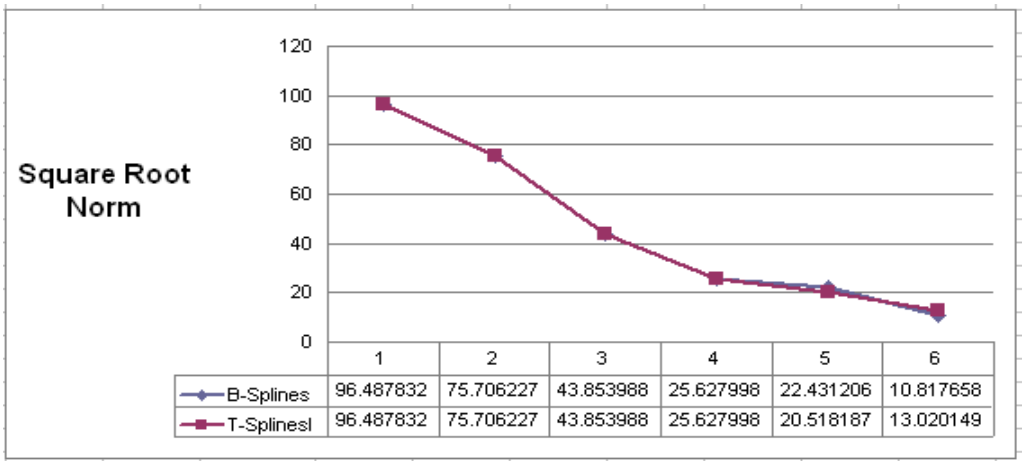
T-spline Model (T.5)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-9.26513	-105.186	-2.01601
Point B	57.52785	43.78103	-49.4035
Point C	306.7462	12.62881	1.870246
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.148253	0	
Point B	0.104395	-0.0336	
Point C	0	-0.04837	
	Square Root Norm		20.51819
	Maximum Norm		12.62881

Table 6-9: Output results of T-spline model (T.6)

T-spline Model (T.6)			
	Stresses		
	σ_{11}	**σ_{22}**	**σ_{12}**
Point A	-5.26607	-100.92	-0.17832
Point B	56.49355	45.14399	-49.1476
Point C	303.3424	7.953875	0.071057
	Coordinates (X and Y)		
Point A	5	0	
Point B	3.535534	3.535534	
Point C	0	5	
	Displacements (X and Y)		
Point A	0.147922	0	
Point B	0.104713	-0.03384	
Point C	0	-0.04817	
	Square Root Norm		13.02015
	Maximum Norm		7.953875

6.3 Comparison of Norms

Figure 6-1 illustrates comparison of square root norms, which are square root of the sum of the square of the difference between closed form solution and σ_{11}, σ_{22} and σ_{12} of all three points of interest created by the finite element program using B-splines and using T-splines. By observation, square root norms for B-spline models and its corresponding T-spline models are really close. In addition, the square root norms converge to zero when more control points are inserted in the control grid.



	1	2	3	4	5	6
B-Splines	96.487832	75.706227	43.853988	25.627998	22.431206	10.817658
T-Splinesl	96.487832	75.706227	43.853988	25.627998	20.518187	13.020149

Figure 6-1: Comparison of square root norms.

Figure 6-2 illustrates comparison of maximum norms, which are maximum absolute difference between closed form solution and σ_{11}, σ_{22} and σ_{12} of all three points of interest generated by finite element program using B-splines and using T-splines. By observation, max norms for B-spline models and its corresponding T-spline models are really close. In addition, the max norms converge to zero when more control points are inserted in the control grid.

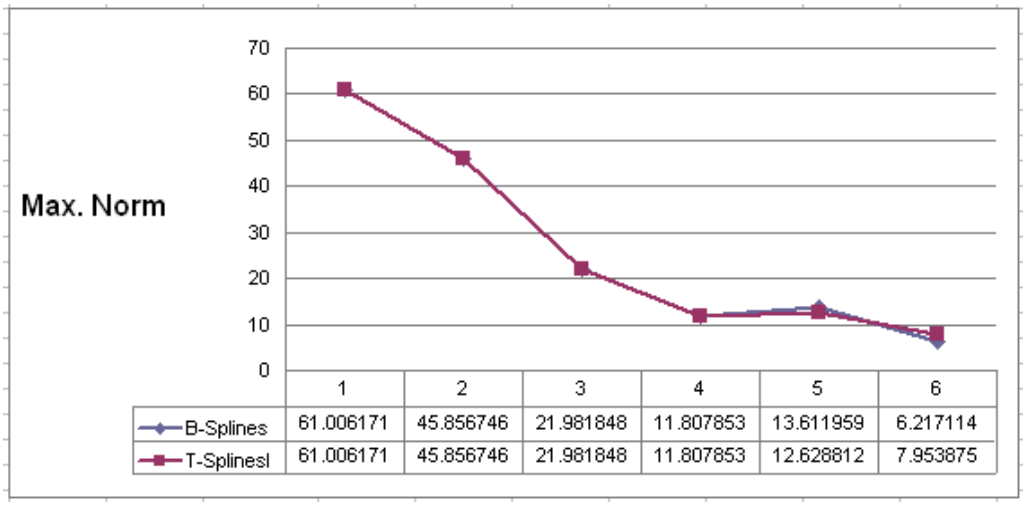


Figure 6-2: Comparison of maximum norms.

6.4 Comparison of Computational Efforts

For numerically-intensive programs such as finite element programs, the computational effort can be measured in terms of floating-point operations or FLOPS. A FLOP is multiplication or division between two real numbers [22]. In this finite element program, the most of the computational effort is devoted to solving the system stiffness equation. The most expensive computational operation for solving that equation is triangularization. Equation 6-5 calculates the FLOPS for triangularization, where ndof stands for the total number of degrees of freedom.

$$FLOPS = \frac{(ndof)^3}{6} + \frac{(ndof)^2}{2} - \frac{(2 \times ndof)}{3} \qquad \textbf{(6-5)}$$

Table 6-10 shows the comparison of the computational efforts. The equation of FLOPS for triangularization is a function of degrees of freedom. Since a T-spline model requires fewer ndof than its corresponding B-spline model, isogeometric analysis using T-splines requires less computational effort than using B-splines.

Table 6-10: Comparison of computational efforts.

	B5	**T5**		**B6**	**T6**
ndof	56	48		84	60
FLOPS	30,800	19,552		102,256	37,760

7 Conclusion

Isogeometric analysis using T-splines has been successfully implemented. A structural mechanics problem consisting of an infinite membrane with circular hole under constant tension was analyzed using B-splines and using T-splines.

The analysis results were obtained and compared with the closed-form solution of the membrane model at selected points of interest. Isogeometric analyses using both B-splines and T-splines converged to the closed-form solution as more control points were added, while T-splines employed substantially fewer control points than B-spline. Isogeometric analysis with T-splines involving fewer control points appears to less costly and less time consuming. T-splines achieved the same accuracy as B-splines with less computational effort in the FEA. In conclusion, we believe the isogeometric analysis using T-splines holds significant potential in computer-aided engineering and is a desirable alternative to the current isogeometric analysis using B-splines.

Although the strength of isogeometric analysis using T-spline has been presented in this thesis, it still needs to be tested on a wider range of analysis applications in the future.

References

[1] 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, 194:4135–4195, 2005.

[2] Y. Bazilevs, L. Beirao 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, 16:1031–1090, 2006.

[3] J. A. Cottrell, A. Reali, and T. J. R. Hughes, "Studies of refinement and continuity in isogeometric structural analysis," Computer Methods in Applied Mechanics and Engineering, 196:4160–4183, 2007.

[4] T. J. R. Hughes, A. Reali, G. Sangalli, "Efficient Quadrature for NURBS-based Isogeometric Analysis," Computer Methods in Applied Mechanics and Engineering, to appear, 2009.

[5] F. Auricchio, L. Beirão de Veiga, A. Buffa, C. Lovadina, A. Reali, G. Sangalli, "A fully "locking-free" isogeometric approach for plane linear elasticity problems: A stream function formulation," Computer Methods in Applied Mechanics and Engineering, Vol. 197, 2007, pp. 160–172.

[6] A. Reali, "An isogeometric analysis approach for the study of structural vibrations," Journal of Earthquake Engineering, 10, s.i. 1, 1–30, 2006.

[7] J. A. Cottrell, A. Reali, Y. Bazilevs, and T. J. R. Hughes, "Isogeometric analysis of structural vibrations," Computer Methods in Applied Mechanics and Engineering, 195:5257–5297, 2006.

[8] Y. Bazilevs, V. M. Calo, T. J. R. Hughes, and Y. Zhang, "Isogeometric fluid-structure interaction: Theory, algorithms and computations," Computational Mechanics, Vol. 43, 3-37, December, 2008.

[9] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J. R. Hughes, "Isogeometric fluid-structure interaction analysis with applications to arterial blood flow," Computational Mechanics, 38:310–322, 2006.

[10] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, and T. J. R. Hughes, "Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow," In Proceedings of the International Meshing Roundtable Conference, 2006.

[11] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "T-splines and T-NURCCS," ACM Transactions on Graphics 22, 3 (July), 477-484, 2003.

[12] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche, "T-spline simplification and local refinement," ACM Transactions on Graphics, 23 (3):276–283, 2004.

[13] W. C. Li, N. Ray, and B. Lévy, "Automatic and interactive mesh to T-spline conversion," in SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 191–200.

[14] Y. Wang, J. Zheng, and H. S. Seah, "Conversion between T-splines and hierarchical B-splines," in Computer Graphics and Imaging, 2005, pp. 8–13.

[15] Y. He, K. Wang, H. Wang, X. Gu, and H. Qin, "Manifold T-spline," in GMP, 2006, pp. 409–422.

[16] J. Zheng, Y. Wang, and H. S. Seah, "Adaptive T-spline surface fitting to z-map models," in GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. New York, NY, USA: ACM, 2005, pp. 405–411.

[17] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. 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, to appear, 2009.

[18] H. Yang, M. Fuchs, B. Jüttler, O. Scherzer, "Evolution of T-spline level sets with distance field constraints for geometry reconstruction and image segmentation," In:Proc. SMI'06, pp. 247–252 (2006).

[19] H, Yang, B, Jüttler, "3D shape metamorphosis based on T-spline level sets," The Visual Computer 23(12): 1015-1025 (2007).

[20] J. Deng, F. Chen and Y. Feng, "Dimensions of spline spaces over T-meshes," J. Comp. Appl. Math. 194 (2006) 267-283.

[21] T. W. Sederberg, “CS 557: Computer-Aided Geometric Design,” Class Notes, 2007.

[22] R. J. Balling, “CEEN 506: Continuum Mechanics and Finite Elements,” BYU Press, 2006.