ICES REPORT 10-45

November 2010

Isogeometric finite element data structures based on Bezier extraction of T-splines

by

M. A. Scott, M. J. Borden, C.V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes

Reference: M. A. Scott, M. J. Borden, C.V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes, "Isogeometric finite element data structures based on Bezier extraction of T-splines", ICES REPORT 10-45, The Institute for Computational Engineering and Sciences, The University of Texas at Austin, November 2010.

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2010; **00**:1–40 *Prepared using nmeauth.cls [Version: 2002/09/18 v2.02]*

Isogeometric finite element data structures based on Bézier extraction of T-splines

Michael A. Scott[1,*], Michael J. Borden[1,2], Clemens V. Verhoosel[3], Thomas W. Sederberg[4], and Thomas J.R. Hughes[1]

[1] *Institute for Computational Engineering and Sciences*
The University of Texas at Austin
1 University Station C0200, Austin, Texas 78712, USA
[2] *Sandia National Laboratories*
Albuquerque, NM 87185, USA
[3] *Department of Mechanical Engineering*
Einhoven University of Technology
5600 MB, Einhoven, The Netherlands
[4] *Department of Computer Science*
Brigham Young University
3361 TMCB PO Box 26576, Provo, Utah 84602, USA

SUMMARY

We develop finite element data structures for T-splines based on Bézier extraction generalizing our previous work for NURBS. As in traditional finite element analysis, the extracted Bézier elements are defined in terms of a fixed set of polynomial basis functions, the so-called Bernstein basis. The Bézier elements may be processed in the same way as in a standard finite element computer program, utilizing exactly the same data processing arrays. In fact, only the shape function subroutine needs to be modified, all other aspects of a finite element program remaining the same. A byproduct of the extraction process is the element extraction operator. This operator localizes the topological and global smoothness information to the element level, and represents a canonical treatment of T-junctions, referred to as "hanging nodes" in finite element analysis and a fundamental feature of T-splines. A detailed example is presented to illustrate the ideas. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: Bézier extraction, isogeometric analysis, T-splines, finite elements

1. Introduction

Isogeometric analysis was introduced in [1] and has been described in detail in [2]. In the isogeometric framework the basis which describes the geometry is also used as the basis for analysis. Investigations using simple tensor product NURBS constructions have shown that the use of a smooth basis in analysis provides computational advantages over standard finite elements in many areas including turbulence [3, 4, 5], fluid-structure interaction [6, 7, 8], incompressibility [9, 10, 11], structural

*Correspondence to: mscott@ices.utexas.edu analysis [12, 13], shells [14, 15, 16], phase-field analysis [17, 18], large deformation with mesh distortion [19], and shape optimization [20].

T-splines, which emanate from Computer Aided Geometric Design (CAGD), overcome the tensor product restriction inherent in NURBS [21]. In fact, NURBS form a restricted subset of T-splines. Additionally, T-splines can be locally refined [22] and can generate analysis-suitable models of arbitrary topological complexity [23]. This makes T-splines an ideal basis for isogeometric analysis. The extension of the isogeometric framework to the more advanced T-spline setting was initiated in [24, 25]. T-spline discretizations have been successfully applied to fracture and damage [26, 27]. In these applications, efficient local refinement plays an important role. Initial investigations extending T-spline-based isogeometric analysis to the arbitrary topology setting in the context of shells have also been promising [16].

In this paper we develop T-splines from the finite element point-of-view, utilizing Bézier extraction. Bézier extraction for T-splines was briefly mentioned in the context of geometric design in [21]. The application of Bézier extraction to isogeometric analysis for the special case of NURBS was detailed in [28]. The idea is to extract the linear operator which maps the Bernstein polynomial basis on Bézier elements to the global T-spline basis. This is referred to as ***Bézier extraction***. The linear transformation is defined by a matrix referred to as the extraction operator. The transpose of the extraction operator maps the control points of the global T-spline to the control points of the Bernstein polynomials. Figure 1 illustrates the idea for a B-spline curve. This provides a finite element representation of T-splines, and facilitates the incorporation of T-splines into existing finite element programs. Only the shape function subroutine needs to be modified. All other aspects of the finite element program remain the same. Additionally, Bézier extraction is automatic and can be applied to any T-spline regardless of topological complexity or polynomial degree. In particular, it represents an elegant treatment of T-junctions, referred to as "hanging nodes" in finite element analysis.

This paper is organized as follows. Basic T-spline concepts are reviewed in Section 2. The fundamental finite element structure underlying T-splines is developed in Section 3. Section 4 describes Bézier extraction of T-splines. T-splines and element extraction operators are then integrated into standard finite element programs in Section 5. This paper is written specifically in terms of bicubic T-splines, although the concepts apply to any degree. T-splines of arbitrary degree are discussed in [24, 29]. This paper is also focused on surfaces, although the ideas extend directly to solids.

2. T-spline fundamentals

We first present fundamental concepts underlying T-spline technology. We illustrate our developments using the physical domain $\Omega \subset \mathbb{R}^2$ shown in Figure 2. Throughout this paper we use p to indicate polynomial degree, d_p to indicate the number of parametric dimensions, and d_s to indicate the number of spatial dimensions.

2.1. The T-mesh

A T-spline is constructed from a T-mesh, denoted by T. For surfaces, i.e. $d_p = 2$, the T-mesh is a mesh of quadrilateral elements[†] which permit T-junctions. In finite element parlance, a T-junction is

[†]What we refer to as T-mesh "elements" are usually referred to as T-mesh "faces" in the CAGD literature [21].

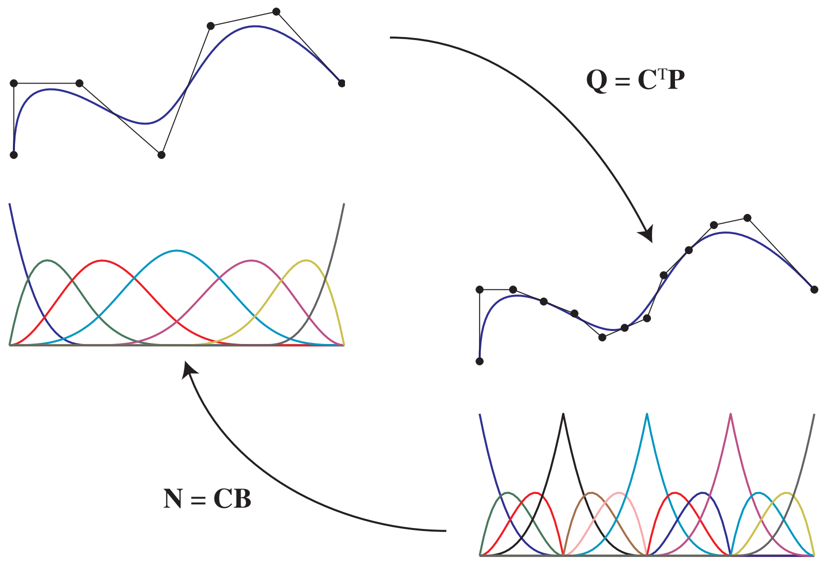

$\mathbf{Q} = \mathbf{C}^T\mathbf{P}$
$\mathbf{N} = \mathbf{C}\mathbf{B}$

Figure 1. Schematic representation of Bézier extraction for a B-spline curve. B-spline basis functions and control points are denoted by $\mathbf{N}$ and $\mathbf{P}$, respectively. Bernstein polynomials and control points are denoted by $\mathbf{B}$ and $\mathbf{Q}$, respectively. The curve $T(\boldsymbol{\xi}) = \mathbf{P}^T\mathbf{N}(\boldsymbol{\xi}) = \mathbf{Q}^T\mathbf{B}(\boldsymbol{\xi})$.

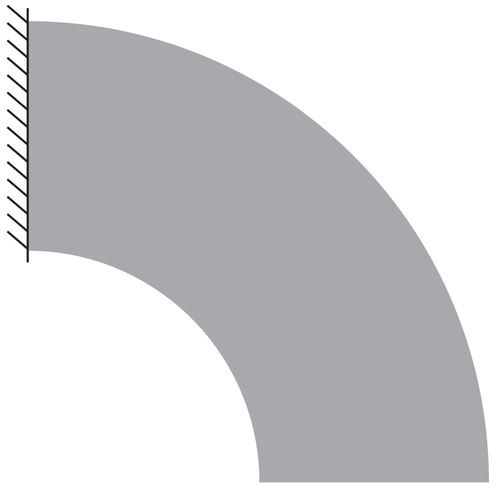


Figure 2. The domain $\Omega \subset \mathbb{R}^2$ for a bivariate ($d_p = 2$), cubic ($p = 3$) T-spline. The curved boundaries are exact quarter circles. The hash marks on the left indicate homogeneous Dirichlet boundary conditions.

analogous to a "hanging node." An element with T-junctions is composed of four corner vertices and any number of additional vertices on any side. The spatial representation of a T-mesh is called a T-

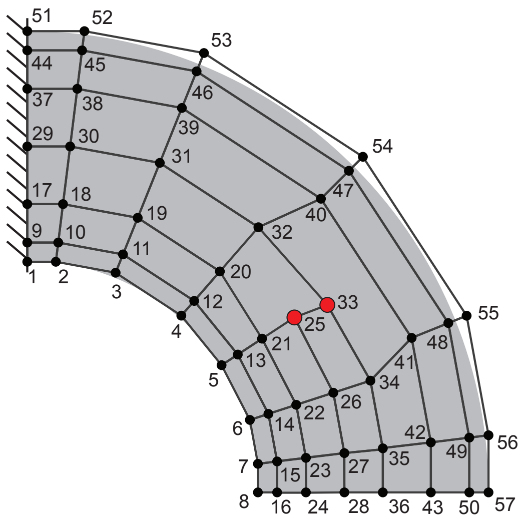


Figure 3. A T-mesh defining a bicubic T-spline geometry. The large red circles are the T-junctions for this T-mesh. The indexing identifies the T-mesh control points.

spline control mesh. In this paper we use T-mesh and T-spline control mesh interchangeably. Every vertex in the T-mesh is assigned a control point, $\mathbf{P}_A \in \mathbb{R}^{d_s}$, and control weight, $w_A \in \mathbb{R}$, where the index A is used to denote a global control point number.

A T-mesh for the domain Ω in Figure 2 is shown in Figure 3. The black and red circles are T-mesh vertices or, equivalently, control points (see Appendix I for values of the control points and weights). The T-junctions in Figure 3 are the red circles $\mathbf{P}_{25}$ and $\mathbf{P}_{33}$. This T-mesh will be used throughout the paper to illustrate the concept of Bézier extraction in finite element analysis. However, we note that this simple geometry could be represented more concisely with NURBS or T-splines. In the case of NURBS, as few as six control points are capable of representing the exact geometry, and for bicubic T-splines, as few as 16 are required. The additional control points in the T-mesh of Figure 3 is representative of the fact that finite element analysis will typically require many more degrees of freedom than geometric design.

A T-mesh does not contain enough information to define a T-spline basis. A valid knot interval configuration must also be assigned. Knot intervals [30] provide a way to assign local parametric information to a T-mesh. A knot interval is a non-negative real number assigned to an edge. A valid knot interval configuration requires that the knot intervals on opposite sides of every T-mesh element sum to the same value.

A valid knot interval configuration for the T-mesh in Figure 3 is shown in Figure 4. Notice that an outer ring of zero-length knot intervals has been assigned to the T-mesh. These zero-length knot intervals play a similar role to open knot vectors in NURBS and ease the imposition of boundary conditions.

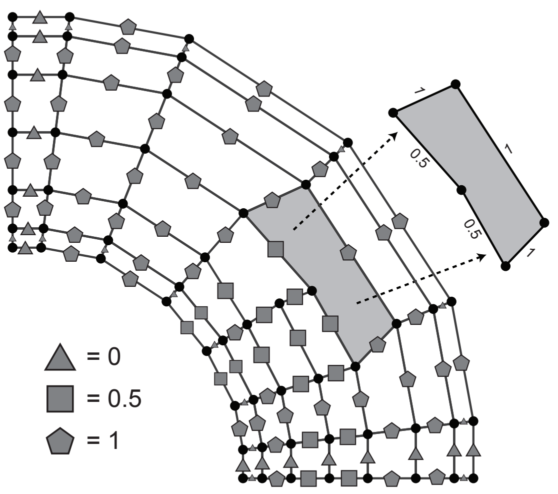


Figure 4. A valid knot interval configuration for the bicubic T-mesh in Figure 3. The triangles correspond to a knot interval of 0, the squares correspond to a knot interval of $\frac{1}{2}$, and the pentagons correspond to a knot interval of 1. A valid knot interval configuration for a T-mesh element is shown in the element callout. Notice that the knot intervals along opposing sides of the element sum to the same value.

2.2. *The T-spline basis* [‡]

T-spline basis functions can be inferred from a T-mesh with a valid knot interval configuration. A T-spline basis function is associated with each vertex in the T-mesh. We illustrate the construction of the T-spline basis functions associated with $\mathbf{P}_{18}$ and $\mathbf{P}_{33}$ in Figures 5 and 6, respectively.

2.2.1. Local knot interval vectors The first step in constructing a T-spline basis function is inferring sequences of knot intervals from the T-mesh in the neighborhood of the associated vertex. These knot intervals are organized into local knot interval vectors. A local knot interval vector is a sequence of knot intervals, $\Delta\Xi = \{\Delta\xi_1, \Delta\xi_2, \ldots, \Delta\xi_{p+1}\}$, such that $\Delta\xi_i = \xi_{i+1} - \xi_i$, and a local knot vector, derivable from any $\Delta\Xi$, is a non-decreasing knot sequence, $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{p+2}\}$. A local knot interval vector possesses all the information in a local knot vector except an origin. For example, if a knot interval vector is $\{1, 3, 2, 1\}$, then we can set the origin to zero and the corresponding local knot vector is $\{0, 1, 4, 6, 7\}$. If instead we are given a local knot vector, then the local knot interval vector is simply the difference between adjacent knots. In general, for T-splines, knot intervals are the method of choice for assigning and retrieving parameter information to and from the T-mesh since no origin is required. It should be noted that all classical B-spline algorithms can be rewritten in terms of knot intervals [32].

To every vertex, A, in the T-mesh we assign a set of local knot interval vectors, $\mathbf{\Delta\Xi}_A = \{\Delta\Xi_A^i\}_{i=1}^{d_p}$, from which a corresponding set of local knot vectors, $\mathbf{\Xi}_A = \{\Xi_A^i\}_{i=1}^{d_p}$, can be derived. The local knot

[‡]The mathematical term "basis" implies linear independence. The proof that T-spline blending functions in fact constitute a basis has recently been given for a class of T-splines [31].

interval vectors $\mathbf{\Delta\Xi}_A$ are constructed by marching through the T-mesh in each topological direction, starting at the the vertex A, until $p-1$ vertices or perpendicular edges are intersected. If a vertex or perpendicular edge is encountered, the knot interval distance traversed since the last intersection is placed in the local knot interval vector. If a T-mesh boundary is crossed before $p-1$ knot intervals are encountered, knot intervals are appended to complete the knot interval vector. In analysis these appended knot intervals are often chosen to be equal to zero to create an open knot vector structure along the boundary of the T-mesh.

In Figure 5, in the upper left corner, the knot intervals used to construct the local knot vectors for T-mesh vertex $\mathbf{P}_{18}$ are shown. Notice that this vertex is near the boundary of the T-mesh. When marching to the left only a single knot interval is encountered before reaching the boundary of the T-mesh. As a result an additional zero knot interval is added to the front of the local knot interval vector. The knot interval vectors for $\mathbf{P}_{18}$ are given by

$$\mathbf{\Delta\Xi}_{18} = \begin{bmatrix} 0,0,1,1 \\ 0,1,1,1 \end{bmatrix}.$$

In Figure 6, in the upper left corner, the knot intervals used to construct the local knot interval vectors for T-mesh vertex $\mathbf{P}_{33}$ are shown. Notice that this is a T-junction vertex. In this case, T-mesh elements must be traversed to form the local knot interval vectors. When an element is traversed the knot interval sum associated with the sides of the element which are parallel to the traversal direction are inserted into the local knot interval vector. For $\mathbf{P}_{33}$, the knot interval vectors are given by,

$$\mathbf{\Delta\Xi}_{33} = \begin{bmatrix} 1,\frac{1}{2},\frac{1}{2},1 \\ \frac{1}{2},\frac{1}{2},1,1 \end{bmatrix}.$$

2.2.2. The local basis function domain Each set of local knot vectors $\mathbf{\Xi}_A$ defines a local basis function domain, $\widehat{\Omega}_A \subset \mathbb{R}^{d_p}$, over which a single T-spline basis function is defined. The local basis function domain is defined as

$$\widehat{\Omega}_A = \bigotimes_{i=1}^{d_p} \widehat{\Omega}^i_A, \tag{1}$$

where $\widehat{\Omega}^i_A = [0, \Delta\xi^i_1 + \Delta\xi^i_2 + \Delta\xi^i_3 + \Delta\xi^i_4] \subset \mathbb{R}$. Each local basis function domain carries a coordinate system $\boldsymbol{\xi}_A = (\xi^1_A, \xi^2_A) = (\xi_A, \eta_A)$. This coordinate system is called the basis coordinate system.

Local basis function domains $\widehat{\Omega}_{18}$ and $\widehat{\Omega}_{33}$ are shown in the bottom right corners of Figures 5 and 6, respectively. In the case of $\widehat{\Omega}_{18}$ we have that $\widehat{\Omega}^1_{18} = [0,2]$ and $\widehat{\Omega}^2_{18} = [0,3]$. In the case of $\widehat{\Omega}_{33}$ we have that $\widehat{\Omega}^1_{33} = [0,3]$ and $\widehat{\Omega}^2_{33} = [0,3]$.

2.2.3. T-spline basis functions Over each local basis function domain $\widehat{\Omega}_A$ we define a single T-spline basis function, $N_A : \widehat{\Omega}_A \to \mathbb{R}^+ \cup 0$. This is done by forming the tensor product of the univariate basis functions $\{N^i_A(\xi^i_A \,|\, \Xi^i_A)\}_{i=1}^{d_p}$ as

$$N_A\left(\boldsymbol{\xi}_A \,|\, \mathbf{\Xi}_A\right) \equiv \prod_{i=1}^{d_p} N^i_A\left(\xi^i_A \,|\, \Xi^i_A\right). \tag{2}$$

The univariate T-spline basis function, $N_A^i : \widehat{\Omega}_A^i \to \mathbb{R}^+ \cup 0$, is defined using a recurrence relation, starting with the piecewise constant ($p = 0$) basis function

$$N_A^i \left(\xi_A^i \,|\, \xi_{A,1}^i, \xi_{A,2}^i\right) = \begin{cases} 1 & \text{if } \xi_{A,1}^i \leq \xi_A^i < \xi_{A,2}^i \\ 0 & \text{otherwise} \end{cases} , \tag{3}$$

where $\xi_{A,k}^i$ is the k^{th} knot value in the localized knot vector Ξ_A^i. For $p > 0$, the basis function is defined using the Cox-de Boor recursion formula:

$$\begin{aligned} N_A^i \left(\xi_A^i \,|\, \xi_{A,1}^i, \xi_{A,2}^i, \ldots, \xi_{A,p+2}^i\right) = {} & \frac{\xi_A^i - \xi_{A,1}^i}{\xi_{A,p+1}^i - \xi_{A,1}^i} N_A^i \left(\xi_A^i \,|\, \xi_{A,1}^i, \xi_{A,2}^i, \ldots, \xi_{A,p+1}^i\right) + \\ & + \frac{\xi_{A,p+2}^i - \xi_A^i}{\xi_{A,p+2}^i - \xi_{A,2}^i} N_A^i \left(\xi_A^i \,|\, \xi_{A,2}^i, \xi_{A,3}^i, \ldots, \xi_{A,p+2}^i\right). \end{aligned} \tag{4}$$

The T-spline basis functions N_{18} and N_{33} are shown in the lower left corner of Figures 5 and 6, respectively. It should be noted that while algorithms exist for computing T-spline basis functions according to (4) (see [33]), using the recursive definition in finite element shape function routines is expensive when compared to the extracted element technology introduced in Section 4 of this paper. Between knots, the one-dimensional basis functions (4) are C^∞ continuous. At knots the continuity is reduced (see [2]).

3. The T-spline element structure

We now develop the finite element structure underlying T-splines. A T-spline element $\Omega^e \subset \mathbb{R}^{d_s}$ is a region in physical space which is bounded by knot lines, which are lines of reduced continuity in the T-spline basis. We use the terminologies knot lines and lines of reduced continuity interchangeably throughout this paper. The basis functions restricted to the interior of the T-spline element are C^∞.

3.1. The local basis function mesh

From any set of local knot interval vectors $\boldsymbol{\Delta\Xi}_A$ (and corresponding local knot vectors Ξ_A) a local basis function mesh, T_A, can be defined as the tensor product mesh representation of the local knot vectors

$$\mathsf{T}_A = \bigotimes_{i=1}^{d_p} \Xi_A^i. \tag{5}$$

Each edge in T_A continues to carry the appropriate knot interval from $\boldsymbol{\Delta\Xi}_A$.

Figure 5, in the upper right corner, shows the local basis function mesh, T_{18}, generated from $\boldsymbol{\Delta\Xi}_{18}$. The shaded region indicates the local basis function mesh elements with positive parametric area. Figure 6, in the upper right corner, shows the local basis function mesh, T_{33}, generated from $\boldsymbol{\Delta\Xi}_{33}$. In this case every element has positive parametric area.

3.2. The elemental T-mesh

A T-mesh element does not necessarily have a one-to-one correspondence with a T-spline element. We recall that a T-mesh element is a quadrilateral in the T-mesh or T-spline control mesh and the T-spline element is a region of the T-spline surface bounded by lines of reduced continuity in the T-spline basis.

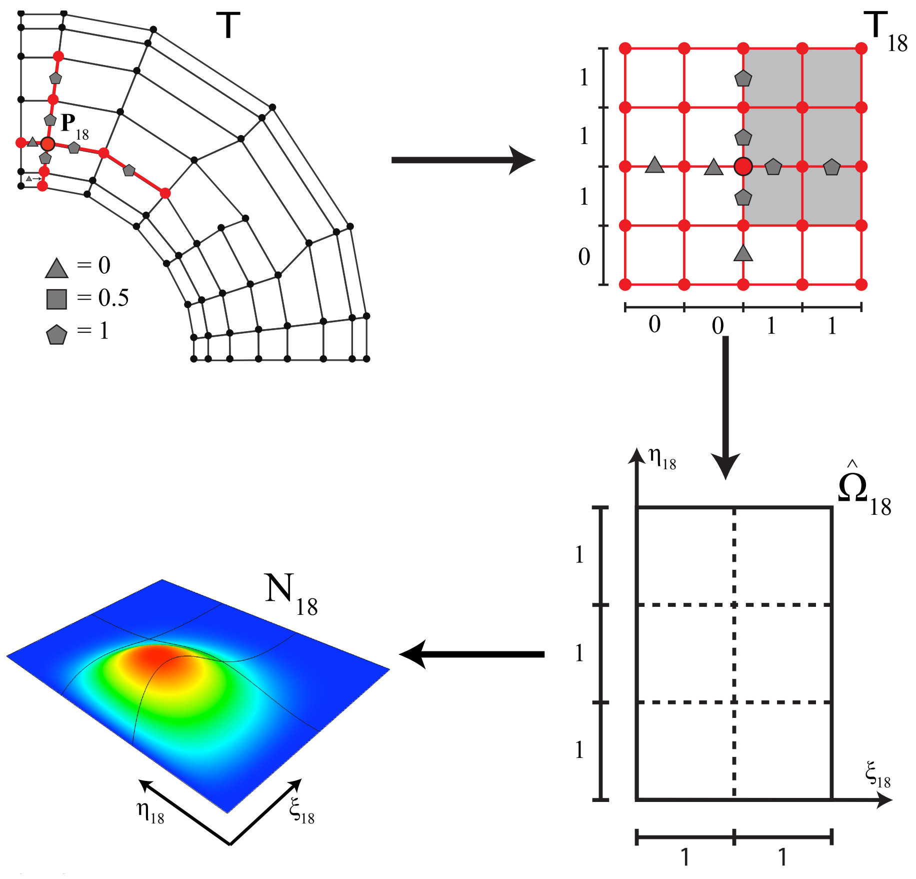


Figure 5. The construction of T-spline basis function N_{18} which is associated with T-mesh vertex $\mathbf{P}_{18}$. Starting at the upper left and going clockwise we have: Inferrence of the local knot vectors from the T-mesh, the resulting local basis function mesh, the local basis function domain, and the T-spline basis function.

This can be seen by drawing the local basis function mesh T_{33} on top of the T-mesh, as in Figure 7. The dashed lines in Figure 7 indicate edges which exist in T_{33} but not in T. Each knot line in the T-spline basis is present in at least one basis function. However, not all these knot lines are represented by corresponding lines in the T-mesh. We form the ***elemental T-mesh*** by augmenting the T-mesh with all these lines. The elemental T-mesh T_f of T is shown in Figure 8. The dashed edges indicate lines of reduced continuity which were not present in the original T-mesh.

3.3. The IEN *array*

Over every element domain there exists a set of T-spline basis functions which are non-zero. The T-spline basis functions are in one-to-one correspondence with the T-mesh control points and are indexed by the global control point numbers. The IEN array maps the local basis function number, a, and the element number, e, to the corresponding global control point number A. There can be different numbers

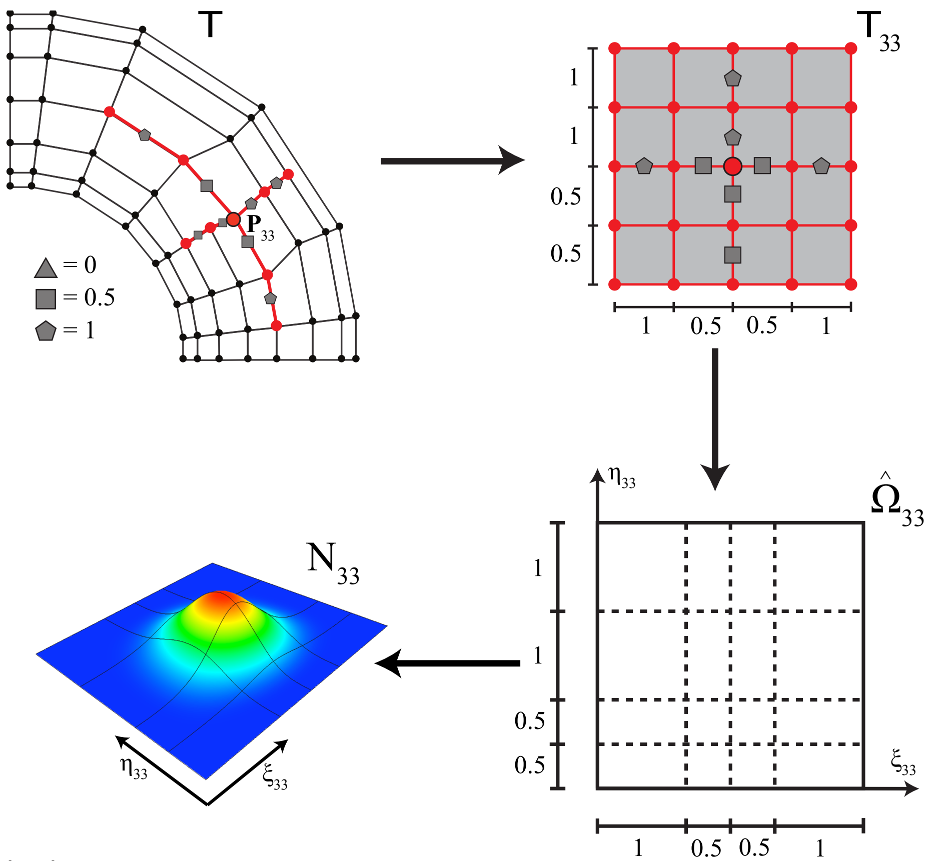


Figure 6. The construction of T-spline basis function N_{33} which is associated with the T-junction T-mesh vertex $\mathbf{P}_{33}$. Starting at the upper left and going clockwise we have: Inferrence of the local knot vectors from the T-mesh, the resulting local basis function mesh, the local basis function domain, and the T-spline basis function.

of T-spline basis functions supported by each element. This is in contrast to NURBS where all elements are in the support of exactly $(p+1)^{d_p}$ NURBS basis functions. The complete IEN array for the T-spline elements in Figure 8 is shown in Table I. Notice that elements 9 and 10 are in the supports of 17 T-spline basis functions. Figure 9 shows T-spline elements 1, 10, 11, and 17 and the T-mesh control points whose corresponding basis functions are non-zero over each of these elements.

3.4. Restricting the global basis to the parent element domain

We develop the finite element point-of-view for T-splines by defining the T-spline basis functions over the parent element domain. The non-local and T-junction structure of the T-spline basis requires additional machinery not found in traditional finite element constructions. We define a set, $\mathbf{S}^e = \{\tilde{\Phi}^e, \{\hat{\Phi}^e_a\}_{a=1}^{n_e}\}$, of affine mappings for the element under consideration where

- $\tilde{\Phi}^e : \tilde{\Omega} \to \widehat{\Omega}^e$ is a one-to-one and onto affine map from the parent element domain onto the

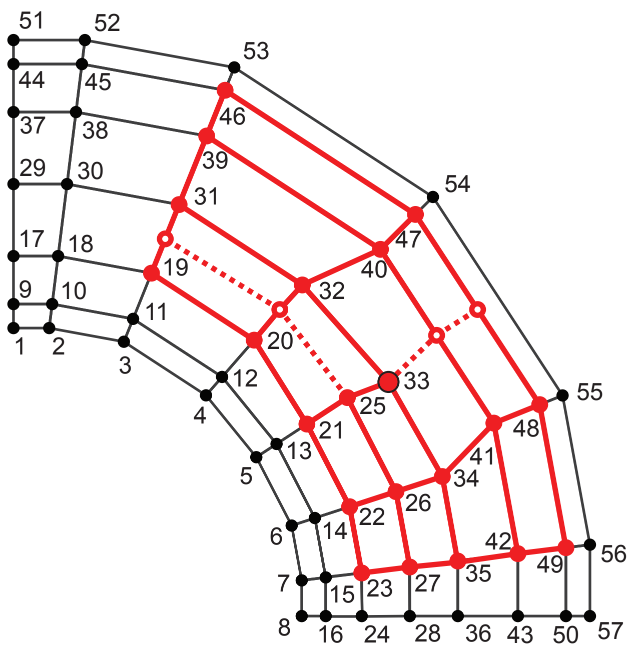


Figure 7. The mapping of T_{33} onto the global T-mesh T. The dashed edges correspond to lines of reduced continuity from T_{33} which are not in T.

element domain:

$$\boldsymbol{\xi}^e = \tilde{\Phi}^e(\tilde{\boldsymbol{\xi}}). \tag{6}$$

- $\hat{\Phi}^e_a : \widehat{\Omega}^e \to \widehat{\Omega}_A$, for $a = 1, \ldots, n_e$ is a one-to-one affine mapping from the element domain into the local basis function domain for basis function $A = \mathrm{IEN}(a, e)$:

$$\boldsymbol{\xi}_A = \hat{\Phi}^e_a(\boldsymbol{\xi}^e). \tag{7}$$

These mappings can be determined using the elemental T-mesh and the IEN array and are used to map from the parent element into each local basis function domain which corresponds to a T-spline basis function which is non-zero over element e. The action of some but not all of the affine maps in $\mathbf{S}^{17}$ is shown in Figure 10.

Using $\mathbf{S}^e$ and the IEN array we can define a localized, element-based definition for the global T-spline basis functions as

$$N_A(\boldsymbol{\xi}_A)\big|_e = N_A\left(\hat{\Phi}^e_a(\boldsymbol{\xi}^e)\right)\Big|_e = N_A\left(\hat{\Phi}^e_a\left(\tilde{\Phi}^e(\tilde{\boldsymbol{\xi}})\right)\right)\Big|_e = N^e_a(\tilde{\boldsymbol{\xi}}) \tag{8}$$

where $|_e$ indicates restriction to the domain of element number e.

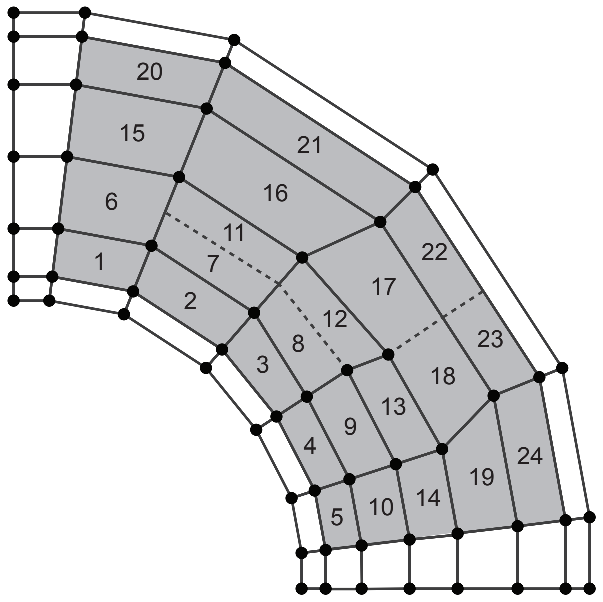


Figure 8. The T-spline elements in the elemental T-mesh T_f of T.

3.5. *The T-spline element geometric map*

With the basis functions defined over the parent element by equation (8), we now define the element geometric map, $\tilde{\mathbf{x}}^e : \tilde{\Omega} \to \Omega^e$, from the parent element domain onto the physical domain as

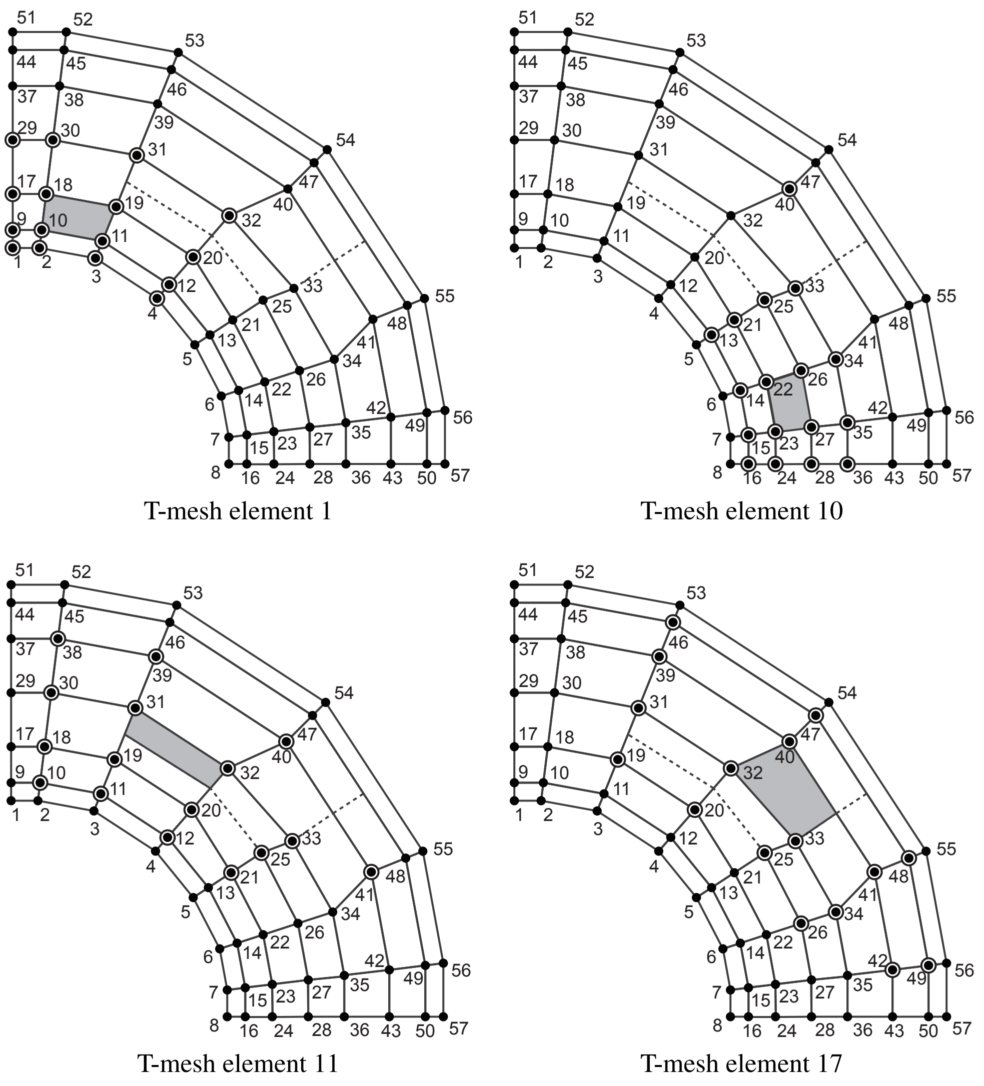
$$\tilde{\mathbf{x}}^e(\tilde{\boldsymbol{\xi}}) = \frac{\sum_{a=1}^{n_e} \mathbf{P}_a^e w_a^e N_a^e(\tilde{\boldsymbol{\xi}})}{W^e(\tilde{\boldsymbol{\xi}})} = \sum_{a=1}^{n_e} \mathbf{P}_a^e R_a^e(\tilde{\boldsymbol{\xi}}) \tag{9}$$

where

$$W^e(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_e} w_a^e N_a^e(\tilde{\boldsymbol{\xi}}) \tag{10}$$

is the element weight function, and $\mathbf{P}_a^e = \mathbf{P}_{\mathrm{IEN(a,e)}}$ and $w_a^e = w_{\mathrm{IEN(a,e)}}$ are the control point and weight, respectively, corresponding to the a^{th} T-spline basis function over element e. R_a^e is the rational form of the T-spline basis function because it includes the weight function in its denominator. Defining the element weight vector $\mathbf{w}^e = \{w_a^e\}_{a=1}^{n_e}$, the diagonal weight matrix $\mathbf{W}^e = \mathrm{diag}(\mathbf{w}^e)$, and element control points $\mathbf{P}^e$ as a matrix of dimension $n_e \times d_s$,

$$\mathbf{P}^e = \begin{bmatrix} P_1^{e,1} & P_1^{e,2} & \dots & P_1^{e,d_s} \\ P_2^{e,1} & P_2^{e,2} & \dots & P_2^{e,d_s} \\ \vdots & \vdots & & \vdots \\ P_{n_e}^{e,1} & P_{n_e}^{e,2} & \dots & P_{n_e}^{e,d_s} \end{bmatrix}, \tag{11}$$



◉ Indicates T-spline basis functions that are supported by the highlighted element

Figure 9. T-mesh elements 1, 10, 11, and 17 in the elemental T-mesh and the T-mesh control points whose corresponding T-spline basis functions are non-zero over these T-spline elements.

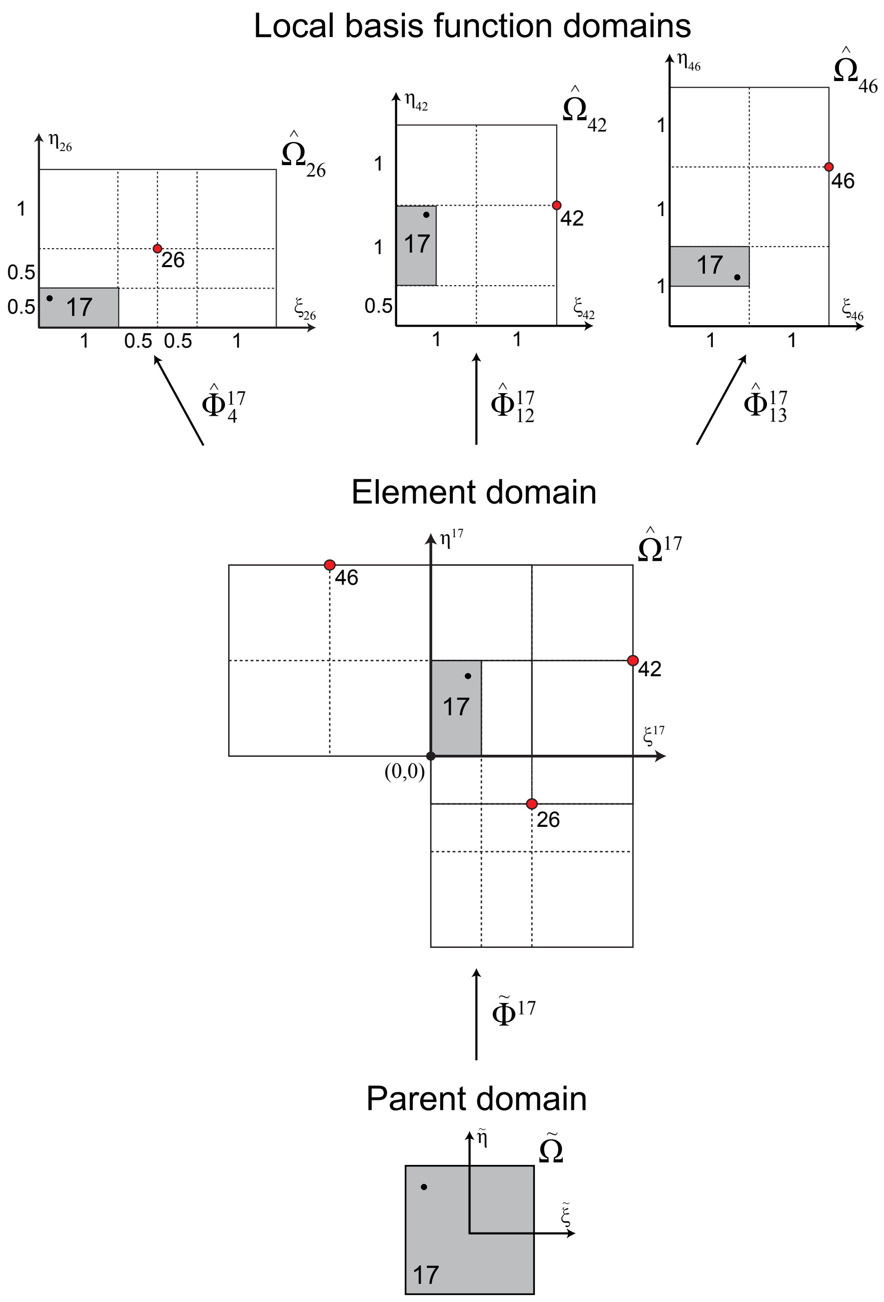


Figure 10. The mappings between parent, element, and basis function coordinate systems for element 17. For simplicity only some of the mappings are shown. $\tilde{\Phi}^{17}$ maps from the parent coordinate system into the element coordinate system. $\hat{\Phi}^{17}_{a}$ maps from the element coordinate system of local basis function number a into the basis function coordinate system of global control point $A = \text{IEN}(a, 17)$. The solid dot (•) on element 17 shows the rotations incorporated in the mappings.

	Element function number (a)																
e	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	9	10	11	12	17	18	19	20	29	30	31	32	
2	2	3	4	5	10	11	12	13	18	19	20	21	25	30	31	32	
3	3	4	5	6	11	12	13	14	19	20	21	22	25	26	31	32	
4	4	5	6	7	12	13	14	15	20	21	22	23	25	26	27	32	
5	5	6	7	8	13	14	15	16	21	22	23	24	25	26	27	28	
6	9	10	11	12	17	18	19	20	29	30	31	32	37	38	39	40	
7	10	11	12	13	18	19	20	21	25	30	31	32	33	38	39	40	
8	11	12	13	14	19	20	21	22	25	26	31	32	33	34	39	40	
9	12	13	14	15	20	21	22	23	25	26	27	32	33	34	35	39	40
10	13	14	15	16	21	22	23	24	25	26	27	28	33	34	35	36	40
11	10	11	12	18	19	20	21	25	30	31	32	33	38	39	40	41	
12	11	12	19	20	21	22	25	26	31	32	33	34	39	40	41	42	
13	12	20	21	22	23	25	26	27	32	33	34	35	39	40	41	42	
14	21	22	23	24	25	26	27	28	33	34	35	36	40	41	42	43	
15	17	18	19	20	29	30	31	32	37	38	39	40	44	45	46	47	
16	18	19	20	25	30	31	32	33	38	39	40	41	45	46	47	48	
17	19	20	25	26	31	32	33	34	39	40	41	42	46	47	48	49	
18	20	25	26	27	32	33	34	35	39	40	41	42	46	47	48	49	
19	25	26	27	28	33	34	35	36	40	41	42	43	47	48	49	50	
20	29	30	31	32	37	38	39	40	44	45	46	47	51	52	53	54	
21	30	31	32	33	38	39	40	41	45	46	47	48	52	53	54	55	
22	31	32	33	34	39	40	41	42	46	47	48	49	53	54	55	56	
23	32	33	34	35	39	40	41	42	46	47	48	49	53	54	55	56	
24	33	34	35	36	40	41	42	43	47	48	49	50	54	55	56	57	

$$A = \text{IEN}(a, e)$$

Table I. The IEN array is constructed using the information from Figure 9. The IEN array maps the local basis function number (a) and the element number (e) to the corresponding global control point (A). The local basis function number indexes the T-spline basis functions supported by the element in question. Note that there can be different numbers of T-spline basis functions associated with different elements.

we can generate the corresponding matrix representation of the geometric map as

$$\tilde{\mathbf{x}}^e(\tilde{\boldsymbol{\xi}}) = \frac{1}{(\mathbf{w}^e)^T \mathbf{N}^e(\tilde{\boldsymbol{\xi}})} (\mathbf{P}^e)^T \mathbf{W}^e \mathbf{N}^e(\tilde{\boldsymbol{\xi}}) \tag{12}$$

$$= (\mathbf{P}^e)^T \mathbf{R}^e(\tilde{\boldsymbol{\xi}}), \tag{13}$$

where $\mathbf{N}^e = \{N_a^e\}_{a=1}^{n_e}$ and $\mathbf{R}^e = \{R_a^e\}_{a=1}^{n_e}$ are the vectors of polynomial and rational T-spline basis functions, respectively, which are non-zero over element e.

4. Bézier Extraction for T-splines

The T-spline element construction presented in Section 3 can be further simplified using Bézier extraction. For each element, Bézier extraction generates an element extraction operator. This operator is a simple, compact, algebraic representation of the topological and smoothness information stored in the elemental T-mesh and T-spline basis. This operator can be easily integrated into the finite element framework, analogous to what was presented for NURBS in [28]. This should not be a surprise since NURBS form a restricted subset of T-splines. In fact, a finite element code capable of handling extraction operators can easily incorporate both NURBS and T-splines.

4.1. *The element extraction operator and the Bézier element*

Bézier extraction for T-splines determines the exact representation of the T-spline basis over each T-spline element e in terms of a set of Bernstein polynomials, $\mathbf{B}(\tilde{\boldsymbol{\xi}})$. Every localized T-spline basis function, $N_a^e(\tilde{\boldsymbol{\xi}})$, can be written as a linear combination of these Bernstein polynomials. In other words, for each localized T-spline basis function, $N_a^e(\tilde{\boldsymbol{\xi}})$, there exists coefficients, $c_{a,b}^e$ such that [§]

$$N_a^e(\tilde{\boldsymbol{\xi}}) = \sum_{b=1}^{(p+1)^{d_p}} c_{a,b}^e B_b(\tilde{\boldsymbol{\xi}}) \tag{14}$$

over element e. In matrix-vector form (14) is written as

$$\mathbf{N}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{C}^e\mathbf{B}(\tilde{\boldsymbol{\xi}}), \tag{15}$$

where $\mathbf{C}^e$ is the element extraction operator. We call the element defined by the Bernstein polynomials the Bézier element.

Note that, in contrast with the T-spline basis functions $\mathbf{N}^e$ and $\mathbf{R}^e$, there are the same number of Bernstein basis functions for all the elements. Also, the use of the operator allows us to standardize the form of the element basis on the parent domain. In other words, each Bézier element is defined in terms of the exact same set of Bernstein basis functions. This may be contrasted with the T-spline basis defined over each T-spline element in which the structure of the basis changes from element to element.

4.2. *The Bernstein basis*

The Bernstein polynomials form a basis for the Bézier element. The univariate Bernstein basis functions are defined over the biunit interval $[-1, 1]$ as

$$B_{i,p}^k(\tilde{\xi}^k) = \frac{1}{2^p}\binom{p}{i-1}(1-\tilde{\xi}^k)^{p-(i-1)}(1+\tilde{\xi}^k)^{i-1} \tag{16}$$

where the binomial coefficient $\binom{p}{i-1} = \frac{p!}{(i-1)!(p+1-i)!}$, $1 \le i \le p+1$. In CAGD, The Bernstein polynomials are usually defined over the unit interval $[0, 1]$, but in finite element analysis the biunit interval is preferred to take advantage of the usual domains for Gauss quadrature. The univariate

[§]Note that the element basis functions are numbered $b = 1, 2, \ldots, (p+1)^{d_p}$. This convention is typical in finite element analysis, but different than that used in CAGD in which it is standard for the indexing to begin with 0.

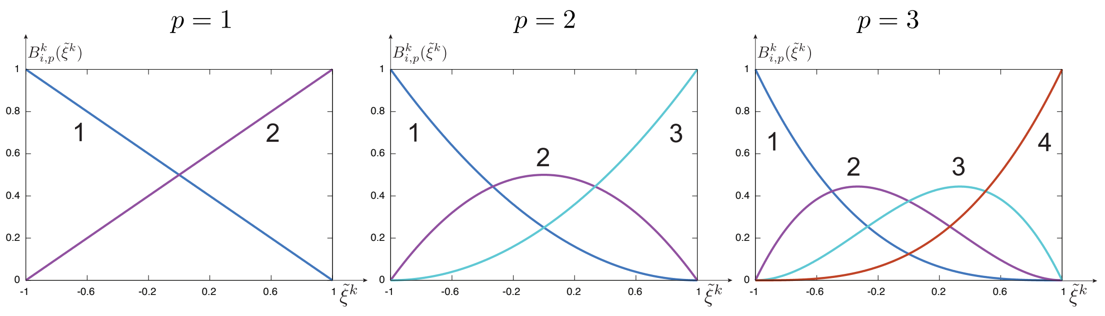


Figure 11. Bernstein basis functions for polynomial degree $p = 1, 2, 3$.

Bernstein basis functions for $p = 1, 2$, and 3 are plotted in Figure 11. The univariate Bernstein basis has the following properties:

- Partition of unity.

$$\sum_{i=1}^{p+1} B^k_{i,p}(\tilde{\xi}^k) = 1 \quad \forall \tilde{\xi}^k \in [-1,1] \tag{17}$$

- Pointwise nonnegativity.

$$B^k_{i,p}(\tilde{\xi}^k) \geq 0 \quad \forall \tilde{\xi}^k \in [-1,1] \tag{18}$$

- Endpoint interpolation.

$$B^k_{1,p}(-1) = B^k_{p+1,p}(1) = 1 \tag{19}$$

- Symmetry.

$$B^k_{i,p}(\tilde{\xi}^k) = B^k_{p+1-i,p}(-\tilde{\xi}^k) \quad \forall \tilde{\xi}^k \in [-1,1] \tag{20}$$

The multivariate Bernstein basis functions of degree p, $B_{a,p} : \tilde{\Omega} \to \mathbb{R}^+ \cup 0$, with $a = 1, \dots, (p+1)^{d_p}$, are formed as the tensor-product of univariate basis functions $B^k_{i,p} : [-1,1] \to \mathbb{R}^+ \cup 0$, with $i = 1, \dots, p+1$. In the bivariate case we have

$$B_{a(i,j),p}(\tilde{\boldsymbol{\xi}}) = B^1_{i,p}(\tilde{\xi}^1) B^2_{j,p}(\tilde{\xi}^2). \tag{21}$$

with

$$a\,(i,j) = (p+1)(j-1) + i. \tag{22}$$

Recall that $\tilde{\boldsymbol{\xi}} = (\tilde{\xi}^1, \tilde{\xi}^2) = (\tilde{\xi}, \tilde{\eta})$ is the coordinate system assigned to the parent element. From the formulas, it is clear that the basis functions are numbered from left to right in one dimension. In two dimensions each row is numbered from left to right, starting with the bottom row and moving upward. See Figure 12.

4.3. *Computing the element extraction operator for T-splines*

For T-splines, no global tensor product structure exists but, as was discussed in Section 2.2.2, a local tensor product parameter domain can be defined for each individual basis function. Thus, for T-splines, the computation of the element extraction operators is performed function-by-function, where

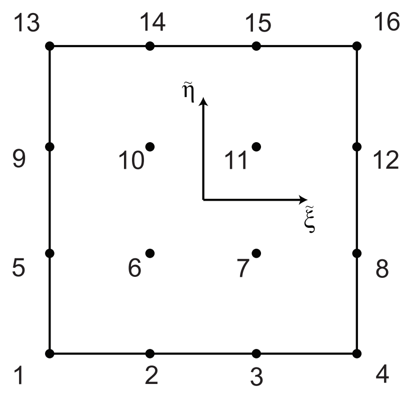


Figure 12. Ordering of the two-dimensional Bernstein polynomials on a bicubic Bézier element.

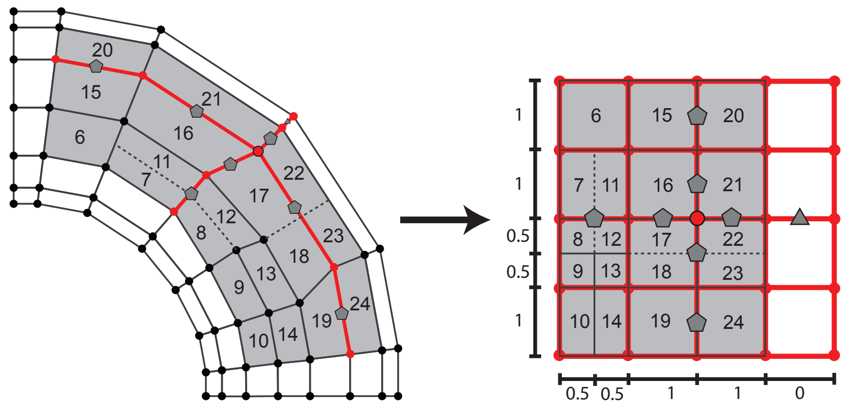


Figure 13. All Bézier elements in the support of T-spline basis function N_{40} in the elemental T-mesh, T_f (left) and the local basis function mesh, T_{40} (right).

each basis function contributes a row to each of the extraction operators corresponding to the Bézier elements in its support. For example, Figure 13 shows the Bézier elements in the support of T-spline basis function N_{40}. Bézier extraction of N_{40} contributes a row to the element extraction operator for each of the elements in its support.

4.3.1. Comparing extraction for NURBS and T-splines Similarly to NURBS, multivariate extraction operators for T-splines can be computed as products of univariate extraction operators. A major difference between NURBS and T-splines is that where NURBS have a global parameterization, T-splines have a local parameterization. This leads to two differences in how the extraction operators are computed. First, with T-splines, we work with local knot vectors that are, in general, not open, i.e., the first and last knot may have multiplicity less than $p + 1$. Second, since the local knot vectors correspond to individual functions, as a local knot vector is processed, we compute a single row for each of the corresponding element extraction operators, as opposed to computing the entire element extraction operator at once. Additionally, since there may be T-junctions in a T-spline, knots may need to be inserted into the knot spans of the local knot vector as part of the extraction process.

4.3.2. The extended knot vector To handle the local knot vectors we introduce the extended knot vector, $\overline{\Xi}_A$. The extended knot vector is constructed by repeating the first and last knots in the local knot vector until they have a multiplicity equal to $p+1$. The extended knot vector does not change the parametric description of the original T-spline basis function, but it adds functions to the local basis function domain. For example, Figure 14 shows the univariate T-spline basis function (solid line) with local knot vector $\Xi = \{0, 1, 2, 3, 4\}$ and the additional basis functions (dashed lines) added by the extended knot vector $\overline{\Xi} = \{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}$. When the functions are numbered from left to right the T-spline basis function will be numbered $n_t + 1$ where n_t is the number of knots added to the front of the local knot vector when constructing the extended knot vector.

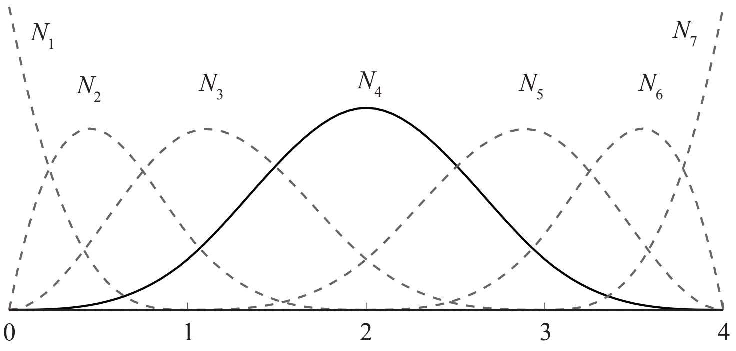


Figure 14. A univariate T-spline basis function (solid line) with local knot vector $\Xi = \{0, 1, 2, 3, 4\}$. The extended knot vector $\overline{\Xi} = \{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}$ adds the additional functions (dashed lines) that will be used to construct the basis functions for the Bézier elements.

4.3.3. Univariate extraction We now describe how the univariate extraction operators are computed for T-splines. Once the extended knot vector has been constructed for the original T-spline basis function, the corresponding rows of the univariate extraction operators can be computed. These extraction operator rows are computed by performing knot insertion on the extended knot vector. Conceptually, the process is similar to computing the extraction operator for NURBS [28]. For example, if the extended knot vector for the function in Figure 14 is considered as a knot vector for a NURBS curve, Bézier extraction would produce the basis functions shown in Figure 15, which are the same basis functions that are produced by extraction for the T-spline basis function. The difference between extraction for NURBS and T-splines is that for T-splines we only compute a single row of the operator. For comparison, Table II shows the full extraction operators that would be computed in the case of NURBS with the rows corresponding to the T-spline basis function from Figure 14 highlighted. The extraction operator rows are computed by Algorithm 1, which is a modified version of the algorithm presented for NURBS in [28].

As a result of the extraction process we have a representation of each T-spline function in terms of

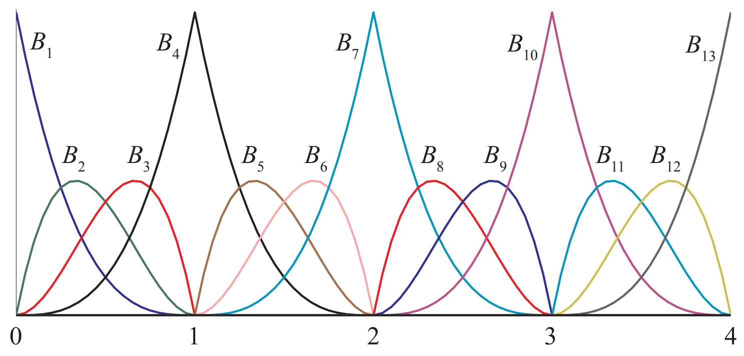


Figure 15. The resulting basis functions of the Bézier elements after knot insertion and refinement of the basis functions shown in Figure 14.

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{Bmatrix}, \quad \begin{Bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{Bmatrix} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 7/12 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{Bmatrix},$$

$$\begin{Bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \begin{Bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{Bmatrix}, \quad \begin{Bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix}$$

Table II. The action of extraction operators on the Bézier element basis computed from the extended knot vector from Figure 14. The full operators would be computed in the case of NURBS, but only the highlighted rows for the T-spline basis function.

the basis functions of the Bézier elements. This can be seen by considering Bézier element e in the support of the T-spline basis function A. The T-spline basis function A restricted to element e is related to the Bernstein basis functions by

$$N_A(\boldsymbol{\xi}_A)|_e = N_a^e(\tilde{\boldsymbol{\xi}}) = \mathbf{e}_a^T \mathbf{N}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{e}_a^T \mathbf{C}^e \mathbf{B}(\tilde{\boldsymbol{\xi}}) = \mathbf{c}_a^{e\,T} \mathbf{B}(\tilde{\boldsymbol{\xi}}) \tag{23}$$

where $\mathbf{e}_a$ is a unit vector which is equal to 1 in entry a and zero elsewhere. The vector $\mathbf{c}_a^e$ extracts basis function $A = \text{IEN}(a, e)$ from Bézier element e.

4.3.4. Multivariate extraction Multivariate Bézier extraction operators $\mathbf{C}^e$ for a T-spline element e are constructed as products of univariate Bézier extraction operators. In the bivariate case, T-spline basis function N_A restricted to element e can be decomposed into two univariate T-spline basis functions as

$$N_A(\boldsymbol{\xi}_A)|_e = N_a^e(\tilde{\boldsymbol{\xi}}) = N_a^{e,1}(\tilde{\xi}^1) N_a^{e,2}(\tilde{\xi}^2), \tag{24}$$

where a and e are such that $A = \text{IEN}(a, e)$ and the superscripts indicate the local parametric direction. Using the extraction procedure outlined above for univariate T-spline basis functions, we can construct extraction operators for the basis functions $N_a^{e,1}(\tilde{\xi}^1)$ and $N_a^{e,2}(\tilde{\xi}^2)$. By substituting equation (23) into (24) we obtain

$$N_a^e(\tilde{\boldsymbol{\xi}}) = \left[\mathbf{c}_a^{e,1\,T}\,\mathbf{B}^1(\tilde{\xi}^1)\right]\left[\mathbf{c}_a^{e,2\,T}\,\mathbf{B}^2(\tilde{\xi}^2)\right]. \tag{25}$$

Using the index mapping from equation (22) it can be shown that this becomes

$$N_a^e(\tilde{\boldsymbol{\xi}}) = \mathbf{c}_a^{e\,T}\,\mathbf{B}(\tilde{\boldsymbol{\xi}}), \tag{26}$$

where now $\mathbf{c}_a^e$ corresponds to the a^{th} row of the bivariate element extraction operator $\mathbf{C}^e$ in equation (15). This process is repeated for each T-spline basis function supported by element e to generate the full extraction operator for the element. Importantly, once the element extraction operators are computed, a finite element code never needs to know anything about T-mesh topology to process the global T-spline basis.

The process and result of extracting $N_{40}(\boldsymbol{\xi}_{40})$ over an element e in its support are shown in Figures 16 and 17, respectively. In Figure 16 the univariate components of the T-spline basis function are shown on the left and right. Note that in both cases a knot interior to a knot span of the extended knot vector must be inserted p times to compute the extraction operator for element e. In Figure 17 we show the resulting extraction coefficients. In Appendix III, full extraction operators corresponding to elements 1, 10, 11 and 17 are presented.

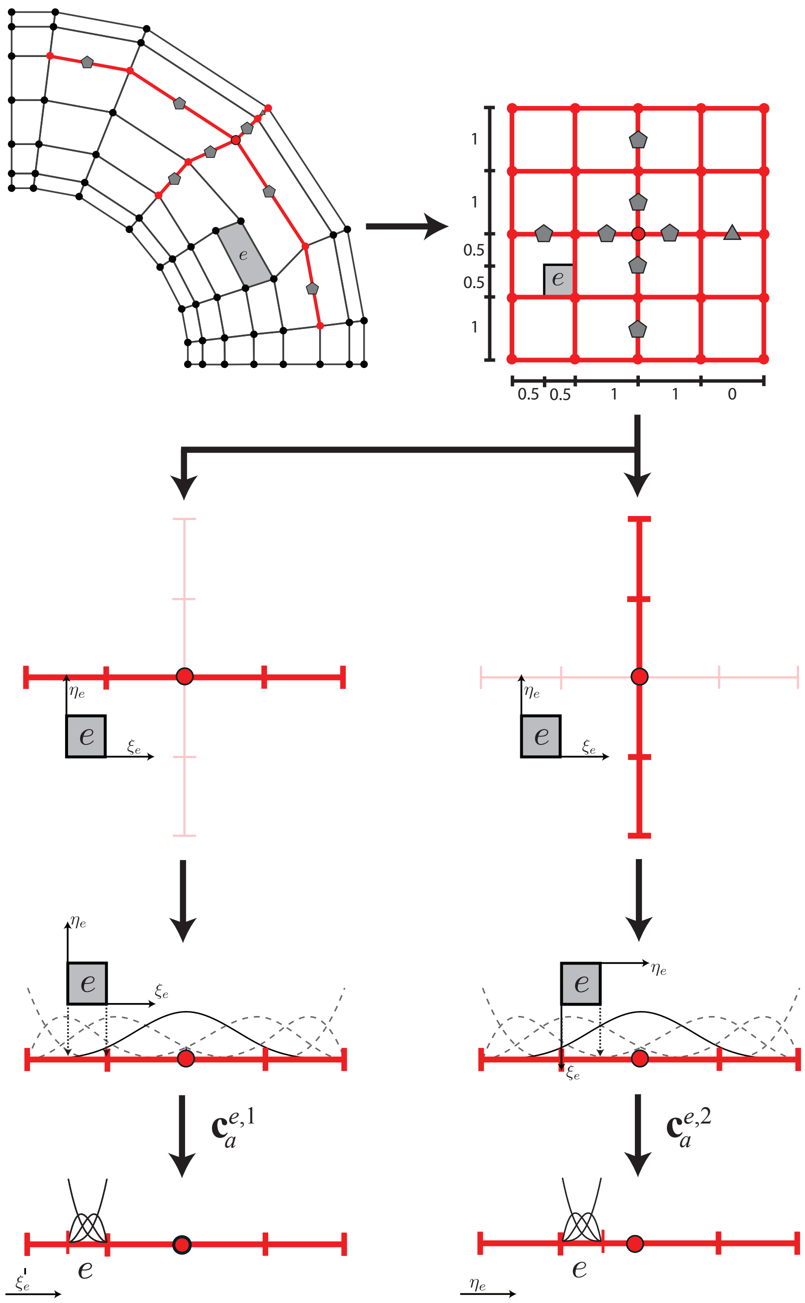
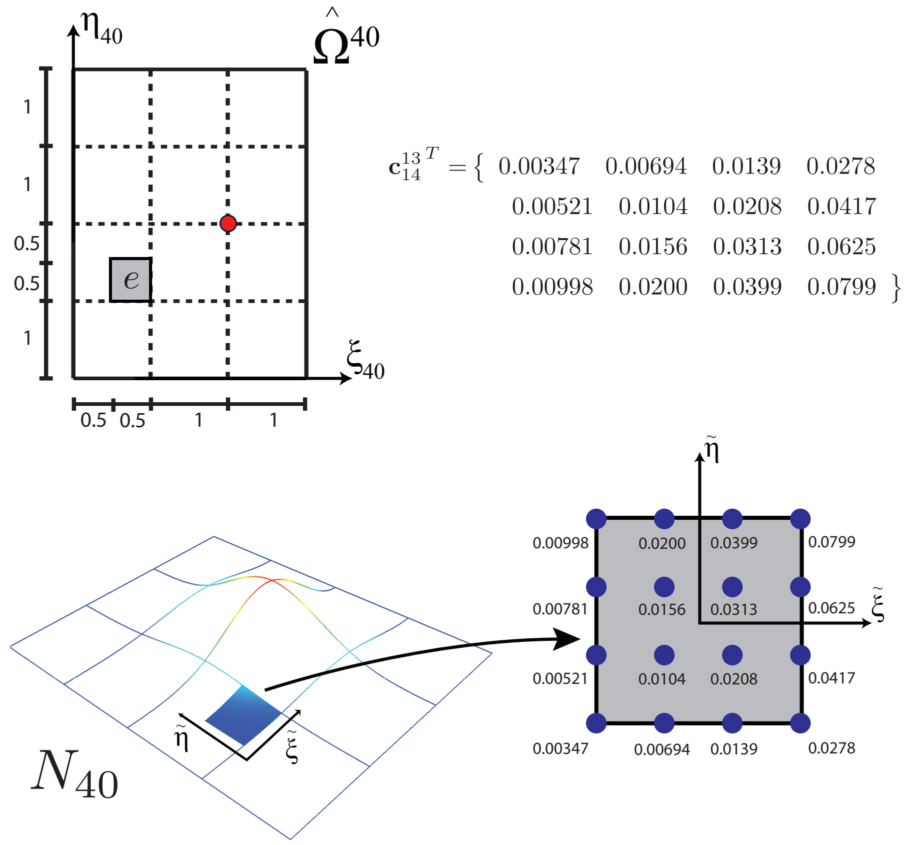


Figure 16. The extraction of a single T-spline basis function. Extraction is decoupled into two univariate extraction operations and then reassembled into a single row of $\mathbf{C}^e$. (The T-spline basis function is the one associated with control point 40, and element e corresponds to element 13 in the elemental T-mesh shown in Figure 8.)



$$
\mathbf{c}_{14}^{13\,T} = \{ \begin{array}{cccc}
0.00347 & 0.00694 & 0.0139 & 0.0278 \\
0.00521 & 0.0104 & 0.0208 & 0.0417 \\
0.00781 & 0.0156 & 0.0313 & 0.0625 \\
0.00998 & 0.0200 & 0.0399 & 0.0799
\end{array} \}
$$

Figure 17. The result of extracting $N_{40}(\boldsymbol{\xi}_{40})$ over element e, where $e = 13$. On the top left, the position of element 13 in the local basis function space of $N_{40}(\boldsymbol{\xi}_{40})$ is shown. On the top right, Bézier extraction of N_{40} over element e (see Figure 16) generates $\mathbf{c}_{14}^{13\,T}$ where 14 is the local index of global control point 40 for element 13. In other words, $40 = \mathrm{IEN}(14, 13)$. Note that this represents a single row of the element extraction operator $\mathbf{C}^{13}$. On the bottom left, $N_{40}(\boldsymbol{\xi}_{40})$ is plotted. The shaded region is the restriction of $N_{40}(\boldsymbol{\xi}_{40})$ to the domain of element 13. On the bottom right, a close-up of element e is shown which portrays graphically the relationship between the extraction coefficients and Bernstein basis functions. Note that $N_{40}(\boldsymbol{\xi}_{40})|_{13} = \mathbf{c}_{14}^{13\,T}\mathbf{B}(\tilde{\boldsymbol{\xi}})$. See equation (26).

4.4. A Bézier extraction algorithm for T-splines

A Bézier extraction algorithm for T-splines consists of the following steps:

1. Infer the T-spline basis from the T-mesh. See Section 2.2.
2. Refine the T-mesh to construct the elemental T-mesh. See Section 3.2.
3. For a T-spline basis function determine the Bézier elements which are in its support. See Section 4.3.
4. For a T-spline basis function perform Bézier extraction. See Sections 4.3.3 and 4.3.4.
5. Repeat steps 3 and 4 for each T-spline basis function.

We note that the topological characterization of a T-mesh described in Section 2.1 is analogous to a quadrilateral mesh with hanging nodes. Thus, traditional quadrilateral meshing data structures and refinement schemes that allow hanging nodes are compatible with T-splines and can be used to perform the first three steps of the algorithm.

The fourth step is the most critical and is not mesh dependent. The primary operation in this step is the univariate extraction of the extended local knot vectors for a T-spline basis function as described in Section 4.3.3. This routine is presented in Algorithm 1.

Algorithm 1 An algorithm to compute the univariate extraction operator rows corresponding to a single univariate T-spline basis function.

```
input   Knot vector, $\Xi = \{\xi_1, \ldots, \xi_{p+2}\}$
        Interior knots to be inserted into $\Xi$, U $= \{\bar{\xi}_1, \ldots, \bar{\xi}_m\}$
        Knot spans in $\Xi$ where new knots will be inserted, spans $= \{\bar{s}_1, \ldots, \bar{s}_m\}$
        Number of interior knots, m
        Curve degree, p
output  Element extraction operator rows $c^e$, $e = 1, 2, \ldots$,p+1+m

// Construct the extended knot vector and count the
// number of knots added to the front of the knot vector.
call compute_extended_knot_vector
    input: $\Xi$
    output: Ubar, nt

// Initialization variables:
a = p+1;
b = a+1;
nb = 1;
$c^1$ = 0;
$c^1$(nt+1) = 1;
mbar = p + 2 + nt + m;
ki = 1;
si = 1;
while  b < mbar  do
   $c^{nb+1}$ = 0; // Initialize the next extraction operator row.
```

```
        // Count multiplicity of the knot at location b.
        add = 0;
        if  si <= m && spans(si) = ki  do
          mult = 0;
          add = 1;
          // Add the new knot to the knot vector.
          Ubar(b+1:mbar+p-m+si) = Ubar(b:mbar+p-m+si-1);
          Ubar(b) = U(si);
          si = si + 1;
        else
           ki = ki+1;
           i = b;
           while  b < m && Ubar(b+1) == Ubar(b)  do  b = b+1;
           mult = b-i+1;
        end

        if  mult < p  do
          numer = Ubar(b)-Ubar(a);
          for  j = p,p-1,...,mult+1  do
             alphas(j-mult) = numer / (Ubar(a+j+add)-Ubar(a));
          end
          r = p-mult;
          // Update the matrix coefficients for r new knots
          for  j=1,2,...,r  do
             save = r-j+1;
             s = mult+j;
             for  k=p+1,p,...,s+1  do
                alpha = alphas(k-s);
                $c^{nb}$(k) = alpha*$c^{nb}$(k) + (1.0-alpha)*$c^{nb}$(k-1);
             end
             if  b < m  do
               // Update overlapping coefficients of the next operator row.
               $c^{nb+1}$(save) = $c^{nb}$(p+1);
             end
          end
          nb = nb + 1; // Finished with the current operator.
          if  b < m   do
            // Update indices for the next operator.
            a = b;
            b = b+1;
          end
        end
    end
```

5. Incorporating $\mathbf{C}^e$ into the finite element formulation

Bézier extraction of T-splines generates a set of Bézier elements (written in terms of the Bernstein basis), the corresponding element extraction operators, $\mathbf{C}^e$, and the IEN array. This structure is identical to what was derived for NURBS in [28] and can be incorporated into a finite element formulation in an analogous fashion. Starting with an abstract weak formulation,

$$(W) \left\{ \begin{array}{l} \text{Given } f \text{, find } u \in \mathcal{S} \text{ such that for all } w \in \mathcal{V} \\ \qquad\qquad a(w,u) = (w,f) \end{array} \right. \tag{27}$$

where $a(\cdot,\cdot)$ is a bilinear form and $(\cdot,\cdot)$ is the L^2 inner-product and $\mathcal{S}$ and $\mathcal{V}$ are the trial solution space and space of weighting functions, respectively, Galerkin's method consists of constructing finite-dimensional approximations of $\mathcal{S}$ and $\mathcal{V}$. In isogeometric analysis these finite-dimensional subspaces $\mathcal{S}^h \subset \mathcal{S}$ and $\mathcal{V}^h \subset \mathcal{V}$ are constructed from the T-spline basis which describes the geometry. The Galerkin formulation is then

$$(G) \left\{ \begin{array}{l} \text{Given } f \text{, find } u^h \in \mathcal{S}^h \text{ such that for all } w^h \in \mathcal{V}^h \\ \qquad\qquad a(w^h,u^h) = (w^h,f) \end{array} \right. \tag{28}$$

In isogeometric analysis, the isoparametric concept is invoked, that is, the field in question is represented in terms of the geometric T-spline basis. We can write u^h and w^h as

$$w^h = \sum_{A=1}^{n} c_A R_A \tag{29}$$

$$u^h = \sum_{B=1}^{n} d_B R_B \tag{30}$$

where c_A and d_B are control variables. Substituting these into (28) yields the matrix form of the problem

$$\mathbf{K}\mathbf{d} = \mathbf{F} \tag{31}$$

where

$$\mathbf{K} = [K_{AB}], \tag{32}$$

$$\mathbf{F} = \{F_A\}, \tag{33}$$

$$\mathbf{d} = \{d_B\}, \tag{34}$$

$$K_{AB} = a\left(R_A, R_B\right), \tag{35}$$

$$F_A = \left(R_A, f\right). \tag{36}$$

The preceding formulation applies to scalar-valued partial differential equations, such as the heat conduction equation. The generalization to vector-valued partial differential equations, such as elasticity, follows standard procedures as described in [34].

5.1. The element shape function routine

As in standard finite elements, the global stiffness matrix, $\mathbf{K}$, and force vector, $\mathbf{F}$, can be constructed by performing integration over the Bézier elements to form element stiffness matrices and force vectors, $\mathbf{k}^e$ and $\mathbf{f}^e$, respectively, and assembling these into their global counterparts. The element form of (35) and (36) is

$$k^e_{ab} = a_e(R^e_a, R^e_b), \tag{37}$$

$$f^e_a = (R^e_a, f)_e \tag{38}$$

where $a_e(\cdot,\cdot)$ denotes the bilinear form restricted to the element, $(\cdot,\cdot)_e$ is the L^2 inner-product restricted to the element, and R^e_a are the element shape functions. The integration is usually performed by Gaussian quadrature. Since T-splines are not, in general, defined over a global parametric domain, all integrals are pulled back directly to the bi-unit parent element domain. No intermediate parametric domain is employed. This requires the evaluation of the global T-spline basis functions, their derivatives, and the Jacobian determinant of the pullback from the physical space to the parent element at each quadrature point in the parent element. These evaluations are done in an element shape function routine. Employing the element extraction operators we have that,

$$\mathbf{R}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{W}^e\mathbf{C}^e\frac{\mathbf{B}(\tilde{\boldsymbol{\xi}})}{W^e(\tilde{\boldsymbol{\xi}})}. \tag{39}$$

where

$$W^e(\tilde{\boldsymbol{\xi}}) = (\mathbf{w}^e)^T\mathbf{C}^e\mathbf{B}(\tilde{\boldsymbol{\xi}}). \tag{40}$$

The derivatives of $\mathbf{R}^e$ with respect to the local parent coordinates, $\tilde{\xi}^i$, are

$$\frac{\partial \mathbf{R}^e(\tilde{\boldsymbol{\xi}})}{\partial \tilde{\xi}^i} = \mathbf{W}^e\mathbf{C}^e\frac{\partial}{\partial \tilde{\xi}^i}\left(\frac{\mathbf{B}(\tilde{\boldsymbol{\xi}})}{W^e(\tilde{\boldsymbol{\xi}})}\right) = \mathbf{W}^e\mathbf{C}^e\left(\frac{1}{W^e(\tilde{\boldsymbol{\xi}})}\frac{\partial \mathbf{B}(\tilde{\boldsymbol{\xi}})}{\partial \tilde{\xi}^i} - \frac{\partial W^e(\tilde{\boldsymbol{\xi}})}{\partial \tilde{\xi}^i}\frac{\mathbf{B}(\tilde{\boldsymbol{\xi}})}{(W^e(\tilde{\boldsymbol{\xi}}))^2}\right). \tag{41}$$

To compute the derivatives with respect to the physical coordinates, $(\tilde{x}^e_1, \tilde{x}^e_2, \tilde{x}^e_3)$, we apply the chain rule to get

$$\frac{\partial \mathbf{R}^e(\tilde{\boldsymbol{\xi}})}{\partial \tilde{x}^e_i} = \sum_{j=1}^{3}\frac{\partial \mathbf{R}^e(\tilde{\boldsymbol{\xi}})}{\partial \tilde{\xi}^j}\frac{\partial \tilde{\xi}^j}{\partial \tilde{x}^e_i}. \tag{42}$$

To compute $\partial\tilde{\boldsymbol{\xi}}/\partial\tilde{\mathbf{x}}^e$ we first compute $\partial\tilde{\mathbf{x}}^e/\partial\tilde{\boldsymbol{\xi}}$ using (13) and (41) and then take its inverse. Since we are integrating over the parent element we must also compute the Jacobian determinant, J^e, of the mapping from the parent element to the physical space. It is computed as

$$J^e = \left|\frac{\partial \tilde{\mathbf{x}}^e}{\partial \tilde{\boldsymbol{\xi}}}\right|. \tag{43}$$

Higher-order derivatives can also be computed as described in [2].

5.2. Finite element data structures

To employ an extracted T-spline in an existing finite element framework requires, in addition to the IEN array and extraction operators, the ID array and Bézier mesh. It is also useful to construct the LM array, which can be defined directly from the IEN and ID arrays.

Remark: The IEN, ID, and LM arrays are standard data processing arrays in finite element programs (see [34] for a full discussion). The acronyms mean, "element nodes," "destination," and "location matrix," respectively. The I's in IEN and ID are indicating the arrays are integer valued, corresponding to the implicit typing convention frequently used in FORTRAN programs. The implicit typing convention defines all variable names beginning with I, J, K, L, M, and N as integer valued, hence, LM is automatically integer valued. For the reader familiar with FORTRAN these remarks are unnecessary. However, FORTRAN is not common in CAGD, and many younger programmers are unfamiliar with it and its conventions. It is still widely utilized in computational mechanics, although the trend is toward C++.

5.2.1. Constructing the ID *array* The ID array for the example domain Ω in Figure 2 is shown in Figure 18, where two degrees-of-freedom are assigned to every control point. This array maps the degree-of-freedom number (e.g., the direction index of a displacement component) and global control point number to the corresponding equation number in the global system. A zero value indicates a degree of freedom that is specified by the boundary conditions and for which the equation has been removed from the global system. For the domain Ω the horizontal and vertical displacements of control points 1, 9, 17, 29, 37, 44, and 51 are assumed to be specified.

5.2.2. Computing the Bézier mesh Once the IEN array and element extraction operators have been computed, the control points for the Bézier elements can be computed by combining equations (9) and (15). In general, we obtain the Bézier control points $\mathbf{Q}^e$ and Bézier weights $\mathbf{w}^{b,e}$ for element e as

$$\mathbf{Q}^e = \left(\mathbf{W}^{b,e}\right)^{-1} (\mathbf{C}^e)^T \, \mathbf{W}^e \mathbf{P}^e, \tag{44}$$

$$\mathbf{w}^{b,e} = (\mathbf{C}^e)^T \, \mathbf{w}^e, \tag{45}$$

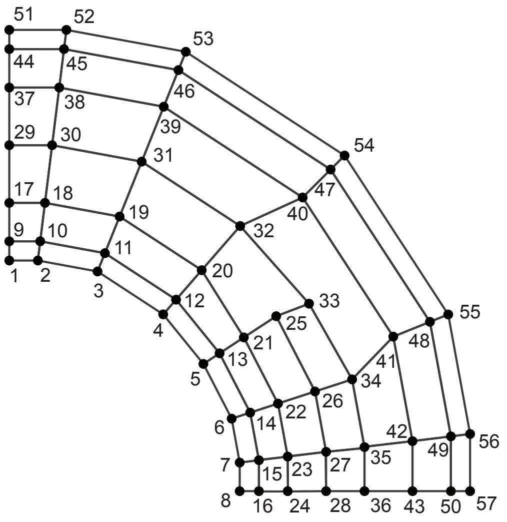
where $\mathbf{P}^e$ are the T-spline control points, $\mathbf{W}^{b,e}$ is the diagonal matrix of Bézier weights, and $\mathbf{W}^e$ is the diagonal matrix of T-spline weights, $\mathbf{w}^e$. Figure 19 shows the result of (44) for Bézier elements 1, 10, 11, 17 (see Appendix II for the element control point coordinates). The T-spline control points which contribute to the location of the Bézier element control points are indicated by the ◯'s. Each T-spline element (see Section 3.5) has an equivalent representation as an extracted Bézier element. In other words

$$\tilde{\mathbf{x}}^e(\tilde{\boldsymbol{\xi}}) = \tilde{\mathbf{x}}^{b,e}(\tilde{\boldsymbol{\xi}}) = \frac{1}{W^{b,e}(\tilde{\boldsymbol{\xi}})} (\mathbf{Q}^e)^T \mathbf{W}^{b,e} \mathbf{B}(\tilde{\boldsymbol{\xi}}). \tag{46}$$

where $W^{b,e}(\tilde{\boldsymbol{\xi}}) = {\mathbf{w}^{b,e}}^T \, \mathbf{B}(\tilde{\boldsymbol{\xi}})$. We have defined three "meshes" for the T-spline example in Figure 2 – the elemental T-mesh, the Bézier control mesh, and the Bézier physical mesh. Figure 20 shows a representative element in each of these meshes and how they are related using $\mathbf{C}^e$. It is important to remember that the Bézier physical mesh defines the set of finite elements used in computations. The extraction operator corresponding to each of these elements is then used to constrain the Bézier degrees-of-freedom such that the original smoothness of the T-spline is present in the finite element solution.

We have implemented a finite element code which incorporates Bézier elements and the extraction operator and have tested this approach extensively. In all cases where a comparison exists, it gives identical results to those obtained using isogeometric implementations which do not utilize T-spline extraction. We have also found that this approach greatly reduces finite element assembly times and reduces the complexity of parallelizing isogeometric codes. Another important practical advantage to this approach is that it allows an analyst to compute with T-splines without understanding the details of

T-spline technology. The extraction paradigm provides a simple interface which is readily accessible to most finite element practitioners.



ID array:

Degree-of-freedom number (i)	Global control point number (A)														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	3	5	7	9	11	13	0	15	17	19	21	23	25
2	0	2	4	6	8	10	12	14	0	16	18	20	22	24	26

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
27	0	29	31	33	35	37	39	41	43	45	47	49	0	51
28	0	30	32	34	36	38	40	42	44	46	48	50	0	52

31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
53	55	57	59	61	63	0	65	67	69	71	73	75	0	77
54	56	58	60	62	64	0	66	68	70	72	74	76	0	78

46	47	48	49	50	51	52	53	54	55	56	57
79	81	83	85	87	0	89	91	93	95	97	99
80	82	84	86	88	0	90	92	94	96	98	100

$$P = \mathrm{ID}(i, A)$$

Figure 18. The ID array maps the degree-of-freedom number (i.e., direction index of the displacement component) and global control point number to the corresponding equation number in the global system. For this example the horizontal and vertical displacements of control points 1, 9, 17, 29, 37, 44, and 51 are specified by boundary conditions. The global equations corresponding to these degrees-of-freedom are removed from the system through the ID array. The LM array is computed as follows: $P = \mathrm{LM}(i, a, e) = \mathrm{ID}(i, \mathrm{IEN}(a, e))$.

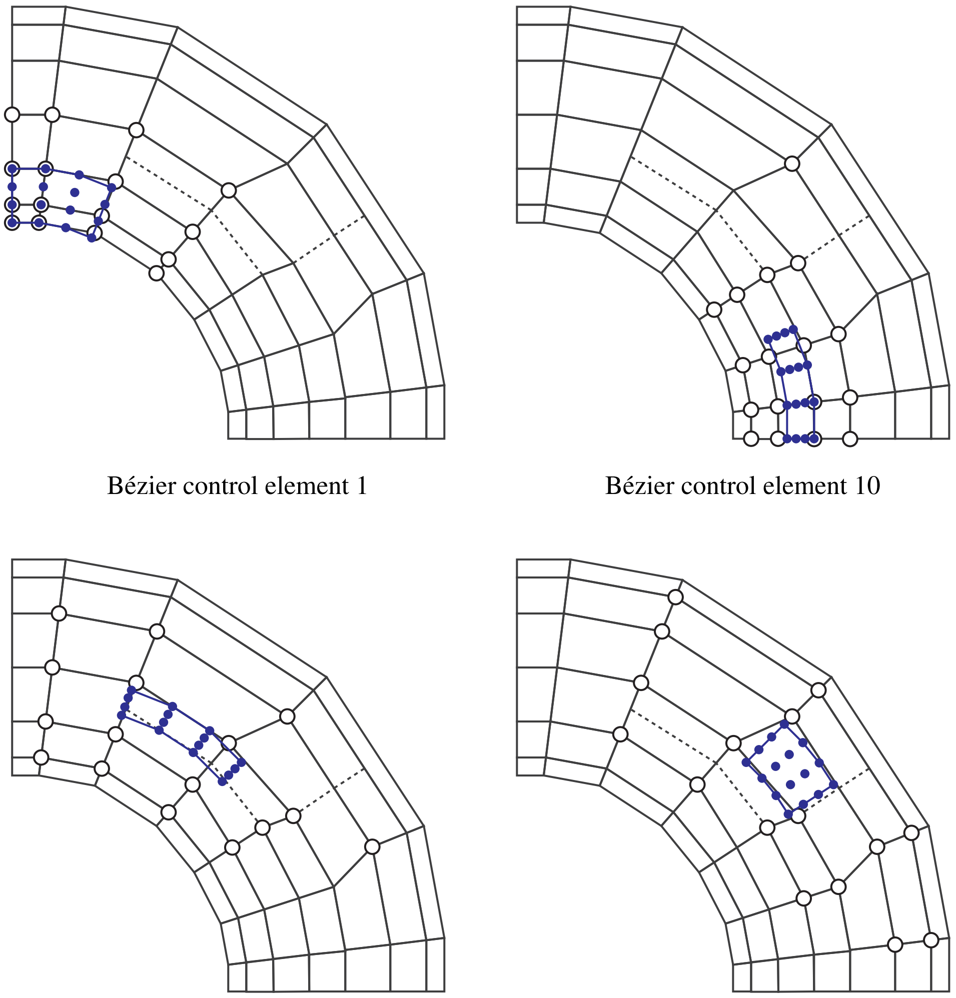

Bézier control element 1
Bézier control element 10
Bézier control element 11
Bézier control element 17

Figure 19. The extraction operators and IEN array can be used to construct the Bézier control elements. For each control element e the $\bigcirc$'s indicate the global T-spline control points which influence the location of Bézier control points, indicated by the $\bullet$'s. The global control points that influence each control element are determined by the IEN array, and the location of the element control points is computed with the element extraction operator $\mathbf{C}^e$.

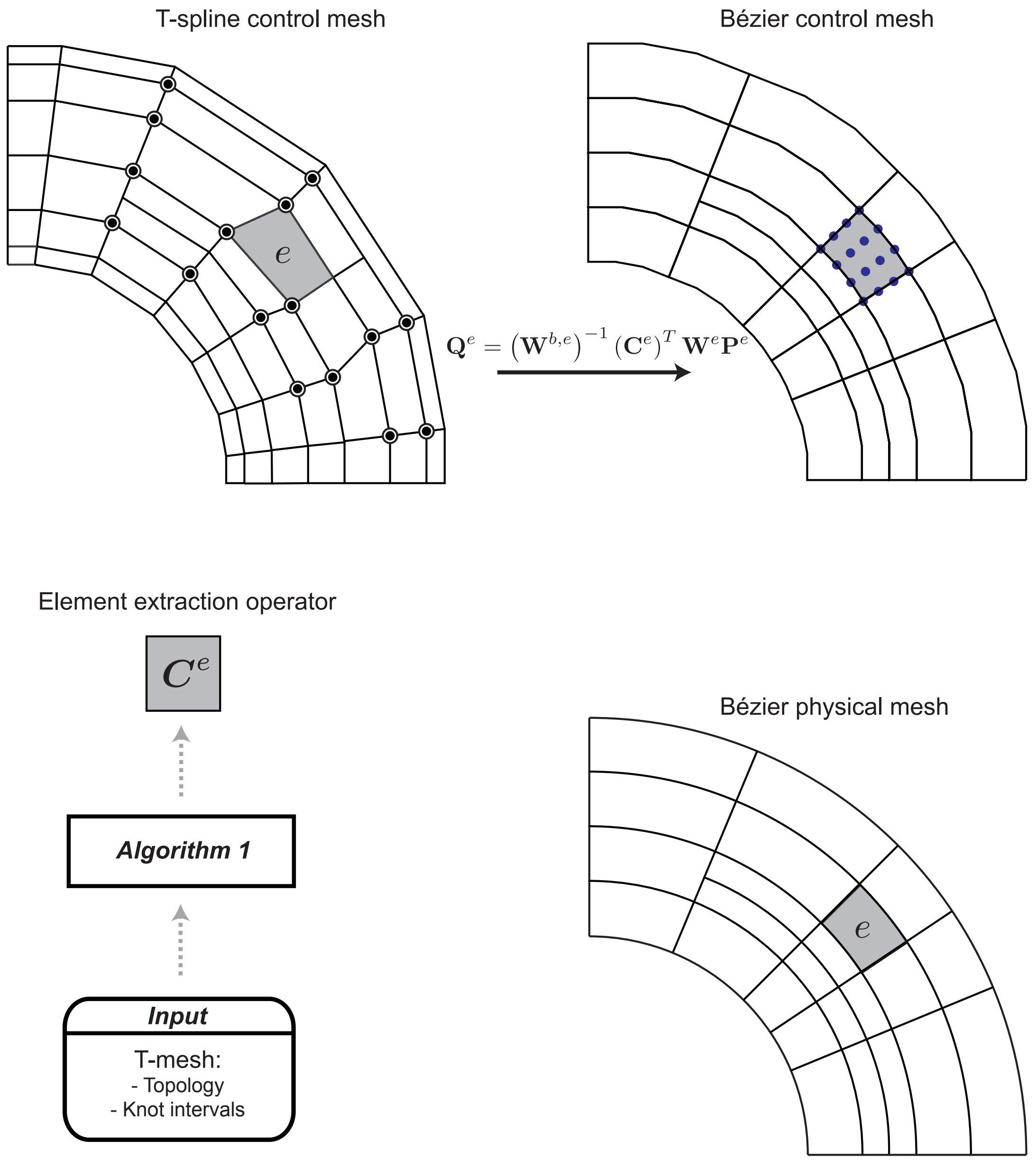

T-spline control mesh
Bézier control mesh
e
$\mathbf{Q}^e = \left(\mathbf{W}^{b,e}\right)^{-1} (\mathbf{C}^e)^T\, \mathbf{W}^e \mathbf{P}^e$
Element extraction operator
C^e
Algorithm 1
Input
T-mesh:
- Topology
- Knot intervals
Bézier physical mesh
e

Figure 20. Various “meshes” of the T-spline example.

6. Conclusion

We have used Bézier extraction to develop a local finite element representation of a global T-spline. The global topological and smoothness information is contained in element extraction operators, which facilitate an elegant implementation of T-junctions, known as "hanging nodes" in finite element analysis. Each element basis has the same form in terms of Bernstein polynomials. The changes necessary to standard finite element programs are confined to shape function subroutines. Element formation routines and overall architecture are unaffected. The assembly of global arrays uses the same data processing arrays as finite element programs (e.g., the IEN, ID, and LM arrays; see Hughes [34]). Existing finite element programs can be easily generalized to perform isogeometric analysis with T-splines using the methodology described.

ACKNOWLEDGEMENTS

This work was supported by grants from the Office of Naval Research (N00014-08-1-0992), the National Science Foundation (CMI-0700807), and SINTEF (UTA10-000374). M.A. Scott was partially supported by an ICES CAM Graduate Fellowship. M.J. Borden was partially supported by Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. This support is gratefully acknowledged.

APPENDIX

I. T-spline control points for the T-spline example

Table III lists the global T-spline control point coordinates for the geometry in Figure 2.

Control point	x	y	w	Control point	x	y	w
1	0.0000	1.5000	1.0000	30	0.2788	2.2500	0.9512
2	0.1858	1.5000	0.9512	31	0.8618	2.1432	0.8780
3	0.5746	1.4288	0.8780	32	1.5033	1.7245	0.8475
4	1.0022	1.1497	0.8475	33	1.9516	1.2194	0.7895
5	1.2637	0.8275	0.8597	34	2.2319	0.7268	0.8963
6	1.4477	0.4714	0.8963	35	2.3125	0.2865	0.9512
7	1.5000	0.1858	0.9512	36	2.3125	0.0000	1.0000
8	1.5000	0.0000	1.0000	37	0.0000	2.6250	1.0000
9	0.0000	1.6250	1.0000	38	0.3252	2.6250	0.9512
10	0.2013	1.6250	0.9512	39	1.0055	2.5004	0.8780
11	0.6224	1.5479	0.8780	40	1.9100	1.9100	0.8413
12	1.0857	1.2455	0.8475	41	2.5004	1.0055	0.8780
13	1.3690	0.8964	0.8597	42	2.6250	0.3252	0.9512
14	1.5683	0.5107	0.8963	43	2.6250	0.0000	1.0000
15	1.6250	0.2013	0.9512	44	0.0000	2.8750	1.0000
16	1.6250	0.0000	1.0000	45	0.3562	2.8750	0.9512
17	0.0000	1.8750	1.0000	46	1.1012	2.7386	0.8780
18	0.2323	1.8750	0.9512	47	2.0919	2.0919	0.8413
19	0.7182	1.7860	0.8780	48	2.7386	1.1012	0.8780
20	1.2527	1.4371	0.8475	49	2.8750	0.3562	0.9512
21	1.5270	0.9999	0.8597	50	2.8750	0.0000	1.0000
22	1.7493	0.5696	0.8963	51	0.0000	3.0000	1.0000
23	1.8425	0.2246	0.9512	52	0.3717	3.0000	0.9512
24	1.8425	0.0000	1.0000	53	1.1491	2.8576	0.8780
25	1.7376	1.1378	0.8597	54	2.1829	2.1829	0.8413
26	1.9906	0.6482	0.8963	55	2.8576	1.1491	0.8780
27	2.0625	0.2555	0.9512	56	3.0000	0.3717	0.9512
28	2.0625	0.0000	1.0000	57	3.0000	0.0000	1.0000
29	0.0000	2.2500	1.0000				

Table III. The control point ($\mathbf{P}$) coordinates (x, y) and weights (w).

II. Bézier element control points for the T-spline example

Table IV lists the Bézier element control points for elements 1, 10, 11, and 17 from the geometry in Figure 2.

$a(i,j)$	x	y	w
1	0.5521	1.3947	0.8902
2	0.5982	1.5109	0.8902
3	0.6442	1.6271	0.8902
4	0.6902	1.7433	0.8902
5	0.3724	1.4658	0.9146
6	0.4035	1.5880	0.9146
7	0.4345	1.7101	0.9146
8	0.4655	1.8323	0.9146
9	0.1858	1.5000	0.9512
10	0.2013	1.6250	0.9512
11	0.2168	1.7500	0.9512
12	0.2323	1.8750	0.9512
13	0.0000	1.5000	1.0000
14	0.0000	1.6250	1.0000
15	0.0000	1.7500	1.0000
16	0.0000	1.8750	1.0000

$e = 1$

$a(i,j)$	x	y	w
1	1.8750	0.0000	1.0000
2	1.9375	0.0000	1.0000
3	2.0000	0.0000	1.0000
4	2.0625	0.0000	1.0000
5	1.8750	0.2323	0.9512
6	1.9375	0.2401	0.9512
7	2.0000	0.2478	0.9512
8	2.0625	0.2555	0.9512
9	1.8323	0.4655	0.9146
10	1.8934	0.4810	0.9146
11	1.9544	0.4966	0.9146
12	2.0155	0.5121	0.9146
13	1.7433	0.6902	0.8902
14	1.8015	0.7132	0.8902
15	1.8596	0.7362	0.8902
16	1.9177	0.7592	0.8902

$e = 10$

$a(i,j)$	x	y	w
1	1.4584	1.4584	0.8536
2	1.5026	1.5026	0.8536
3	1.5468	1.5468	0.8536
4	1.5910	1.5910	0.8536
5	1.2570	1.6598	0.8536
6	1.2951	1.7101	0.8536
7	1.3332	1.7604	0.8536
8	1.3713	1.8107	0.8536
9	1.0202	1.8143	0.8658
10	1.0512	1.8693	0.8658
11	1.0821	1.9243	0.8658
12	1.1130	1.9793	0.8658
13	0.7592	1.9177	0.8902
14	0.7822	1.9758	0.8902
15	0.8052	2.0339	0.8902
16	0.8282	2.0920	0.8902

$e = 11$

$a(i,j)$	x	y	w
1	1.8832	1.2312	0.8627
2	1.9879	1.2996	0.8627
3	2.0925	1.3680	0.8627
4	2.1971	1.4364	0.8627
5	1.7985	1.3608	0.8566
6	1.8985	1.4364	0.8566
7	1.9984	1.5120	0.8566
8	2.0983	1.5876	0.8566
9	1.7008	1.4811	0.8536
10	1.7953	1.5634	0.8536
11	1.8898	1.6457	0.8536
12	1.9843	1.7280	0.8536
13	1.5910	1.5910	0.8536
14	1.6794	1.6794	0.8536
15	1.7678	1.7678	0.8536
16	1.8561	1.8561	0.8536

$e = 17$

Table IV. Bézier element control points for elements 1, 10, 11 and 17, as shown in Figure 9. The array $a(i,j)$ is defined in (22).

III. Bézier element extraction operators for the T-spline example

The extraction operators for elements 1, 10, 11 and 17 are listed below. These operators can be used to retrieve the Bézier control points listed in Appendix II from the global T-spline control points in Appendix I using (44). We note that because most extraction operators have many zero entries it is important to employ a sparse matrix storage format (see [35] and references therein.)

$$\mathbf{C}^1 = \left[\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.5000 & 0.0000 & 0.0000 & 0.0000 \\
0.5500 & 0.0000 & 0.0000 & 0.0000 & 0.5000 & 0.0000 & 0.0000 & 0.0000 \\
0.2000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2500 & 0.1250 & 0.0625 & 0.0000 & 0.5000 & 0.2500 & 0.1250 \\
0.0000 & 0.5500 & 0.2750 & 0.1375 & 0.0000 & 0.5000 & 0.2500 & 0.1250 \\
0.0000 & 0.2000 & 0.1000 & 0.0500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.1250 & 0.1458 & 0.0000 & 0.0000 & 0.2500 & 0.2917 \\
0.0000 & 0.0000 & 0.2750 & 0.3208 & 0.0000 & 0.0000 & 0.2500 & 0.2917 \\
0.0000 & 0.0000 & 0.1000 & 0.1167 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0833 \\
0.0000 & 0.0000 & 0.0000 & 0.0917 & 0.0000 & 0.0000 & 0.0000 & 0.0833 \\
0.0000 & 0.0000 & 0.0000 & 0.0333 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right.$$

$$\left.\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\
1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.5000 & 0.2500 \\
0.0000 & 1.0000 & 0.5000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5000 & 0.5833 \\
0.0000 & 0.0000 & 0.5000 & 0.5833 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1667 \\
0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right]$$

$$
\mathbf{C}^{10} = \left[\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.0000 & 0.0000 & 0.0000 \\
0.1111 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5556 & 0.5000 & 0.2500 & 0.1250 \\
0.5556 & 0.5000 & 0.2500 & 0.1250 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.3333 & 0.5000 & 0.7500 & 0.7500 \\
0.3333 & 0.5000 & 0.7500 & 0.7500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1250 \\
0.0000 & 0.0000 & 0.0000 & 0.1250 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array} \right.
$$

$$
\left. \begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0370 & 0.0000 & 0.0000 & 0.0000 \\
0.0741 & 0.0000 & 0.0000 & 0.0000 & 0.0617 & 0.0000 & 0.0000 & 0.0000 \\
0.0370 & 0.0000 & 0.0000 & 0.0000 & 0.0123 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1852 & 0.1667 & 0.0833 & 0.0417 \\
0.3704 & 0.3333 & 0.1667 & 0.0833 & 0.3086 & 0.2778 & 0.1389 & 0.0694 \\
0.1852 & 0.1667 & 0.0833 & 0.0417 & 0.0617 & 0.0556 & 0.0278 & 0.0139 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.1667 & 0.2500 & 0.2500 \\
0.2222 & 0.3333 & 0.5000 & 0.5000 & 0.1852 & 0.2778 & 0.4167 & 0.4167 \\
0.1111 & 0.1667 & 0.2500 & 0.2500 & 0.0370 & 0.0556 & 0.0833 & 0.0833 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0417 \\
0.0000 & 0.0000 & 0.0000 & 0.0833 & 0.0000 & 0.0000 & 0.0000 & 0.0694 \\
0.0000 & 0.0000 & 0.0000 & 0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0139 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0035
\end{array} \right]
$$

$$
\mathbf{C}^{11} = \left[\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0021 & 0.0000 & 0.0000 & 0.0000 & 0.0063 & 0.0000 & 0.0000 & 0.0000 \\
0.0188 & 0.0000 & 0.0000 & 0.0000 & 0.0250 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0312 & 0.0250 & 0.0167 & 0.0111 & 0.0937 & 0.0750 & 0.0500 & 0.0333 \\
0.2812 & 0.2250 & 0.1500 & 0.1000 & 0.3750 & 0.3000 & 0.2000 & 0.1333 \\
0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.2500 & 0.2500 & 0.1667 & 0.1111 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0319 & 0.0389 & 0.0444 & 0.0444 & 0.0958 & 0.1167 & 0.1333 & 0.1333 \\
0.2875 & 0.3500 & 0.4000 & 0.4000 & 0.3833 & 0.4667 & 0.5333 & 0.5333 \\
0.0417 & 0.0833 & 0.1667 & 0.2000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0035 & 0.0069 & 0.0139 & 0.0278 & 0.0069 & 0.0139 & 0.0278 & 0.0556 \\
0.0139 & 0.0278 & 0.0556 & 0.1111 & 0.0139 & 0.0278 & 0.0556 & 0.1111 \\
0.0000 & 0.0000 & 0.0000 & 0.0111 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right.
$$

$$
\left.\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0078 & 0.0000 & 0.0000 & 0.0000 \\
0.0187 & 0.0000 & 0.0000 & 0.0000 & 0.0172 & 0.0000 & 0.0000 & 0.0000 \\
0.0125 & 0.0000 & 0.0000 & 0.0000 & 0.0063 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1172 & 0.0937 & 0.0625 & 0.0417 \\
0.2812 & 0.2250 & 0.1500 & 0.1000 & 0.2578 & 0.2063 & 0.1375 & 0.0917 \\
0.1875 & 0.1500 & 0.1000 & 0.0667 & 0.0938 & 0.0750 & 0.0500 & 0.0333 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1198 & 0.1458 & 0.1667 & 0.1667 \\
0.2875 & 0.3500 & 0.4000 & 0.4000 & 0.2635 & 0.3208 & 0.3667 & 0.3667 \\
0.1917 & 0.2333 & 0.2667 & 0.2667 & 0.0958 & 0.1167 & 0.1333 & 0.1333 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0052 & 0.0104 & 0.0208 & 0.0417 \\
0.0139 & 0.0278 & 0.0556 & 0.1111 & 0.0122 & 0.0243 & 0.0486 & 0.0972 \\
0.0069 & 0.0139 & 0.0278 & 0.0556 & 0.0035 & 0.0069 & 0.0139 & 0.0278 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right]
$$

$$\mathbf{C}^{17} = \left[\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0208 & 0.0000 & 0.0000 & 0.0000 & 0.0417 & 0.0000 & 0.0000 & 0.0000 \\
0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 \\
0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0833 & 0.0833 & 0.0417 & 0.0208 & 0.1667 & 0.1667 & 0.0833 & 0.0417 \\
0.4500 & 0.6000 & 0.3000 & 0.1500 & 0.4500 & 0.6000 & 0.3000 & 0.1500 \\
0.0750 & 0.1000 & 0.0500 & 0.0250 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0035 & 0.0069 & 0.0139 & 0.0122 & 0.0069 & 0.0139 & 0.0278 & 0.0243 \\
0.0799 & 0.1597 & 0.3194 & 0.2795 & 0.0972 & 0.1944 & 0.3889 & 0.3403 \\
0.0312 & 0.0937 & 0.2812 & 0.2578 & 0.0250 & 0.0750 & 0.2250 & 0.2062 \\
0.0021 & 0.0062 & 0.0187 & 0.0172 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0052 & 0.0000 & 0.0000 & 0.0000 & 0.0104 \\
0.0000 & 0.0000 & 0.0000 & 0.1198 & 0.0000 & 0.0000 & 0.0000 & 0.1458 \\
0.0000 & 0.0000 & 0.0000 & 0.1172 & 0.0000 & 0.0000 & 0.0000 & 0.0937 \\
0.0000 & 0.0000 & 0.0000 & 0.0078 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array} \right.$$

$$\left. \begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0111 & 0.0000 & 0.0000 & 0.0000 \\
0.0833 & 0.0000 & 0.0000 & 0.0000 & 0.1000 & 0.0000 & 0.0000 & 0.0000 \\
0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0444 & 0.0444 & 0.0222 & 0.0111 \\
0.3333 & 0.3333 & 0.1667 & 0.0833 & 0.4000 & 0.4000 & 0.2000 & 0.1000 \\
0.3000 & 0.4000 & 0.2000 & 0.1000 & 0.2000 & 0.2667 & 0.1333 & 0.0667 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0139 & 0.0278 & 0.0556 & 0.0486 & 0.0278 & 0.0556 & 0.1111 & 0.0972 \\
0.1111 & 0.2222 & 0.4444 & 0.3889 & 0.1111 & 0.2222 & 0.4444 & 0.3889 \\
0.0167 & 0.0500 & 0.1500 & 0.1375 & 0.0111 & 0.0333 & 0.1000 & 0.0917 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0208 & 0.0000 & 0.0000 & 0.0000 & 0.0417 \\
0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 \\
0.0000 & 0.0000 & 0.0000 & 0.0625 & 0.0000 & 0.0000 & 0.0000 & 0.0417 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array} \right]$$

REFERENCES

1. Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. Computer Methods in Applied Mechanics and Engineering 2005; **194**:4135–4195.
2. Cottrell JA, Hughes TJR, Bazilevs Y. Isogeometric analysis: Toward Integration of CAD and FEA. Wiley: Chichester, 2009.
3. Akkerman I, Bazilevs Y, Calo V, Hughes TJR, Hulshoff S. The role of continuity in residual-based variational multiscale modeling of turbulence. Computational Mechanics 2008; **41**:371–378.
4. Bazilevs Y, Calo VM, Cottrell JA, Hughes TJR, Reali A, Scovazzi G. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Computer Methods in Applied Mechanics and Engineering 2007; **197**:173–201.
5. Bazilevs Y, Akkerman I. Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and residual-based variational multiscale method. Journal of Computational Physics 2010; **229**:3402–3414.
6. Bazilevs Y, Calo VM, Zhang Y, Hughes TJR. Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. Computational Mechanics 2006; **38**:310–322.
7. Bazilevs Y, Calo VM, Hughes TJR, Zhang Y. Isogeometric fluid-structure interaction: Theory, algorithms, and computations. Computational Mechanics 2008; **43**:3–37.
8. Zhang Y, Bazilevs Y, Goswami S, Bajaj C, Hughes TJR. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Computer Methods in Applied Mechanics and Engineering 2007; **196**:2943–2959.
9. Auricchio F, da Veiga LB, Lovadina C, Reali A. The importance of the exact satisfaction of the incompressibility constraint in nonlinear elasticity: mixed fems versus NURBS-based approximations. Computer Methods in Applied Mechanics and Engineering 2010; **199**:314–323.
10. Auricchio F, da Veiga LB, Buffa A, Lovadina C, Reali A, Sangalli G. A fully "locking-free" isogeometric approach for plane linear elasticity problems: A stream function formulation. Computer Methods in Applied Mechanics and Engineering 2007; **197**:160–172.
11. Elguedj T, Bazilevs Y, Calo VM, Hughes TJR. $\bar{B}$ and $\bar{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements. Computer Methods in Applied Mechanics and Engineering 2008; **197**:2732–2762.
12. Cottrell JA, Hughes TJR, Reali A. Studies of refinement and continuity in isogeometric analysis. Computer Methods in Applied Mechanics and Engineering 2007; **196**:4160–4183.
13. Cottrell JA, Reali A, Bazilevs Y, Hughes TJR. Isogeometric analysis of structural vibrations. Computer Methods in Applied Mechanics and Engineering 2006; **195**:5257–5296.
14. Benson DJ, Bazilevs Y, Hsu MC, Hughes TJR. Isogeometric shell analysis: The Reissner-Mindlin shell. Computer Methods in Applied Mechanics and Engineering 2010; **199**(5-8):276 – 289.
15. Benson DJ, Bazilevs Y, Hsu MC, Hughes TJR. A large deformation, rotation-free, isogeometric shell. International Journal for Numerical Methods in Engineering to appear, 2010; .
16. Benson DJ, Bazilevs Y, De Luycker E, Hsu MC, Scott MA, Hughes TJR, Belytschko T. A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM. International Journal for Numerical Methods in Engineering 2010; **83**:765–785.
17. Gomez H, Calo VM, Bazilevs Y, Hughes TJR. Isogeometric analysis of the Cahn-Hilliard phase-field model. Computer Methods in Applied Mechanics and Engineering 2008; **197**:4333–4352.
18. Gomez H, Hughes TJR, Nogueira X, Calo VM. Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations. Computer Methods in Applied Mechanics and Engineering 2010; **in press, doi:10.1016/j.cma.2010.02.010**.
19. Lipton S, Evans JA, Bazilevs Y, Elguedj T, Hughes TJR. Robustness of isogeometric structural discretizations under severe mesh distortion. Computer Methods in Applied Mechanics and Engineering 2010; **199**(5-8):357 – 373.
20. Wall WA, Frenzel MA, Cyron C. Isogeometric structural shape optimization. Computer Methods in Applied Mechanics and Engineering 2008; **197**:2976–2988.
21. Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCSs. ACM Transactions on Graphics 2003; **22 (3)**:477–484.
22. Sederberg TW, Cardon DL, Finnigan GT, North NS, Zheng J, Lyche T. T-spline simplification and local refinement. ACM Transactions on Graphics 2004; **23 (3)**:276–283.
23. Sederberg TW, Finnigan GT, Li X, Lin H. Watertight trimmed NURBS. ACM Transactions on Graphics 2008; **27**(3):Article no. 79.
24. Bazilevs Y, Calo VM, Cottrell JA, Evans JA, Hughes TJR, Lipton S, Scott MA, Sederberg TW. Isogeometric analysis using T-splines. Computer Methods in Applied Mechanics and Engineering 2010; **199**(5-8):229 – 263, doi:DOI: 10.1016/j.cma.2009.02.036.
25. Dörfel M, Jüttler B, Simeon B. Adaptive isogeometric analysis by local *h*-refinement with T-splines. Computer Methods in Applied Mechanics and Engineering 2009; **199**(5–8):264–275.
26. Verhoosel CV, Scott MA, de Borst R, Hughes TJR. An isogeometric approach to cohesive zone modeling. International Journal for Numerical Methods in Engineering 2010; **accepted for publication**.

27. Verhoosel CV, Scott MA, Hughes TJR, de Borst, R. An isogeometric analysis approach to gradient damage models. International Journal for Numerical Methods in Engineering 2010; **submitted for publication**.
28. Borden MJ, Scott MA, Evans JA, Hughes TJR. Isogeometric finite element data structures based on Bézier extraction of NURBS. International Journal for Numerical Methods in Engineering 2010; **in press, doi: 10.1002/nme.2968**.
29. Finnigan GT. Arbitrary degree T-splines. Master's Thesis, Brigham Young University August 2008.
30. Sederberg TW, Zheng J, Song X. Knot intervals and multi-degree splines. Computer Aided Geometric Design 2003; **20**:455–468.
31. Li X, Zheng J, Sederberg TW, Hughes TJR, Scott MA. On linear independence of T-splines. Computer Aided Geometric Design 2010; Submitted for publication (see also ICES Report 10-40 at http://www.ices.utexas.edu/research/reports).
32. Sederberg TW. Computer-Aided Geometric Design: Course Notes 2010. http://www.tsplines.com/educationportal.html.
33. Piegl L, Tiller W. The NURBS Book (Monographs in Visual Communication), 2nd ed. Springer-Verlag: New York, 1997.
34. Hughes TJR. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover Publications: Mineola, NY, 2000.
35. Saad Y. Iterative methods for sparse linear systems. PWS Pub. Co.: Albany, NY, 1996.