coreform

U-SPLINES: SPLINES OVER UNSTRUCTURED MESHES

Patent-pending

Inquiries: info@coreform.com

U-splines: splines over unstructured meshes

Derek C. Thomas[a,*], Luke Engvall[a], Steven K. Schmidt[b,a], Kevin Tew[a], Michael A. Scott[b,a]

[a] *Coreform LLC, P.O. Box 970336, Orem, Utah 84097, USA*
[b] *Department of Civil and Environmental Engineering, Brigham Young University, Provo, UT 84602, USA*

Abstract

U-splines are a novel approach to the construction of a spline basis for representing smooth objects in Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE). A spline is a piecewise-defined function that satisfies continuity constraints between adjacent elements in a mesh. U-splines differ from existing spline constructions, such as Non-Uniform Rational B- splines (NURBS), subdivision surfaces, T-splines, and hierarchical B-splines, in that they can accommodate local geometrically exact adaptivity in h (element size), p (polynomial degree), and k (smoothness) simultaneously over more varied mesh topology. U-splines have no restrictions on the placement of T-junctions in the mesh. Mixed element meshes (e.g., triangle and quadrilateral elements in the same surface mesh) are also supported. We conjecture that the U- spline basis is positive, forms a partition of unity, is linearly independent, and provides optimal approximation when used in analysis.

Keywords: unstructured spline, unstructured mesh, computer-aided geometric design (CAGD), isogeometric analysis (IGA), computer-aided engineering (CAE), finite element analysis (FEA)

1. Introduction

1.1. Motivation

Splines are used extensively in computer-aided design (CAD) for the representation of shape. The nonuniform, rational B-spline (NURBS) underlies virtually all CAD systems. The NURBS construction is inherently limited as a single NURBS object can only represent deformations of a square. To overcome this limitation, the use of boundary representations (B-reps) based on trimmed B-splines has become standard. While the shortcomings of B-reps have long been recognized by CAD practitioners, the overall utility of the approach for design has been proven over the past decades.

Computer-based simulation is conceptually complementary to the CAD process as it can provide feedback regarding the expected behavior of a given part before costly fabrication is undertaken. The predominant simulation technique in current use is finite element analysis (FEA). However, the requirements of this analysis are dramatically different from the requirements of the design. Consequently, simulation of a design is typically preceded by a process known as meshing in which an approximation of the original CAD design is constructed in order to satisfy the requirements of the analysis pipeline. Inconsistencies in the original model must resolved. Since inconsistencies often include small gaps between adjacent faces in the model, the final mesh approximation that resolves these inconsistencies is referred to as "watertight". The generation of a clean, watertight mesh from a B-rep is notoriously difficult to automate. The analysis mesh is typically constructed using linear interpolation functions and therefore can only approximate the curved, smooth features of the original shape. This leads to the common practice of removing small features to avoid excessively dense meshes. This is known as defeaturing. The approximate nature of the analysis mesh and the intentional removal of features both affect the accuracy of the resulting analysis. These challenges are a significant barrier between CAD and analysis that restricts the utility of analysis in the design iteration process.

Although the potential power of splines has long been recognized by analysts [1], most splines other than simple C^0 splines were viewed as too expensive or the construction was too complex for use in general-purpose FEA. Efforts were made to improve the geometric definition in analysis through the use of subdivision surfaces [2] and NURBS in finite element analysis [3] among others but it was not until the introduction of the concept of isogeometric analysis (IGA) [4] that a large-scale effort to more closely integrate design and analysis commenced. The past decade has seen an explosion in research on the unification of geometry and analysis under the banner

*Corresponding author

of IGA. The potential benefits of this integration have become clear [5, 6, 7] but analysis of complex geometries remains a significant barrier to broader adoption. Promising work on conducting analysis directly on trimmed objects [8] has been carried out but the quality of the analysis can depend strongly on the choice of auxiliary parameters and on the relationship between the underlying parameteric basis and the trimming curve. Analysis efforts based on subdivision algorithms [9], T-splines [10], and other constructions were initiated early on in the literature but each failed to completely realize the full vision of IGA.

A true isogeometric representation for industrial-scale problems requires a spline technology that is capable of meeting the needs of both design and analysis. This representation must be defined over meshes of arbitrary topology, support the local modification of element size and interelement continuity, and provide a locally supported, positive basis that forms a partition of unity and is linearly independent.

1.2. Previous work

The work presented here has a shared heritage in both CAGD and FEA. The reader is referred to [11, 12] for a history of CAGD and to [13, 14, 15] for information on the history of FEA. Despite common goals of representing geometry for the purpose of computation, as stated previously, CAGD and CAE developed distinctly different approaches to the representation of shape. Rather than giving a full history, we attempt to provide a brief overview of publications directly related to the current work.

The need for unstructured surface representations in graphics led to the development of both subdivision surfaces [16] and T-splines [17] for use in computer graphics. A significant benefit of the T-spline construction is its compatibility with NURBS representations. Additional developments to follow on the advances of subdivision surfaces and T-splines include PHT-splines [18] and polynomials splines over T-meshes [19]. In these works, the continuity of the splines is restricted to be less than half of the polynomial degrees on adjacent elements. The importance of handling singular or extraordinary points smoothly has long been recognized and many approaches have been proposed but the work of Reif [20] and derived efforts [21, 22] are particularly relevant.

The need for smooth unstructured surfaces was recognized in the analysis community [23]. Despite these early efforts, the majority of finite element research was carried out on C^0 constructions and so finite element analysis came to be associate primarily with C^0 basis functions. More recently, subdivision surfaces were applied to shells by Cirak et al. [2]. Shortly after the original introduction of the IGA vision [4], work commenced on isogeometric analysis based on T-splines [10]. This was motivated by both the unstructured nature of T-splines and the need for adaptive local refinement. The need for guarantees on analysis properties of the basis led to the introduction of analysis-suitable T-splines [24]. Other efforts to produce refinable splines suitable for use in analysis followed. This included locally refined (LR) B-splines [25] hierarchical B-splines and truncate hierarchical B-splines [26, 27]. Constructions based on geometric rather than parametric continuity include the work of Groisser and Peters [28] and Kapl et al. [29].

There has also been significant work on new spline constructions over unstructured meshes within the numerical analysis community although most of these were not adopted in the context of IGA. Classic approaches commonly employed to produce continuity greater than C^0 include Arqyris elements [23], Clough-Tocher elements [30], and Powell-Sabin splines [31, 32] among others. Significant work has been carried out on the dimension of spline spaces for both triangle [33, 34] and T-meshes [35, 19]. Schumaker and Wang considered meshes consisting of both squares and triangles with potentially hanging vertices although they considered only splines of continuity C^0. Schumaker and Wang [37] proved the approxmation power of splines over T-meshes for splines of reduced continuity greater than C^0. Several types of simplex splines have been introduced to facilitate the construction of splines in unstructured settings. Neamtu [38] gives a very elegant construction that relies only on the placement of points, not on any mesh connectivity. As such, it cannot be applied to predefined meshes. Several adaptations of simplex splines to Powell-Sabin and other splits have been proposed to allow their use on unstructured meshes [39, 40, 41, 42]. Additional methods combine the solution of continuity constraints together with the solution of the governing PDEs [43, 44, 45]. Splines based on both triangles [46, 47, 48] and tetrahedra [49] have been employed in isogeometric analysis.

Mixed degree or multi-degree splines have not seen extensive use in isogeometric analysis although basic constructions are used in *hp*-adaptive methods [50]. In CAGD, univariate mixed degree or multi-degree splines were proposed by Sederberg et al. [51] with a basis given by Shen and Wang [52]. An alternate method for constructing a basis was given by Toshniwal et al. [53]. Multivariate multidegree splines with higher continuity have been constructed on triangulations without a basis for the purpose of numerical analysis [44].

1.3. Current work

Each of the prior technologies has provided important advances and served to demonstrate the power and utility of the isogeometric paradigm; however each has also been unable to achieve the flexibility required for

a full integration of design and analysis. Each has different limitations in the level of continuity enabled, the placement of local refinement features, the polynomial degree supported, or the quality of the basis. To this end we present U-splines, a technology for the construction of spline basis functions over unstructured meshes. An innovation underlying U-splines is a method for solving a series of highly localized nullspace problems of size bounded by the local characteristics of the basis chosen for each element, the local mesh topology, and the associated smoothness constraints. Each local nullspace problem is solved to determine exactly one spline basis function. The U-spline algorithm guarantees that each U-spline basis function is positive and that the resulting basis satisfies a partition of unity. The algorithm is expressed entirely in terms of integers and requires no floating point operations until the indices of the nonzero coefficients of a U-spline basis function have been determined. A novel indexing scheme is used to uniquely identify each U-spline basis function in the mesh and to construct the associated control mesh. The only requirement for the initialization of the algorithm is that a Bernstein-like basis be defined over each element in the mesh. A mixture of standard polynomial Bernstein bases over cuboidal and simplicial (triangular) elements can be used in addition to more exotic Bernstein-like bases based on exponential, trigonometric, and other special functions.

The use of U-splines in commercial CAD and CAE implementations can improve the quality and flexibility of shape representation, the accuracy, robustness, and efficiency of simulation, and create fully integrated isogeometric analysis (IGA) workflows that may significantly reduce the significant time spent in translating data between CAD and CAE representations in aerospace, automotive, defense, and other industries. The unique U-spline basis construction process generates a minimal number of control points or degrees of freedom required for a given application. This may also offer unique benefits for emerging applications like topology optimization, generative design, and additive manufacturing.

We note that the name U-spline was originally used to refer to the definition of splines over unordered knot sequences [54]. Because the need for unstructured splines is significant and the application of splines over unordered knot sequences has not yet achieved widespread use, we instead use the U-spline designation for our splines over unstructured meshes.

2. Bernstein representations

2.1. Polynomial basis

The univariate Bernstein basis is defined as

$$B_i^p(s) = \binom{p}{i} s^i (1-s)^{p-i}, \tag{1}$$

where p is the polynomial degree, $s \in [0,1]$, and the binomial coefficient is $\binom{p}{i} = \frac{p!}{(i)!(p-i)!}$, $0 \leq i \leq p$. The Bernstein basis possesses many remarkable properties but the primary property of interest for this work is the ordering of the derivatives of the basis functions. A function $f : \mathbb{R} \to \mathbb{R}$ vanishes n times at a real value a if $f(a) = 0$ and $f^{(i)}(a) = 0$ for all $i \in [0, n]$. The ith Bernstein basis function B_i^p on the interval $[0,1]$ vanishes i times at 0 and $p-i$ times at 1. This property can be observed in Table 1 where the evaluations of the Bernstein basis and its derivatives are shown at the endpoints of the interval $[0,1]$.

Two constructions for the multivariate setting are employed. One construction is the tensor-product of multiple univariate Bernstein bases:

$$B_{\mathbf{i}}^{\mathbf{p}}(\mathbf{s}) = \prod_{j=0}^{d} B_{i_j}^{p_j}(s_j). \tag{2}$$

This is defined over box-like elements. In this work, bold-face symbols are used to represent tuples, vectors and matrices. The associated italic symbol with subscripts may be used to refer to individual components: the symbol $\mathbf{i}$ represents an index tuple, i_j refers to the jth entry in the tuple. For complex objects, square brackets followed by subscripts may be used to refer to components: $[\mathbf{i}]_j \equiv i_j$.

For example, the tensor-product Bernstein function with index $\mathbf{i} = (1,2)$, degree $\mathbf{p} = (3,2)$ is defined as

$$B_{(1,2)}^{(3,2)}(s,t) = B_1^3(s) B_2^2(t) \tag{3}$$

The other construction is the the multivariate Bernstein polynomial basis defined over simplicial elements. For a simplex of dimension d, the basis of degree p is defined in terms of the barycentric coordinates λ_i, $i \in [0, d]$

Table 1: Derivatives of Bernstein polynomial $\frac{d^n}{ds^n} B_i^p(s)$ evaluated at the endpoints of the interval.

			$i=0$	$i=1$	$i=2$	$i=3$
$p=1$	$n=0$	$s=0$	1	0	-	-
		$s=1$	0	1	-	-
	$n=1$	$s=0$	-1	1	-	-
		$s=1$	-1	1	-	-
$p=2$	$n=0$	$s=0$	1	0	0	-
		$s=1$	0	0	1	-
	$n=1$	$s=0$	-2	2	0	-
		$s=1$	0	-2	2	-
	$n=2$	$s=0$	2	-4	2	-
		$s=1$	2	-4	2	-
$p=3$	$n=0$	$s=0$	1	0	0	0
		$s=1$	0	0	0	1
	$n=1$	$s=0$	-3	3	0	0
		$s=1$	0	0	-3	3
	$n=2$	$s=0$	6	-12	6	0
		$s=1$	0	6	-12	6
	$n=3$	$s=0$	-6	18	-18	6
		$s=1$	-6	18	18	6

and an index tuple $\boldsymbol{\alpha}$ of size $d+1$ of positive integers whose sum is equal to p. The barycentric coordinates are defined in the usual fashion with $\lambda_d = 1 - \sum_{i=0}^{d-1} \lambda_i$. The $\boldsymbol{\alpha}$-th basis function is given by

$$B_{\boldsymbol{\alpha}}^p(\boldsymbol{\lambda}) = p! \prod_{i=0}^{d} \frac{\lambda_i^{\alpha_i}}{\alpha_i!}. \tag{4}$$

For example, the two dimensional Bernstein polynomial of degree $p=6$ with index $\boldsymbol{\alpha} = (1,2,3)$ is given by

$$\begin{aligned} B_{(1,2,3)}^3(\lambda_0,\lambda_1,\lambda_2) &= 6! \frac{(\lambda_0)^1}{1!} \frac{(\lambda_1)^2}{2!} \frac{(\lambda_2)^3}{3!} \\ &= 60\lambda_0(\lambda_1)^2(1-\lambda_0-\lambda_1)^3. \end{aligned} \tag{5}$$

The multivariate Bernstein basis functions possess similar ordering properties to the univariate basis. Additionally, for each boundary of the simplex, the nonzero entries in the basis are precisely the basis for the simplex of dimension $d-1$.

2.2. Bernstein-like basis

Although the focus is primarily on the polynomial Bernstein basis in this work, this is not a necessary requirement. Mazure proved that quasi extended Chebyshev (QEC) spaces possess a Bernstein-like basis with the following property: Let $\mathcal{E}$ be an $(n+1)$-dimensional QEC-space on the bounded closed interval $[a,b]$. Then, $\mathcal{E}$ possesses a quasi Bernstein-like basis relative to (a,b), that is, a basis $B_0,\ldots,B_n$ such that:

- $B_0(a) \neq 0$, and B_0 vanishes n times at b; $B_n(b) \neq 0$, and B_n vanishes n times at a;
- for $1 \leq i \leq n-1$, B_i vanishes exactly i times at a and exactly $(n-i)$ times at b.
- for $0 \leq i \leq n$, B_i is positive on $]a,b[$.

This property is the key requirement for the U-spline definition and construction and so U-splines can be constructed from meshes with a QEC space assigned to each element.

2.3. Change of basis

The Bernstein representation admits many closed-form expressions for the change of basis [55]. Here several relations for Bernstein polynomials are given. Similar results can be obtained for other Bernstein-like bases or computed directly.

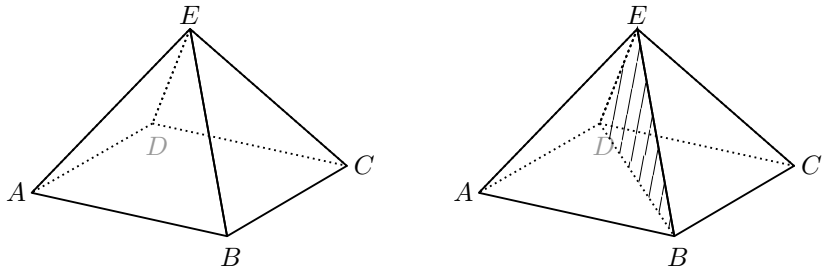

Figure 1: Subdivision of the pyramid with base $ABCD$ and elevated point E to produce two tetrahedra $ABDE$ and $BCDE$.

The coefficients $\mathbf{b}$ of a Bernstein polynomial of degree p may be converted to coefficients $\bar{\mathbf{b}}$ of a Bernstein polynomial of degree $p+r$ by multiplication by a matrix:

$$\bar{\mathbf{b}} = \mathbf{E}^{p,r}\mathbf{b} \tag{6}$$

where the nonzero entries of the matrix are given by

$$E_{ij}^{p,r} = \frac{\binom{r}{i-j}\binom{p}{j}}{\binom{p+r}{i}}, \quad \max(0, i-r) \leq j \leq \min(p, i) \tag{7}$$

Similarly, the coefficients $\bar{\mathbf{b}}$ of the Bernstein basis over some interval (s_0, s_1) can be obtained from the coefficients $\mathbf{b}$ of the Bernstein basis over the standard unit interval by matrix multiplication:

$$\bar{\mathbf{b}} = \mathbf{R}^{s_0,s_1}\mathbf{b} \tag{8}$$

where the nonzero entries of the matrix are

$$R_{jk}^{s_0,s_1} = \sum_{i=\max(0,j+k-p)}^{\min(j,k)} B_{k-i}^{p-j}(s_0)B_i^j(s_1), j \leq n, 0 \leq k. \tag{9}$$

3. The Bézier mesh

3.1. Topology and parameterization

Unstructured splines, or U-splines, can be defined over unstructured meshes constructed entirely of elements that permit either tensor-product or simplicial parameterization. T-junctions are allowed to occur in the mesh. Each element is assigned a local parametric coordinate system. This system is assumed to be Cartesian on box-like elements and the parametric coordinates are denoted by either s, t, u or s_i. Barycentric coordinates are employed on simplicial elements and the symbols λ_i are used. It should be noted that for three-dimensional meshes this approach permits pentahedral prisms produced by a tensor product of a triangle with a line but does not permit pyramid elements. However, any pyramid element can be replaced by two tetrahedra. This is illustrated in Figure 1 wherein a pyramid is subdivided into two tetrahedra.

Each element is also assigned parametric dimensions. The most general description possible for the parametric size of each element is adopted. Each element/edge pair in the mesh is assigned a parametric length. The parametric dimensions of the boundary of an element must be consistent. For a box-like element, this means that opposite faces must have the same size in the local parameterization of the element. Adjacent elements must also satisfy a cycle condition: for each vertex in the mesh, the product of the ratios of adjacent edge lengths on a subset of the edges emanating from the vertex traversed in a oriented closed loop must be equal to one. This corresponds to fixing a seamless similarity map between adjacent elements. This approach was first introduced for T-splines over conformal seamless similarity maps by Campen and Zorin [56].

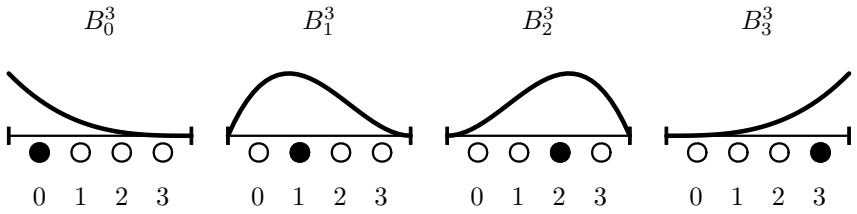

Figure 2: Univariate cubic Bernstein basis shown with the adopted indexing convention and corresponding circular markers. Indices corresponding to nonzero coefficient values of the basis are indicated with filled markers.

3.2. Element basis and interelement continuity

Each element in the mesh is assigned a basis. The symbol e is used to denote an element index. For polynomial splines, this amounts to assigning a polynomial degree p^e to each simplicial element and a tuple of polynomial degrees $\mathbf{p}^e$ to each tensor-product element (one for each parametric direction). If more general Bernstein-like bases are to be employed then these must be assigned to each element. The symbol $\mathbf{B}^e$ is used to denote the set of Bernstein basis functions assigned to an element e. The set of all Bernstein functions over all the elements in a mesh M is denoted by $\mathbf{B}^M$.

Each interface between elements in the mesh is also assigned a required minimum continuity that represents the minimum number of derivatives that are continuous perpendicular to the interface. Given an interface I, let $k(I)$ represent the minimum order of continuity of the interface. Note that for certain mesh configurations, the U-spline basis may be smoother than the specified continuity conditions.

3.3. Element basis indexing

Once an element in the mesh has been assigned a Bernstein-like basis, the individual basis functions on each element can be uniquely identified in order to impose the required continuity constraints.

The Bernstein basis admits a natural indexing. This indexing is most naturally expressed in terms of the tuples used in the definition of the polynomial Bernstein basis. A similar indexing exists for Bernstein-like bases of QEC spaces.

A univariate Bernstein basis is indexed by one number, a tensor-product Bernstein basis is indexed by a tuple of size d (one number for each direction), and a simplex is indexed by a tuple of size $d+1$. This is called the Bernstein index. The index set of the Bernstein basis set $\mathbf{B}$ is denoted by $\mathbf{I}(\mathbf{B})$.

For example, if the set of Bernstein basis functions is the univariate cubic Bernstein polynomials,

$$\mathbf{B} = \{B_0^3, B_1^3, B_2^3, B_3^3\}, \tag{10}$$

then the index set is

$$\mathbf{I}(\mathbf{B}) = \{0, 1, 2, 3\}. \tag{11}$$

This basis is shown in Figure 2. The indexing is shown beneath the plot with the marker corresponding to the plotted function being filled. In most diagrams that follow, the numeric indices are suppressed and dots are used to indicate the degree of the basis employed. Indices corresponding to nonzero coefficients are filled.

If the set of Bernstein basis functions is the tensor-product Bernstein polynomials of mixed degree $\mathbf{p} = (1, 2)$,

$$\mathbf{B} = \{B_{(0,0)}^{(1,2)}, B_{(1,0)}^{(1,2)}, B_{(0,1)}^{(1,2)}, B_{(1,1)}^{(1,2)}, B_{(0,2)}^{(1,2)}, B_{(1,2)}^{(1,2)}\}, \tag{12}$$

then the index set is

$$\mathbf{I}(\mathbf{B}) = \{(0,0), (1,0), (0,1), (1,1), (0,2), (1,2)\}. \tag{13}$$

The indices can be used as coordinates and markers placed at each of the corresponding locations in the element can be used to indicate the number of basis functions on each element. This is shown for a basis with polynomial degrees $\mathbf{p} = (2, 3)$. Rather than including explicit numeric values for the indices, markers can be used. Both numeric values and the corresponding markers are shown in Figure 3.

If the set of Bernstein basis functions is the bivariate simplicial Bernstein polynomials of degree $p = 2$,

$$\mathbf{B} = \{B_{(0,0,2)}^2, B_{(1,0,1)}^2, B_{(2,0,0)}^2, B_{(0,1,1)}^2, B_{(0,2,0)}^2, B_{(1,1,0)}^2\}, \tag{14}$$

then the index set is

$$\mathbf{I}(\mathbf{B}) = \{(0,0,2), (1,0,1), (2,0,0), (0,1,1), (0,2,0), (1,1,0)\}. \tag{15}$$

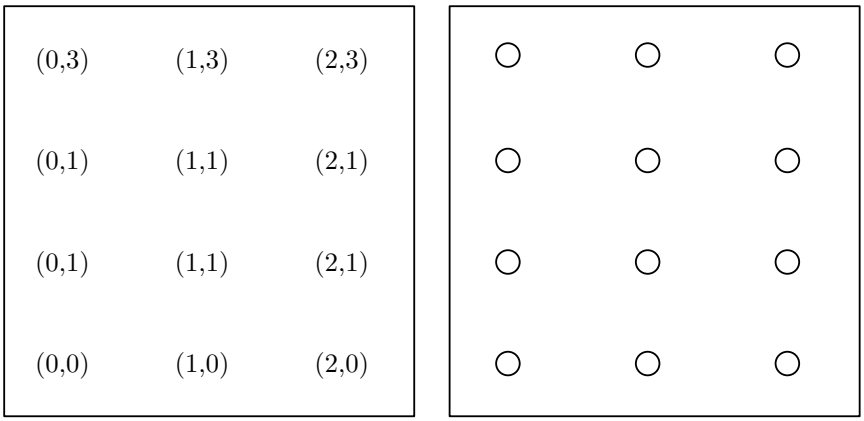

Figure 3: Bernstein index and simplified graphical representations.

If the basis used on each element has been specified, each function can be uniquely identified by the element index and the local Bernstein index. The element and Bernstein indices can be combined to produce a global Bernstein index $(e, \mathbf{i})$. This unambiguously identifies the Bernstein basis function $B_{e,\mathbf{i}}$. The polynomial degree p can vary from element to element and so is omitted. Index sets over a specific element or sets of elements are assumed to include the element and local Bernstein indices; e.g., for the basis $\mathbf{B}^e$ associated with element e, having polynomial degree $\mathbf{p}^e = (1, 2)$, the index set is

$$\mathbf{I}(\mathbf{B}^e) = \{(e,(0,0)),(e,(1,0)),(e,(0,1)),(e,(1,1)),(e,(0,2)),(e,(1,2))\}. \tag{16}$$

For two bilinear elements, e and e', the index set is

$$\begin{aligned}\mathbf{I}(\mathbf{B}^e \cup \mathbf{B}^{e'}) = \{&(e,(0,0)),(e,(1,0)),(e,(0,1)),(e,(1,1)),\\ &(e',(0,0)),(e',(1,0)),(e',(0,1)),(e',(1,1))\}.\end{aligned} \tag{17}$$

When the meaning is clear from the context, the explicit element index will be suppressed and bold symbols will be used instead to refer to global indices containing both element index and the local Bernstein index.

3.4. Element index distances

The Bernstein indices on an element form a discrete finite-dimensional space. It is useful to define the difference vector:

$$\boldsymbol{\delta}(\mathbf{i}, \mathbf{j}) = \mathbf{j} - \mathbf{i} \tag{18}$$

and the distance vector d whose entries are the absolute value of the entries of the difference vector:

$$d_k(i_k, j_k) = |\delta(i_k, j_k)|. \tag{19}$$

A function that provides the distances between an index and the element boundaries is needed. To indicate direction the parametric index and the sign in the associated direction $d_i^{\pm}$ can be specified. For tensor-product elements,

$$d_j^{\sigma}(\mathbf{i}) = \begin{cases} p_j - i_j, & \sigma = + \\ i_j, & \sigma = - \end{cases} \tag{20}$$

where $\mathbf{p} = (p_0, \ldots, p_{d-1})$ represents the polynomial degree (or one less than the total number of Bernstein basis functions in the corresponding dimension for Bernstein-like bases).

For example, consider a tensor product element of degree $\mathbf{p} = (2, 3)$ and a Bernstein index $\mathbf{i} = (1, 3)$. The output of the distance function is:

$$\begin{aligned} d_0^+((1,3)) &= 1 \\ d_1^+((1,3)) &= 0 \\ d_0^-((1,3)) &= 1 \\ d_1^-((1,3)) &= 3. \end{aligned}$$

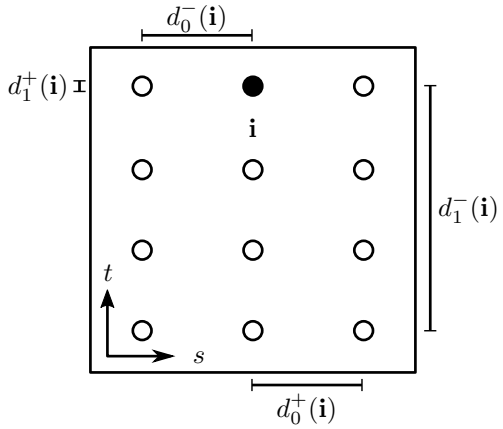

Figure 4: An example of element index distances for a given element.

This set of distances is illustrated in Figure 4.

The distance to a specified boundary within an element can also be determined. For an element e bounded by an interface I the distance from the index $\mathbf{i}$ in element e to the boundary I is denoted by $d_I(\mathbf{i})$.

3.5. Bernstein-Bézier form

Having defined a mesh with a basis defined over each element, any piecewise function that lies in the function space of each element in the mesh can be represented in terms of the local basis on each element. This is accomplished by assigning a coefficient to each basis function on each element. This is known as the Bernstein-Bézier form of the function. These coefficients are indexed by the same indices defined for the Bernstein basis: $b_{\mathbf{i}}$, where the index $\mathbf{i}$ is a global index. Thus, a function f over a mesh M is given by

$$f = \sum_{\mathbf{i} \in \mathbf{I}(\mathbf{B}^M)} b_{\mathbf{i}} B_{\mathbf{i}}. \tag{21}$$

4. Continuity constraints and splines

A spline is defined as a piecewise function over a given mesh that satisfies continuity constraints on every interface in the mesh. Splines are often expressed in Bernstein-Bézier form where the vector of Bernstein coefficients that define the spline satisfy the set of continuity constraints at every interface in the mesh.

4.1. Continuity constraints

Continuity constraints on functions expressed in Bernstein-Bézier form are now considered. A function over a mesh is said to have continuity of order k or be C^k at an interface if all derivatives of order less than or equal to k are continuous between the elements on either side of the interface. A key property of a Bernstein-like basis is that only the coefficients nearest the interface participate in a constraint. This is illustrated for a univariate case in Figure 5. These continuity constraints can be expressed in terms of the coefficients of the Bernstein-Bézier form. In general, each continuity constraint for the interface between an element e and a neighbor e', sharing an interface I, is expressed as a linear system involving only the Bernstein coefficients having indices within a distance k of the interface:

$$\sum_{\substack{\mathbf{i} \in \mathbf{I}(\mathbf{B}^e) \\ d_I(\mathbf{i}) \leq k}} c_{\mathbf{i}}^{I,1} b_{\mathbf{i}} = \sum_{\substack{\mathbf{j} \in \mathbf{I}(\mathbf{B}^{e'}) \\ d_I(\mathbf{j}) \leq k}} c_{\mathbf{j}}^{I,1} b_{\mathbf{j}} \tag{22}$$

$$\vdots$$

$$\sum_{\substack{\mathbf{i} \in \mathbf{I}(\mathbf{B}^e) \\ d_I(\mathbf{i}) \leq k}} c_{\mathbf{i}}^{I,q} b_{\mathbf{i}} = \sum_{\substack{\mathbf{j} \in \mathbf{I}(\mathbf{B}^{e'}) \\ d_I(\mathbf{j}) \leq k}} c_{\mathbf{j}}^{I,q} b_{\mathbf{j}}. \tag{23}$$

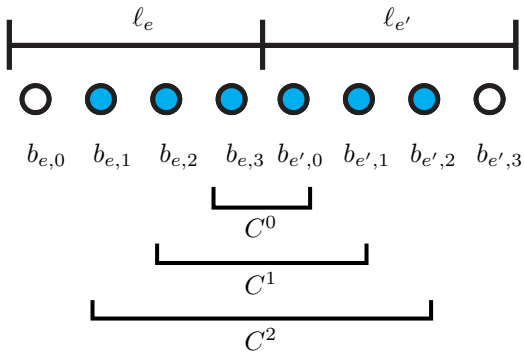

Figure 5: The coefficients that may participate in a C^2 constraint are indicated by the filled index sites. The indices of the coefficients required for each order of continuity are marked with labeled brackets.

The number of constraints q is dependent on the basis on each element adjacent to the interface and the order of continuity required at the interface. The values of the constraint coefficients $c_{\mathbf{i}}^{I,f}$ are specific to the local Bernstein basis on each element. Smoothness constraints can also be expressed in matrix-vector format

$$\mathbf{S}_e^I \mathbf{b}_e = \mathbf{S}_{e'}^I \mathbf{b}_{e'} \tag{24}$$

where $\mathbf{S}_e^I$ is a $q \times n_e$ matrix; n_e is the number of basis functions on element e. The entries of the $\mathbf{S}^I$ matrices are the c coefficients. This matrix equation can be written in nullspace form:

$$\left[\mathbf{S}_e^I \,|\, -\mathbf{S}_{e'}^I\right] \left[\frac{\mathbf{b}_e}{\mathbf{b}_{e'}}\right] = \mathbf{0}. \tag{25}$$

The continuity constraints from all interfaces in the mesh M can be collected in a global nullspace expression:

$$\mathbf{S}^M \mathbf{b} = \mathbf{0}. \tag{26}$$

The vector $\mathbf{b}$ represents the Bernstein coefficients of every element in the entire mesh. The Bernstein coefficients of any function that satisfies the smoothness constraints must lie in the nullspace of the smoothness matrix $\mathbf{S}^M$.

4.2. Univariate constraints

One may begin with continuity constraints on functions expressed in terms of univariate Bernstein bases. If the elements are placed so that the interface I lies at the end of element e and at the beginning of element e', then the constraint coefficients for the constraint of order k are given by

$$c_{\mathbf{i}} = \left.\frac{d^k B_{\mathbf{i}}}{ds^k}\right|_{s=\ell_e} \tag{27}$$

$$c_{\mathbf{i}'} = \rho_{e'}^e \left.\frac{d^k B_{\mathbf{i}'}}{ds^k}\right|_{s=0} \tag{28}$$

where $\rho_{e'}^e = \ell_e/\ell_{e'}$ and ℓ_e and $\ell_{e'}$ are the parametric lengths of the elements e and e', respectively. For example, any function in Bernstein-Bézier form spanning an interface of continuity $k = 2$, bounded by elements having a Bernstein basis of degree 3, must satisfy the following constraints on its coefficients:

$$b_{e,3} = b_{e',0} \tag{29}$$

$$3\left(b_{e,2} - b_{e,3}\right) = 3\rho_{e'}^e \left(b_{e',0} - b_{e',1}\right) \tag{30}$$

$$6\left(b_{e,1} - 2b_{e,2} + b_{e,3}\right) = 6(\rho_{e'}^e)^2 \left(b_{e',0} - 2b_{e',1} + b_{e',2}\right) \tag{31}$$

where $\rho = \frac{\ell_e}{\ell_{e'}}$. These constraints can be expressed in matrix form as

$$\mathbf{S}_e \mathbf{b}_e = \mathbf{S}_{e'} \mathbf{b}_{e'} \tag{32}$$

where

$$\mathbf{S}_e = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 3 & -3 \\ 0 & 6 & -12 & 6 \end{bmatrix} \tag{33}$$

and

$$\mathbf{S}_{e'} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3(\rho^e_{e'}) & -3(\rho^e_{e'}) & 0 & 0 \\ 6(\rho^e_{e'})^2 & -12(\rho^e_{e'})^2 & 6(\rho^e_{e'})^2 & 0 \end{bmatrix}. \tag{34}$$

These matrices can be combined into a single matrix

$$\mathbf{S} = [\mathbf{S}_e \,|\, -\mathbf{S}_{e'}] = \begin{bmatrix} 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 3 & -3 & -3(\rho^e_{e'}) & 3(\rho^e_{e'}) & 0 & 0 \\ 0 & 6 & -12 & 6 & -6(\rho^e_{e'})^2 & 12(\rho^e_{e'})^2 & -6(\rho^e_{e'})^2 & 0 \end{bmatrix} \tag{35}$$

and the coefficients into a single vector

$$\mathbf{b} = \left[\frac{\mathbf{b}_e}{\mathbf{b}_{e'}}\right] = \begin{bmatrix} b_{e,0} & b_{e,1} & b_{e,2} & b_{e,3} & b_{e',0} & b_{e',1} & b_{e',2} & b_{e',3} \end{bmatrix}^T \tag{36}$$

It is not necessary to restrict the method to the configurations having the same Bernstein space on each element of the mesh. For example, the smoothness matrices and coefficient vector for an interface of continuity C^1 where the polynomial degree of the element e is 2 rather than 3 are

$$\mathbf{S}_e = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & -2 \end{bmatrix}, \tag{37}$$

$$\mathbf{S}_{e'} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3(\rho^e_{e'}) & -3(\rho^e_{e'}) & 0 & 0 \end{bmatrix}, \tag{38}$$

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 2 & -2 & -3(\rho^e_{e'}) & 3(\rho^e_{e'}) & 0 & 0 \end{bmatrix}, \tag{39}$$

$$\mathbf{b} = \begin{bmatrix} b_{e,0} & b_{e,1} & b_{e,2} & b_{e',0} & b_{e',1} & b_{e',2} & b_{e',3} \end{bmatrix}^T. \tag{40}$$

4.3. A general approach to multivariate constraints

Although the expressions for multivariate constraints vary based on the element types on either side of the interface (tensor-product or simplicial), a common process may be applied. For two elements e and e' that share an interface I, the trace spaces of the Bernstein spaces on each element restricted to the interface I is denoted by $\mathcal{B}^e|_I$ and $\mathcal{B}^{e'}|_I$. The interface space $\mathcal{I}$ is defined to be the smallest Bernstein-like space of dimension $d-1$ that contains both trace spaces. In other words, $\mathcal{B}^{e'}|_I \cup \mathcal{B}^{e'}|_I \subseteq \mathcal{I}$. On each element, a superspace, $\bar{\mathcal{B}}^e$, is introduced such that $\mathcal{B}^e \subseteq \bar{\mathcal{B}}^e$ and $\mathcal{I} \subseteq \bar{\mathcal{B}}^e|_I$ and an associated transformation matrix $\mathbf{M}_e : \mathcal{B}^e \to \bar{\mathcal{B}}^e$ is determined. Several examples of transformation matrices for common cases are given in Section 2.3. The continuity constraints are then imposed on the coefficients in the superspace through the application of $\mathbf{M}_e$. To be more precise, the system of equations that constrain the Bernstein coefficients across an interface can be written as

$$\bar{\mathbf{S}}^I_e \mathbf{M}^I_e \mathbf{b}_e = \bar{\mathbf{S}}^I_{e'} \mathbf{M}^I_{e'} \mathbf{b}_{e'} \tag{41}$$

where the matrices $\bar{\mathbf{S}}^I_e$ and $\bar{\mathbf{S}}^I_{e'}$ are the constraint matrices across the interface I whose coefficients are written in terms of the superspace on each element.

It is important to note that any continuity requirement between two adjacent elements will reduce the local function space at the interface to the intersection of trace spaces of the two adjacent elements. This means that Bernstein spaces that share only constants at the interface will only be able to represent constants. This is primarily of concern for elements with basis functions drawn from QEC spaces. Polynomial bases will always be able to represent polynomials of the same degree as the element of lowest degree.

4.4. Constraints between tensor-product elements

The tensor-product basis can naturally be separated into trace and perpendicular function spaces. The base case occurs when both elements share the same trace space. In this case, each line of coefficients perpendicular to the interface must satisfy the univariate constraints presented previously. All other cases may be put in this form by first performing suitable subdivision and degree elevation operations in the directions parallel to the interface as described in Section 2.3.

The operators required for the multivariate interface constraints can be constructed by Kronecker products of the appropriate univariate matrices in each parametric direction. As an example, if the elements e and e' share an interface, as shown in part (a) of Figure 6, then the transformation matrices $\mathbf{M}_e$ and $\mathbf{M}_{e'}$ are given by:

$$\mathbf{M}_e^I = \mathbf{R}^{0,a} \otimes \mathbf{I}_3, \tag{42}$$

$$\mathbf{M}_{e'}^I = \mathbf{I}_4 \otimes (\mathbf{E}^{2,1}\mathbf{R}^{0,b}) \tag{43}$$

where a and b are the lengths of the interface in the local coordinate system of each element, $\mathbf{I}_d$ is the identity matrix of dimension d, and the matrices $\mathbf{E}$ and $\mathbf{R}$ are defined in Equations (6) and (9). If a and b each occur at $3/4$ along the side of their respective elements, then the matrices are:

$$\mathbf{R}^{0,a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 \\ 1/16 & 3/8 & 9/16 & 0 \\ 1/64 & 9/64 & 27/64 & 27/64 \end{bmatrix}, \tag{44}$$

$$\mathbf{R}^{0,b} = \begin{bmatrix} 1 & 0 & 0 \\ 1/4 & 3/4 & 0 \\ 1/16 & 3/8 & 9/16 \end{bmatrix}, \tag{45}$$

$$\mathbf{E}^{2,1} = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 2/3 & 0 \\ 0 & 2/3 & 1/3 \\ 0 & 0 & 1 \end{bmatrix}. \tag{46}$$

The final constraint matrices of superspace basis coefficients are found by computing the constraints required to constrain the elements shown in part (b) of Figure 6 to satisfy C^1 continuity. Using the univariate constraint matrices

$$\mathbf{S}_e = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & -2 \end{bmatrix}, \tag{47}$$

$$\mathbf{S}_{e'} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3(\rho_{e'}^e) & -3(\rho_{e'}^e) & 0 & 0 \end{bmatrix} \tag{48}$$

where the ratio of element lengths $\rho_{e'}^e$ is the ratio of the perpendicular lengths of the elements, the final constraint matrices are

$$\bar{\mathbf{S}}_e^I = \mathbf{S}_e \otimes \mathbf{I}_3, \tag{49}$$

$$\bar{\mathbf{S}}_{e'}^I = \mathbf{J}_4 \otimes \mathbf{S}_{e'} \tag{50}$$

where $\mathbf{J}_4$ is the exchange matrix:

$$\mathbf{J}_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{51}$$

The exchange matrix is required because the indexing of element e' along the interface runs opposite the indexing of element e.

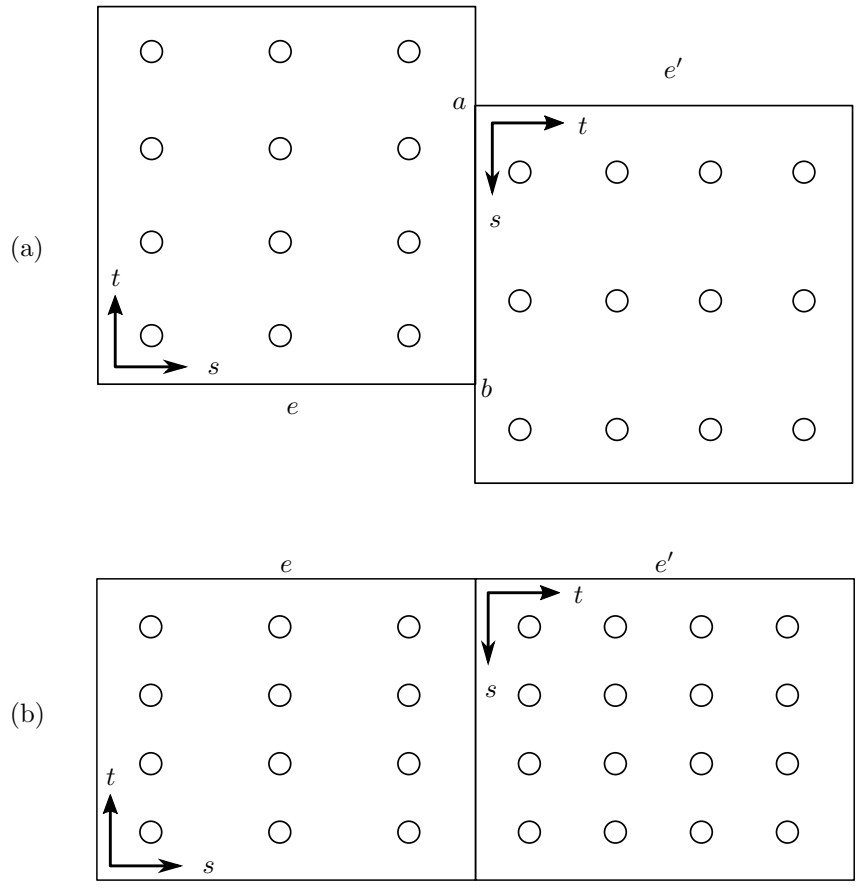

Figure 6: Example of the application of constraints for nonmatching tensor-product elements.

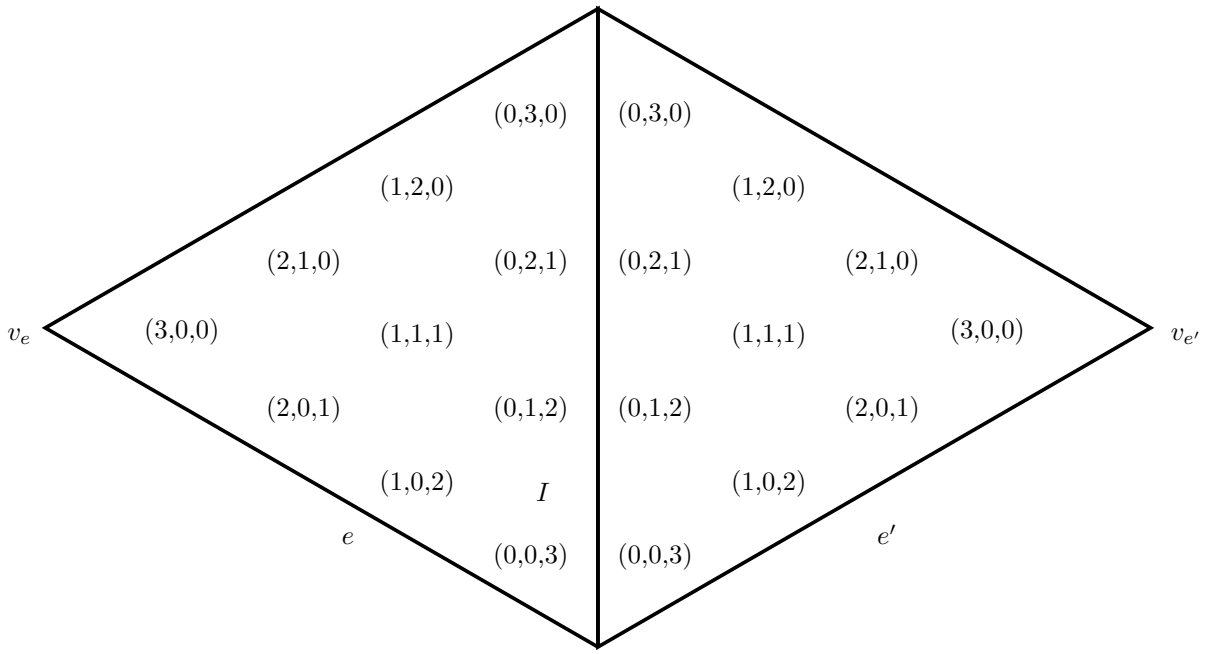

Figure 7: Example of the indexing used to define simplicial continuity constraints.

4.5. Constraints between simplicial elements

The continuity constraints between simplicial elements have a simple expression in terms of the barycentric coordinate of the opposite vertex of one simplex expressed in terms of the other. Without loss of generality, the interface I between the elements e and e' is assumed to be opposite the vertex with barycentric coordinate $(1, 0, \ldots, 0)$ in both elements. These vertices are denoted v_e and $v_{e'}$. Let $\lambda^e_{e'}$ be the barycentric location of the vertex $v_{e'}$ in the barycentric coordinate system of element e. This convention is illustrated in Figure 7. Introducing the notation $\boldsymbol{\alpha}_0 = (0, \alpha_1, \ldots, \alpha_{d+1})$ and $\boldsymbol{\alpha}_m = (m, \alpha_1, \ldots, \alpha_{d+1})$ the constraints of order k can be written as

$$b_{e,\boldsymbol{\alpha}_m} = \sum_{\|\boldsymbol{\mu}\|_1 = m} b_{e',\boldsymbol{\alpha}_0+\boldsymbol{\mu}} B^{m,e}_{\boldsymbol{\mu}}(\boldsymbol{\lambda}^e_{e'}) \tag{52}$$

for $0 \leq m \leq k$. The coordinates $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_m$ satisfy $\|\boldsymbol{\alpha}_m\|_1 = p$ and $\|\boldsymbol{\alpha}_0\|_1 = p - m$. The C^0 and C^1 constraints on the coefficients for the example shown in Figure 7 are

$$b_{e,(0,0,3)} = b_{e',(0,0,3)}, \tag{53}$$
$$b_{e,(0,1,2)} = b_{e',(0,1,2)}, \tag{54}$$
$$b_{e,(0,2,1)} = b_{e',(0,2,1)}, \tag{55}$$
$$b_{e,(0,3,0)} = b_{e',(0,3,0)}, \tag{56}$$
$$b_{e,(1,0,2)} = b_{e',(0,0,3)} + b_{e',(0,1,2)} - b_{e',(1,0,2)}, \tag{57}$$
$$b_{e,(1,1,1)} = b_{e',(0,1,2)} + b_{e',(0,2,1)} - b_{e',(1,1,1)}, \tag{58}$$
$$b_{e,(1,2,0)} = b_{e',(0,2,1)} + b_{e',(0,3,0)} - b_{e',(1,2,0)}. \tag{59}$$

Constraints between simplicial elements having differing polynomial degrees can be handled similarly to the tensor-product case where a transformation matrix from the lower degree to the higher degree coefficients is computed and used to transfer the constraints. This was considered by Hu et al. [44]. Refinement is handled in a similar fashion to the tensor-product case where a refined basis is constructed and the constraints are imposed on the refined coefficients and then transferred to the original coefficients.

4.6. Constraints between elements of differing type

In order to compute constraints between elements of differing type (simplicial and cuboidal), a simplicial subdomain T of the cuboidal element that shares the interface with the adjacent simplicial element is chosen. An

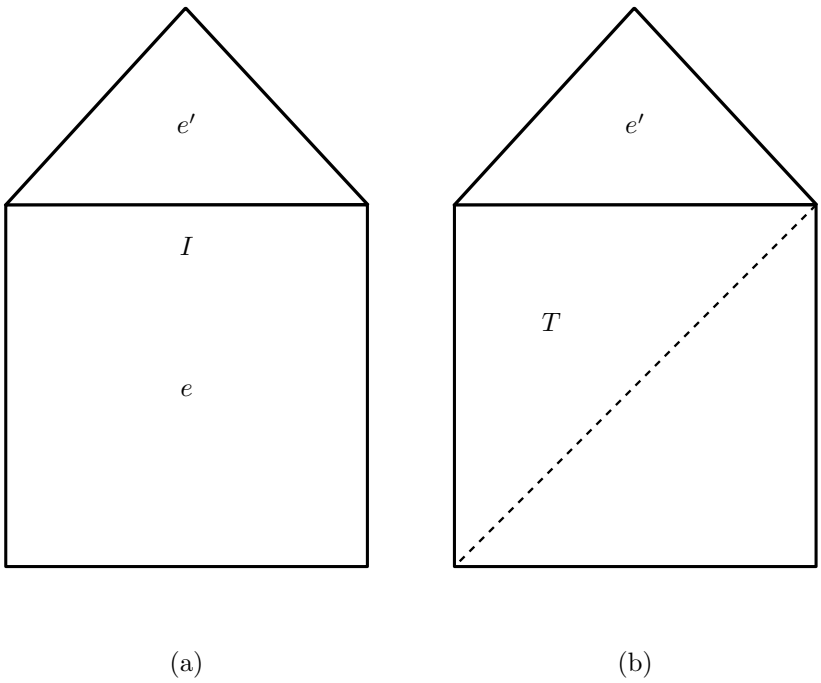

Figure 8: Diagram showing the regions used to define constraints between simplicial and cuboidal elements.

auxiliary basis capable of representing the tensor-product basis is defined over this domain and the transformation matrix $\mathbf{T}$ that expresses coefficients of the tensor-product basis in terms of the simplicial basis is computed. If the element e is cuboidal then the constraint equation Equation (41) becomes

$$\bar{\mathbf{S}}_T^I \mathbf{M}_T^I \mathbf{T} \mathbf{b}_e = \bar{\mathbf{S}}_{e'}^I \mathbf{M}_{e'}^I \mathbf{b}_{e'}. \tag{60}$$

An example of this is shown in Figure 8. The constraints across the interface I between the quadrilateral element e and the triangular element e' shown in part (a) are computed by introducing the auxiliary subdomain T shown in part (b).

4.7. Splines as a solution to a nullspace problem

The Bernstein coefficients that define a spline must lie in the nullspace of the mesh smoothness constraint matrix $\mathbf{S}^M$:

$$\mathbf{S}^M \mathbf{b} = \mathbf{0}. \tag{61}$$

The matrix $\mathbf{S}^M$ is produced by assembling the smoothness constraint matrices from every interface into one global matrix. A spline space $\mathcal{S}^M$ is the function space consisting of all splines over a given mesh M with a local basis assigned to each element in the mesh and smoothness conditions assigned to each interface in the mesh.

One significant area of research in the theory of splines is the determination of the dimension of a spline space from a mesh and assigned smoothness constraints. Because the Bernstein coefficient vectors of all functions in a spline space lie in the nullspace of the mesh smoothness constraint matrix, the dimension of the spline space can theoretically be determined from the rank-nullity theorem [34, 35]:

$$\dim(\mathcal{S}^M) = \dim(\mathcal{B}^M) - \operatorname{rank}(\mathbf{S}^M). \tag{62}$$

This approach quickly becomes intractable for meshes of even moderate size; accurate determination of even the rank of a large matrix of floating point numbers is a difficult problem. One approach was given by Alfeld [34].

4.8. Spline bases

An important tool in the definition, construction, and use of splines is the spline basis. As with any vector space, any spline in a spline space can be represented as a linear combination of the members of a linearly

independent set of functions having the same number of elements as the dimension of the spline space. Such a set of functions is referred to as a basis for the spline space.

Let $\mathbf{\Sigma}^M$ be a set of vectors that span the nullspace of $\mathbf{S}^M$. In other words, $\text{span}(\mathbf{\Sigma}^M) = \text{Null}(\mathbf{S}^M)$. Assume that the entries of the vectors can be indexed using the ordering given by $\mathsf{I}(\mathbf{B}^M)$. Then a set of basis functions $\mathbf{S}^M$ that span the spline space $\mathcal{S}^M$ is given by

$$\mathbf{S}^M = \{N | N = \sum_{\mathbf{i} \in \mathsf{I}(\mathbf{B}^M)} [\boldsymbol{\sigma}]_{\mathbf{i}} B_{\mathbf{i}}, \boldsymbol{\sigma} \in \mathbf{\Sigma}^M\}. \tag{63}$$

Here $\mathbf{B}^M$ is the set of Bernstein basis functions for the entire mesh.

The most recognized example of a spline space and its associated basis is the B-splines or basis splines. Originating with Schoenberg [57, 58] and commonly defined using the recursive approach of Cox [59], de Boor, and Mansfield [60], the basis spline functions are the minimally supported spline functions on a partitioning of an interval. The minimal, compact support of B-splines is significant for both design and analysis. Indeed, nonuniform rational B-splines (NURBS) form the basis of virtually all previous CAD modeling environments and have been employed extensively in isogeometric analysis [61] and its predecessors [3].

Much of the previous work on splines outside of B-splines has focused on determining the dimension of the spline space and then finding a basis for the spline space. This basis is often constructed as an object known as a minimal determining set, especially in the case of splines over triangulations. Finding a minimal determining set corresponds to finding a basis of the appropriate size for the spline space. However, it does not provide any insight into the quality or utility of a basis other than existence. Of more practical use is an algorithm for the direct construction of a basis for a spline space that satisfies the desirable properties of the B-splines: minimal support and positivity. An important corollary of the minimal support property of the B-splines that is not often appreciated is the fact that the minimal support property requires that when a single B-spline function is expressed in Bernstein-Bézier form, the function is minimally supported in the number of nonzero Bernstein coefficients. In algebraic terms, this means that the vectors of Bernstein coefficients of the B-spline functions form the sparsest basis of the nullspace of the smoothness constraint matrix. The problem of finding the sparsest basis for the nullspace of a matrix is known as the Null Space Problem and has been shown to be NP-hard [62, 63].

The sparsity property of the basis can be observed in the basis of the nullspace of the smoothness constraint matrix presented in Equation (36). If both elements are the same length, then the ratio $\rho = 1$ and the smoothness constraint matrix is

$$\mathbf{S}^M = \begin{bmatrix} 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 3 & -3 & -3 & 3 & 0 & 0 \\ 0 & 6 & -12 & 6 & -6 & 12 & -6 & 0 \end{bmatrix} \tag{64}$$

It can be shown that the sparsest basis for the matrix $\mathbf{S}^M$ is

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1/2 \\ 1/4 \\ 1/4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/4 \\ 1/4 \\ 1/2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{65}$$

and that these vectors correspond to the rows of the global Bézier extraction matrix for this mesh; i.e. these are the coefficients of the B-splines expressed in Bernstein-Bézier form. The associated spline basis functions are shown in Figure 9.

5. U-spline mesh topology

U-splines are created by grouping collections of Bernstein coefficients into topological blocks, the size of which is determined by the continuity constraints assigned to the underlying mesh. We now present the underlying topological considerations required to construct minimal sets of Bernstein coefficients which are the used to construct U-spline basis function.

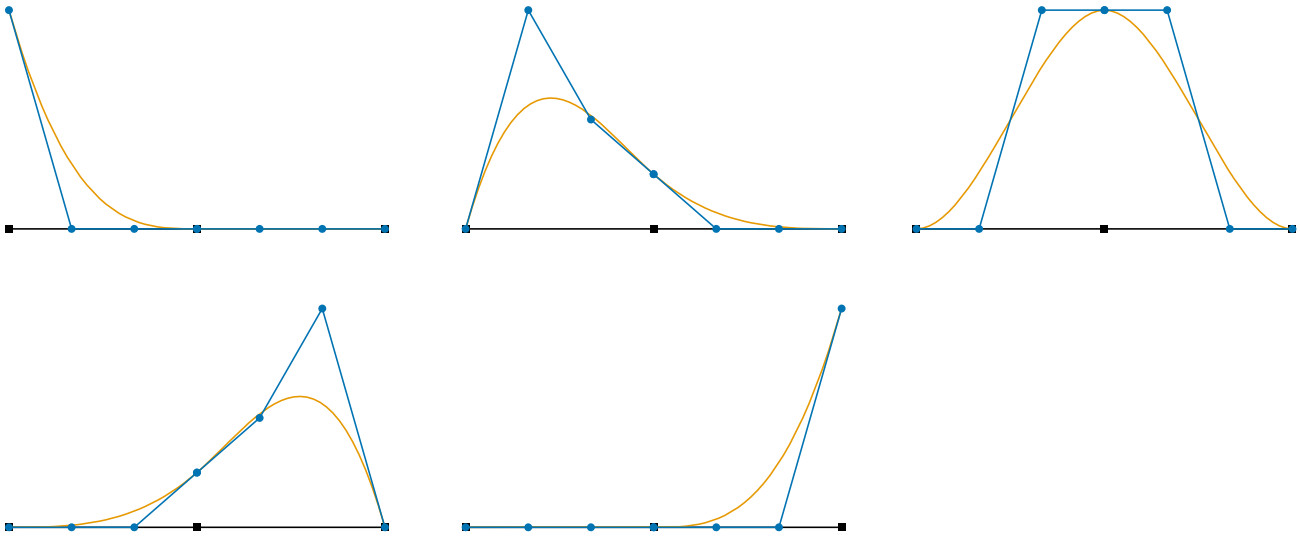

Figure 9: Plots of the B-spline basis function coefficient values (round markers) and the corresponding spline basis functions (thin lines).

5.1. Element index blocks

A basic structure used to organize the local function indices of the Bernstein bases used in construction of the U-spline basis is the element index block. An element index block is a set of element function indices grouped into an oriented block. To specify a block of indices on an element e an orientation $\boldsymbol{\sigma}$ is required that indicates the outward orientation of the block in each parametric direction, the inner index value $\boldsymbol{\mu}_i$, and the barrier and outer index values ($\boldsymbol{\mu}_b$ and $\boldsymbol{\mu}_o$, respectively) in each parametric direction. The symbol $\boldsymbol{\beta}$ is used to represent an element index block. The various properties are indicated by subscript or superscript:

$$\boldsymbol{\beta}^{e,\boldsymbol{\sigma}}_{\boldsymbol{\mu}_i,\boldsymbol{\mu}_b,\boldsymbol{\mu}_o}. \tag{66}$$

In situations where not all data is required, the sub- and superscripts are omitted to simplify presentation.

The distance operators for the element index block relative to an interface I must be defined in the parametric direction(s) perpendicular to the interface and in the direction(s) parallel to the interface. The direction in which the distance is measured depends on whether the inner, barrier, or outer index bounds is of interest. The relevant bound is indicated by a superscript. The distances of the inner and barrier bounds are always measured opposite the outward orientation while the outer bound is always measured in the direction of the outward orientation. Let $\mathbf{I}(I^{\parallel})$ represent the set of element indices parallel to the interface I. Then the the inward distances are given by

$$[\mathbf{d}^a_{I^\perp}(\boldsymbol{\beta}^{e,\boldsymbol{\sigma}}_{\boldsymbol{\mu}_i,\boldsymbol{\mu}_b,\boldsymbol{\mu}_o})]_j = \begin{cases} \mu_{a,j}, & \sigma_j = + \\ p_j - \mu_{a,j}, & \sigma_j = - \end{cases}, j \notin \mathbf{I}(I^{\parallel}), a \in \{i, b\} \tag{67}$$

$$[\mathbf{d}^a_{I^\parallel}(\boldsymbol{\beta}^{e,\boldsymbol{\sigma}}_{\boldsymbol{\mu}_i,\boldsymbol{\mu}_b,\boldsymbol{\mu}_o})]_j = \begin{cases} \mu_{a,j}, & \sigma_j = + \\ p_j - \mu_{a,j}, & \sigma_j = - \end{cases}, j \in \mathbf{I}(I^{\parallel}), a \in \{i, b\} \tag{68}$$

and the outward distances by

$$[\mathbf{d}^o_{I^\perp}(\boldsymbol{\beta}^{e,\boldsymbol{\sigma}}_{\boldsymbol{\mu}_i,\boldsymbol{\mu}_b,\boldsymbol{\mu}_o})]_j = \begin{cases} p_j - \mu_{o,j}, & \sigma_j = + \\ \mu_{o,j}, & \sigma_j = - \end{cases}, j \notin \mathbf{I}(I^{\parallel}) \tag{69}$$

$$[\mathbf{d}^o_{I^\parallel}(\boldsymbol{\beta}^{e,\boldsymbol{\sigma}}_{\boldsymbol{\mu}_i,\boldsymbol{\mu}_b,\boldsymbol{\mu}_o})]_j = \begin{cases} p_j - \mu_{o,j}, & \sigma_j = + \\ \mu_{o,j}, & \sigma_j = - \end{cases}, j \in \mathbf{I}(I^{\parallel}). \tag{70}$$

Unless they are required for clarity, the sub- and superscripts $\boldsymbol{\mu}_a, a \in \{i, b, o\}$ and $\boldsymbol{\sigma}$ are suppressed.

As an example, consider an index block on an element e of degree $\mathbf{p}^e = (3, 3)$, with $\boldsymbol{\sigma} = (-, +)$, $\boldsymbol{\mu}_i = (0, 2)$, $\boldsymbol{\mu}_b = (1, 2)$, $\boldsymbol{\mu}_o = (2, 2)$ with the interface along the minimum s boundary. For brevity, let $\boldsymbol{\beta}^e = \boldsymbol{\beta}^{e,\boldsymbol{\sigma}}_{\boldsymbol{\mu}_i,\boldsymbol{\mu}_b,\boldsymbol{\mu}_o}$. The

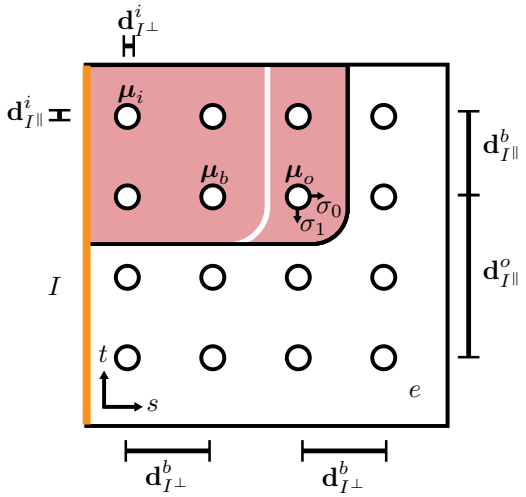

Figure 10: Example of a single element index block in an element with $p^e = (3, 3)$. The boundary values that define the index block are $\boldsymbol{\mu}_i = (3, 0)$, $\boldsymbol{\mu}_b = (1, 2)$ and $\boldsymbol{\mu}_o = (2, 2)$. The orientation of the block is given by $\boldsymbol{\sigma} = (+, -)$. The distances are all measured relative to the interface I.

index block distances for this example are

$$\mathbf{d}^i_{I\perp}(\boldsymbol{\beta}^e) = (0) \tag{71}$$
$$\mathbf{d}^i_{I\parallel}(\boldsymbol{\beta}^e) = (0) \tag{72}$$
$$\mathbf{d}^b_{I\perp}(\boldsymbol{\beta}^e) = (1) \tag{73}$$
$$\mathbf{d}^b_{I\parallel}(\boldsymbol{\beta}^e) = (1) \tag{74}$$
$$\mathbf{d}^o_{I\perp}(\boldsymbol{\beta}^e) = (1) \tag{75}$$
$$\mathbf{d}^o_{I\parallel}(\boldsymbol{\beta}^e) = (2). \tag{76}$$

The relevant boundaries and distances for this example are shown in Figure 10.

5.2. Constrained index blocks

5.2.1. Introduction

The idea of a constrained index block can be motivated by several simple observations emanating from the properties of Bernstein-like bases in one-dimension. The key observation is this: Bernstein-like basis functions vanish n times at an interface where n is the index distance between the function index $\mathbf{i}$ and the interface I. As a result, if the distance from the index to the interface is greater than the continuity of the interface (i.e., $n > k(I)$), then setting the value of coefficient $b_{\mathbf{i}}$ on element e has no impact on the coefficients associated with basis functions between $\mathbf{i}$ and the interface I on element e or on element e'. In this case, the constrained index block only spans the single index $\mathbf{i}$. However, if the index distance is less than or equal to the continuity of the interface then the constrained index block spans all the coefficients between $\mathbf{i}$ and the interface I on element e and those having an index within a distance $k(I) - n$ of the interface I on element e'. In other words, all these coefficients are constrained by choosing the value of the coefficient associated with index $\mathbf{i}$ on element e.

It is convenient to formalize these observations in terms of element index blocks. Starting with an index block $\boldsymbol{\beta}^e$ having an outward orientation pointing away from interface I the bounds of the block are initialized so that the originating index is the only member of $\boldsymbol{\beta}^e$. If the inner bound of the index block is within a distance less than or equal to the continuity of the interface I (i.e., $\mathbf{d}^i_{I\perp}(\boldsymbol{\beta}^e) \leq k(I)$), then the inner bound is adjusted to match the index location of the interface. An index block in the adjacent element e' is then constructed with bounds chosen such that

$$\mathbf{d}^i_{I\perp}(\boldsymbol{\beta}^{e'}) = 0$$

and

$$\mathbf{d}^b_{I\perp}(\boldsymbol{\beta}^{e'}) = k(I) - \mathbf{d}^i_{I\perp}(\boldsymbol{\beta}^e).$$

Here the outward bounds match the barrier: $\boldsymbol{\mu}_o = \boldsymbol{\mu}_b$. This will always be the case for one-dimensional meshes. If the index block is far enough from a boundary that it doesn't couple through that boundary, then another

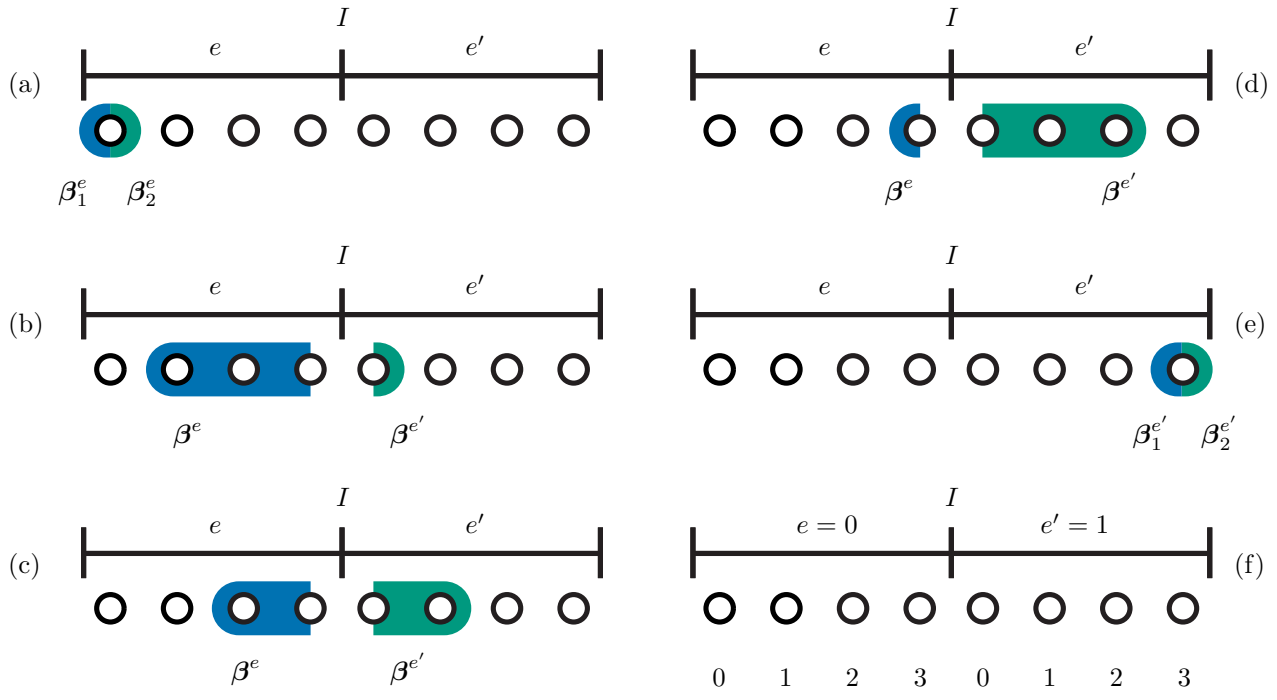

Figure 11: The constrained index blocks which can be defined on two elements of degree 3 with a C^2 interface ($k(I) = 2$) between the elements. In this simple case, each constrained index blocks coincides with the function support of a B-spline function. The basis functions for this example are shown in Figure 9. The orientation and bounding values for this figure are shown in Table 2.

block is added to represent a closed set of blocks. Several examples for a two-element mesh are shown in Figure 11. Both elements have polynomial degree $p = 3$. The interface I between the elements is C^2 ($k(I) = 2$). All possible constrained index blocks are shown. The constrained index blocks in this example correspond to the indices of the sets of nonzero coefficients of the basis functions over this mesh. These basis functions were considered previously and are shown in Figure 9. The values of each defining property of the element index blocks that make up each constrained index block shown in Figure 11 are given in Table 2.

5.2.2. Notation

The symbol $\boldsymbol{\kappa}$ is used to represent a constrained index block. Corners are key features of a constrained index block. The corners of a constrained index block are the indices for which at least d neighboring indices are not in the index set (d is the parametric dimension of the mesh). The set of indices corresponding to the corners of a constrained index block are denoted by $\mathbf{C}(\boldsymbol{\kappa})$. The set of originating indices from which the constrained index block can be constructed is called a seed set (seed($\boldsymbol{\kappa}$)). In the univariate case, the corner and seed sets always coincide but this is not the case for the general multivariate setting.

It is also helpful to define notation for the expansion of a constrained index block across an interface. Given a constrained index block $\boldsymbol{\kappa}$, the expansion of $\boldsymbol{\kappa}$ is the constrained index block produced by identifying the corner index $\mathbf{i}_c$ of $\boldsymbol{\kappa}$ that is nearest the interface I, determining the index block $\boldsymbol{\beta}_a$ that contains the corner, and constructing a new constrained index block that has an index block $\boldsymbol{\beta}_b$ that shares the corner $\mathbf{i}_c$ and has all orientations equal to $\boldsymbol{\beta}_a$ except the one in the direction of the interface I. A constrained index block is denoted by $\boldsymbol{\epsilon}_I(\boldsymbol{\beta}_a)$. An example of a constrained index block $\boldsymbol{\kappa}$ and its expansion across an interface I is shown in Figure 12.

5.2.3. The multivariate setting

A multivariate constrained index block is defined as the smallest set of element index blocks having a seed index $\mathbf{i}$ in its corner set and where for each index block $\boldsymbol{\beta}^e$ in the set, there is another index block on an adjacent element $\boldsymbol{\beta}^{e'}$ such that if $\boldsymbol{\sigma}_{I\parallel}(\boldsymbol{\beta}^e) = \boldsymbol{\sigma}_{I\parallel}(\boldsymbol{\beta}^{e'})$ then either $\boldsymbol{\beta}^e|_I \subseteq \boldsymbol{\beta}^{e'}|_I$ or $\boldsymbol{\beta}^{e'}|_I \subseteq \boldsymbol{\beta}^e|_I$; if $\boldsymbol{\sigma}_{I\parallel}(\boldsymbol{\beta}^e) \neq \boldsymbol{\sigma}_{I\parallel}(\boldsymbol{\beta}_0^{e'})$ then there is a $\boldsymbol{\beta}_1^{e'}$ having $\boldsymbol{\sigma}_{I\parallel}(\boldsymbol{\beta}_1^{e'}) = \boldsymbol{\sigma}_{I\parallel}(\boldsymbol{\beta}^e)$, $\boldsymbol{\beta}^e|_I \subseteq \boldsymbol{\beta}_1^{e'}|_I$ or $\boldsymbol{\beta}_1^{e'}|_I \subseteq \boldsymbol{\beta}^e|_I$ and $\mathbf{d}^a_{I\perp}(\boldsymbol{\beta}_0^{e'}) = \mathbf{d}^a_{I\perp}(\boldsymbol{\beta}_1^{e'})\, \forall a \in \{i, b, o\}$.

(a)	$\boldsymbol{\beta}_1^e$	$\sigma = (-)$
		$\boldsymbol{\mu}_i = (0)$
		$\boldsymbol{\mu}_b = (0)$
		$\boldsymbol{\mu}_o = (0)$
	$\boldsymbol{\beta}_2^e$	$\sigma = (+)$
		$\boldsymbol{\mu}_i = (0)$
		$\boldsymbol{\mu}_b = (0)$
		$\boldsymbol{\mu}_o = (0)$
(b)	$\boldsymbol{\beta}^e$	$\sigma = (-)$
		$\boldsymbol{\mu}_i = (3)$
		$\boldsymbol{\mu}_b = (1)$
		$\boldsymbol{\mu}_o = (1)$
	$\boldsymbol{\beta}^{e'}$	$\sigma = (+)$
		$\boldsymbol{\mu}_i = (0)$
		$\boldsymbol{\mu}_b = (0)$
		$\boldsymbol{\mu}_o = (0)$
(c)	$\boldsymbol{\beta}^e$	$\sigma = (-)$
		$\boldsymbol{\mu}_i = (3)$
		$\boldsymbol{\mu}_b = (2)$
		$\boldsymbol{\mu}_o = (2)$
	$\boldsymbol{\beta}^{e'}$	$\sigma = (+)$
		$\boldsymbol{\mu}_i = (0)$
		$\boldsymbol{\mu}_b = (1)$
		$\boldsymbol{\mu}_o = (1)$
(d)	$\boldsymbol{\beta}^e$	$\sigma = (-)$
		$\boldsymbol{\mu}_i = (3)$
		$\boldsymbol{\mu}_b = (3)$
		$\boldsymbol{\mu}_o = (3)$
	$\boldsymbol{\beta}^{e'}$	$\sigma = (+)$
		$\boldsymbol{\mu}_i = (0)$
		$\boldsymbol{\mu}_b = (2)$
		$\boldsymbol{\mu}_o = (2)$
(e)	$\boldsymbol{\beta}_1^{e'}$	$\sigma = (-)$
		$\boldsymbol{\mu}_i = (3)$
		$\boldsymbol{\mu}_b = (3)$
		$\boldsymbol{\mu}_o = (3)$
	$\boldsymbol{\beta}_2^{e'}$	$\sigma = (+)$
		$\boldsymbol{\mu}_i = (3)$
		$\boldsymbol{\mu}_b = (3)$
		$\boldsymbol{\mu}_o = (3)$

Table 2: Orientation and bounding values for the element index blocks in Figure 11.

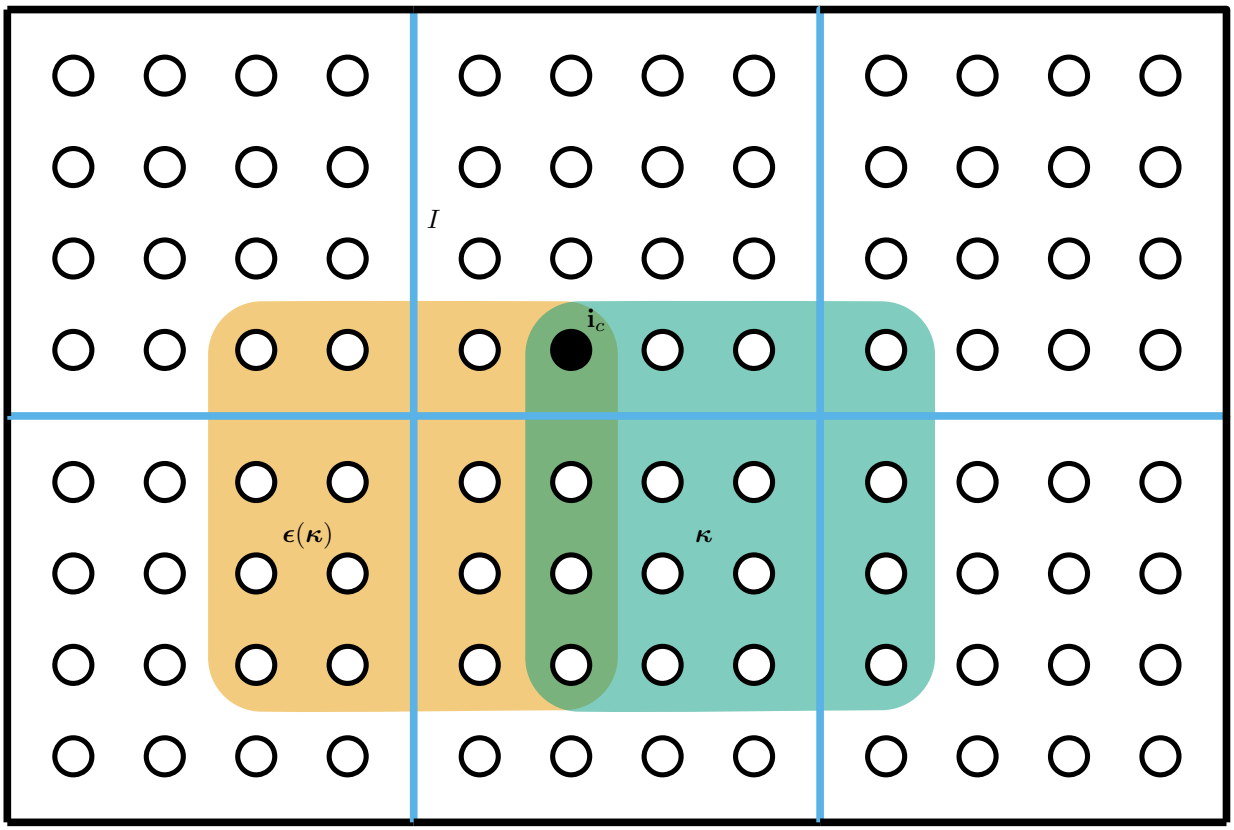

Figure 12: Illustration of the expansion $\boldsymbol{\epsilon}_I(\boldsymbol{\kappa})$ of a constrained index block $\boldsymbol{\kappa}$ across the interface I.

Such a set can be produced by a flood algorithm. Various considerations required to successfully transition across different types of interfaces are given in the sections that follow.

5.2.4. Block transfer across matching interfaces

Two elements match across an interface if the shared interface spans the entire side of both elements and if the basis in each parametric direction matches in each shared parametric direction parallel to the interface. For polynomial Bernstein bases this requires $\mathbf{p}^{e}_{I^{\parallel}} = \mathbf{p}^{e'}_{I^{\parallel}}$. In general, the orientations of the two elements e and e' will not be aligned and so the equality is assumed to hold only after an appropriate identification between parametric directions in the two elements has been made. The subscript $I^{\parallel}$ is used to represent quantities that have been appropriately transformed to lie on the interface I. Quantities in the perpendicular direction are marked with $I^{\perp}$.

In order to produce a function that is continuous across an element interface, the coefficients in each line perpendicular to the interface must satisfy the one-dimensional continuity constraints. When constructing a constrained index block that spans the interface, this requires that the perpendicular size of the element index blocks on both sides of the interface must match. Additionally, the same rule used in the one-dimensional case to construct an element index block on an adjacent element must be applied in the perpendicular direction.

In precise terms, given elements e and e' that share an interface I and an element index block $\boldsymbol{\beta}^{e}$ that satisfies $\mathbf{d}^{b}_{I^{\perp}}(\boldsymbol{\beta}^{e}) \leq k(I)$ and $\mathbf{d}^{i}_{I^{\perp}}(\boldsymbol{\beta}^{e}) = 0$, construct an element index block $\boldsymbol{\beta}^{e'}$ such that

$$\mathbf{d}^{i}_{I^{\parallel}}(\boldsymbol{\beta}^{e'}) = \mathbf{d}^{i}_{I^{\parallel}}(\boldsymbol{\beta}^{e}), \tag{77}$$

$$\mathbf{d}^{b}_{I^{\parallel}}(\boldsymbol{\beta}^{e'}) = \mathbf{d}^{b}_{I^{\parallel}}(\boldsymbol{\beta}^{e}), \tag{78}$$

$$\mathbf{d}^{o}_{I^{\parallel}}(\boldsymbol{\beta}^{e'}) = \mathbf{d}^{o}_{I^{\parallel}}(\boldsymbol{\beta}^{e}), \tag{79}$$

$$\boldsymbol{\sigma}_{I^{\parallel}}(\boldsymbol{\beta}^{e'}) = \boldsymbol{\sigma}_{I^{\parallel}}(\boldsymbol{\beta}^{e}), \tag{80}$$

$$\mathbf{d}^{i}_{I^{\perp}}(\boldsymbol{\beta}^{e'}) = 0, \tag{81}$$

$$\mathbf{d}^{b}_{I^{\perp}}(\boldsymbol{\beta}^{e'}) = k(I) - \mathbf{d}^{b}_{I^{\perp}}(\boldsymbol{\beta}^{e}), \tag{82}$$

$$\boldsymbol{\sigma}_{I^{\perp}}(\boldsymbol{\beta}^{e'}) \neq \boldsymbol{\sigma}_{I^{\perp}}(\boldsymbol{\beta}^{e}). \tag{83}$$

These requirements are general. There are specific subtleties associated with some configurations but the basic requirements on adjacent blocks remain the same.

An example of the required relationships is given in Figures 13 and 14. Figure 13 shows two possible orientations of two elements that share a matching interface I along with the indexing of the Bernstein basis on each element. Figure 14 shows the layout of two element index blocks that satisfy Equations (77) to (83). The orientation is indicated by rounding the outermost corner. An explicit indexing is not required to illustrate the blocks. The bounding values and orientations for the blocks under the two indexing options shown in Figure 13 are given in Table 3.

5.2.5. Block transfer across interfaces adjacent to T-junctions

In order to transition across an interface that is adjacent to a T-junction in the mesh four cases shown in Figure 15 are considered. Each case corresponds to two possible scenarios: computing an element index block in the large element, given an element index block in a small adjacent element or computing an element index block in a small element, given an element index block in the large element. The interface between the elements is indicated by I. The interface J represents the side of the element that lies in the inward direction of the element index block and is not parallel to the interface I.

First, consider the scenario of constructing an element index block in the small element, given an element index block in the large element $\boldsymbol{\beta}^{e_0}$. If $\mathbf{d}^{i}_{J^{\perp}} = 0$ as shown in part (a) of Figure 15 and the index nearest the T-junction vertex in the small element lies in the inward direction of the outward bounds induced by $\boldsymbol{\beta}^{e_0}$ then Equations (77) to (83) are used to produce the index block $\boldsymbol{\beta}^{e_1}$. If $\mathbf{d}^{i}_{J^{\perp}} = 0$ but the interface I is specified so that the index nearest the T-junction vertex in the small element lies in the outward direction of the induced outward bound as shown in part (b) of Figure 15, then Equations (77) to (83) are used to produce one index block $\boldsymbol{\beta}^{e_2}_{0}$. Another element index block $\boldsymbol{\beta}^{e_2}_{1}$ is also created that satisfies the perpendicular block requirements

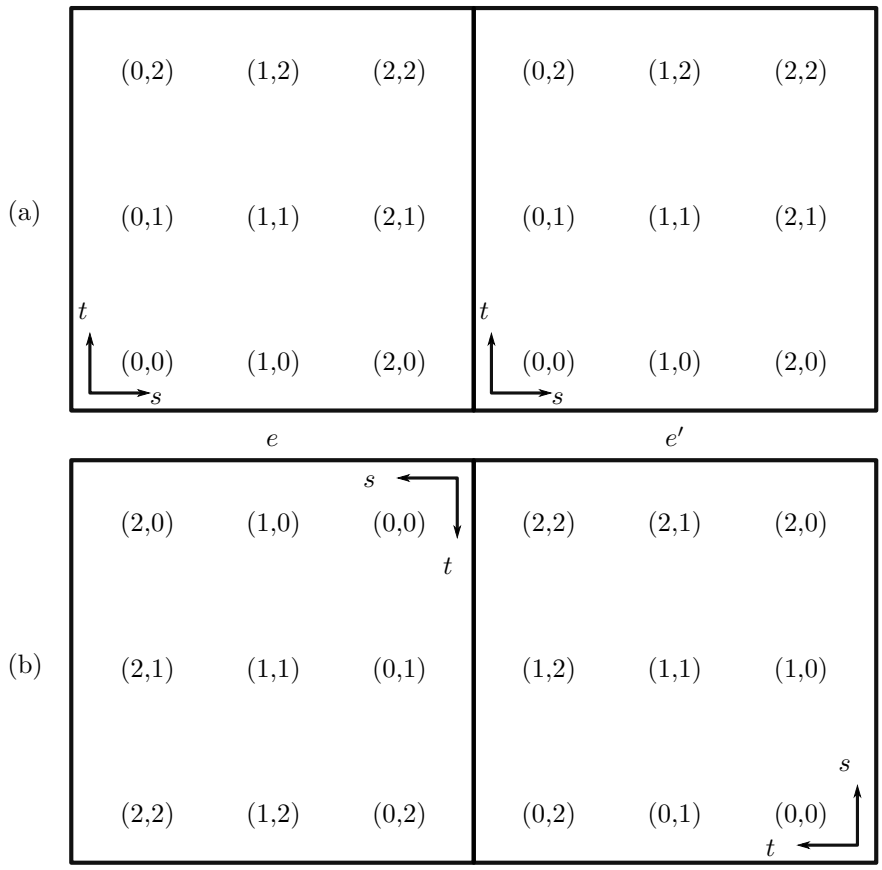

Figure 13: Two possible orientations of two elements sharing a matching interface.

(a)	$\boldsymbol{\beta}^e$	$\boldsymbol{\sigma} = (-,-)$
		$\boldsymbol{\mu}_i = (2,2)$
		$\boldsymbol{\mu}_b = (1,1)$
		$\boldsymbol{\mu}_o = (1,1)$
	$\boldsymbol{\beta}^{e'}$	$\boldsymbol{\sigma} = (+,-)$
		$\boldsymbol{\mu}_i = (0,2)$
		$\boldsymbol{\mu}_b = (0,1)$
		$\boldsymbol{\mu}_o = (0,1)$
(b)	$\boldsymbol{\beta}^e$	$\boldsymbol{\sigma} = (+,+)$
		$\boldsymbol{\mu}_i = (0,0)$
		$\boldsymbol{\mu}_b = (1,1)$
		$\boldsymbol{\mu}_o = (1,1)$
	$\boldsymbol{\beta}^{e'}$	$\boldsymbol{\sigma} = (-,-)$
		$\boldsymbol{\mu}_i = (2,2)$
		$\boldsymbol{\mu}_b = (1,2)$
		$\boldsymbol{\mu}_o = (1,2)$

Table 3: Orientation and bounding values for the element index blocks in Figure 14. The (a) and (b) refer to the two different indexing schemes shown in Figure 13

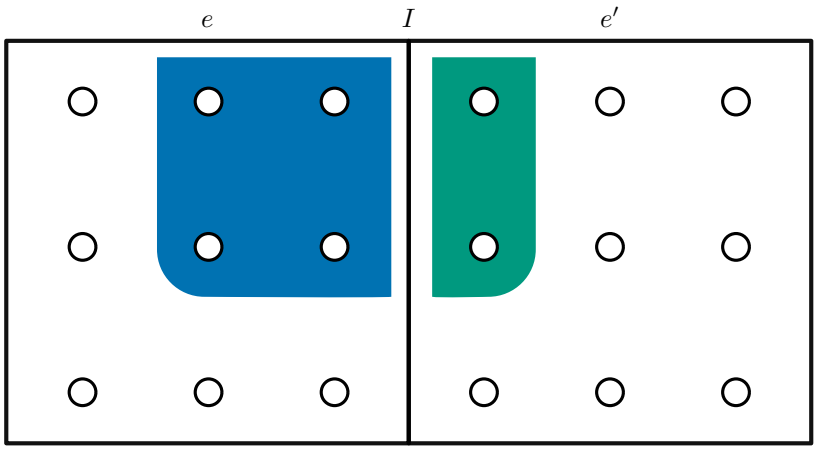

Figure 14: Adjacent element index blocks that satisfy Equations (77) to (83) on adjacent matching elements. The degree of each element is $\mathbf{p} = (2, 2)$ and the interface continuity is $k(I) = 1$.

Equations (81) to (83) but has the following requirements relative to $\boldsymbol{\beta}_0^{e_2}$ for the direction parallel to the interface:

$$\boldsymbol{\sigma}_{I\|}(\boldsymbol{\beta}_1^{e_2}) = -\boldsymbol{\sigma}_{I\|}(\boldsymbol{\beta}_0^{e_2}), \tag{84}$$

$$\mathbf{d}^i_{I\|}(\boldsymbol{\beta}_1^{e_2}) = 0, \tag{85}$$

$$\mathbf{d}^b_{I\|}(\boldsymbol{\beta}_1^{e_2}) = \mathbf{d}^o_{I\|}(\boldsymbol{\beta}_0^{e_2}), \tag{86}$$

$$\mathbf{d}^o_{I\|}(\boldsymbol{\beta}_1^{e_2}) = \mathbf{d}^b_{I\|}(\boldsymbol{\beta}_0^{e_2}). \tag{87}$$

If the distance $\mathbf{d}^i_{I\|}I(\boldsymbol{\beta}^{e_0}) > 0$ and the T-junction is in the inward direction on the small element then the element index block on the smaller element is expanded so that $\mathbf{d}^i_{I\|}I(\boldsymbol{\beta}^{e_0}) = 0$. This is illustrated in part (c) of Figure 15.

If the distance $\mathbf{d}^i_{I\|}I(\boldsymbol{\beta}^{e_0}) > 0$ and the T-junction is in the outward direction on the small element then the element index block on the smaller element is not expanded and the secondary block is computed as before. This is shown in part (d) of Figure 15.

5.2.6. Block transfer across interfaces between elements of different degree

The matching requirements for element index blocks given in Equations (77) to (83) can be used without modification to construct element index blocks between elements with differing polynomial degree. It is important to note that although $\mathbf{d}^b_{I\|}(\boldsymbol{\beta}^{e'}) = \mathbf{d}^b_{I\|}(\boldsymbol{\beta}^{e})$ and $\mathbf{d}^o_{I\|}(\boldsymbol{\beta}^{e'}) = \mathbf{d}^o_{I\|}(\boldsymbol{\beta}^{e})$, the values of the bounds in the element index block on the element of higher degree (e') are not equivalent. In this case, $\boldsymbol{\mu}_o(\boldsymbol{\beta}^{e'}) \neq \boldsymbol{\mu}_b(\boldsymbol{\beta}^{e'})$. A two element example with one element having degree $\mathbf{p}^e = (2, 2)$ and the other degree $\mathbf{p}^{e'} = (3, 3)$ is shown in Figure 16. The distinct barrier bound is indicated by a white line. For this case,

$$\mathbf{d}^i_{I\|}(\boldsymbol{\beta}^{e}) = \mathbf{d}^i_{I\|}(\boldsymbol{\beta}^{e'}) = 0, \tag{88}$$

$$\mathbf{d}^b_{I\|}(\boldsymbol{\beta}^{e}) = \mathbf{d}^b_{I\|}(\boldsymbol{\beta}^{e'}) = 1, \tag{89}$$

$$\mathbf{d}^o_{I\|}(\boldsymbol{\beta}^{e}) = \mathbf{d}^o_{I\|}(\boldsymbol{\beta}^{e'}) = 1. \tag{90}$$

5.2.7. Block transfer across interfaces between elements of different type

The algorithms presented here can also be adapted to accommodate meshes containing elements of differing types, for example quadrilaterals and triangles or tetrahedra and hexahedra. All elements possess a Bernstein basis and each Bernstein basis reduces to a Bernstein basis of dimension one less on the boundaries. As such, the same reasoning can be applied to transfer index blocks across interfaces. The only consideration is the differing parametric descriptions used on each side. An example of a transfer between a square and a triangle element is shown in Figure 17. The interface has continuity $k(I) = 1$.

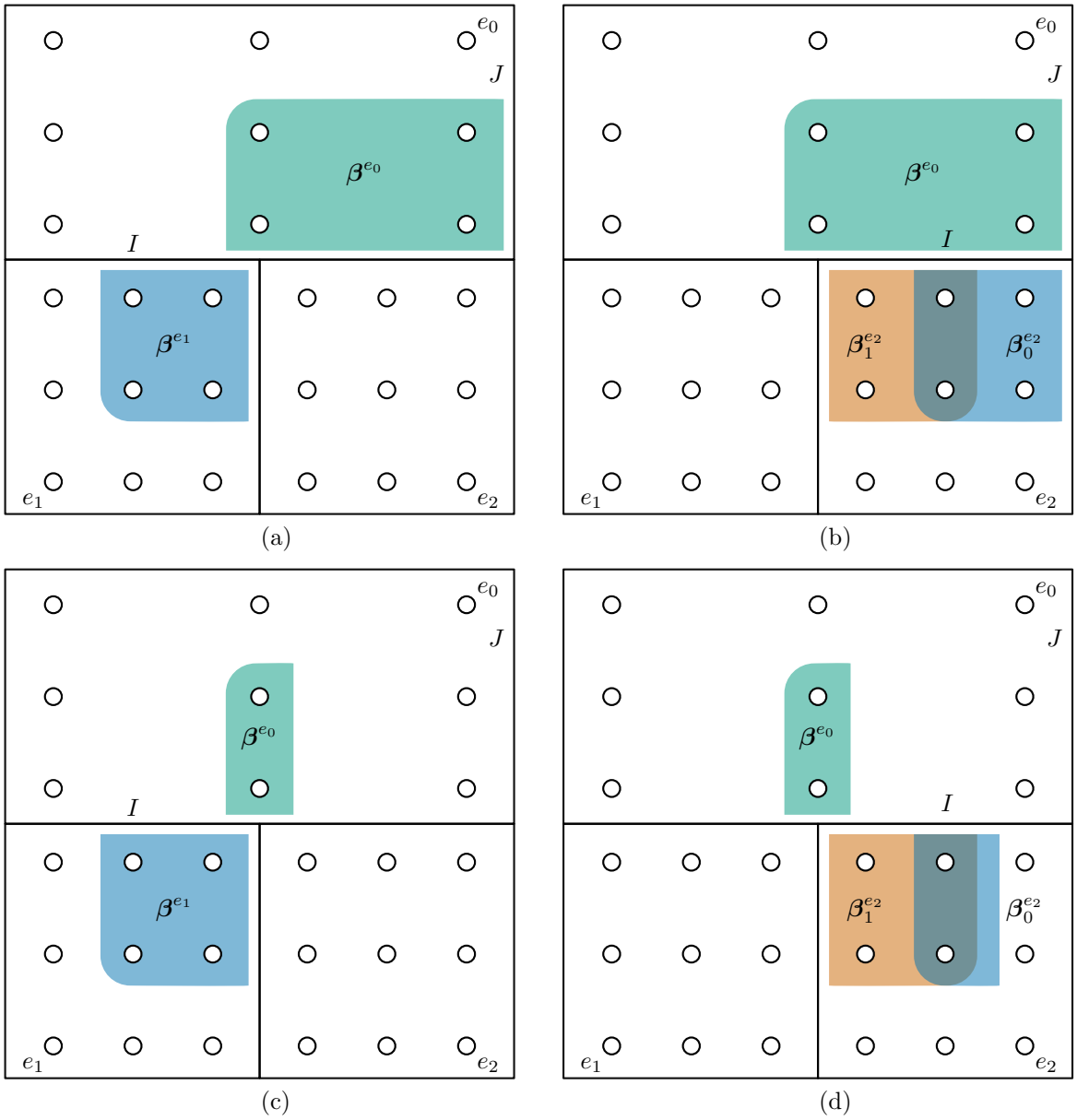

Figure 15: Examples of index blocks computed to maintain compatibility across the interface I For these examples, the continuity of the interface is $c(I) = 2$.

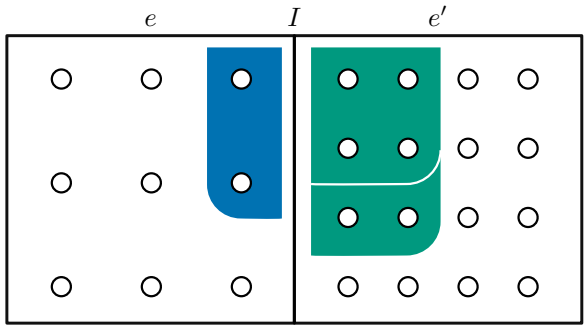

Figure 16: Adjacent element index blocks that satisfy Equations (77) to (83) on adjacent elements having different polynomial degree. The degree of the element e is $\mathbf{p}^e = (2, 2)$, the degree of element e' is $\mathbf{p}^{e'} = (3, 3)$, and the interface continuity is $k(I) = 2$. Due to the difference in degree between the elements, the barrier bound is different from the outer bound. The barrier bound is indicated by a white line.

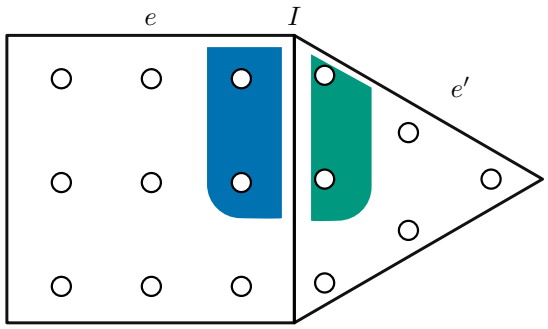

Figure 17: Example of compatible index blocks across the interface I where the adjacent elements are of different type. The continuity of the interface is $k(I) = 0$.

5.2.8. An illustrative example containing multiple adjacent T-junctions

Consider the example in Figure 18. In part (a) , the algorithm is initialized with the element index block $\boldsymbol{\beta}_a$. The notion of connections between index blocks is used here. Typical flooding methods require data as to which directions have already been processed. A connection between two index blocks indicates a processed direction. Part (b) shows the result of constructing the element index block $\boldsymbol{\beta}_b$ across the interface I and connecting it to the original block $\boldsymbol{\beta}_a$. The connections are used to illustrate the relationships between the blocks and are also useful in determining open boundaries for the flood. Similarly, the element index block $\boldsymbol{\beta}_c$ is constructed and connected across the interface J. Both of the these blocks conform to $\boldsymbol{\beta}_a$ across their respective shared interfaces: $\boldsymbol{\beta}_a|_I = \boldsymbol{\beta}_b|_I$ and $\boldsymbol{\beta}_a|_J = \boldsymbol{\beta}_c|_J$. In part (c), the blocks produced by processing $\boldsymbol{\beta}_c$ on interface K and then processing the resulting block $\boldsymbol{\beta}_d$ on interface L to obtain $\boldsymbol{\beta}_e$ are shown. Part (d) shows the blocks that are added to account for the presence of the T-junction. Each of the new blocks has an orientation opposite the base block produced earlier in the flood: $\boldsymbol{\sigma}_{I_1^{\parallel}}(\boldsymbol{\beta}_c) \neq \boldsymbol{\sigma}_{I_1^{\parallel}}(\boldsymbol{\beta}_f)$, $\boldsymbol{\sigma}_{I_1^{\parallel}}(\boldsymbol{\beta}_d) \neq \boldsymbol{\sigma}_{I_1^{\parallel}}(\boldsymbol{\beta}_g)$, and $\boldsymbol{\sigma}_{I_3^{\parallel}}(\boldsymbol{\beta}_e) \neq \boldsymbol{\sigma}_{I_3^{\parallel}}(\boldsymbol{\beta}_h)$. Each block is connected to the the same blocks as the base block across the relevant interface. Thus, block $\boldsymbol{\beta}_f$ is connected to blocks $\boldsymbol{\beta}_a$ and $\boldsymbol{\beta}_d$. Similar connections are made for blocks $\boldsymbol{\beta}_f, \boldsymbol{\beta}_g, \boldsymbol{\beta}_h$. The remaining interfaces perpendicular to I_0, I_1, I_2, I_3 are now processed. Part (e) shows the result of processing the open index blocks $\boldsymbol{\beta}_d$ and $\boldsymbol{\beta}_e$ and the resulting connection between them obtained by processing the interface I_3. There is no element in the open direction for either of these blocks and so the placeholder blocks $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_j$ are inserted and connected. Processing $\boldsymbol{\beta}_a$ across interface I_4 produces $\boldsymbol{\beta}_k$ (part (f)) while processing $\boldsymbol{\beta}_g$ across the same interface produces $\boldsymbol{\beta}_l$ (part (g)). Similarly, in part (h), the blocks $\boldsymbol{\beta}_b, \boldsymbol{\beta}_f$ produce the element index blocks $\boldsymbol{\beta}_m, \boldsymbol{\beta}_n$, respectively, when processed across the interface I_5 and the element index blocks $\boldsymbol{\beta}_c$ and $\boldsymbol{\beta}_h$ processed across the interface I_6 produce the blocks $\boldsymbol{\beta}_o$ and $\boldsymbol{\beta}_p$. The flood now terminates as the remaining element index blocks without connections across interfaces $(\boldsymbol{\beta}_k, \boldsymbol{\beta}_l, \boldsymbol{\beta}_m, \boldsymbol{\beta}_n, \boldsymbol{\beta}_o, \boldsymbol{\beta}_p)$ are subsumed by other blocks that do have connections across those interfaces (i.e., $\boldsymbol{\beta}_n \subset \boldsymbol{\beta}_b$, etc.).

5.3. Function index support

The set of indices associated with nonzero Bernstein coefficients in the support of one basis function is referred to as a function index support. A function index support $\boldsymbol{\phi}$ is the union of the indices in a set of constrained index blocks such that

$$\boldsymbol{\phi} = \bigcup_{\boldsymbol{\kappa} \in \mathbf{K_i}} \mathbf{I}(\boldsymbol{\kappa}) \tag{91}$$

where the set of constrained index blocks $\mathbf{K_i}$ associated with the index $\mathbf{i}$ is defined as the smallest set of constrained index blocks where every member shares at least one corner with another member, at least one member of the set has the index $\mathbf{i}$ as a member of its corner set, and every index $\mathbf{o}$ in the boundary set $\partial\mathbf{I}(\mathbf{K}_i)$ is greater than a distance $k(I)$ from every interface I perpendicular to the outward boundary direction associated with the index $\mathbf{o}$. Additionally, for each element index block $\boldsymbol{\beta}$ in each possible extension of a constrained index block $\boldsymbol{\kappa}$, there must be some other constrained index block $\boldsymbol{\kappa}'$ in the set that has an index block $\boldsymbol{\beta}'$ satisfying

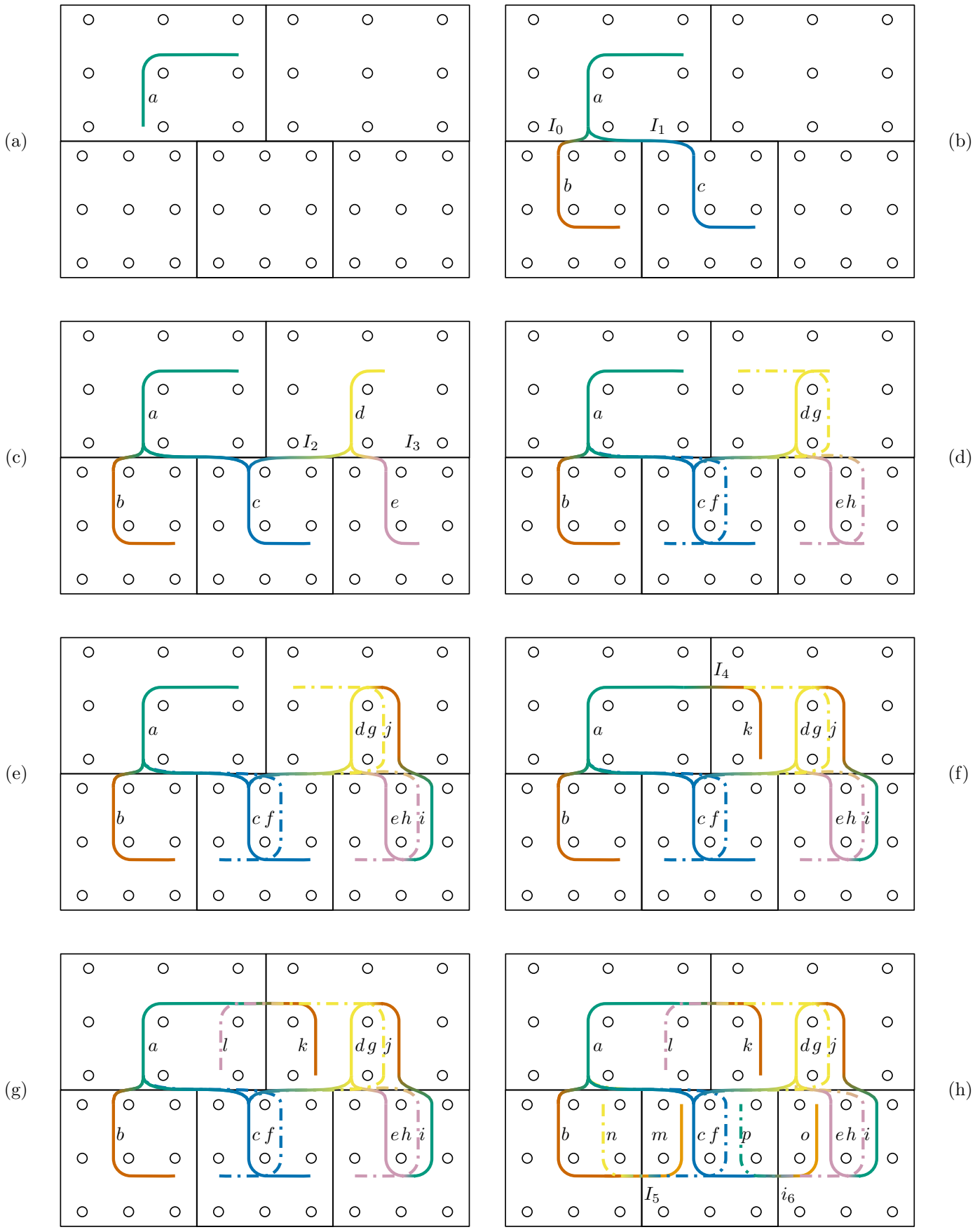

Figure 18: Steps in the flooding algorithm to produce a constrained index block in the neighborhood of a system of T-junctions.

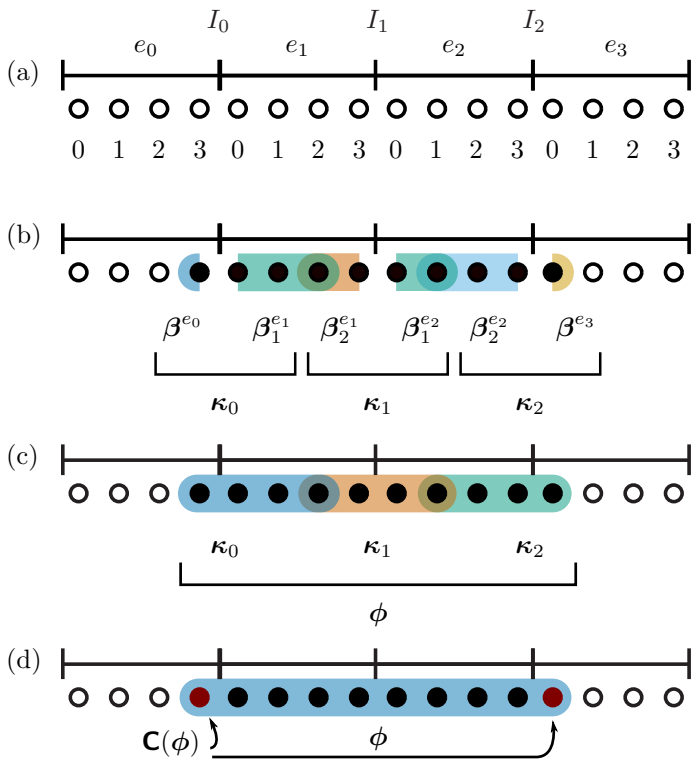

Figure 19: Univariate mesh consisting of 4 elements with a basis of polynomial degree 3 on each element. The figure shows index, element, and interface labels (a), the element index blocks that make up the support of the function (b), the constrained index blocks formed from the element index blocks (c), and the full function index support $\boldsymbol{\phi}$ with the indices that are part of the corner set $\mathbf{C}(\boldsymbol{\phi})$ shown in light grey (d). The properties of the element index blocks are given Table 4.

$\boldsymbol{\beta} \subseteq \boldsymbol{\beta}'$. More precisely, in set-theoretic notation, these conditions can be written as

$$\exists \boldsymbol{\kappa} \in \mathbf{K_i} : \mathbf{i} \in \mathbf{C}(\boldsymbol{\kappa}), \tag{92}$$

$$\forall \boldsymbol{\kappa} \in \mathbf{K_i}, \exists \boldsymbol{\kappa}' \in \mathbf{K_i} : \mathbf{C}(\boldsymbol{\kappa}) \cap \mathbf{C}(\boldsymbol{\kappa}') \neq \varnothing, \tag{93}$$

$$\forall \mathbf{o} \in \partial \mathbf{I}(\mathbf{K_i}), \forall I \in M : I \perp \mathbf{n}(\mathbf{o}), \mathbf{d}_{I^\perp}(\mathbf{o}) > k(I), \tag{94}$$

$$\forall \boldsymbol{\kappa} \in \mathbf{K_i}, \forall I \in \partial\Omega(\mathbf{I}(\boldsymbol{\kappa})), \forall \boldsymbol{\beta} \in \boldsymbol{\epsilon}_I(\boldsymbol{\kappa}) \exists \boldsymbol{\kappa}' \in \mathbf{K_i} : \boldsymbol{\beta} \subseteq \boldsymbol{\beta}', \boldsymbol{\beta}' \in \boldsymbol{\kappa}'. \tag{95}$$

The set of all function index supports defined over a given mesh is denoted by $\{\boldsymbol{\phi}_A\}$ and the corner set of a function index support by $\mathbf{C}(\boldsymbol{\phi}_A)$. Here we use the operator ∂ to indicate the boundary of a set. If the set is a set of indices, then the boundary is all members of the set that are not surrounded on all sides by other members of the set. The operator producing the normal with respect to an index $\mathbf{n}$ is also used. This provides a parametric direction in which the input index has no neighbors. The parametric support of an index set is represented by $\Omega(\mathbf{I})$. This represents the union of the parametric domains of all elements having indices in the set $\mathbf{I}$.

These properties are illustrated for the univariate case through an example. Consider the univariate mesh shown in Figure 19. It consists of 4 elements, $\{e_i \,|\, i \in \{0,1,2,3\}\}$ and 3 interfaces $\{I_i \,|\, i \in \{0,1,2\}\}$. Each element is assigned a basis with polynomial degree $p^{e_i} = 3$ and the continuity of each interface is $k(I_i) = 2$. A seed index $\mathbf{i}_s$ is selected. Here it is assumed that the index carries both the element e and the local index of a single function. This corresponds to a single Bernstein coefficient that will be nonzero in our final function. The minimum set of coefficients that must be nonzero while still satisfying the continuity constraints on either side of element e are then determined. The indexing of the relevant quantities is shown in part (a). The element index blocks that make up the function index support are shown and labeled in part (b). The element index blocks on elements with more than one block carry an additional identifying subscript. The element index blocks that make up each of the constrained index blocks $\boldsymbol{\kappa}_i$ are indicated. The constrained index blocks are shown with labels in part (c). Finally, the function index support $\boldsymbol{\phi}$ is shown in part (d) marked in blue along with the members of the corner index set $\mathbf{C}(\boldsymbol{\phi})$ marked in red.

$\boldsymbol{\beta}^{e_0}$	$\sigma = (-)$	$\boldsymbol{\beta}_1^{e_1}$	$\sigma = (+)$	$\boldsymbol{\beta}_2^{e_2}$	$\sigma = (-)$
	$\boldsymbol{\mu}_i = (3)$		$\boldsymbol{\mu}_i = (0)$		$\boldsymbol{\mu}_i = (3)$
	$\boldsymbol{\mu}_b = (3)$		$\boldsymbol{\mu}_b = (2)$		$\boldsymbol{\mu}_b = (2)$
	$\boldsymbol{\mu}_o = (3)$		$\boldsymbol{\mu}_o = (2)$		$\boldsymbol{\mu}_o = (2)$
$\boldsymbol{\beta}_1^{e_2}$	$\sigma = (+)$	$\boldsymbol{\beta}_2^{e_2}$	$\sigma = (-)$	$\boldsymbol{\beta}^{e_3}$	$\sigma = (+)$
	$\boldsymbol{\mu}_i = (0)$		$\boldsymbol{\mu}_i = (3)$		$\boldsymbol{\mu}_i = (0)$
	$\boldsymbol{\mu}_b = (1)$		$\boldsymbol{\mu}_b = (1)$		$\boldsymbol{\mu}_b = (0)$
	$\boldsymbol{\mu}_o = (1)$		$\boldsymbol{\mu}_o = (1)$		$\boldsymbol{\mu}_o = (0)$

Table 4: Orientation and bounding values for the element index blocks in Figure 19.

Several examples of multivariate functions index supports are shown in Figure 20. The constrained index blocks that make up each function index support are assigned indices from 1-9. The seed indices are marked with filled circles. Note that some care must be taken when forming the underlying constrained index blocks in the multivariate setting as described in Section 5.2.

6. U-spline basis construction

As noted previously, for a given mesh there is a basis for the associated spline space that corresponds to the sparsest basis for the nullspace of the smoothness constraint matrix. Using a brute force approach to solve the nullspace problem and determine the members of this basis is an intractable problem. To avoid these issues, the U-spline approach leverages the mesh topology and properties of a Bernstein-like basis in order to incrementally construct member functions of the sparsest possible spline basis without directly solving the global nullspace problem. Note that although this approach is generally applicable to bases that satisfy the properties of a Bernstein-like basis given in Section 2.2, for simplicity, only examples using polynomial Bernstein bases are considered.

6.1. Outline of U-spline basis construction

A basic outline of how U-spline basis functions are constructed is as follows:

1. Input a mesh containing:
   - Cuboidal or simplicial elements and connectivity between adjacent elements.
   - Parameteric data assigned to each edge of every element, including parametric length and direction within the local coordinate system of the element.
   - Specification of the desired level of continuity on each interface between elements.
   - Sufficient data to define a Bernstein-like basis on each element. For polynomial Bernstein bases, it is sufficient to specify the polynomial degree in each parametric direction.
2. For each seed Bernstein index:
   (a) Construct the function index support that has the seed as a corner through the following steps:
      i. Determine a constrained index block having the seed as a corner and mark it.
      ii. Mark any unmarked constrained index blocks sharing corners and having minimal overlap with previously marked blocks until no new blocks can be marked.
      iii. if the seed index is not a corner of the function support then continue to the next available seed.
   (b) Determine whether a function with the same function index support has already been created.
   (c) If the function index support is new, determine the coefficient values through the following steps:
      i. Form the smoothness constraint matrix for the coefficients corresponding to the indices in the function index support.
      ii. Solve for the vector that defines the nullspace of the constraint matrix. The entries in this vector are the Bernstein coefficients that define the function.
3. The functions determined in the previous step must be normalized so that for each index in the mesh, the sum of all nonzero coefficients sharing that index for all basis functions over the mesh is equal to one. This is accomplished by forming and solving a linear system.

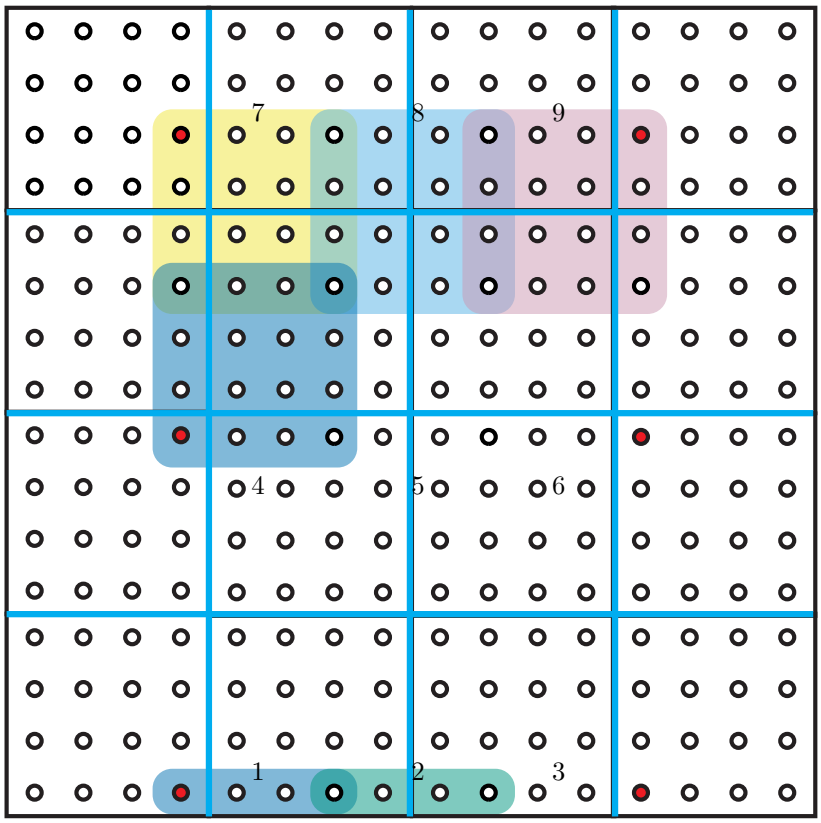

Figure 20: Selected examples of the constrained index blocks used to construct function index supports in the multivariate setting.

6.2. Construction of function index supports

To make the function index support determination phase of the U-spline construction process more concrete, additional important details are given. Beginning with a seed index $\mathbf{i}$, Step (1) adds all constrained index blocks $\boldsymbol{\kappa}$ satisfying $\mathbf{i} \in \text{seed}(\boldsymbol{\kappa}) \cap \mathbf{C}(\boldsymbol{\kappa})$ to the set $\mathbf{K_i}$. Then, in Step (2) the corners of this new set are computed. In Step (3), all constrained index blocks satisfying $\mathbf{C}(\boldsymbol{\kappa}) \cap \text{seed}(\boldsymbol{\kappa}) \cap \mathbf{C}(\mathbf{K_i}) \neq \varnothing$ are added to the set. Steps (2) and (3) are repeated until no new constrained index blocks are added. The steps of this algorithm are illustrated in Figure 21 for a single function. This example begins with a seed index that is not a member of the final corner set. Although this approach works well in one-dimension, in a multivariate setting it is wise to choose seed indices that are located within an element such that they do not couple with any interfaces in at least two parametric directions.

6.3. Enumeration and uniqueness of function index supports

In contrast to the tensor-product B-spline basis where a natural ordered indexing is inherited from the ordering of the univariate B-splines and the tensor-product construction, U-splines do not possess a natural ordering. Instead, a property inherited from the sparsity of the U-spline basis is employed. By construction, each function must have a unique function index support and because each function is the sparsest possible function in the index space, each function must have at least one index corner that it shares with no other function. As a result, the label set $\mathbf{L}(\boldsymbol{\phi}_A)$ of a function index support can be defined as the subset of the corner set where

$$\mathbf{L}(\boldsymbol{\phi}_A) = \mathbf{C}(\boldsymbol{\phi}_A) \setminus \bigcup_{B \neq A} \mathbf{C}(\boldsymbol{\phi}_B). \tag{96}$$

A unique natural number is then assigned to each label set. Leveraging label sets is particularly important for performant implementations of U-splines where the usage of the full support of a function as a key in a map is impractical.

It should be noted that there is no direct topological connection between the parametric mesh and the natural connectivity of the function control points. The construction of U-spline control meshes will be addressed in a future work.

6.4. Determining function coefficient values

Once the function index support of a single function has been determined, the values of the coefficients associated with the indices in the support can be determined. This is accomplished by forming a restricted smoothness constraint matrix for only the coefficients with indices that lie in the function index support. Given a function index support set $\boldsymbol{\phi}$, the smoothness constraint matrix $\mathbf{S}_{\boldsymbol{\phi}}$ is formed by removing all columns from the global smoothness matrix $\mathbf{S}$ that correspond to indices $\mathbf{i} \notin \boldsymbol{\phi}$. Also, any rows consisting entirely of zeros after this removal step are also removed. If the remaining matrix is empty, then the values of all coefficients will be one (this will only occur for supports with only one entry). Otherwise, the nullspace of the resulting matrix $\mathbf{S}_{\boldsymbol{\phi}}$ will be one-dimensional. A vector of coefficients $\mathbf{b}_{\boldsymbol{\phi}}$ is then obtained by solving the following linear constraint system:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{b}_{\boldsymbol{\phi}}\|, && (97)\\ \text{subject to} \quad & \mathbf{S}_{\boldsymbol{\phi}}\mathbf{b}_{\boldsymbol{\phi}} = \mathbf{0}, && (98)\\ & \mathbf{b}_{\boldsymbol{\phi}} > 0. && (99) \end{aligned}$$

This problem can be solved as a linear programming problem by a simplex method or similar technique.

6.5. Normalization of the basis

As described previously, each U-spline basis function is defined by a function coefficient vector $\mathbf{b}_{\boldsymbol{\phi}_A}$. For simplicity $\boldsymbol{\phi}_A$ is dropped in favor of A. These vectors represent the coefficients of a non-normalized basis for the U-spline space. To produce a basis that forms a partition of unity, the coefficient vectors must be normalized. The normalized basis is obtained by solving the linear system

$$\begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_n \end{bmatrix} \boldsymbol{\nu} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{100}$$

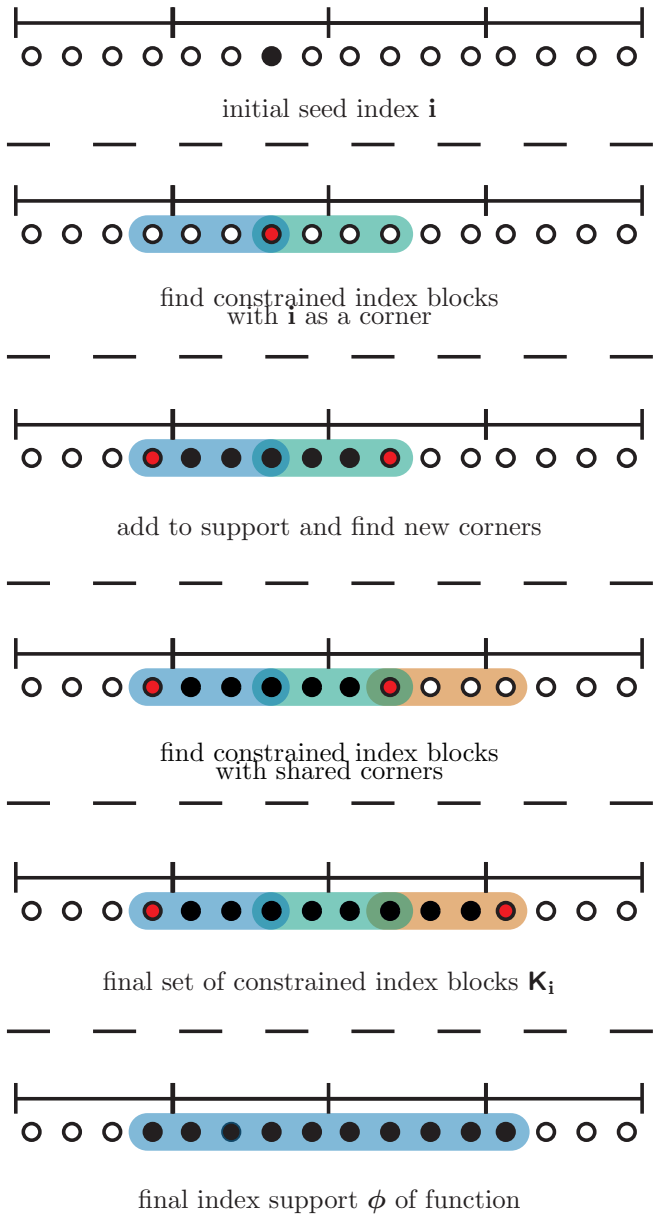

Figure 21: Illustration of the steps of the flooding algorithm for determining the function index support for the function shown in Figure 19.

where n is the total number of basis functions defined over a mesh. The normalized vectors of coefficients are given by $\nu_A \mathbf{b}_A$. Because any basis for a U-spline space can be normalized, all coefficient vectors presented hereafter are assumed to be normalized.

Theorem 1. *Any basis for a U-spline space can be normalized to form a partition of unity.*

Proof. By definition, a U-spline space must satisfy the continuity conditions at every interface in the mesh. When expressed in Bernstein-Bézier form, each U-spline basis function in the spline space corresponds to a vector of Bernstein coefficients $\mathbf{b}_A$. Each of these vectors lies in the nullspace of the global smoothness constraint matrix. The simplest nontrivial member of the spline space is the constant function. This function is represented in Bernstein-Bézier form as a constant vector. By definition, a set of U-spline basis functions $\{\bar{N}_A\}$ for the U-spline space spans the space and is linearly independent. Because the constant vector is a member of the nullspace, there is a unique set of coefficients ν_A such that

$$\sum_A \nu_A \bar{N}_A = 1. \tag{101}$$

□

6.6. An illustrative mixed degree example in one-dimension

The U-spline algorithm provides a natural construction for the basis of multi- or mixed-degree splines. The algorithm presented here functions equally well regardless of the polynomial degree as long as the continuity for each interface between elements is less than or equal to the polynomial degree in the direction perpendicular to the interface on both elements adjacent to the interface.

An example of a three element mesh with mixed degree is shown in Figure 22. The polynomial degrees for the elements are $p^{e_0} = 2, p^{e_1} = 3$, and $p^{e_2} = 4$. The element lengths are $\ell_{e_0} = 1/4, \ell_{e_1} = 1/3$ and $\ell_{e_2} = 5/12$. The continuity of the interfaces is $k(I_0) = 1$ and $k(I_1) = 2$. The nonzero coefficients of each basis function are indicated by the filled dots below the plots of the figures. The constrained index blocks for each function are also shown.

The global smoothness constraint matrix for this example, as determined by the U-spline algorithm, is

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -8/3 & 8/3 & 3 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -15/4 & 15/4 & 4 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 75/8 & -75/4 & 75/8 & -12 & 24 & -12 & 0 & 0 \end{bmatrix}. \tag{102}$$

The resulting coefficient vectors which define the U-spline basis functions can be arranged as the rows of a matrix commonly called the extraction operator:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8/17 & 8/17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9/17 & 9/17 & 1 & 155/331 & 75/331 & 75/331 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 176/331 & 19360/38727 & 19360/38727 & 55/117 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 32/117 & 32/117 & 62/117 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{103}$$

It can be verified that the extraction operator satisfies $\mathbf{S}\mathbf{C}^T = \mathbf{0}$. The extraction operator can be used to generate the spline basis in terms of the Bernstein basis over the mesh:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8/17 & 8/17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9/17 & 9/17 & 1 & 155/331 & 75/331 & 75/331 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 176/331 & 19360/38727 & 19360/38727 & 55/117 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 32/117 & 32/117 & 62/117 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{B}^M = \begin{bmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} \tag{104}$$

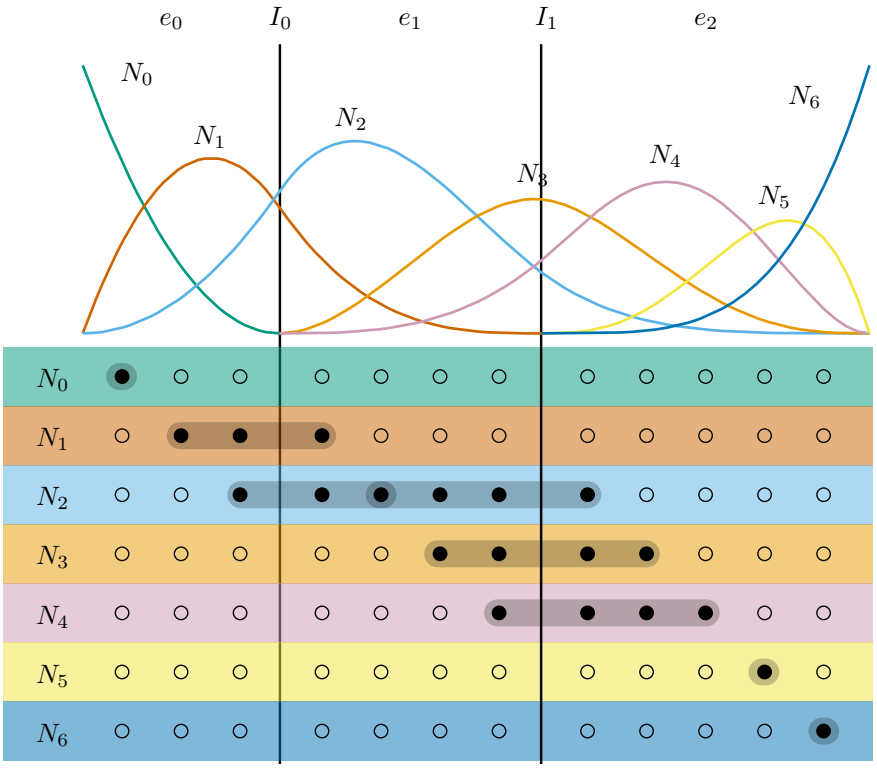

Figure 22: Example of a basis constructed over a mesh of mixed degree. The elements have degree 2, 3, 4, from left to right. The elements have lengths of 3, 4, and 5, respectively. The interfaces have continuity $k(I_0) = 1$ and $k(I_1) = 2$. The functions are plotted with the index support indicated below the plotted functions. Constrained index blocks are marked with shaded regions.

where the vector of Bernstein basis functions over the mesh is

$$\mathbf{B}^M = \left[\begin{array}{c} \mathbf{B}_{e_0} \\ \hline \mathbf{B}_{e_1} \\ \hline \mathbf{B}_{e_2} \end{array}\right] \tag{105}$$

and the element Bernstein basis vectors are

$$\mathbf{B}_{e_0}^T = \begin{bmatrix} B_{e_0,0}^2 & B_{e_0,1}^2 & B_{e_0,2}^2 \end{bmatrix}, \tag{106}$$

$$\mathbf{B}_{e_1}^T = \begin{bmatrix} B_{e_1,0}^3 & B_{e_1,1}^3 & B_{e_1,2}^3 & B_{e_1,3}^3 \end{bmatrix}, \tag{107}$$

$$\mathbf{B}_{e_2}^T = \begin{bmatrix} B_{e_2,0}^4 & B_{e_2,1}^4 & B_{e_2,2}^4 & B_{e_2,3}^4 & B_{e_2,4}^4 \end{bmatrix}. \tag{108}$$

6.7. Construction of multivariate U-splines

The U-spline algorithm for multivariate meshes operates on a similar principle of constructing function supports from sets of compatible constrained index blocks with accommodations made for the structure of the multivariate mesh. Whereas only a perpendicular direction exists for each interface in a one-dimensional mesh, higher dimensional meshes have directions that lie parallel to the interface which must be properly handled during block transfer across interfaces. These block transfer rules have been presented in detail for two-dimensional meshes in Section 5.2. Analogous rules can easily be defined for three-dimensional meshes (or higher dimensions).

It should be noted that there are actually several distinct approaches to computing function index supports that could be considered. The first, which has been presented in this work, uses the rules given here to construct constrained index blocks and then to find minimally supported collections of constrained index blocks by flooding in order to generate the function index supports. Second, it is also possible to use the rules for constrained index block construction given here to compute constrained index blocks and then find the minimal combination by searching for a minimal contiguous set of constrained index blocks that satisfy the continuity constraints. This is computationally more expensive. Third, it is possible to compute the constrained index blocks directly from

the constraint matrix without leveraging the principles given here and then computing a minimal set of these blocks. This is more efficient than the generic nullspace problem but is the least efficient method given here. All three methods leverage the locality of constrained index blocks to improve over the global method.

6.8. Latent constraints

In regions of the mesh where transitions in the properties assigned to mesh features occur in close proximity, additional complexities arise. In this work, transitions of scale (T-junctions), degree, and continuity are considered. Nonpolynomial Bernstein-like bases can introduce a fourth class of transitions that is functionally similar to the case of degree transitions.

The algorithms for the construction of constrained index blocks presented to this point have illustrated minimal sets of Bernstein indices required by simple interface continuity constraints. In some cases, the coefficients that lie beyond the minimal set of a single constraint must be considered. One example of this is shown in Figure 23. Part (a) of Figure 23 shows the indices of all the Bernstein coefficients that are coupled by the C^2 constraint. Part (b) shows the constrained index block created from the index indicated by the filled dot. The continuity constraint also induces latent constraints on coefficients with indices within a distance 2 of the interface. Latent extentions to the constrained index block are illustrated in part (c) of Figure 23. The coefficients with indices inside the gray areas are constrained by their proximity to the interface. If any one of the coefficients in one of the gray regions is nonzero then all of the coefficients in that region must also be nonzero. If one attempted to set one of the coefficients to be nonzero without the others, it would be impossible to satisfy the continuity constraints. This follows from the fact that multiple basis functions from the small elements are required to represent a single function from the large element.

A similar situation occurs at the interface between elements of differing polynomial degree. This actually holds for elements with differing numbers of basis functions having a nontrivial intersection of their trace spaces such as QEC Bernstein-like bases containing polynomials of some order. In this situation, more than one function is required to represent an adjacent function. An example is shown in Figure 24 where the constrained index blocks associated with indices on the lower degree (cubic) side are shown by the filled areas. The index locations are labeled. If the coefficient associated with the index $\mathbf{i}_a$ is nonzero and the coefficient associated with the index $\mathbf{i}_b$ is also nonzero, then the coefficient associated with $\mathbf{i}_d$ must also be nonzero. The same restriction applies to each pair of indices surrounded by a gray block. Another case is shown for the index $\mathbf{i}_f$.

In the simple examples considered heretofore, these latent constraints have not been considered but they are required for more complex examples, especially those involving adjacent transitions in perpendicular directions.

Consider the mesh of biquadratic elements shown in Figure 25. Part (a) of the figure shows the constrained index block computed from the seed index marked with a filled dot without considering the latent constrained index blocks. The latent blocks are shown in part (b) with the indices at which they intersect the previously computed support marked with filled dots. In part (c) the constrained index block has been expanded to include the relevant latent regions. The function index support computation is then continued to produce the upper left block shown in part (d).

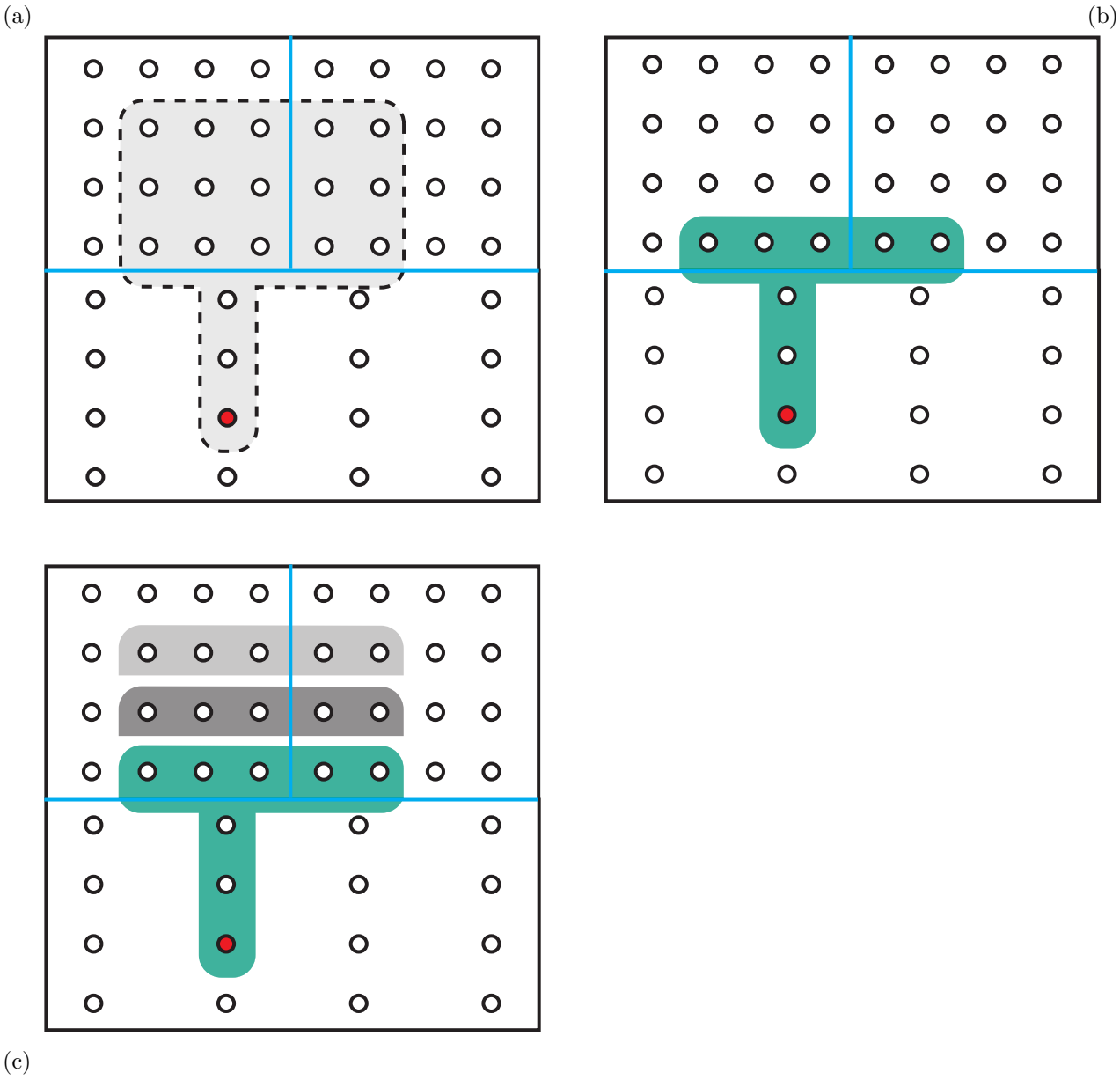

Figure 23: Indices of coefficients constrained by C^2 interfaces and a T-junction (a). Constrained index block generated by the index associated with the filled dot (b). Latent extensions to the constrained index block are shown in gray (c).

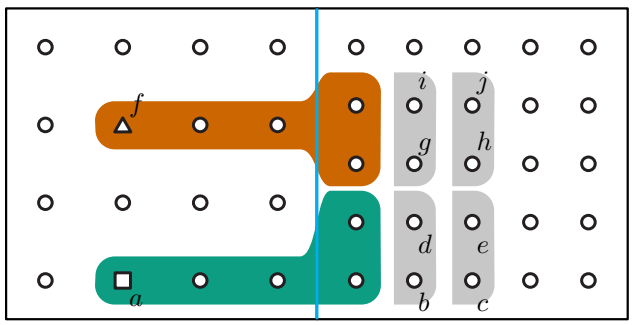

Figure 24: Selected examples of constrained index blocks for two elements of different polynomial degrees and the latent constrained index blocks (gray).

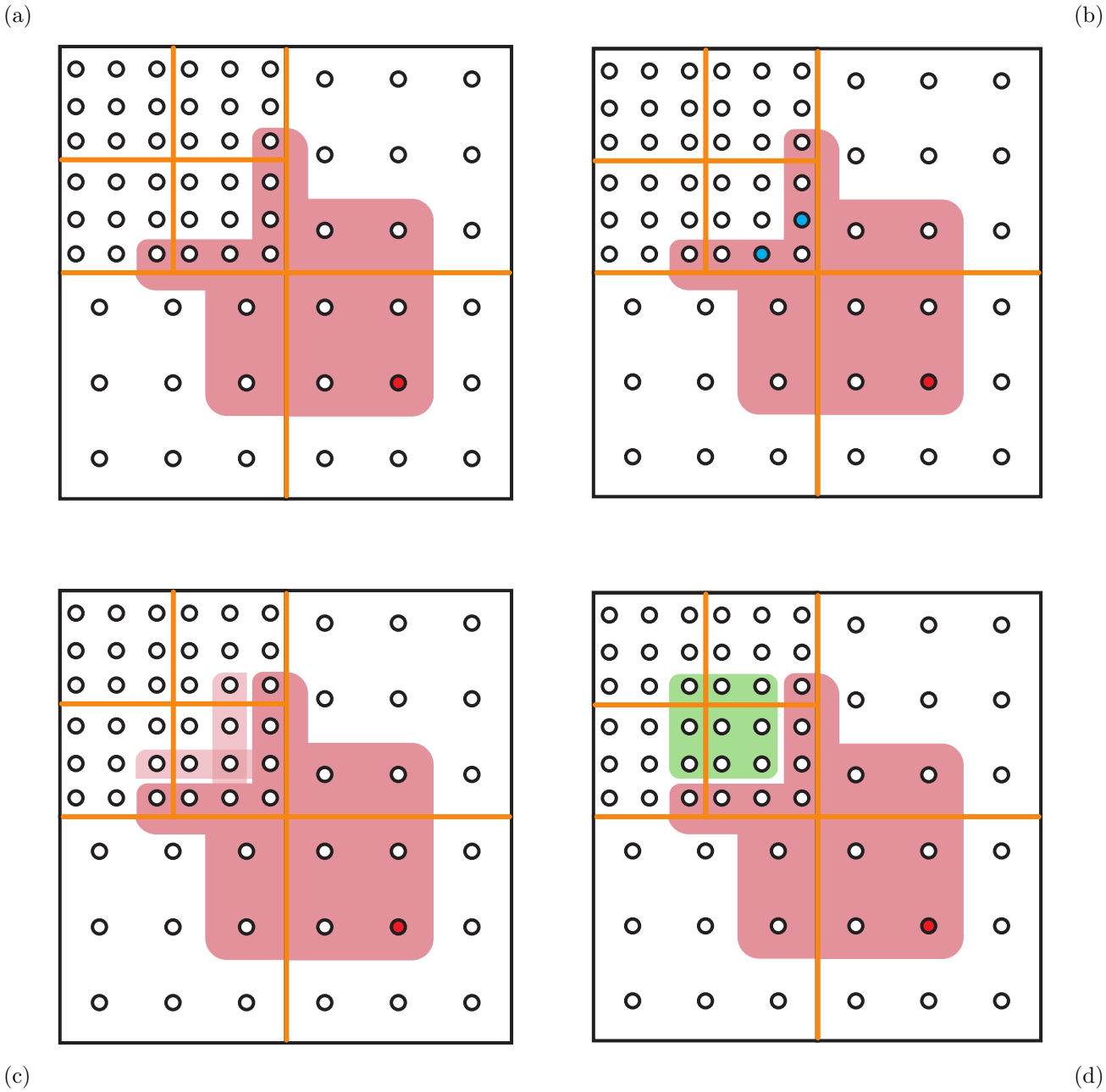

Figure 25: Example of constrained index blocks computed for a function over elements containing adjacent perpendicular T-junctions.

A similar situation occurs for perpendicular adjacent continuity transitions. A simple mesh containing edges with continuity C^1 and C^2 is shown in Figure 26. The edges with C^1 continuity are marked with dashed lines while the edges with C^2 continuity are marked with dotted lines. All elements have been assigned a bicubic polynomial basis. Part (a) of Figure 26 shows the constrained index block computed from any of the indices having a filled marker. The index blocks that make up the constrained index block are outlined with solid lines. The latent constrained index block due to the continuity transition is indicated by the dashed line. The constraint system assembled for the coefficients inside the index blocks bounded by the solid lines cannot satisfy the continuity constraints. In order to successfully compute a basis function, the latent block must be added to the support as shown in part (b) of Figure 26. Contrast the constrained index block shown in part (b) where the index support is square with the constrained index block shown in part (c) where the index support is not square. The seeds of the block shown in part (c) are positioned so that the support of the constrained index block extends beyond the support of any latent blocks.

One last example is presented that illustrates the implications of the function support requirement given in Equation (95). Part (a) of Fig. Figure 27 shows the function support that satisfies the function support requirements given in Equations (92) to (94) produced by using the index marked with a black filled dot as the seed. The shaded region indicates a portion of a latent constrained index block. Adding this block and satisfying (95) requires the addition of the index blocks shown in part (b) of Figure 27 to the function support. The full final support is shown in part (c).

We define the following concepts in order to incorporate latent constraints into the U-spline algorithm:

Augmented constrained index block. We define the augmentation of a constrained index block as the set of all index blocks that can be formed by starting with the index blocks in the input constrained index block and then adding any index block that can be produced by transferring across an interface without expanding the support of the initial constrained index block. We also add any blocks that when transferred across a face will produce an index block that exists in the set. An example is shown in Figure 28. Part (a) shows the constrained index block constructed in the neighborhood of two adjacent continuity transitions. In part (b), the index blocks added to form the augmented version are shown with red dashed lines.

Connections between constrained index blocks. Two constrained index blocks can be considered to be connected if any corner of one overlaps with the corner of any index block in the augmented constrained index block of the other and if the index blocks that share a corner are oriented in all the same directions but one. Any corners that can be connected to another constrained index block are termed closed in the direction of the connection.

Activation of latent constraints. We say that a latent constraint is activated if it overlaps with any of the corners of the augmented constrained index blocks in the support of the function. An example of this is shown in Figure 27.

Measurement of constrained index blocks. In the construction of multivariate U-splines, there are multiple choices of open corners at each step. In order to sort these choices, we introduce the parametric size of a constrained index block. Once a constrained index block has been created, the parametric size is computed by counting the number of active or nonzero Bernstein coefficients on each cell and multiplying that number by the parametric size of the cell divided by the total number of Bernstein basis functions in the basis defined over the cell. We have found that sorting potential constrained index blocks by the parametric size to be useful.

An example is shown in Figure 29. If the outer square has a parametric length of 1 on each side, then element e_1 has area $1/2$ while elements e_2 and e_3 have area $1/4$. There are 9 functions on each element so each nonzero coefficient contributes $1/9$. There are 4 nonzero coefficients on small elements in block 1 and 2 nonzero coefficients on the large element so the total size is $2/9$. Block 2 consists of 3 nonzero coefficients on the small elements so the total size is $1/12$.

6.8.1. Modifications to the U-spline algorithm

We modify the U-spline algorithm to accommodate the definitions and concepts presented in this section by replacing steps 2 (a) i-iii with the following steps:

i. Determine a constrained index block having the seed as a corner and mark it.

ii. Iterate all open corners of previously processed constrained index blocks and add the largest constrained index block having one of the open corners as a corner to the set.

iii. Compute the augmented constrained index block of the new block and activate any latent blocks.

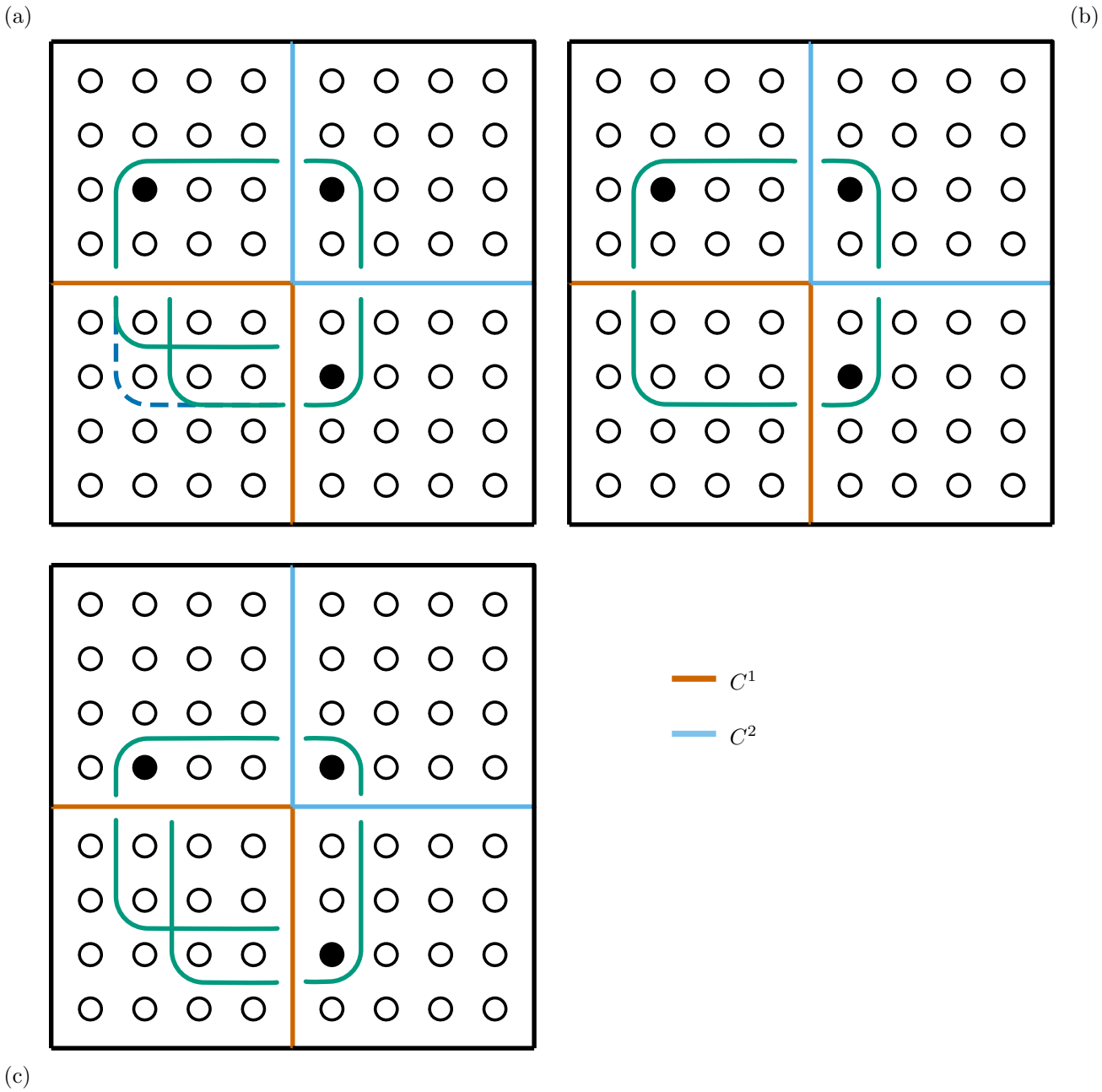

Figure 26: Example of constrained index blocks computed for a function over elements containing adjacent perpendicular continuity transitions.

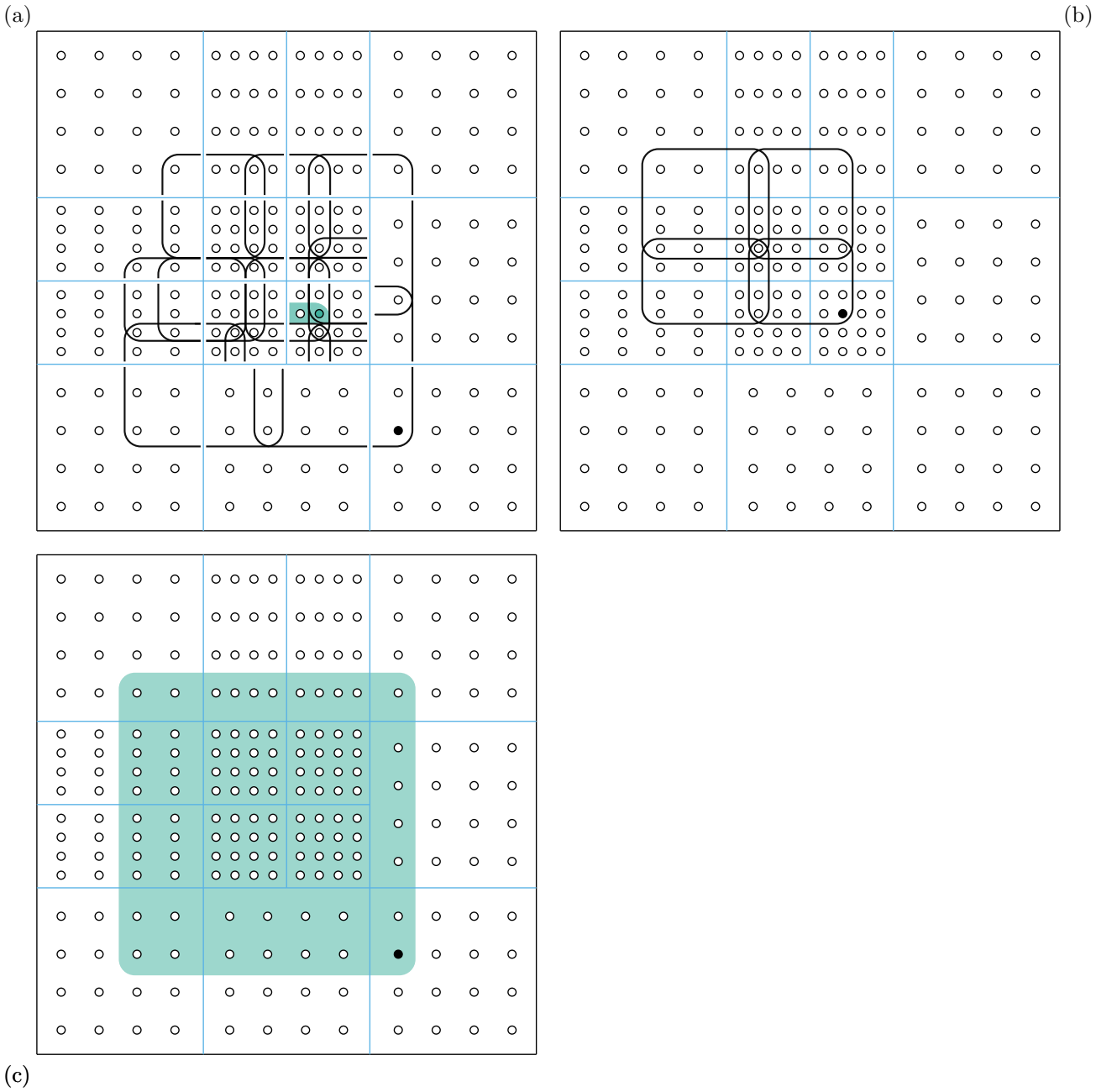

Figure 27: Example of a cubic function built from the index marked with a black dot to satisfy the support requirements given in Equations (92) to (94) in part (a). Part (b) shows the additional index blocks required by activation of the latent block marked with a filled region in part (a) and subsequently satisfying (95). The full final support is shown in part (c).

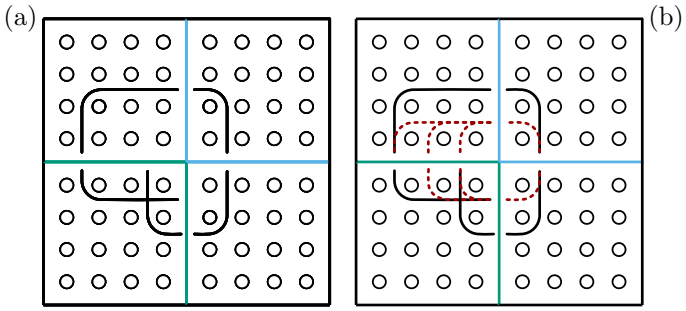

Figure 28: Part (a) shows the initial constrained index block. Part (b) shows the resulting augmented constrained index block with the additions highlighted with red dashed lines.

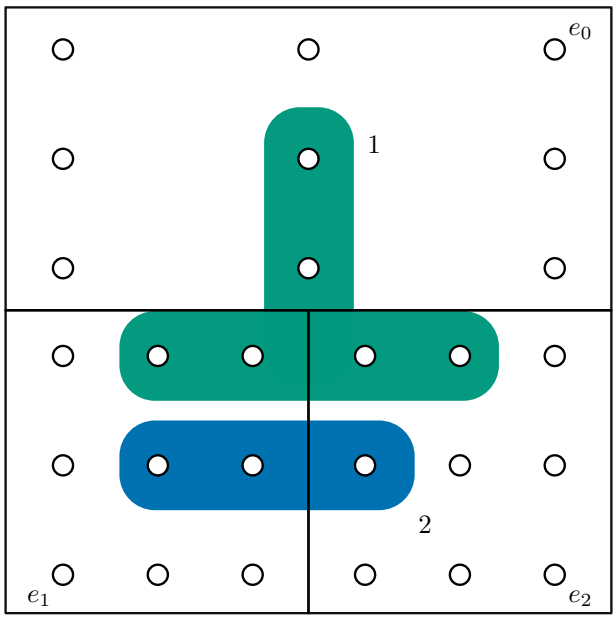

Figure 29: Two examples of measurement of constrained index blocks. The parametric size of block 1 is $\frac{2}{9}$ and the size of block 2 is $\frac{1}{12}$.

iv. If no open corners produce constrained index blocks that extend the support, then terminate.

7. Storage and representation of U-splines

Practical applications of the U-spline basis require efficient methods to store the basis functions and data associated with the basis functions. In order to represent a U-spline, the mesh must be stored along with the parametric data specifying edge lengths and the order of continuity of every interface. Enough data to specify the basis on each element must also be saved. For cuboidal elements with polynomial bases, it is sufficient to store the polynomial degree in each parametric direction. More general Bernstein-like bases require additional data. For simplicial elements, just a single polynomial degree is sufficient.

The primary requirement for use of a U-spline is a persistent indexing of the basis functions so that arrays of properties may be associated with the proper basis function. This is accomplished by pairing an integer index with at least one Bernstein index that is unique to the spline basis function. This unique index is sufficient to rebuild the function index support by using it as a seed. Other storage strategies are also possible; it may be beneficial to store all of the corners unique to a given function or all of the elements in the function support. Regardless of the extra data that may also stored, one unique corner must be stored. Any additional data is effectively prioritizing computation over storage as any properties of the basis may be computed from the unique corner using the corner as a seed for the U-spline construction.

It may also be advantageous to store the representation of the basis. The simplest approach is to store a map for each basis function that returns the coefficient value associated with each index in the function index support. An optimal approach is to store pointers to the unique mathematical expressions that, when evaluated at a specific parametric location, evaluate the basis. In this way, the redundancy in uniform mesh regions can be significantly reduced.

When using U-splines in computations, it is necessary to know which functions are defined on each element and what the values of the coefficients of the local basis that represent each function are. This data can be stored explicitly, but again, by storing pointers to the unique expressions used in the basis, all information necessary to use the basis in computations can be stored in an optimally compressed format.

8. Examples of multivariate basis construction

Several examples that illustrate fundamental differences between the U-spline approach and previously developed technologies are now presented.

To begin, an example of a single U-spline basis function is shown to demonstrate several of the unique features of the construction. Part (a) of Figure 30 shows a Bézier mesh. Each element has a basis of polynomial degree 3 in each direction. Each internal interface has C^2 continuity. The nonzero indices associated with this function are shown with filled circles. The shade of the circle indicates the relative value of the associated coefficient. Part (b) shows a contour plot of the function. Part (c) shows a three-dimensional surface produced by setting the z value of the control point associated with the function to 1. This example was chosen to highlight a significant difference between U-spline basis functions and many other splines such as B-splines, T-splines, and LR-splines among others. Most splines rely on tensor-product bases which are square. The U-spline construction is clearly capable of producing bases that are not generated by a tensor product.

A Bézier mesh consisting of nested T-junctions is considered next. Three instances of the same mesh are shown in Figures 31 to 33. In Figure 31, each element is assigned a bilinear basis and each edge is C^0. In Figures 32 and 33, the elements have biquadratic and bicubic bases, respectively. The interfaces of the mesh with biquadratic elements are C^1 while the mesh with bicubic elements has C^2 interfaces.

The control points and boundaries of the elements are shown in part (a) of each figure. It can be seen from the element boundaries that the basis is capable of representing a linear map. The basis function associated with the control point marked with a filled black circle is shown in parts (b-d) of each figure. Part (b) shows the indices of the nonzero Bernstein coefficients, shaded by the relative amplitude. The index support of each function possesses a notch in the upper-right hand corner of the support that would not be present in a tensor-product construction. The contour plot in part (c) of each figure exhibits an indentation due to the notch in the index support. This is most apparent in the linear case shown in Figure 31. Again each function clearly cannot be produced by a tensor product. Part (d) of each figure shows the surface produced by elevating just the control point associated with the highlighted basis function.

An example to illustrate one potential application of the ability to mix elements of differing polynomial degrees is now given. The polynomial degree of each element in the mesh shown in part (b) of Figure 34 can be

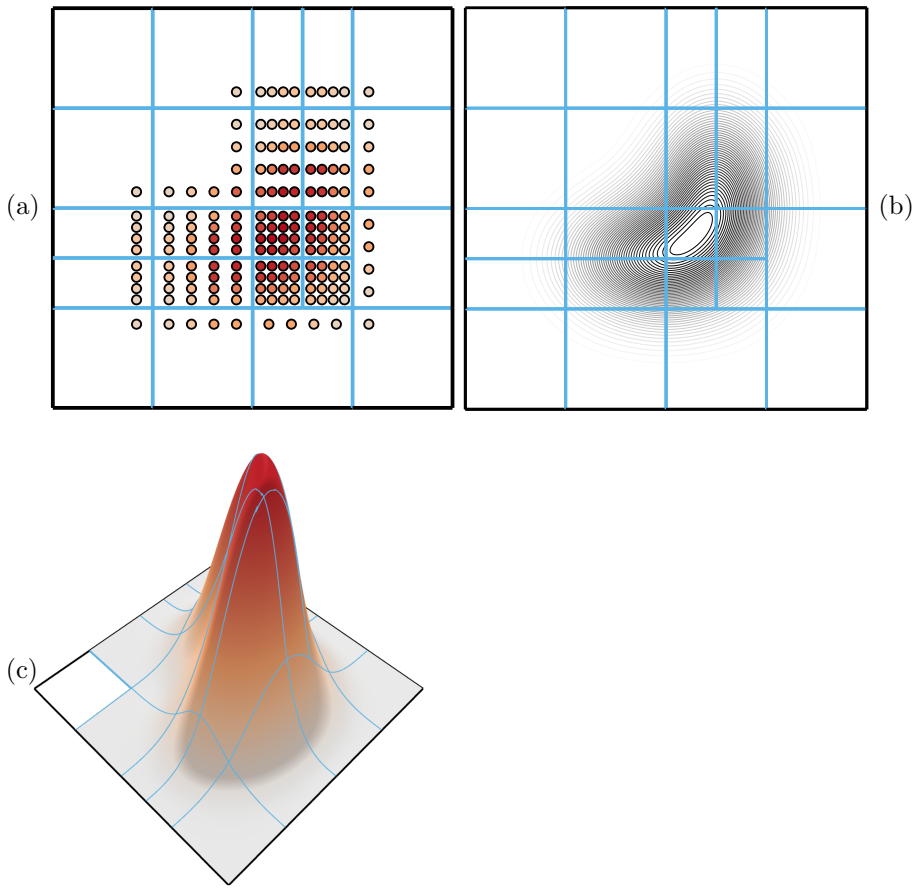

Figure 30: A single U-spline basis function. Part (a) shows the nonzero Bernstein-Bézier coefficients, part (b) shows a contour plot of the function while part (c) shows a three-dimensional plot.

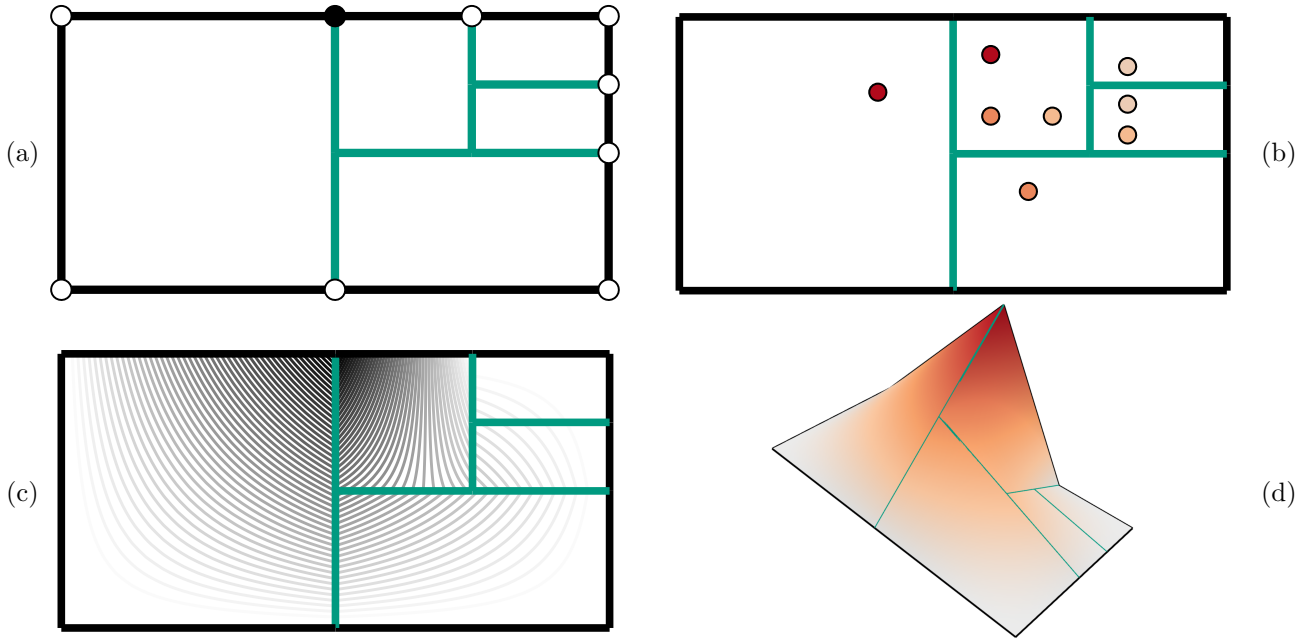

Figure 31: An example basis function from a Bézier mesh containing nested T-junctions. Each element has a basis of polynomial degree 1. The control points of all functions are shown in part (a) with the control point of one function marked. Parts (b-d) show the values of the Bernstein coefficients, a contour plot, and a three dimensional plot of the function, respectively.

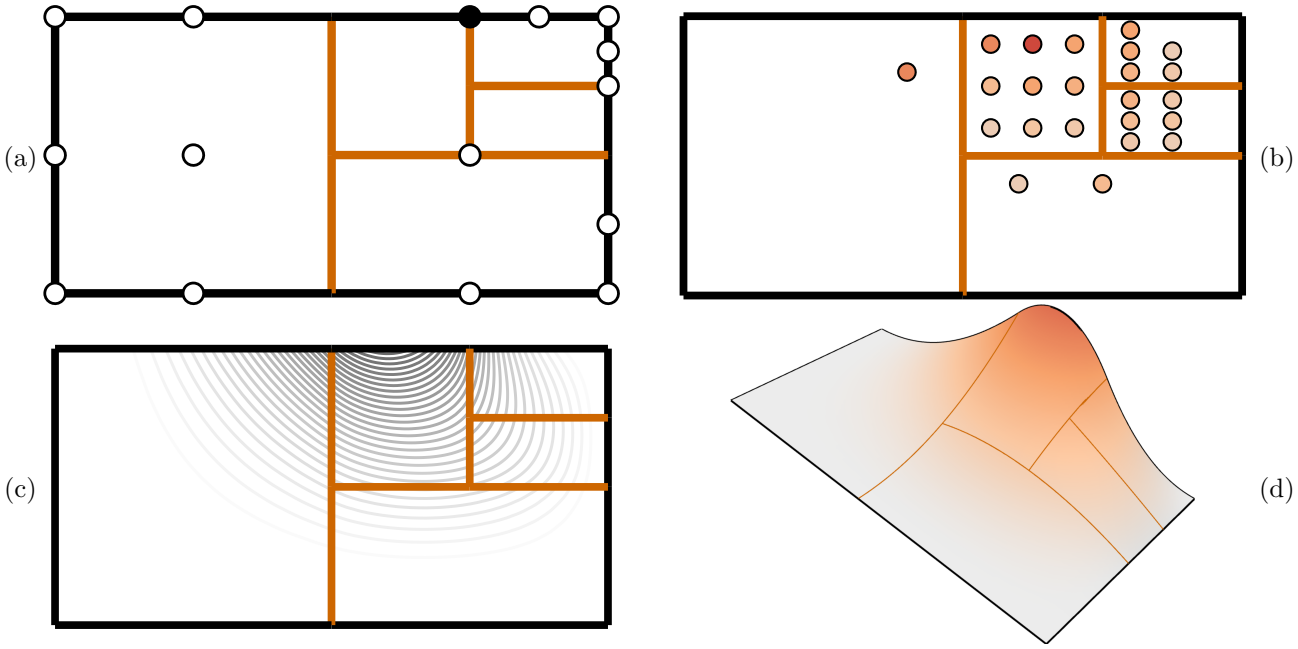

Figure 32: An example basis function from a mesh containing nested T-junctions. Each element has a basis of polynomial degree 2. The control points of all functions are shown in part (a) with the control point of one function marked. Parts (b-d) show the values of the Bernstein coefficients, a contour plot, and a three dimensional plot of the function, respectively.

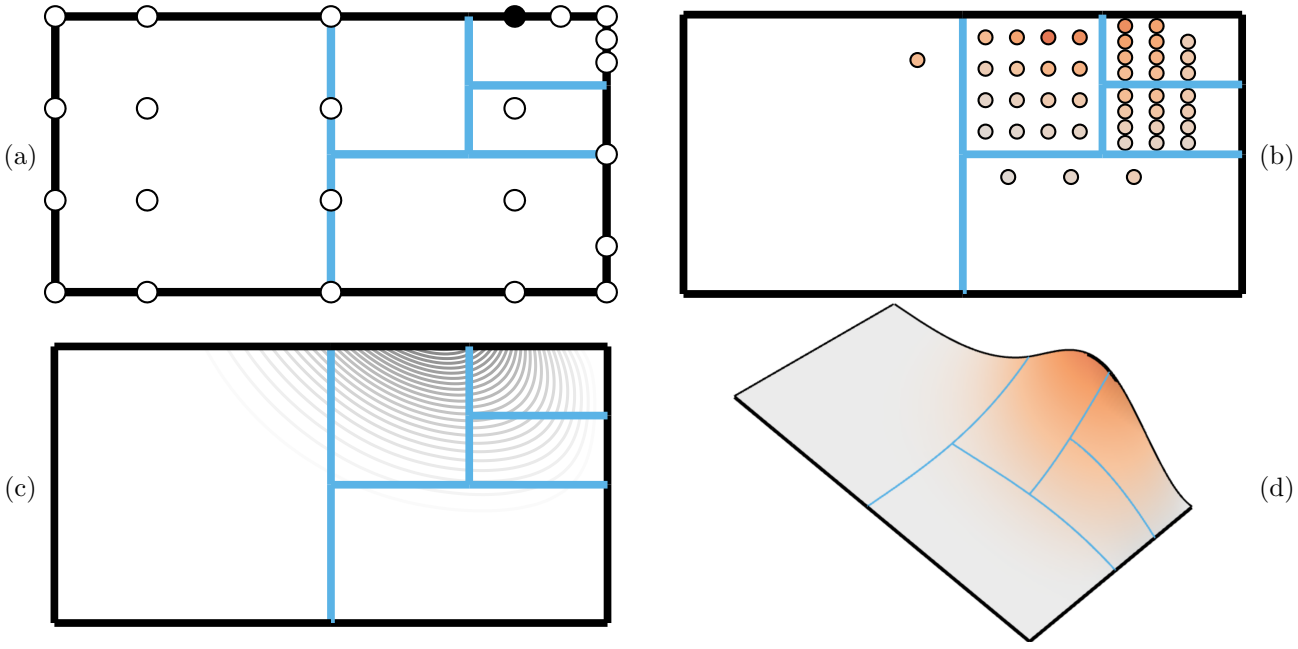

Figure 33: An example basis function from a mesh containing nested T-junctions. Each element has a basis of polynomial degree 3. The control points of all functions are shown in part (a) with the control point of one function marked. Parts (b-d) show the values of the Bernstein coefficients, a contour plot, and a three dimensional plot of the function, respectively.

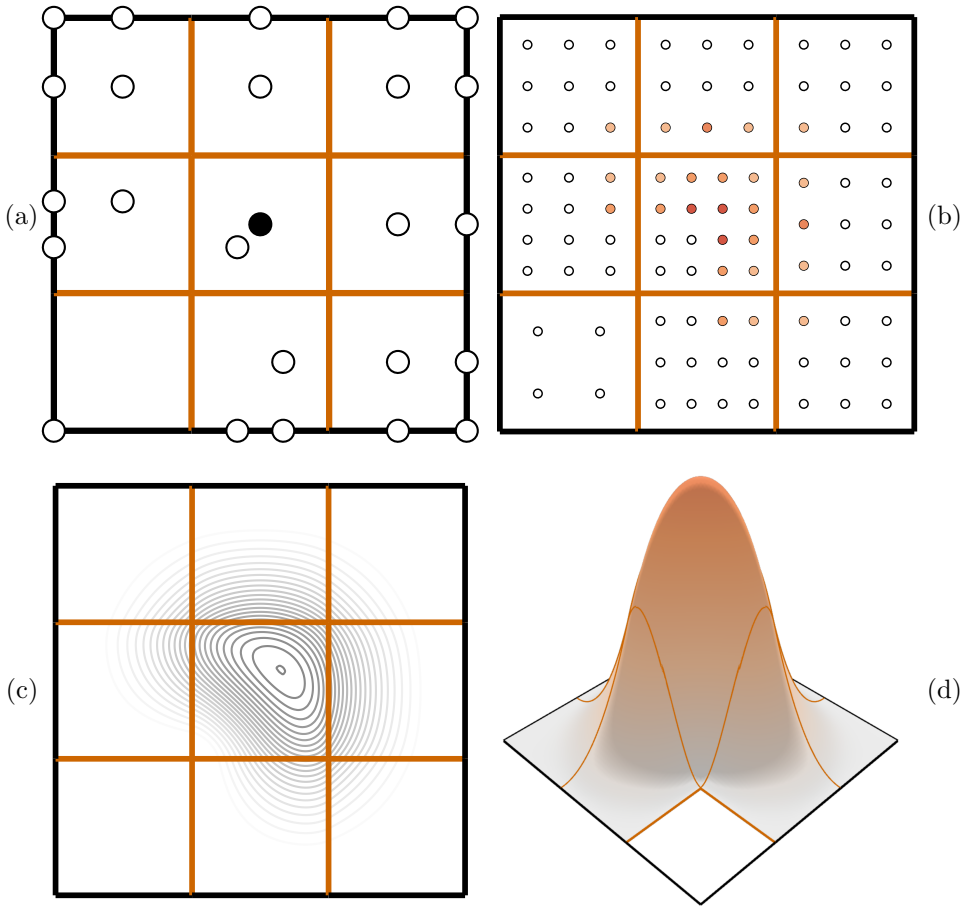

Figure 34: Example of a single basis function defined over a mesh of mixed polynomial degree. The control points producing a linear parameterization are shown in part (a). The control point corresponding to the highlighted function is highlighted. The index support and coefficients (b), contour plot (c), and three-dimensional surface (d) of the function are shown.

determined from the number of dots drawn on each element. The degree in each direction is one less than the number of dots. It can be seen that the mesh has one bilinear element, 5 biquadratic elements, one cubic element, one element of degree $(2, 3)$ and one of degree $(3, 2)$. All interfaces are C^1. The bilinear element is joined smoothly to the adjacent elements and so were it not for the increase in degree in the elements immediately adjacent to the bilinear element, the linear functions would impact functions defined over elements not immediately adjacent to the lower degree element. The introduction of the cubic region isolates the bilinear element from all other elements in the mesh other than those that are immediately adjacent. This is a generally useful technique to isolate local features.

All of the control points that define a linearly parameterized geometry for the basis are shown in part (a) of Figure 34. The basis function highlighted corresponds to the control point marked with a filled dot. The nonzero coefficients indicated by filled dots in part (b) of the figure form an L. This pattern is reflected in the basis function. The contour plot shown in part (c) and the three-dimensional surface plot of the function in part (d) both clearly exhibit an L shape. This is another instance in which the U-spline construction produces functions that cannot be produced by other methods. It should also be noted that the function is formed from a mixture of smoothly joined quadratic and cubic basis functions.

Another function from the same basis is shown in Figure 35. The highlighted function corresponding to the filled dot in part (a) is chosen to illustrate the smooth transition between the bilinear and higher degree elements in the U-spline.

Figure 36 shows a simple mesh consisting of both triangular and quadrilateral elements. All elements have polynomial degree 2. The edges of the triangle are C^0 while all other edges are C^1. The highlighted function spans both element types. The C^1 transitions on all edges not shared with triangles are apparent.

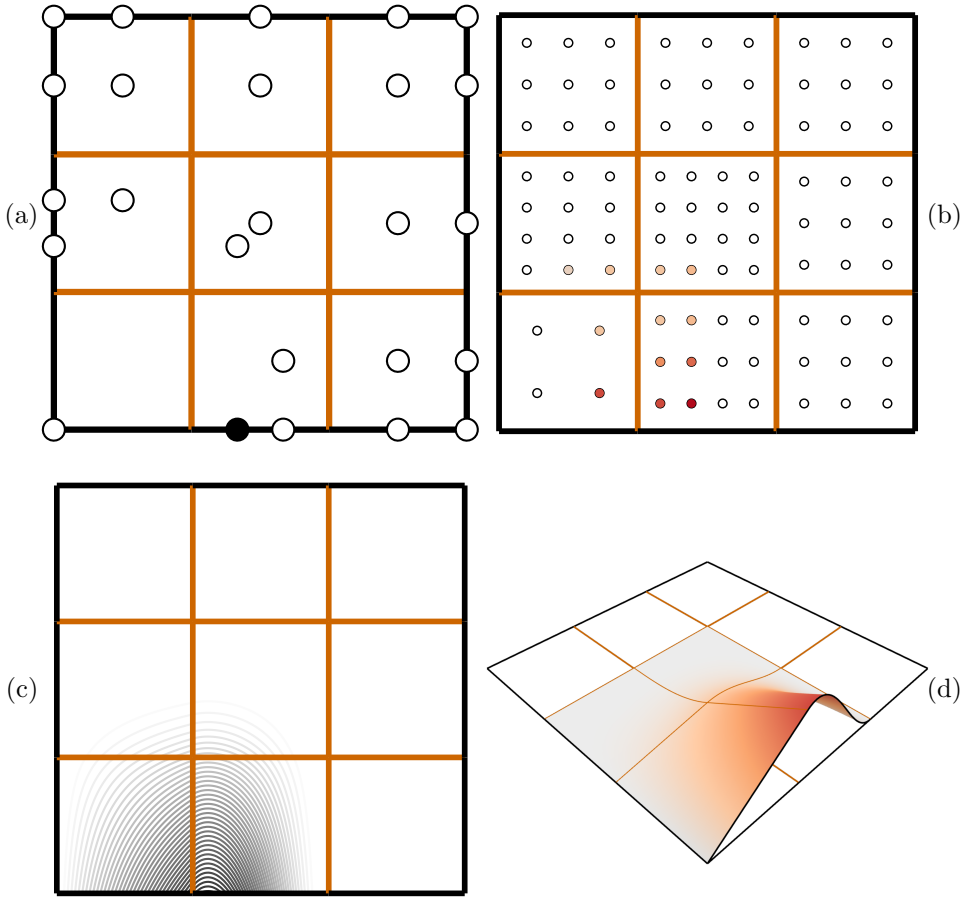

Figure 35: Example of a single basis function defined over a mesh of mixed polynomial degree. The control points producing a linear parameterization are shown in part (a). The control point corresponding to the highlighted function is highlighted. The index support and coefficients (b), contour plot (c), and three-dimensional surface (d) of the function are shown.

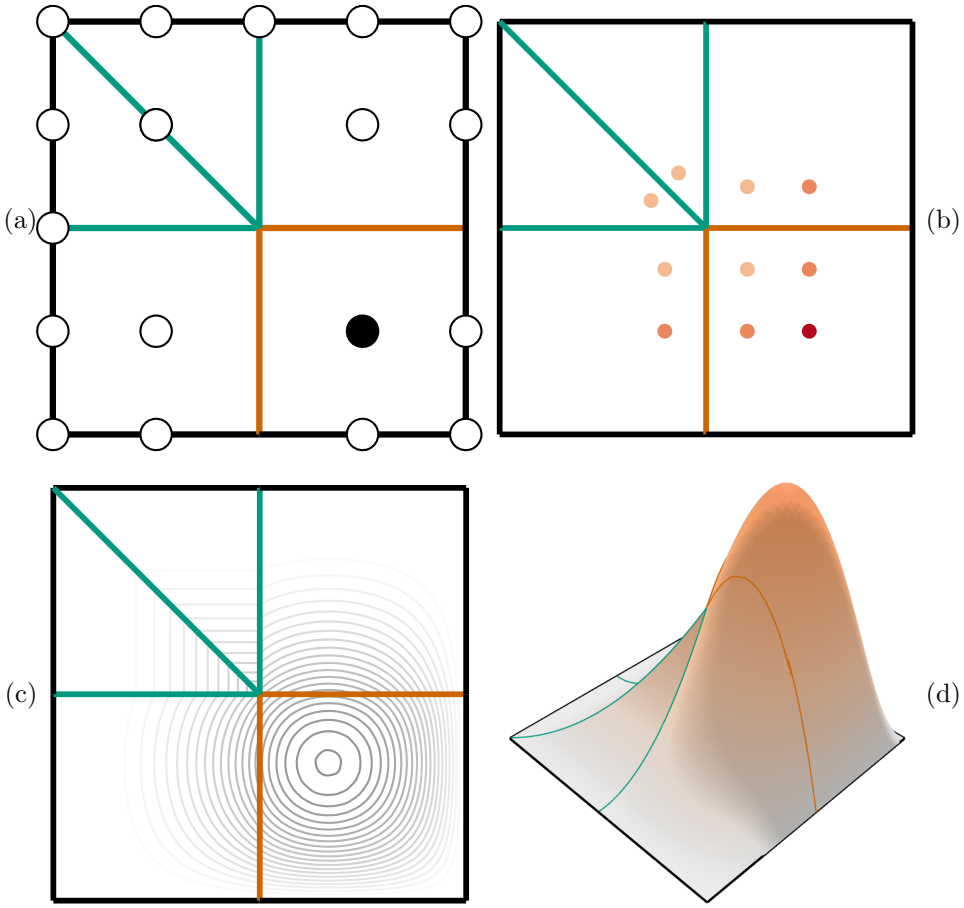

Figure 36: An example basis function from a mesh having both triangular and quadrilateral elements.

9. Conclusion

This paper describes an algorithm to construct smooth spline bases over unstructured meshes. We call the resulting splines U-splines. The U-spline construction relies on the introduction of an atomic unit for the construction of basis functions derived from the local continuity constraints on the Bernstein-Bézier form of the local basis. These units are referred to here as constrained index blocks. The construction of these units relies only on the properties of a Bernstein-like basis on each element. A basis function is constructed by determining a minimally coupled set of constrained index blocks. The members of this set represent the nonzero coefficients of a single basis function. The numerical values of the corresponding coefficients can be solved through linear programming. We conjecture that for appropriate meshes, the spline basis functions constructed using this method have minimal support in the number of nonzero Bernstein coefficients. This reduces the sparsest null-space problem associated with determining the minimally supported spline basis from a NP-hard global problem to a series of local linear programming problems. The determination of the nonzero coefficients relies only on the mesh topology, the specified interelement contiuity, and the dimension of the local Bernstein basis on each element.

Previous constraint based approaches such as polynomial splines over T-meshes have restricted the local degree of elements to be greater than twice the interelement continuity [19, 64]. This reduces the support and coupling of the basis functions but also prevents the construction of basis functions possessing maximal continuity. The U-spline algorithm enables the construction of bases with maximal continuity over more complex meshes.

The flexibility with which continuity, local basis, and scale may be mixed in the U-spline construction while still producing a well-defined basis is unprecedented and provides significant new opportunities in design, analysis, optimzation, and beyond.

9.1. Future work

The use of constrained index blocks is a novel approach to determining the nonzero Bernstein coefficients for explicit construction of basis functions. It remains to be seen if this perspective will provide new avenues for the explicit determination of spline basis dimension.

It has been observed that spline basis dimension can be sensitive to the relative scale of local mesh features [35]. We conjecture that in these settings the U-spline algorithm will produce a basis for the lower dimensional space. The U-spline construction is purely topological and hence cannot detect these features.

A proof of the uniqueness of the basis functions produced on arbitrary meshes has not been attempted. We have not observed any degenerate configurations in our research.

Another area in which the U-spline approach may yield significant advances is the construction of spline bases on triangular or tetrahedral meshes. Only the C^0 case was considered here but the basic philosophy of the constrained index block is sufficiently general to be applied to higher continuity interfaces.

We also conjecture that the unique corner property of the U-spline basis can be connected to both global linear independence and polynomial completeness of the basis.

9.2. Benefits and applications enabled by U-splines

9.2.1. U-spline shape representations

U-splines are an efficient and robust representation of shape due to their unprecedented flexibility and mathematical precision. This is especially true in the context of CAD data. U-splines possess the precision of NURBS, the current CAD standard, with far more capability to represent complex geometry and topology in a watertight, mathematically rigorous fashion. No superfluous design parameters are required in U-spline CAD due to the ability to perform geometrically exact element subdivision, change the degree of collections of faces, and change the smoothness of edges.

9.2.2. U-spline CAE technologies

U-splines can be used directly in CAE applications because the underlying basis is believed to be analysis-suitable. The unique properties of the U-spline basis, such as smoothness, enable more robust, accurate, and efficient simulation results than traditional approaches to CAE, such is finite element analysis (FEA). Introducing exact U-spline geometry into simulation also improves simulation behavior since a faceted approximation is replaced by the smooth exact CAD geometry.

9.2.3. CAD-CAE integration

The integration of U-splines into existing CAD and CAE software, as well as the creation of new U-spline-based CAD-CAE software, has the potential to substantially address and eliminate the following serious inefficiencies:

- Preparing an entire automobile for a crash simulation is extremely expensive, requiring many weeks in manual labor to convert the CAD data and prepare the simulation mesh for analysis, and results in the CAE and CAD data getting out of sync.
- In the aerospace, defense, and automotive industries, nearly 80% of simulation time is spent converting and preparing the CAD geometry (Hardwick and Clay, cited on page 3 of Cottrell et al. [61]).
- In smaller firms that don't employ simulation experts, products are commonly over-engineered or improperly engineered because of the complexity and limitations of getting data into CAE software, which prevents its proper use in the design process.
- Even expert CAE engineers run into limitations when accuracy is lost as they convert exact CAD data to faceted CAE meshes; this especially limits the possibility of exploiting new manufacturing processes like generative design and additive manufacturing.

The result of these inefficiences is that hundreds of millions of dollars are wasted annually across key global industries like automotive, aerospace, and defense due to the difficulty in transitioning CAD data to CAE software to run simulations [65]. Attempts to eliminate this data translation problem in the past have resulted in limited simulation capability or created an inferior process.

In contrast, U-splines have an underlying mathematical formulation which makes it possible to use a U-spline basis for both design and simulation, resulting in a completely integrated IGA approach.

9.2.4. Topology optimization, shape optimization, and generative design

Structural design based on topology and shape optimization and other generative design techniques are becoming increasingly important since they are capable of producing optimal, lightweight structures that can be manufactured using three-dimensional printing techniques. However, the resulting designs are often complex organic shapes represented as a dense triangulation that is not suitable for CAD. The process of turning the triangulation into a CAD object is a manual, error-prone, labor intensive process which limits the practical application of the approach. U-splines have the potential to significantly improve this situation because they are the first CAD representation which is flexible enough to be used directly in the optimization framework and as the output CAD format. As a result, all data translation steps can be eliminated and the output U-spline CAD can be either converted back into a traditional CAD format based on NURBS or taken directly as the CAD object.

The small number of parameters required to represent smooth shapes using U-splines is also advantageous in optimzation problems as it has the potential to dramatically reduce the size of the system. It is also guaranteed that the smoothness of the input shape will be preserved.

9.2.5. Volumetric data representation

There are many applications in which data needs to be representated throughout the volume rather than just on the surface or boundary. U-splines provide an efficient representation of the internal region of an object and thus enable true volumetric representation of data.

New additive manufacturing techniques make it possible to design and manufacture complex parts with non-uniform material composition. Traditional CAD B-reps can only describe the outside envelope of a part and assume uniform material composition. Techniques such as B-rep slicing can be used to extend the amount of internal material variance possible within a B-rep solid, but still place strict limitations on how detailed the material layout can be. In contrast to B-rep CAD, the precise mathematical definition of U-splines can be extended to three-dimensional volumetric representations which can then be tailored through local adaptivity to capture complex material composition in a precise manner.

10. Acknowledgements

The authors would like to acknowledge Matt Sederberg for his support in editing and improving this work.

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0017051. Support was also provided by the Department of Defense, Navy, Contract: N68335-18-C-0289. Funding was also provided by Honeywell Federal Manufacturing & Technologies under Contract No. DE-NA0002839 with the U.S. Department of Energy. Additional support was provided by a grant from the Utah Science Technology and Research Initiative Technology Acceleration Program.

This work was supported by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] G. Strang, Piecewise polynomials and the finite element method, Bulletin of the American Mathematical Society 79 (6) (1973) 1128–1137, ISSN 0002-9904, 1936-881X, doi:10.1090/S0002-9904-1973-13351-8, URL https://www.ams.org/home/page/.

[2] F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: a new paradigm for thin-shell finite-element analysis, International Journal for Numerical Methods in Engineering 47 (12) (2000) 2039–2072, ISSN 1097-0207, doi:10.1002/(SICI)1097-0207(20000430)47:12¡2039::AID-NME872¿3.0.CO;2-1.

[3] K. Höllig, Finite Element Methods with B-Splines, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, ISBN 978-0-89871-699-3, doi:10.1137/1.9780898717532, URL https://epubs.siam.org/doi/book/10.1137/1.9780898717532, 2003.

[4] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39) (2005) 4135–4195, ISSN 0045-7825, doi:10.1016/j.cma.2004.10.008, URL http://www.sciencedirect.com/science/article/pii/S0045782504005171.

[5] J. A. Cottrell, A. Reali, Y. Bazilevs, T. J. R. Hughes, Isogeometric analysis of structural vibrations, Computer Methods in Applied Mechanics and Engineering 195 (41–43) (2006) 5257–5296, ISSN 0045-7825, doi:10.1016/j.cma.2005.09.027, URL http://www.sciencedirect.com/science/article/pii/S0045782505005451.

[6] S. Morganti, F. Auricchio, D. J. Benson, F. I. Gambarin, S. Hartmann, T. J. R. Hughes, A. Reali, Patient-specific isogeometric structural analysis of aortic valve closure, Computer Methods in Applied Mechanics and Engineering 284 (2015) 508–520, ISSN 0045-7825, doi:10.1016/j.cma.2014.10.010, URL http://www.sciencedirect.com/science/article/pii/S0045782514003806.

[7] C. Garoni, C. Manni, S. Serra-Capizzano, D. Sesana, H. Speleers, Spectral analysis and spectral symbol of matrices in isogeometric Galerkin methods, Mathematics of Computation 86 (305) (2017) 1343–1373, ISSN 0025-5718, 1088-6842, doi:10.1090/mcom/3143, URL https://www.ams.org/home/page/.

[8] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, K. U. Bletzinger, Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures, Computer Methods in Applied Mechanics and Engineering 284 (2015) 401–457, ISSN 0045-7825, doi:10.1016/j.cma.2014.09.033, URL http://www.sciencedirect.com/science/article/pii/S0045782514003569.

[9] B. Juttler, A. Mantzaflaris, R. Perl, M. Rumpf, On numerical integration in isogeometric subdivision methods for PDEs on surfaces, Computer Methods in Applied Mechanics and Engineering 302 (2016) 131–146, ISSN 0045-7825, doi:10.1016/j.cma.2016.01.005, URL http://www.sciencedirect.com/science/article/pii/S0045782516000074.

[10] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 229–263, ISSN 0045-7825.

[11] G. Farin, J. Hoschek, M.-S. Kim, Handbook of Computer Aided Geometric Design, Elsevier, ISBN 978-0-08-053340-7, google-Books-ID: GGXcUS5p2CIC, 2002.

[12] D. Rogers, An Introduction to NURBS: With Historical Perspective, The Morgan Kaufmann Series in Computer Graphics, Elsevier Science, ISBN 978-0-08-050920-4, URL https://books.google.com/books?id=MaW4XiScJ7cC, 2000.

[13] L. T. Tenek, J. Argyris, A brief history of FEM, in: L. T. Tenek, J. Argyris (Eds.), Finite Element Analysis for Composite Structures, Solid Mechanics and Its Applications, Springer Netherlands, Dordrecht, ISBN 978-94-015-9044-0, 17–25, doi:10.1007/978-94-015-9044-0-2, 1998.

[14] R. W. Clough, Early history of the finite element method from the view point of a pioneer, International Journal for Numerical Methods in Engineering 60 (1) (2004) 283–287, ISSN 1097-0207, doi:10.1002/nme.962.

[15] T. J. R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, ISBN 978-0-486-13502-1, 2012.

[16] J. Peters, U. Reif, Subdivision Surfaces, Geometry and Computing, Springer Berlin Heidelberg, ISBN 978-3-642-09527-6, URL https://books.google.com/books?id=ndaecQAACAAJ, 2010.

[17] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, in: ACM SIGGRAPH 2003 Papers, SIGGRAPH '03, ACM, New York, NY, USA, ISBN 978-1-58113-709-5, 477–484, doi:10.1145/1201775.882295, URL http://doi.acm.org/10.1145/1201775.882295, 2003.

[18] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes, Graphical Models 70 (4) (2008) 76–86, ISSN 1524-0703, doi:10.1016/j.gmod.2008.03.001, URL http://www.sciencedirect.com/science/article/pii/S1524070308000039.

[19] X. Li, J. Deng, F. Chen, Polynomial splines over general T-meshes, The Visual Computer 26 (4) (2010) 277–286, ISSN 0178-2789, 1432-2315, doi:10.1007/s00371-009-0410-9, URL https://link.springer.com/article/10.1007/s00371-009-0410-9.

[20] U. Reif, A Refineable Space of Smooth Spline Surfaces of Arbitrary Topological Genus, Journal of Approximation Theory 90 (2) (1997) 174–199, ISSN 0021-9045, doi:10.1006/jath.1996.3079, URL http://www.sciencedirect.com/science/article/pii/S0021904596930798.

[21] T. Nguyen, J. Peters, Refinable C1 spline elements for irregular quad layout, Computer Aided Geometric Design 43 (2016) 123–130, ISSN 0167-8396, doi:10.1016/j.cagd.2016.02.009, URL http://www.sciencedirect.com/science/article/pii/S0167839616300103.

[22] D. Toshniwal, H. Speleers, T. J. R. Hughes, Smooth cubic spline spaces on unstructured quadrilateral meshes with particular emphasis on extraordinary points: Geometric design and isogeometric analysis considerations, Computer Methods in Applied Mechanics and Engineering 327 (2017) 411–458, ISSN 0045-7825, doi:10.1016/j.cma.2017.06.008, URL http://www.sciencedirect.com/science/article/pii/S0045782517305303.

[23] J. H. Argyris, I. Fried, D. W. Scharpf, The TUBA Family of Plate Elements for the Matrix Displacement Method, The Aeronautical Journal 72 (692) (1968) 701–709, ISSN 0001-9240, 2059-6464, doi:10.1017/S000192400008489X.

[24] X. Li, M. A. Scott, Analysis-suitable T-splines: characterization, refineability, and approximation, Mathematical Models and Methods in Applied Science 24 (06) (2014) 1141–1164.

[25] T. Dokken, T. Lyche, K. F. Pettersen, Polynomial splines over locally refined box-partitions, Computer Aided Geometric Design 30 (3) (2013) 331–356, ISSN 0167-8396, doi:10.1016/j.cagd.2012.12.005, URL http://www.sciencedirect.com/science/article/pii/S0167839613000113.

[26] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: The truncated basis for hierarchical splines, Computer Aided Geometric Design 29 (7) (2012) 485–498, ISSN 0167-8396, doi:10.1016/j.cagd.2012.03.025, URL http://www.sciencedirect.com/science/article/pii/S0167839612000519.

[27] M. A. Scott, D. C. Thomas, E. J. Evans, Isogeometric spline forests, Computer Methods in Applied Mechanics and Engineering 269 (2014) 222–264, ISSN 0045-7825, doi:10.1016/j.cma.2013.10.024, URL http://www.sciencedirect.com/science/article/pii/S0045782513002764.

[28] D. Groisser, J. Peters, Matched -constructions always yield -continuous isogeometric elements, Computer Aided Geometric Design 34 (2015) 67–72, ISSN 0167-8396, doi:10.1016/j.cagd.2015.02.002, URL http://www.sciencedirect.com/science/article/pii/S0167839615000151.

[29] M. Kapl, G. Sangalli, T. Takacs, Construction of analysis-suitable G1 planar multi-patch parameterizations, Computer-Aided Design 97 (2018) 41–55, ISSN 0010-4485, doi:10.1016/j.cad.2017.12.002, URL http://www.sciencedirect.com/science/article/pii/S0010448517302439.

[30] P. Alfeld, L. L. Schumaker, Smooth macro-elements based on Clough-Tocher triangle splits, Numerische Mathematik 90 (4) (2002) 597–616, ISSN 0945-3245, doi:10.1007/s002110100304, URL https://doi.org/10.1007/s002110100304.

[31] H. Speleers, C. Manni, F. Pelosi, From NURBS to NURPS geometries, Computer Methods in Applied Mechanics and Engineering 255 (2013) 238–254, ISSN 0045-7825, doi:10.1016/j.cma.2012.11.012, URL http://www.sciencedirect.com/science/article/pii/S0045782512003507.

[32] H. Speleers, Construction of Normalized B-Splines for a Family of Smooth Spline Spaces Over Powell–Sabin Triangulations, Constructive Approximation 37 (1) (2012) 41–72, ISSN 0176-4276, 1432-0940, doi:10.1007/s00365-011-9151-x, URL http://link.springer.com/article/10.1007/s00365-011-9151-x.

[33] M.-J. Lai, L. L. Schumaker, Spline Functions on Triangulations, Cambridge University Press, ISBN 978-0-521-87592-9, google-Books-ID: 6hvqGgbBmEoC, 2007.

[34] P. Alfeld, Bivariate spline spaces and minimal determining sets, Journal of Computational and Applied Mathematics 119 (1–2) (2000) 13–27, ISSN 0377-0427, doi:10.1016/S0377-0427(00)00369-1, URL http://www.sciencedirect.com/science/article/pii/S0377042700003691.

[35] X. Li, F. Chen, On the instability in the dimension of splines spaces over T-meshes, Computer Aided Geometric Design 28 (7) (2011) 420–426, ISSN 0167-8396, doi:10.1016/j.cagd.2011.08.001, URL http://www.sciencedirect.com/science/article/pii/S0167839611000896.

[36] L. L. Schumaker, L. Wang, Spline spaces on TR-meshes with hanging vertices, Numerische Mathematik 118 (3) (2011) 531–548, ISSN 0945-3245, doi:10.1007/s00211-010-0353-0, URL https://doi.org/10.1007/s00211-010-0353-0.

[37] L. L. Schumaker, L. Wang, Spline spaces on TR-meshes with hanging vertices, Numerische Mathematik 118 (3) (2011) 531–548, ISSN 0945-3245, doi:10.1007/s00211-010-0353-0, URL https://doi.org/10.1007/s00211-010-0353-0.

[38] M. Neamtu, Delaunay configurations and multivariate splines: A generalization of a result of B. N. Delaunay, Transactions of the American Mathematical Society 359 (7) (2007) 2993–3004, ISSN 0002-9947, 1088-6850, doi:10.1090/S0002-9947-07-03976-1, URL http://www.ams.org/tran/2007-359-07/S0002-9947-07-03976-1/.

[39] I. H. Stangeby, Simplex Splines on the Powell-Sabin 12-Split URL https://www.duo.uio.no/handle/10852/64070.

[40] T. Lyche, G. Muntingh, Simplex Spline Bases on the Powell-Sabin 12-Split: Part I, Oberwolfach Reports 12 (2) (2015) 1139–1200, ISSN 1660-8933, doi:10.4171/OWR/2015/21, URL http://arxiv.org/abs/1505.01798, arXiv: 1505.01798.

[41] T. Lyche, G. Muntingh, Simplex Spline Bases on the Powell-Sabin 12-Split: Part II, Oberwolfach Reports 12 (2) (2015) 1139–1200, ISSN 1660-8933, doi:10.4171/OWR/2015/21, URL http://arxiv.org/abs/1505.01801, arXiv: 1505.01801.

[42] E. Cohen, T. Lyche, R. Riesenfeld, A B-spline-like basis for the Powell-Sabin 12-split based on simplex splines, Mathematics of Computation 82 (283) (2013) 1667–1707, ISSN 0025-5718, 1088-6842, doi:10.1090/S0025-5718-2013-02664-6, URL https://www.ams.org/home/page/.

[43] G. Awanou, M.-j. Lai, P. Wenston, The Multivariate Spline Method for Scattered Data Fitting . . ., 2005.

[44] X. Hu, D. Han, M. Lai, Bivariate Splines of Various Degrees for Numerical Solution of Partial Differential Equations, SIAM Journal on Scientific Computing 29 (3) (2007) 1338–1354, ISSN 1064-8275, doi:10.1137/060667207, URL http://epubs.siam.org/doi/abs/10.1137/060667207.

[45] L. Schumaker, Spline Functions: Basic Theory, Cambridge University Press, ISBN 978-0-521-70512-7, 2007.

[46] F. Pelosi, C. Giannelli, C. Manni, M. L. Sampoli, H. Speleers, Splines over regular triangulations in numerical simulation, Computer-Aided Design 82 (2017) 100–111, ISSN 0010-4485, doi:10.1016/j.cad.2016.08.002, URL http://www.sciencedirect.com/science/article/pii/S0010448516300902.

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

[48] S. Xia, X. Wang, X. Qian, Continuity and convergence in rational triangular Bézier spline based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 297 (2015) 292–324, ISSN 0045-7825, doi:10.1016/j.cma.2015.09.001, URL http://www.sciencedirect.com/science/article/pii/S0045782515002777.

[49] S. Xia, X. Qian, Isogeometric analysis with Bézier tetrahedra, Computer Methods in Applied Mechanics and Engineering 316 (2017) 782–816, ISSN 0045-7825, doi:10.1016/j.cma.2016.09.045, URL http://www.sciencedirect.com/science/article/pii/S0045782516312774.

[50] L. Demkowicz, J. T. Oden, W. Rachowicz, O. Hardy, Toward a universal h-p adaptive finite element strategy, part 1. Constrained approximation and data structure, Computer Methods in Applied Mechanics and Engineering 77 (1) (1989) 79–112, ISSN 0045-7825, doi:10.1016/0045-7825(89)90129-1, URL http://www.sciencedirect.com/science/article/pii/0045782589901291.

[51] T. W. Sederberg, J. Zheng, X. Song, Knot intervals and multi-degree splines, Computer Aided Geometric Design 20 (7) (2003) 455–468, ISSN 0167-8396, doi:10.1016/S0167-8396(03)00096-7, URL http://www.sciencedirect.com/science/article/pii/S0167839603000967.

[52] W. Shen, G. Wang, A basis of multi-degree splines, Computer Aided Geometric Design 27 (1) (2010) 23–35, ISSN 0167-8396, doi:10.1016/j.cagd.2009.08.005, URL http://www.sciencedirect.com/science/article/pii/S0167839609000946.

[53] D. Toshniwal, H. Speleers, R. R. Hiemstra, T. J. R. Hughes, Multi-degree smooth polar splines: A framework for geometric modeling and isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1005–1061, ISSN 0045-7825, doi:10.1016/j.cma.2016.11.009, URL http://www.sciencedirect.com/science/article/pii/S004578251631533X.

[54] J. Peters, Splines and unsorted knot sequences, Computer Aided Geometric Design 30 (7) (2013) 733–741, ISSN 0167-8396, doi:10.1016/j.cagd.2013.06.001, URL http://www.sciencedirect.com/science/article/pii/S016783961300054X.

[55] R. T. Farouki, The Bernstein polynomial basis: A centennial retrospective, Computer Aided Geometric Design 29 (6) (2012) 379–419, ISSN 0167-8396, doi:10.1016/j.cagd.2012.03.001, URL http://www.sciencedirect.com/science/article/pii/S0167839612000192.

[56] M. Campen, D. Zorin, Similarity Maps and Field-guided T-splines: A Perfect Couple, ACM Trans. Graph. 36 (4) (2017) 91:1–91:16, ISSN 0730-0301, doi:10.1145/3072959.3073647, URL http://doi.acm.org/10.1145/3072959.3073647.

[57] I. J. SCHOENBERG, CONTRIBUTIONS TO THE PROBLEM OF APPROXIMATION OF EQUIDISTANT DATA BY ANALYTIC FUNCTIONS: PART A.—ON THE PROBLEM OF SMOOTHING OR GRADUATION. A FIRST CLASS OF ANALYTIC APPROXIMATION FORMULAE, Quarterly of Applied Mathematics 4 (1) (1946) 45–99, ISSN 0033-569X, URL https://www.jstor.org/stable/43633538.

[58] I. J. SCHOENBERG, CONTRIBUTIONS TO THE PROBLEM OF APPROXIMATION OF EQUIDISTANT DATA BY ANALYTIC FUNCTIONS: PART B—ON THE PROBLEM OF OSCULATORY INTERPOLATION. A SECOND CLASS OF ANALYTIC APPROXIMATION FORMULAE, Quarterly of Applied Mathematics 4 (2) (1946) 112–141, ISSN 0033-569X, URL https://www.jstor.org/stable/43633544.

[59] M. G. Cox, The Numerical Evaluation of B-Splines, IMA Journal of Applied Mathematics 10 (2) (1972) 134–149, ISSN 0272-4960, doi:10.1093/imamat/10.2.134, URL https://academic.oup.com/imamat/article/10/2/134/687696.

[60] C. de Boor, On calculating with B-splines, Journal of Approximation Theory 6 (1) (1972) 50–62, ISSN 0021-9045, doi:10.1016/0021-9045(72)90080-9, URL http://www.sciencedirect.com/science/article/pii/0021904572900809.

[61] J. Cottrell, T. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, Wiley, ISBN 978-0-470-74909-8, URL https://books.google.com/books?id=9Q9y0Xtz5E4C, 2009.

[62] T. Coleman, A. Pothen, The Null Space Problem I. Complexity, SIAM Journal on Algebraic Discrete Methods 7 (4) (1986) 527–537, ISSN 0196-5212, doi:10.1137/0607059, URL http://epubs.siam.org/doi/abs/10.1137/0607059.

[63] T. Coleman, A. Pothen, The Null Space Problem II. Algorithms, SIAM Journal on Algebraic Discrete Methods 8 (4) (1987) 544–563, ISSN 0196-5212, doi:10.1137/0608045, URL http://epubs.siam.org/doi/abs/10.1137/0608045.

[64] L. L. Schumaker, L. Wang, Approximation power of polynomial splines on T-meshes, Computer Aided Geometric Design 29 (8) (2012) 599–612, ISSN 0167-8396, doi:10.1016/j.cagd.2012.04.003, URL http://www.sciencedirect.com/science/article/pii/S0167839612000556.

[65] S. B. Brunnermeier, S. A. Martin, Interoperability Cost Analysis of the U.S. Automotive Supply Chain. National Institute of Standards and Technology., NIST Planning Report 99-1 URL http://www.nist.gov/director/prog-ofc/report99-1.pdf].