

Blended B-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis

Xiaodong Wei^a, Yongjie Jessica Zhang^{a,*}, Deepesh Toshniwal^b, Hendrik Speleers^c,
Xin Li^d, Carla Manni^c, John A. Evans^e, Thomas J.R. Hughes^b

^a Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

^b Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA

^c Department of Mathematics, University of Rome “Tor Vergata”, Rome, 00133, Italy

^d School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui, China

^e Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO 80309, USA

Received 20 November 2017; received in revised form 4 May 2018; accepted 11 July 2018

Available online 17 July 2018

Abstract

We present a novel blended B-spline method to construct bicubic/tricubic splines over unstructured quadrilateral and hexahedral meshes for isogeometric analysis. C^1 and (truncated) C^2 B-spline functions are used in regular elements, whereas C^0 and (truncated) C^1 B-spline functions are adopted in boundary elements and interior irregular elements around extraordinary edges/vertices. The truncation mechanism is employed for a seamless transition from irregular to regular elements. The resulting smoothness of the blended construction is C^2 -continuous everywhere except C^0 -continuous around extraordinary edges and C^1 -continuous across the interface between irregular and regular elements. The blended B-spline construction yields consistent parameterization during refinement and exhibits optimal convergence rates. Spline functions in the blended construction form a non-negative partition of unity, are linearly independent, and support Bézier extraction such that the construction can be used in existing finite element frameworks. Several examples provide numerical evidence of optimal convergence rates.

© 2018 Elsevier B.V. All rights reserved.

Keywords: Blended B-spline construction; Extraordinary vertices/edges; Optimal convergence rates; Unstructured quadrilateral and hexahedral meshes; Mixed smoothness; Isogeometric analysis

1. Introduction

Isogeometric analysis (IGA) was introduced to bridge the gap between computer-aided design (CAD) and traditional finite element analysis (FEA) by utilizing the same basis of a CAD representation in analysis [1,2]. In

addition to CAD representations, unstructured quadrilateral (quad) and hexahedral (hex) meshes can also serve as important input control meshes¹ for IGA. For instance, many techniques have been developed to convert imaging data to such meshes [3]. Unstructured quad/hex meshes inevitably involve extraordinary vertices/edges. In the interior of a quad/hex mesh, an *extraordinary vertex/edge* is a vertex/edge shared by other than four quad/hex elements, respectively. Endpoints of extraordinary edges in 3D are also extraordinary vertices. Generally, in a hex mesh, there exist 3D extraordinary vertices that cannot be obtained by sweeping 2D counterparts. How to deal with extraordinary vertices/edges is the key to employing unstructured quad/hex meshes in IGA, and developing a spline basis with desired properties, such as non-negative partition of unity, linear independence, smoothness (preferably G^1 or better), nested spline spaces, and exhibiting optimal convergence rates, is a challenge. We are particularly interested in optimal convergence rates in this paper.

While advances have been made for unstructured quad meshes, very few methods [4–6] have been studied for unstructured hex meshes in IGA, and none exhibits optimal convergence rates while maintaining higher-order smoothness. Catmull–Clark subdivision [4,7–9] has often been used for unstructured quad/hex meshes, where a patch near an extraordinary vertex is represented by an infinite series of subpatches of uniform bicubic/tricubic B-splines. Such an infinite representation needs an enormous number of Gauss points in analysis to guarantee integration accuracy [7–10]. However, optimal convergence has not been observed even with accurate integration [7]. Manifold splines and Hermite-type splines were used in IGA and optimal convergence rates were observed [11,12], but manifold splines also require many integration points and Hermite-type splines need additional continuity constraints on derivatives. A template-based C^0 -parameterization method, taking advantage of T-spline local knot vectors, was proposed to convert unstructured quad meshes to watertight T-spline representations [5,13]. However, this method was focused on geometry and convergence behavior was not studied. The multi-patch B-spline/NURBS methods, including C^0 -parameterization [14,15] and G^1 -parameterization methods [16–22], treat the region around an extraordinary vertex in a multi-patch manner. Although optimal convergence rates can be achieved in both C^0 and G^1 constructions, locally splitting an unstructured mesh (especially an unstructured hex mesh) around extraordinary vertices/edges into multiple B-spline/NURBS patches is not trivial, primarily because many extraordinary vertices might be in close proximity. Global refinement can separate such adjacent extraordinary vertices, but it introduces a large number of unnecessary degrees of freedom (DOF). Bézier extraction expresses basis functions around extraordinary vertices as linear combinations of Bernstein polynomials [6,23–29]. In [28], a dynamic weighted refinement scheme was proposed to improve the convergence rate, but optimal convergence rates were still not achieved. With a C^1 -continuous construction in the vicinity of extraordinary points in 2D, two methods were proposed based on degenerated bivariate Bézier patches [30]: one in the context of PHT-splines [31] with emphasis on refinability around extraordinary vertices [24], and the other from the isogeometric analysis point of view [25]. Both methods have achieved optimal convergence rates.

It is far from a trivial task to generalize a 2D method to 3D that can simultaneously build a smooth parameterization and achieve optimal convergence rates. In fact, constructing a smooth parameterization on an unstructured hex mesh is still an open challenging problem, whereas there is no existing method that can achieve optimal convergence rates for unstructured hex meshes even under a parameterization that is C^0 around extraordinary vertices and high-order smooth elsewhere. We focus on the latter problem in this paper. It is straightforward to construct global C^0 parameterization using Bézier basis functions, which, however, is not favored as it engenders too many DOF, especially in 3D. We are interested in building a subspace that yields optimal convergence rates, but with substantially fewer DOF. The challenging problem here is how to construct consistent C^0 parameterization for irregular elements that can be seamlessly connected with regular C^2 parameterization. A *consistent parameterization* maps a given point in the parametric domain to the same point in the physical domain before and after refinement. Extra DOF need to be added in irregular elements to maintain consistent parameterization, but the number needs to be minimized. In this paper, we present a novel blended $C^0/C^1/C^2$ B-spline parameterization method, referred to as the *C012 construction*, on unstructured quad and hex meshes that can achieve optimal convergence rates with minimal extra DOF introduced. Given an unstructured quad/hex mesh as the input control mesh, we distinguish two submeshes: an *irregular submesh* composed of boundary elements as well as irregular elements that contain extraordinary vertices, and a *regular submesh* consisting of regular elements only. We use bicubic/tricubic splines throughout this paper. C^0 and (truncated) C^1 B-spline functions are added in the irregular submesh, whereas C^1 and (truncated) C^2 B-spline functions are

¹ The term “control mesh” implies the odd degree case, where each control point corresponds to a mesh vertex.

adopted in the regular submesh. Across the interface of the two submeshes, the truncation mechanism [32] is used to connect them. The resulting geometry of a blended B-spline representation is C^2 -continuous in the regular subdomain, C^1 -continuous across the two subdomains, and C^0 -continuous in the irregular subdomain. Such a blended construction limits the influence of extraordinary vertices/edges to within their one-ring neighborhoods. Necessary DOF (i.e., Bézier control points) are introduced in the irregular submesh to guarantee consistent parameterization. The refinement scheme of the blended construction is simply the knot insertion algorithm [33,34]. The basis functions in the blended construction form a non-negative partition of unity, are globally linearly independent, and support Bézier extraction. Optimal convergence rates are observed for this construction in all tested examples with unstructured quad/hex meshes.

The remainder of the paper is organized as follows. A review of B-splines and Bézier extraction on unstructured quad and hex meshes is given in Section 2. We then discuss the blended B-spline construction on unstructured quad and hex meshes in Sections 3 and 4, respectively. Several properties of the blended B-spline construction are proved in Section 5. We present in Section 6 numerical examples to verify that the proposed method can achieve optimal convergence rates. Conclusions and future work are presented in Section 7. Two variants of the C012 construction are given in Appendix.

2. Review of B-splines and Bézier extraction on unstructured quad/hex meshes

In this section, we review B-splines and Bézier extraction on unstructured quad/hex meshes. Related details can be found in [2,6,34,35].

2.1. B-splines

We start with a brief review to B-splines. A univariate B-spline is defined on a set of non-decreasing real numbers, the so-called *knot vector* denoted by $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$, where ξ_i is the i th knot, n is the number of B-spline basis functions and p is the polynomial degree. The interval $[\xi_i, \xi_{i+1}]$ ($1 \leq i \leq n+p$) is called a *knot interval*. A knot vector is *uniform* if all the knot intervals have the same length. B-spline basis functions $\{N_{i,p}\}_{i=1}^n$ can be obtained by the Cox-de Boor recursion formula [36], starting from $p = 0$,

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

and for $p \geq 1$, we have

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \quad (2)$$

A B-spline basis function $N_{i,p}$ is non-negative and has a local support in $[\xi_i, \xi_{i+p+1}]$. $N_{i,p}$ is C^∞ -continuous inside knot spans (ξ_{i+k}, ξ_{i+k+1}) and C^{p-m} -continuous across knots with multiplicity m ($m \leq p+1$). Bivariate and trivariate B-spline basis functions are simply tensor products of univariate ones.

In Fig. 1, we show several cubic C^2 and C^1 B-spline basis functions. Later we will use them in our blended B-spline construction. For instance in Fig. 1(a), given a uniform knot vector of degree three, $\{0, 1, \dots, 7\}$, there are four C^2 B-spline basis functions. With another knot vector also of degree three in Fig. 1(c), $\{0, 0, 1, 1, \dots, 7, 7\}$, where every knot is repeated, twelve C^1 B-spline functions are defined. The corresponding C^2 and C^1 functions with support on the knot span $[3, 4]$ are shown in Fig. 1(b, d), respectively. We will only consider bicubic and tricubic splines, so in the rest of the paper $p = 3$ in each parametric direction.

2.2. Bézier extraction on unstructured quad meshes

In Fig. 2, we introduce several terminologies to facilitate our developments. An unstructured quad mesh can be treated as a T-mesh without T-junctions [37], consists of vertices (or control points), edges and elements. The number of elements sharing a vertex is called its *valence*. An interior vertex of valence other than four, as well as a boundary vertex of valence other than two or one, is called an *extraordinary vertex*, where there is no (local) tensor-product mesh structure and special treatment is needed to define spline functions. Edges touching an extraordinary vertex are called *spoke edges*. An element is a boundary element if it has at least

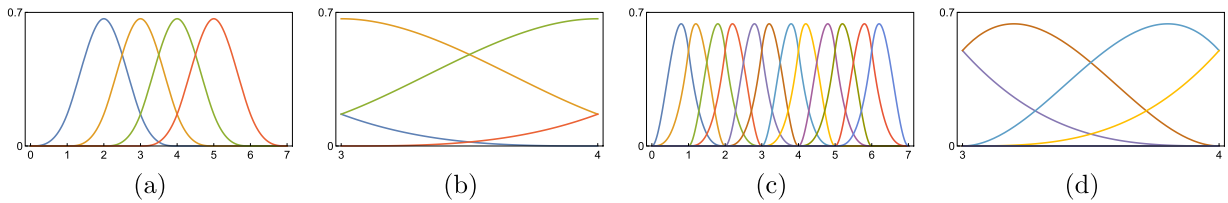


Fig. 1. Cubic C^2 and C^1 B-spline functions in 1D. (a) C^2 B-spline functions defined on a uniform knot vector, (b) the C^2 B-spline functions with support on the knot interval $[3, 4]$, (c) C^1 B-spline functions defined on a knot vector with each knot repeated, and (d) the C^1 B-spline functions with support on the knot interval $[3, 4]$.

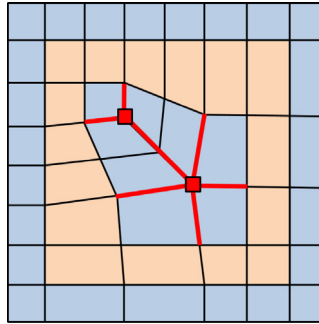


Fig. 2. An unstructured quad mesh with two extraordinary vertices (red squares). Spoke edges are marked in red. Irregular and regular elements are shaded blue and orange, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

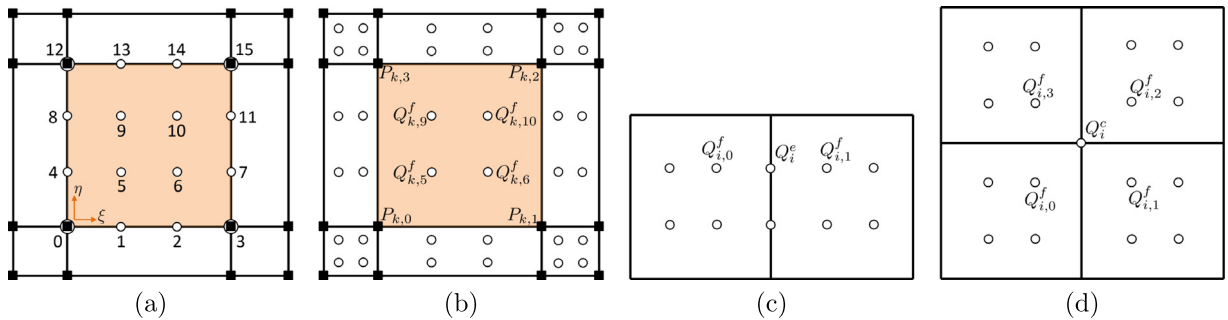


Fig. 3. Bézier extraction in 2D for a regular element (shaded orange). (a) The 16 Bézier control points (open circles) determined from the local mesh (black squares), and face points (b), edge points (c) and corner points (d).

one vertex lying on the boundary, and otherwise it is an interior element. An interior element is an *irregular element* if any of its four vertices is an extraordinary vertex. Otherwise it is a *regular element*. Note that a boundary element may also have certain vertices being extraordinary vertices, but in this paper we do not further distinguish such cases, and instead we treat all the boundary elements as irregular elements to simplify the implementation. An element and its one-ring neighboring elements (i.e., elements sharing vertices with it) form a *local (control) mesh* of the element.

Bézier extraction for an element involves its local mesh. Each edge is assigned with a knot interval, which is used for Bézier extraction and will be detailed in the following. Uniform knot vectors are assumed in this paper to satisfy the condition that knot intervals of opposite edges in each element coincide [37], which is used for Bézier extraction for irregular elements.

We start with an introduction to Bézier extraction on a regular element and later extend the idea to irregular and boundary elements. As shown in Fig. 3(a), Bézier extraction for a regular element Ω_k (shaded orange) involves calculating 16 Bézier control points (open circles) from the vertices (black squares) of its local mesh $\mathcal{M}_k$. These

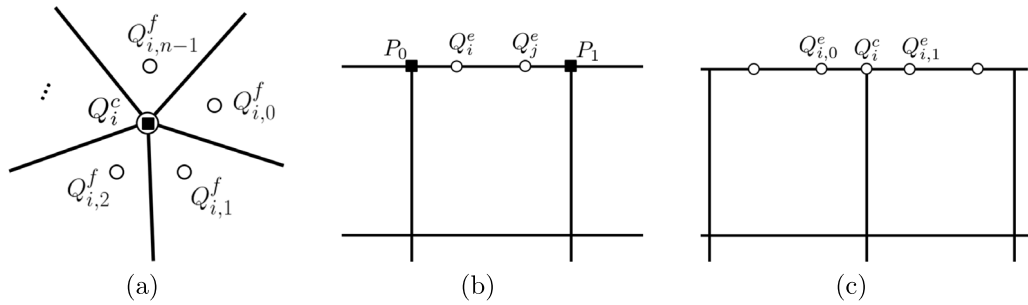


Fig. 4. Calculation of a corner point corresponding to an extraordinary vertex (a), edge points corresponding to a boundary edge (b), and corner points corresponding to a boundary vertex (c).

16 Bézier control points can be divided into three categories: face points $Q_{k,i}^f$ ($i \in \{5, 6, 9, 10\}$), edge points $Q_{k,i}^e$ ($i \in \{1, 2, 4, 7, 8, 11, 13, 14\}$) and corner points $Q_{k,i}^c$ ($i \in \{0, 3, 12, 15\}$). Face points are calculated as a convex combination of four vertices of Ω_k , as shown in Fig. 3(b). We have

$$\mathbf{Q}_k^f = \begin{bmatrix} Q_{k,5}^f \\ Q_{k,6}^f \\ Q_{k,10}^f \\ Q_{k,9}^f \end{bmatrix} = \begin{bmatrix} a & b & c & b \\ b & a & b & c \\ c & b & a & b \\ b & c & b & a \end{bmatrix} \begin{bmatrix} P_{k,0} \\ P_{k,1} \\ P_{k,2} \\ P_{k,3} \end{bmatrix} = \mathbf{M}_k^f \mathbf{P}_k, \quad (3)$$

where $a = 4/9$, $b = 2/9$, and $c = 1/9$ are coefficients obtained from the knot insertion algorithm [33] for uniform knot vectors. Face points are calculated for each element in the local mesh $\mathcal{M}_k$. We have

$$\mathbf{Q}^f = \mathbf{M}^f \mathbf{P}, \quad (4)$$

where $\mathbf{Q}^f$ is the vector of all the face points in $\mathcal{M}_k$, $\mathbf{P}$ is the vector of all the vertices in $\mathcal{M}_k$, and $\mathbf{M}^f$ is assembled from $\mathbf{M}_k^f$ (Eq. (3)). For uniform knot vectors, edge and corner points are then calculated as an average of their neighboring face points. In Fig. 3(c), the edge point Q_i^e is calculated as

$$Q_i^e = \frac{1}{2} (Q_{i,0}^f + Q_{i,1}^f), \quad (5)$$

where $Q_{i,0}^f$ and $Q_{i,1}^f$ are the two neighboring face points with respect to Q_i^e , and the subscripts $(i, 0)$ and $(i, 1)$ indicate their indices in $\mathbf{Q}^f$. Likewise in Fig. 3(d), the corner point Q_i^c is obtained by

$$Q_i^c = \frac{1}{n} \sum_{j=0}^{n-1} Q_{i,j}^f, \quad (6)$$

where we have valence $n = 4$ in the regular case.

We next study Bézier extraction for an interior irregular element. Compared to Bézier extraction of a regular element, the only difference occurs in the calculation of a corner point. In this case we still use Eq. (6), but $n \neq 4$; see Fig. 4(a). Obviously, Bézier extraction for a regular element is a special case of that for an irregular element. Regarding Bézier extraction for a boundary element, we compute edge/corner Bézier points corresponding to a boundary edge/vertex by treating it as the 1D Bézier extraction for a cubic B-spline curve. As shown in Fig. 4(b), two edge points (Q_i^e and Q_j^e) are calculated as convex combinations of two endpoints (P_0 and P_1) of the edge, and we have

$$Q_i^e = \frac{1}{3} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \end{bmatrix}. \quad (7)$$

There are two ways to compute a boundary corner point, depending on desired continuity for the boundary curve across the corner point. When C^2 continuity is desired, the corner point is calculated as an average of its two neighboring boundary edge points; see Fig. 4(c). Otherwise, when C^0 is intended (e.g. sharp features), the corner point is set as the corresponding vertex.

From Eqs. (4), (5) and (6), we can observe that all the 16 Bézier points of Ω_k , denoted by $\mathbf{Q}^0$, can be obtained from the face points $\mathbf{Q}^f$ in $\mathcal{M}_k$, and also from the mesh vertices $\mathbf{P}$. We have

$$\mathbf{Q}^0 = \mathbf{M}^a \mathbf{Q}^f = \mathbf{M}^a \mathbf{M}^f \mathbf{P} = \mathbf{M} \mathbf{P}, \quad (8)$$

where $\mathbf{M}^a$ is filled with the coefficients (1 for face points, 1/2 for edge points, and 1/n for corner points) in Eqs. (5) and (6) by taking averages of neighboring face points, and $\mathbf{M} = \mathbf{M}^a \mathbf{M}^f$ is called the *Bézier extraction matrix*. Note that Eqs. (4), (5) and (6) cannot be immediately extended to polynomial degrees other than the cubic case. Based on Eq. (8), we can derive three types of control meshes from an input quad mesh:

- A *vertex-based control mesh* formed by the input mesh vertices ($\mathbf{P}$), e.g., black squares in Fig. 3(a);
- A *face-point-based control mesh* formed by the face points ($\mathbf{Q}^f$) from Eq. (4) to each element, e.g., open circles in Fig. 3(b); and
- A *Bézier control mesh* formed by all the Bézier points ($\mathbf{Q}^0$) of each element, e.g., open circles in Fig. 3(a).

We next introduce three types of basis functions associated with these three types of control meshes: *vertex-associated functions* $\mathbf{B}^v(\xi, \eta)$, *face-point-associated functions* $\mathbf{B}^f(\xi, \eta)$ and *Bézier functions* $\mathbf{B}^0(\xi, \eta)$ (note that Bézier functions are merely C^0 B-splines). Given an element (regular or irregular), $\mathbf{B}^v(\xi, \eta)$ can be represented by linear combinations of $\mathbf{B}^0(\xi, \eta)$ through the transpose of the Bézier extraction matrix $\mathbf{M}$. Without loss of generality, we assume each element is locally parameterized on a unit square $[0, 1]^2$, and we have

$$\mathbf{B}^v(\xi, \eta) = \mathbf{M}^T \mathbf{B}^0(\xi, \eta), \quad \mathbf{B}^0(\xi, \eta) = [B_0^0(\xi, \eta), B_1^0(\xi, \eta), \dots, B_{15}^0(\xi, \eta)]^T, \quad (9)$$

where

$$B_i^0(\xi, \eta) = b_{i \% 4}(\xi) b_{i/4}(\eta), \quad i = 0, 1, \dots, 15. \quad (10)$$

Here, “%” represents “modulo” and “/” stands for “integer division by”. The functions $b_j(t)$ ($j = 0, \dots, 3$) are univariate cubic Bernstein polynomials,

$$b_0(t) = (1-t)^3, \quad b_1(t) = 3(1-t)^2t, \quad b_2(t) = 3(1-t)t^2, \quad b_3(t) = t^3. \quad (11)$$

Face-point-associated functions $\mathbf{B}^f(\xi, \eta)$ can also be represented by $\mathbf{B}^0(\xi, \eta)$ via the transpose of $\mathbf{M}^a$, and we have

$$\mathbf{B}^f(\xi, \eta) = (\mathbf{M}^a)^T \mathbf{B}^0(\xi, \eta). \quad (12)$$

Moreover, $\mathbf{B}^v(\xi, \eta)$ can also be expressed by linear combinations of $\mathbf{B}^f(\xi, \eta)$ via the transpose of $\mathbf{M}^f$, and we have

$$\mathbf{B}^v(\xi, \eta) = (\mathbf{M}^f)^T \mathbf{B}^f(\xi, \eta). \quad (13)$$

Note that on a regular element, $\mathbf{B}^v(\xi, \eta)$ and $\mathbf{B}^f(\xi, \eta)$ are simply C^2 and C^1 B-splines, respectively, but they are only C^0 -continuous across irregular elements. Corresponding to Eqs. (9), (12) and (13), we have three pairs of parent–child relationships. A Bézier function B_j^0 is called a *Bézier child* of a vertex-associated function B_i^v (or a face-point-associated function B_j^f) if the i th row and j th column element of $\mathbf{M}^T$ (or $(\mathbf{M}^a)^T$) is nonzero. Likewise, a face-point-associated function B_j^f is a *face-point-associated child* of a vertex-associated function B_i^v if the i th row and j th column element of $(\mathbf{M}^f)^T$ is nonzero.

In Eqs. (9) and (12), we can observe that each vertex-associated (or face-point-associated) function is uniquely determined by a row of coefficients in $\mathbf{M}^T$ (or $(\mathbf{M}^f)^T$), which are referred to as the *ordinates* of the vertex-associated (or face-point-associated) function. We illustrate the ordinates of several vertex-associated and face-point-associated functions defined on a regular element (shaded orange) in Figs. 5 and 6, respectively. In Fig. 5(a–c), we show the ordinates of three vertex-associated functions $\mathbf{B}^v(\xi, \eta)$ (the black square) in terms of Bézier functions $\mathbf{B}^0(\xi, \eta)$ (open circles). These $\mathbf{B}^0(\xi, \eta)$ are uniform C^2 B-splines on the element since it is regular. A vertex-associated function B_i^v can be written as a linear combination of the face-point-associated functions $\mathbf{B}^f(\xi, \eta)$ in its one-ring neighborhood. The corresponding ordinates in Fig. 5(d), where open circles represent face points. In Fig. 6(a), the ordinates of three face-point-associated functions $\mathbf{B}^f(\xi, \eta)$ (the green square) on a given regular element (shaded orange) are marked with green filled circles, whereas those marked with black open circles have no support on the orange element. Due to symmetry, we show the ordinates of three $\mathbf{B}^f(\xi, \eta)$ (green filled circles) in Fig. 6(b–d), respectively, where open circles indicate Bézier points. These $\mathbf{B}^f(\xi, \eta)$ are C^1 B-splines on the regular element.

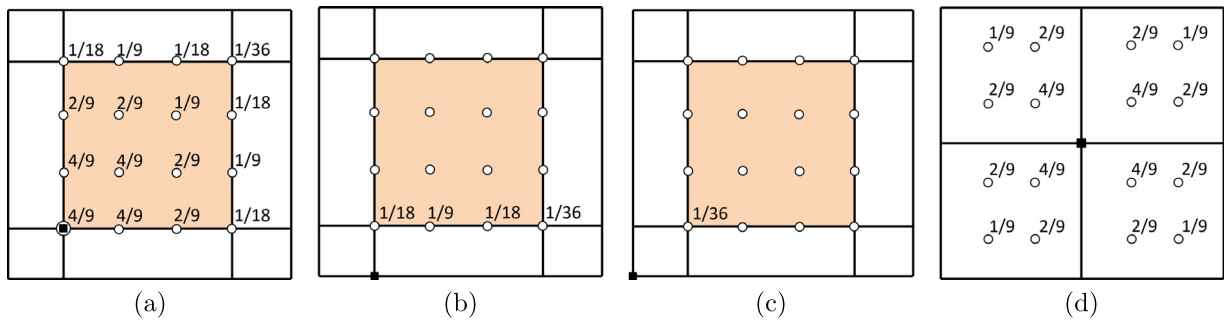


Fig. 5. On a regular element (shaded orange), vertex-associated functions $\mathbf{B}^v(\xi, \eta)$ are expressed as linear combinations of Bézier functions $\mathbf{B}^0(\xi, \eta)$ (a–c) and of face-point-associated functions $\mathbf{B}^f(\xi, \eta)$ (d). The black squares represent $\mathbf{B}^v(\xi, \eta)$ of interest. The open circles indicate Bézier points in (a–c) and represent face points in (d).

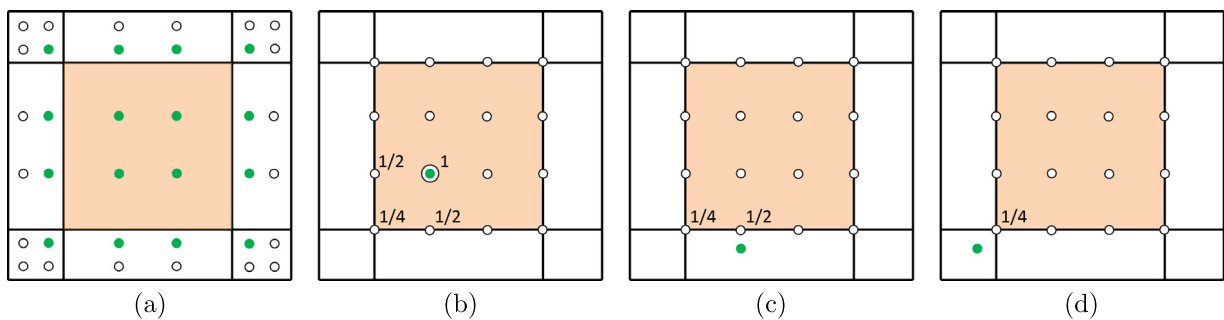


Fig. 6. On a regular element (shaded orange), face-point-associated functions $\mathbf{B}^f(\xi, \eta)$ are expressed as linear combinations of Bézier functions $\mathbf{B}^0(\xi, \eta)$. (a) Face-point-associated functions $\mathbf{B}^f(\xi, \eta)$ (green filled circles) with support on the element shaded orange, and (b–d) the ordinates of three $\mathbf{B}^f(\xi, \eta)$.

Through Bézier extraction of an element Ω_k , we have three equivalent representations for the same surface patch $S_k(\xi, \eta)$: (1) the *vertex-based representation* with the local mesh and the vertex-associated functions $\mathbf{B}^v(\xi, \eta)$, (2) the *face-point-based representation* with the face points and the face-point-associated functions $\mathbf{B}^f(\xi, \eta)$, and (3) the *Bézier representation* with the Bézier control points and the associated Bézier functions $\mathbf{B}^0(\xi, \eta)$. According to Eqs. (4), (8), (9), (12) and (13), we have the following equivalent relationships,

$$S(\xi, \eta) = \mathbf{P}^T \mathbf{B}^v(\xi, \eta) = (\mathbf{Q}^f)^T \mathbf{B}^f(\xi, \eta) = (\mathbf{Q}^0)^T \mathbf{B}^0(\xi, \eta). \quad (14)$$

Note that even though the face points stay the same in the face-point-based and Bézier representations, their associated basis functions are different. In the face-point-based representation, a face-point-associated function can be represented as a linear combination of $\mathbf{B}^0(\xi, \eta)$; see Fig. 6(b) for example, whereas in the Bézier representation, the basis function associated with a face point is merely a Bézier function.

In summary, for each element in an unstructured quad mesh, all its 16 Bézier points can be calculated from vertices of the local mesh through the Bézier extraction matrix. Three equivalent representations are available through Bézier extraction and have been utilized in the literature. The vertex-based representation was used for all the elements in [6,23,27], while the face-point-based representation was adopted in [24]. In [25,26], the face-point-based representation was adopted around extraordinary vertices whereas the vertex-based representation was used for the remaining regular region. A global Bézier representation is not favored because it introduces a large number of DOF.

Recall that the above three types of basis functions ($\mathbf{B}^v$, $\mathbf{B}^f$ and $\mathbf{B}^0$) are restricted to bicubic or higher-degree B-splines with higher continuity are defined on it; otherwise it is irregular. Note that in the higher degree case, a local (control) mesh involves more neighboring elements. Splitting the domain in regular

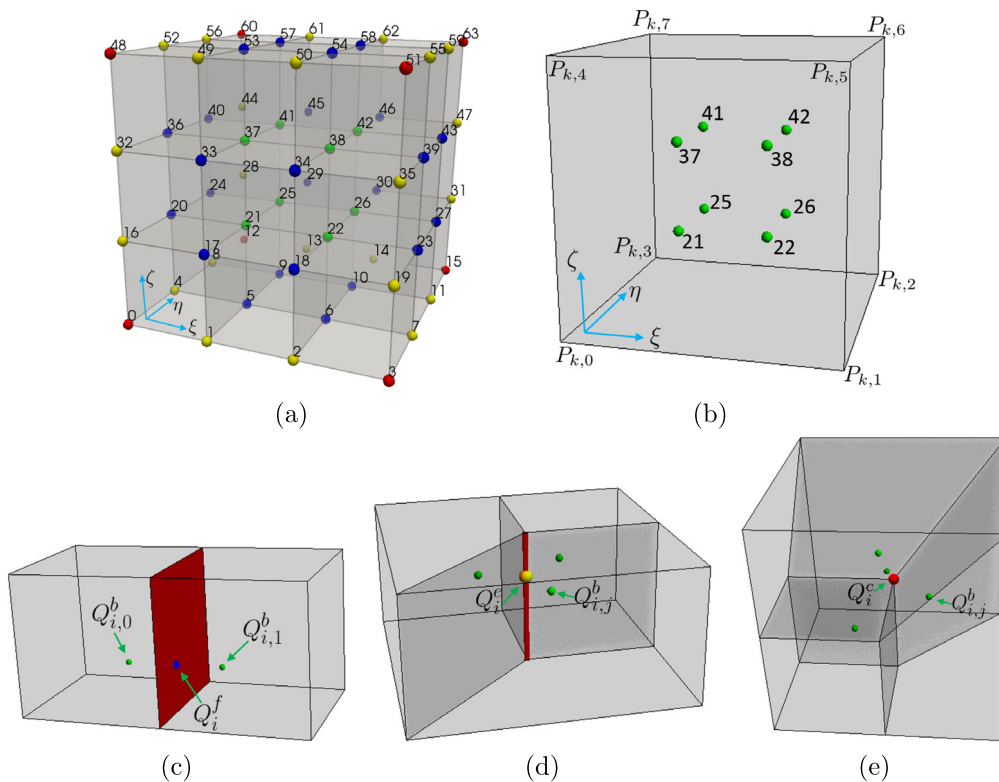


Fig. 7. Calculation of Bézier control points for an interior hex element. (a) Locally labeled 64 Bézier points, (b) computing each body point (8 green circles) as a convex combination of eight vertices of this element, (c) computing each face point (the blue circle) as the average of its neighboring body points (green circles), (d) computing each edge point (the yellow circle) as the average of its neighboring body points (green circles), and (e) computing each corner point (the red circle) as the average of its neighboring body points (green circles). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and irregular regions in this way depends on the degree because a higher degree indicates a larger local mesh and an element is regular only when this local mesh has a tensor-product structure. Even though extension in this manner is straightforward in the regular region, defining spline functions in the irregular region needs further study and even/odd degrees must be treated separately.

2.3. Bézier extraction on unstructured hex meshes

Similar to Section 2.2, we first introduce several necessary terminologies in an unstructured hex mesh. An unstructured hex mesh consists of vertices (i.e., control points), edges, faces and volumes. Its boundary is a closed surface or unstructured quad mesh. The number of elements sharing an edge/vertex is called the *valence* of the edge/vertex, respectively. An interior edge with valence other than four, or a boundary edge of valence other than two or one is called an *extraordinary edge*. Endpoints of an extraordinary edge are *extraordinary vertices*. A boundary element is an element with at least one vertex lying on the boundary, and otherwise it is an interior element. An interior element is called an *irregular element* if any of its vertices is an extraordinary vertex. Otherwise it is a regular element. If extraordinary vertices may also lie on the boundary, we treat all the boundary elements as irregular elements to facilitate implementation. Bézier extraction for an element involves its local (control) mesh, which is formed by its one-ring neighboring elements. Uniform knot vectors are also assumed here.

Bézier extraction for a hex element Ω_k involves computing 64 Bézier control points from the local mesh $\mathcal{M}_k$. The element Ω_k can be regular or irregular. These Bézier control points are locally labeled in Fig. 7(a) and can be divided into body points (Q_i^b), face points (Q_i^f), edge points (Q_i^e), and corner points (Q_i^c). Each body point is computed as a

convex combination of eight vertices ($P_{k,0}, \dots, P_{k,7}$) of Ω_k ; see Fig. 7(b). We have

$$\mathbf{Q}_k^b = \begin{bmatrix} Q_{k,21}^b \\ Q_{k,22}^b \\ Q_{k,26}^b \\ Q_{k,25}^b \\ Q_{k,37}^b \\ Q_{k,38}^b \\ Q_{k,42}^b \\ Q_{k,41}^b \end{bmatrix} = \begin{bmatrix} a & b & c & b & b & c & d & c \\ b & a & b & c & c & b & c & d \\ c & b & a & b & d & c & b & c \\ b & c & b & a & c & d & c & b \\ b & c & d & c & a & b & c & b \\ c & b & c & d & b & a & b & c \\ d & c & b & c & c & b & a & b \\ c & d & c & b & b & c & b & a \end{bmatrix} \begin{bmatrix} P_{k,0} \\ P_{k,1} \\ P_{k,2} \\ P_{k,3} \\ P_{k,4} \\ P_{k,5} \\ P_{k,6} \\ P_{k,7} \end{bmatrix} = \mathbf{M}_k^b \mathbf{P}_k, \quad (15)$$

where

$$a = \frac{8}{27}, \quad b = \frac{4}{27}, \quad c = \frac{2}{27}, \quad d = \frac{1}{27}. \quad (16)$$

These coefficients (a, b, c and d) are obtained as a tensor-product extension of those in Eq. (3). Body points need to be computed for all elements in the local mesh $\mathcal{M}_k$ to further compute face/edge/corner points. We have

$$\mathbf{Q}^b = \mathbf{M}^b \mathbf{P}, \quad (17)$$

where $\mathbf{Q}^b$ and $\mathbf{P}$ are the vectors of all the body points and vertices in $\mathcal{M}_k$, respectively, and $\mathbf{M}^b$ is assembled by matrices in Eq. (15). Similar to 2D, we treat interior and boundary elements separately. For a Bézier control point corresponding to an interior face/edge/vertex, it is calculated as an average of neighboring body points. In Fig. 7(c–e), face points Q_i^f , edge points Q_i^e and corner points Q_i^c are computed as

$$Q_i^f = \frac{1}{2} (Q_{i,0}^b + Q_{i,1}^b), \quad Q_i^e = \frac{1}{N_e} \sum_{j=0}^{N_e-1} Q_{i,j}^b, \quad Q_i^c = \frac{1}{N_v} \sum_{j=0}^{N_v-1} Q_{i,j}^b, \quad (18)$$

where the subscripts (i, j) indicate indices of neighboring body points in $\mathbf{Q}^b$, N_e and N_v are the valence of the edge and the vertex, respectively. Note that $N_e = 4$ and $N_v = 8$ hold for a regular element. Summarizing the relationships in Eq. (18) in matrix form, we have

$$\mathbf{Q}^0 = \mathbf{M}^a \mathbf{Q}^b, \quad (19)$$

where $\mathbf{Q}^0$ is the vector of all the 64 Bézier control points in Ω_k , and $\mathbf{M}^a$ is filled with coefficients in Eq. (18) computed by taking an average of neighboring body points, that is, 1 for body points, 1/2 for face points, 1/ N_e for edge points and 1/ N_v for corner points. On the other hand, for a Bézier control point corresponding to a boundary face/edge/vertex, it is calculated in the same manner as the Bézier extraction on an unstructured quad mesh; see Section 2.2. To this end, all the 64 Bézier points $\mathbf{Q}^0$ can be obtained from the local mesh through the Bézier extraction matrix $\mathbf{M}$, and we have

$$\mathbf{Q}^0 = \mathbf{M} \mathbf{P} \quad \text{and} \quad \mathbf{M} = \mathbf{M}^a \mathbf{M}^b. \quad (20)$$

Similar to 2D, we also have three types of control meshes, which are formed by input mesh vertices, body points and Bézier points. Correspondingly, we have vertex-associated functions $\mathbf{B}^v(\xi, \eta, \zeta)$, body-point-associated functions $\mathbf{B}^b(\xi, \eta, \zeta)$ and Bézier functions $\mathbf{B}^0(\xi, \eta, \zeta)$ that can be defined on a given element (regular or irregular). $\mathbf{B}^v(\xi, \eta, \zeta)$ can be represented by $\mathbf{B}^b(\xi, \eta, \zeta)$ and $\mathbf{B}^0(\xi, \eta, \zeta)$, and we have

$$\mathbf{B}^v(\xi, \eta, \zeta) = (\mathbf{M}^b)^T \mathbf{B}^b(\xi, \eta, \zeta) = (\mathbf{M}^b)^T (\mathbf{M}^a)^T \mathbf{B}^0(\xi, \eta, \zeta) = \mathbf{M}^T \mathbf{B}^0(\xi, \eta, \zeta), \quad (21)$$

where each Bézier function in $\mathbf{B}^0(\xi, \eta, \zeta)$ is a tensor product of three univariate Bernstein polynomials. $\mathbf{B}^b(\xi, \eta, \zeta)$ can be represented by linear combinations of $\mathbf{B}^0(\xi, \eta, \zeta)$, and we have

$$\mathbf{B}^b(\xi, \eta, \zeta) = (\mathbf{M}^b)^T \mathbf{B}^0(\xi, \eta, \zeta). \quad (22)$$

Three equivalent representations are also available for each hex element, that is, the *vertex-based representation*, the *body-point-based representation*, and the *Bézier representation*. They have expressions similar to Eq. (14). Based on

Eqs. (21) and (22), we also have three parent–child relationships similar to 2D, but note that we have *body-point-associated children* instead of face-point-associated children for a vertex-associated function.

On a regular element, we show the ordinates of a body-point-associated function, e.g., $Q_{k,21}^b$ in Fig. 7(a). Here we write the ordinates in a list of pairs (i, c) , where $c \in \mathbb{R}$ is the ordinate corresponding to the i th ($0 \leq i \leq 63$) Bézier function. We only list nonzero ordinates, and for the body-point-associated function of $Q_{k,21}^b$, we have $\{(0, 1/8), (1, 1/4), (4, 1/4), (5, 1/2), (16, 1/4), (17, 1/2), (20, 1/2), (21, 1)\}$. For a body-point-associated function defined on an interior irregular element, the ordinate corresponding to an extraordinary edge is simply replaced by $1/N_e$, where N_e is the valence of the extraordinary edge. The same argument applies to the ordinate corresponding to an extraordinary vertex.

Although there are three representations (Eq. (14)) that we can choose from for an unstructured hex mesh, each of them has its own limitations. When either a vertex-based or body-point-based representation is employed, refinement in irregular hex elements cannot preserve the parameterization. Refinement schemes such as Catmull–Clark subdivision do not yield optimal convergence rates [6,38]. If a global Bézier representation is adopted, an enormous number of DOF needs to be introduced, leading to a huge computation burden for the subsequent refinement. In this paper, we present a blended B-spline construction to achieve optimal convergence rates with minimal extra DOF introduced.

3. Blended B-spline construction on unstructured quad meshes

In this section, we present a new blended $C^0/C^1/C^2$ B-spline construction method (called the *C012 construction*) for unstructured quad meshes. We only use the 2D construction for illustration and explanation such that later we can easily extend the idea to unstructured hex meshes in a tensor-product manner. It is worth emphasizing that several smooth parameterization methods in 2D that can achieve optimal convergence rates are already available in IGA [11,17,18,24,25].

3.1. Construction of spline functions

Recall that we have defined several terminologies for an unstructured quad mesh; see Fig. 2. The entire unstructured quad mesh can be divided into a regular submesh and an irregular submesh, consisting of regular and irregular elements, respectively. In Fig. 8(a), the regular and irregular submeshes are shaded orange and blue, respectively. Note that irregular elements also include boundary elements for open surfaces. The interface of these two submeshes is formed by a set of edges; see the green edges in Fig. 8(a). An element is called a *transition element* if it contains one or more vertices on this interface, and a transition element can be regular or irregular.

There are three steps to construct a blended B-spline representation: (1) adding face points and edge/corner Bézier points to the quad mesh, (2) defining spline functions on each element, and (3) performing truncation for irregular elements and regular transition elements.

Step 1: Adding face points and edge/corner Bézier points to unstructured quad mesh. We add four face points to each irregular element, two edge Bézier points to each C^0 edge, and one corner Bézier point to each C^0 vertex. C^0 edges include spoke edges and boundary edges, whereas C^0 vertices are extraordinary vertices and boundary vertices; see red edges and red squares in Fig. 8(a), respectively. Correspondingly in Fig. 8(b), all the added points are shown in green and red filled circles. In this way, the boundary curve is represented as a piecewise Bézier curve. In our blended B-spline construction, extraordinary vertices are allowed on the boundary and multiple extraordinary vertices are permitted in a single element. Separating extraordinary vertices is usually needed in isogeometric spline forests [14], which involves global refinement of the input mesh and thus introduces a large number of unnecessary DOE. Although such separation can be handled locally, it still needs to modify the input mesh [13]. In contrast, our blended method adds explicit functions without modifying the input mesh, which significantly simplifies the construction, especially for 3D.

Step 2: Defining spline functions on each element. Consider vertex-associated functions B_i^v , face-point-associated functions B_i^f , and Bézier functions B_k^0 . We first look at these three types of spline functions from a global point of view. B_i^v and B_j^f have support on a two-ring and one-ring neighborhood, respectively, whereas B_k^0 only has support on elements sharing its corresponding Bézier point. To ensure linear independence, we distinguish active and

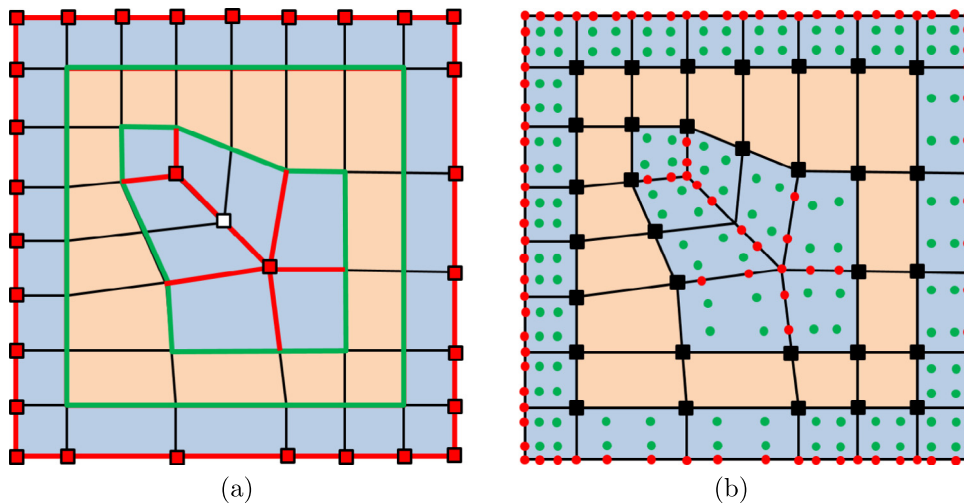


Fig. 8. Adding face points and edge/corner Bézier points to an unstructured quad mesh. (a) The input unstructured quad mesh with regular elements (orange), irregular elements (blue), the interface (green edges), C^0 edges (red edges) and C^0 vertices (red squares), and (b) adding face points (green filled circles) and edge/corner Bézier points (red filled circles) on C^0 edges/vertices, respectively. The black filled squares represent active vertices, whereas all the other mesh vertices are passive. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1

Possible spline functions defined on different types of quad elements.

Element type	Vertex-associated functions B_i^v	Face-point-associated functions B_j^f	Bézier functions B_k^0
Regular non-transition	Yes	No	No
Irregular non-transition	No	Yes	Yes
Regular transition	Yes	Yes	No
Irregular transition	Yes	Yes	Yes

passive functions and only adopt active ones in the blended construction. B_i^v is *passive* if all its one-ring neighboring elements are irregular, and otherwise it is *active*. All B_j^f and B_k^0 corresponding to face/Bézier points added in Step 1 are active. Their associated vertices or face/Bézier points are also distinguished to be active or passive. Note that each passive B_i^v can be represented as a linear combination of neighboring active B_j^f (see Fig. 3(d)) and thus it cannot be included to ensure linear independence. According to this definition, all the extraordinary vertices and boundary vertices are passive. A regular vertex may also be passive; see the black open square in Fig. 8(a) as an example. In Fig. 8(b), only the black squares are associated with active vertex-associated functions, whereas all the other mesh vertices are passive.

Next, we discuss how spline functions are defined on different types of elements. There are four types of elements: regular non-transition elements, irregular non-transition elements, regular transition elements and irregular transition elements. For any element, it suffices to search in its local mesh all the functions with support on it. A regular non-transition element, whose local mesh is formed by regular elements, only has vertex-associated functions B_i^v defined on it, and these B_i^v are simply uniform C^2 B-splines on it. The local mesh of an irregular non-transition element only contains irregular elements, so face-point-associated functions B_j^f and Bézier functions B_k^0 are defined on such elements. On the other hand, there exist both regular and irregular elements in the local mesh of a transition element. Both B_i^v and B_j^f are defined on a regular transition element, where all active B_i^v are uniform C^2 B-splines and all B_j^f are defined on its faces. All three types of functions (B_i^v , B_j^f , B_k^0) are defined on an irregular transition element; see blue elements in Fig. 8(b). We summarize in Table 1 the types of spline functions defined on these four different types of elements.

Step 3: Performing truncation on irregular elements and regular transition elements. After the above two steps, we need to guarantee that neighboring surface patches join one another seamlessly. This can be achieved by

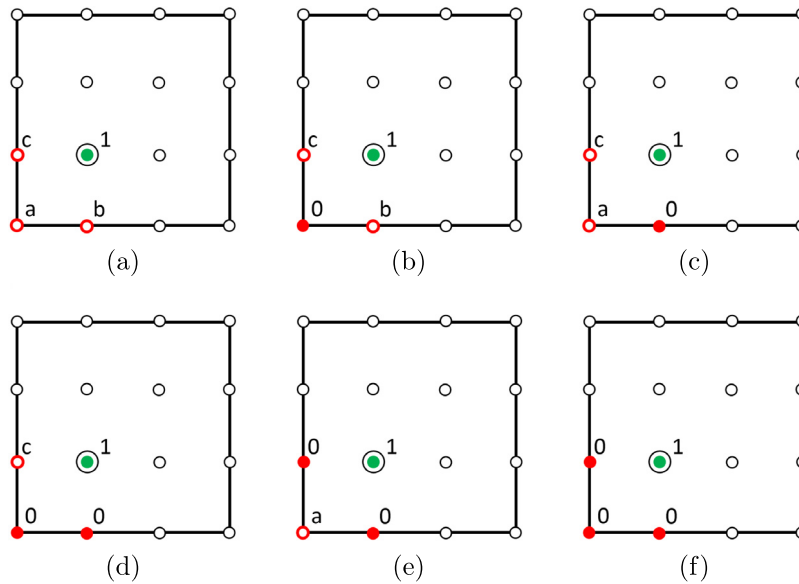


Fig. 9. Truncation of a generic face-point-associated function B_j^f (green filled circles). (a) The ordinates of B_j^f , where three of its Bézier children (red open circles) may be active, and (b–f) five possible configurations of active Bézier points (red filled circles). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

employing the truncation mechanism [32] in irregular elements and regular transition elements, where different types of functions are blended together. The truncation mechanism was originally developed in the context of hierarchical B-splines to form a partition of unity and to reduce support overlapping of basis functions, thus yielding sparser stiffness matrices [32]. Truncation also improves stability of the hierarchical basis [39] and simplifies the construction of hierarchical approximation schemes [40]. Later, this idea was extended to Catmull–Clark subdivision functions [8,9], T-splines [27], and hierarchical splines on unstructured hex meshes [6]. The essential idea of the truncation mechanism is to discard repeated contributions of active **child** basis functions. In the following, we explain how to perform the truncation mechanism in our blended B-spline construction.

Recall that according to Eqs. (12), (13) and (9), we have defined Bézier children of a face-point-associated function B_j^f , as well as face-point-associated children and Bézier children of a vertex-associated function B_i^v . Performing truncation on a B_i^v (or B_j^f) is to set the ordinates of its active children to be zero. Correspondingly, we have three types of truncation: the truncation of B_j^f with respect to its active Bézier children, and the truncation of B_i^v with respect to its face-point-associated children or Bézier children.

With reference to Fig. 9(a), we start with the truncation of a generic B_j^f (the green filled circle), where a , b and c are its ordinates and they vary in different elements. These three ordinates correspond to three Bézier points (red open circles). B_j^f needs a truncation when any of these three Bézier points is active. Note that active Bézier points correspond to C^0 edges/vertices only, and this type of truncation is always needed in irregular elements. Fig. 9(b–f) show all the possible cases of truncated B_j^f , where red filled circles are active Bézier points. Particularly in Fig. 9(f), all three ordinates are zero and B_j^f degenerates to a Bézier function. In Fig. 10(a) and (b), we also show contour plots of B_j^f corresponding to Fig. 9(c) before and after truncation, respectively. We can observe C^0 continuity across element boundaries after truncating B_j^f .

Instead of distinguishing different cases as in Fig. 9, in practice truncation is performed elementwise by manipulating related matrices. Given an element Ω_k , we first express the parent–child relationship in Eq. (12) by distinguishing active and passive² Bézier functions ($\mathbf{B}_a^0$ and $\mathbf{B}_p^0$, respectively). We have

$$\mathbf{M}^a \mathbf{B}^0 = \begin{bmatrix} (\mathbf{M}_{aa}^a)^T & (\mathbf{M}_{ap}^a)^T \\ (\mathbf{M}_{pa}^a)^T & (\mathbf{M}_{pp}^a)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^0 \\ \mathbf{B}_p^0 \end{bmatrix}, \quad (23)$$

² Passive face-point-associated functions and passive Bézier functions only serve an auxiliary purpose to explain the truncation mechanism and they are not used in a blended construction.

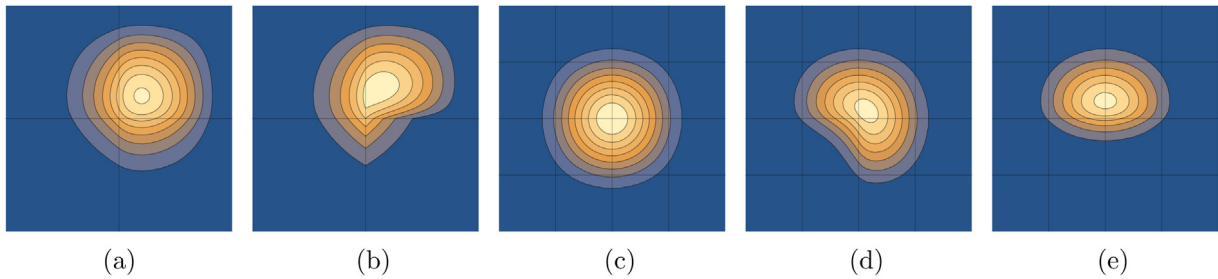


Fig. 10. (a, b) Contour plots of a face-point-associated function B_i^f corresponding to Fig. 9(c) before and after truncation, (c) the contour plot of a vertex-associated function B_i^v , and (d, e) contour plots of truncated B_i^v corresponding to Fig. 11(a, b), respectively.

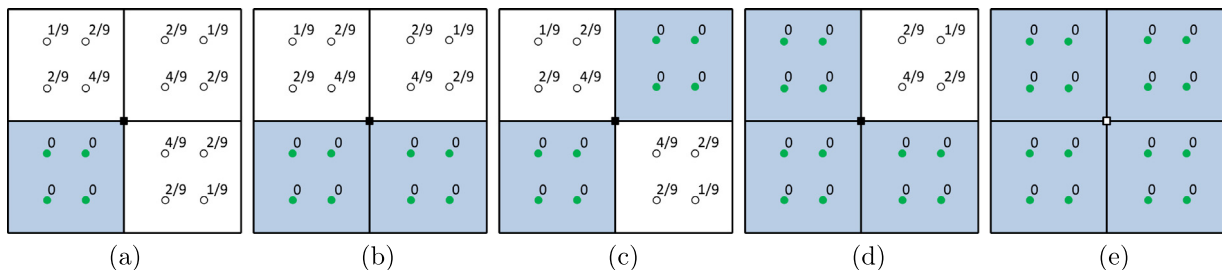


Fig. 11. Truncation of a vertex-associated function B_i^v (the black filled or open square) with different configurations of irregular elements in the one-ring neighborhood. Green filled circles and black open circles are associated with active and passive face-point-associated functions B_j^f , respectively. Black filled squares represent active B_i^v , whereas the black open square represents passive B_i^v .

where $\mathbf{B}_a^f$ and $\mathbf{B}_p^f$ are active and passive face-point-associated functions, respectively. Performing truncation is equivalent to setting the submatrices $(\mathbf{M}_{aa}^a)^T = \mathbf{0}$ and $(\mathbf{M}_{pa}^a)^T = \mathbf{0}$. Thus we obtain truncated face-point-associated functions $\mathbf{B}_t^f$ on Ω_k

$$\mathbf{B}_t^f = \begin{bmatrix} \mathbf{B}_{a,t}^f \\ \mathbf{B}_{p,t}^f \end{bmatrix} = \begin{bmatrix} \mathbf{0} & (\mathbf{M}_{ap}^a)^T \\ \mathbf{0} & (\mathbf{M}_{pp}^a)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^0 \\ \mathbf{B}_p^0 \end{bmatrix} = \begin{bmatrix} (\mathbf{M}_{ap}^a)^T \\ (\mathbf{M}_{pp}^a)^T \end{bmatrix} \mathbf{B}_p^0. \quad (24)$$

We next study the two types of truncation for an active B_i^v , which must be associated with an interior regular vertex. We first introduce the truncation of B_i^v with respect to face-point-associated children. Recall that in Fig. 5(d), B_i^v (associated with the black square) can be represented by a linear combination of its 16 face-point-associated children (open circles), which are located within its one-ring neighborhood. If any element in the one-ring neighborhood is irregular, then all its four face-point-associated children are active and B_i^v is truncated by setting their corresponding ordinates to be zero. We show all possible configurations of such a truncation in Fig. 11, where irregular elements are shaded blue, and green filled circles and black open circles represent active and passive face-point-associated children, respectively. Particularly in Fig. 11(e), all the one-ring neighboring elements are irregular and all the ordinates become zero, leading to B_i^v zero. Recall that we set such a B_i^v to be passive and do not use it in the blended construction. Also note that all the active B_i^v (black squares) need to be truncated in Fig. 8(b). Fig. 10(c) shows the contour plot of B_i^v before truncation, whereas Fig. 10(d, e) shows two examples of truncation corresponding to Fig. 11(a, b), respectively. Fig. 10(e) illustrates how the truncation mechanism works. In practice, we only need to deal with related submatrices during truncation. Rewriting Eq. (13) by distinguishing active and passive face-point-associated functions

$$\mathbf{B}^v = \begin{bmatrix} (\mathbf{M}_a^f)^T & (\mathbf{M}_p^f)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^f \\ \mathbf{B}_p^f \end{bmatrix}. \quad (25)$$

Table 2

Truncation types in different elements.

Element type	Truncating $\mathbf{B}^v$ w.r.t. face-point-associated children	Truncating $\mathbf{B}^v$ w.r.t. Bézier children	Truncating $\mathbf{B}^f$ w.r.t. Bézier children
Regular non-transition	No	No	No
Irregular non-transition	No	No	Yes
Regular transition	Yes	No	No
Irregular transition	Yes	No	Yes

Since in the end we prefer to expressing every function in terms of Bézier functions $\mathbf{B}^0$, we further replace $\mathbf{B}^f$ with $\mathbf{B}^0$ via Eq. (23). Eq. (25) becomes

$$\mathbf{B}^v = [(\mathbf{M}_a^f)^T \quad (\mathbf{M}_p^f)^T] (\mathbf{M}^a)^T \mathbf{B}^0 = [(\mathbf{M}_a^f)^T \quad (\mathbf{M}_p^f)^T] \begin{bmatrix} (\mathbf{M}_{aa}^a)^T & (\mathbf{M}_{ap}^a)^T \\ (\mathbf{M}_{pa}^a)^T & (\mathbf{M}_{pp}^a)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^0 \\ \mathbf{B}_p^0 \end{bmatrix}. \quad (26)$$

We set $(\mathbf{M}_a^f)^T = \mathbf{0}$ when truncating $\mathbf{B}^v$ with respect to face-point-associated children, and $(\mathbf{M}_{aa}^a)^T = \mathbf{0}$ and $(\mathbf{M}_{pa}^a)^T = \mathbf{0}$ when truncating $\mathbf{B}^f$ with respect to Bézier children; see Eq. (24). After truncating $\mathbf{B}^v$ in Eq. (26), we obtain

$$\mathbf{B}_t^v = [\mathbf{0} \quad (\mathbf{M}_p^f)^T] \begin{bmatrix} \mathbf{0} & (\mathbf{M}_{ap}^a)^T \\ \mathbf{0} & (\mathbf{M}_{pp}^a)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^0 \\ \mathbf{B}_p^0 \end{bmatrix} = (\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T \mathbf{B}_p^0. \quad (27)$$

The last type is the truncation of B_i^v with respect to its active Bézier children. Actually it has been included after performing the other two types of truncation; see Eq. (27), where all the ordinates corresponding to active Bézier children are already zero. Therefore, there is no need to explicitly truncate B_i^v with respect to their active Bézier children. Based on Table 1, we list the types of truncation involved in different elements in Table 2.

3.2. Spline representation and refinement

Following the above three steps, we obtain spline functions with a prescribed smoothness on the input unstructured quad mesh. Each spline function is represented as a linear combination of Bézier functions $\mathbf{B}^0$, with certain ordinates set to zero when truncation is needed. Without loss of generality, we write the vector of spline functions $\mathbf{B}$ defined on any element as

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_t^v \\ \mathbf{B}_{a,t}^f \\ \mathbf{B}_a^0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & (\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T \\ \mathbf{0} & (\mathbf{M}_{ap}^a)^T \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^0 \\ \mathbf{B}_p^0 \end{bmatrix} = \mathbf{T} \mathbf{B}^0, \quad (28)$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{T}$ is the Bézier extraction matrix in the blended construction. Note that for convenience we include all the vertex-associated functions, where a passive vertex-associated function simply has all its ordinates equal to zero. Furthermore, note that some subvectors/submatrices in Eq. (28) might be empty on certain elements. For example, the vector $\mathbf{B}_a^0$ will be empty on regular elements as there are no active Bézier functions supported on them, and the vector $\mathbf{B}_{a,t}^f$ will be empty on regular non-transition elements as there are no active face-point-associated functions supported on them. This notation allows us to generally represent the surface patch corresponding to any element by

$$S(\xi, \eta) = \mathbf{P}_{all}^T \mathbf{B} = [\mathbf{P}^T \quad (\mathbf{Q}_a^f)^T \quad (\mathbf{Q}_a^0)^T] \begin{bmatrix} \mathbf{B}_t^v \\ \mathbf{B}_{a,t}^f \\ \mathbf{B}_a^0 \end{bmatrix}, \quad (29)$$

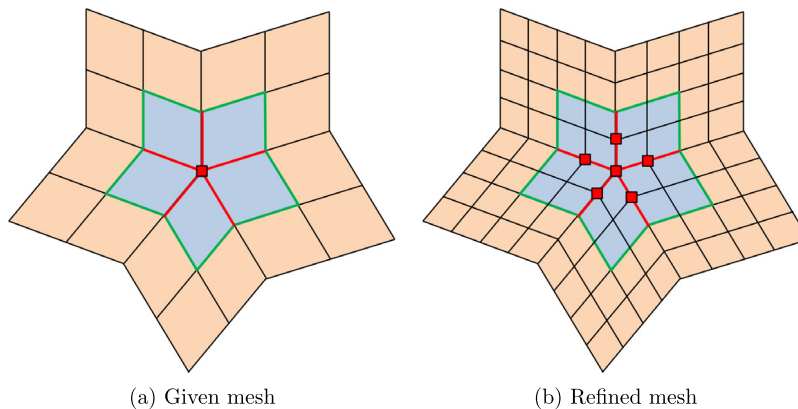


Fig. 12. Refinement of elements around an extraordinary vertex, with regular and irregular elements shaded orange and blue, respectively. Red squares and red edges represent C^0 vertices and C^0 edges, respectively. Green edges represent the interface between regular and irregular elements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

simplifies the implementation, where we only need to distinguish irregular from regular elements, together with the truncation as a separate operation. The relation between the representations in Eqs. (29) and (14) will be discussed in Section 5; see Lemma 1.

The purpose of building spline functions in such a blended manner is to keep consistent parameterization during refinement. This property will be proved in Section 5. Here we focus on how to perform refinement in a blended construction, which involves how to update vertices/Bézier points during refinement and how to define spline functions on the refined mesh. Overall, the standard knot insertion algorithm [33,34] is used for refinement of all types of elements to obtain refined vertices or Bézier points. We use the vertex-based representation and the Bézier representation to refine a regular and an irregular element, respectively. Note that applying the knot insertion algorithm to a Bézier patch is equivalent to performing the de Casteljau's algorithm [41]. By refinement, a regular surface patch results in four uniform B-spline subpatches, whereas a Bézier patch is subdivided into four Bézier subpatches. Next, to define spline functions on the refined mesh, we only need to discuss how irregular elements and C^0 edges/vertices are updated. Spline functions can be defined in the same way as in Section 3.1 by following the three-step construction. The key idea of passing irregular elements and C^0 information to refined meshes is to maintain the continuity across initial spoke edges during refinement. We refine elements by splitting them into 2×2 sub-elements.³ All the four sub-elements of an irregular element are marked as irregular. Recall that in the input mesh, spoke edges, boundary edges, extraordinary vertices and boundary vertices are identified with C^0 tags. After refinement, the sub-edges of a C^0 edge remain C^0 , whereas newly added edges are not C^0 edges. The midpoint of a C^0 edge is also marked as a C^0 vertex. All the C^0 vertices stay with C^0 tags during refinement. By recursively updating irregular elements, C^0 edges and C^0 vertices, we always maintain C^0 continuity during refinement across the spoke edges in the input mesh. This is critical in preserving consistent parameterization as well as achieving optimal convergence rates. An example of updating C^0 information is given in Fig. 12, where Fig. 12(a) and (b) represent a mesh before and after refinement, respectively. C^0 edges and C^0 vertices are marked as red edges and red squares, respectively. Regular and irregular elements are shaded orange and blue, respectively, and their interface is represented by green edges.

Discussion 3.1. Unstructured quad/hex meshes generally involve boundary extraordinary vertices, where several additional layers of elements need to be created before open knot vectors can be used. In our blended construction, instead of adopting open knot vectors, we treat boundary elements as irregular elements. In this manner, we do not need to modify the input mesh and thus can simplify the implementation.

Discussion 3.2. Truncation plays the key role in a blended B-spline construction. It reduces the support of certain boundary and vertex-associated functions, and ends up with a set of truncated functions that form a non-negative partition of unity (see Proposition 1 in Section 5). Truncated vertex-associated functions (or truncated

³ Each vertex in the coarse mesh retains its valency along the refinement, and all the newly inserted interior vertices have valency four.

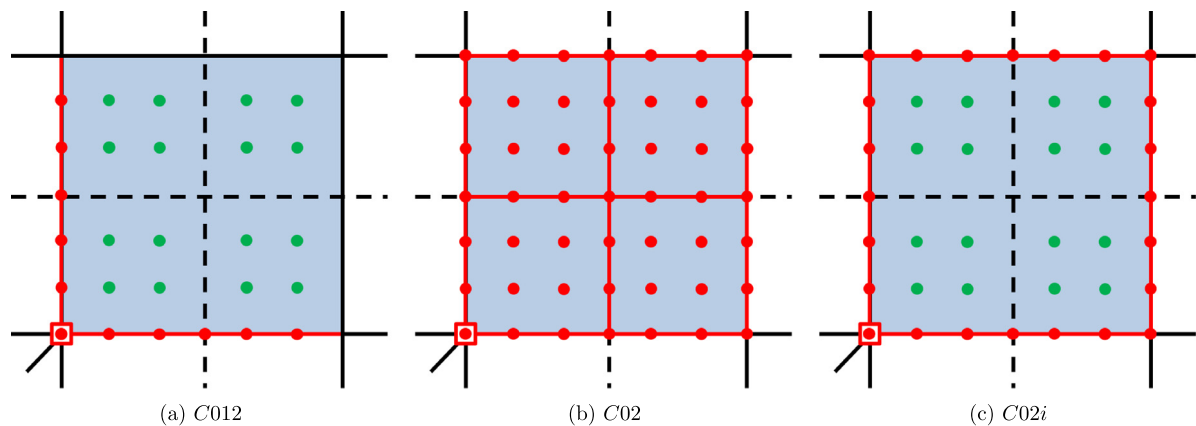


Fig. 13. Adding face points (green filled circles) and Bézier points (red filled circles) to irregular elements (shaded blue) after refinement under different constructions. The solid black lines represent the mesh before refinement, and dashed black lines indicate refinement. Red edges are C^0 edges after refinement. The red open square is an extraordinary vertex. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Additional properties in different blended constructions.

Construction type	Linear independence	C^1 smooth transition	Number of DOF
$C012$	Global	Yes	+
$C02$	Local	No	+++
$C02i$	Local	No	++

Note: More “+” means more DOF.

face-point-associated functions) are equivalent to truncated C^2 B-splines (or truncated C^1 B-splines). In other words, truncation removes “irregular” child functions from vertex-associated and face-point-associated functions. Essentially, we blend C^0 B-splines and truncated C^1/C^2 B-splines together to form a basis. It enables a seamless transition from the irregular submesh to the regular submesh, and provides us the flexibility to blend different types of spline functions. “Flexibility” here means that we can design various blended constructions with different properties. One can easily switch to another blended construction by slightly modifying Step 1.

Discussion 3.3. The above discussed blended construction blends C^0 , truncated C^1 and truncated C^2 B-splines; this motivates why we refer to it as the $C012$ construction. It provides a C^1 smooth transition from the irregular submesh to the regular submesh, but with only global linear independence. Two alternative blended constructions are given in Appendix, where more Bézier functions are added to the interface of the irregular and regular submeshes to ensure local linear independence, leading to a C^0 transition across the interface. These two alternatives are called the $C02$ construction and the $C02i$ construction. The $C02$ construction only blends vertex-associated functions and Bézier functions, where all the Bézier points are added in irregular elements in the input mesh as well as subsequent refined meshes. As an improvement of the $C02$ construction, the $C02i$ construction introduces face-point-associated functions within each irregular element in refined meshes and thus yields fewer DOF. For easy comparison, Fig. 13 shows how face points and edge/corner Bézier points are added to the same irregular elements after refinement under these three constructions. Details can be found in Appendix. In all three constructions ($C012$, $C02$ and $C02i$), spline functions form a partition of unity, are globally linearly independent, and preserve consistent parameterization during refinement. These properties will be proved under the $C012$ construction in Section 5. The proofs under the other two constructions are very similar. We list several different properties in Table 3 for easy reference.

The Trial Version 3.4. Refinement schemes for unstructured quad meshes with C^1 continuity are available in [24,25,30]. However, to the best of our knowledge, extending such refinement schemes to unstructured hex meshes is far from a trivial task. Focusing only on quad meshes, C^1 and C^2 functions are blended in [25] by properly scaling certain C^1 functions. In [26], an intermediate C^0 space is constructed for unstructured quad meshes using face-point-associated

Table 4

Dimensions of different basis functions in 2D.

	$\mathcal{B}^v$	$\mathcal{B}^f$	$\mathcal{B}^0$	$\mathcal{B}$
Dimension	N_{arv}	$4N_{ie}$	$2N_{ed0} + N_{v0}$	$N_{arv} + 4N_{ie} + 2N_{ed0} + N_{v0}$

functions with necessary Bézier functions; this space is later modified into a C^1 space. In this paper, we focus on a blended C^0 construction that can be extended to 3D. We use not only face-point-associated and Bézier functions, but also vertex-associated functions to reduce the number of DOF, which is beneficial especially in 3D. Moreover, we introduce the truncation mechanism to conveniently blend these three types of functions, such that extension to unstructured hex meshes can be straightforward.

Discussion 3.5. From the global point of view, the dimensions of different basis functions depend on the number of irregular elements (N_{ie}), the number of edges with C^0 tags (N_{ed0}), the number of vertices with C^0 tags (N_{v0}) and the number of active regular vertices (N_{arv}). Note that in different constructions (i.e., $C012$, $C02$ and $C02i$), edges and vertices with C^0 tags are defined differently. Let $\mathcal{B}$ denote the basis of a blended B-spline construction. We have $\mathcal{B} = \mathcal{B}^v \cup \mathcal{B}^f \cup \mathcal{B}^0$, where $\mathcal{B}^v$, $\mathcal{B}^f$ and $\mathcal{B}^0$ represent active vertex-associated functions, active face-point-associated functions and active Bézier functions, respectively. According to the previous discussion on the blended B-spline construction, we summarize the dimensions of different basis functions in Table 4.

4. Blended B-spline construction on unstructured hex meshes

Following the construction idea of unstructured quad meshes, in this section we extend our discussion to unstructured hex meshes. In addition to the notations we have introduced in Section 2.3, we here introduce several more as an extension from 2D. An unstructured hex mesh is decomposed into a *regular submesh* and an *irregular submesh*, which contain regular and irregular elements, respectively. The interface of these two submeshes consists of a set of faces, edges and vertices. An *interface face* must be shared by two elements, one being regular and the other being irregular. Elements containing one or more vertices on this interface are *transition elements*, which can be either regular or irregular. Similar to 2D, there are three steps in our blended B-spline construction for unstructured hex meshes: (1) adding body points and face/edge/corner Bézier points to the hex mesh, (2) identifying spline functions with support on each element, and (3) performing truncation for irregular elements and regular transition elements. The main difference from 2D lies in Step 1.

In Step 1, C^0 tags are assigned to extraordinary vertices, extraordinary edges, spoke faces and boundary faces/edges/vertices in the input mesh. A *spoke face* is a face touching an extraordinary edge. Body points are added to each irregular element, and Bézier points are added to faces, edges and vertices with C^0 tags. An irregular element is shown in Fig. 14(a) with one of its edges being an extraordinary edge (marked in red). According to the definition, its two endpoints are extraordinary vertices (black circles) and two faces (shaded dark gray) sharing the extraordinary edge are spoke faces. In Fig. 14(b), the added face points and face/edge/corner Bézier points are marked with green and red circles, respectively. We show two examples in Fig. 14(c, d) explaining how Bézier points are added to a spoke face, where red edges and red open squares represent extraordinary edges and extraordinary vertices, respectively. A spoke face may have other configurations of extraordinary edges, where Bézier points can be added analogously.

The next two steps are very similar to those in 2D. In Step 2, given an element Ω_k , we loop through all the elements in its local mesh and collect active vertex-associated functions B_i^v , body-point-associated functions B_j^b as well as Bézier functions B_k^0 . A B_i^v is passive if all of its one-ring neighboring elements are irregular. Regarding types of spline functions defined on different elements, we can obtain similar results as Table 1 in 2D, where we only need to change “face-point-associated functions B_j^f ” to “body-point-associated functions B_j^b ”. In Fig. 15, we use three configurations to show neighboring B_j^b with support on an element of interest (orange elements). A neighboring element may share a face (Fig. 15(a)), an edge (Fig. 15(b)), or a vertex (Fig. 15(c)) with the orange element. If the neighboring element is irregular, eight B_j^b (green circles and black circles) are added. However, only four B_j^b (green circles) have support on the orange element.

In Step 3, following the same manner as in 2D, we perform three types of truncation in irregular elements and regular transition elements: body-point-associated functions B_j^b with respect to Bézier children, and vertex-associated functions B_i^v with respect to body-point-associated or Bézier children. By truncation, we set ordinates corresponding

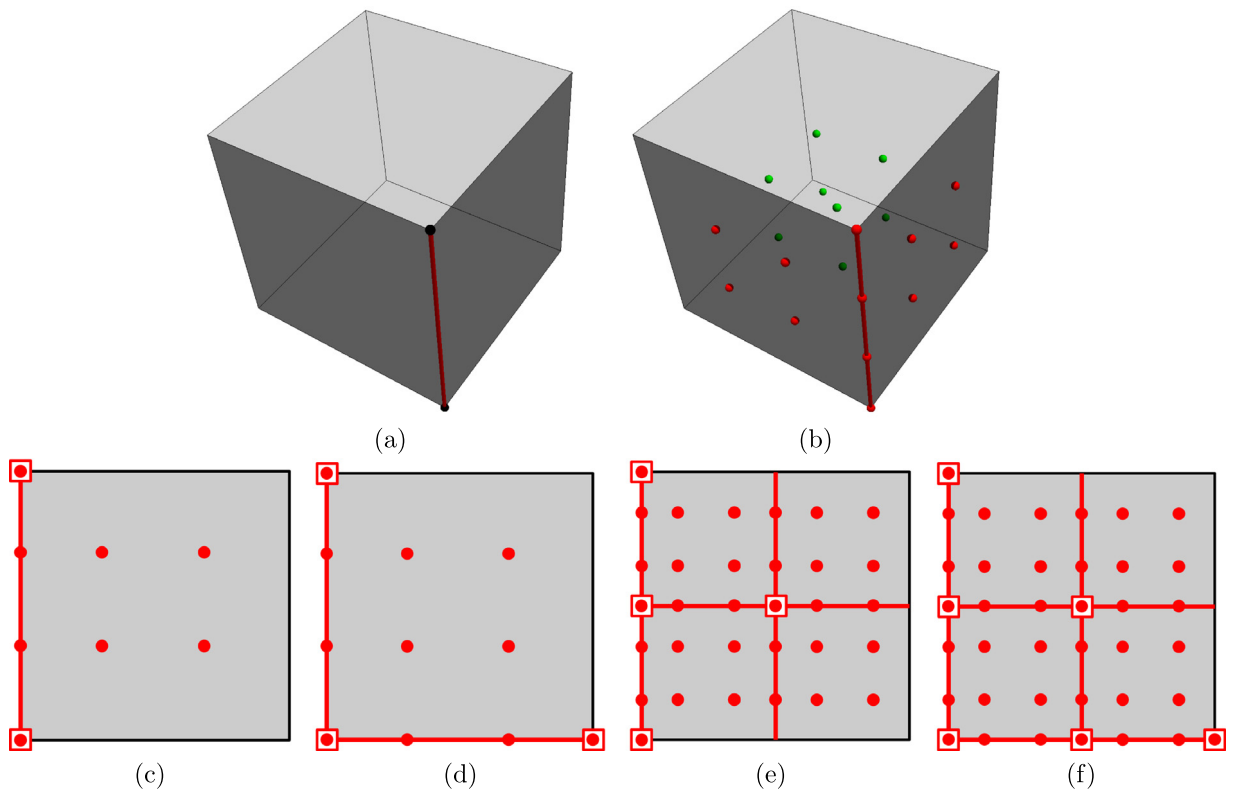


Fig. 14. (a) An irregular element with an extraordinary edge (the red edge) and two extraordinary vertices (black circles), (b) adding body points (green circles) and face/edge/corner Bézier points (red circles) to the element in (a), (c, d) two examples of adding Bézier points (red circles) to spoke faces, and (e, f) adding Bézier points to C^0 faces after refining the spoke faces in (c, d), respectively. Spoke faces and C^0 faces are shaded dark gray in (a, b). In (c–f), C^0 edges and C^0 vertices are represented by red edges and red open squares, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

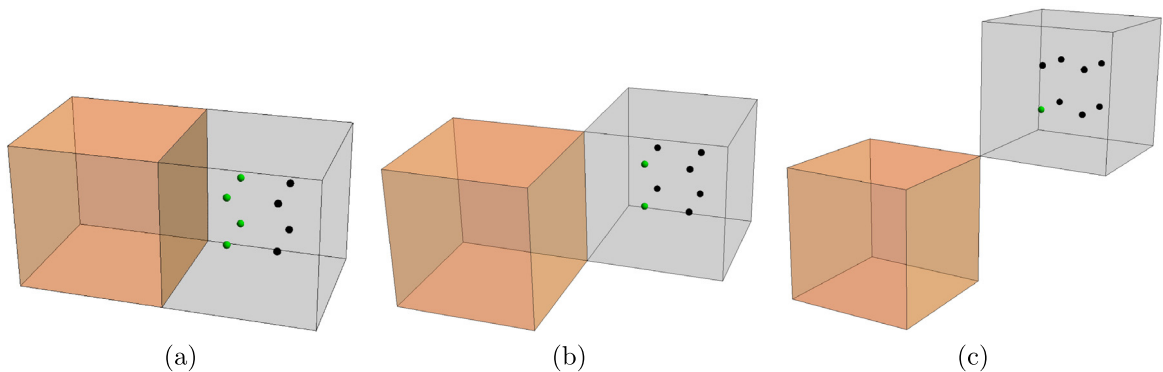


Fig. 15. Three different configurations of neighboring body-point-associated functions B_j^b with support on an element of interest (shaded orange). Neighboring irregular elements are shaded gray. Only those B_j^b associated with green circles have support on the orange element. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

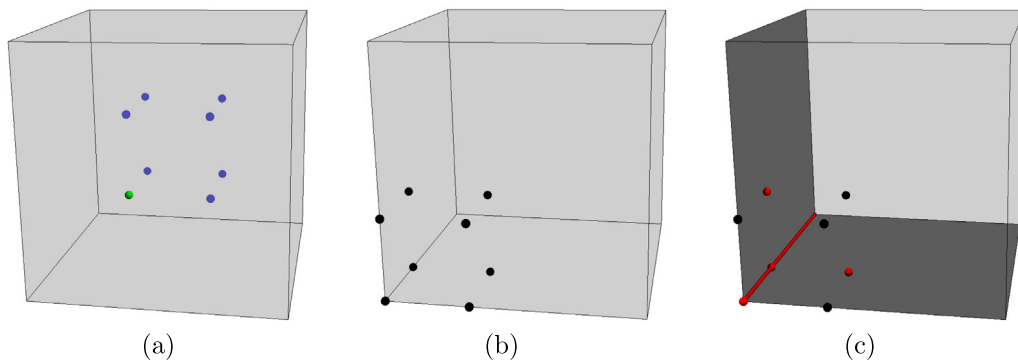


Fig. 16. Truncation of a body-point-associated function B_J^b with respect to active Bézier children. (a) A body-point-associated function B_J^b of interest (the green circle), (b) the eight Bézier children (black circles) of B_J^b , and (c) active Bézier children (red circles) due to presence of an extraordinary edge (the red edge) and two spoke faces (dark gray faces). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Truncating B_i^v with respect to body-point-associated children is a direct extension from 2D, where we only need to check if there exists any irregular element in its one-ring neighborhood. Truncating B_i^v with respect to Bézier children is straightforward and has been included when performing the other two types of truncation. Eq. (28) also works for truncation in 3D, where we replace “ $\mathbf{B}_{a,t}^f$ ” with “ $\mathbf{B}_{a,t}^b$ ” and replace “ $\mathbf{M}_p^f$ ” with “ $\mathbf{M}_p^b$ ”.

The above three-step procedure gives us a set of spline functions on the input unstructured hex mesh. Next, we discuss how to perform refinement. It is a tensor-product extension from 2D, where a regular volume patch is refined in the same way as a tricubic uniform B-spline and an irregular patch employs the de Casteljau’s algorithm for a tricubic Bézier patch. Similar to 2D, tracking C^0 tags is again the key to maintain C^0 continuity across the initial spoke faces during refinement. A hex element is subdivided into 8 sub-elements with one center vertex, 6 edges and 12 faces generated in the interior. All the sub-elements of an irregular element are tagged “irregular”, although some of them do not have any extraordinary edge. Sub-faces of a C^0 face and sub-edges of a C^0 edge inherit C^0 tags. The newly generated center vertex and four edges of a C^0 face as well as the middle vertex of a C^0 edge are also tagged C^0 . The given C^0 vertices stay the same. In Fig. 14(e, f), we show two cases how C^0 tags are assigned after refining a spoke face in Fig. 14(c, d), respectively. C^0 faces/edges/vertices are denoted by gray shade, red edges and red open squares, respectively.

Discussion 4.1. The truncation mechanism provides a convenient way to seamlessly connect regular and irregular patches, where we only need to manipulate the Bézier extraction matrix $\mathbf{T}$ in Eq. (28). Although T-spline local knot vectors can be used as well in 2D to connect regular and irregular patches [13,15], the underlying mesh needs to be modified by inserting edges around extraordinary vertices. Extending this idea to 3D becomes very difficult due to the complex connectivity of an unstructured hex mesh.

Discussion 4.2. The vertex-based representation is employed for every hex element for truncated hierarchical splines in [6], with the solid Catmull–Clark subdivision rule [4] used for refinement. A suboptimal convergence behavior was observed. Special treatment is also needed to construct hierarchical splines due to the lack of refinability. Refinability can be enabled in our blended construction but requires enlargement of the irregular region as well as assigning C^0 tags to additional faces/edges/vertices at each refinement step. We postpone the detailed discussion as part of our future work in the context of hierarchical refinement.

Discussion 4.3. The dimensions of different basis functions can be obtained in a similar manner as in 2D; see the 3D result in Table 5, with N_{f0} representing the number of faces with C^0 tags.

The Trial Version of pdfElement is straightforward to use C^0 Bézier functions throughout an unstructured hex mesh, but it would introduce too many DOF especially in 3D. Later in Section 6 we can observe that the C02 construction already adds many more DOF than the C012 construction (Table 8), not to mention the fact that Bézier functions are only added to irregular elements in the C02 construction. The goal of a blended construction is to introduce minimal DOF while

Table 5

Dimensions of different basis functions in 3D.

	$\mathcal{B}^v$	$\mathcal{B}^b$	$\mathcal{B}^0$	$\mathcal{B}$
Dimension	N_{arv}	$8N_{ie}$	$4N_{f0} + 2N_{ed0} + N_{v0}$	$N_{arv} + 8N_{ie} + 4N_{f0} + 2N_{ed0} + N_{v0}$

maintaining optimal convergence rates, and it is particularly beneficial when a given hex mesh is dominant with regular elements. Moreover, in the future we will incorporate local refinement to the blended construction, where it is desired to start with a mesh with minimal DOF, leaving the remaining to be added based on simulation results.

5. Properties of blended B-spline construction

In this section, we prove several important properties under the C012 blended B-spline construction, including non-negative partition of unity, (global) linear independence, and consistent parameterization. These properties also hold for the other two constructions (the C02 construction and the C02i construction) and they can be proved in a similar manner. Local linear independence is a property only available for the C02 and C02i constructions, which will be discussed in [Appendix](#). Here we start with the proof of non-negative partition of unity.

Proposition 1. *The (active) spline functions in the C012 blended B-spline construction form a non-negative partition of unity.*

Proof. We prove this proposition elementwise. Recall that Eq. (28) is a unified expression of spline functions defined on all types of elements and it is also generic in both 2D and 3D. In the 2D case we have $\mathbf{B} = \mathbf{T}\mathbf{B}^0$ where

$$\mathbf{T} = \begin{bmatrix} \mathbf{0} & (\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T \\ \mathbf{0} & (\mathbf{M}_{ap}^a)^T \\ \mathbf{I} & \mathbf{0} \end{bmatrix}. \quad (30)$$

Each spline function in $\mathbf{B}$ is non-negative because all entries in the Bézier extraction matrix $\mathbf{T}$ are non-negative. The next step is to prove that the spline functions in $\mathbf{B}$ form a partition of unity. With Bézier functions $\mathbf{B}^0$ forming a partition of unity, we only need to verify that each column sum of $\mathbf{T}$ equals 1. Any column sum of the identity matrix is one, so we only need to verify each column sum of the remaining two submatrices, $(\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T$ and $(\mathbf{M}_{ap}^a)^T$. Note that in 3D, $\mathbf{M}_p^b$ is used instead of $\mathbf{M}_p^f$. We define

$$\mathbf{M}_1 := \begin{bmatrix} \mathbf{0} & (\mathbf{M}_p^f)^T \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \quad (31)$$

and

$$\mathbf{M}_2 := \mathbf{M}_1 (\mathbf{M}^a)^T = \begin{bmatrix} \mathbf{0} & (\mathbf{M}_p^f)^T \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} (\mathbf{M}_{aa}^a)^T & (\mathbf{M}_{ap}^a)^T \\ (\mathbf{M}_{pa}^a)^T & (\mathbf{M}_{pp}^a)^T \end{bmatrix} = \begin{bmatrix} (\mathbf{M}_p^f)^T (\mathbf{M}_{pa}^a)^T & (\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T \\ (\mathbf{M}_{aa}^a)^T & (\mathbf{M}_{ap}^a)^T \end{bmatrix}. \quad (32)$$

Each column sum of $(\mathbf{M}_p^f)^T$ equals 1 because its transpose $\mathbf{M}_p^f$ is used to compute passive face points as convex combinations⁴ of element corners. Therefore, each column sum of $\mathbf{M}_1$ is 1. Similarly, each column sum of $(\mathbf{M}^a)^T$ equals 1 since $\mathbf{M}^a$ is used to compute face/edge/corner Bézier points as convex combinations of neighboring face (or body) points. We can then easily obtain that each column sum of their multiplication $\mathbf{M}_2$ also equals 1. Comparing $\mathbf{T}$ and $\mathbf{M}_2$ in Eqs. (30) and (32), especially their second columns of submatrices, we can conclude that each column sum of $\mathbf{T}$ is also 1. Therefore, spline functions in the C012 blended construction form a non-negative partition of unity. $\square$

Linear independence of spline functions in the C012 blended B-spline construction. In particular, they are (globally) linearly independent on the entire domain, but may be locally linearly dependent on certain elements individually. In [Appendix](#), we will discuss local linear independence of the alternative C02 and C02i constructions.

⁴ A convex combination is defined to be a linear combination where all the coefficients are non-negative and their sum equals 1.

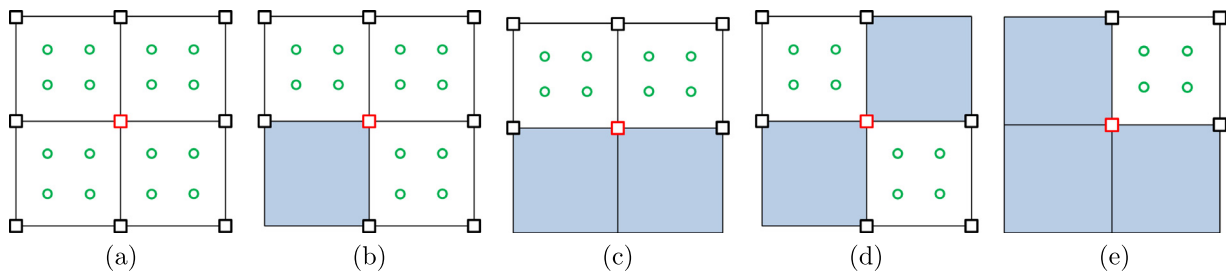


Fig. 17. Five possible cases in 2D where linear dependence may occur. $B_{t,i}^v$, $B_{t,j}^v$ and their face-point-associated children in Eq. (33) are associated with the red open square, black open squares and green open circles, respectively. The irregular region is shaded blue. The dimension of the underlying matrix $\mathbf{M}$ in (a–e) is 9×16 , 8×12 , 6×8 , 7×8 and 4×4 , respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Proposition 2. *The (active) spline functions in the C012 blended B-spline construction are linearly independent on the entire domain.*

Proof. We use 2D terminologies for explanation, but the arguments are the same in 3D. We first restrict ourselves to the regular subdomain determined by the regular submesh, where only vertex-associated functions and face-point-associated functions have support. These functions are actually uniform C^2 B-splines and C^1 B-splines, respectively, with certain C^2 B-splines truncated with respect to C^1 B-splines. Linear dependence occurs only when a (truncated) vertex-associated function $B_{t,i}^v$ can be represented by a linear combination of other (truncated) vertex-associated functions $B_{t,j}^v$ and active face-point-associated functions $B_{a,k}^f$. Under the assumption of linear dependence, we have

$$B_{t,i}^v = \sum_{j \neq i} c_{ij} B_{t,j}^v + \sum_k d_{ik} B_{a,k}^f, \quad (33)$$

where c_{ij} and d_{ik} are the corresponding coefficients.

Recall that a vertex-associated function can be expressed in terms of its face-point-associated children (here only C^1 B-splines), where we only need to study its one-ring neighborhood. Because of the local linear independence of classical B-splines, a necessary condition for Eq. (33) to hold is that, on the regular subdomain, the face-point-associated children of $B_{t,i}^v$ coincide with $B_{a,k}^f$ and the face-point-associated children of $B_{t,j}^v$. We actually do not need to include $B_{a,k}^f$ since all d_{ik} must be zero according to the truncation mechanism. Next, we only consider $B_{t,i}^v$ and $B_{t,j}^v$ and we represent them using their face-point-associated children in matrix form $\mathbf{B}_t^v = \mathbf{M}\mathbf{B}^f$, where all the face-point-associated children $\mathbf{B}^f$ are C^1 B-splines and they are linearly independent, and $\mathbf{M}$ can be obtained by looping through regular elements and assembling matrices in Eqs. (3) and (15). We then only need to check the rank of $\mathbf{M}$. Depending on the number and positions of irregular elements (shaded blue) in the one-ring neighborhood of $B_{t,i}^v$, we have five cases in 2D (Fig. 17) and twenty cases in 3D (Fig. 18) where linear dependence may occur. In Fig. 17, $B_{t,i}^v$, $B_{t,j}^v$ and their face-point-associated children are marked with red open squares, black open squares and green open circles, respectively. In Fig. 18, $B_{t,i}^v$ is marked with red circles and $B_{t,j}^v$ are associated with all the corners of white elements. We have built the matrix in each case and verified that $\mathbf{M}$ has full rank for all the cases in both 2D and 3D using Mathematica. Therefore, $B_{t,i}^v$ and $B_{t,j}^v$ are linearly independent, which contradicts the assumption in Eq. (33). This concludes that (truncated) vertex-associated functions and face-point-associated functions are linearly independent on the regular subdomain.

Then we only need to verify linear independence of the spline functions with support fully contained in the irregular subdomain, which are face-point-associated functions and Bézier functions. Actually, linear independence of these functions is proved in Proposition 4.2 in [26]. This concludes the proof. $\square$

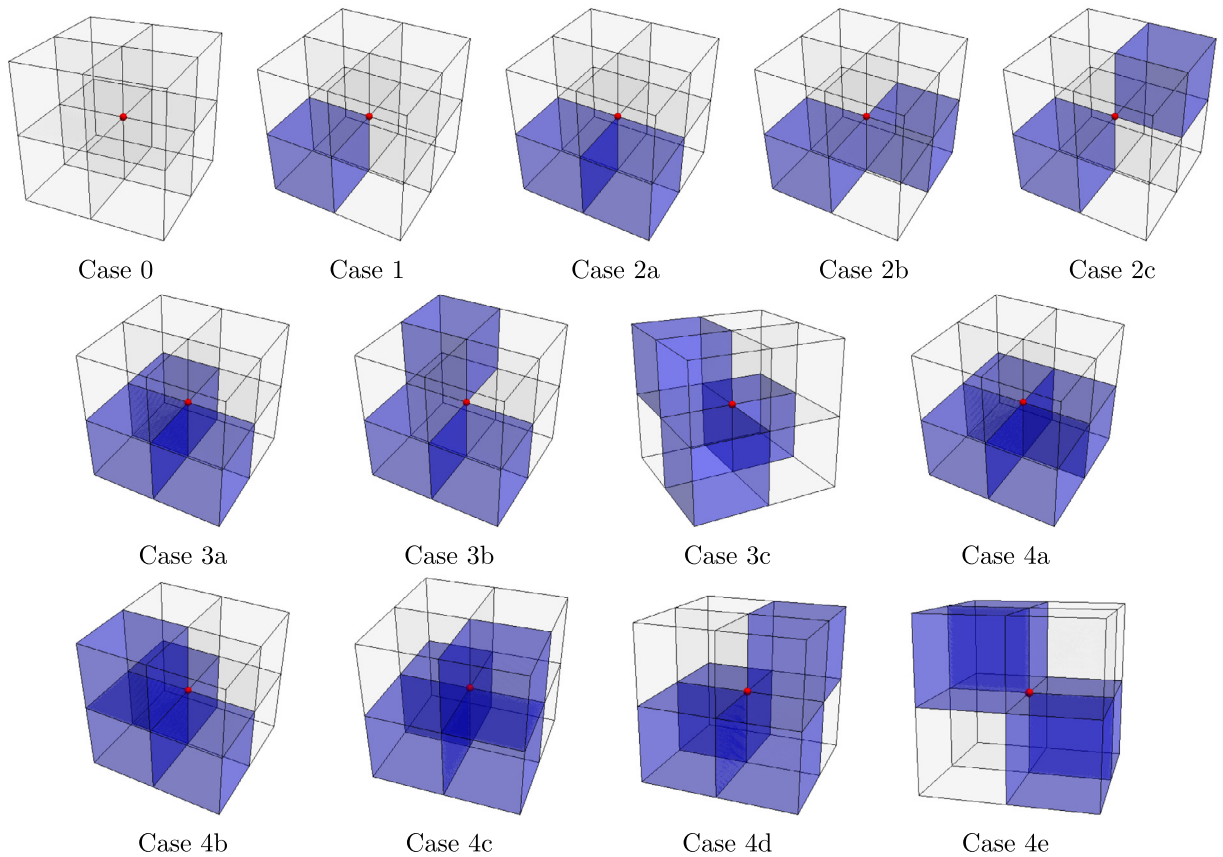


Fig. 18. Thirteen cases in 3D with possible linear dependence, where the vertex-associated function of interest is associated with the red circle and irregular elements are shaded blue. Seven cases that involve more than four irregular elements can still be shown using these figures (Cases 1, 2a, 2b, 2c, 3a, 3b and 3c), where irregular and regular elements are shaded white and blue, respectively.

Lemma 1. For a fixed element, the C012 blended B-spline representation in Eq. (29) is equivalent to the representations in Eq. (14), provided that the active vertices $\mathbf{P}_a$, face points $\mathbf{Q}_a^f$ and Bézier points $\mathbf{Q}_a^0$ from Eq. (29) are the same in the respective representations in Eq. (14).

Proof. It suffices to prove that the C012 blended B-spline representation $\mathbf{P}_{\text{all}}^T \mathbf{B}$ is equivalent to the vertex-based representation $\mathbf{P}^T \mathbf{B}^v$, as the others (face-point-based representation and Bézier representation) are the same. For a fixed element, we start from its vertex-based representation $\mathbf{P}^T \mathbf{B}^v$ and express the vertex-associated functions $\mathbf{B}^v$ in terms of active and passive face-point-associated children according to Eq. (25). We have

$$\mathbf{P}^T \mathbf{B}^v = \mathbf{P}^T \begin{bmatrix} (\mathbf{M}_a^f)^T & (\mathbf{M}_p^f)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^f \\ \mathbf{B}_p^f \end{bmatrix} = \begin{bmatrix} (\mathbf{M}_a^f \mathbf{P})^T & (\mathbf{M}_p^f \mathbf{P})^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^f \\ \mathbf{B}_p^f \end{bmatrix} = \begin{bmatrix} (\mathbf{Q}_a^f)^T & (\mathbf{Q}_p^f)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^f \\ \mathbf{B}_p^f \end{bmatrix}, \quad (34)$$

where $\mathbf{Q}_a^f = \mathbf{M}_a^f \mathbf{P}$ and $\mathbf{Q}_p^f = \mathbf{M}_p^f \mathbf{P}$ are active and passive face points, respectively. We further substitute face-point-associated functions ($\mathbf{B}_a^f$ and $\mathbf{B}_p^f$) with Bézier functions ($\mathbf{B}_a^0$ and $\mathbf{B}_p^0$) according to Eq. (23). We have

$$\begin{aligned} & \begin{bmatrix} (\mathbf{Q}_p^f)^T \end{bmatrix} \begin{bmatrix} (\mathbf{M}_{aa}^a)^T & (\mathbf{M}_{ap}^a)^T \\ (\mathbf{M}_{pa}^a)^T & (\mathbf{M}_{pp}^a)^T \end{bmatrix} \begin{bmatrix} \mathbf{B}_a^0 \\ \mathbf{B}_p^0 \end{bmatrix} \\ &= (\mathbf{M}_{aa}^a \mathbf{Q}_a^f + \mathbf{M}_{pa}^a \mathbf{Q}_p^f)^T \mathbf{B}_a^0 + (\mathbf{M}_{ap}^a \mathbf{Q}_a^f + \mathbf{M}_{pp}^a \mathbf{Q}_p^f)^T \mathbf{B}_p^0 \\ &= (\mathbf{M}_{aa}^a \mathbf{Q}_a^f + \mathbf{M}_{pa}^a \mathbf{Q}_p^f)^T \mathbf{B}_a^0 + (\mathbf{Q}_a^f)^T (\mathbf{M}_{ap}^a)^T \mathbf{B}_p^0 + \mathbf{P}^T (\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T \mathbf{B}_p^0. \end{aligned} \quad (35)$$

Table 6

Statistics of input control meshes.

Models	# Vertices	# Elements	# Extraordinary points (interior)	# Irregular elements (interior)
Square	143	120	8	13
Manifold	161	128	23	99
Cube	517	448	24	56
Rod	2011	1376	24	56
Hook	6327	5121	2572	3952

From Eq. (8), we have $\mathbf{Q}_a^0 = \mathbf{M}_{aa}^a \mathbf{Q}_a^f + \mathbf{M}_{pa}^a \mathbf{Q}_p^f$, where $\mathbf{Q}_a^0$ represent active Bézier points. From Eq. (28), we have $\mathbf{B}_t^v = (\mathbf{M}_p^f)^T (\mathbf{M}_{pp}^a)^T \mathbf{B}_p^0$ and $\mathbf{B}_{a,t}^f = (\mathbf{M}_{ap}^a)^T \mathbf{B}_p^0$. Therefore, Eq. (35) becomes

$$\mathbf{P}^T \mathbf{B}^v = (\mathbf{Q}_a^0)^T \mathbf{B}_a^0 + (\mathbf{Q}_a^f)^T \mathbf{B}_{a,t}^f + \mathbf{P}^T \mathbf{B}_t^v = [\mathbf{P}^T \quad (\mathbf{Q}_a^f)^T \quad (\mathbf{Q}_a^0)^T] \begin{bmatrix} \mathbf{B}_t^v \\ \mathbf{B}_{a,t}^f \\ \mathbf{B}_a^0 \end{bmatrix}. \quad (36)$$

Comparing Eqs. (36) and (29), we immediately deduce $\mathbf{P}^T \mathbf{B}^v = \mathbf{P}_{\text{all}}^T \mathbf{B}$. This concludes that the blended representation is equivalent to the vertex-based representation. $\square$

We are now ready to prove consistent parameterization during refinement with the aid of Lemma 1.

Proposition 3. *The parameterization corresponding to any element in the C012 blended B-spline construction stays the same during refinement.*

Proof. We use 2D terminologies for explanation, but the arguments are the same in 3D. From Lemma 1 we know how the C012 blended B-spline representation $\mathbf{P}_{\text{all}}^T \mathbf{B}$ relates to the vertex-based representation $\mathbf{P}^T \mathbf{B}^v$, face-point-based representation $(\mathbf{Q}^f)^T \mathbf{B}^v$ and Bézier representation $(\mathbf{Q}^0)^T \mathbf{B}^0$. Recall that the face points stay the same in the face-point-based and Bézier representations. In the following, we use “ $\sim$ ” to denote all objects (points as well as spline functions) related to the refined mesh. Refinement of $\mathbf{P}_{\text{all}}^T \mathbf{B}$ is computed by applying the standard knot insertion algorithm, where we refine a regular element on its vertex-based representation ($\mathbf{P}^T \mathbf{B}^v = \tilde{\mathbf{P}}^T \tilde{\mathbf{B}}^v$) and refine an irregular element on its Bézier representation ($(\mathbf{Q}^0)^T \mathbf{B}^0 = (\tilde{\mathbf{Q}}^0)^T \tilde{\mathbf{B}}^0$). This implies that the refined version $\tilde{\mathbf{P}}_{\text{all}}^T \tilde{\mathbf{B}}$ shares the active vertices $\tilde{\mathbf{P}}_a$, face points $\tilde{\mathbf{Q}}_a^f$ and Bézier points $\tilde{\mathbf{Q}}_a^0$ with the respective representations $\tilde{\mathbf{P}}^T \tilde{\mathbf{B}}^v$, $(\tilde{\mathbf{Q}}^f)^T \tilde{\mathbf{B}}^f$ and $(\tilde{\mathbf{Q}}^0)^T \tilde{\mathbf{B}}^0$. Using Lemma 1, we conclude that $\tilde{\mathbf{P}}_{\text{all}}^T \tilde{\mathbf{B}} = \mathbf{P}_{\text{all}}^T \mathbf{B}$. $\square$

6. Numerical examples

In this section, we apply the blended B-spline construction to two quad meshes in Fig. 19 and three hex meshes in Figs. 20–22 and verify that the blended construction yields optimal convergence rates. We summarize the statistics of these five input meshes in Table 6.

We first test the C012 construction by solving Poisson’s equation on planar unstructured quad meshes. Two input control meshes are studied: one discretizing a square domain and the other defining a manifold domain; see Fig. 19(a, d), respectively. We adopt the same exact solution for both meshes,

$$u(x, y) = xy(1 - x)(1 - y)(1 + y \sin(x) + x \sin(y)). \quad (37)$$

For each model, Dirichlet boundary conditions are imposed on the entire boundary. The boundary conditions can be strongly or weakly imposed. Since only Bézier functions are defined on the boundary and they are non-interpolatory, we perform a least-square fitting for a strong imposition to approximate the given Dirichlet data. Nitsche’s method is used for a weak imposition [42]. We observe that strong imposition and weak imposition yield almost the same convergence rates (in terms of L^2 and H^1 error). We create a series of meshes through global refinement, and build blended spline functions on them. In Fig. 19(b, e), we plot the convergence curves of the L^2 - and H^1 -norm errors with respect to the maximum element size (h_{max}). We observe optimal convergence rates for both models. Note that when bicubic splines are used as a basis, the expected optimal convergence rate is 4 for the L^2 -norm error and 3 for the H^1 -norm error. Plotting convergence curves in terms of the square root of DOF in Fig. 19(c, f) shows optimal rates as well.

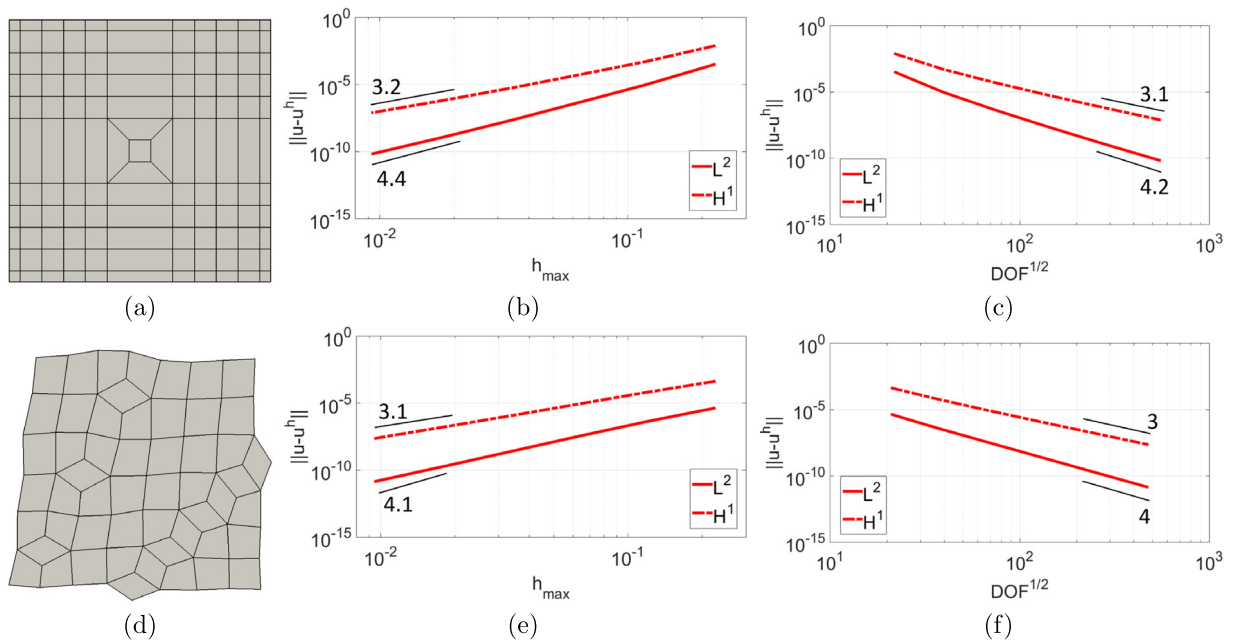


Fig. 19. Solving Poisson's equation on a square domain (a–c) and on a manifold domain (d–f). (a, d) Input control meshes, (b, e) convergence curves with respect to the maximum element size ($h_{\max}$), and (c, f) convergence curves with respect to the square root of DOF ($\text{DOF}^{1/2}$).

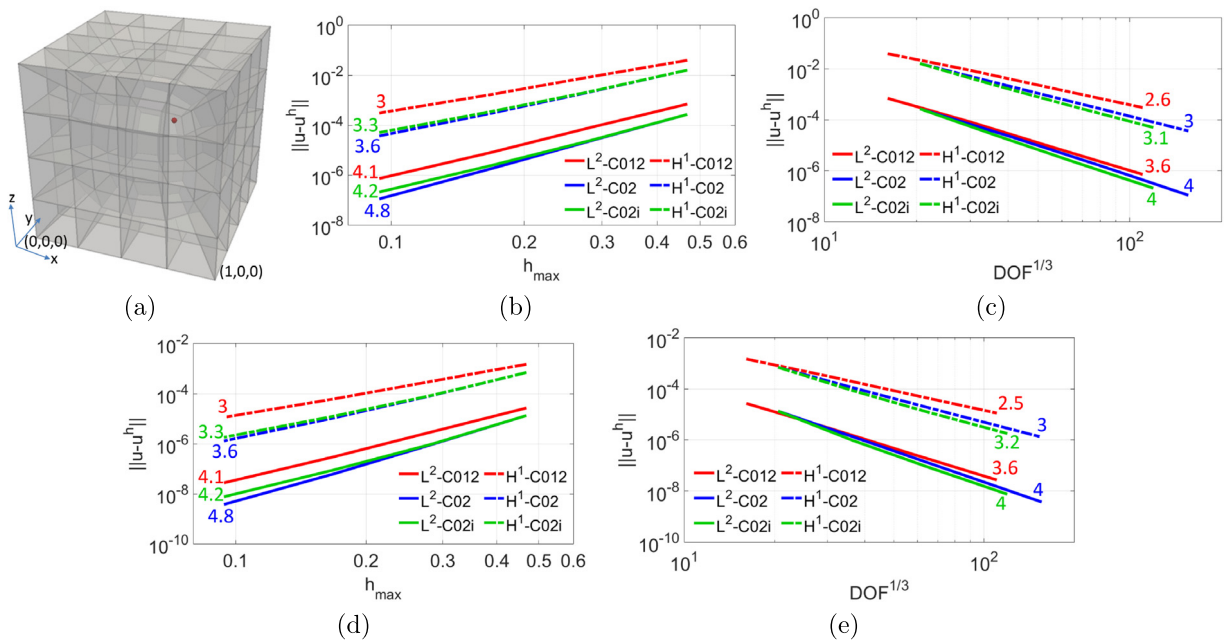


Fig. 20. Solving Poisson's equation on a cube model. (a) The input control mesh, (b, c) convergence to the exact solution in Eq. (38), and (d, e) convergence to the exact solution in Eq. (39). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Next for 3D models, we first perform a patch test by solving Poisson's equation on a unit cube domain $[0, 1]^3$. The input control mesh is shown in Fig. 20(a), where general 3D extraordinary vertices (not generated by sweeping 2D

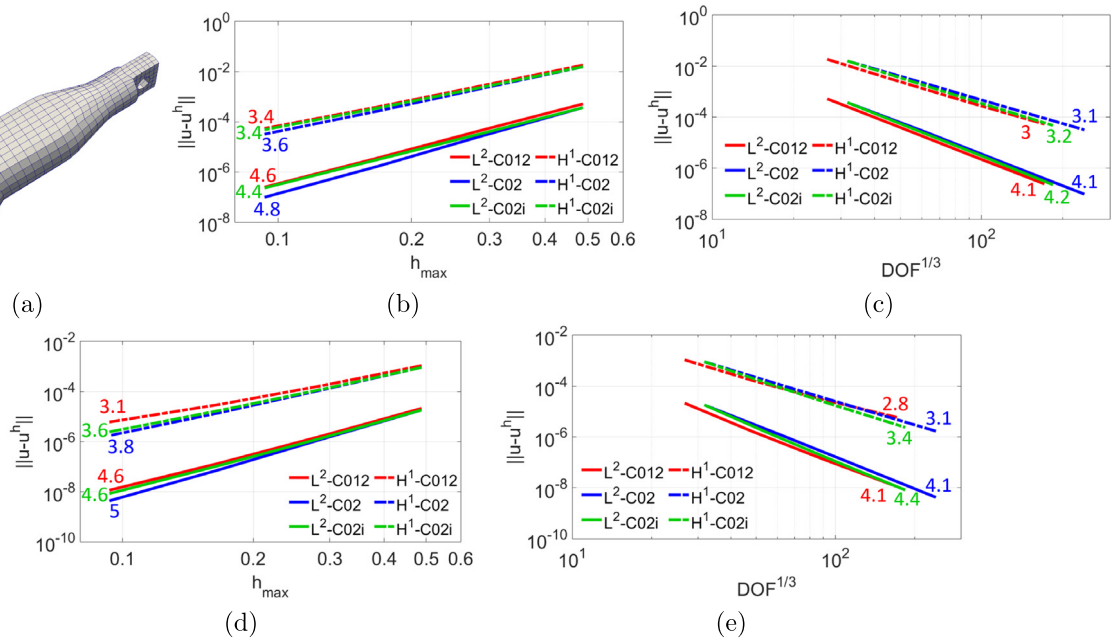


Fig. 21. Solving Poisson's equation on a rod model. (a) The input control mesh, (b, c) convergence to the exact solution in Eq. (38), and (d, e) convergence to the exact solution in Eq. (39). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ones) are allowed; see the red circle for example. The solution field is given as $u(x, y, z) = x$. Dirichlet boundary conditions are strongly imposed at boundary faces $x = 0$ and $x = 1$. We solve the problem using three blended constructions: $C012$, $C02$ and $C02i$ constructions. We point out that $C02$ and $C02i$ constructions are identical on an input mesh. The patch test is passed with machine precision under all three constructions, with the overall L^2 and H^1 errors in the order of 10^{-15} and 10^{-14} , respectively. Similar results are observed for patch tests along y and z directions.

We further study the convergence behavior on three 3D models; see their corresponding input control meshes in Figs. 20(a), 21(a) and 22(a). Poisson's equation is again adopted with the following two exact solutions,

$$u(x, y, z) = \sin(\pi x) \sin(\pi y) \sin(\pi z) \quad (38)$$

and

$$u(x, y, z) = e^{(x+y+z)/3}. \quad (39)$$

As in 2D, only Dirichlet boundary conditions are considered. They can be strongly or weakly imposed, yielding almost the same results (L^2 and H^1 error) in our tests. For each model, we apply all three constructions ($C012$, $C02$ and $C02i$). Global refinement is performed for each model. All the computations of the 3D models were carried out in the Bridges system at the Pittsburgh Supercomputing Center [43,44]. In Figs. 20–22, the convergence curves are plotted with respect to the maximum element size ($h_{\max}$) as well as the cube root of DOF ($\text{DOF}^{1/3}$). The L^2 -norm error is shown in solid lines whereas the H^1 -norm error is shown in dashed lines. Red, blue and green curves indicate the results using $C012$, $C02$ and $C02i$ constructions, respectively. We can observe even faster convergence rates than in 2D (4 for the L^2 -norm error and 3 for the H^1 -norm error) when the rates are plotted with respect to $\text{DOF}^{1/3}$. We observe that given the same element size, $C02$ and $C02i$ constructions introduce more DOF and thus yield smaller error compared to the $C012$ construction.

When plotting them with respect to $\text{DOF}^{1/3}$, the convergence rates are slightly different from those computed in terms of $h_{\max}$. This is because the change of $\text{DOF}^{1/3}$ is not the same as the change of $h_{\max}$ during refinement. We

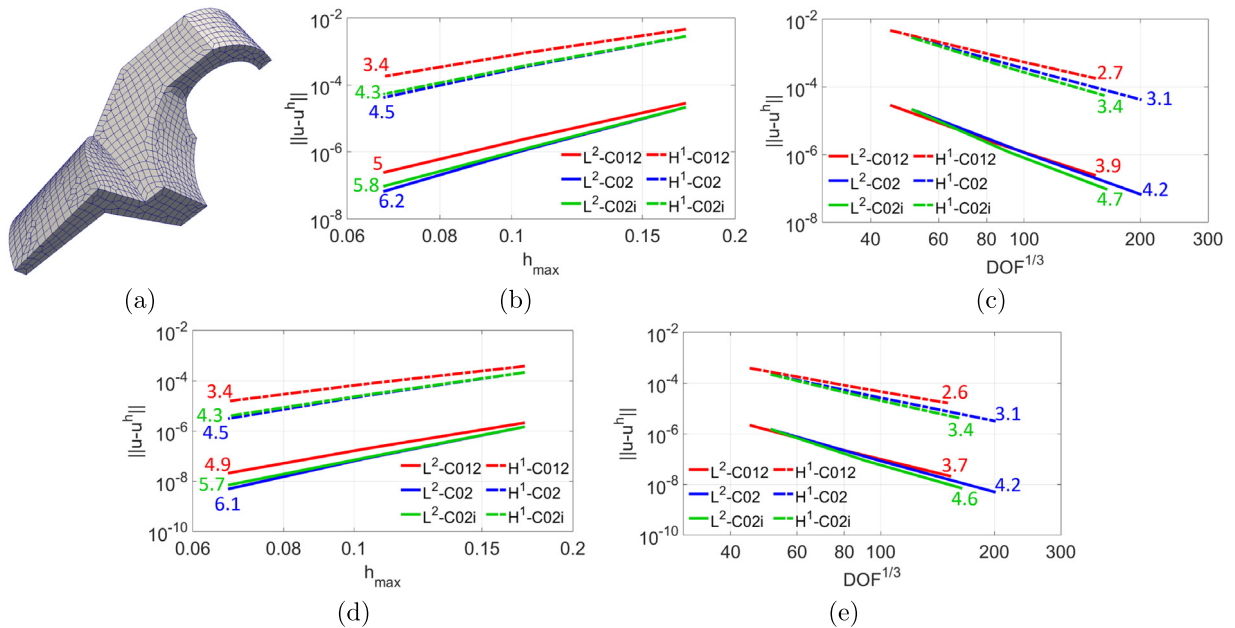


Fig. 22. Solving Poisson's equation on a hook model. (a) The input control mesh, (b, c) convergence to the exact solution in Eq. (38), and (d, e) convergence to the exact solution in Eq. (39). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7

The ratio γ in unstructured hex meshes.

Models	C012 construction	C02 construction	C02i construction
Cube	1.18	1.2	1.06
Rod	1.12	1.2	1.06
Hook	1.32	1.46	1.23

define a ratio to measure these changes

$$\gamma = \log \left(\frac{\text{DOF}^{1/3} \text{ of refined mesh}}{\text{DOF}^{1/3} \text{ of given mesh}} \right) / \log \left(\frac{h_{\max} \text{ of given mesh}}{h_{\max} \text{ of refined mesh}} \right). \quad (40)$$

For each 3D model, the ratio γ is constant for a specific blended construction; see Table 7. We can observe that γ is always larger than 1, which is caused by introducing extra DOF in the blended constructions. It implies that in 3D, DOF change faster than $h_{\max}$. In other words, more than twice the number of DOF (i.e., $\gamma + 1$) are introduced when the element size decreases by half. The additional portion of DOF ($\gamma + 1 - 2 = \gamma - 1$) further improves the solution accuracy, which helps yield a faster convergence rate than the expected one when plotting using $h_{\max}$. We can also observe that γ is particularly large in the hook model, where there is a large portion of irregular elements with many extra DOF introduced. Note that in the rod model shown in Fig. 21, all the extraordinary vertices can be obtained by sweeping 2D counterparts.

We define another ratio called the *DOF/element ratio* as the total DOF divided by the number of elements. In Table 8, we summarize this ratio for each hex model in three blended constructions. Compared to the cube and rod model, the hook model contains the most extraordinary vertices where many extra DOF are introduced in irregular elements. Therefore, its DOF/element ratio is also the largest. In each construction, the DOF/element ratio decreases as the element size decreases, and the change of the ratio between two consecutive refinement steps becomes smaller and smaller. As expected, a C02 blended construction generally needs a lot more DOF than a C012 construction. We also observe that even though C02 and C02i constructions have the same DOF/element ratio in the input mesh, the ratio decreases much faster in a C02i construction because it introduces many fewer DOF in each refinement step.

Table 8

The DOF/element ratio during refinement.

Refinement step	C012 construction				C02 construction				C02i construction			
	0	1	2	3	0	1	2	3	0	1	2	3
Cube	9.6	7.7	6.4	5.7	20.0	17.6	16.5	16.0	20.0	13.4	9.4	7.2
Rod	14.3	10.2	8.0	6.9	24.0	21.2	20.0	19.2	24.0	16.1	11.3	8.6
Hook	18.2	13.8	10.8	N/A	26.6	25.1	24.4	N/A	26.6	18.5	13.2	N/A

Moreover, the DOF/element ratio becomes large even in a C012 construction if an input hex mesh is unstructured with many extraordinary vertices (e.g. the hook model), where many Bézier points are introduced in irregular elements.

Discussion 6.1. In our numerical examples, all the three blended constructions can yield the expected optimal convergence rates with respect to the maximum element size. C02 and C02i constructions generally exhibit slightly higher convergence rates than the C012 construction, so they are better options when only global refinement is needed. However, they introduce many more DOF to the input mesh than the C012 construction; see the DOF/element ratio at refinement step 0 in Table 8. When local refinement is desired, the C012 construction serves as the most efficient candidate, because it introduces the minimal extra DOF and minimum C^0 faces/edges/vertices to the input mesh. We will further study local refinement based on the blended B-spline construction in the future.

7. Conclusion and future work

In this paper, we have presented a new blended B-spline method exhibiting optimal convergence rates when unstructured quad/hex meshes are taken as control meshes in IGA. Various spline functions are defined on different types of elements, with the truncation mechanism employed to connect regular and irregular patches. Three blended constructions (C012, C02 and C02i) were studied, each introducing a different number of DOF and possessing different properties (Table 3). In all these blended constructions, the proposed spline functions form a non-negative partition of unity, are linearly independent, and preserve consistent parameterization in refinement. We investigated these three blended constructions in IGA and observed optimal convergence rates in all the tested 2D and 3D models.

In the future, applying the blended B-spline construction to the Kirchhoff–Love shell would be promising [45], where we need to impose G^1 continuity around 2D extraordinary vertices instead of C^0 . Building hierarchical splines based on a blended B-spline construction would also be interesting, where we can rigorously study local refinement on unstructured hex meshes. Another problem worthy of investigation is the memory-efficient storage of Bézier extraction matrices in 3D. When tricubic splines are used, the dimension of such matrices for each element is in the order of 64×64 . Currently such matrices are stored as dense matrices in our implementation. Switching to sparse matrices could significantly reduce memory consumption. Another challenging problem is how to impose G^1 continuity across spoke faces in an unstructured hex mesh. This problem has not been studied in the literature.

Acknowledgments

X. Wei and Y. Zhang were supported in part by the PECASE Award N00014-16-1-2254 and NSF CAREER Award OCI-1149591. D. Toshniwal and T.J.R. Hughes were partially supported by the Office of Naval Research (Grant Nos. N00014-17-1-2119, N00014-17-1-2039, and N00014-13-1-0500), and by the Army Research Office (Grant No. W911NF-13-1-0220). H. Speleers and C. Manni were supported in part by the MIUR “Futuro in Ricerca 2013” Programme through the project DREAMS (RBFR13FBI3). X. Li was supported by the NKBRPC (2011CB302400), SRF for ROCS SE, and the Youth Innovation Promotion Association CAS. J.A. Evans was supported by the Air Force Office of Scientific Research under Grant No. FA9550-14-1-0113. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the NSF Award OCI-1053575. In particular, we run our codes on the Bridges system (NSF Award number ACI-1445606) at the Pittsburgh Supercomputing Center

In addition to the C012 blended construction in Sections 3 and 4, we here present two alternatives. One is the C02 construction, which only involves vertex-associated functions B_i^v and Bézier functions B_k^0 . The other, called the C02i construction, is an improvement of C02 by introducing fewer DOF in refinement.

Table 9

Possible spline functions defined on different types of elements in C02 and C02i constructions.

Construction type	C02 construction			C02i construction		
	B_i^v	B_j^f	B_k^0	B_i^v	B_j^f	B_k^0
Regular non-transition	Yes	No	No	Yes	No	No
Irregular non-transition	No	No	Yes	No	Yes	Yes
Regular transition	Yes	No	Yes	Yes	No	Yes
Irregular transition	No	No	Yes	No	Yes	Yes

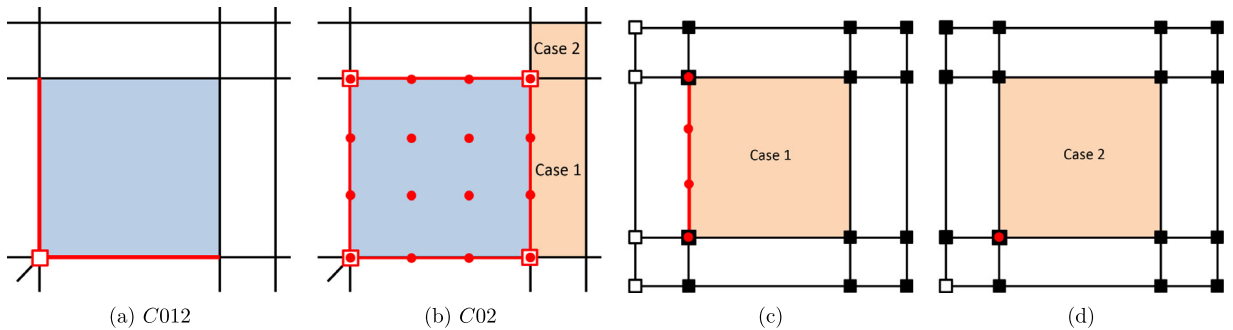


Fig. 23. (a) C^0 edges (red edges) and a C^0 vertex (the red open square) in the C012 construction, (b) C^0 edges (red edges) and C^0 vertices (red open squares) in the C02 construction, and (c, d) two cases of Bézier functions B_k^0 with support on a regular transition element (shaded orange). In (a, b), the blue shade is an irregular element and red circles represent added B_k^0 . In (c, d), the black filled squares and black open squares represent truncated vertex-associated functions B_i^v with and without support on the orange element, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Compared to the C012 construction, the C02 construction utilizes different ways to address: (1) how C^0 tags are assigned in Step 1, and (2) how C^0 tags pass on in refinement. Recall that in a C012 construction, C^0 tags are assigned to extraordinary edges/vertices, spoke faces/edges and boundary faces/edges/vertices. In contrast, the C02 construction assigns C^0 tags to all the faces, edges and vertices of an irregular element; see Fig. 23(a, b) for a comparison, where the red open square and red edges represent a C^0 vertex and C^0 edges, respectively. As a result, more Bézier points are introduced in the C02 construction and all face-point-associated functions B_j^f (or body-point-associated functions B_j^b) are essentially Bézier functions B_k^0 after truncation. Therefore, only B_k^0 have influence on each irregular element. In other words, the Bézier representation is employed for each irregular element. On the other hand, only uniform C^2 B-splines are defined on a regular non-transition element, whereas both B_i^v and B_k^0 are defined on a regular transition element. Given an irregular element (shaded blue) in Fig. 23(b), all its edges and vertices are assigned with C^0 tags in the C02 construction, so all the corresponding 16 B_k^0 (red circles) are added. Some of these B_k^0 also have support on neighboring regular transition elements. As shown in Fig. 23(c, d), a regular transition element (shaded orange) may share either an edge (Case 1) or only a vertex (Case 2) with this irregular element, where those B_k^0 associated with red circles have support on the orange element. In Table 9, we summarize the types of spline functions (B_i^v , B_j^f and B_k^0) defined on different elements for the C02 and C02i constructions in 2D.

Truncation is only needed in regular transition elements, where certain B_i^v are truncated with respect to active Bézier children. Compared with Fig. 5(b, c), after truncation of involved B_i^v in Fig. 23(c, d), the ordinates of the functions associated with black open squares are all zero on the orange element. In other words, these truncated B_i^v do not have support on it. Therefore, only the remaining truncated B_i^v (black filled squares) and added B_k^0 (red circles) are defined on a regular transition element, and the total number of these functions is always 16. Moreover, it is easy to verify that these 16 functions are locally linearly independent on the regular transition element (shaded orange). The underlying Bézier extraction matrix (with truncation) is a 16×16 matrix of full rank. The same argument can be easily extended to 3D, where a regular transition element may share a face, an edge or a vertex with

⁵ Note that two red circles overlay with black filled squares in Fig. 23(c) and one red circle overlays with a black filled square in Fig. 23(d).

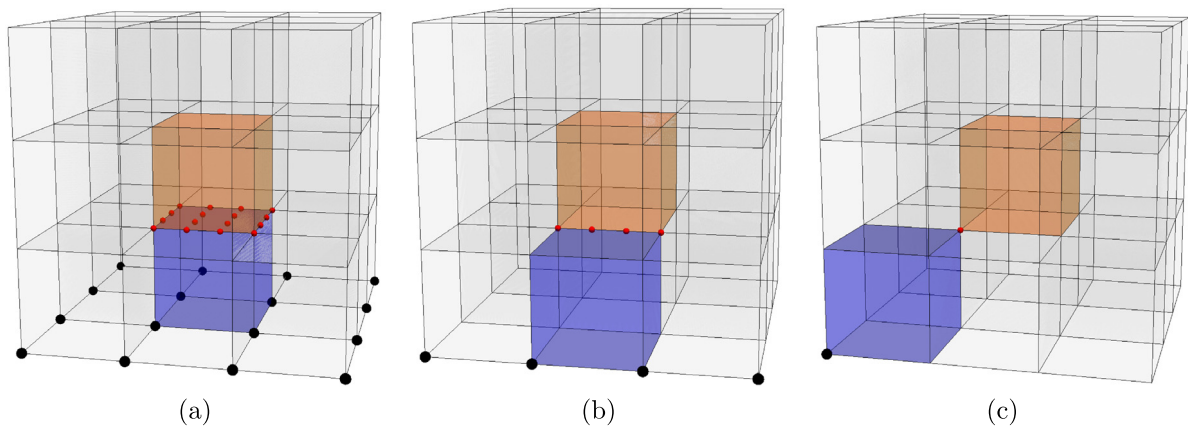


Fig. 24. A regular transition element (shaded orange) sharing a face (a), an edge (b) or a vertex (c) with a neighboring irregular element (shaded blue). Red circles represent added Bézier functions B_k^0 to the orange element, whereas black circles represent vertex-associated functions B_i^v that no longer have support on the orange element after truncation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a neighboring irregular element. We show these three cases in Fig. 24 where the regular transition element and its neighboring irregular element are shaded orange and blue, respectively. Red circles represent added B_k^0 to the orange element, whereas black circles represent B_i^v that no longer have support on the orange element after truncation. In all cases, there are always 64 functions defined on a regular transition element. Furthermore, it can be verified that the Bézier extraction matrix (with truncation) in each case is a 64×64 matrix of full rank.

Refinement in the $C02$ construction is the same as that of the $C012$ construction, except the step of how to update C^0 tags in the refined mesh. In the $C02$ construction, all the sub-elements of an irregular element are irregular and all the newly generated faces/edges/vertices involving refinement of an irregular element are assigned with C^0 tags. As a result, the Bézier representation is also used for all the sub-elements of an irregular element; see Fig. 13(b). Passing C^0 tags in such a “dense” manner is easy for implementation, where every patch is determined by either a Bézier control mesh or a uniform B-spline control mesh.

We now describe the $C02i$ construction, an improved version of the $C02$ construction that introduces fewer DOF during refinement. During refinement, instead of assigning all the newly generated faces/edges/vertices of an irregular element with C^0 tags, we follow the same way as in the $C012$ construction to assign C^0 tags to: (1) sub-faces, newly generated edges and the center vertex of each C^0 face, (2) sub-edges and the midpoint of each C^0 edges, and (3) existing C^0 vertices. As a result, the $C02i$ construction introduces B_j^f or B_j^b within the irregular submesh, while leaving B_k^0 on the interface of irregular and regular submeshes. Fig. 13(c) shows an example of the $C02i$ construction on the refined mesh of Fig. 23(a). C^0 edges are marked in red, whereas B_j^f and B_k^0 are represented by green circles and red circles, respectively. The types of functions defined on different elements are also summarized in Table 9. Performing truncation in a regular transition element is the same as the $C02$ construction, while truncating B_j^f with respect to active Bézier children is the same as in the $C012$ construction. Local linear independence can be verified in a manner similar to the $C02$ construction.

Discussion A.1. Both $C02$ and $C02i$ constructions give rise to locally linearly independent spline functions. This is an advantage over the $C012$ construction, which only features (global) linear independence. Compared to the $C012$ construction, the implementation of $C02$ and $C02i$ constructions is easier because they have at most two types of functions defined on a particular element, whereas all three types of functions are involved in the $C012$ construction (regular element, irregular element, and regular transition element). On the other hand, $C02$ and $C02i$ constructions require more DOF as shown in the examples in Section 6.

- [1] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Comput. Methods Appl. Mech. Engrg.* 194 (2005) 4135–4195.

- [2] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, John Wiley & Sons, 2009.
- [3] Y. Zhang, *Geometric Modeling and Mesh Generation from Scanned Images*, Chapman and Hall/CRC, 2016.
- [4] D. Burkhart, B. Hamann, G. Umlauf, Iso-geometric finite element analysis based on Catmull-Clark subdivision solids, *Comput. Graph. Forum* 29 (2010) 1575–1584.
- [5] W. Wang, Y. Zhang, L. Liu, T.J.R. Hughes, Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology, *Comput. Aided Des.* 45 (2) (2013) 351–360.
- [6] X. Wei, Y. Zhang, T.J.R. Hughes, Truncated hierarchical tricubic C^0 spline construction on unstructured hexahedral meshes for isogeometric analysis applications, *Comput. Math. Appl.* 74 (9) (2017) 2203–2220.
- [7] T. Nguyen, K. Karčiauskas, J. Peters, A comparative study of several classical, discrete differential and isogeometric methods for solving Poisson's equation on the disk, *Axioms* 3 (2014) 280–300.
- [8] X. Wei, Y. Zhang, T.J.R. Hughes, M.A. Scott, Truncated hierarchical Catmull-Clark subdivision with local refinement, *Comput. Methods Appl. Mech. Engrg.* 291 (2015) 1–20.
- [9] X. Wei, Y. Zhang, T.J.R. Hughes, M.A. Scott, Extended truncated hierarchical Catmull-Clark subdivision, *Comput. Methods Appl. Mech. Engrg.* 299 (2016) 316–336.
- [10] M.A. Scott, T-splines as a Design-Through-Analysis Technology (Ph.D. thesis), The University of Texas at Austin, 2011.
- [11] M. Majeed, F. Cirak, Isogeometric analysis using manifold-based smooth basis functions, *Comput. Methods Appl. Mech. Engrg.* 316 (2017) 547–567.
- [12] M. Wu, B. Mourrain, A. Galligo, B. Nkonga, Hermite type spline spaces over rectangular meshes with complex topological structures, *Commun. Comput. Phys.* 21 (3) (2017) 835–866.
- [13] W. Wang, Y. Zhang, G. Xu, T.J.R. Hughes, Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline, *Comput. Mech.* 50 (2012) 65–84.
- [14] M.A. Scott, D.C. Thomas, E.J. Evans, Isogeometric spline forests, *Comput. Methods Appl. Mech. Engrg.* 269 (2014) 222–264.
- [15] F. Buchegger, B. Jüttler, A. Mantzafaris, Adaptively refined multi-patch B-splines with enhanced smoothness, *Appl. Math. Comput.* 272 (2016) 159–172.
- [16] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications, *Comput. Aided Des.* 45 (2) (2013) 395–404.
- [17] M. Kapl, V. Vitrih, B. Jüttler, K. Birner, Isogeometric analysis with geometrically continuous functions on two-patch geometries, *Comput. Math. Appl.* 70 (7) (2015) 1518–1538.
- [18] A. Collin, G. Sangalli, T. Takacs, Approximation properties of multi-patch C^1 isogeometric spaces. 2015. arXiv preprint arXiv:1509.07619.
- [19] A. Collin, G. Sangalli, T. Takacs, Analysis-suitable G^1 multi-patch parametrizations for C^1 isogeometric spaces, *Comput. Aided Geom. Design* 47 (2016) 93–113.
- [20] B. Mourrain, R. Vidunas, N. Villamizar, Dimension and bases for geometrically continuous splines on surfaces of arbitrary topology, *Comput. Aided Geom. Design* 45 (2016) 108–133.
- [21] G. Sangalli, T. Takacs, R. Vázquez, Unstructured spline spaces for isogeometric analysis based on spline manifolds, *Comput. Aided Geom. Design* 47 (2016) 61–82.
- [22] M. Bercovier, T. Matskewich, *Smooth Bézier Surfaces over Unstructured Quadrilateral Meshes*, Springer, 2017.
- [23] M.A. Scott, R.N. Simpson, J.A. Evans, S. Lipton, S.P.A. Bordas, T.J.R. Hughes, T.W. Sederberg, Isogeometric boundary element analysis using unstructured T-splines, *Comput. Methods Appl. Mech. Engrg.* 254 (2013) 197–221.
- [24] T. Nguyen, J. Peters, Refinable C^1 spline elements for irregular quad layout, *Comput. Aided Geom. Design* 43 (2016) 123–130.
- [25] 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, *Comput. Methods Appl. Mech. Engrg.* 327 (2017) 411–458.
- [26] D. Toshniwal, H. Speleers, T.J.R. Hughes, Analysis-suitable spline spaces of arbitrary degree on unstructured quadrilateral meshes. 2017. ICES REPORT 17-16, Institute for Computational Engineering and Sciences, The University of Texas at Austin.
- [27] X. Wei, Y. Zhang, L. Liu, T.J.R. Hughes, Truncated T-splines: fundamentals and methods, *Comput. Methods Appl. Mech. Engrg.* 316 (2017) 349–372.
- [28] X. Yuan, K. Tang, Rectified unstructured T-splines with dynamic weighted refinement for improvement in geometric consistency and approximation convergence, *Comput. Methods Appl. Mech. Engrg.* 316 (2017) 373–399.
- [29] G. Xu, M. Li, B. Mourrain, T. Rabczuk, J. Xu, S.P.A. Bordas, Constructing IGA-suitable planar parameterization from complex CAD boundary by domain partition and global/local optimization, *Comput. Methods Appl. Mech. Engrg.* 328 (2018) 175–200.
- [30] U. Reif, A refineable space of smooth spline surfaces of arbitrary topological genus, *J. Approx. Theory* 90 (2) (1997) 174–199.
- [31] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes, *Graph. Models* 70 (2008) 76–86.
- [32] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: the truncated basis for hierarchical splines, *Comput. Aided Geom. Design* 29 (2012) 485–498.
- [33] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes, *Graph. Models* 70 (2008) 76–86.
- [34] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes, *Graph. Models* 70 (2008) 76–86.
- [35] T.W. Sederberg, T.J.R. Hughes, Local refinement of analysis-suitable T-splines, *Comput. Methods Appl. Mech. Engrg.* 222 (2004) 221–232.
- [36] T.W. Sederberg, T.J.R. Hughes, Local refinement of analysis-suitable T-splines, *Comput. Methods Appl. Mech. Engrg.* 222 (2004) 221–232.
- [37] T.W. Sederberg, T.J.R. Hughes, Local refinement of analysis-suitable T-splines, *Comput. Methods Appl. Mech. Engrg.* 222 (2004) 221–232.
- [38] P.B. Bornemann, F. Cirak, A subdivision-based implementation of the hierarchical B-spline finite element method, *Comput. Methods Appl. Mech. Engrg.* 253 (2013) 584–598.



- [39] C. Giannelli, B. Jüttler, H. Speleers, Strongly stable bases for adaptively refined multilevel spline spaces, *Adv. Comput. Math.* 40 (2) (2014) 459–490.
- [40] H. Speleers, C. Manni, Effortless quasi-interpolation in hierarchical spaces, *Numer. Math.* 132 (2016) 155–184.
- [41] W. Boehm, A. Müller, On de Casteljau's algorithm, *Comput. Aided Geom. Design* 16 (7) (1999) 587–605.
- [42] A. Embar, J. Dolbow, I. Harari, Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements, *Internat. J. Numer. Methods Engrg.* 83 (7) (2010) 877–898.
- [43] J.T. Towns, M. Cockerill, I. Dahan, K. Foster, A. Gaither, V. Grimshaw, S. Hazlewood, D. Lathrop, G.D. Lifka, R. Peterson, J.R. Roskies, N. Scott, Wilkins-Diehr, XSEDE: accelerating scientific discovery, *Comput. Sci. Eng.* 16 (5) (2014) 62–74.
- [44] N.A. Nystrom, M.J. Levine, R.Z. Roskies, J.R. Scott, Bridges: a uniquely flexible HPC resource for new communities and data analytics, in: *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, 2015, pp. 30:1–30:8.
- [45] H. Casquero, L. Liu, Y. Zhang, A. Reali, J. Kiendl, H. Gomez, Arbitrary-degree T-splines for isogeometric analysis of fully nonlinear Kirchhoff-Love shells, *Comput. Aided Des.* 82 (2017) 140–153.