Comput. Methods Appl. Mech. Engrg. 269 (2014) 222–264

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Isogeometric spline forests

M.A. Scott [a,*], D.C. Thomas [b], E.J. Evans [c]

[a] Department of Civil and Environmental Engineering, Brigham Young University, Provo, UT 84602, USA
[b] Department of Physics & Astronomy, Brigham Young University, Provo, UT 84602, USA
[c] Department of Mathematics, Brigham Young University, Provo, UT 84602, USA

ARTICLE INFO

Article history:
Received 7 August 2013
Received in revised form 18 October 2013
Accepted 23 October 2013
Available online 8 November 2013

Keywords:
Isogeometric analysis
Hierarchical splines
Adaptive mesh refinement
Advection–diffusion

ABSTRACT

In this paper we present isogeometric spline forests. An isogeometric spline forest is a hierarchical spline representation capable of representing surfaces or volumes of arbitrarily complex geometry and topological genus. Spline forests can accomodate arbitrary degree and smoothness in the underlying hierarchical basis as well as non-uniform knot interval configurations. We describe adaptive *h*-refinement and coarsening algorithms for isogeometric spline forests and develop a Bézier extraction framework which provides a simple and efficient single level finite element description of the complex multi-level, unstructured hierarchical spline basis. We then demonstrate the potential of spline forests as a basis for analysis in the context of transient advection–diffusion problems where fully integrated adaptivity is demonstrated for the first time in an isogeometric simulation. In all cases, the adaptive process remains local (even in the case of moving fronts) and preserves exact geometry at the coarsest level of the discretization. The accuracy and robustness of the approach is demonstrated in all cases.

1. Introduction

A spline forest is an unstructured conforming arrangement of hierarchical spline trees. A spline forest can represent complex geometry of arbitrary topological genus for both surfaces and solids. Additionally, arbitrary degree and higher-order smoothness can be accomodated in the forest. A renewed emphasis has been placed on hierarchical spline representations [1–5] in response to the popularity of isogeometric analysis (IGA) [6]. IGA has emerged as an important alternative to traditional design and analysis methodologies. IGA, introduced in the finite element analysis (FEA) community [6,7], establishes a fundamental link with computer aided design (CAD) geometry by using the geometric basis directly as the basis for analysis. Consequently, the finite element mesh *is* an exact representation of the geometry. Areas of application of isogeometric analysis are varied and continue to grow rapidly. A small sampling includes fluids and turbulence [8–11], fluid–structure interaction [12–15], incompressibility [16–18], structural and vibration analysis [19–21], plates and shells [22–26], fracture and damage [27–29], phase-field analysis [30,31,29], large deformation structural analysis with severe mesh distortion [32], shape optimization [33–36], acoustics [37], and electromagnetics [38]. Interestingly, this success has attracted researchers within the computer aided geometric design (CAGD) community to develop new "analysis-aware" geometric technologies [39–46].

In the context of analysis, hierarchical splines [1] present the attractive possibility of defining extremely powerful and flexible adaptive multiresolution schemes which are smooth, highly localized, and geometrically exact. Unfortunately, like NURBS and B-splines, current hierarchical spline formulations are restricted to four-sided domains. This severely limits their

* Corresponding author.
E-mail address: michael.scott@byu.edu (M.A. Scott).

0045-7825/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cma.2013.10.024

utility as a design tool and as a basis for isogeometric analysis. Spline forests overcome this four-sided topological limitation and allow for the definition of hierarchical spline representations of complex geometry. Fundamental to this is the development of intertree coordinate transformations, and accompanying algorithms, which allow for arbitrary relative orientations of two- and three-dimensional spline trees and mixed periodic and nonperiodic identification of tree boundaries. To accomplish this in a scalable manner, we leverage recent developments in the finite element adaptive mesh refinement community [47,48] where forests of octrees are used for massively parallel computations. Using these transformations, all algorithms for a single hierarchical spline, such as local refinement, coarsening, and Bézier extraction, can be extended to the forest in a single consistent, geometrically exact manner. For simplicity of exposition, we restrict ourselves to a C^0 formulation across tree interfaces. We note that smoother interface conditions are under development and will be described in a forthcoming publication.

On a fundamental level, spline forests can be viewed as a superset of both single-level spline representations like B-splines, NURBS, and analysis-suitable T-splines [49–54] and multi-level spline representations like hierarchical B-splines, inheriting the advantages of each in a single unified framework. As an integrative technology, the possibilities are intriguing. We envision advanced analysis-suitable design operations being performed on the first level of the forest, and additional levels being added to accomodate high-performance, multi-resolution analysis needs in downstream applications, all in a geometrically exact, analysis-suitable setting. Additionally, since spline forests are constructed from quadrilateral and hexahedral topological decompositions they are fully compatible with existing quadrilateral and hexahedral meshing technology. In fact, given a conforming quadrilateral or hexahedral mesh a corresponding spline forest can be constructed automatically. Spline forests provides a firm foundation upon which integrated engineering design and analysis methodologies can be built [55].

1.1. Structure and content of the paper

In Section 2, 3 introductory material is presented for Bézier and B-spline curves, surfaces, and solids with an eye towards spline forest generalizations. Hierarchical B-splines are then described in detail in Section 4, 5. Section 6 describes h-adaptivity for spline trees and forests. Detailed algorithms are presented in all cases. We introduce the basic theory of spline forests in Section 7, focusing on topological considerations and intertree transformations and accompanying algorithms. Section 8 generalizes the Bézier extraction framework, introduced in [56,57], to the spline forest setting with detailed derivations. We then demonstrate the potential of spline forests as a basis for isogeometric analysis in Section 9. We focus on transient advection–diffusion problems where full adaptivity (coupled refinement/coarsening/extraction/analysis) is demonstrated for the first time for both surfaces and solids in an isogeometric simulation. Finally, in Section 10 we draw conclusions.

We note that the term spline "forest" was chosen to convey the idea that a forest is composed of a number of carefully arranged spline "trees." The spline trees are characterized by their hierarchical structure. Forests of octrees are common in massively parallel adaptive mesh refinement [47]. Terminology like leaves, parent, child, siblings, and pruning is analogous to that used to describe quad and octree data structures.

We use d_p and d_s throughout to denote the number of parametric and spatial dimensions, respectively. For quick reference, Section A lists the most important notational conventions used throughout the text and where they are defined.

2. Bézier fundamentals

Bézier curves, surfaces, and volumes are a fundamental building block for more advanced computer aided geometric design (CAGD) technologies like B-splines, NURBS, T-splines, and hierarchical B-splines [58,59]. Additionally, as described in Section 8.2 and [57,56], they are used as finite elements in the context of Bézier extraction and isogeometric analysis.

2.1. The univariate Bernstein basis

The univariate Bernstein basis functions are written as

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

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

- *Partition of unity.*

$$\sum_{i=1}^{p+1} B_{i,p}(\xi) = 1 \quad \forall \xi \in [-1,1]$$

- *Pointwise nonnegativity.*

$$B_{i,p}(\xi) \geqslant 0 \quad \forall \xi \in [-1,1]$$

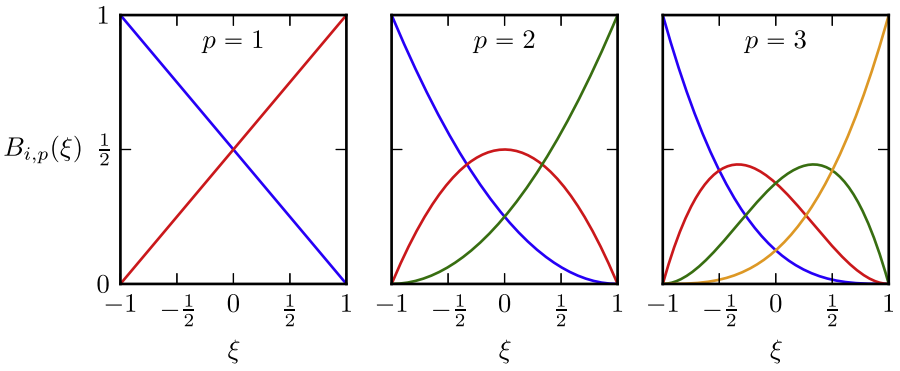


Fig. 1. The Bernstein basis for polynomial degrees $p = 1, 2, 3$.

- *Endpoint interpolation.*

$$B_{1,p}(-1) = B_{p+1,p}(1) = 1$$

- *Symmetry.*

$$B_{i,p}(\xi) = B_{p+1-i,p}(-\xi) \quad \forall \xi \in [-1, 1]$$

Fig. 1 shows the Bernstein basis for polynomial degrees $p = 1, 2, 3$.

2.2. The multivariate Bernstein basis

We denote a multivariate Bernstein basis function of degree $\mathbf{p} = \{p_\ell\}_{\ell=1}^{d_p}$ by $B_{a,\mathbf{p}} : \bar{\Omega} \to \mathbb{R}^+ \cup 0$ where $a = 1, \ldots, n_b$, $n_b = \prod_{\ell=1}^{d_p}(p_\ell + 1)$, and $\bar{\Omega} = [-1, 1]^{d_p}$. A multivariate Bernstein basis function is formed as the tensor product of univariate basis functions $B_{i,p_\ell} : [-1, 1] \to \mathbb{R}^+ \cup 0$, with $i = 1, \ldots, p_\ell + 1$. For the bivariate case we have that

$$B_{a(i,j),\mathbf{p}}(\boldsymbol{\xi}) = B_{i,p_1}(\xi_1) B_{j,p_2}(\xi_2) \tag{2}$$

with

$$a(i,j) = (p_1 + 1)(j - 1) + i \tag{3}$$

and for the trivariate case we have that

$$B_{a(i,j,k),\mathbf{p}}(\boldsymbol{\xi}) = B_{i,p_1}(\xi_1) B_{j,p_2}(\xi_2) B_{k,p_3}(\xi_3) \tag{4}$$

with

$$a(i,j,k) = (p_1 + 1)(p_2 + 1)(k - 1) + (p_1 + 1)(j - 1) + i \tag{5}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_{d_p}) \in \bar{\Omega}$.

3. B-spline fundamentals

We present fundamental concepts related to B-splines in preparation for later generalizations to spline forests where hierarchies of B-splines are employed in an unstructured setting. In some places, new notational conventions are introduced to facilitate the transition to localized hierarchical concepts. In Section 3.1 we generalize the notion of the index space [6] to allow for index coordinates which are real numbers rather than integers. This facilitates and simplifies the introduction of hierarchical concepts for non-uniform knot configurations with arbitrary knot multiplicities. Additionally, we develop the tensor product topology in Section 3.3 adopting many conventions from T-splines [52,51,53]. This allows us to break free from the traditional global tensor product point of view of multivariate B-splines and develop a precise notation for localized topological quantities in a hierarchy.

3.1. Index and parametric knot vectors

A univariate B-spline space is constructed by specifying a polynomial degree p, a *global* index knot vector $\mathsf{I} = \{i_1 < i_2 < \cdots < i_{n+p+1}\}$ of increasing knot values $i_j \in \mathbb{R}$ where $i_1 = 0$ and $i_{n+p+1} = 1$, and a *global* parametric knot vector $\mathsf{G} = \{s_{i_1} \leqslant s_{i_2} \leqslant \cdots \leqslant s_{i_{n+p+1}}\}$ of non-decreasing knot values $s_{i_j} \in \mathbb{R}$. The global index domain is denoted by $\hat{\Omega}$ and the global parametric domain is denoted by $\hat{\Omega}$. We require that G be open. In other words, the first and last knots have multiplicity

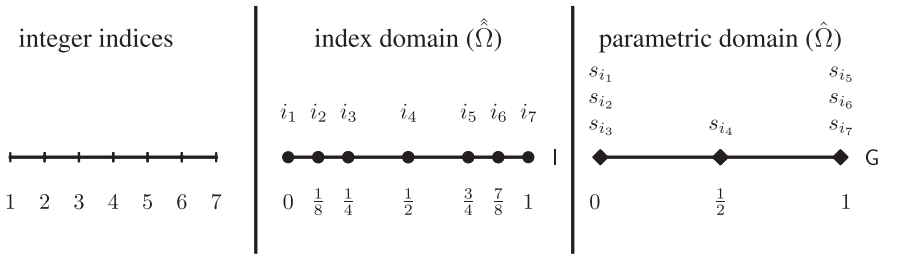


Fig. 2. Integer indices and the corresponding index and parametric domains for a B-spline of degree 2. The set of natural numbers that indexes the knots is shown on the left, the corresponding knots in the index domain in the middle, and the knots in the parametric domain are shown on the right. Index knots are marked with circles (•) and parametric knots are marked with diamonds (♦). Each sequence indexes the sequence to the right. The knot value is given below the knot, while the corresponding symbol is shown above. Repeated knots in the parametric domain are indicated by the number of parametric knot labels s_{i_j} above the knot. Thus, $s_{i_1} = s_{i_2} = s_{i_3} = 0$ and the multiplicity $\mu(\mathbf{G}, s = 0) = \mu(\mathbf{G}, s = 1) = 3$.

$p+1$. We also require that $s_{i_1} = \cdots = s_{i_{p+1}} = 0$ and that $s_{i_{n+1}} = \cdots = s_{i_{n+p+1}} = 1$. Thus, we have that $\hat{\hat{\Omega}} = \hat{\Omega} = [0,1]$. The multiplicity for any knot value $s \in \hat{\Omega}$ is $0 \leqslant \mu(\mathsf{G}, s) \leqslant p+1$ where it will be zero if $s \notin \mathsf{G}$. The relationship between integer indices and the index and parametric knot vectors is illustrated in Fig. 2. Note that the knot values in index space are real numbers, not integers.

3.2. Univariate B-spline basis functions

A univariate B-spline space is spanned by n B-spline basis functions $N_{i_j,p}(s) = N[\mathsf{L}_{i_j}](s)$ where $\mathsf{L}_{i_j} = \{s_{i_j}, s_{i_{j+1}}, \ldots, s_{i_{j+p+1}}\}, j = 1, \ldots, n$, is a subsequence of G called a *local* knot vector. Note that a B-spline basis function is indexed using knots in I but defined in $\hat{\Omega}$. A B-spline basis function $N_{i_j,p}(s)$ can be defined recursively by the Cox-de Boor recursion formula as follows:

$$N_{i_j,0}(s) = \begin{cases} 1 & s_{i_j} \leqslant s < s_{i_{j+1}} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

$$N_{i_j,p}(s) = \frac{s - s_{i_j}}{s_{i_{j+p}} - s_{i_j}} N_{i_j,p-1}(s) + \frac{s_{i_{j+p+1}} - s}{s_{i_{j+p+1}} - s_{i_{j+1}}} N_{i_{j+1},p-1}(s). \tag{7}$$

The de Boor algorithm [58] provides a standard method for evaluating a B-spline, although other possibilities exist [60,61]. A univariate B-spline basis has the following properties:

- *Partition of unity.*

$$\sum_{j=1}^{n} N_{i_j,p}(s) = 1, \quad \forall s \in [0,1]$$

- *Pointwise nonnegativity.*

$$N_{i_j,p}(s) \geqslant 0, \quad j = 1, \ldots, n, \quad \forall s \in [0,1]$$

- *Global linear independence.*

$$\sum_{j=1}^{n} c_j N_{i_j,p}(s) = 0 \Longleftrightarrow c_i = 0, \quad i = 1, \ldots, n, \quad \forall s \in [0,1]$$

- *Local linear independence.* Given an open set $\hat{\Omega}' \subseteq \hat{\Omega}$ the B-spline basis functions having some support in $\hat{\Omega}'$ are linearly independent on $\hat{\Omega}'$.
- *Compact support.*

$$\{s \in [0,1] : N_{i_j,p}(s) > 0\} \subset [s_{i_j}, s_{i_{j+p+1}}]$$

- *Control of continuity.* If s_{i_j} has multiplicity k(i.e., $s_{i_j} = s_{i_{j+1}} = \cdots = s_{i_{j+k-1}}$), then the basis functions are C^{p-k}-continuous at s_{i_j}. When $k = p$, the basis is C^0 and interpolatory at that location.

Fig. 3 shows the B-spline basis functions defined by the global index and parametric knot vectors in Fig. 2. In this case, we construct four quadratic B-spline basis functions $N_{0,2}, N_{\frac{1}{8},2}, N_{\frac{1}{4},2}, N_{\frac{1}{2},2}$ from the local knot vectors

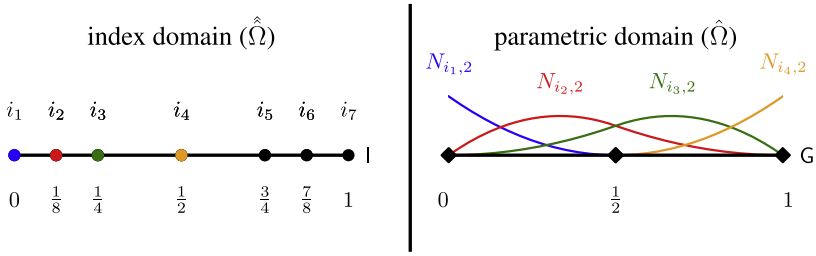


Fig. 3. The quadratic B-spline basis functions associated with the index and parametric knot vectors in Fig. 2. Note that the B-spline basis functions are indexed using knots in I but are defined in the parametric domain $\hat{\Omega}$.

$$\mathsf{L}_0 = \{s_0, s_{\frac{1}{8}}, s_{\frac{1}{4}}, s_{\frac{1}{2}}\} = \left\{0, 0, 0, \frac{1}{2}\right\},$$
$$\mathsf{L}_{\frac{1}{8}} = \{s_{\frac{1}{8}}, s_{\frac{1}{4}}, s_{\frac{1}{2}}, s_{\frac{3}{4}}\} = \left\{0, 0, \frac{1}{2}, 1\right\},$$
$$\mathsf{L}_{\frac{1}{4}} = \{s_{\frac{1}{4}}, s_{\frac{1}{2}}, s_{\frac{3}{4}}, s_{\frac{7}{8}}\} = \left\{0, \frac{1}{2}, 1, 1\right\},$$
$$\mathsf{L}_{\frac{1}{2}} = \{s_{\frac{1}{2}}, s_{\frac{3}{4}}, s_{\frac{7}{8}}, s_1\} = \left\{\frac{1}{2}, 1, 1, 1\right\}.$$

3.3. The tensor product topology

Given corresponding sets of global index and parametric knot vectors $\mathbf{I} = \{\mathsf{I}_\ell\}_{\ell=1}^{d_p}$ and $\mathbf{G} = \{\mathsf{G}_\ell\}_{\ell=1}^{d_p}$ we can define a multi-dimensional mesh topology of vertices, cells, and anchors:

- *Vertices*. We denote a vertex set by $\mathbf{V} = \{\mathbf{v} = \{v_\ell\}_{\ell=1}^{d_p} : v_\ell \in \mathsf{I}_\ell, \forall \ell = 1, \ldots, d_p\}$. This induces a local to global index map such that for any $\mathbf{v} \in \mathbf{V}$, $i_j^\ell = v_{j(\ell)}, \ell = 1, \ldots, d_p$ where $i_j^\ell \in \mathsf{I}_\ell$.
- *Cells*. We denote a cell set by $\mathbf{C} = \{\mathbf{c} = \{c_\ell\}_{\ell=1}^{d_p} : c_\ell \in \{i_1^\ell, \ldots, i_{n_\ell+p_\ell}^\ell\} \subset \mathsf{I}_\ell, \forall \ell = 1, \ldots, d_p\}$. This induces a local to global index map such that for any $\mathbf{c} \in \mathbf{C}$, $i_j^\ell = c_{j(\ell)}, \ell = 1, \ldots, d_p$ where $i_j^\ell \in \mathsf{I}_\ell$. The vertex set of a cell $\mathbf{c} \in \mathbf{C}$ is $\mathbf{V}(\mathbf{c}) = \{\mathbf{v} = \{v_\ell\}_{\ell=1}^{d_p} \in \mathbf{V} : v_\ell \in \{c_{j(\ell)}, \ldots, c_{j(\ell)+1}\} \subset \mathsf{I}_\ell, \forall \ell = 1, \ldots, d_p\}$. The index domain of a cell is $\hat{\hat{\Omega}}(\mathbf{c}) = \times_{\ell=1}^{d_p}[c_{j(\ell)}, c_{j(\ell)+1}]$ and the parametric domain of a cell is $\hat{\Omega}(\mathbf{c}) = \times_{\ell=1}^{d_p}[s_{c_{j(\ell)}}, s_{c_{j(\ell)+1}}]$. We denote the set of cells with non-zero parametric area by $\mathbf{PC} \subset \mathbf{C}$. In the T-spline literature the set of cells with non-zero parametric area constitute the active region [52,51].
- *Anchors*. We denote an anchor set by $\mathbf{F} = \{\mathbf{f} = \{f_\ell\}_{\ell=1}^{d_p} : f_\ell \in \{i_1^\ell, \ldots, i_{n_\ell}^\ell\} \subset \mathsf{I}_\ell, \forall \ell = 1, \ldots, d_p\}$. This induces a local to global index map such that for any $\mathbf{f} \in \mathbf{F}$, $i_j^\ell = f_{j(\ell)}, \ell = 1, \ldots, d_p$ where $i_j^\ell \in \mathsf{I}_\ell$. The vertex set of an anchor $\mathbf{f} \in \mathbf{F}$ is $\mathbf{V}(\mathbf{f}) = \{\mathbf{v} = \{v_\ell\}_{\ell=1}^{d_p} \in \mathbf{V} : v_\ell \in \{f_{j(\ell)}, \ldots, f_{j(\ell)+p_\ell+1}\} \subset \mathsf{I}_\ell, \forall \ell = 1, \ldots, d_p\}$ and the cell set of an anchor $\mathbf{f} \in \mathbf{F}$ is $\mathbf{C}(\mathbf{f}) = \{\mathbf{c} = \{c_\ell\}_{\ell=1}^{d_p} \in \mathbf{C} : c_\ell \in \{f_{j(\ell)}, \ldots, f_{j(\ell)+p_\ell}\} \subset \mathsf{I}_\ell, \forall \ell = 1, \ldots, d_p\}$. The index domain of an anchor is $\hat{\hat{\Omega}}(\mathbf{f}) = \times_{\ell=1}^{d_p}[f_{j(\ell)}, f_{j(\ell)+p_\ell+1}]$ and the parametric domain of an anchor is $\hat{\Omega}(\mathbf{f}) = \times_{\ell=1}^{d_p}[s_{f_{j(\ell)}}, s_{f_{j(\ell)+p_\ell+1}}]$.

Note that the tensor product mesh topology is a simplification of the T-mesh topology presented in [52,51,53]. A tensor product topology for a quadratic by cubic (i.e., $\mathbf{p} = \{2, 3\}$) example is shown graphically in Fig. 4. Note that the tensor product topology is defined in the global index domain $\hat{\hat{\Omega}}$. Fig. 5 illustrates several function and cell quantities from the tensor product topology.

3.4. Multivariate B-spline basis functions

Given a polynomial degree $\mathbf{p} = \{p_\ell\}_{\ell=1}^{d_p}$ we associate a tensor product polynomial B-spline basis function with each anchor $\mathbf{f} \in \mathbf{F}$. A multivariate B-spline basis function $N_{\mathbf{f},\mathbf{p}} : \hat{\Omega} \to \mathbb{R}$ is written as

$$N_{\mathbf{f},\mathbf{p}}(\mathbf{s}) = \prod_{\ell=1}^{d_p} N_{f_\ell,p_\ell}^\ell(s_\ell) \tag{8}$$

where $\mathbf{s}$ is a parametric coordinate in $\hat{\Omega}$, $N_{f_\ell,p_\ell}^\ell(s_\ell) = N[\mathsf{L}_{f_\ell,p_\ell}^\ell](s_\ell)$, and $\mathsf{L}_{f_\ell,p_\ell}^\ell$ is a local knot vector in G_ℓ. A B-spline space $\mathcal{B}$ is spanned by $n = n_1 n_2 \ldots n_{d_p}$ tensor product B-spline basis functions. We denote the set of all B-spline basis functions by $\mathbf{N} = \{N_{\mathbf{f},\mathbf{p}}\}_{\mathbf{f}\in\mathbf{F}}$. Given a weight $w_\mathbf{f} \in \mathbb{R}^+$ for each $\mathbf{f} \in \mathbf{F}$ a *rational* B-spline basis function $R_{\mathbf{f},\mathbf{p}} : \hat{\Omega} \to \mathbb{R}$ can be written as

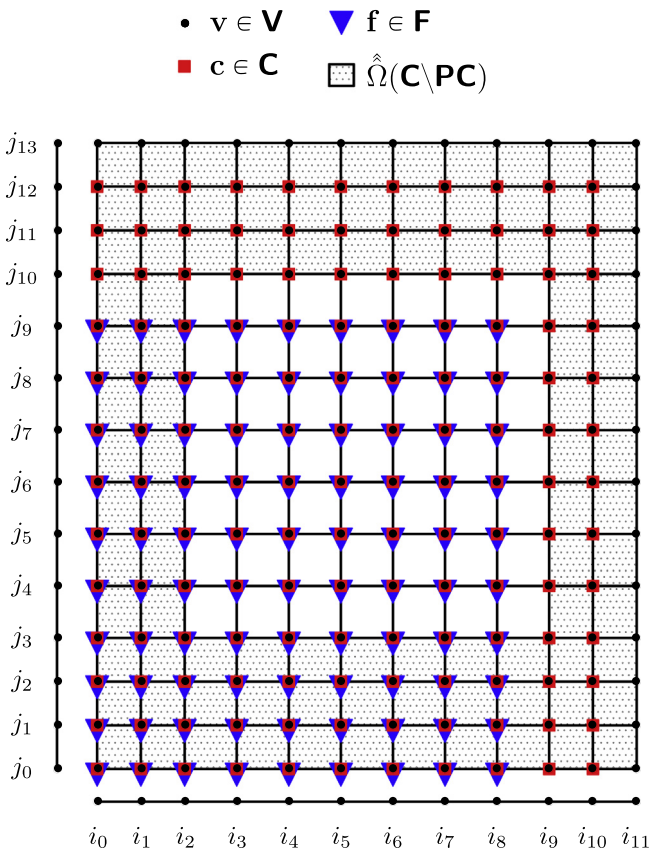


Fig. 4. A quadratic by cubic (i.e., $\mathbf{p} = \{2, 3\}$) tensor product topology.

$$R_{\mathbf{f},\mathbf{p}}(\mathbf{s}) = \frac{N_{\mathbf{f},\mathbf{p}}(\mathbf{s})}{\sum_{\mathbf{f}\in\mathbf{F}} w_{\mathbf{f}} N_{\mathbf{f},\mathbf{p}}(\mathbf{s})} \tag{9}$$

$$= \frac{N_{\mathbf{f},\mathbf{p}}(\mathbf{s})}{w(\mathbf{s})} \tag{10}$$

where $w(\mathbf{s}) : \hat{\Omega} \to \mathbb{R}$ is called a weight function. If $w_{\mathbf{f}} = 1$ for every $\mathbf{f} \in \mathbf{F}$ then $w(\mathbf{s}) = 1$ and $R_{\mathbf{f},\mathbf{p}}(\mathbf{s}) = N_{\mathbf{f},\mathbf{p}}(\mathbf{s})$. Fig. 6 shows the B-spline basis functions and cells in the parametric domain corresponding to those illustrated in the tensor product topology in Fig. 5.

3.5. The NURBS geometric map

Given vector valued control points $\mathbf{P}_{\mathbf{f}} \in \mathbb{R}^{d_s}$ a non-uniform rational B-spline (NURBS) geometric map $\mathbf{x} : \hat{\Omega} \to \mathbb{R}^{d_s}$ can be written as

$$\mathbf{x}(\mathbf{s}) = \sum_{\mathbf{f}\in\mathbf{F}} \mathbf{P}_{\mathbf{f}} w_{\mathbf{f}} R_{\mathbf{f},\mathbf{p}}(\mathbf{s}). \tag{11}$$

3.6. Active sets and domains

Using the tensor product topology described in Section 3.3 we can introduce the notion of active sets and domains. In the hierarchical setting this will allow us to stack tensor product topologies on top of each other where each level may have different active sets and domains. The active sets and domains on a particular level are used to determine the set of active B-spline basis functions on that level.

We denote the set of active vertices by $\mathbf{AV} \subseteq \mathbf{V}$. A cell $\mathbf{c}$ is active if $\mathbf{V}(\mathbf{c}) \subseteq \mathbf{AV}$. We denote the set of all active cells by $\mathbf{AC}$ and the set of active parametric cells by $\mathbf{APC}$. We denote the active cell index domain by $\hat{\hat{\Omega}}(\mathbf{AC}) = \bigcup_{\mathbf{c}\in\mathbf{AC}} \hat{\hat{\Omega}}(\mathbf{c})$ and the active cell parametric domain by $\hat{\Omega}(\mathbf{AC}) = \bigcup_{\mathbf{c}\in\mathbf{AC}} \hat{\Omega}(\mathbf{c})$. An anchor $\mathbf{f}$ is active if $\mathbf{C}(\mathbf{f}) \subset \mathbf{AC}$ or, equivalently, $\mathbf{V}(\mathbf{f}) \subset \mathbf{AV}$. We denote the set of active anchors by $\mathbf{AF}$ and the set of active B-spline basis functions by $\mathbf{AN} = \{N_{\mathbf{f},\mathbf{p}}\}_{\mathbf{f}\in\mathbf{AF}}$. We denote the active function

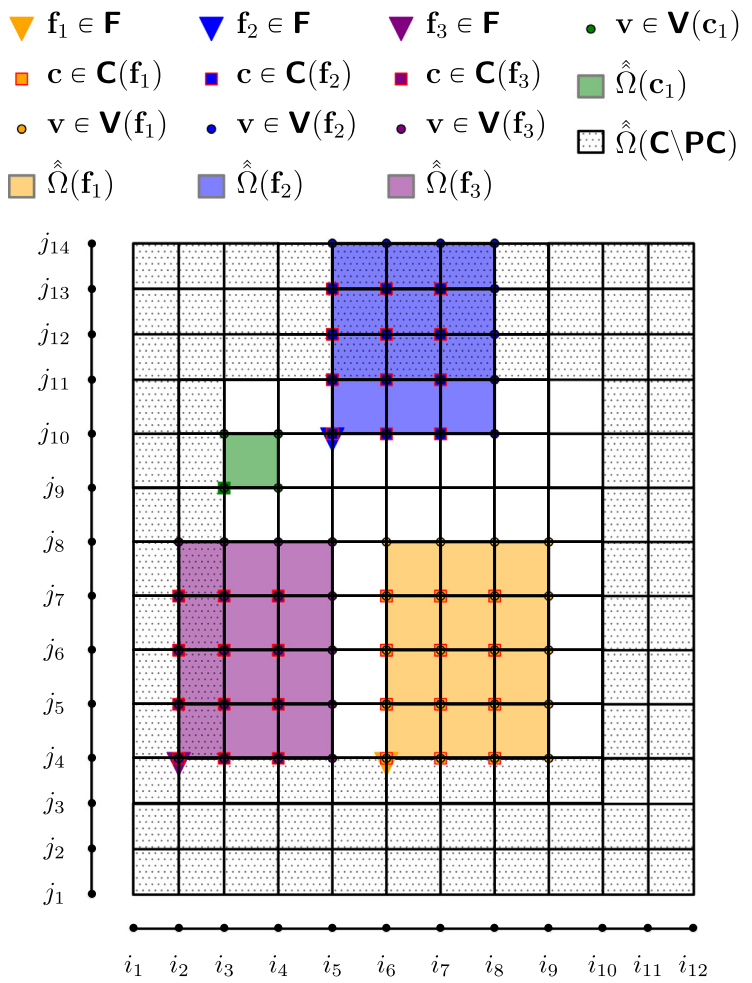


Fig. 5. Illustrated function and cell quantities for a quadratic by cubic (i.e., $\mathbf{p} = \{2,3\}$) tensor product topology.

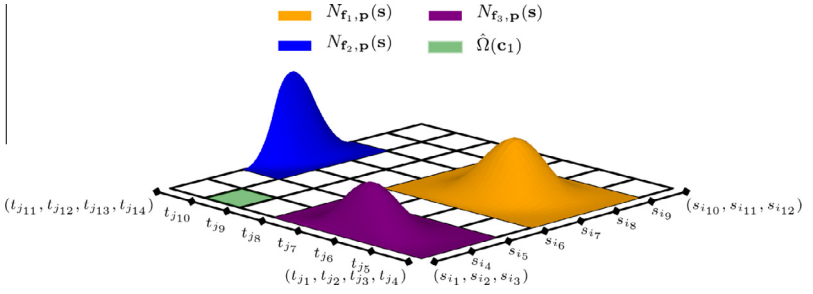


Fig. 6. B-spline basis functions and cells in the parametric domain $\hat{\Omega}$ corresponding to those illustrated in Fig. 5. In this case, $\mathbf{p} = \{2,3\}$.

index domain by $\hat{\hat{\Omega}}(\mathbf{AF}) = \bigcup_{\mathbf{f}\in\mathbf{AF}} \hat{\hat{\Omega}}(\mathbf{f})$ and the active function parametric domain by $\hat{\Omega}(\mathbf{AF}) = \bigcup_{\mathbf{f}\in\mathbf{AF}} \hat{\Omega}(\mathbf{f})$. Note that it may be the case that $\hat{\hat{\Omega}}(\mathbf{AF}) \neq \hat{\hat{\Omega}}(\mathbf{AC})$. Active sets and domains are illustrated in Fig. 7 for a quadratic by cubic (i.e., $\mathbf{p} = \{2,3\}$) tensor product topology.

4. Hierarchical spline trees

A hierarchical B-spline space is constructed from a sequence of N nested B-spline spaces $\mathcal{B}^{\alpha} \subset \mathcal{B}^{\alpha+1}, \alpha = 1, \ldots, N-1$ and properly formulated active sets and domains on each level. The mathematical properties of these spaces have been studied

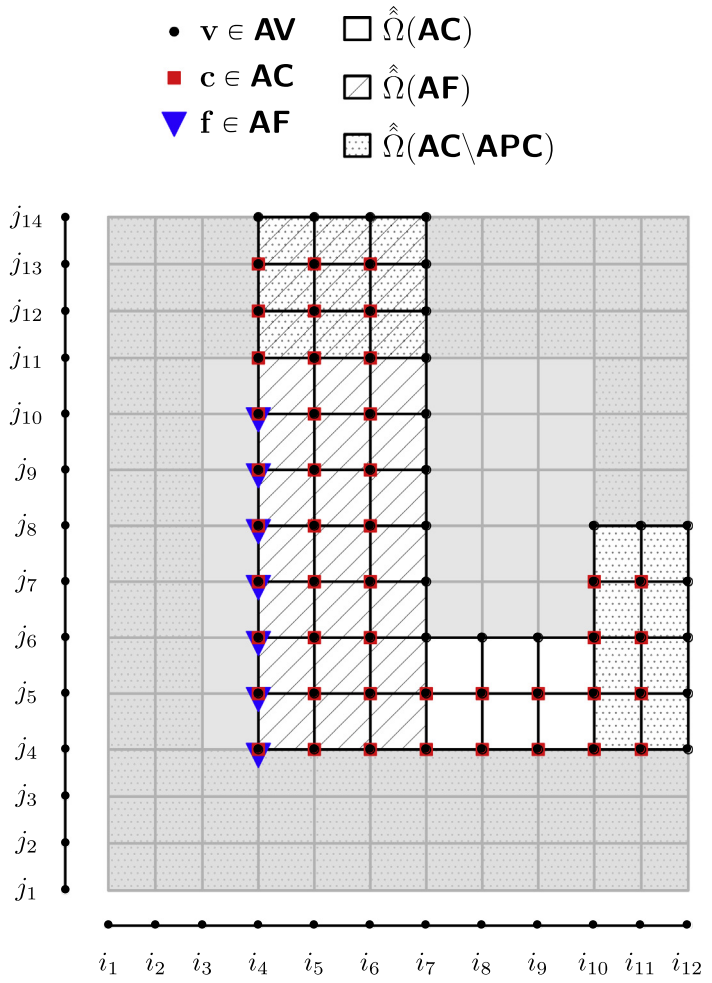


Fig. 7. Active sets and domains for a quadratic by cubic ($\mathbf{p} = \{2,3\}$) tensor product topology. All squares with a white background denote active cells, regardless of the pattern. The hatched squares denote cells which support active functions. The dotted squares denote cells with zero parametric area.

extensively in recent years [2,3,62], stimulated, in large part, to the widespread interest in isogeometric analysis. Note that throughout this paper we use the term hierarchical spline and spline tree interchangeably.

4.1. The hierarchical topology

Each B-spline space $\mathcal{B}^{\alpha+1} \supset \mathcal{B}^{\alpha}$ in the hierarchy is defined by:

- The polynomial degree $\mathbf{p}^{\alpha+1} = \{p_\ell^{\alpha+1}\}_{\ell=1}^{d_p}$ where $p_\ell^{\alpha+1} \geqslant p_\ell^{\alpha}, \ell = 1,\ldots,d_p$, $\alpha = 1,\ldots,N-1$.
- Corresponding global index and knot vectors $\mathbf{I}^{\alpha+1}$ and $\mathbf{G}^{\alpha+1}$ where $\mathsf{I}_\ell^{\alpha} \subset \mathsf{I}_\ell^{\alpha+1}$ and $\mathsf{G}_\ell^{\alpha} \subset \mathsf{G}_\ell^{\alpha+1}$, and for any $\mathbf{s} \in \hat{\Omega}$, $\mu(\mathsf{G}_\ell^{\alpha+1}, s_\ell) - \mu(\mathsf{G}_\ell^{\alpha}, s_\ell) \geqslant p_\ell^{\alpha+1} - p_\ell^{\alpha}, \ell = 1,\ldots,d_p$, $\alpha = 1,\ldots,N-1$.
- Sets of active vertices $\mathbf{AV}^{\alpha+1}$ such that $\hat{\hat{\Omega}}(\mathbf{AC}^1) = \hat{\Omega}^1$ and $\hat{\hat{\Omega}}(\mathbf{AC}^{\alpha}) \supseteq \hat{\hat{\Omega}}(\mathbf{AC}^{\alpha+1})$, $\alpha = 1,\ldots,N-1$. We require that for every $\mathbf{c}^{\alpha} \in \mathbf{AC}^{\alpha}$, $\hat{\hat{\Omega}}(\mathbf{c}^{\alpha}) \bigcap \hat{\hat{\Omega}}(\mathbf{AC}^{\alpha+1})$ is either empty or equal to $\hat{\hat{\Omega}}(\mathbf{c}^{\alpha})$. This is called a strong boundary condition in [2].

We denote a hierarchical topology by

$$\mathbb{H} = \mathbb{H}(\{\mathbf{p}^{\alpha}\}, \{\mathbf{I}^{\alpha}\}, \{\mathbf{G}^{\alpha}\}, \{\mathbf{AV}^{\alpha}\}).$$

Fig. 8 shows an example three level hierarchical topology of degree 2 in index space and Fig. 9 shows the same hierarchical topology in parametric space.

4.2. Hierarchical B-spline spaces

Given a hierarchical topology $\mathbb{H}$, the set of hierarchical anchors is

$$\mathbf{HF} = \left\{ \mathbf{f}^{\alpha} \in \mathbf{AF}^{\alpha} : \hat{\hat{\Omega}}(\mathbf{f}^{\alpha}) \not\subseteq \bigcup_{\beta=\alpha+1}^{N} \hat{\hat{\Omega}}(\mathbf{AF}^{\beta}), \alpha = 1, \ldots, N \right\}. \tag{12}$$

A hierarchical B-spline spline space $\mathcal{H}$ is spanned by the set of hierarchical B-spline blending functions, denoted by **HN**, which are associated with the anchors in **HF**. Note that the blending functions in **HN** are linearly independent [2] and thus form a basis for the space. The basis functions corresponding to the hierarchical topology in Figs. 8 and 9 are shown in Fig. 10.

5. The geometry of a hierarchical representation

An important difference between hierarchical splines and NURBS or T-splines is in the way geometry is represented. For single-level representations like NURBS or T-splines each basis function is associated with a control point and all control points influence the shape of the geometry in the same way. For hierarchical splines, the set of hierarchical basis functions **HN** does not constitute the full set of B-spline blending functions which can be defined over each level of the hierarchy. Those which are linearly dependent on higher levels of the hierarchy are not included in the set. The set of blending functions used to define geometric quantities, denoted by **GN**, may or may not consist of functions in **HN**. The two most common scenarios are:

1. $\mathbf{GN} \subseteq \mathbf{HN}$.
2. $\mathbf{GN} \not\subseteq \mathbf{HN}$ but for any $N^{\alpha}_{\mathbf{f},\mathbf{p}} \in \mathbf{GN}$ we have that $N^{\alpha}_{\mathbf{f},\mathbf{p}}(\mathbf{s}) = \sum_{\mathbf{g}^{\beta} \in \mathbf{HF}} c^{\mathbf{f}}_{\mathbf{g}} N^{\beta}_{\mathbf{g},\mathbf{p}}(\mathbf{s})$ where $c^{\mathbf{f}}_{\mathbf{g}} \in \mathbb{R}$, $N^{\beta}_{\mathbf{g},\mathbf{p}} \in \mathbf{HN}$, and $\beta \geqslant \alpha$.

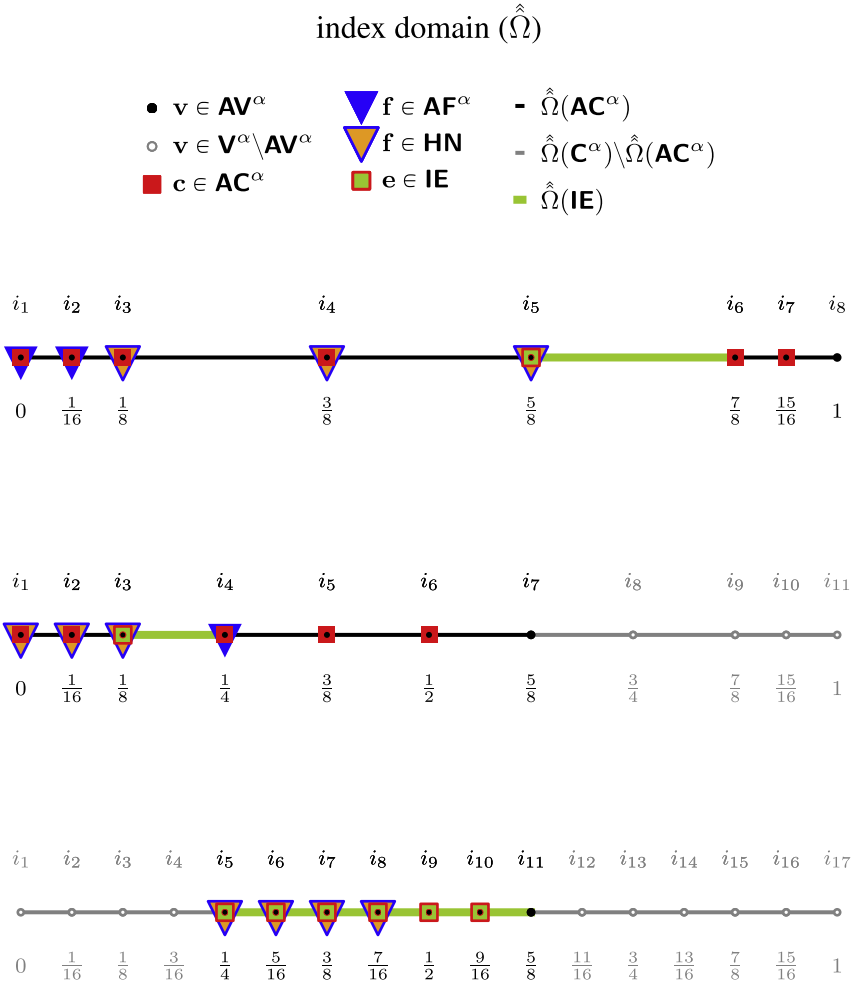


Fig. 8. A three level hierarchical topology in the global index domain.

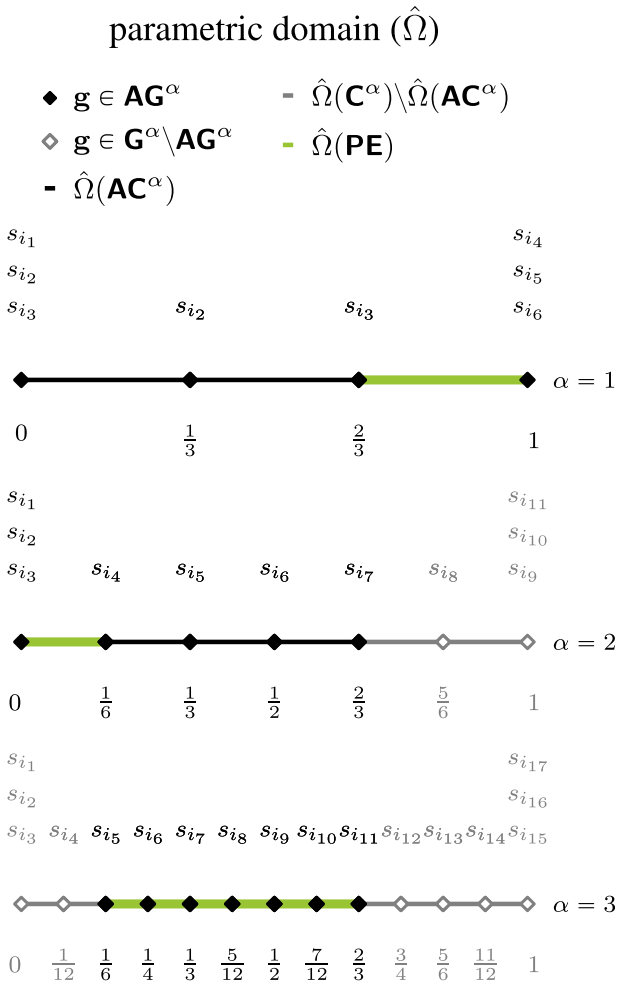


Fig. 9. A three level hierarchical topology in the global parametric domain.

The most common case is when $\mathbf{GN} = \mathbf{AN}^1$. This is an attractive choice since the number of basis functions in $\mathbf{AN}^1$ is usually far fewer than those in **HN**. In either case, the geometry representation can be unified using Bézier extraction as described in Section 8. We denote the set of anchors associated with blending functions in **GN** by **GF**.

5.1. Hierarchical NURBS basis functions

Given a hierarchical B-spline basis function $N^\alpha_{\mathbf{f},\mathbf{p}} \in \mathbf{HN}$ and weight $w^\beta_{\mathbf{g}} \in \mathbb{R}^+$ for each $\mathbf{g}^\beta \in \mathbf{GF}$ a *rational* hierarchical B-spline basis function can be written as

$$R^\alpha_{\mathbf{f},\mathbf{p}}(\mathbf{s}) = \frac{N^\alpha_{\mathbf{f},\mathbf{p}}(\mathbf{s})}{\sum_{\mathbf{g}^\beta \in \mathbf{GF}} w^\beta_{\mathbf{g}} N^\beta_{\mathbf{g},\mathbf{p}}(\mathbf{s})} \tag{13}$$

$$= \frac{N^\alpha_{\mathbf{f},\mathbf{p}}(\mathbf{s})}{w(\mathbf{s})}. \tag{14}$$

Note that the weight function $w(\mathbf{s})$ is defined using the blending functions in **GN**.

5.2. The hierarchical NURBS geometric map

The hierarchical NURBS geometric map is simply a NURBS geometric map constructed from the blending functions in **GN**. In other words, to each $\mathbf{g}^\alpha \in \mathbf{GF}$ we associate a vector valued control point $\mathbf{P}^\alpha_{\mathbf{g}} \in \mathbb{R}^{d_s}$ and write the geometric map as

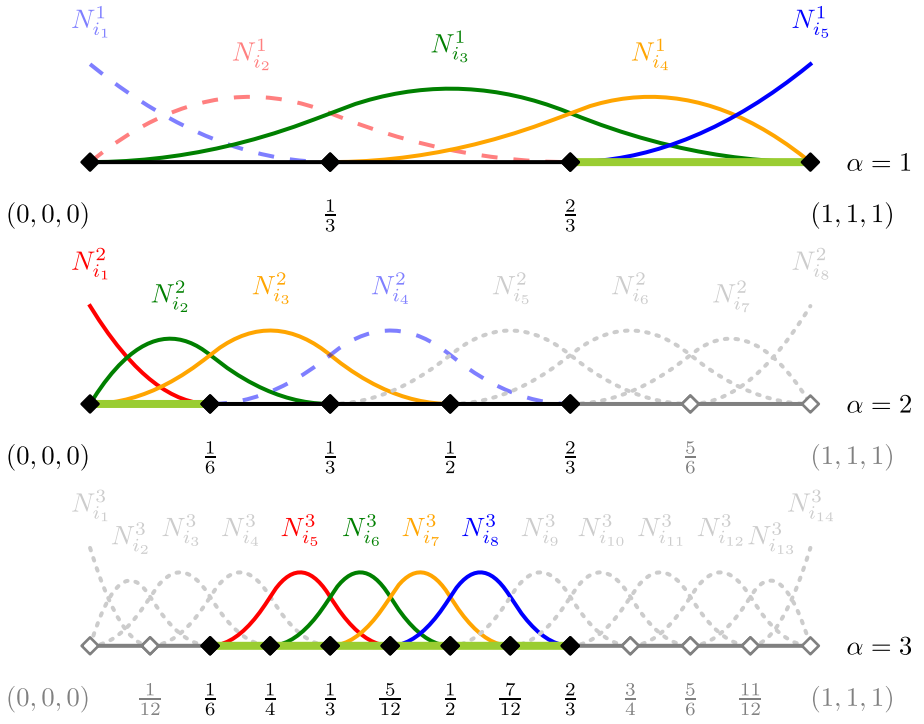


Fig. 10. Basis functions for the hierarchical spline space inferred from the hierarchical topology in Fig. 8, 9. Functions in **HN** are indicated by solid colored lines. Functions in **AF** \ **HN** are indicated by dashed colored lines. Inactive functions are indicated by grey dotted lines. Active cells are black and leaf cells are green. Note that functions in **HN** are supported entirely by leaf cells (denoted by green lines). See Section 6.1 for a description of leaf cells.

$$\mathbf{x}(\mathbf{s}) = \sum_{\mathbf{g}^\alpha \in \mathbf{GF}} \mathbf{P}^\alpha_{\mathbf{g}} w^\alpha_{\mathbf{g}} R^\alpha_{\mathbf{g},\mathbf{p}}(\mathbf{s}). \tag{15}$$

Alternatively, the geometric map $\mathbf{x}(\mathbf{s})$ can be written in matrix–vector notation by defining a vector of geometric basis functions $\mathbf{GN}(\mathbf{s})$ and a vector of control points $\mathbf{P}$ such that

$$\mathbf{x}(\mathbf{s}) = \mathbf{P}^{\mathrm{T}} \cdot \mathbf{GN}(\mathbf{s}). \tag{16}$$

If $\mathbf{GN} \not\subseteq \mathbf{HN}$, but any function in **GN** can be written as a linear combination of functions in **HN**, then the two sets of functions can be related by

$$\mathbf{GN}(\mathbf{s}) = \mathbf{M} \cdot \mathbf{HN}(\mathbf{s}), \tag{17}$$

where $\mathbf{HN}(\mathbf{s})$ is a vector composed of functions in **HN** and **M** is a real matrix called the refinement matrix or operator [50]. The entries of **M** are usually computed through knot insertion. The geometric map can then be written in terms of the functions in **HN** by substituting Eq. 17 in Eq. 16 to obtain

$$\mathbf{x}(\mathbf{s}) = \mathbf{P}^{\mathrm{T}} \cdot \mathbf{M} \cdot \mathbf{HN}(\mathbf{s}). \tag{18}$$

The vector of control points associated with the functions in **HN** is

$$\mathbf{Q} = \mathbf{M}^{\mathrm{T}}\mathbf{P} \tag{19}$$

and thus

$$\mathbf{x}(\mathbf{s}) = \mathbf{Q}^{\mathrm{T}} \cdot \mathbf{HN}(\mathbf{s}). \tag{20}$$

As an example, consider the hierarchical basis shown in Fig. 10. If the geometry is expressed in terms of the functions in $\mathbf{AN}^1$ then $\mathbf{GN} = \mathbf{AN}^1$. The vector of geometric basis functions is defined as

$$\mathbf{GN}(\mathbf{s}) = \begin{bmatrix} N^1_1(\mathbf{s}) & N^1_2(\mathbf{s}) & N^1_3(\mathbf{s}) & N^1_4(\mathbf{s}) & N^1_5(\mathbf{s}) \end{bmatrix}^{\mathrm{T}},$$

and the associated control points are

$$\mathbf{P}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{P}_2 = \begin{bmatrix} 3/2 \\ -1 \end{bmatrix}, \quad \mathbf{P}_3 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{P}_4 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \quad \mathbf{P}_5 = \begin{bmatrix} 4 \\ 1 \end{bmatrix}.$$

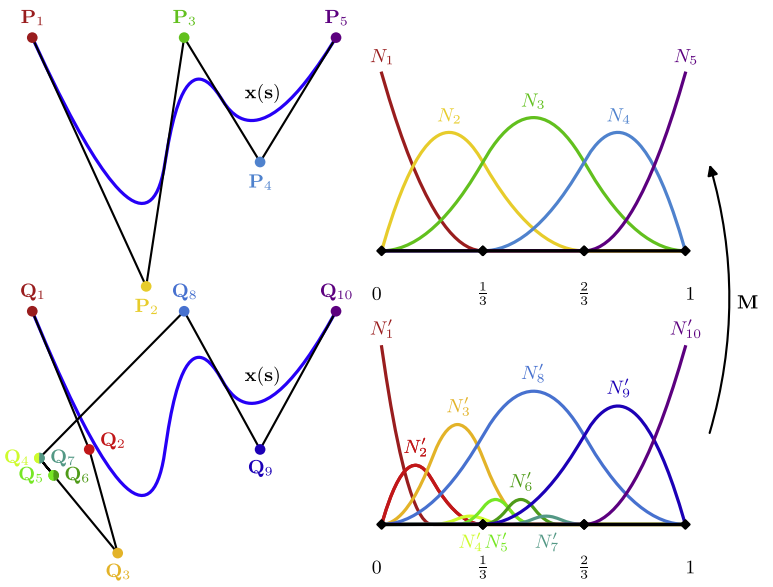


Fig. 11. The geometric map $\mathbf{x}(\mathbf{s})$ defined in terms of functions in **GN** (top) and **HN** (bottom) using the basis shown in Fig.10.

The basis functions and the geometric map defined by these control points are shown in Fig. 11, at the top. If the vector of functions $\mathbf{GN}(\mathbf{s})$ is instead taken to be all the functions in **HN**,

$$\mathbf{GN} = \begin{bmatrix} N_1^2 & N_2^2 & N_3^2 & N_5^3 & N_6^3 & N_7^3 & N_8^3 & N_3^1 & N_4^1 & N_5^1 \end{bmatrix}^{\mathrm{T}},$$

then we have that

$$\mathbf{M} = \begin{bmatrix} 1 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 3/4 & 1/16 & 3/16 & 3/16 & 1/16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{M}$ is obtained using standard knot insertion algorithms on B-spline basis functions. The control points associated with the functions in $\mathbf{GN}(\mathbf{s})$ are found from Eq. 19:

$$\mathbf{Q}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{Q}_2 = \begin{bmatrix} 3/4 \\ 0 \end{bmatrix}, \quad \mathbf{Q}_3 = \begin{bmatrix} 9/8 \\ -3/4 \end{bmatrix},$$
$$\mathbf{Q}_4 = \begin{bmatrix} 3/32 \\ 1/16 \end{bmatrix}, \quad \mathbf{Q}_5 = \begin{bmatrix} 9/32 \\ -3/16 \end{bmatrix}, \quad \mathbf{Q}_6 = \begin{bmatrix} 9/32 \\ -3/16 \end{bmatrix},$$
$$\mathbf{Q}_7 = \begin{bmatrix} 3/32 \\ 1/16 \end{bmatrix}, \quad \mathbf{Q}_8 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{Q}_9 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \quad \mathbf{Q}_{10} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}.$$

The control points $\mathbf{Q}_i$ and the resulting geometric map are shown in Fig. 11 on the bottom. Also shown are the hierarchical basis functions N_i' which are functions from $\mathbf{GN}(\mathbf{s})$ scaled by the corresponding nonzero entries in $\mathbf{M}$.

Note that in general the basis functions in **HN** do not form a partition of unity and consequently the convex hull property common to B-splines is lost. However, this does not mean that constants cannot be represented by a hierarchical basis. In fact, due to the nestedness of the N B-spline spaces $\mathcal{B}^\alpha \subset \mathcal{B}^{\alpha+1}$, $\alpha = 1, \ldots, N-1$ used to construct the hierarchical space, there exists a set of coefficients such that $1 = \sum_{\mathbf{f}^\alpha \in \mathbf{HF}} c_{\mathbf{f}}^\alpha N_{\mathbf{f},\mathbf{p}}^\alpha$. In general, if a function is contained in $\mathcal{B}^1$, it is also contained in the hierarchical spline space. This is an extremely important property for analysis. Specifically, all patch tests are satisfied *a priori* by hierarchical spline discretizations. We note that change of basis techniques can be used to recover the partition of unity property for hierarchical spline spaces [3] and thus improve the conditioning of the basis.

6. Hierarchical adaptivity

The multi-resolution characteristics of hierarchical spline representations can be leveraged to develop simple and efficient local refinement and coarsening algorithms. These algorithms only modify the activity state of the different levels of

the hierarchy. Linear independence can be assured and the conditioning of the resulting basis can be controlled. Additionally, since nested spaces are created during refinement and coarsening, exact geometry is preserved. In the context of isogeometric analysis, refinement and coarsening can be coupled to create adaptive algorithms capable of tracking and resolving moving interfaces. Several demanding analysis benchmarks which use these algorithms are shown in Section 9.5. Additionally, as described in Section 7, these algorithms can be applied directly to unstructured forests of spline trees for both surfaces and volumes.

6.1. Leaf elements

Leaf elements become a primary object of interest when devising adaptive analysis routines using spline trees. Additionally, as shown in Section 8, Bézier extraction can be performed on the leaf elements of a forest of spline trees collapsing the complex, unstructured hierarchical structure of the forest into a canonical finite element representation which can be easily integrated into existing finite element frameworks. The leaf elements in the index domain of a spline tree are $\mathbf{IE} = \{\mathbf{e}^\alpha \in \mathbf{AC}^\alpha : \hat{\hat{\Omega}}(\mathbf{e}^\alpha) \cap \bigcup_{\beta=\alpha+1}^{N} \hat{\hat{\Omega}}(\mathbf{AC}^\beta) = \emptyset, \alpha = 1, \ldots, N\}$. The leaf elements in the parametric domain of a hierarchical B-spline are $\mathbf{PE} = \{\mathbf{e}^\alpha \in \mathbf{APC}^\alpha : \hat{\Omega}(\mathbf{e}^\alpha) \cap \bigcup_{\beta=\alpha+1}^{N} \hat{\Omega}(\mathbf{APC}^\beta) = \emptyset, \alpha = 1, \ldots, N\}$. The set of hierarchical basis functions supported by $\mathbf{e}^\alpha \in \mathbf{PE}$ are $\mathbf{HN}(\mathbf{e}^\alpha) = \left\{N^\beta_{\mathbf{f},\mathbf{p}} \in \mathbf{HN} : \hat{\Omega}(\mathbf{e}^\alpha) \subseteq \hat{\Omega}(\mathbf{f}^\beta), \beta \leqslant \alpha\right\}$. The set of anchors associated with $\mathbf{HN}(\mathbf{e}^\alpha)$ is denoted by $\mathbf{HF}(\mathbf{e}^\alpha)$. The set of geometric basis functions supported by $\mathbf{e}^\alpha \in \mathbf{PE}$ are $\mathbf{GN}(\mathbf{e}^\alpha) = \left\{N^\beta_{\mathbf{g},\mathbf{p}} \in \mathbf{GN} : \hat{\Omega}(\mathbf{e}^\alpha) \subseteq \hat{\Omega}(\mathbf{g}^\beta)\right\}$. The set of anchors associated with $\mathbf{GN}(\mathbf{e}^\alpha)$ are denoted by $\mathbf{GF}(\mathbf{e}^\alpha)$. The leaf elements (denoted by green lines) and associated quantities in both the index and parametric domains for a three level spline tree are shown in Figs. 8–10.

6.2. Local refinement

A hierarchical local refinement algorithm can be constructed by appropriately modifying the set of active vertices on different levels of the hierarchy. Suppose an N level hierarchical topology $\mathbb{H}^1$ is given where $\mathbf{AV}^\alpha = \emptyset$ for $\alpha = M+1, \ldots, N$ for some $M \geqslant 1$. A refined topology $\mathbb{H}^2$ can be created by selecting a set of leaf elements $\mathbf{X} \subseteq \mathbf{IE}$ and requiring that for each $\mathbf{e}^\alpha \in \mathbf{X}$ that $\mathbf{V}^{\alpha+1}(\mathbf{e}^\alpha) \subseteq \mathbf{AV}^{\alpha+1}$. In other words, if any vertex in $\mathbf{V}^{\alpha+1}(\mathbf{e}^\alpha)$ is not active it is activated. If a sufficient number of active vertices are added to the hierarchical topology then additional hierarchical basis functions will be activated. As a result we have that $\mathcal{H}^1 \subseteq \mathcal{H}^2$.

In Algorithm 1 a local refinement algorithm is presented. A set of leaf cells is specified along with a global mesh balance parameter n and a boolean specifying whether empty leaf cells should be pruned from the tree. The balance parameter n is used to control the extent of basis function interaction at different levels of the hierarchy. Ideally, we want to that ensure that none of the leaf cells which support a function from level α are on level β where $\beta \gg \alpha$. Otherwise, when the basis is used in the context of isogeometric analysis, poor conditioning of the system matrices will result.

An empty leaf cell is a leaf cell on level α which does not support any hierarchical basis functions from level α. We note that this global pruning operation is optional and a valid Bézier extraction of the hierarchical mesh can be constructed in either case as described in Section 8. For high performance applications it may be desirable to leave orphaned leaf cells in the hierarchical mesh to avoid the additional performance hit associated with global pruning. In fact, in almost all cases, the computational cost of assembling the additional empty leaf elements in an isogeometric simulation will be negligible. We also note that empty leaf elements have no affect on the accuracy of a simulation as the finite element basis is identical with or without them. It is similar to performing Bézier extraction of a B-spline curve and then performing a de Casteljau subdivision of one of the elements. The number of basis functions does not change but there are more elements than are strictly necessary to represent each C^∞ region of the basis.

If $\mathbf{e}^\alpha \in \mathbf{PE}$ then the cells on the next level $\alpha+1$ which are contained in $\hat{\hat{\Omega}}(\mathbf{e}^\alpha)$ are activated using Algorithm 2. Finally, a global iterative procedure is used to balance (see Algorithm 4) and, if requested, prune (see Algorithm 7) the modified hierarchical topology. The iterations continue until the mesh reaches a steady state. In practice this is usually no more than one or two iterations.

Fig. 12 shows the results of four successive refinements along the diagonal of a unit square biquadratic hierarchical mesh. The control points are specified so that a linear parameterization of the geometry results. All parametric leaf cells contained in the same level of the hierarchy have the same color. In each case, the cells whose geometric center $\mathbf{x}^\mathbf{c}$ satisfies $x_1^\mathbf{c} + 0.05 > x_2^\mathbf{c} > x_1^\mathbf{c} - 0.05$ are refined using Algorithm 1. Fig. 12 shows the results of local refinement for the mesh balance parameter $n = 0$ (on the left) and $n = 1$ (on the right). Notice that as n increases the effect of refinement becomes less local. Fig. 13 shows the results of the fourth diagonal refinement for $n = 0$ and $n = 2$. The Greville abscissae of the hierarchical basis functions on the first level are shown in each case (black squares). For $n = 0$ the basis functions on level one are supported by cells on every level. However, when $n = 2$ the basis functions on level one are only supported by cells up through level three.

Algorithm 1. A spline forest (or tree) local refinement algorithm with global balancing and pruning

```
1: procedure refineForest(leaves, n, prune)
2:   for i ← 1, size (leaves) do
3:     c^α ← leaves(i)
4:     activateChildrenForest(c^α)        ▷ see Algorithm 2
5:   end for
6:   while true do
7:     n_bal ← balanceForest(n)           ▷ see Algorithm 3
8:     if prune then
9:       n_pr ← pruneForest()             ▷ see Algorithm 7
10:    end if
11:    if n_bal = 0 & n_pr = 0 then
12:      break
13:    end if
14:  end while
15: end procedure
```

Algorithm 2. Activate the vertices of the children of a cell $\mathbf{c}^{\alpha}$

```
1: procedure activateChildrenForest(c^α)
2:   children ← getChildCells(c^α)
3:   for i ← 1, size(children) do
4:     c^{α+1} ← children(i)
5:     verts ← getVertices(c^{α+1})
6:     for j ← 1, size(verts) do
7:       v^{α+1} ← verts(j)
8:       activate(v^{α+1})
9:     end for
10:  end for
11: end procedure
```

Algorithm 3. Balance a forest (or tree) given the balance parameter n

```
1: procedure balanceForest(n)
2:   while true do
3:     leaves ← leafElements()
4:     changed = false
5:     for i ← 1,size(leaves) do
6:       e^α ← leaves(i)
7:       balanceForest(e^α, n, changed)   ▷ see Algorithm 4
8:     end for
9:     if ¬changed then
10:      break
11:    end if
12:  end while
13: end procedure
```

Algorithm 4. Balance the n-ring neighborhood around $\mathbf{c}^{\alpha}$.

```
1: procedure balanceForest(c^α, n, changed)
2:   if α > 1 then
3:     c^{α−1} ← getParentCell(c^α)
4:     ring ← getRingCells(c^{α−1}, n)
5:     for i ← 1, size(ring) do
6:       c^{α−1} ← ring(i)
7:       if isInactive(c^{α−1}) then
8:         balanceForest (c^{α−1}, n, changed)
9:         changed = true
10:          c^{α−2} ← getParentCell(c^{α−1})
11:          activateChildrenForest(c^{α−2})      ▷ see Algorithm 2
12:        end if
13:      end for
14:    end if
15: end procedure
```

6.3. Local coarsening

Local coarsening is similar to local refinement except active vertices on different levels of the hierarchy are deactivated. Suppose an N level hierarchical topology $\mathbb{H}^1$ is given with a corresponding set of leaf elements $\mathbf{IE}^1$. A coarsened hierarchical topology $\mathbb{H}^2$ can be created by first selecting a set of leaf elements $\mathbf{X} \subseteq \mathbf{I}E^1$ such that $\alpha > 1$ for every $\mathbf{e}^{\alpha} \in \mathbf{X}$. For every $\mathbf{e}^{\alpha} \in \mathbf{X}$ find the unique cell $\mathbf{c}^{\alpha-1} \in \mathbf{AC}^{\alpha-1}$ such that $\hat{\Omega}(\mathbf{e}^{\alpha}) \subseteq \hat{\Omega}(\mathbf{c}^{\alpha-1})$. We denote this set of cells by $\mathbf{Y}$. Now form a set $\mathbf{T}$ consisting of all leaf elements $\mathbf{e}^{\beta} \in \mathbf{IE}^1$ such that $\hat{\Omega}(\mathbf{e}^{\beta}) \subseteq \hat{\Omega}(\mathbf{c}^{\alpha-1})$ for some $\mathbf{c}^{\alpha-1} \in \mathbf{Y}$. Now deactivate the vertices in $\bigcup_{\beta=\alpha}^{N} \mathbf{AV}^{\beta}$ such that the coarsened set of leaf elements becomes $\mathbf{IE}^2 = (\mathbf{IE}^1 \setminus \mathbf{T}) \cup \mathbf{Y}$. As a result we have that $\mathcal{H}^2 \subseteq \mathcal{H}^1$.

In Algorithm 5 a local coarsening algorithm is presented. A set of leaf cells is specified along with a global mesh balance parameter n and a boolean specifying whether empty leaf cells should be pruned from the tree. As described for local refinement (see Algorithm 1), an iterative procedure is employed to both balance and prune the tree until a steady state is reached.

Algorithm 5. A spline forest (or tree) local coarsening algorithm with global balancing and pruning

```
1: procedure coarsenForest(leaves, n, prune)
2:   for i ← 1,size(leaves) do
3:     c^α ← leaves(i)
4:     if α > 1 then
5:       c^{α−1} ← getParentCell(c^α)              ▷ ensure strong boundary condition
6:       pruneVerticesForest(c^{α−1}, deactivate)  ▷ see Algorithm 6
7:       for i ← 1, size(deactivate) do            ▷ deactive all pruned vertices
8:         v^β ← deactivate(i)
9:         deactivate(v^β)
10:      end for
11:    end if
12:  end for
13:  while true do
14:    n_bal ← balanceForest(n)                    ▷ see Algorithm 3
15:    if prune then
16:      n_pr ← pruneForest()                      ▷ see Algorithm 7
17:    end if
18:    if n_bal = 0 & n_pr = 0 then
19:      break
20:    end if
21:  end while
22: end procedure
```

Algorithm 6. Returns the descendant vertices of $\mathbf{c}^{\alpha}$ to be deactivated during coarsening

```
1: procedure pruneVerticesForest(c^α, deactivate)
2:   children ← getActiveChildCells(c^α)
3:   for i ← 1, size(children) do
4:     c^{α+1} ← children(i)
5:     pruneVerticesForest(c^{α+1}, deactivate)
6:     ring ← ring ∪getVertices(c^{α+1})
7:   end for
8:   for (i ← 1, sizering) do
9:     v^{α+1} ← ring(i)
10:    touching ← touchingActiveCells(v^{α+1})
11:  switch ← true
12:  for j ← size(touching) do
13:    c^{α+1} ← touching(j)
14:    if contains(children, c^{α+1}) and isActive(c^{α+1}) then
15:      switch ← false
16:      break
17:    end if
18:  end for
19:  if switch then
20:    deactivate ← deactivate ∪ v^{α+1}
21:  end if
22:  end for
23: end procedure
```

Algorithm 7. Prunes any orphaned leaf cells from the forest (or tree).

```
1: procedure pruneForest()
2:   while true do
3:     leaves ← leafElements()        ▷ get all leaf elements in forest
4:     for i ← 1, size(leaves) bf do
5:       e^α ← leaves(i)
6:       funcs ← getActiveFuncs(e^α)
7: if size(funcs) = 0 then
8:         c^{α−1} ← getParentCell(e^α)
9:         pruneVerticesForest(c^{α−1}, deactivate)      ▷ see Algorithm 6
10:      end if
11:    end for
12:    for i ← 1, size(deactivate) do
13:      v^α ← deactivate(i)
14:      deactivate(v^α)
15:    end for
16:    if size(deactivate) = 0 then
17:      break
18:    end if
19:  end while
20: end procedure
```

Fig. 14 shows the results of four successive coarsenings along the diagonal of a unit square biquadratic hierarchical mesh starting with the final meshes on the bottom of Fig. 12. In each case, the cells whose geometric center $\mathbf{x}^{\mathbf{c}}$ satisfies $x_1^{\mathbf{c}} + 0.05 > x_2^{\mathbf{c}} > x_1^{\mathbf{c}} - 0.05$ are coarsened using Algorithm 5 The coarsening results for $n = 0$ are shown on the left and the results for $n = 1$ are shown on the right.

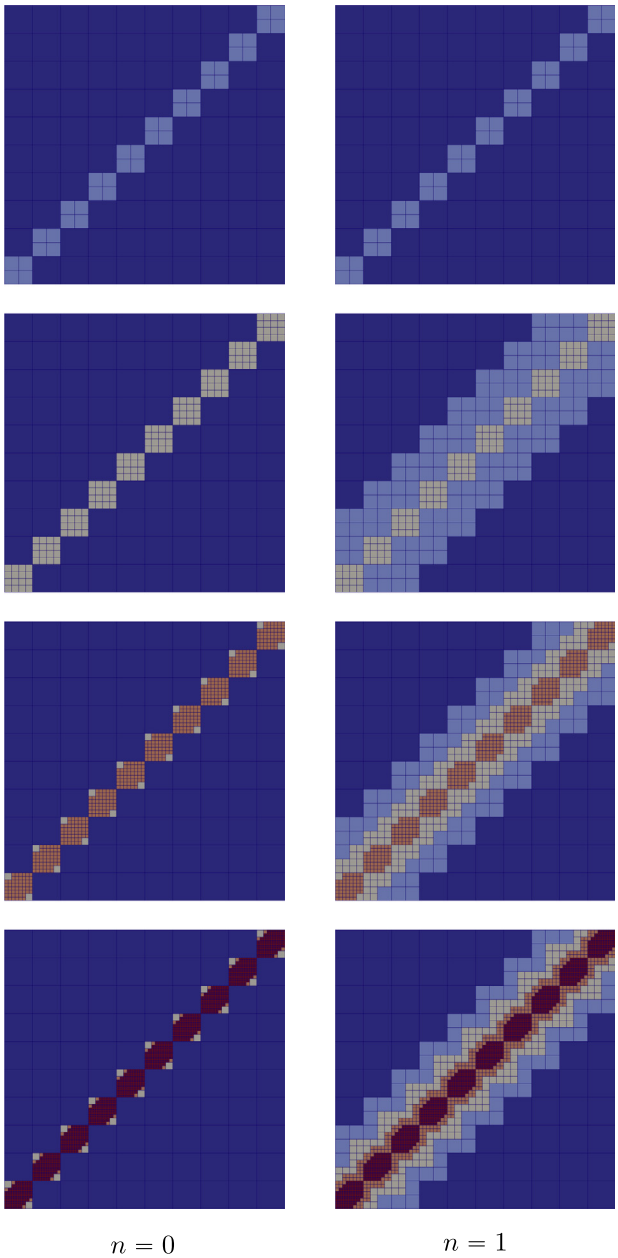
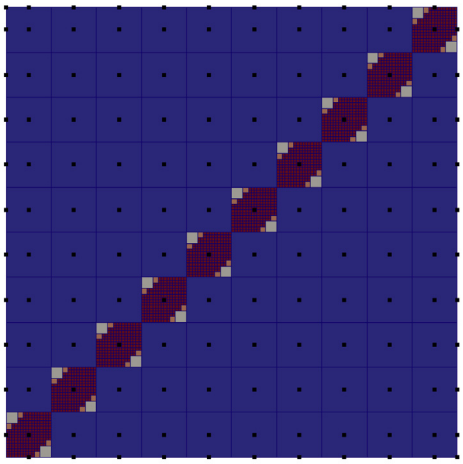


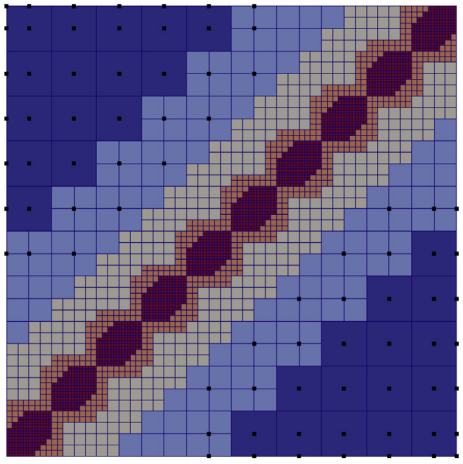
Fig. 12. Four successive local refinements along the diagonal of a unit square biquadratic hierarchical mesh for a balance parameter $n = 0, 1$. All parametric leaf cells contained in the same level of the hierarchy have the same color. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

7. Spline forests

In a spline forest, the geometric domain $\Omega \subset \mathbb{R}^{d_s}$ is the composition of K conforming hierarchical subdomains. In other words, $\Omega = \bigcup_{k=1}^{K} \Omega_k$ where $\Omega_k \subset \mathbb{R}^{d_s}$. Each subdomain is the image of a hierarchical parametric domain $\hat{\Omega}_k$ under a corresponding hierarchical NURBS geometric map $\mathbf{x}_k$. In other words, $\Omega_k = \mathbf{x}_k(\hat{\Omega}_k)$. Fig. 15 shows a two tree forest and the corresponding conforming maps. As a consequence, spline forests can represent domains of arbitrary topological genus; thus spline forests overcome the strict rectangular topology of a single spline tree. The additional complexity in a spline forest comes from maintaining functional compatibility between adjacent trees in a completely unstructured setting for both surfaces and solids.



$n = 0$



$n = 2$

Fig. 13. The resulting biquadratic hierarchical mesh after four successive diagonal refinements for $n = 0$ and $n = 2$. The Greville abscissae of the hierarchical basis functions on the first level are denoted by black squares. All parametric leaf cells contained in the same level of the hierarchy have the same color. For $n = 0$ the basis functions on the level one are supported by cells on every level. However, when $n = 2$ the basis functions on level one are only supported by cells up through level three. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

In this paper, we restrict the smoothness of the forest basis functions across subdomain interfaces to be C^0. Smoother interface basis functions are under development in the context of T-spline forests. We note that in what follows all forest subdomain connectivity operations are computed discretely (integer-based) [47] using a *zero-based* indexing scheme. No floating-point arithmetic is used; this approach avoids topological errors due to roundoff and maintains efficiency for high performance parallel applications.

7.1. Forest topology

To construct the forest basis functions and to perform forest local refinement and coarsening we require an efficient encoding of forest topology. Subdomains can share faces, sides, and corners and can be arbitrarily rotated against each other. Our topology representation scheme closely follows that proposed in [47]. Each subdomain is composed of faces $\mathsf{f} \in \{0, \ldots, 2d_p - 1\}$, sides (3D only) $\mathsf{s} \in \{0, \ldots, 11\}$, and corners $\mathsf{c} \in \{0, \ldots, 2^{d_p} - 1\}$ as shown in Fig. 16. Each face has face corners $\mathsf{fc} \in \{\mathsf{c}_0 < \cdots < \mathsf{c}_{2^{d_p-1}-1}\}$ and each side has side corners $\mathsf{sc} \in \{\mathsf{c}_0 < \mathsf{c}_1\}$. The face and side corners are enumerated in the same order as they appear in the corner numbering of the parent subdomain. In the parametric space of the subdomain we fix the origin at the front lower left corner and use s before t before u (3D only) for faces, sides (3D only), and corners. Note that our indexing scheme for faces, sides, and corners is zero-based. This will allow use to write efficient topological routines

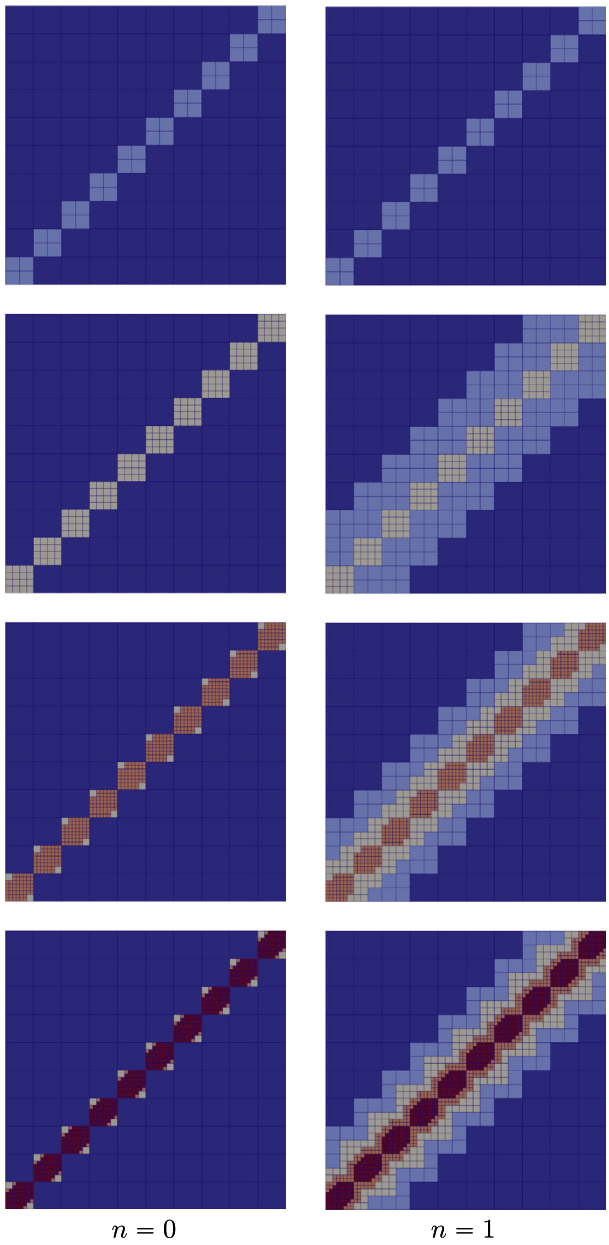


Fig. 14. Four successive local coarsenings along the diagonal of a unit square biquadratic hierarchical mesh for a balance parameter $n = 0, 1$. All parametric leaf cells contained in the same level of the hierarchy have the same color. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

using bitwise operations. Within a single subdomain, simple lookup tables can be constructed to relate faces, sides, and corners to neighboring entities.

7.1.1. Face connectivity

Given a subdomain k and face f we need to encode the neighboring subdomain k' connected through face f, the index of the shared face in the coordinate system of k', denoted f', and the relative orientation r between k and k'. This information is stored in the following maps: $k' = \textsc{domain}(k, \mathsf{f})$, $\mathsf{f}' = \textsc{face}(k, \mathsf{f})$, and $r = \textsc{orient}(k, \mathsf{f})$. If the subdomain face f is on the boundary of the domain -1 is returned in all cases. Given a shared face between two subdomains the orientation r is determined as follows:

Denote the subdomain which assigns a lower face number to the shared face by a and the other as b. Denote the shared face corners of a by $\mathsf{fc}^a = \{\mathsf{c}^a_0, \ldots, \mathsf{c}^a_{2^{d_p-1}-1}\}$ and the shared face corners of b by $\mathsf{fc}^b = \{\mathsf{c}^b_0, \ldots, \mathsf{c}^b_{2^{d_p-1}-1}\}$. The relative orientation r is the index such that c^b_r is c^a_0.

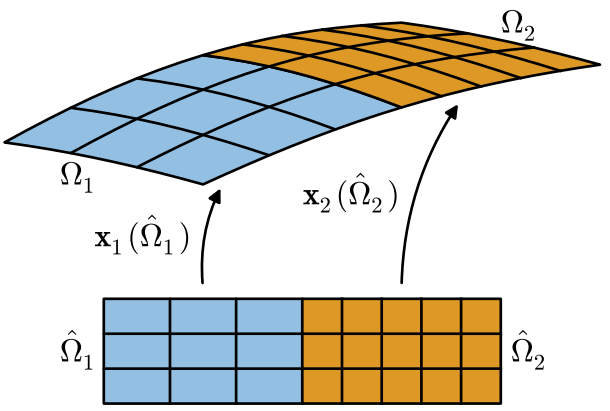


Fig. 15. A two tree forest and the corresponding conforming geometric maps.

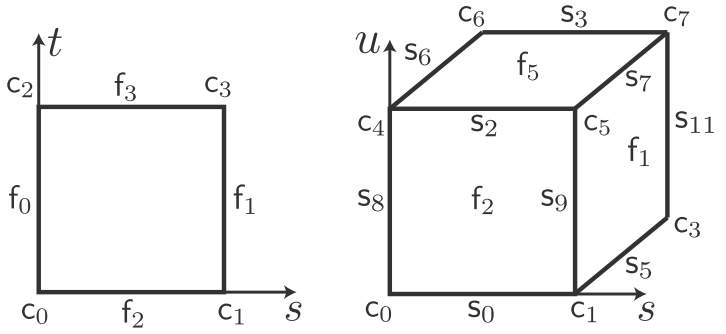


Fig. 16. The enumeration of faces, sides (3D only), and corners composing the topology of a subdomain.

If f and f′ are equal either perspective gives the same result. To illustrate, the topological layout of a single level of a biquadratic five tree spline forest is shown in Fig. 17. In this case, $k_1 = \text{DOMAIN}(k_5, 1)$, $1 = \text{FACE}(k_5, 1)$, and $1 = \text{ORIENT}(k_5, 1)$ for subdomain k_5.

7.1.2. Side connectivity

For each side s of a subdomain k there may be any number of connected subdomains which share that side. We denote the number of side connected subdomains by n_s^k. For each side connection $i = 1, \ldots, n_s^k$ we store the connected subdomain k_i', its connected side s_i', and the relative orientation r_i. These are stored in the following maps: $k_i' = \text{DOMAIN}(k, s, i)$, $s_i' = \text{SIDE}(k, s, i)$, and $r_i = \text{ORIENT}(k, s, i)$. Given a shared side between two subdomains the orientation r is determined as follows:

Denote the subdomain k by a and the other by b. Denote the shared side corners of a by $sc^a = \{c_0^a, c_1^a\}$ and the shared side corners of b by $sc^b = \{c_0^b, c_1^b\}$. The relative orientation r is the index such that c_r^b is c_0^a.

7.1.3. Corner connectivity

For each corner c of a subdomain k there may be any number of connected subdomains which share that corner. We denote the number of corner connected subdomains by n_c^k. For each corner connection $i = 1, \ldots, n_c^k$ we store the connected subdomain k_i' and its connected corner c_i'. These are stored in the following maps: $k_i' = \text{DOMAIN}(k, c, i)$ and $c_i' = \text{CORNER}(k, c, i)$. Note that no relative orientation can be determined for a corner connection. Referring again to Fig. 17, we have that $k_4 = \text{DOMAIN}(k_5, 3, 1), k_2 = \text{DOMAIN}(k_5, 3, 2), k_3 = \text{DOMAIN}(k_5, 3, 3), k_1 = \text{DOMAIN}(k_5, 3, 4)$, and $0 = \text{CORNER}(k_5, 3, 1), 1 = \text{CORNER}(k_5, 3, 2), 1 = \text{CORNER}(k_5, 3, 3), 1 = \text{CORNER}(k_5, 3, 4)$ for subdomain k_5. Note that the order of the corner connected subdomains is not important.

7.2. Topological transformations

Using the subdomain connectivity information from Section 7.1 we can develop efficient algorithms for computing transformations between coordinate systems of neighboring subdomains. Algorithms 8–10 are the basic topological building blocks for local refinement, coarsening, and Bézier extraction of unstructured spline forests.

7.2.1. Face transformations

To pass information from one subdomain to another through a shared face a coordinate transformation must be computed if the two subdomains do not have identically aligned coordinate systems. Algorithm 8 presents an efficient algorithm for computing coordinate transformations of vertex, cell, or anchor indices through a shared face. If the parameter $h = 0, 1$ or 2 then $\mathbf{i}^\alpha$ corresponds to a vertex, cell, or anchor, respectively, and appropriate offsets are accounted for. First, the two coordinate axes tangential to the shared face and the normal axis are identified for subdomain k, denoted by s_{a_0}, s_{a_1}, and s_{a_2}. These axes are matched with a separate ordering of axes in the neighboring subdomain k', denoted by s_{b_0}, s_{b_1}, and s_{b_2}. To assist in enumerating all possible tangential axis orderings in 3D we use the following map [47]

$$\mathrm{R}(\mathrm{f}_1, \mathrm{f}_2) = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 & 1 & 0 \\ 0 & 2 & 2 & 0 & 0 & 1 \\ 0 & 2 & 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 2 & 2 & 0 \end{pmatrix}. \tag{21}$$

This map can be constructed by simply enumerating all possible orientations. The tangential axes can be aligned in parallel or anti-parallel coordinate directions. Once the two local face coordinate systems are determined each component of the transformed index $(\mathbf{i}^\alpha)'$ is computed. For example, referring to Fig. 17, we have that TRANSFORMFACE$(k_5, 1, \mathbf{i}^\alpha = \{8, 9\}, 0)$ returns the vertex $(\mathbf{i}^\alpha)' = \{8, 0\}$ in k_1.

Algorithm 8. Compute the transformed vertex, cell, or anchor index $(\mathbf{i}^\alpha)'$ in k', corresponding to an index $\mathbf{i}^\alpha$ in k, across a shared face

procedure TRANSFORMFACE(k, f, $\mathbf{i}^\alpha$, h)
  $k' \leftarrow$ DOMAIN(k, f) ▷ neighbor subdomain
  f′ $\leftarrow$ FACE(k, f) ▷ neighbor face
  $r \leftarrow$ ORIENT(k, f) ▷ relative orientation
  $a_2 \leftarrow$ f **div** 2 ▷ a_2 is origin normal axis, **div** denotes integer division
  $b_2 \leftarrow$ f′ **div** 2 ▷ b_2 is neighbor normal axis
  if $d_p = 2$ **then**
    $a_0 \leftarrow 1 - a_2; b_0 \leftarrow 1 - b_2; s_0 \leftarrow r$
  else
    $a_0 \leftarrow (f < 2)\,?\,1 : 0; a_1 \leftarrow (f < 4)\,?\,2 : 1$ ▷ a_0, a_1, a_2 are coordinate axes of f
    $u \leftarrow \mathrm{R0}, \mathrm{f} \oplus \mathrm{R0}, \mathrm{f}' \oplus (r = 0 \,|\, r = 3)$ ▷ flag for tangential axis ordering, $\oplus$ denotes bitwise exclusive OR
    $b_u \leftarrow (\mathrm{f}' < 2)\,?\,1 : 0; b_{1-u} \leftarrow (\mathrm{f}' < 4)\,?\,2 : 1$ ▷ b_0, b_1, b_2 are coordinate axes of f′
    $v \leftarrow (\mathrm{Rf}, \mathrm{f}' = 1)$
    $s_v \leftarrow r \,\&\, 1; s_{1-v} \leftarrow r \,\&\, 2$ ▷ s_0, s_1 are antiparallel flags for axes 0 and 1, & represents bitwise AND
  end if
  $g_0 \leftarrow (h < 2)\,?\,n^\alpha_{a_0} + p^\alpha_{a_0} - h : n^\alpha_{a_0} - 1$
  $(i^\alpha_{b_0})' \leftarrow (s_0 = 0)\,?\,i^\alpha_{a_0} : g_0 - i^\alpha_{a_0}$ ▷ write each component of $(\mathbf{i}^\alpha)'$
  if $d_p = 3$ **then**
    $g_1 \leftarrow (h < 2)\,?\,n^\alpha_{a_1} + p^\alpha_{a_1} - h : n^\alpha_{a_1} - 1$
    $(i^\alpha_{b_1})' \leftarrow (s_1 = 0)\,?\,i^\alpha_{a_1} : g_1 - i^\alpha_{a_1}$
  end if
  $o \leftarrow \mathrm{f} \,\&\, 1; o' \leftarrow \mathrm{f}' \,\&\, 1$
  if $o \,\&\, \neg o'$ **then**
  $(i^\alpha_{b_2})' \leftarrow -n^\alpha_{a_2} - p^\alpha_{a_2} + i^\alpha_{a_2}$
  else if $o \,\&\, o'$ **then**
    $(i^\alpha_{b_2})' \leftarrow (h < 2)\,?\,(n^\alpha_{b_2})' + (p^\alpha_{b_2})' + n^\alpha_{a_2} + p^\alpha_{a_2} - i^\alpha_{a_2} - h : (n^\alpha_{b_2})' + n^\alpha_{a_2} + p^\alpha_{a_2} - i^\alpha_{a_2} - 1$
  else if $\neg o \,\&\, \neg o'$ **then**
    $(i^\alpha_{b_2})' \leftarrow (h < 2)\,?\,-i^\alpha_{a_2} - h : -i^\alpha_{a_2} - (p^\alpha_{b_2})' - 1$
  else
    $(i^\alpha_{b_2})' \leftarrow (n^\alpha_{b_2})' + (p^\alpha_{b_2})' + i^\alpha_{a_2}$
  end if
  return $(\mathbf{i}^\alpha)'$
end procedure

7.2.2. Side transformations

Algorithm 9 presents a routine for computing coordinate transformations of vertex, cell, or anchor indices through a shared side. If the parameter $h = 0, 1$ or 2 then $\mathbf{i}^\alpha$ corresponds to a vertex, cell, or anchor, respectively, and appropriate offsets are accounted for. Note that if the neighbor subdomain is also connected to k through a face, a face transformation is instead computed using Algorithm 8. Otherwise, a local side coordinate system $s_{b_0}, s_{b_1}, s_{b_2}$ is determined for k'_i and each component of the transformed index $(\mathbf{i}^\alpha)'$ is computed. Note that, in a completely unstructured forest, where any number of subdomains can share a side, only s_{a_0} and s_{b_0}, corresponding to the shared side can be compared.

Algorithm 9. Compute the transformed vertex, cell, or anchor index $(\mathbf{i}^\alpha)'$ in k'_i, corresponding to an index $\mathbf{i}^\alpha$ in k, across a shared side

procedure TRANSFORMSIDE($k, i, \mathrm{s}, \mathbf{i}^\alpha, h$)
  $k'_i \leftarrow$ DOMAINk, s, i ▷ i^{th} neighbor subdomain through s
  $s'_i \leftarrow$ SIDE$k, (\mathrm{s}, i)$ ▷ shared side with respect to k'_i
  $r_i \leftarrow$ ORIENT(k, s, i) ▷ relative side orientation
  $faces \leftarrow$ SIDEFACES(k, s) ▷ transform through a face if possible
  for $j \leftarrow$ SIZE$(faces)$ **do**
    if $k'_i =$ DOMAIN$(k, faces(j))$ **then**
      return TRANSFORMFACE$(k, faces(j), \mathbf{i}^\alpha, h)$
    end if
  end for
  $b_0 \leftarrow \mathrm{s}'_i$ **div** 4 ▷ b_0, b_1, b_2 are coordinate axes of s'_i, **div** denotes integer division
  $b_1 \leftarrow (s'_i < 4)\,?\,1 : 0$
  $b_2 \leftarrow (s'_i < 8)\,?\,2 : 1$
  $a_0 \leftarrow \mathrm{s}$ **div** 4 ▷ write each component of $(\mathbf{i}^\alpha)'$
  $g_0 \leftarrow (h < 2)\,?\,(n^\alpha_{b_0})' + (p^\alpha_{b_0})' - h : (n^\alpha_{b_0})' - 1$
  $g_1 \leftarrow (h < 2)\,?\,(n^\alpha_{b_1})' + (p^\alpha_{b_1})' - h : (n^\alpha_{b_1})' - 1$
  $g_2 \leftarrow (h < 2)\,?\,(n^\alpha_{b_2})' + (p^\alpha_{b_2})' - h : (n^\alpha_{b_2})' - 1$
  $(i^\alpha_{b_0})' \leftarrow r_i g_0 + (1 - 2r_i) i^\alpha_{a_0}$
  $(i^\alpha_{b_1})' \leftarrow ((s'_i \,\&\, 1) = 0)\,?\,0 : g_1$
  $(i^\alpha_{b_2})' \leftarrow ((s'_i \,\&\, 2) = 0)\,?\,0 : g_2$
  return $(\mathbf{i}^\alpha)'$
end procedure

7.2.3. Corner transformations

Algorithm 10 presents an algorithm for computing coordinate transformations of vertex, cell, or anchor indices through a shared corner. If the parameter $h = 0, 1$ or 2, then $\mathbf{i}^\alpha$ corresponds to a vertex, cell, or anchor, respectively, and appropriate offsets are accounted for. Note that if the neighbor subdomain is also connected to k through a face or a side, a face or side transformation is instead computed using Algorithms 8 or 9. Otherwise, a local corner coordinate system $s_{b_0}, s_{b_1}, s_{b_2}$ is determined for k'_i and each component of the transformed index $(\mathbf{i}^\alpha)'$ is computed. Note that, in a completely unstructured forest, where any number of subdomains can share a corner, no coordinate system correspondence can be established between k and k'_i.

Algorithm 10. Compute the transformed vertex, cell, or anchor index $(\mathbf{i}^\alpha)'$ in k'_i, corresponding to an index $\mathbf{i}^\alpha$ in k, across a shared corner

procedure TRANSFORMCORNER($k, i, \mathrm{c}, \mathbf{i}^\alpha, h$)
  $k'_i \leftarrow$ DOMAIN(k, c, i) ▷ i^{th} neighbor subdomain through c
  $c'_i \leftarrow$ CORNER(k, c, i) ▷ shared corner with respect to k'_i
  $faces \leftarrow$ CORNERFACESk, c ▷ transform through a face if possible
  for $j \leftarrow$ SIZE$(faces)$ **do**
    if $k'_i =$ DOMAIN$(k, faces(j))$ **then**
      return TRANSFORMFACE$(k, faces(j), \mathbf{i}^\alpha, h)$
    end if
  end for
  if $d_p = 3$ **then**

(*continued on next page*)

```
    sides ← CORNERSIDES(k, c)        ▷ transform through a side if possible
    for j ← 1, SIZE(sides) do
      for jj ← 1, NUMSIDECONNECTIONS(k, sides(j)) do
        if k'_i = DOMAIN(k, sides(j), jj) then
          return TRANSFORMSIDE(k, jj, sides(j), i^α, h)
        end if
      end for
    end for
  end if
  for j ← 1, d_p do        ▷ write each component of (i^α)'
  step ← (h < 2) ? (n^α_j)' + (p^α_j)' − h : (n^α_j)' − 1
  (i^α_j)' ← (c'_i & 2^j) ? step : 0
  end for
end procedure
```

7.3. Interface basis functions and forest adaptivity

In this paper, we restrict our development to forest basis functions which are C^0 across subdomain interfaces. Smoother constructions are possible but necessitate the extension of analysis-suitable T-splines to the forest setting which is beyond the scope of this paper. Defining C^0 interface basis functions requires that the face, side, and corner transformations between adjacent subdomains can be determined as described in Section 7.2. Additionally, the construction of interface basis functions requires that for any two adjacent trees in the forest, the univariate index and parametric knot vectors at any level and the degree defined along the shared interface must be identical.

A straightforward way to visualize the construction of interface basis functions is in terms of the control mesh structure on each level of the forest hierarchy as shown in Figs. 18–20. In each case, only a single level of a forest hierarchy is shown since the same procedure is applied to every level. A five tree bilinear forest (each tree is bilinear) is shown in Fig. 18. To construct the interface basis functions across tree boundaries adjacent control points (or basis functions) are connected to form a single control point (or basis function) as shown in Fig. 18 on the left. This generates the C^0 forest shown on the right. The dark blue rectangles correspond to cells with zero parametric area and the white rectangles correspond to cells with non-zero parametric area. The open red circles

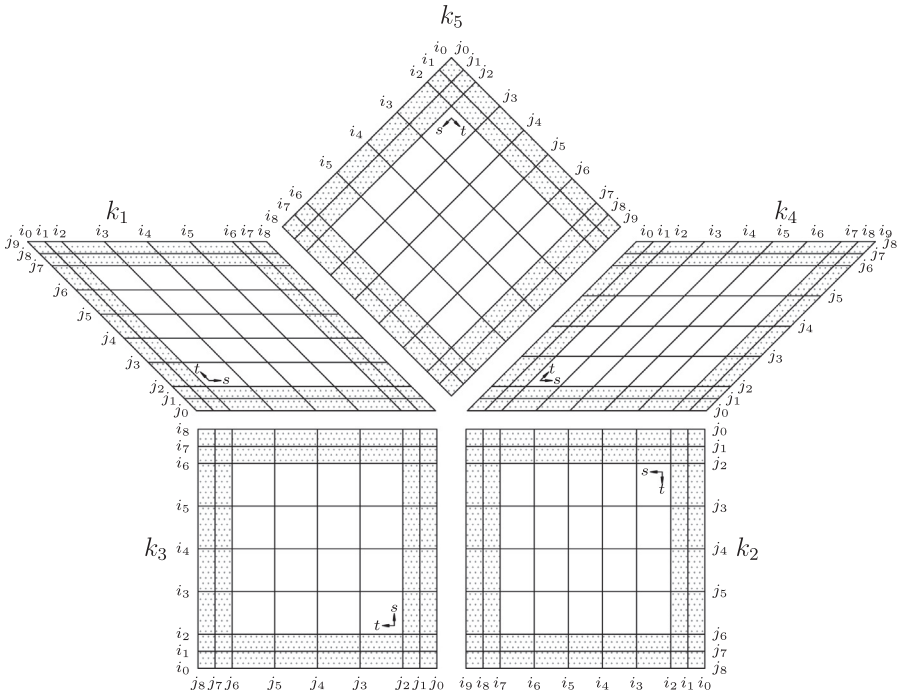


Fig. 17. The topological layout of a single level of a biquadratic five tree spline forest. Each spline tree has a unique coordinate system. Face and corner connectivity can be established between adjacent trees as described in Section 7.1.1, 7.1.3.

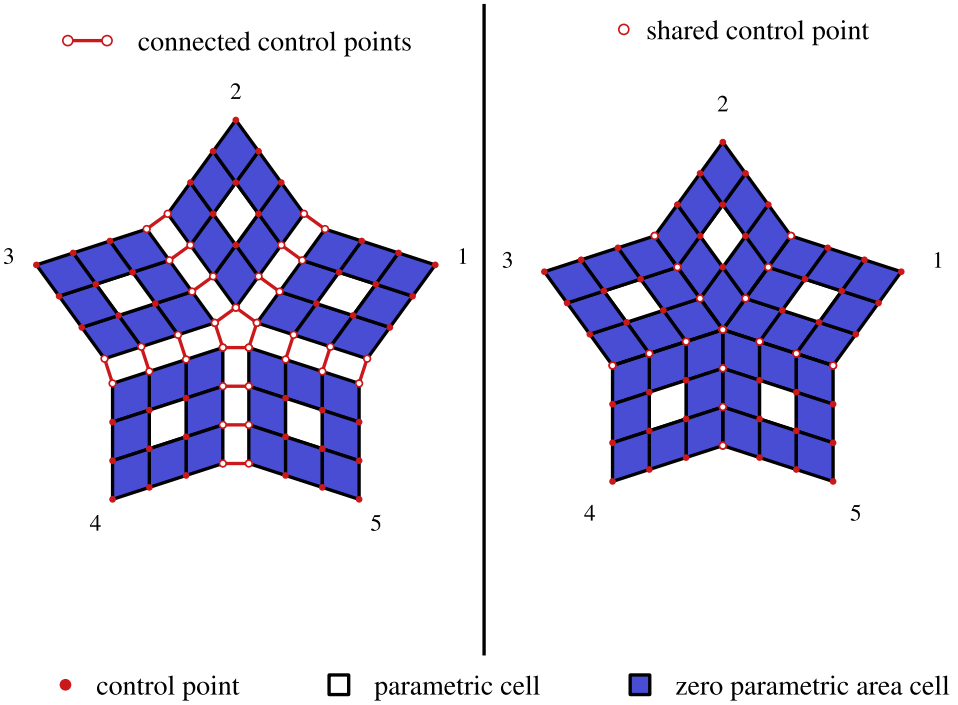


Fig. 18. Construction of a five tree bilinear C^0 forest. The connected control points (or, analogously, basis functions), denoted by red open circles on the left, are merged to form the C^0 forest on the right. The dark grey rectangles correspond to cells with zero parametric area and the white rectangles correspond to cells with non-zero parametric area. The open red circles denote connected control points while the solid red circles denote control points corresponding to a single tree in the forest. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

denote connected control points while the solid red circles denote control points corresponding to a single tree in the forest. The bilinear case is the simplest case. A slighly more complex case is shown in Fig. 19 for a biquadratic five tree forest. In this case there are shared control points and cells. The connected control points are shown on the left and the resulting C^0 forest is shown on the right. The light blue rectangles denote shared cells across tree boundaries.

Fig. 20 shows the construction of a mixed degree C^0 five tree forest. In this case, care must be taken to ensure that the interfaces share the same degree and index and parametric knot vectors. In this case, k_1 is degree 3-by-3, k_2 is degree 3-by-4, k_3 is degree 4-by-7, k_4 is degree 8-by-7, and k_5 is degree 8-by-3. The individual tree coordinate systems are shown by each tree. Fig. 21 plots several basis functions for a single level of a five tree biquadratic forest.

Once interface basis functions are defined, Algorithms 1 and 5 can be applied directly to refine or coarsen the forest. Fig. 22 shows an example refinement of a five tree biquadratic forest between levels α and $\alpha + 1$. All active leaf cells are shown in black while all inactive cells are in grey. On the left is the forest before refinement where several leaf cells, highlighted in red, have been selected for refinement. The result of Algorithm 1 is shown on the right where several cells on level $\alpha + 1$ have now been activated and have become new leaf cells in the forest. Note that, in our implementation, the refinement propagates through zero area parametric cells as shown in Fig. 22. This simplifies some aspects of the implementation but is not an essential ingredient.

8. Bézier extraction of spline forests

In a spline forest, a smooth, non-local, unstructured B-spline basis is represented across multiple hierarchical levels. Additionally, a distinct set of blending functions, also represented across the hierarchy, is used to represent geometric quantities. This makes the direct incorporation of hierarchical B-splines and requisite algorithms into an existing finite element framework impractical. Instead, we can use Bézier extraction [56,57], to generate a canonical finite element representation where the entire spline hierarchy is collapsed onto a single level finite element mesh and where both the hierarchical basis *and* geometry are represented in terms of the same set of Bernstein basis functions.

8.1. Localization

A forest finite element can be defined for every leaf element $\mathbf{e}^{\alpha,k} \in \mathbf{PE}_k$ where k is used to denote the particular hierarchical spline tree containing $\mathbf{e}^{\alpha,k}$. To simplify notation, we construct several index maps:

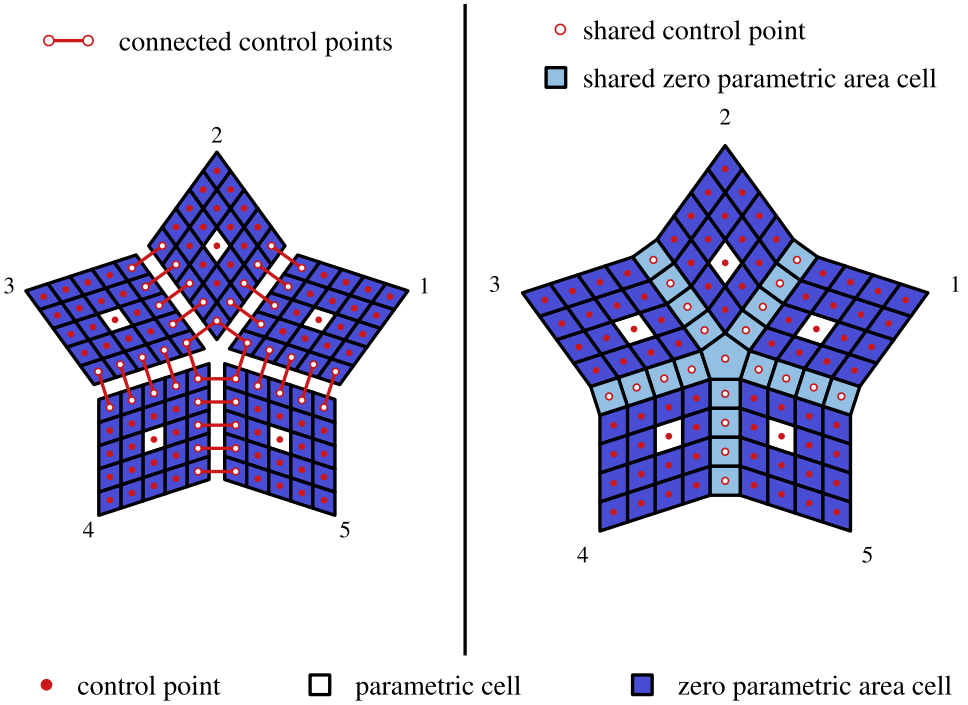


Fig. 19. Construction of a five tree biquadratic C^0 forest. The connected control points (or, analogously, basis functions), denoted by red open circles on the left, are merged to form the C^0 forest on the right. The dark grey rectangles correspond to cells with zero parametric area and the white rectangles correspond to cells with non-zero parametric area. The light grey rectangles denote shared cells across tree boundaries. The open red circles denote connected control points while the solid red circles denote control points corresponding to a single tree in the forest. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

1. The map which assigns a global element index e to every leaf element in $\mathbf{PE}_k, k = 1, \ldots, K$, such that $e = e(\mathbf{e}^{\alpha,k}), e = 1, \ldots, n_{el}$ where n_{el} is the total number of leaf elements in the forest.
2. The map which assigns a global function index A to every basis function in $\mathbf{HN}_k, k = 1, \ldots, K$, such that $A = A(\mathbf{f}^{\alpha,k}), A = 1, \ldots, n_f$ where n_f is the total number of hierarchical basis functions in the forest.
3. The map which assigns a global geometric function index G to every blending function in $\mathbf{GN}_k, k = 1, \ldots, K$, such that $G = G(\mathbf{g}^{\alpha,k}), G = 1, \ldots, n_g$ where n_g is the total number of geometric blending functions in the forest.
4. The standard local to global element index map which assigns a global function index A given an element index e and the local function index $a, a = 1, \ldots, n_f^e$, where n_f^e is the number of hierarchical basis functions supported by element e. We denote the map by $A = A(e, a)$ [63].
5. A local to global element index map which assigns a global geometric function index G given an element index e and the local geometric function index $g, g = 1, \ldots, n_g^e$, where n_g^e is the number of geometric blending functions supported by element e. We denote the map by $G = G(e, g)$ [63].

Note that the index maps (1), (2), and (4) will be familiar to finite element practitioners (see [63] for additional details) while index maps (3) and (5) are peculiar to spline forests. For simplicity we use $\hat{\Omega}^e$ to denote the parametric domain of element e and $\Omega^e \subset \mathbb{R}^{d_s}$ to denote the physical domain corresponding to $\hat{\Omega}^e$ under an appropriate geometric map. Using these index maps, we can transition to a standard index-based finite element notation independent of the topological complexity of the forest. For example, we have that

$$N_a^e = N_{A(e,a)} = N_{A(\mathbf{f}^{\alpha,k})} = N_{\mathbf{f},\mathbf{p}}^{\alpha,k}. \tag{22}$$

A rational hierarchical B-spline basis function, restricted to element e, can be written as

$$R_a^e(\mathbf{s}) = \frac{N_a^e(\mathbf{s})}{\sum_{g=1}^{n_g^e} w_g^e N_g^e(\mathbf{s})} \tag{23}$$

$$= \frac{N_a^e(\mathbf{s})}{w^e(\mathbf{s})} \tag{24}$$

where $\mathbf{s} \in \hat{\Omega}^e$ and $w^e(\mathbf{s})$ is the element weight function. The element geometric map $\mathbf{x} : \hat{\Omega}^e \to \Omega^e$ can be written as

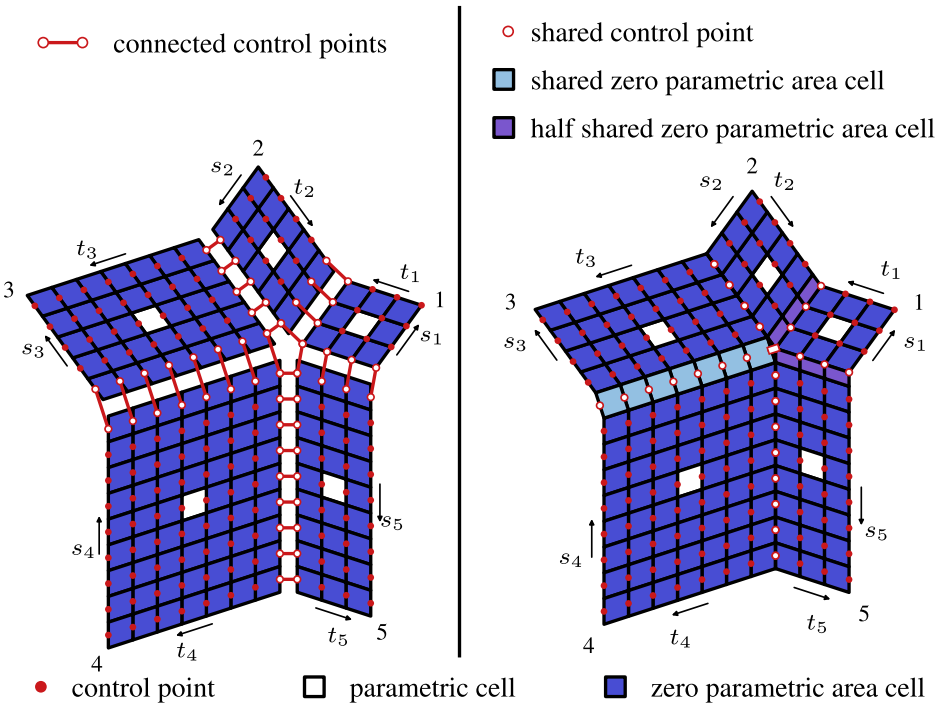


Fig. 20. Construction of a five tree mixed degree C^0 forest. Care must be taken to ensure that the interfaces between adjacent trees share the same degree and index and parametric knot vectors. In this case, k_1 is degree 3-by-3, k_2 is degree 3-by-4, k_3 is degree 4-by-7, k_4 is degree 8-by-7, and k_5 is degree 8-by-3. The individual tree coordinate systems are shown by each tree. The connected control points (or, analogously, basis functions), denoted by red open circles on the left, are merged to form the C^0 forest on the right. The dark blue rectangles correspond to cells with zero parametric area and the white rectangles correspond to cells with non-zero parametric area. The light blue rectangles denote shared cells across tree boundaries and the purple cells denote cells that are half shared. The open red circles denote connected control points while the solid red circles denote control points corresponding to a single tree in the forest. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

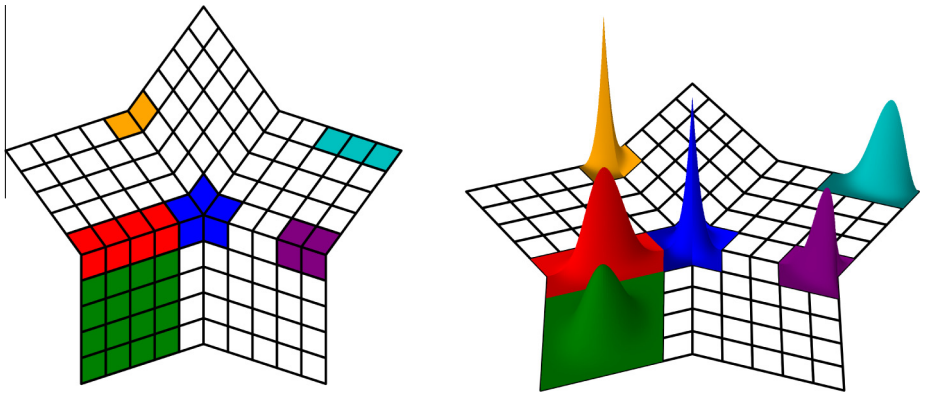


Fig. 21. Plots of several hierarchical B-spline basis functions on a five tree biquadratic forest.

$$\mathbf{x}^e(\mathbf{s}) = \sum_{g=1}^{n_g^e} \mathbf{P}_g^e w_g^e R_g^e(\mathbf{s}). \tag{25}$$

8.2. Bézier extraction

A Bézier element is a region of the spline forest in physical space bounded by knot lines. Each knot line in physical space is the image of a line of reduced continuity in at least one spline forest basis function. There is a one-to-one correspondence between the Bézier elements and the leaf elements in $\mathbf{PE}_k, k = 1, \ldots, K$. We call the collection of Bézier elements the Bézier mesh. We denote the Bézier mesh associated with a forest by F.

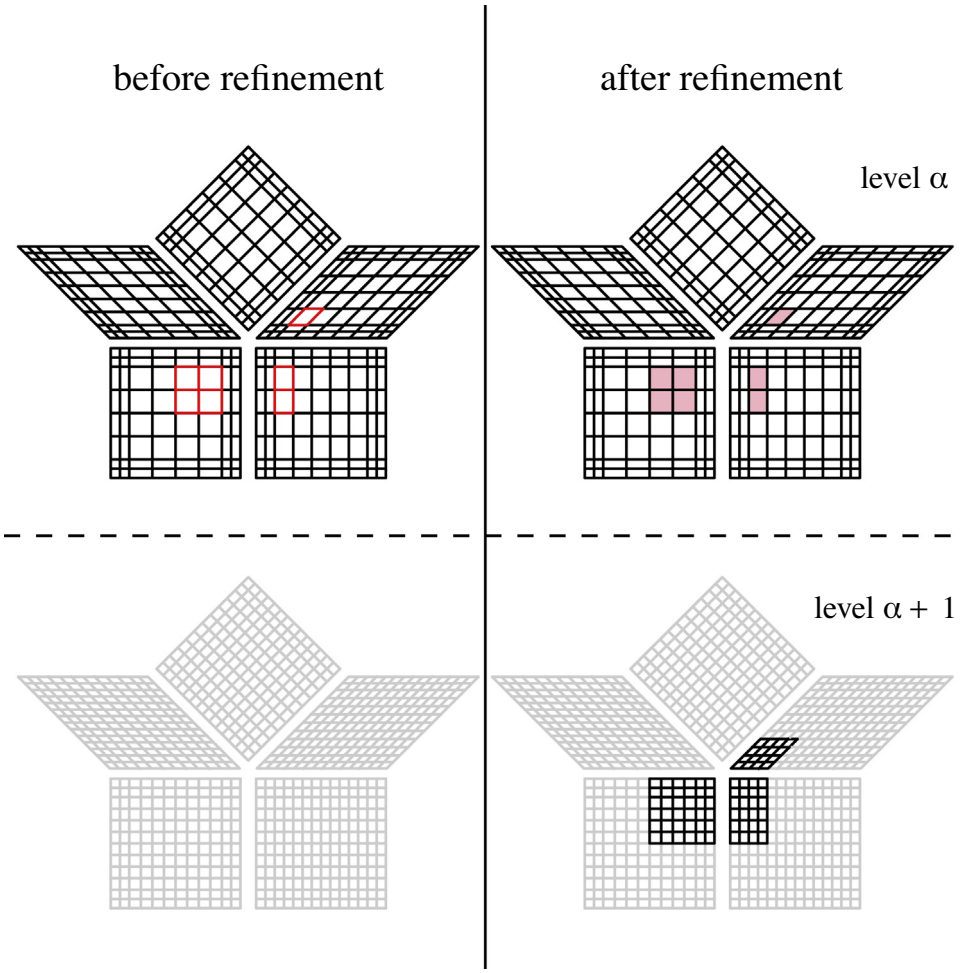


Fig. 22. An example refinement of a five tree biquadratic forest.

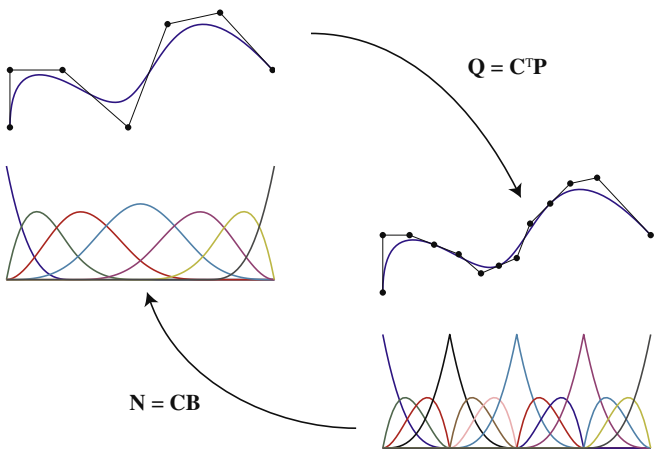


Fig. 23. Bézier extraction for a B-spline curve. B-spline basis functions and control points are denoted by **N** and **P**, respectively. Bernstein polynomials and control points are denoted by **B** and **Q**, respectively. The curve $T(\xi) = \mathbf{P}^T\mathbf{N}(\xi) = \mathbf{Q}^T\mathbf{B}(\xi)$. The extraction operator can be localized to the individual elements. Note that the Bernstein basis is the same for each element. Formation of element arrays can thus be standardized.

The distinguishing difference between Bézier extraction for single-level spline representations such as B-splines, NURBS, and T-splines [56,57] and multi-level representations like spline forests is that the set of basis functions extracted over a forest element e may span multiple levels of the hierarchy. To accommodate this, Bézier extraction must be performed for every level of the hierarchy.

To present the basic ideas, Bézier extraction for a B-spline curve is shown graphically in Fig. 23. Bézier extraction builds a linear operator for each Bézier element. The linear transformation is defined by a matrix referred to as the extraction operator. The extraction operator maps a Bernstein polynomial basis defined on Bézier elements to the global B-spline basis. The transpose of the extraction operator maps the control points of the B-spline curve to the Bézier control points. This element form can then be integrated into existing finite element frameworks in a straightforward manner.

In a spline forest, each hierarchical basis function supported by element e can be written in Bézier form as

$$N_a^e(\mathbf{s}(\xi)) = \sum_{b=1}^{n_b} c_b^a B_b(\xi) \tag{26}$$

$$= \overline{N}_a^e(\xi) \tag{27}$$

where the dependence of the Bernstein polynomial $B_b(\xi)$ on the polynomial degrees $\mathbf{p}$ has been suppressed for simplicity. The Bézier coefficients c_b^a are computed using standard knot insertion techniques [56]. Note that we use the overbar to denote a quantity which is written in terms of the local Bernstein basis defined over the element domain $\bar{\Omega}$. Denoting the vector of hierarchical basis functions supported by element e by $\bar{\mathbf{N}}^e(\xi)$ and the set of Bernstein polynomials by $\mathbf{B}(\xi)$ we have that

$$\bar{\mathbf{N}}^e(\xi) = \mathbf{C}^e \cdot \mathbf{B}(\xi) \tag{28}$$

where $\mathbf{C}^e$ is called the *element extraction operator*. In other words, the element extraction operator is composed of the Bézier coefficients c_b^a.

We then have that the element weight function can be written as

$$w^e(\mathbf{s}(\xi)) = \sum_{g=1}^{n_g^e} w_g^e N_g^e(\mathbf{s}(\xi)) \tag{29}$$

$$= \sum_{g=1}^{n_g^e} w_g^e \sum_{b=1}^{n_b} c_b^g B_b(\xi) \tag{30}$$

$$= \sum_{b=1}^{n_b} \left(\sum_{g=1}^{n_g^e} w_g^e c_b^g \right) B_b(\xi) \tag{31}$$

$$= \sum_{b=1}^{n_b} w_b^e B_b(\xi) \tag{32}$$

$$= \bar{w}^e(\xi). \tag{33}$$

We can write the rational hierarchical basis functions as

$$R_a^e(\mathbf{s}(\xi)) = \frac{N_a^e(\mathbf{s}(\xi))}{w^e(\mathbf{s}(\xi))} \tag{34}$$

$$= \frac{\overline{N}_a^e(\xi)}{\bar{w}^e(\xi)} \tag{35}$$

$$= \overline{R}_a^e(\xi). \tag{36}$$

Finally, the element geometric map can be written as

$$\mathbf{x}^e(\mathbf{s}(\xi)) = \frac{\sum_{g=1}^{n_g^e} \mathbf{P}_g^e w_g^e N_g^e(\mathbf{s}(\xi))}{w^e(\mathbf{s}(\xi))} \tag{37}$$

$$= \frac{\sum_{g=1}^{n_g^e} \mathbf{P}_g^e w_g^e \sum_{b=1}^{n_b} c_b^g B_b(\xi)}{\bar{w}^e(\xi)} \tag{38}$$

$$= \frac{\sum_{b=1}^{n_b}\left(\sum_{g=1}^{n_g^e}\mathbf{P}_g^e w_g^e C_b^g\right)B_b(\boldsymbol{\xi})}{\bar{w}^e(\boldsymbol{\xi})} \tag{39}$$

$$= \frac{\sum_{b=1}^{n_b}\mathbf{Q}_b^e w_b^e B_b(\boldsymbol{\xi})}{\bar{w}^e(\boldsymbol{\xi})} \tag{40}$$

$$= \bar{\mathbf{x}}^e(\boldsymbol{\xi}). \tag{41}$$

The implementation of a finite element framework based on Bézier extraction is described in detail in [56,57].

9. Applications in isogeometric analysis

We now explore the use of spline forests in the context of isogeometric analysis. Specifically, we focus on transient advection-diffusive transport problems in both 2D and 3D. We use a simple explicit residual-based *a posteriori* error estimator to drive the adaptivity and demonstrate the feasibility of the approach for advection-dominated phenomena, a notoriously difficult class of problems to solve accurately using finite element methods [64], and one for which smooth isogeometric discretizations have shown promise [6]. We use a geometrically exact and integrated adapt-extraction-analysis framework [29], in parallel, to solve each problem.

9.1. Preliminaries

Let Ω be a bounded region in $\mathbb{R}^{d_s}$ and assume Ω has a piecewise smooth boundary Γ. Let $\mathbf{x} = \{x_i\}_{i=1}^{d_s}$ denote a general point in $\overline{\Omega}$ and let $\mathbf{n} = \{n_i\}$ be the outward normal vector to Γ. Let Γ_g and Γ_h be subsets of Γ such that $\overline{\Gamma_g \cup \Gamma_h} = \Gamma$ and $\Gamma_g \cap \Gamma_h = \emptyset$. In this case, the superposed bar represents set closure and the $\emptyset$ denotes the empty set. The summation convention on repeated indices is assumed and a comma is used to denote partial differentiation. The Kronecker delta is denoted by δ_{ij}; if $i = j$, then $\delta_{ij} = 1$, otherwise $\delta_{ij} = 0$.

9.2. The transient advection–diffusion equation: Strong and weak formulations of the continuous problem

Let the temperature at a point $\mathbf{x} \in \Omega$ at time $t \in [0,T]$ be denoted by $\phi(\mathbf{x},t) \in \mathbb{R}$. Given time-dependent Dirichlet and Neumann boundary data $g : \Gamma_g \times]0,T[\to \mathbb{R}$ and $h : \Gamma_h \times]0,T[\to \mathbb{R}$, an initial temperature $\phi_0 : \Omega \to \mathbb{R}$, and a time-dependent source $f : \Omega \times]0,T[\to \mathbb{R}$ the advection–diffusion boundary value problem consists of finding the temperature ϕ such that

$$\begin{aligned}
&\phi_{,t} + \mathbf{u}\cdot\nabla\phi - \nabla\cdot(\boldsymbol{\kappa}\nabla\phi) = f \quad \text{on } \Omega\times]0,T[\\
&\phi = g \quad \text{on} \quad \Gamma_g \times]0,T[\\
&(\boldsymbol{\kappa}\nabla\phi)\cdot\mathbf{n} = h \quad \text{on} \quad \Gamma_h \times]0,T[\\
&\phi(\mathbf{x},0) = \phi_0 \quad \mathbf{x}\in\Omega
\end{aligned} \tag{42}$$

where $\mathbf{u} : \Omega \to \mathbb{R}$ and $\boldsymbol{\kappa} : \Omega \to \mathbb{R}^{d_s \times d_s}$ are the spatially varying solenoidal velocity vector and symmetric, positive-definite, diffusivity tensor, respectively. Note that in this paper we define $\boldsymbol{\kappa} = \kappa\delta_{ij}$ where κ is a positive constant called the diffusivity coefficient. Defining the solution and weighting spaces as

$$\mathcal{S}_t = \{\phi(\cdot,t) \in H^1(\Omega) \,|\, \phi(\mathbf{x},t) = g(\mathbf{x},t) \text{ on } \Gamma_g\} \tag{43}$$

$$\mathcal{V} = \{w \in H^1(\Omega) \,|\, w = 0 \text{ on } \Gamma_g\} \tag{44}$$

and

$$B(w,\phi)_t = (w,\phi_{,t})_\Omega + (w,\mathbf{u}\cdot\nabla\phi)_\Omega + (\nabla w,\boldsymbol{\kappa}\nabla\phi)_\Omega \tag{45}$$

$$L(w) = (w,f)_\Omega + (w,h)_\Gamma \tag{46}$$

where $(\cdot,\cdot)_\Omega$ denotes the L^2-inner product on Ω, the variational counterpart of Eq. 42 is to find $\phi(t) \in \mathcal{S}_t, t \in [0,T]$ such that for all $w \in \mathcal{V}$

$$\begin{aligned}
&B(w,\phi)_t = L(w)\\
&(w,\phi(0))_\Omega = (w,\phi_0)_\Omega.
\end{aligned} \tag{47}$$

9.3. The semi-discrete SUPG method

As described in Section 8, Bézier extraction discretizes Ω into element subdomains $\Omega^e, e = 1,\ldots,n_{el}$ such that $\bigcup_e \overline{\Omega^e} = \Omega$ and $\bigcap_e \Omega^e = \emptyset$. Let $\mathcal{S}_t^h \subset \mathcal{S}_t$ and $\mathcal{V}^h \subset \mathcal{V}$ be the finite dimensional approximations to the function spaces used in Eq. 47. The semi-discrete SUPG form of the problem is to find $\phi^h \in \mathcal{S}_t^h$ such that for all $w^h \in \mathcal{V}^h$

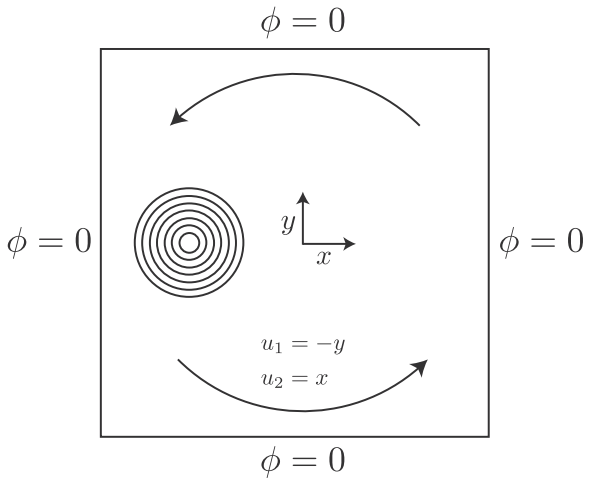


Fig. 24. The rotating cone in a square problem statement.

$$\begin{aligned} &B(w^h, \phi^h)_t + \sum_{e=1}^{n_{el}} (\mathbf{u} \cdot \nabla w^h \tau^e, \mathcal{L}_t \phi^h - f)_{\Omega^e} = L(w^h) \\ &(w^h, \phi^h(0))_\Omega = (w^h, \phi_0)_\Omega \end{aligned} \tag{48}$$

where $\mathcal{L}_t = \phi_{,t} + \mathbf{u} \cdot \nabla \phi - \nabla \cdot \boldsymbol{\kappa} \nabla \phi$. The intrinsic element time scale, τ^e, is taken to be

$$\tau^e = \frac{h^e_{\mathbf{u}}}{2|\mathbf{u}|} \min\left(1, \frac{1}{3p^2} Pe\right). \tag{49}$$

The element Peclet number Pe, is defined as

$$Pe = \frac{|\mathbf{u}| h^e_{\mathbf{u}}}{2|\boldsymbol{\kappa}|}, \tag{50}$$

where $h^e_{\mathbf{u}}$ is the element size in the direction of the flow and p is the polynomial order of the basis. We note that there are many possible choices for the stabilization parameter τ^e in the literature (see [9], and references therein), each with different strengths and weaknesses. Our choice is based on simplicity and seems to be adequate for the class of problems considered in this paper. The explicit representation of ϕ^h and w^h, restricted to an extracted Bézier element e, in terms of forest basis functions and nodal control variables is

$$\phi^h|_e = \sum_{a=1}^{n^e_f} \overline{R}^e_a(\xi) d^e_a(t) \tag{51}$$

$$w^h|_e = \sum_{a=1}^{n^e_f} \overline{R}^e_a(\xi) c^e_a(t). \tag{52}$$

We integrate in time using the generalized-α method [65,66]. For the sake of brevity we omit the details of the algorithm and refer the interested reader to [7,9]. We only mention in passing that, for all the examples in this paper, the spectral radius $\rho_\infty \in [0, 1]$ was set to 0.5. This has been shown to be an efficient choice for fluids computations [9]. The spectral radius provides a convenient mechanism to control high-frequency dissipation in semi-discrete formulations. The effect of high-frequency damping on the unresolved error and subsequent adaptivity as well as efficient time stepping schemes for adaptive isogeometric applications are important areas of future research.

9.4. Error estimation and element scaling

The development of adaptive methods based on explicit residual-based *a posteriori* error estimators for fluids emanating from the variational multiscale theory has been shown to be a robust and simple approach for many problem classes [67–70]. The exact solution, ϕ, is decomposed into

$$\phi = \phi^h + \phi' \tag{53}$$

where ϕ^h are the resolved scales (the finite element solution), and ϕ' are the unresolved scales or error. The variational multiscale theory is then applied to find an approximation of a local norm of ϕ'. For the results in this paper we use the error estimate given by

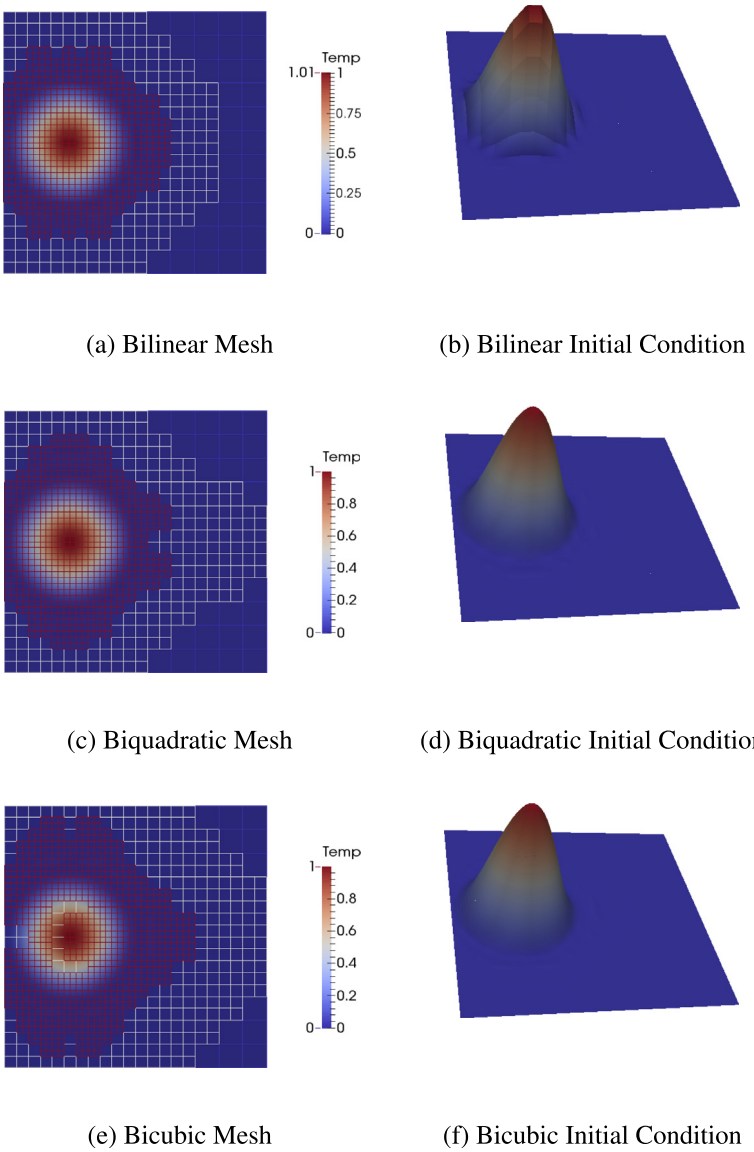
$$||\phi'||_{\Omega^e} \approx \tau^e ||\mathcal{L}_t \phi^h - f||_{\Omega^e}. \tag{54}$$

Note that this error estimator underestimates the error for diffusion-dominated flows but is adequate for the advection-dominated benchmarks presented in this paper [69,70].

When moving from Bézier mesh F_i to Bézier mesh F_{i+1} during an adaptive step we use the element error estimate and standard arguments [63] to define an appropriate Bézier element rescaling. Given a user defined error tolerance, *tol*, we have that

$$r = \frac{h^e_{(i+1)}}{h^e_{(i)}} = \left(\frac{tol}{||\phi'||_{\Omega^e}} \right)^{\frac{1}{\beta}} \tag{55}$$

where $h^e_{(i)}, h^e_{(i+1)}$ are the mesh size distributions for F_i and F_{i+1}, respectively, and β is the order of convergence of the method. The element size, h^e, is taken to be the square root of the element area for surfaces and the cube root of the element volume

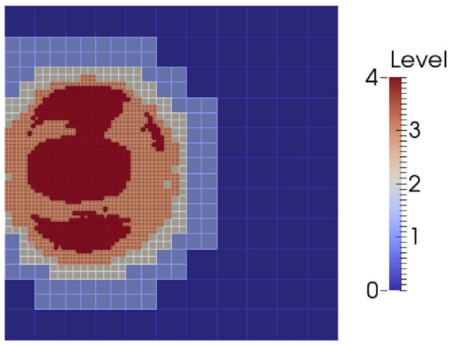


(a) Bilinear Mesh (b) Bilinear Initial Condition

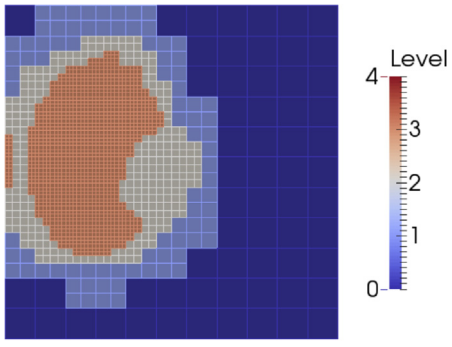
(c) Biquadratic Mesh (d) Biquadratic Initial Condition

(e) Bicubic Mesh (f) Bicubic Initial Condition

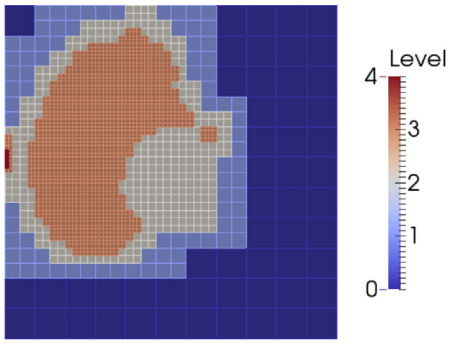
Fig. 25. The cosine hill initial condition after adaptive L_2-projection for bilinear, biquadratic, and bicubic hierarchical B-splines.



(a) Bilinear Mesh



(b) Biquadratic Mesh



(c) Bicubic Mesh

Fig. 26. Adaptive bilinear, biquadratic, and bicubic Bézier meshes at time step 600 for the rotating cone problem (see Fig. 24).

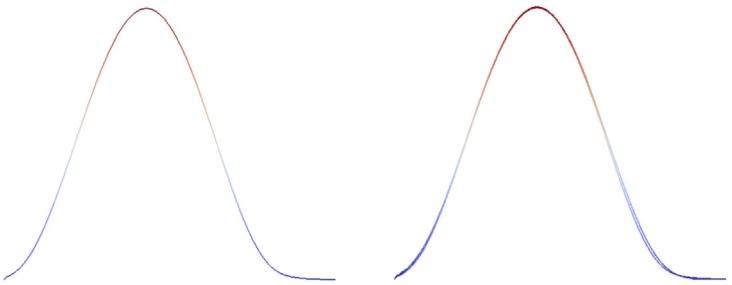
for solids. We then flag elements either for refinement or coarsening if $r < C_r$ or $r > C_c$ where C_r and C_c are user defined thresholds. These elements are then processed by Algorithms 1 and 5 to generate an adapted forest. This adaptive process can be repeated until a specified convergence tolerance is attained or a maximum number of hierarchical levels are introduced. In our implementation we perform adaptivity within each time step although other alternatives exist.

9.5. Results

We study the behavior of an integrated adaptive design and analysis environment on several benchmark problems. In all cases, refinement, coarsening, Bézier extraction, and isogeometric analysis are tightly coupled, geometrically exact, and scalable. To validate our approach, all simulations were performed in parallel using iterative solvers. While an important and interesting topic in its own right, the specifics of the parallel implementation will be treated in a forthcoming paper.

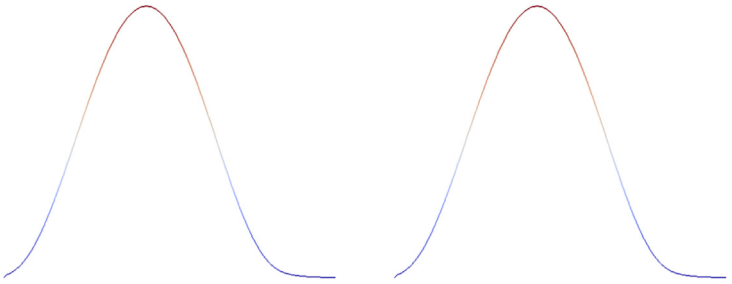
9.5.1. A rotating cone in a square

We first study the behavior of adaptive isogeometric analysis for the classical rotating cone in a square benchmark [64]. The problem setup is shown in Fig. 24. The flow is rigid rotation about the center of the bi-unit square domain, with velocity



(a) Overkill solution at time step 1200

(b) Overkill and bilinear solution at time step 1200



(c) Overkill and biquadratic solution at time step 1200

(d) Overkill and bicubic solution at time step 1200

Fig. 27. A slice of the solution along the x-axis at time step 1200 for an overkill solution and all solutions (bilinear, bicubic, biquadratic) superimposed on the overkill solution. Note that the only visible variance in the solutions is in the linear solution. The biquadratic and bicubic solutions are indistinguishable from the overkill solution.

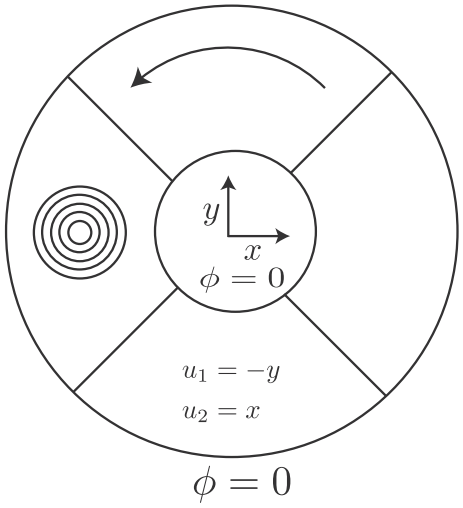


Fig. 28. The rotating cone in an annulus problem statement.

components given by $u_1 = -y$ and $u_2 = x$. The initial condition is a cosine hill. The problem is advection dominated, with diffusivity of 10^{-6}. Along the external boundary, ϕ is set to zero. We study the behavior of the method for C^0 bilinear, C^1 biquadratic, and C^2 bicubic hierarchical B-splines. In all cases a linear parameterization of the geometry is utilized with an initial 11-by-11 Bézier mesh. A spectral radius of 0.5 is used in the generalized-α method. Two full rotations of the cosine hill are performed in 1200 time steps. The time step Δt is chosen such that $\Delta t \approx h_{min}$ where h_{min} is the smallest element appearing in

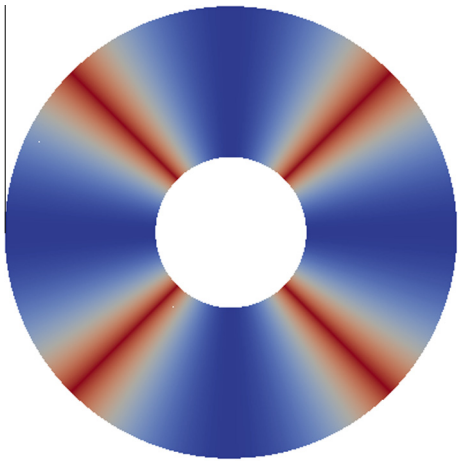


Fig. 29. The weight function used to define the annulus in Fig. 28. A non-unity weight function engenders rational hierarchical basis functions.

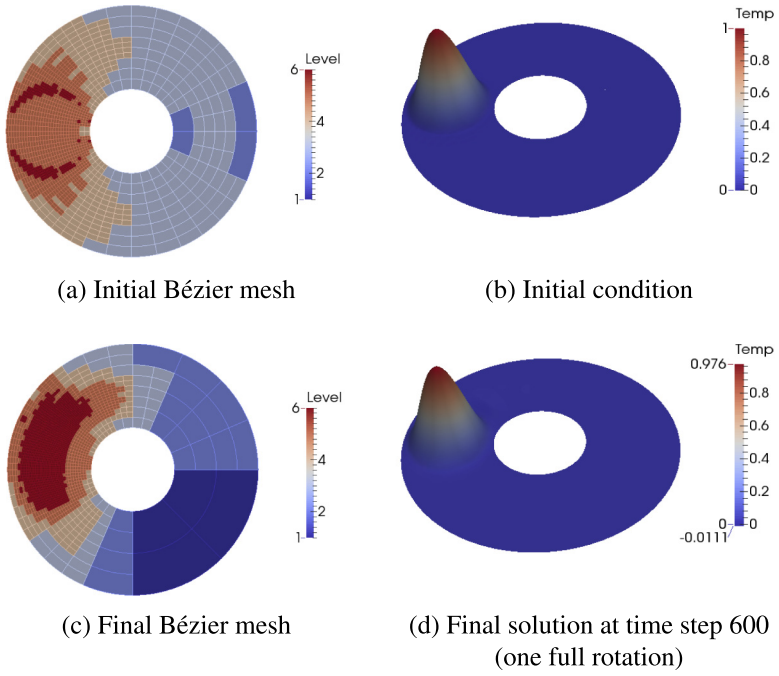


Fig. 30. The initial and final Bézier meshes and solutions for the rotating cone in an annulus problem.

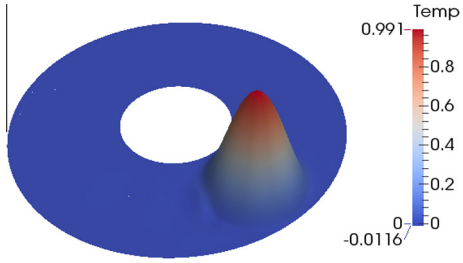


Fig. 31. The solution of the rotating cone in an annulus problem at time step 225. At this time step the solution straddles a tree interface. Notice that no interface artifacts are present in the solution despite six levels of hierarchical refinement.

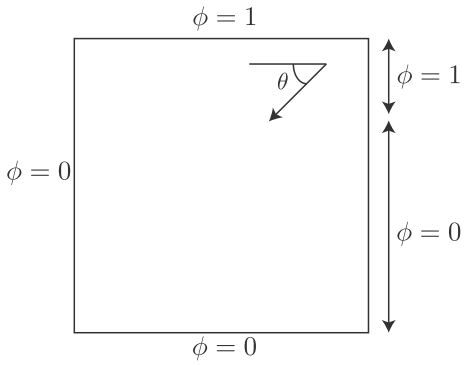
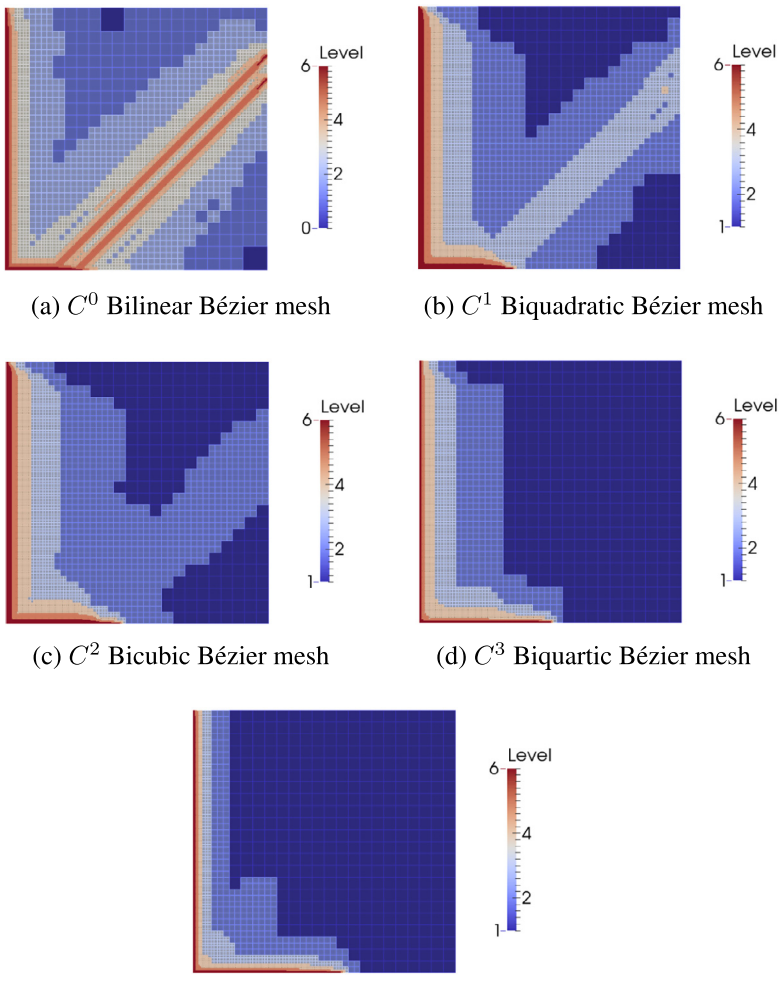


Fig. 32. The advection skew to the mesh problem statement.



(a) C^0 Bilinear Bézier mesh

(b) C^1 Biquadratic Bézier mesh

(c) C^2 Bicubic Bézier mesh

(d) C^3 Biquartic Bézier mesh

(e) C^4 Biquintic Bézier mesh

Fig. 33. Converged Bézier meshes for the static advection skew to the mesh problem corresponding to bilinear, biquadratic, bicubic, biquartic, and biquintic hierarchical B-splines of maximal smoothness.

the mesh during the adaptive simulation. This is a challenging benchmark due to the length of the simulation (two full rotations) and the difficulty in maintaining spatial accuracy such that excessive amplitude decay and phase error do not pollute the solution. Additionally, passing the solution between fine and coarse grids at every time step introduces the possibility for accuracy loss.

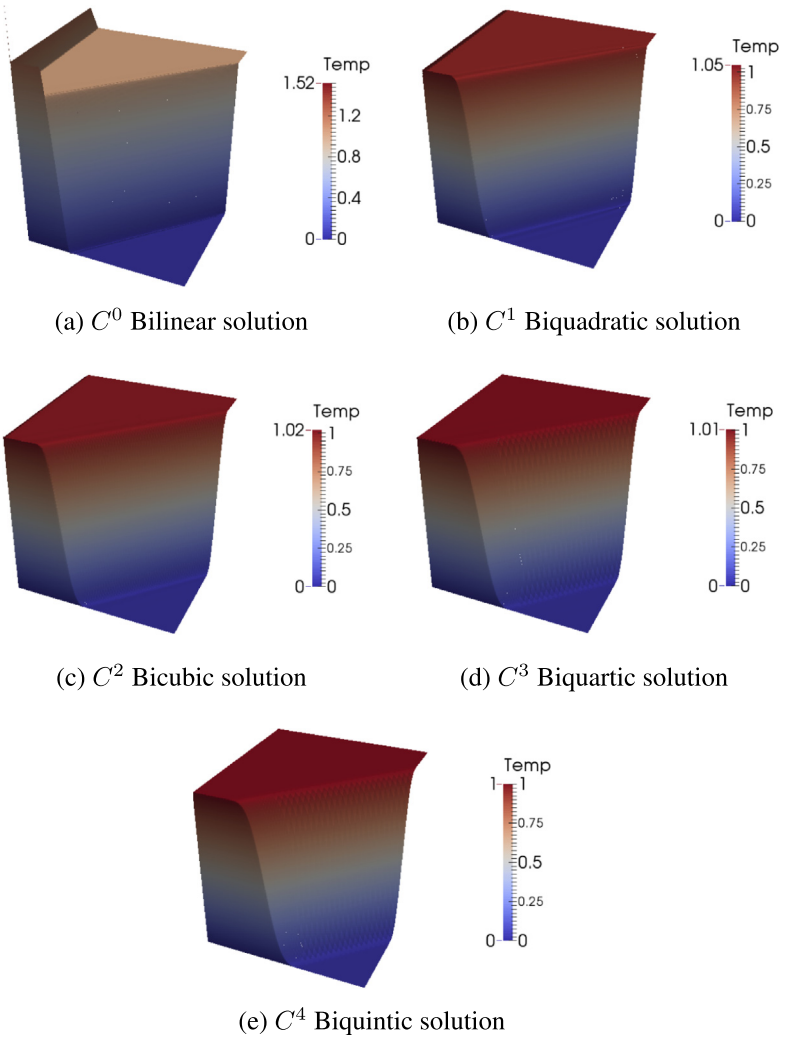


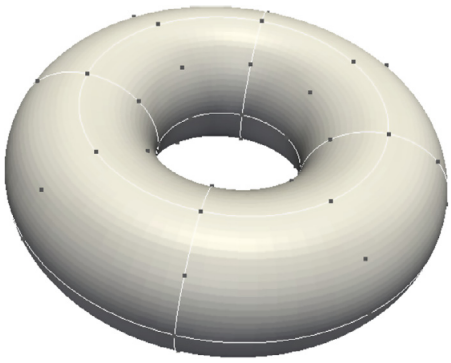
Fig. 34. Converged static solutions to the advection skew to the mesh problem corresponding to bilinear, biquadratic, bicubic, biquartic, and biquintic hierarchical B-splines of maximal smoothness. These solutions correspond to the Bézier meshes in Fig. 33.

Table 1
Maximum overshoots and undershoots for the advection skew to the mesh problem shown in Fig. 32.

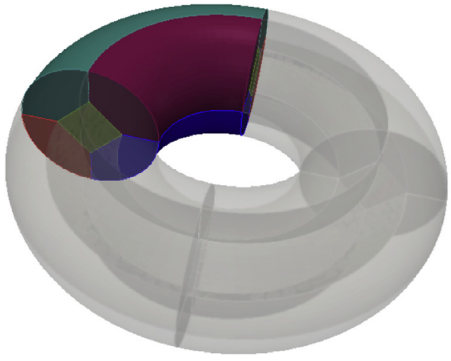
Degree	Max overshoot	Max undershoot
Bilinear (C^0)	1.5201	−0.005742
Biquadratic (C^1)	1.0491	−0.002316
Bicubic (C^2)	1.0214	−0.000622
Biquartic (C^3)	1.0061	−0.0003354
Biquintic (C^4)	1.0022	−0.00005266

An adaptive L_2-projection is used to fit the initial mesh to the initial conditions. The result of the projection is shown in Fig. 25 where the solution is superimposed on the adaptively refined Bézier mesh on the left and an elevation view of the resulting initial condition is shown on the right. As expected, the smooth biquadratic and bicubic meshes are better at capturing the smooth cosine hill initial condition.

The balance parameter, n, for both coarsening and refinement is set to 1 to ensure a gradual transition between coarse and fine regions of the Bézier mesh. The mesh is refined during every time step and coarsened every 10 time steps. The *a posteriori* error tolerance is set to 10^{-4} and the user-defined thresholds for refinement, C_r, and coarsening, C_c, are set to 1 and 10, respectively. We have found that a conservative approach to coarsening is important to maintain spatial accuracy in the solution. The maximum level allowed in the hierarchical mesh is 4 and pruning is performed during mesh adaptivity.



(a) An exact torus



(b) The topology of the underlying 20 tree forest.

Fig. 35. A twenty tree biquadratic spline forest of an exact torus. (a) The geometry of the torus. The greville abscissae of the biquadratic B-spline basis functions are denoted by black circles and the outlines of the Bézier elements are delineated by white lines. (b) An interior view of the torus tree topology and parameterization.

A preconditioned GMRES [71] iterative solver with a tolerance of 10^{-10} is used to solve the linear system. We use the linear solvers found in the Trilinos open-source package for all the examples in this paper [72].

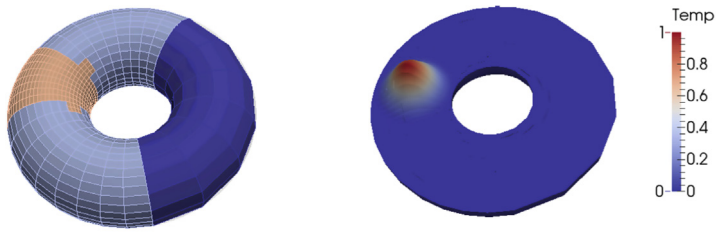
Because nested spaces are generated during refinement, the current solution is maintained exactly when passed from coarse to fine grids. For coarsening, however, a global L_2-projection is performed to generate a lossy representation of the current solution on the fine grid in terms of the coarse grid control variables.

The Bézier mesh at time step 600 is shown in Fig. 26. The adaptivity has closely followed the advected cosine hill. Notice that many level four bilinear elements (Fig. 26(a)) are active where very few level four biquadratic or bicubic elements are active. In other words, more bilinear elements are required to resolve the solution. This is a trend that has been observed generally for isogeometric discretizations [73]. Smooth discretizations provide increased spatial accuracy with fewer elements than C^0 discretizations. This has a dramatic impact on the efficiency of dynamic simulations. In this case, the C^1 biquadratic discretizations ran faster than the C^0 bilinear discretizations for the same *a posteriori* error tolerance.

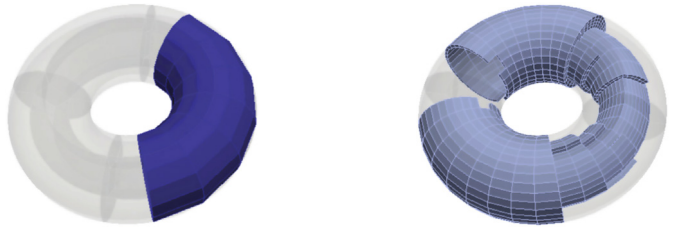
The final solutions at time step 1200 (two full rotations) for the bilinear, biquadratic, and bicubic simulations are compared to an overkill solution in Fig. 27. The overkill solution is computed using a uniformly refined C^1 biquadratic mesh of approximately 50,000 elements. The final peak amplitude in the overkill solution is 0.949. Note that there is some diffusion in the overkill solution at the end of the simulation but this can be attributed to the small amount of diffusion that is present in the method itself. In Fig. 27(b) through Fig. 27(d) all solutions (bilinear, bicubic, biquadratic) are superimposed on the overkill solution. Note that the only visible variance in the solutions is due to the linear solution. The biquadratic and bicubic solutions are indistinguishable from the overkill solution. This is a compelling example of the accuracy of the approach and the efficiency of the error estimator for this class of problems.

9.5.2. A rotating cone in an annulus

In this example, we solve the rotating cone problem on a four tree biquadratic forest of an annulus. In this case, the geometry is non-trivial and the adaptivity must correctly propagate through the boundaries of adjacent trees. The problem



(a) Initial Bézier mesh (b) Initial condition



(c) Bézier mesh at level 2 (d) Bézier mesh at level 3

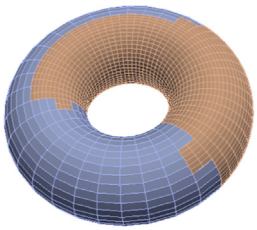
(e) Bézier mesh at level 4 (f) Bézier mesh at level 5

Fig. 36. The initial Bézier mesh (a), initial condition (b), and Bézier meshes at each hierarchical level (c)–(f) for the advection in a torus problem.

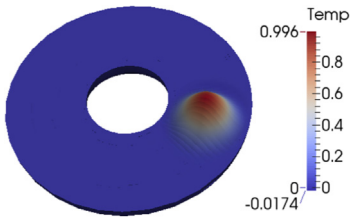
statement is shown in Fig. 28 where the boundaries of the four trees are also delineated. The inner and outer radii are 0.5 and 1.5, respectively. The weight function for the annulus is shown graphically in Fig. 29. This engenders rational basis functions in the simulation as described in Section 5.1.

The problem is advection dominated, with diffusivity of 10^{-6}. The initial condition is a cosine hill. Along the external boundary ϕ is set to zero. As before, the spectral radius is 0.5, generalized-α is used to advance the solution in time, the balance parameter for refinement and coarsening is 1, and the mesh is refined every time step and coarsened every 10 time steps. The *a posteriori* error tolerance is set to 10^{-4} and the user defined thresholds for refinement, C_r, and coarsening, C_c, are set to 1 and 10, respectively. The maximum level allowed in the hierarchical mesh is 7. To solve the linear system GMRES is used. In this case, 600 time steps are performed which constitutes one full rotation around the annulus.

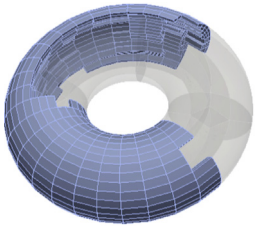
Fig. 30 shows the initial and final Bézier meshes as well as the initial condition and final solution for the rotating cone in an annulus problem. The initial Bézier mesh is adaptively refined to fit the initial condition. Notice that the exact geometry is preserved throughout the simulation. Six levels of refinement are performed during this simulation. Using the balance parameter to control basis conditioning it is possible to use iterative solvers even with high levels of refinement and smooth basis functions. Similar accuracy is obtained for this problem as for the rotating cone in a square problem. We note that more advanced definitions of the element stabilization parameter, τ^e, which correctly account for curved geometry, may produce even better solutions and will be explored in future works. Fig. 31 shows the solution of the rotating cone in an annulus problem at time step 225. At this point in the simulation the solution straddles a tree interface. Notice that no interface artifacts are present in the solution despite six levels of hierarchical refinement. Additionally, due the integer based encoding of the forest topology and efficient transformation algorithms presented in Section 7.1, no appreciable degradation of performance is observed for many tree forests.



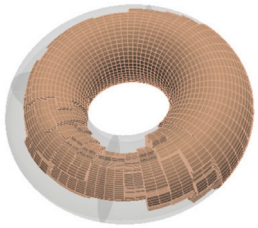
(a) Bézier mesh at time step 200



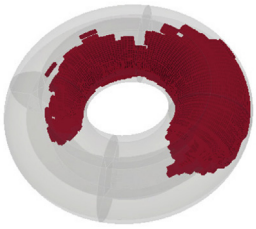
(b) Solution at time step 200



(c) Bézier mesh at level 3 at time step 200



(d) Bézier mesh at level 4 at time step 200



(e) Bézier mesh at level 5 at time step 200

Fig. 37. The Bézier mesh (a), solution (b), and Bézier meshes at each hierarchical level (c)–(e) at time step 200 (halfway around the torus) for the advection in a torus problem.

9.5.3. Advection skew to the mesh

We now study the behavior of higher-order smooth adaptive hierarchical discretizations in the presence of sharp interior and boundary layers. The problem setup is the advection skew to the mesh problem shown in Fig. 32. The problem is advection dominated, with diffusivity of 10^{-6}. Along the external boundary, the boundary conditions are selected such that sharp interior and boundary layers are present in the solution. In this case, $\theta = 45$ degrees. This is a demanding problem for any discretization scheme. Smooth isogeometric discretizations have been shown to perform especially well in resolving boundary layers [6]. The error tolerance is set to 10^{-4} and the user defined threshold for refinement, C_r, is set to 1. No coarsening is performed in this problem. No maximum refinement level is set for this problem and full convergence to the specified error tolerance is allowed. GMRES is used to solve the linear system.

We solve the problem with bilinear, biquadratic, bicubic, biquartic, and biquintic hierarchical B-splines of maximal smoothness. The converged Bézier meshes are shown in Fig. 33. Notice that less elements are required for convergence as the smoothness and order of the basis increases. The converged solutions are shown in Fig. 34. The overall trend can be seen by comparing the maximum overshoot and undershoot in the solutions as the order of the basis and smoothness increases. The bilinear solution has very large overshoots while the biquintic examples is essentially converged. Table 1 shows the maximum values of the overshoots and undershoots for each mesh.

9.5.4. Volumetric forests

We now investigate the possibility of defining complex parameterizations of solids using isogeometric spline forests and their use in an adaptive isogeometric analysis setting. As an example, we construct a biquadratic twenty tree forest of an exact torus. In this case, the tree layout is chosen such that no degeneracies in the parameterization exist; this is done to avoid degradation in the

performance of the discretization in an analysis context [74,75]. The exact torus and interior tree topology and parameterization are shown in Fig. 35. In Fig. 35(a) the Greville abscissae of the biquadratic B-spline basis functions are denoted by black circles and the outline of each level zero Bézier element is delineated by white lines. The inner and outer radii are 1 and 3, respectively. Note that in this case there are four *extraordinary* sides running through the interior of the torus (see Fig. 35(a)). In other words, there are subdomain sides which are coincident with n subdomains, $n \neq 4$. The ability to correctly and efficiently handle extraordinary sides (or vertices) is essential for the representation of geometry of arbitrary topological genus.

The problem is advection dominated, with diffusivity of 10^{-6}. The initial condition is a volumetric cosine hill. Along the external boundary ϕ is set to zero. The spectral radius is 0.5, generalized-α is used to advance the solution in time, the balance parameter for refinement and coarsening is 1, and the mesh is refined every 5 time steps and coarsened every 10 time steps. The *a posteriori* error tolerance is set to 10^{-4} and the user defined thresholds for refinement, C_r, and coarsening, C_c, are set to 1 and 10, respectively. The maximum level allowed in the hierarchical mesh is 5. To solve the linear system GMRES is used with incomplete LU factorization used as a preconditioner. In this case, 400 time steps are performed which constitutes one full rotation around the torus.

The initial Bézier mesh, initial condition, and initial Bézier meshes at each hierarchical level are shown in Fig. 36. An adaptive L_2-projection is used to generate the initial Bézier mesh.

Fig. 37 shows the result of the torus simulation at time step 200 or halfway around the torus. The solution is shown in Fig. 37(b). The behavior of the method in the volumetric setting is similar to what was observed in the surface case: very little amplitude decay or phase error is present in the solution. The Bézier meshes at each hierarchical level of refinement are shown in Fig. 37(c) through Fig. 37(e). During the adaptive simulation, approximately 150,000 Bézier elements are used. Note that to achieve the same accuracy with a globally refined NURBS model of the torus would require several million elements.

10. Conclusion

We have presented spline forests and demonstrated their potential as a basis for isogeometric analysis. We developed h-refinement and coarsening algorithms as well as a Bézier extraction framework for hierarchical spline representations. These algorithms can be applied to refinement and coarsening across tree boundaries. This generalizes the Bézier extraction framework introduced in [57,56]. We then demonstrated the analysis potential of spline forests by applying them to challenging transient advection–diffusion benchmark problems for both surfaces and solids. To drive the adaptivity we used a simple residual-based *a posteriori* error estimator emanating from the variational multiscale theory. In all cases, remarkable accuracy was achieved with very few degrees-of-freedom.

In future work we will extend the definition of spline forests to the T-spline regime. This will allow us to accomodate smooth interfaces and directly interface with advanced commercial design capabilities based on T-spline surfaces [76]. This will lead to revolutionary new design-through-analysis methodologies.

Appendix A. Notational conventions

In Tables A.1 and A.2 the notational conventions used throughout the text are listed as well as the section where they are defined.

Table A.1
Notational conventions used throughout the text and where they are defined.

Symbol	Description	Section
d_p	Number of parametric dimensions	Section 1.1
d_s	Number of spatial dimensions	Section 1.1
$\mathbf{p}$	Polynomial degree	Section 2.2
$B_{a,\mathbf{p}}$	A Bernstein basis function	Section 2.2
$\overline{\Omega}$	Parent element domain	Section 2.2
$\boldsymbol{\xi}$	A coordinate in $\overline{\Omega}$	Section 2.2
I	A global index knot vector	Section 3.1
i_j	An index knot value	Section 3.1
G	A global parametric knot vector	Section 3.1
s_{i_j}	A parametric knot value	Section 3.1
$\hat{\Omega}$	Global index domain	Section 3.1
$\hat{\Omega}$	Global parametric domain	Section 3.1
L_{i_j}	A local knot vector	Section 3.2
$\mathbf{V}$	A vertex set	Section 3.3
$\mathbf{C}$	A cell set	Section 3.3
$\mathbf{V}(\mathbf{c})$	The vertex set of a cell	Section 3.3
$\hat{\Omega}(\mathbf{c})$	The index domain of a cell	Section 3.3
$\hat{\Omega}(\mathbf{c})$	The parametric domain of a cell	Section 3.3

(*continued on next page*)

Table A.1 (*continued*)

Symbol	Description	Section
$\mathbf{PC}$	Non-zero area parametric cells	Section 3.3
$\mathbf{F}$	An anchor set	Section 3.3
$\mathbf{V}(\mathbf{f})$	The vertex set of an anchor	Section 3.3
$\mathbf{C}(\mathbf{f})$	The cell set of an anchor	Section 3.3
$\hat{\Omega}(\mathbf{f})$	The index domain of an anchor	Section 3.3
$\hat{\Omega}(\mathbf{f})$	The parametric domain of an anchor	Section 3.3
$N_{\mathbf{f},\mathbf{p}}$	A B-spline basis function	Section 3.4
$\mathbf{s}$	A coordinate in $\hat{\Omega}$	Section 3.4
$\mathcal{B}$	A B-spline space	Section 3.4
N	A set of B-spline basis functions	Section 3.4
$w_{\mathbf{f}}$	A weight	Section 3.4
$R_{\mathbf{f},\mathbf{p}}$	A rational B-spline basis function	Section 3.4
$\mathbf{P}_{\mathbf{f}}$	A control point	Section 3.5
$\mathbf{AV}$	Active vertices	Section 3.6
$\mathbf{AC}$	Active cells	Section 3.6
$\mathbf{APC}$	Active parametric cells	Section 3.6
$\hat{\Omega}(\mathbf{AC})$	Active cell index domain	Section 3.6
$\hat{\Omega}(\mathbf{AC})$	Active cell parametric domain	Section 3.6
$\mathbf{AF}$	Active anchors	Section 3.6
$\mathbf{AN}$	Active B-spline basis functions	Section 3.6
$\hat{\Omega}(\mathbf{AF})$	Active function index domain	Section 3.6
$\hat{\Omega}(\mathbf{AF})$	Active function parametric domain	Section 3.6

Table A.2
Notational conventions used throughout the text and where they are defined.

Symbol	Description	Section
α	A hierarchical level index	Section 4.1
N	Number of hierarchical levels	Section 4.1
$\mathbb{H}$	A hierarchical topology	Section 4.1
$\mathbf{HF}$	The set of hierarchical anchors	Section 4.2
$\mathcal{H}$	A hierarchical space	Section 4.2
$\mathbf{HN}$	The set of hierarchical B-spline basis functions	Section 4.2
$\mathbf{GN}$	The set of geometric B-spline blending functions	Section 5
$\mathbf{GF}$	The set of geometric anchors	Section 5
$\mathbf{IE}$	The set of leaf elements in the index domain	Section 6.1
$\mathbf{PE}$	The set of leaf elements in the parametric domain	Section 6.1
$\mathbf{HN}(\mathbf{e}^{\alpha})$	Hierarchical basis functions supported by $\mathbf{e}^{\alpha}$	Section 6.1
$\mathbf{HF}(\mathbf{e}^{\alpha})$	The set of anchors for $\mathbf{HN}(\mathbf{e}^{\alpha})$	Section 6.1
$\mathbf{GN}(\mathbf{e}^{\alpha})$	Hierarchical geometric functions supported by $\mathbf{e}^{\alpha}$	Section 6.1
$\mathbf{GF}(\mathbf{e}^{\alpha})$	The set of anchors for $\mathbf{GN}(\mathbf{e}^{\alpha})$	Section 6.1
n	The balance parameter	Section 6.2
K	Number of tree subdomains in a forest	Section 7
k	A tree index	Section 7
f	A face index	Section 7.1
s	A side index	Section 7.1
c	A corner index	Section 7.1
fc	A face corner index	Section 7.1
sc	A side corner index	Section 7.1
r	Relative orientation between subdomains	Section 7.1.1
e	An element index	Section 8.1
n_{el}	Number of Bézier elements in the forest	Section 8.1
A	A global function index	Section 8.1
n_f	Number of basis functions in the forest	Section 8.1
G	A global geometry function index	Section 8.1
n_g	Number of geometric blending functions in the forest	Section 8.1
a	A local function index	Section 8.1
n_f^e	Number of basis functions supported by element e	Section 8.1
g	A local geometric function index	Section 8.1
n_g^e	Number of geometric blending functions supported by element e	Section 8.1
$\hat{\Omega}^e$	The parametric domain of element e	Section 8.1
Ω^e	The physical domain of element e	Section 8.1
$\mathbf{C}^e$	Element extraction operator	Section 8.2
F	A forest Bézier mesh	Section 8.2

References

[1] D.R. Forsey, R.H. Bartels, Hierarchical B-spline refinement, ACM SIGGRAPH Computer Graphics 22 (4) (1988) 205–212.
[2] A. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 200 (49–52) (2011) 3554–3567.
[3] C. Giannelli, B. Jüttler, H. Speleers, THB–splines: The truncated basis for hierarchical splines, Computer Aided Geometric Design 29 (7) (2012) 485–498.
[4] D. Schillinger, L. Dedé, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, T.J.R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Computer Methods in Applied Mechanics and Engineering 249–252 (2012) 116–150.
[5] D. Schillinger, J.A. Evans, A. Reali, M.A. Scott, T.J.R. Hughes, Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations, Computer Methods in Applied Mechanics and Engineering 267 (2013) 170–232.
[6] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements NURBS, exact geometry, and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135–4195.
[7] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric analysis: Toward Integration of CAD and FEA, Wiley, Chichester, 2009.
[8] I. Akkerman, Y. Bazilevs, V. Calo, T.J.R. Hughes, S. Hulshoff, The role of continuity in residual-based variational multiscale modeling of turbulence, Computational Mechanics 41 (2008) 371–378.
[9] Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer Methods in Applied Mechanics and Engineering 197 (2007) 173–201.
[10] Y. Bazilevs, I. Akkerman, Large eddy simulation of turbulent Taylor–Couette flow using isogeometric analysis and residual-based variational multiscale method, Journal of Computational Physics 229 (2010) 3402–3414.
[11] Y. Bazilevs, C. Michler, V.M. Calo, T.J.R. Hughes, Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes, Computer Methods in Applied Mechanics and Engineering 199 (13–16) (2010) 780–790.
[12] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J.R. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Computational Mechanics 38 (2006) 310–322.
[13] Y. Bazilevs, V.M. Calo, T.J.R. Hughes, Y. Zhang, Isogeometric fluid-structure interaction: theory, algorithms, and computations, Computational Mechanics 43 (2008) 3–37.
[14] Y. Zhang, Y. Bazilevs, S. Goswami, C. Bajaj, T.J.R. Hughes, Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow, Computer Methods in Applied Mechanics and Engineering 196 (2007) 2943–2959.
[15] Y. Bazilevs, J.R. Gohean, T.J.R. Hughes, R.D. Moser, Y. Zhang, Patient-specific isogeometric fluid-structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device, Computer Methods in Applied Mechanics and Engineering 198 (45–46) (2009) 3534–3550.
[16] F. Auricchio, L.B. da Veiga, C. Lovadina, A. Reali, The importance of the exact satisfaction of the incompressibility constraint in nonlinear elasticity: mixed FEMs versus NURBS-based approximations, Computer Methods in Applied Mechanics and Engineering 199 (2010) 314–323.
[17] F. Auricchio, L.B. da Veiga, A. Buffa, C. Lovadina, A. Reali, G. Sangalli, A fully locking-free isogeometric approach for plane linear elasticity problems: A stream function formulation, Computer Methods in Applied Mechanics and Engineering 197 (2007) 160–172.
[18] T. Elguedj, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, $\bar{B}$ and $\bar{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements, Computer Methods in Applied Mechanics and Engineering 197 (2008) 2732–2762.
[19] J.A. Cottrell, T.J.R. Hughes, A. Reali, Studies of refinement and continuity in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 196 (2007) 4160–4183.
[20] J.A. Cottrell, A. Reali, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of structural vibrations, Computer Methods in Applied Mechanics and Engineering 195 (2006) 5257–5296.
[21] 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, Computer Methods in Applied Mechanics and Engineering 254 (2013) 197–221.
[22] D.J. Benson, Y. Bazilevs, M.C. Hsu, T.J.R. Hughes, Isogeometric shell analysis: the Reissner–Mindlin shell, Computer Methods in Applied Mechanics and Engineering 199 (5–8) (2010) 276–289.
[23] D.J. Benson, Y. Bazilevs, M.C. Hsu, T.J.R. Hughes, A large deformation, rotation-free, isogeometric shell, International Journal for Numerical Methods in Engineering 200 (2011) 1367–1378.
[24] D.J. Benson, Y. Bazilevs, E. De Luycker, M.C. Hsu, M.A. Scott, T.J.R. Hughes, T. Belytschko, A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM, International Journal for Numerical Methods in Engineering 83 (2010) 765–785.
[25] R. Echter, M. Bischoff, Numerical efficiency, locking and unlocking of NURBS finite elements, Computer Methods in Applied Mechanics and Engineering 199 (5–8) (2010) 374–382.
[26] J. Kiendl, Y. Bazilevs, M.C. Hsu, R. Wuechner, K.U. Bletzinger, The bending strip method for isogeometric analysis of Kirchhoff–Love shell structures comprised of multiple patches, Computer Methods in Applied Mechanics and Engineering 199 (37–40) (2010) 2403–2416.
[27] C.V. Verhoosel, M.A. Scott, T.J.R. Hughes, R. de Borst, An isogeometric analysis approach to gradient damage models, International Journal for Numerical Methods in Engineering 86 (2011) 115–134.
[28] C.V. Verhoosel, M.A. Scott, R. de Borst, T.J.R. Hughes, An isogeometric approach to cohesive zone modeling, International Journal for Numerical Methods in Engineering 87 (2011) 336–360.
[29] M.J. Borden, M.A. Scott, C.V. Verhoosel, C.M. Landis, T.J.R. Hughes, A phase-field description of dynamic brittle fracture, Computer Methods in Applied Mechanics and Engineering 217 (2012) 77–95.
[30] H. Gomez, V.M. Calo, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of the Cahn–Hilliard phase-field model, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4333–4352.
[31] H. Gomez, T.J.R. Hughes, X. Nogueira, V.M. Calo, Isogeometric analysis of the isothermal Navier–Stokes–Korteweg equations, Computer Methods in Applied Mechanics and Engineering 199 (25–28) (2010) 1828–1840.
[32] S. Lipton, J.A. Evans, Y. Bazilevs, T. Elguedj, T.J.R. Hughes, Robustness of isogeometric structural discretizations under severe mesh distortion, Computer Methods in Applied Mechanics and Engineering 199 (5–8) (2010) 357–373.
[33] W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization, Computer Methods in Applied Mechanics and Engineering 197 (2008) 2976–2988.
[34] X. Qian, Full analytical sensitivities in NURBS based isogeometric shape optimization, Computer Methods in Applied Mechanics and Engineering 199 (29–32) (2010) 2059–2071.
[35] A.P. Nagy, M.M. Abdalla, Z. Gurdal, Isogeometric sizing and shape optimization of beam structures, Computer Methods in Applied Mechanics and Engineering 199 (17–20) (2010) 1216–1230.
[36] A.P. Nagy, M.M. Abdalla, Z. Gurdal, On the variational formulation of stress constraints in isogeometric design, Computer Methods in Applied Mechanics and Engineering 199 (41–44) (2010) 2687–2696.
[37] R.N. Simpson, M.A. Scott, M. Taus, D. Thomas, H. Lian, Acoustic isogeometric boudary element analysis, Computer Methods in Applied Mechanics and Engineering (in press).
[38] A. Buffa, G. Sangalli, R. Vázquez, Isogeometric analysis in electromagnetics: B-splines approximation, Computer Methods in Applied Mechanics and Engineering 199 (17–20) (2010) 1143–1152.
[39] E. Cohen, T. Martin, R.M. Kirby, T. Lyche, R.F. Riesenfeld, Analysis-aware modeling: understanding quality considerations in modeling for isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 199 (5–8) (2010) 334–356.

[40] J. Lu, Circular element: isogeometric elements of smooth boundary, Computer Methods in Applied Mechanics and Engineering 198 (30–32) (2009) 2391–2402.
[41] H. Kim, Y. Seo, S. Youn, Isogeometric analysis for trimmed CAD surfaces, Computer Methods in Applied Mechanics and Engineering 198 (37–40) (2009) 2982–2995.
[42] P. Costantini, C. Manni, F. Pelosi, M.L. Sampoli, Quasi-interpolation in isogeometric analysis based on generalized B-splines, Computer Aided Geometric Design 27 (8) (2010) 656–668.
[43] M. Aigner, C. Heinrich, B. Juettler, E. Pilgerstorfer, B. Simeon, A.V. Vuong, Swept volume parameterization for isogeometric analysis, in: E.R. Hancock, R.R. Martin, M.A. Sabin (Eds.), Mathematics of Surfaces XIII, Lecture Notes in Computer Science, vol. 5654, 2009, pp. 19–44.
[44] T. Martin, E. Cohen, R.M. Kirby, Volumetric parameterization and trivariate B-spline fitting using harmonic functions, Computer Aided Geometric Design 26 (6) (2009) 648–664.
[45] W. Wang, Y. Zhang, Wavelets-based NURBS simplification and fairing, Computer Methods in Applied Mechanics and Engineering 199 (5–8) (2010) 290–300.
[46] W. Wang, Y. Zhang, M.A. Scott, T.J.R. Hughes, Converting an unstructured quadrilateral mesh to a standard T-spline surface, Computational Mechanics 48 (2011) 477–498.
[47] C. Burstedde, L.C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33 (3) (2011) 1103–1133.
[48] W. Bangerth, C. Burstedde, T. Heister, M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, ACM Transactions on Mathematical Software 38 (2) (2011) 14:1–14:28.
[49] T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, ACM Transactions on Graphics 22 (2003) 477–484.
[50] M.A. Scott, X. Li, T.W. Sederberg, T.J.R. Hughes, Local refinement of analysis-suitable T-splines, Computer Methods in Applied Mechanics and Engineering 213 (2012) 206–222.
[51] L.B. da Veiga, A. Buffa, G. Sangalli, R. Vázquez, Analysis-suitable T-splines of arbitrary degree: definition, linear independence, and approximation properties, Mathematical Models and Methods in Applied Sciences 23 (11) (2013). in press.
[52] X. Li, M.A. Scott, Analysis-suitable T-splines: characterization, refineability, and approximation, Mathematical Models and Methods in Applied Science (in press).
[53] L. Beirao da Veiga, A. Buffa, D. Cho, G. Sangalli, Analysis-suitable T-splines are dual-compatible, Computer Methods in Applied Mechanics and Engineering 249–252 (2012) 42–51.
[54] X. Li, J. Zheng, T.W. Sederberg, T.J.R. Hughes, M.A. Scott, On linear independence of T-spline blending functions, Computer Aided Geometric Design 29 (2012) 63–76.
[55] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, T.W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5–8) (2010) 229–263.
[56] M.A. Scott, M.J. Borden, C.V. Verhoosel, T.W. Sederberg, T.J.R. Hughes, Isogeometric finite element data structures based on Bézier extraction of T-splines, International Journal for Numerical Methods in Engineering 88 (2011) 126–156.
[57] M.J. Borden, M.A. Scott, J.A. Evans, T.J.R. Hughes, Isogeometric finite element data structures based on Bézier extraction of NURBS, International Journal for Numerical Methods in Engineering 87 (2011) 15–47.
[58] G.E. Farin, NURBS Curves and Surfaces: from projective geometry to practical use, A.K. Peters Ltd., Natick, MA, 1999.
[59] G.E. Farin, Curves and Surfaces for CAGD, A Practical Guide, fifth ed., Morgan Kaufmann Publishers, San Francisco, 1999.
[60] T.W. Sederberg, Computer Aided Geometric Design: Course Notes. <http://www.tsplines.com/educationportal.html>, 2010
[61] L. Ramshaw, Blossoms are polar forms, Computer Aided Geometric Design 6 (4) (1989) 323–358.
[62] C. Giannelli, B. Jüttler, Bases and dimensions of bivariate hierarchical tensor-product splines, Journal of Computational and Applied Mathematics 239 (2013) 162–178.
[63] T.J.R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover Publications, Mineola, NY, 2000.
[64] A.N. Brooks, T.J.R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 32 (1982) 199–259.
[65] J. Chung, G.M. Hulbert, A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method, Journal of Applied Mechanics 60 (1993) 371–375.
[66] K.E. Jansen, C.H. Whiting, G.M. Hulbert, A generalized-α method for integrating the filtered Navier-Stokes equations with a stabilized finite element method, Computer Methods in Applied Mechanics and Engineering 190 (1999) 305–319.
[67] T.J.R. Hughes, Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods, Computer Methods in Applied Mechanics and Engineering 127 (1995) 387–401.
[68] M.G. Larson, A. Målqvist, Adaptive variational multiscale methods based on a posteriori error estimation: Duality techniques for elliptic problems, in: B. Engquist, O. Runborg, P. Lötstedt (Eds.), Multiscale Methods in Science and Engineering, Lecture Notes in Computational Science and Engineering, vol. 44, Springer, Berlin Heidelberg, 2005, pp. 181–193.
[69] G. Hauke, M.H. Doweidar, M. Miana, The multiscale approach to error estimation and adaptivity, Computer Methods in Applied Mechanics and Engineering 195 (13–16) (2006) 1573–1593.
[70] G. Hauke, M.H. Doweidar, S. Fuentes, Mesh adaptivity for the transport equation led by variational multiscale error estimators, International Journal for Numerical Methods in Fluids 69 (12) (2012) 1835–1850.
[71] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal of Scientific and Statistical Computing 7 (1986) 856–869.
[72] M. Heroux, R. Bartlett, V.H.R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, An overview of trilinos, Tech. Rep. SAND2003-2927, Sandia National Laboratories, 2003.
[73] J.A. Evans, Y. Bazilevs, I. Babuška, T.J.R. Hughes, n-Widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method, Computer Methods in Applied Mechanics and Engineering 198 (21–26) (2009) 1726–1741.
[74] T. Takacs, B. Jüttler, H^2 regularity properties of singular parameterizations in isogeometric analysis, Graphical models 74 (6) (2012) 361–372.
[75] T. Takacs, B. Jüttler, Existence of stiffness matrix integrals for singularly parameterized domains in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 200 (2011) 3568–3582.
[76] Autodesk, Autodesk T-Splines Plug-in for Rhino user manual, Autodesk, 2012.