A subdivision-based implementation of the hierarchical b-spline finite element method

P.B. Bornemann, F. Cirak[*]

Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, U.K.

Abstract

A novel technique is presented to facilitate the implementation of hierarchical b-splines and their interfacing with conventional finite element implementations. The discrete interpretation of the two-scale relation, as common in subdivision schemes, is used to establish algebraic relations between the basis functions and their coefficients on different levels of the hierarchical b-spline basis. The subdivision projection technique introduced allows us first to compute all element matrices and vectors using a fixed number of same-level basis functions. Their subsequent multiplication with subdivision matrices projects them, during the assembly stage, to the correct levels of the hierarchical b-spline basis. The proposed technique is applied to convergence studies of linear and geometrically nonlinear problems in one, two and three space dimensions.

Keywords: Finite elements, hierarchical b-splines, subdivision schemes, isogeometric analysis

1. Introduction

In finite element analysis it is essential to adapt the spatial resolution and polynomial degree of the basis functions to the solution field. Often the resolution of the given CAD geometry differs from the spatial resolution requirements of the analysis model. The analysis mesh needs to be refined at locations where the solution field has high gradients and, by contrast, the resolution of the geometry model is determined by the size and location of the smallest geometric features. In order to aid the implementation of adaptive analysis tools it is necessary to have a unique mapping between the geometry and finite element models. To this end isogeometric analysis using b-splines and related basis functions provides a bidirectional mapping between the geometry and analysis models and holds the promise to provide a comprehensive solution to adaptive analysis [1, 2].

In computer graphics and geometric modelling there are a host of techniques available to adapt locally, i.e. refine or coarsen the spatial resolution of spline surfaces. Most of these techniques rely on the refinability property of b-splines according to which the spatial resolution of a spline surface can be increased without altering its geometry. Refinability is crucial to a number of multiresolution editing techniques, such as knot insertion [3], subdivision curves and surfaces [4–6], t-splines [7], hierarchical refinement [8] and wavelets [9, 10]. In algebraic terms, refinability enables the presentation of the b-splines defined on a coarse knot sequence as a linear combination of b-splines defined on a finer knot sequence. Thus the refinability property of b-splines is sometimes also referred to as the two-scale relation. In the context of multiresolution mesh editing refinability is used locally to replace or overlay selected coarse b-splines with finer b-splines [8]. The refined b-spline geometry is able exactly to reproduce the coarse one and beyond that geometric details can be added as necessary.

The common adaptive b-spline refinement techniques available in computer graphics and geometric modelling cannot be directly employed in adaptive finite element analysis. Among others, these techniques usually do not lead to linearly independent basis functions, which is important for the finite element formalism. Recently, two b-spline techniques originating in computer graphics, namely t-splines and hierarchical b-splines, have been extended to finite element analysis [11–13]. In t-splines local refinement is performed by inserting one or more knots into an existing tensor product b-spline patch. This is followed by a second step in which additional knots are introduced so that all b-splines have the complete set of knot intervals. Hierarchical b-spline refinement is performed by overlaying a coarse tensor product patch with smaller and finer tensor product patches. In both refinement approaches certain rules have to be followed so that the resulting basis functions are linearly independent and retain their polynomial reproduction properties and smoothness. In terms of software implementation these rules lead to conceptually straightforward but intricate algorithms. An additional source of complexity is the basis function focus of b-spline refinement techniques and the element focus of conventional finite element software.

In this paper we make use of concepts from subdivision schemes to ease the implementation of hierarchical b-splines and to simplify their integration with existing finite element software. As known, subdivision schemes for curves and surfaces provide an alternative viewpoint to b-splines [5, 6, 14, 15]. They create a smooth curve or surface starting from a coarse control mesh by repeated refinement and averaging, which converges to a b-spline curve or surface for certain choices of averaging weights. In computer graphics applications subdivision

[*]Corresponding author
Email address: `f.cirak@eng.cam.ac.uk` (F. Cirak)

schemes are often used as simple mesh refinement algorithms for creating sufficiently fine faceted presentations of smooth surfaces. As we shall demonstrate, this algorithmic mesh-based view of b-splines can greatly ease the implementation of hierarchical b-splines. More specifically it makes it straightforward to establish algebraic relations between the basis functions and their coefficients defined on different refinement levels of the mesh. For instance, these relations simplify the evaluation of the element matrices and vectors which depend on basis functions defined over several levels. In the present approach element matrices and vectors are all evaluated at the finest available level and are subsequently projected to the hierarchical b-spline space during the finite element assembly stage. This is accomplished by multiplying the element matrices and vectors with subdivision matrices. Note that the mapping of coefficients between different hierarchical b-spline levels is also relevant during the adaptive solution of nonlinear and history-dependent problems.

The outline of this paper is as follows. Section 2 begins with a brief review of univariate and multivariate b-splines and summarises their properties relevant for subsequent discussions. In Section 3 the hierarchical b-splines as defined by Kraft [16] are introduced. Subsequently, in Section 4 the hierarchical b-splines are used for finite element discretisation of boundary value problems. The subdivision projection technique is introduced as an effective means of dealing with the complexities of hierarchical b-splines and for interfacing them with conventional finite elements. Adaptive finite element analysis based on hierarchical b-splines is applied to different linear and geometrically nonlinear examples in Section 5.

2. Review of b-splines and subdivision

In following we briefly review the definition of a few important properties of uniform b-splines. For further details on b-splines we refer to standard textbooks, e.g., [3, 17, 18].

2.1. Univariate b-spline basis functions and their refinability

We consider the uniform b-spline basis functions over a parameter space with equidistantly spaced knots $\xi_i = 0, 1, 2, 3, \ldots$. The corresponding b-spline basis functions B_i^μ of degree μ are defined with the recursive averaging formula

$$\begin{aligned} B_i^0(\xi) &= \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ B_i^\mu(\xi) &= \frac{\xi - \xi_i}{\xi_{i+\mu} - \xi_i} B_i^{\mu-1}(\xi) + \frac{\xi_{i+\mu+1} - \xi}{\xi_{i+\mu+1} - \xi_{i+1}} B_{i+1}^{\mu-1}(\xi) \end{aligned} \tag{1}$$

Without going into detail, a b-spline of degree μ is non-zero over an interval $\xi_i \leq \xi < \xi_{i+\mu+1}$. The uniform b-splines of degree μ computed with (1) are the shifted (or, translated) instances of each other, i.e. $B_i^\mu(\xi) = B_0^\mu(\xi - i)$. At the top half of Figure 1 translates of a few quadratic b-splines are shown and the indexing (labelling) scheme implied by (1) is indicated. The index of a b-spline B_i^μ is the same as the index of the knot located at its support's lower boundary $\xi_i = i$. As will become apparent later, the present labelling scheme allows us to consider odd and even degree b-splines within the same algorithmic framework.

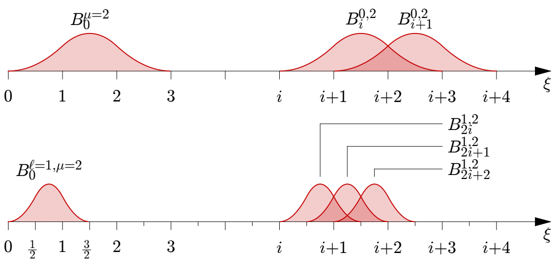


Figure 1: Quadratic b-splines $B_i^{\ell,\mu}$ on levels $\ell = 0$ and $\ell = 1$ obtained by translation and scaling of $B_0^{\mu=2}$.

In addition to the knot sequence considered so far we introduce an ℓ-times bisected knot sequence $\xi_i^\ell = 0/2^\ell, 1/2^\ell, 2/2^\ell, 3/2^\ell, 4/2^\ell, \ldots$ with level $\ell \geq 0$. This means that the index i of a particular knot $\xi_i^0 = i$ on the initial knot sequence with $\ell = 0$ becomes index $2^\ell i$ at level ℓ, that is $\xi_{2^\ell i}^\ell = \xi_i^0$. A scaling relation holds between the basis functions on the initial knot sequence with $\ell = 0$ and the ℓ-times bisected knot sequence

$$B_i^{\ell,\mu}(\xi) = B_0^\mu(2^\ell \xi - i) \tag{2}$$

where the factor 2^ℓ in the argument scales the support size of a basis function at level ℓ with respect to the initial support size.

B-spline basis functions are refinable in the sense that the b-splines $B_i^{\ell,\mu}(\xi)$ on level ℓ can be represented as a linear combination of $B_j^{\ell+1,\mu}(\xi)$ on level $\ell + 1$

$$B_i^{\ell,\mu}(\xi) = \sum_k S_{i,k}^\mu B_{2i+k}^{\ell+1,\mu}(\xi) \quad \text{with} \quad S_{i,k}^\mu = \frac{1}{2^\mu}\binom{\mu+1}{k} \tag{3}$$

Here, $S_{i,k}^\mu$ is the subdivision matrix and its entries, the subdivision weights, are given in terms of the usual binomial coefficients [4, 5, 19]. The subdivision matrix is a banded sparse matrix with identical rows and columns. Its components are independent of the knot index i and the level ℓ, but depend on the polynomial degree μ of the considered b-spline. The $\mu + 2$ b-splines $B_{2i+k}^{\ell+1,\mu}$ on the refined knot sequence reproducing the b-spline $B_i^{\ell,\mu}$ are referred to as the children of $B_i^{\ell,\mu}$. See Figure 2 for an illustration of the refinement relation in case of quadratic b-splines.

At times it is more convenient to write (3) in matrix notation

$$\boldsymbol{B}^{\ell,\mu} = \boldsymbol{S}^\mu \boldsymbol{B}^{\ell+1,\mu} \tag{4}$$

2.2. Subdivision refinement of univariate splines

A spline on level ℓ is defined as the product of the basis functions B_i^ℓ with the coefficients u_i^ℓ

$$u(\xi) = \sum_i B_i^\ell(\xi)\, u_i^\ell = \boldsymbol{B}^\ell(\xi) \cdot \boldsymbol{u}^\ell \tag{5}$$

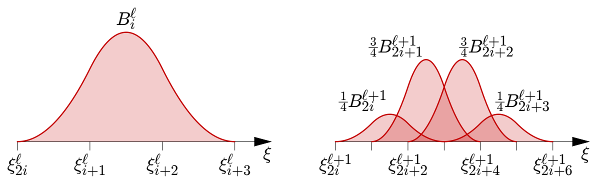


Figure 2: The quadratic b-spline $B_i^{\ell,2}$ on the left can be represented as a linear combination of its four children $B_{2i+k}^{\ell+1,2}$ on the right. In the figure the index 2 indicating the polynomial degree has been omitted for clarity.

To ease the notation we omitted here and ocasionally in the following the superscript μ indicating the polynomial degree. Introducing the refinement relation (4) into (5), the spline can be represented on the next, finer level $\ell + 1$

$$u(\xi) = \left(\boldsymbol{S}\boldsymbol{B}^{\ell+1}\right) \cdot \boldsymbol{u}^\ell = \boldsymbol{B}^{\ell+1} \cdot \left(\boldsymbol{S}^\top \boldsymbol{u}^\ell\right) \tag{6}$$

from which it follows that the coefficients of the refined level are given with the subdivision relation

$$\boldsymbol{u}^{\ell+1} = \boldsymbol{S}^\top \boldsymbol{u}^\ell \tag{7}$$

It is worth emphasising that the coarse level b-splines $\boldsymbol{B}^\ell$ with the coefficients $\boldsymbol{u}^\ell$ and the fine level b-splines $\boldsymbol{B}^{\ell+1}$ with the coefficients $\boldsymbol{u}^{\ell+1}$ represent exactly the same spline. Of course, the b-splines on level $\ell+1$ define a larger function space and are able to represent splines with finer details if the coefficients $\boldsymbol{u}^{\ell+1}$ are not constrained by (7).

In subdivision schemes (7) is recursively used to create increasingly finer presentations of a given set of coefficients. As can be shown, the coefficients generated with (7) converge in the limit of infinite refinement to the true b-spline associated with the initial coarse knot sequence. Furthermore, in subdivision literature the algebraic relation (7) is usually expressed graphically in form of stencils.

2.3. *Tensor product b-splines*

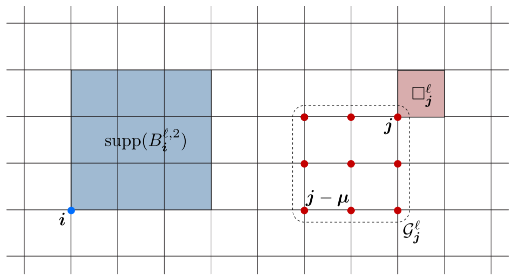
Multivariate uniform b-splines are defined with the tensor product formalism. The corresponding parameter space is an in-all-directions equally spaced grid. Similar to the univariate case, the knots on this d-variate grid are labelled with multi-indices $\boldsymbol{i} = (i^1, i^2, \dots, i^d)$ and have the parametric coordinates $\boldsymbol{\xi}_{\boldsymbol{i}} = (\xi^1_{i^1}, \dots, \xi^d_{i^d})$. The cells of the grid are labelled with multi-indices as well

$$\square_{\boldsymbol{i}}^\ell = \frac{1}{2^\ell}([i^1, i^1+1) \times \dots \times [i^d, i^d+1)) \tag{8}$$

where $\ell \geq 0$ is the refinement level of the grid. According to (8), a cell has the index $\boldsymbol{i}$ and the knot $\boldsymbol{\xi}_{\boldsymbol{i}}^\ell = \boldsymbol{i}/2^\ell$ at its lower cell corner, see Figure 3. The multivariate b-splines are the tensor products of univariate b-splines

$$B_{\boldsymbol{i}}^{\ell,\mu}(\boldsymbol{\xi}) = B_{i^1}^{\ell,\mu}(\xi^1) \times \dots \times B_{i^d}^{\ell,\mu}(\xi^d) \tag{9}$$

Here and in the following it is assumed that each of the b-splines in the d coordinate directions has the same polynomial degree μ.



$\mathrm{supp}(B_{\boldsymbol{i}}^{\ell,2})$
$\square_{\boldsymbol{j}}^\ell$
$\boldsymbol{j}$
$\boldsymbol{j}-\boldsymbol{\mu}$
$\boldsymbol{i}$
$\mathcal{G}_{\boldsymbol{j}}^\ell$

Figure 3: Support cells of a biquadratic b-spline (left) and the biquadratic non-zero b-splines over a cell (right). The lower corner of the b-spline support and the cell are indicated by a dot.

In analogy to the univariate case, the knot intervals covered by the support of a multivariate b-spline are

$$\mathrm{supp}\, B_{\boldsymbol{i}}^{\ell,\mu} = \frac{1}{2^\ell}([i^1, i^1+\mu+1) \times \dots \times [i^d, i^d+\mu+1)) \tag{10}$$

which is comprised of $(\mu+1)^d$ cells

$$\mathrm{supp}\, B_{\boldsymbol{i}}^{\ell,\mu} = \bigcup_{\boldsymbol{j}=\boldsymbol{i}}^{\boldsymbol{i}+\boldsymbol{\mu}+1} \square_{\boldsymbol{j}}^\ell$$

As an example, in Figure 3 the support of a biquadratic b-spline is shown.

Lastly, in preparation for the finite element discretisation we define the set $\mathcal{G}_{\boldsymbol{j}}^\ell$ which collects the b-splines whose supports have non-empty intersection with the cell $\square_{\boldsymbol{j}}^\ell$ and, hence are non-zero in $\square_{\boldsymbol{j}}^\ell$,

$$\mathcal{G}_{\boldsymbol{j}}^\ell = \{\boldsymbol{i} \in \mathbb{Z}^d \,|\, \square_{\boldsymbol{j}}^\ell \subset \mathrm{supp}\, B_{\boldsymbol{i}}^{\ell,\mu}\} = \{\boldsymbol{j}-\boldsymbol{\mu}, \dots, \boldsymbol{j}\} \tag{11}$$

The b-splines $B_{\boldsymbol{g}}^\ell, \mu$ with $\boldsymbol{g} \in \mathcal{G}_{\boldsymbol{j}}^\ell$ are the (same-level) non-zero b-splines of the cell. In Figure 3 the nine non-zero biquadratic b-splines over the cell $\square_{\boldsymbol{j}}^\ell$ are indicated by dots. The offset location of the dots with respect to the cell is an artefact of the indexing scheme used.

2.4. *Subdivision refinement of tensor product splines*

A tensor product spline on level ℓ is defined as the linear combination of basis functions $B_{\boldsymbol{i}}^\ell$ and coefficients $u_{\boldsymbol{i}}^\ell$

$$u(\boldsymbol{\xi}) = \sum_{\boldsymbol{i}} B_{\boldsymbol{i}}^\ell(\boldsymbol{\xi})\, u_{\boldsymbol{i}}^\ell = \boldsymbol{B}^\ell(\boldsymbol{\xi}) \cdot \boldsymbol{u}^\ell \tag{12}$$

As discussed for univariate splines, the same spline can be represented on the next, finer level $\ell+1$, without loss of information

$$u(\boldsymbol{\xi}) = (\boldsymbol{S}\boldsymbol{B}^{\ell+1}) \cdot \boldsymbol{u}^\ell = \boldsymbol{B}^{\ell+1} \cdot (\boldsymbol{S}^\top \boldsymbol{u}^\ell)$$

The corresponding subdivision weights are constructed as the tensor products of univariate subdivision weights (3). Accordingly, the subdivision matrix $\boldsymbol{S}^\mu$ for the multivariate case is given by

$$\boldsymbol{S}^\mu = S_{\boldsymbol{i},\boldsymbol{k}}^\mu = S^\mu_{(i^1,\dots,i^d),(k^1,\dots,k^d)} = S^\mu_{i^1,k^1} \times \dots \times S^\mu_{i^d,k^d} \tag{13}$$

With this subdivision matrix the refinement relation for multivariate b-splines and the subdivision relation for their coefficients can be expressed in the same way as in the univariate case, see (4) and (7).

3. Hierarchical b-splines (hb-splines)

The hierarchical b-splines first introduced by Forsey et al. [8] allow local refinement of a given coarse b-spline patch by overlaying it with finer b-spline patches. The basis functions on different levels are simultaneously considered since the motivation for developing hierarchical b-splines was multiresolution mesh editing. Hence the resulting basis functions are by construction non-local and not always linearly independent. An alternative definition of hierarchical b-splines provided by Kraft [16] leads to basis functions better suited to finite element applications. More specifically, the basis functions are linearly independent and the support size of the basis functions is relatively small.

The hierarchical b-splines can be understood as a technique for locally enriching the approximation space by replacing selected coarse grid b-splines with fine grid b-splines [12, 13, 20]. To this end, it is crucial that the basis functions satisfy a refinement relationship as introduced in Section 2.4. The change of focus from element refinement (as in conventional finite elements) to basis refinement is crucial to local b-spline refinement. Building on the notion of basis refinement and Kraft's original algorithm, we introduce in this section the hb-splines which are constructed in three steps, referred to as the *refinement*, *compilation* and *shrinkage*. In short, *refinement* introduces new b-splines at a finer level; *compilation* assures linear independence by removing basis functions; and *shrinkage* discards b-splines at the boundaries with a support not intersecting the effective domain. As will be specified later the effective domain is the subdomain on which the coarse b-splines on level $\ell = 0$ form a partition of unity.

3.1. Hierarchical univariate b-splines

To begin with we consider the local refinement of univariate b-splines defined over a one-dimensional domain. The presented approach and definitions closely follow Kraft [16]. All the steps are illustrated with the help of a locally refined quadratic b-spline example.

Our starting point is a set of b-spline basis functions whose support is fully contained in a parameter domain $\square$ with equidistantly spaced knots. For simplicity it is assumed that $\square$ is a connected domain. These b-splines and the domain represent the coarse grid to be refined; they are referred to as the 0-th level b-splines and initial domain.

$$\square := \square^0 = \bigcup_{i \in \mathcal{I}^0} \operatorname{supp} B_i^0 \tag{14}$$

where $\mathcal{I}^0$ is the index set containing all the considered b-splines. Figure 4 shows a sample initial domain with $|\mathcal{I}^0| = 8$ quadratic b-splines over the knot interval $\square^0 = [0, |\mathcal{I}^0| + \mu)$.

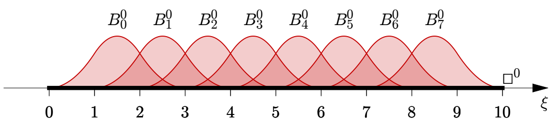


Figure 4: Initial domain for quadratic b-splines.

Refinement step. The approximation space formed by the b-splines in $\mathcal{I}^0$ can be refined by replacing selected b-splines B_i^0 with b-splines B_i^1. The b-splines B_i^1 on level 1 are defined on a grid with cells half as large as those on the initial domain $\square^0$. In this step we use the refinement relation introduced in Section 2.1. Specifically, the Kraft algorithm requires that, if a certain b-spline B_i^0 is to be refined, all its children on level 1 have to be introduced, as stipulated by the refinement relation (3). The union of the supports of b-splines on level 1 constitutes the domain

$$\square^1 = \bigcup_{i \in \mathcal{I}^1} \operatorname{supp} B_i^1 \tag{15}$$

Note that in general $\square^1$ need not be a connected domain. However, the subdomain $\square^1$ can always, by construction, be fully covered with the supports of b-splines on level 0, because all the children of each refined coarse b-spline B_i^0 are present on level 1. This implies for the domain on level 1 that

$$\square^1 = \bigcup_{j \in \mathcal{I}^1} \operatorname{supp} B_j^1 = \bigcup_{j \in \mathcal{M}^0} \operatorname{supp} B_j^\ell \tag{16}$$

where $\mathcal{M}^0$ is the index set containing all the refined b-splines. In the Kraft algorithm the subdomains are not allowed to intersect with the boundary of the coarser, enclosing domain, i.e.

$$\partial \square^0 \cap \partial \square^1 = \varnothing \tag{17}$$

This means that there is always one cell distance between the boundary of the coarse and fine grid.

In practice the refinement step is applied recursively so that it leads to a sequence of successively refined domains and b-splines. The refinement of a domain at level ℓ is carried out in the same way as the refinement of the initial domain at level 0 described above. The non-intersection requirement (17) has now to be satisfied between domains on two consecutive refinement levels

$$\partial \square^\ell \cap \partial \square^{\ell+1} = \varnothing \tag{18}$$

The recursive application of the refinement step combined with the non-intersection requirement leads to a sequence of nested refined domains

$$\square := \square^0 \supset \square^1 \supset \square^2 \supset \ldots \supset \square^{\ell_{\max}} \tag{19}$$

where $\ell_{\max}$ is the global maximum refinement level or global level width reached during the recursive refinement.

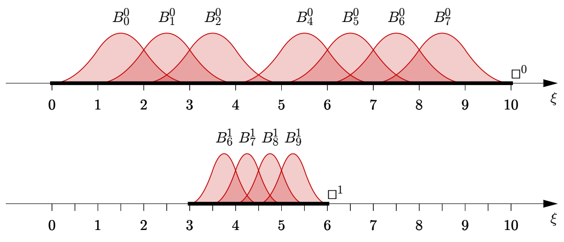


Figure 5: First refinement of quadratic b-splines (after compilation).

Compilation step. The set of b-splines created in the refinement step is, in general, not linearly independent[1] and hence not suitable for finite element purposes. The aim of the compilation step is to assemble a linearly independent basis by systematically adding and discarding redundant b-splines.

To this end we first assume that on each domain $\square^\ell$, with $\ell = 0, \ldots, \ell_{\max}$, all the b-splines with $\operatorname{supp} B_i^\ell \subset \square^\ell$ are present. Subsequently, starting from level 0 and proceeding level-by-level, we discard any coarse b-spline B_i^ℓ that can be presented as a linear combination of existing fine b-splines $B_i^{\ell+1}$. This is accomplished by checking wether all the children of the coarse b-spline B_i^ℓ (as stipulated by the refinement relation) are present on the next finer level $\ell + 1$. Evidently this is equivalent to checking that $\operatorname{supp} B_i^\ell \subset \square^{\ell+1}$. The compilation terminates when level $\ell_{\max} - 1$ is reached.

After discarding the redundant b-splines those remaining on each level are collected into index sets

$$\mathcal{D}^\ell = \{ i \in \mathbb{Z} \mid \operatorname{supp} B_i^\ell \subseteq \square^\ell \text{ and } \operatorname{supp} B_i^\ell \nsubseteq \square^{\ell+1} \} \qquad (20)$$

The union of the b-splines in all index sets $\mathcal{D}^\ell$ establish the hierarchical b-spline basis

$$\mathcal{B}_{\text{hb}}(\square) := \operatorname{span}\{ B_k^\ell \mid 0 \le \ell \le \ell_{\max} \text{ and } k \in \mathcal{D}^\ell \} \qquad (21)$$

As proven by Kraft [16], the hierarchical b-spline basis $\mathcal{B}_{\text{hb}}$ is linearly independent on the domain $\square$. In this basis the sum of the b-splines in certain cells can be different from one. However, the basis is able to reproduce, except close to the boundaries of the initial domain $\square^0$, polynomials up to the b-spline degree.

Illustrative example. Continuing with the quadratic b-spline example shown in Figure 4, we describe step-by-step its refinement with the technique described above.

Firstly, we refine a single b-spline B_3^0. In the refinement step, its four children on level 1, namely B_6^1, B_7^1, B_8^1 and B_9^1, are introduced. The union of their supports is the subdomain $\square^1 = [3, 6)$. Recall, a quadratic b-spline has four children due to the refinement relation introduced in Section 2.1 (c.f. also Fig. 2). The result of the refinement step is shown in Figure 5. In the compilation step only the b-spline B_3^0 is discarded because $\operatorname{supp} B_3^0 \subset \square^1$. The other b-splines on level 0 have to remain as they cannot be presented as a linear combination of the b-splines present on level 1. The index sets of the active b-splines on the two levels are $\mathcal{D}^0 = \{0, 1, 2, 4, \ldots, 7\}$ and $\mathcal{D}^1 = \{6, \ldots, 9\}$.

[1]A set of functions $\{B_i^\ell(\xi) \mid \boldsymbol{i} \in \mathcal{D}^\ell, \ell \geq 0\}$ is linearly independent for all $\xi \in \bigcup_{i \in \mathcal{D}^\ell, \ell \geq 0} \operatorname{supp} B_i^\ell$ if the equation $0 = \sum_{i \in \mathcal{D}^\ell, \ell \geq 0} u_i^\ell B_i^\ell(\xi)$ has only the (trivial) solution $u_i^\ell = 0$ for all $\boldsymbol{i} \in \mathcal{D}^\ell$, $\ell \geq 0$.

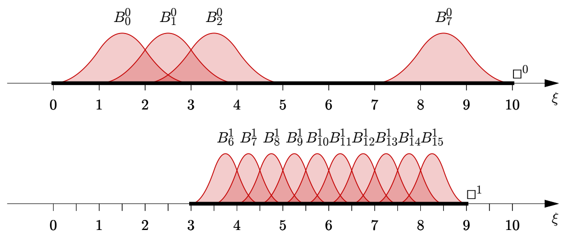


Figure 6: Second refinement of quadratic b-splines (after compilation).

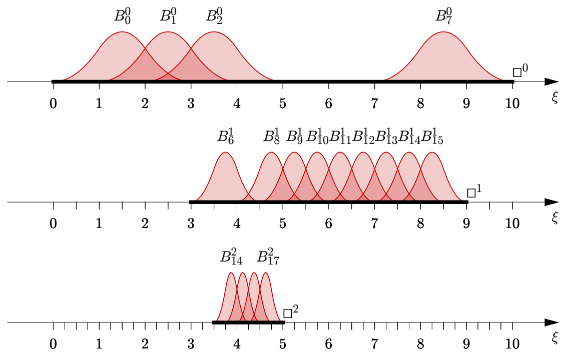


Figure 7: Third refinement of quadratic b-splines (after compilation).

Next, in addition to B_3^0 the b-spline B_6^0 is also refined. In the refinement step we introduce the four children B_{12}^1, B_{13}^1, B_{14}^1 and B_{15}^1 of B_6^0, see Figure 6. The union of all level 1 b-spline supports now becomes $\square^1 = [3, 9)$. In the compilation step we at first assume that all b-splines on $\square^0$ and $\square^1$ are present. Thus the set $\mathcal{D}^1 = \{6, \ldots, 15\}$ contains 10 b-splines, c.f. Figure 6, and not just the eight children introduced in the refinement step. After compilation the set of remaining active indices for the zero level is $\mathcal{D}^0 = \{0, 1, 2, 7\}$. Note that in addition to B_3^0 and B_6^0 also B_4^0 and B_5^0 are discarded because $\operatorname{supp} B_4^0 \subset \square^1$ and $\operatorname{supp} B_5^0 \subset \square^1$.

Finally, we refine B_7^1 on level 1. In the refinement step its four children B_{14}^2, B_{15}^2, B_{16}^2 and B_{17}^2 are introduced, see Figure 7. After the compilation we obtain the set of active indices $\mathcal{D}^0 = \{0, 1, 2, 7\}$, $\mathcal{D}^1 = \{6, 8, \ldots, 15\}$ and $\mathcal{D}^2 = \{14, \ldots, 17\}$.

3.2. Hierarchical tensor product b-splines

It is straightforward to generalise the univariate hierarchical b-splines to the tensor product case. To this end it is helpful to remember that the key idea in hierarchical b-spline refinement is the replacement of coarse grid b-splines with fine grid b-splines. Starting from this premise, tensor product b-splines can be locally refined by following the algorithm introduced in Section 3.1.

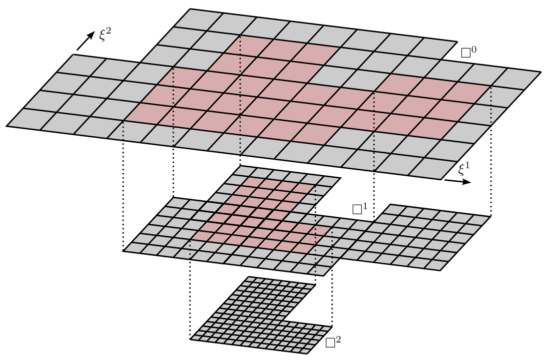


Figure 8: Nested domains over three refinement levels with quadratic hb-splines.

Evidently, in generalising the univariate refinement algorithm, we have to take into account the change of dimension of the support and the number of children. As introduced in Section 2.3, the number of children of a d-variate b-spline $B_{\boldsymbol{i}}^{\ell}(\boldsymbol{\xi})$ is $(\mu+2)^d$, where μ is its polynomial degree, and $\operatorname{supp} B_{\boldsymbol{i}}^{\ell}(\boldsymbol{\xi})$ is a d-dimensional domain. Recall that the index $\boldsymbol{i}$ is a multi-index $\boldsymbol{i} = (i^1, i^2, \dots, i^d)$. In the multivariate case the definition of the subdomains (16) becomes

$$\square^{\ell} = \bigcup_{\boldsymbol{i}\in\mathcal{I}^{\ell}} \operatorname{supp} B_{\boldsymbol{i}}^{\ell} \tag{22}$$

where $\mathcal{I}^{\ell}$ is the index set of the b-splines on level ℓ. As previously stated, for hierarchical refinement these domains have to be non-intersecting and nested, c.f. (18) and (19), and see Figure 8. According to (20) the index set of active b-splines on each level is

$$\mathcal{D}^{\ell} = \{\boldsymbol{i}\in\mathbb{Z}^d \,|\, \operatorname{supp} B_{\boldsymbol{i}}^{\ell} \subseteq \square^{\ell} \text{ and } \operatorname{supp} B_{\boldsymbol{i}}^{\ell} \nsubseteq \square^{\ell+1}\} \tag{23}$$

As in the univariate case, the union of active b-splines over all levels establishes the linear hierarchical tensor product b-spline basis

$$\mathcal{B}_{\text{hb}}(\square) := \operatorname{span}\{B_{\boldsymbol{k}}^{\ell} \,|\, 0 \le \ell < \ell_{\max} \text{ and } \boldsymbol{k}\in\mathcal{D}^{\ell}\} \tag{24}$$

Illustrative example. As a continuation of the univariate example in the foregoing section, a two-dimensional example is considered to highlight the refinement and compilation steps in the multi-variate case, see Figure 9. In this and subsequent figures each b-spline is indicated by a dot at the lower corner of its support.

Firstly, we refine the b-spline $B^0_{(3,1)}$, which corresponds to the missing dot on level 0 in Figure 9. Its support, consisting of 3×3 cells, is the highlighted subdomain in $\square^0$. In the refinement step the 16 children of $B^0_{(3,1)}$ on level 1 are introduced. The union of their supports is $\square^1 = [3,6)\times[1,4)$. In the compilation step only the b-spline $B^0_{(3,1)}$ is discarded because $\operatorname{supp} B^0_{(3,1)} \subseteq \square^1$. The index set of the active b-splines on level 1 is $\mathcal{D}^1 = [6,\dots,9]\times[2,\dots,5]$.

Next, in addition to $B^0_{(3,1)}$ we also refine $B^0_{(6,1)}$. The supports of both b-splines form the connected subdomain $\square^1 = [3,9)\times$

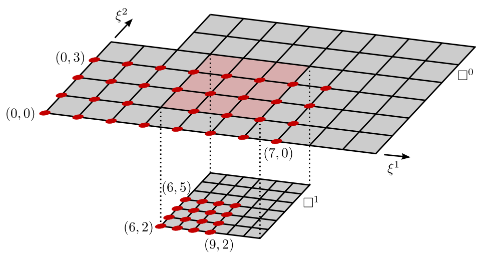


Figure 9: First refinement of quadratic hb-splines (after compilation).

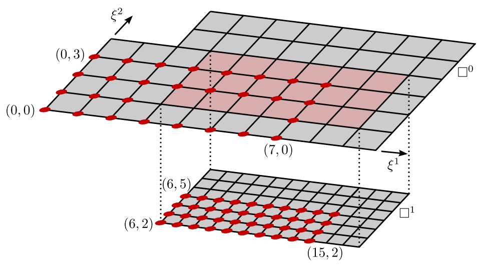


Figure 10: Second refinement of quadratic hb-splines (after compilation).

$[1,4)$, c.f. Figure 10. In the compilation step we first assume that all b-splines on $\square^0$ and $\square^1$ are present. This means the set $\mathcal{D}^1$ comprises in addition to the children of $B^0_{(3,1)}$ and $B^0_{(6,1)}$ the b-splines with the indices $\boldsymbol{k}\in[10,11]\times[2,\dots,5]$. In the compilation step, from the set $\mathcal{D}^0$ the b-splines $B^0_{(4,1)}$ and $B^0_{(5,1)}$ are removed because their supports are a subset of $\square^1$. The result after the compilation step is shown in Figure 10.

3.3. Hierarchical tensor product b-splines on effective domain

The introduced hb-spline basis is incomplete in cells close to the initial domain boundary $\partial\square^0$. Its polynomial reproduction properties are compromised because in boundary cells some of the b-splines are missing. As can be deduced from the univariate case, see Figure 4, for b-splines of degree μ only cells that are μ cells away from the domain boundary have the full set of non-zero b-splines. We therefore introduce the notion of an *effective domain* $\boxminus := \boxminus^0$, which is comprised of cells that have the complete set of non-zero b-splines, see Figure 11. The cells between the effective and initial domains may be regarded as guard or ghost cells.

The effective domain can be determined by counting the intersecting b-splines of the cells on the initial level. If the number of b-splines in

$$\mathcal{P}_{\boldsymbol{j}}^0 = \{\boldsymbol{i}\in\mathcal{I}^0 \,|\, \square_{\boldsymbol{j}}^0 \subset \operatorname{supp} B_{\boldsymbol{i}}^0\}$$

on level 0 is equal to $(\mu+1)^d$, then a complete basis is available on cell $\square_{\boldsymbol{j}}^0$. The union of all cells with complete basis forms the effective domain

$$\boxminus = \bigcup_{\boldsymbol{j}\in\mathcal{J}^0} \square_{\boldsymbol{j}}^0 \quad \text{with} \quad \mathcal{J}^0 = \{\boldsymbol{j}\in\mathbb{Z}^d \,|\, |\mathcal{P}_{\boldsymbol{j}}^0| = (\mu+1)^d\}$$

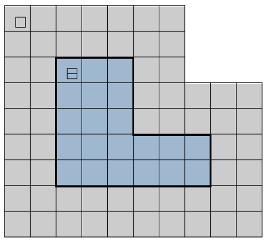


Figure 11: Initial and effective domains in case of quadratic b-splines ($\mu = 2$). The effective domain is the union of the blue shaded cells.

In [16] Kraft proved the linear independence of the hb-basis on the domain $\square^0$. This property is not immediately guaranteed on the effective domain $\boxminus$. However, Kraft's proof can be adapted to the effective domain if the index set of active b-splines $\mathcal{D}^\ell$, see (23), is modified to incorporate the effective domain, i.e.

$$\begin{aligned}\mathcal{D}^\ell = \{\boldsymbol{i} \in \mathbb{Z}^d \,|\, &(\operatorname{supp} B_{\boldsymbol{i}}^\ell \cap \boxminus) \subseteq \square^\ell \quad \text{and} \\ &(\operatorname{supp} B_{\boldsymbol{i}}^\ell \cap \boxminus) \not\subseteq \square^{\ell+1} \quad \text{and} \\ &(\operatorname{supp} B_{\boldsymbol{i}}^\ell \cap \boxminus) \neq \varnothing \quad \}\end{aligned} \tag{25}$$

The three conditions in the index set $\mathcal{D}^\ell$ reflect what we referred to as compilation, refinement and shrinkage of b-splines in the refinement algorithm, see also Appendix A. The three conditions are complemented with the nestedness property of the subdomains, i.e. $\square^\ell \supset \square^{\ell+1}$, see also (19).

The hierarchical tensor product b-splines on the effective domain form the space

$$\mathcal{B}_{\text{hb}}(\boxminus) := \operatorname{span}\{B_{\boldsymbol{k}}^\ell \,|\, 0 \leq \ell \leq \ell_{\max} \text{ and } \boldsymbol{k} \in \mathcal{D}^\ell\}$$

which will be used in Section 4 to discretise partial differential equations with the finite element method.

3.4. Subdivision refinement of hierarchical b-splines

The subdivision refinement of hb-spline interpolants is formally identical with the subdivision refinement of b-spline interpolants introduced in Sections 2.2 and 2.3. For instance, a spline given on level 0

$$u(\boldsymbol{\xi}) = \sum_{\boldsymbol{i}\in\mathcal{I}^0} B_{\boldsymbol{i}}^0(\boldsymbol{\xi})\, u_{\boldsymbol{i}}^0 = \boldsymbol{B}^0(\boldsymbol{\xi}) \cdot \boldsymbol{u}^0 \tag{26}$$

can also be interpolated with hb-splines from the space $\mathcal{B}_{\text{hb}}(\boxminus)$, that is

$$u(\boldsymbol{\xi}) = \sum_\ell \sum_{\boldsymbol{k}\in\mathcal{D}^\ell} B_{\boldsymbol{k}}^{\ell,\mu}(\boldsymbol{\xi})\, u_{\boldsymbol{k}}^\ell = \sum_\ell \boldsymbol{B}_{\text{hb}}^\ell(\boldsymbol{\xi}) \cdot \boldsymbol{u}_{\text{hb}}^\ell \tag{27}$$

where the vectors $\boldsymbol{B}_{\text{hb}}^\ell$ and $\boldsymbol{u}_{\text{hb}}^\ell$ contain the active b-splines and their coefficients on level ℓ, respectively. Equations (26) and (27) represent the same spline $u(\boldsymbol{\xi})$ only when the coefficients $\boldsymbol{u}_{\text{hb}}^\ell$ are determined with a refinement relation.

The coefficients $\boldsymbol{u}_{\text{hb}}^\ell$ are computed using the subdivision refinement relation (7) selectively applied to the coefficients of inactive b-splines. We rewrite (26) as follows

$$u(\boldsymbol{\xi}) = \boldsymbol{B}_{\text{hb}}^0(\boldsymbol{\xi}) \cdot \boldsymbol{u}_{\text{hb}}^0 + \sum_{\boldsymbol{j}\in\mathcal{I}^0\setminus\mathcal{D}^0} B_{\boldsymbol{j}}^0(\boldsymbol{\xi})\, u_{\boldsymbol{j}}^0 \tag{28}$$

with the second term containing the inactive b-splines on level 0. The coefficients of the inactive b-splines at level 0 are transferred to the b-splines on hierarchy level 1 using subdivision, i.e.

$$u_{\boldsymbol{j}}^1 = \sum_{\boldsymbol{i}\in\mathcal{I}^0\setminus\mathcal{D}^0} S_{\boldsymbol{i},\boldsymbol{j}-2\boldsymbol{i}}^\mu\, u_{\boldsymbol{i}}^0 \tag{29}$$

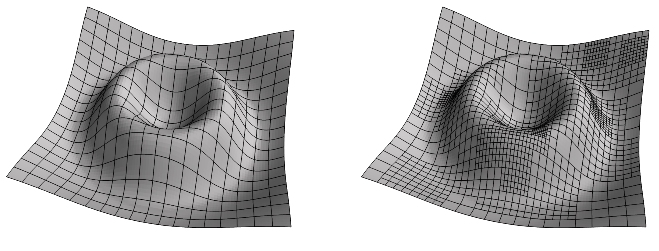
The coefficients $u_{\boldsymbol{j}}^1$ comprise the coefficients of inactive and active hb-splines on level 1. After extracting the coefficients $\boldsymbol{u}_{\text{hb}}^1$ of the active b-splines $\boldsymbol{B}_{\text{hb}}^1$ on level 1, the spline (26) can be rewritten as

$$u(\boldsymbol{\xi}) = \boldsymbol{B}_{\text{hb}}^0(\boldsymbol{\xi}) \cdot \boldsymbol{u}_{\text{hb}}^0 + \boldsymbol{B}_{\text{hb}}^1(\boldsymbol{\xi}) \cdot \boldsymbol{u}_{\text{hb}}^1 + \sum_{\boldsymbol{j}\in\mathcal{I}^1\setminus\mathcal{D}^1} B_{\boldsymbol{j}}^1(\boldsymbol{\xi})\, u_{\boldsymbol{j}}^1 \tag{30}$$

The last term now holds the inactive b-splines on level 1 which are next to be replaced with hb-splines on higher levels. The corresponding coefficients can be recursively determined with the selective subdivision (29). This recursive subdivision procedure is repeated until all coefficients $\boldsymbol{u}_{\text{hb}}^\ell$ in (27) are determined.

The hb-spline basis of $\mathcal{B}_{\text{hb}}(\boxminus)$ spans a larger space than that of the b-splines on level 0. If the selective subdivision relation, i.e. (29) and similarly for higher levels, is not required to hold, geometries can have more detail when interpolating with hb-splines from $\mathcal{B}_{\text{hb}}(\boxminus)$.

As an example, in Figure 12 the interpolation of a surface with cubic tensor product b-splines and randomly refined hb-splines is shown. The b-spline coefficients in this example are coordinate vectors in three dimensions. The grid lines on both spline surfaces are the projections of the tensor product grid in the two-dimensional parameter space. As can be visually confirmed, the surfaces in Figures 12a and 12b are identical. This has been achieved by determining the coefficients of the hb-spline interpolation with the subdivision relation (29).



(a) Initial representation. (b) Locally refined representation.

Figure 12: Same surface represented by two distinct cubic hb-spline bases.

4. Finite element discretisation of boundary value problems with hb-splines

The hb-splines provide a linearly independent, locally refinable basis with the smoothness and polynomial reproduction properties of b-splines. Thus they are ideal for adaptive discretisation of boundary value problems with the finite element method. See [12, 13, 19, 20] for finite element related discretisation techniques with hb-splines. As discussed in Section 3, hb-splines are inherently basis function centric. At first sight this appears to be at odds with the element focus of traditional finite elements. In this section we introduce the *subdivision projection* technique which provides an elegant means to reconciling the two differing viewpoints and to interface traditional finite element implementations with hb-splines.

4.1. Governing equations and finite element discretisation

We consider the Poisson equation as a representative second-order boundary value problem. The Poisson equation on the physical domain Ω is given by

$$\begin{aligned} -\nabla \cdot \nabla u &= f \quad \text{in } \Omega \\ u &= \overline{u} \quad \text{on } \Gamma_D \\ \boldsymbol{n} \cdot \nabla u &= \overline{g} \quad \text{on } \Gamma_N \end{aligned} \tag{31}$$

where $\overline{u}$ is the prescribed solution field on the Dirichlet boundary Γ_D and $\overline{g}$ is the prescribed flux on the Neumann boundary with the outward normal $\boldsymbol{n}$. The weak formulation of the Poisson equation can be stated according to Nitsche [21] as: Find $u \in H^1(\Omega)$ such that

$$\overbrace{\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega}^{a(u,v)} = \overbrace{\int_\Omega f\, v \, \mathrm{d}\Omega + \int_{\Gamma_N} \overline{g}\, v \, \mathrm{d}\Gamma}^{b(v)} - \overbrace{\gamma \int_{\Gamma_D} (u - \overline{u})\, v \, \mathrm{d}\Gamma}^{\gamma p(u,v)} \\ + \underbrace{\int_{\Gamma_D} \Big((u-\overline{u})\, \boldsymbol{n} \cdot \nabla v + (\boldsymbol{n} \cdot \nabla u)\, v \Big) \mathrm{d}\Gamma}_{l(u,v)} \tag{32}$$

for all $v \in H^1(\Omega)$. Herein, γ is a parameter which has to be suitably chosen, see [22, 23]. Our decision to weakly enforce the Dirichlet boundary conditions is motivated by the observation that b-splines are non-interpolating on the boundaries. Alternative approaches for weakly enforcing Dirichlet boundary conditions include i-splines [24] and Lagrange multipliers [25].

The trial and test functions are discretised with hb-splines $u^h, v^h \in \mathcal{B}_{\text{hb}}(\Omega) \subset H^1(\Omega)$, i.e.

$$\begin{aligned} u^h(\boldsymbol{\xi}) &= \sum_\ell \boldsymbol{B}^\ell_{\text{hb}}(\boldsymbol{\xi}) \cdot \boldsymbol{u}^\ell_{\text{hb}} \\ v^h(\boldsymbol{\xi}) &= \sum_\ell \boldsymbol{B}^\ell_{\text{hb}}(\boldsymbol{\xi}) \cdot \boldsymbol{v}^\ell_{\text{hb}} \end{aligned} \tag{33}$$

Here the summations are over the levels of the hb-spline basis.

By invoking the isoparametric formalism the physical domain Ω is expressed as the projection of the effective parameter domain $\boxminus$ into the physical space.

$$\boldsymbol{x}^h(\boldsymbol{\xi}) = \sum_\ell \boldsymbol{B}^\ell_{\text{hb}}(\boldsymbol{\xi}) \cdot \boldsymbol{x}^\ell_{\text{hb}} \tag{34}$$

In practice only the coordinates of the control points $\boldsymbol{x}^0$ on the initial coarse grid are given. The subdivision technique introduced in Section 3.4 is used to generate the coefficients $\boldsymbol{x}^\ell_{\text{hb}}$ for higher levels such that they exactly replicate the original geometry described by the coefficients $\boldsymbol{x}^0$.

Introducing the interpolation equations (33) and (34) into the generalised weak form (32) yields a linear system of equations with the unknowns $\boldsymbol{u}^\ell$. For instance, the bilinear form $a(u, v)$ in (32) becomes after discretisation

$$a(u^h, v^h) = \sum_m \sum_\ell {\boldsymbol{v}^m_{\text{hb}}}^\top \int_\Omega \nabla \boldsymbol{B}^m_{\text{hb}} \, {\nabla \boldsymbol{B}^\ell_{\text{hb}}}^\top \, \mathrm{d}\Omega \, \boldsymbol{u}^\ell_{\text{hb}} = \mathbf{v}^\top \mathbf{A} \mathbf{u} \tag{35}$$

in which the unknowns and the integral are assembled into the vector $\mathbf{u}$ and matrix $\mathbf{A}$, respectively. In (35) the two summations are over the levels of the hb-spline basis. Although $\boldsymbol{v}^m_{\text{hb}}$ and $\boldsymbol{u}^\ell_{\text{hb}}$ are by definition level-specific, the final vectors $\mathbf{u}$ and $\mathbf{v}$ contain all the coefficients in the hb-spline space. The discretisation of the other integrals in the generalised weak form (32) proceeds along the lines of the discretisation of the bilinear form.

4.1.1. Evaluation of the discretised weak form

The discretised integrals resulting from the weak form (32), like the bilinear form (35), are to be integrated numerically. As usual the integration is accomplished by generating a non-overlapping tiling of the parameter domain and thus the physical domain. A canonical tiling of the parameter domain is given by the cells of the tensor product grid. As discussed in Section 3.2, hb-splines have a hierarchy of nested tensor product grids so that the generation of a non-overlapping tiling requires care. We shall refer to the non-overlapping tiles as integration cells or elements, although they are not strictly elements in the traditional finite element sense.

Before specifying a tiling of the parameter domain, we define the scalar variable *level width*

$$\Lambda(\boldsymbol{\xi}) = \max_\ell \{\ell \geq 0 \,|\, \boldsymbol{\xi} \in \operatorname{supp} B^\ell_{\boldsymbol{i}},\, \boldsymbol{i} \in \mathcal{D}^\ell\} \tag{36}$$

which is equal to the local maximum of b-spline refinement level ℓ present at the coordinate $\boldsymbol{\xi}$. As defined in (25), the set $\mathcal{D}^\ell$ refers to the index set of active b-splines on level ℓ. The level width $\Lambda(\boldsymbol{\xi})$ is a piecewise constant scalar function due to the nestedness of the subdomains in hb-spline refinement.

With the definition of $\Lambda(\boldsymbol{\xi})$ to hand, the integration cells are defined as the union of the cells which are on the level $\Lambda(\boldsymbol{\xi})$; see also the illustrative example below. Formally, the index set of the integration cells can be identified with

$$\mathcal{T}^\ell = \{\boldsymbol{i} \in \mathbb{Z}^d \,|\, \square^\ell_{\boldsymbol{i}} \subset \boxminus \text{ and } \Lambda(\square^\ell_{\boldsymbol{i}}) = \ell\} \tag{37}$$

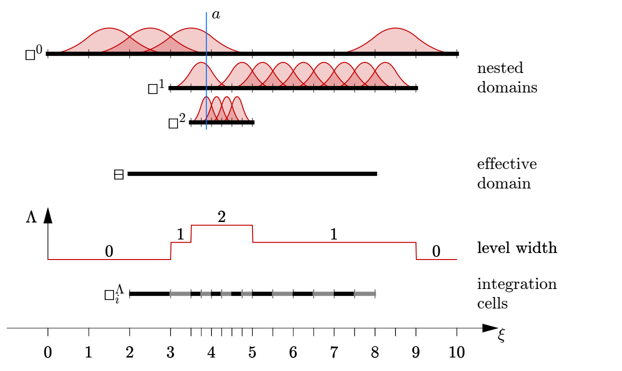


Figure 13: Nested domains of univariate, quadratic b-spline example shown in Figure 7 and its effective domain, level width and integration cells.

This definition contains the additional constraint that only indices of cells which lie within the effective domain $\boxminus$ are considered. The introduced tiling is non-overlapping such that

$$\boxminus = \bigcup_{j \in \mathcal{J}^0} \square_j^0 = \bigcup_{\substack{k \in \mathcal{T}^\ell \\ \ell \geq 0}} \square_{\boldsymbol{k}}^\ell \tag{38}$$

In this tiling integration cells are identical with the knot intervals of the finest active b-splines. This ensures the numerical integration can be performed as efficiently as possible.

Illustrative example. We reconsider the univariate quadratic b-spline example presented in Section 3.1 to illustrate the definition of integration cells and the evaluation of b-splines. In Figure 13 the hb-spline basis functions after two refinement and compilation steps are reproduced. In addition, Figure 13 contains a plot of the level width $\Lambda(\xi)$ over the domain. As mentioned, $\Lambda(\xi)$ is a piecewise constant function with jumps at knots. Below the plot of $\Lambda(\xi)$, the integration cells are pictured as an alternating band of black and grey line segments.

Continuing with the illustrative example, we consider next the evaluation of hb-splines at the point $\xi = a$ indicated in Figure 13. This point lies in the integration cell $\square_{15}^2 = [3.75, 4)$ and can be thought of as a quadrature point within that cell. From Figure 13 can be seen that four b-splines are non-zero at the evaluation point. Thus the interpolation within the cell $\square_{15}^2$ involves four b-splines over three distinct levels. In contrast, the interpolation within most other cells in this example involves only three b-splines over one single level.

4.2. Evaluation of the element integrals with subdivision projection

In hb-spline finite elements the interpolation within some integration cells may depend on b-splines defined across multiple levels. In addition, the number of degrees of freedom per element, i.e. integration cell, is not constant across the mesh. It is, however, possible to evaluate the element matrices and vectors using a constant number of b-splines from the finest level. The matrices and vectors are subsequently projected to

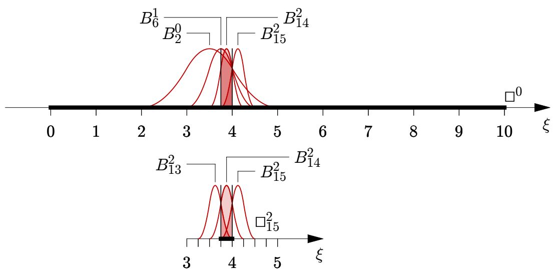


Figure 14: Representation of hb-splines on integration cell in univariate example with quadratic b-splines, Figure 7.

the correct levels using subdivision weights. We refer to this projection operation as subdivision projection. This new technique facilitates the interfacing of hb-splines with conventional finite element implementations which rely on a constant number of shape functions. To this purpose a possible alternative approach would be to adapt the Bezier extraction technique proposed in [26, 27] to hb-splines.

4.2.1. *Basic idea and approach*

As discussed throughout the paper, using the refinement relation (4) a b-spline can be expressed as a linear combination of b-splines on the next finer level. Consequently, when the refinement relation is recursively applied, a b-spline can be expressed as a linear combination of b-splines at any finer level. This observation is used to express all non-zero b-splines within an integration cell using b-splines from one single level. As a result, during numerical integration all element integrals will depend on a fixed number of functions. The subdivision projection weights are subsequently used to map the fixed size element vectors and matrices into the hierarchic approximation space.

Revisiting the univariate example in Figure 13, the sample evaluation point $a = 3.875$ lies in integration cell $\square_{15}^2 = [3.75, 4)$. The upper half of Figure 14 shows all active hb-splines in $\square_{15}^2$, namely B_2^0, B_6^1, B_{14}^2 and B_{15}^2. The bottom half of Figure 14 shows the three same-level non-zero b-splines B_{13}^2, B_{14}^2 and B_{15}^2 of the integration cell $\square_{15}^2$. Our aim is to represent each non-zero hb-spline over $\square_{15}^2$ as a linear combination of its three same-level non-zero b-splines. To begin with, we write for the same-level non-zero b-splines B_{14}^2 and B_{15}^2 the identities

$$B_{14}^2(\xi) = \begin{bmatrix} B_{13}^2 & B_{14}^2 & B_{15}^2 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{t}_{14}^{2,2}}$$

$$B_{15}^2(\xi) = \begin{bmatrix} B_{13}^2 & B_{14}^2 & B_{15}^2 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{t}_{15}^{2,2}}$$

The lower level hb-splines B_2^0 and B_6^1 can also be represented using the three same-level non-zero b-splines of $\square_{15}^2$. Considering the two-scale relation for quadratic hb-splines (Figure 2), the b-spline B_6^1 on level one can be expressed with

$$B_6^1(\xi) = \begin{bmatrix} B_{13}^2 & B_{14}^2 & B_{15}^2 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \frac{3}{4} & \frac{3}{4} & \frac{1}{4} \end{bmatrix}}_{\boldsymbol{t}_6^{1,2}}$$

Similarly, the weights for the b-spline B_2^0 on level zero can be found by recursively considering the two-scale relation

$$B_2^0(\xi) = \begin{bmatrix} B_{13}^2 & B_{14}^2 & B_{15}^2 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \frac{3}{4} & \frac{5}{8} & \frac{3}{8} \end{bmatrix}}_{\boldsymbol{t}_2^{0,2}}$$

The introduced subdivision projection vectors $\boldsymbol{t}_i^{\ell,2}$ for the sample integration cell $\square_{15}^2$, allow us to represent the b-splines on coarser levels with the same-level non-zero b-splines of the integration cell.

In terms of interpolation of a spline curve $u(\boldsymbol{\xi})$, the subdivision projection has the following consequences. The spline segment over the integration cell $\square_{15}^2$ uses the coefficients u_i^ℓ, i.e.

$$u\big|_{\square_{15}^2} = B_2^0\, u_2^0 + B_6^1\, u_6^1 + B_{14}^2\, u_{14}^2 + B_{15}^2\, u_{15}^2$$

This spline curve can also be equally written with

$$u\big|_{\square_{15}^2} = \begin{bmatrix} B_{13}^2 & B_{14}^2 & B_{15}^2 \end{bmatrix} \cdot \left(\boldsymbol{t}_2^{0,2} u_2^0 + \boldsymbol{t}_6^{1,2} u_6^1 + \boldsymbol{t}_{14}^{2,2} u_{14}^2 + \boldsymbol{t}_{15}^{2,2} u_{15}^2 \right)$$

using the same-level non-zero b-splines of the integration cell $\square_{15}^2$ and the subdivision projection vectors $\boldsymbol{t}_i^{\ell,2}$.

4.2.2. *Subdivision projection*

In this section we formalise the subdivision projection technique and elaborate on the efficient computation of the subdivision projection vectors introduced in the foregoing section. As highlighted in the preceding section, the non-zero hb-splines $B_i^\ell \in \mathcal{B}_{\text{hb}}$ over an integration cell $\square_k^\Lambda$ can be expressed as a linear combination of same-level non-zero b-splines $\boldsymbol{B}^\Lambda$

$$B_i^\ell = \boldsymbol{B}^\Lambda \cdot \boldsymbol{t}_i^{\ell,\Lambda} \tag{39}$$

where the vector $\boldsymbol{t}_i^{\ell,\Lambda}$ is associated with the specific b-spline B_i^ℓ. For the following it is important that $\boldsymbol{B}^\Lambda$ and $\square_k^\Lambda$ have the same level Λ. Moreover, recall that always $\ell \le \Lambda$ so that the projection vectors $\boldsymbol{t}_i^{\ell,\Lambda}$ are always well defined.

A computationally convenient approach to determining the projection vectors is the subdivision refinement algorithm introduced in Sections 2.4 and 3.4. To this end, consider the interpolation of a spline with $B_i^\ell \in \mathcal{B}_{\text{hb}}$ and the finer b-splines $\boldsymbol{B}^\Lambda$, i.e.

$$u(\boldsymbol{\xi}) = \sum_{\boldsymbol{i}} B_{\boldsymbol{i}}^\ell(\boldsymbol{\xi})\, u_{\boldsymbol{i}}^\ell = \sum_{\boldsymbol{j}} B_{\boldsymbol{j}}^\Lambda(\boldsymbol{\xi})\, u_{\boldsymbol{j}}^\Lambda = \boldsymbol{B}^\Lambda(\boldsymbol{\xi}) \cdot \boldsymbol{u}^\Lambda \tag{40}$$

and restricted to the spline segment over the cell $\square_k^\Lambda$ only the b-splines that are non-zero on the cell are needed

$$u\big|_{\square_k^\Lambda} = \sum_{\boldsymbol{j} \in \mathcal{G}_k^\Lambda} B_{\boldsymbol{j}}^\Lambda(\boldsymbol{\xi})\, u_{\boldsymbol{j}}^\Lambda = \boldsymbol{B}_k^\Lambda(\boldsymbol{\xi}) \cdot \boldsymbol{u}^\Lambda \tag{41}$$

In line with the standard subdivision approach, introducing (39) in to (40) yields a relation for the coefficients of the spline on the two distinct levels

$$\boldsymbol{u}^\Lambda = \sum_{\boldsymbol{i}} \boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda} u_{\boldsymbol{i}}^\ell \tag{42}$$

The coefficients of the finer level Λ are a linear combination of the coefficients of the coarser level ℓ. When the difference between the levels ℓ and Λ is one, the projection vector $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ is a row of the previously introduced subdivision matrix (3). However, when the difference between ℓ and Λ is more than one, $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ represents the result of several subdivision steps.

As proposed in [14], the components of the projection vector $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ can be conveniently determined with the aid of the subdivision algorithm. It is worth repeating that $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ corresponds to a specific control point $\boldsymbol{i}$ at the level ℓ. Hence, in order to determine $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ with the subdivision algorithm we can assign a value 1 to the control point $\boldsymbol{i}$ at the level ℓ and values 0 to all others. After applying the subdivision refinement algorithm the coefficients of $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ can be collected from the refined mesh.

As an example, in Figure 15 the computation of the projection vectors using subdivision is illustrated. Imagine the highlighted cell $\square_k^2$ in Figure 15b is an integration cell and the b-spline $B_{\boldsymbol{i}}^0$ of level 0 is an active hb-spline. The b-spline $B_{\boldsymbol{i}}^0$ is to be expressed as a linear combination of b-splines $B_{\boldsymbol{i}}^2$ on the same level as the integration cell. The series of three figures in the bottom half of Figure 15 show how the weights $\boldsymbol{t}_{\boldsymbol{i}}^{0,2}$ are determined using subdivision. In Figure 15b a value 1 is assigned to the control point $\boldsymbol{i}$ and values 0 to all other control points. The coefficients after one and two subdivision refinement steps are shown in Figures 15c and 15d. The values depicted in Figure 15d are the components of the vector $\boldsymbol{t}_{\boldsymbol{i}}^{0,2}$.

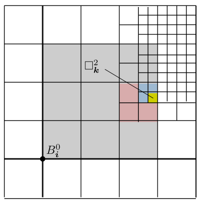
For subsequent derivations it is convenient to introduce a picking matrix $\boldsymbol{G}_k^\ell$ which extracts all non-zero hb-splines $B_{\boldsymbol{i}}^\ell(\boldsymbol{\xi})$ in $\mathcal{B}_{\text{hb}}$ over an integration cell $\square_k^\Lambda$. The integration-cell-specific matrix $\boldsymbol{G}_k^\ell$ picks from each level ℓ the hb-splines which are non-zero over the cell $\square_k^\Lambda$. Thus the non-zero b-splines of the cell $\square_k^\Lambda$ are the b-splines $\boldsymbol{G}_k^\ell \boldsymbol{B}_{\text{hb}}^\ell$ over multiple levels. With this definition to hand we can formally write for the projection relation (39)

$$\boldsymbol{G}_k^\ell \boldsymbol{B}_{\text{hb}}^\ell(\boldsymbol{\xi}) = \boldsymbol{T}_k^{\ell,\Lambda} \boldsymbol{B}_k^\Lambda(\boldsymbol{\xi}) \quad \text{in } \boldsymbol{\xi} \in \square_k^\Lambda \tag{43}$$

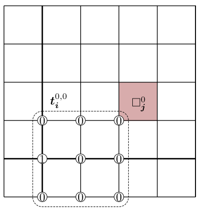
Here the new matrix $\boldsymbol{T}_k^{\ell,\Lambda}$ is a collection of projection vectors $\boldsymbol{t}_{\boldsymbol{i}}^{\ell,\Lambda}$ associated to the non-zero hb-splines over the integration cell.

Application of subdivision projection to discretisation. We now proceed to the evaluation of the bilinear and linear forms appearing in the hb-spline discretized weak form (32). Firstly, with the introduced tiling of the problem domain in Section 4.1.1, the domain integrals in the weak form can be determined by summation of element integrals. After the domain integrals are split into element contributions, for example, the discretized bilinear form (35) reads

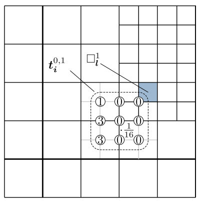
$$a(u^h, v^h) = \sum_m \sum_\ell \boldsymbol{v}_{\text{hb}}^{m\,\top} \left(\sum_k \int_\omega \nabla \boldsymbol{B}_{\text{hb}}^m \, \nabla \boldsymbol{B}_{\text{hb}}^{\ell\,\top} \, \mathrm{d}\omega_k^\Lambda \right) \boldsymbol{u}_{\text{hb}}^\ell \tag{44}$$



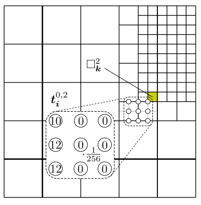
(a) Supporting cells of b-spline B_i^0 and integration cell $\square_k^2$ on two levels higher.



(b) Weights on level 0.



(c) Weights on level 1.



(d) Weights on level 2.

Figure 15: Determining the subdivision projection weights $\boldsymbol{t}_i^{0,2}$ for the quadratic b-spline B_i^0 on the integration cell $\square_k^2$ to represent B_i^0 with the same-level non-zero b-splines $\boldsymbol{B}_k^2$ of the integration cell.

The element integrals over ω_k^Λ can be transformed into integrals over integration cells $\square_k^\Lambda$ in the parametric domain by invoking the isoparametric mapping. In general only few b-splines $B_i^\ell \in \mathcal{B}_{\text{hb}}$ are non-zero over an element. As introduced previously, the non-zero hb-splines over an element ω_k^Λ are denoted with $\boldsymbol{G}_k^\ell \boldsymbol{B}_{\text{hb}}^\ell$, where $\boldsymbol{G}_k^\ell$ is a matrix to pick the relevant hb-splines from level ℓ. Thus we can equally write for (44)

$$a(u^h, v^h) = \sum_m \sum_\ell \boldsymbol{v}_{\text{hb}}^{m\,\top} \left(\sum_k \int_\omega \boldsymbol{G}_k^m \nabla \boldsymbol{B}_{\text{hb}}^m \, \nabla \boldsymbol{B}_{\text{hb}}^{\ell\,\top} \boldsymbol{G}_k^{\ell\,\top} \, \mathrm{d}\omega_k^\Lambda \right) \boldsymbol{u}_{\text{hb}}^\ell \tag{45}$$

The number of non-zero hb-splines over an element ω_k^Λ is not constant. Therefore we introduce next the subdivision projection (43) into (45) so that the element integrals depend on a fixed number of b-splines.

$$a(u^h, v^h) = \sum_m \sum_\ell \boldsymbol{v}_{\text{hb}}^{m\,\top} \left(\sum_k \boldsymbol{T}_k^{m,\Lambda} \int_\omega \nabla \boldsymbol{B}_k^\Lambda \, \nabla \boldsymbol{B}_k^{\Lambda\,\top} \, \mathrm{d}\omega_k^\Lambda \, \boldsymbol{T}_k^{\ell,\Lambda\,\top} \right) \boldsymbol{u}_{\text{hb}}^\ell \tag{46}$$

The element integrals now depend on a fixed number of b-splines all from the same level as the integration cell. As a result element matrices can be evaluated without consideration of the hierarchic nature of the hb-spline basis. It is equally possible to obtain the element matrices from a conventional finite element implementation that uses b-splines as shape functions. The multiplication of the element matrices so obtained with the $\boldsymbol{T}_k^{\ell,\Lambda}$ projects them to the hb-spline space. Importantly, in a finite element framework the multiplication with $\boldsymbol{T}_k^{\ell,\Lambda}$ can be deferred until the assembly stage of the global system matrix.

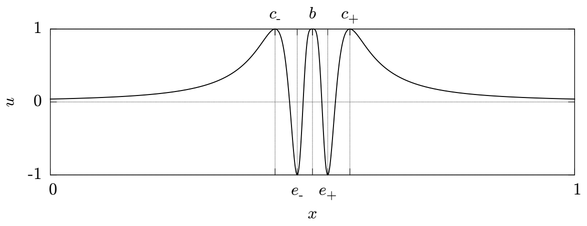


Figure 16: Solution of the one-dimensional Poisson problem.

Although, we focused in this section on the element matrices, the derivations also apply to element vectors.

Note that it is possible to express the b-splines appearing in (46) as the multiplication of Lagrange basis functions with an additional projection matrix. Expressing (46) with Lagrange basis functions would enable us immediately to port existing finite element implementations into the hb-spline finite element framework.

5. Examples

In this section the optimal convergence and the robustness of the presented hb-spline finite element method is established using standard linear elasticity benchmark examples. Convergence results for uniform and adaptive refinement are given. In addition, a geometrically nonlinear elasticity problem is analysed to demonstrate the method's applicability to nonlinear problems. In all examples the element integrals have been evaluated using full Gauss integration. There are more efficient integration schemes for b-splines which we could alternatively have used [28]. As introduced in Section 4.2.2, the element matrices and vectors are first computed on common finite elements which all have the same number of shape functions. They are subsequently assembled with subdivision projection into global matrices and vectors. As anticipated, in all numerical examples the presented refinement algorithms yield a linearly independent hb-spline basis.

5.1. One-dimensional Poisson problem

As an introductory example we consider the one-dimensional Poisson problem $\mathrm{d}^2 u / \mathrm{d}x^2 + f = 0$ on the domain $\Omega = (0, 1)$. The right hand side f is chosen so that the solution is equal to

$$u(x) = \sin\left(\frac{1}{a^2 (x-b)^2 + 2/(5\pi)} \right) \quad \text{with } a = 10, b = 1/2 \tag{47}$$

This solution is plotted in Figure 16. It oscillates close to the domain centre $x = b = 1/2$ with maxima $c_\mp = b \mp \frac{2\sqrt{10}}{a 5 \sqrt{\pi}}$ and minima $e_\mp = b \mp \frac{2}{a\sqrt{15\pi}}$. It tends, however, to level off towards the both ends of the domain. The hb-splines are ideally suited to capture the local oscillations in the solution.

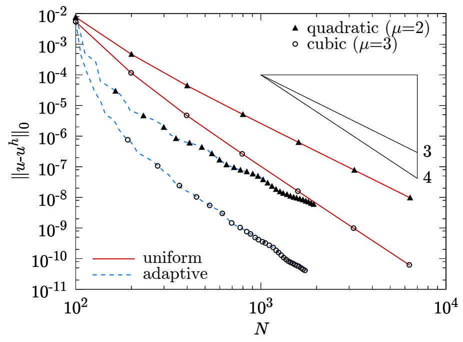


Figure 17: Convergence FE sinusoidal solution.

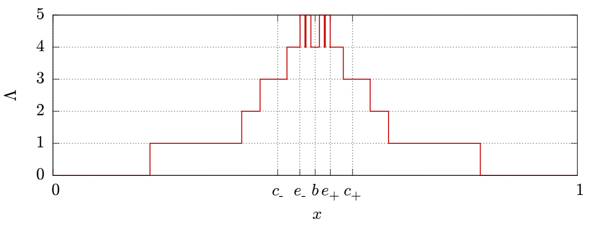


Figure 18: Local level width after 40 adaptive steps for cubic b-spline case.

For this example, despite the solution $u(x)$ being oscillatory, optimal convergence rates can be achieved for uniform refinement. The relevant standard finite element error estimate reads

$$\|u - u^h\|_0 \leq \frac{C}{N^{\mu+1}} \tag{48}$$

where u^h is the finite element solution, C is a constant, N is the total number of degrees of freedom, μ the polynomial degree of the b-splines and $\|\cdot\|_0$ is the standard L_2-norm. As shown in Figure 17, uniform refinement essentially leads to the predicted convergence rates for quadratic and cubic b-splines. Also shown in Figure 17 are the convergence of adaptively refined hb-splines. As implemented here, adaptive refinement is an iterative procedure and at each refinement step the hb-splines of the element with maximum error are refined. In Figure 17 only every 10th refinement step is indicated with '▲' or '∘', respectively. As expected, adaptive refinement yields the maximum convergence rate as the uniform refinement. However, the constant C in estimate (48) is significantly lower in case of adaptive refinement. As a result, the L_2-error for adaptive refinement is in comparison with uniform refinement by orders of magnitude smaller for a given number of N b-splines.

Overall 40 adaptive refinement steps have been performed leading to about 2000 active b-splines. As indicated in Figure 18 the maximum refinement occurs close to the centre of the domain. The maximum generational difference between the coarse and fine level b-splines is 5. Hence, the smallest refined cells are 2^5 times smaller than the original coarse cells.

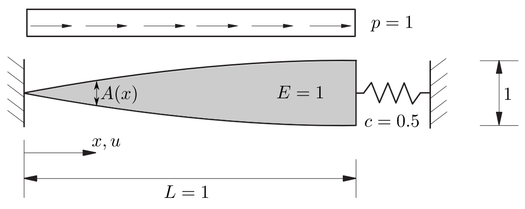


Figure 19: Problem description of bar with variable cross-section.

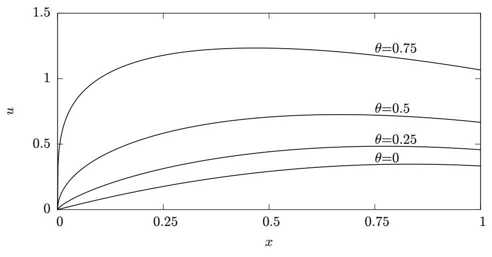


Figure 20: Bar with variable cross-section. Displacements for four singularity strengths θ and spring constant $c = 0.5$.

5.2. *Bar with geometric singularity*

An elastic bar with variable cross-section $A(x) = x^\theta$, with $0 < \theta < 1$, has a singular solution and as such it is useful to study the performance of refinement techniques [29, 30]. The boundary value problem for the bar shown in Figure 19 reads

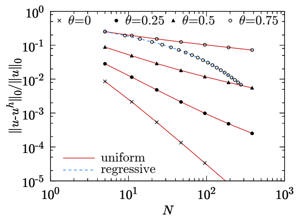
$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}\Big(EA(x)\frac{\mathrm{d}u}{\mathrm{d}x}\Big) + p &= 0 && \text{in } \Omega = (0, L) \\ u(x) &= 0 && \text{at } x = 0 \\ EA(x)\frac{\mathrm{d}u}{\mathrm{d}x} &= -cu(x) && \text{at } x = L \end{aligned} \tag{49}$$

Its solution in dependence of the free parameter θ is easily determined

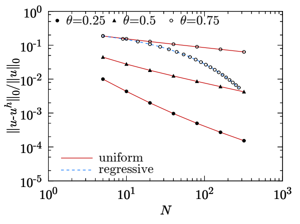
$$u(x) = -\frac{x^{2-\theta}}{2-\theta} + \frac{E + \frac{c}{2-\theta}}{E + \frac{c}{1-\theta}}\frac{x^{1-\theta}}{1-\theta} \tag{50}$$

The displacements have a singularity at $x = 0$ for $\theta > 0$ and the choice of θ controls the strength of the singularity. For instance, for $\theta = 1/2$ the solution exhibits a certain similarity to a crack tip in a two-dimensional plate. In Figure 20, the solutions $u(x)$ for four representative values of θ are shown.

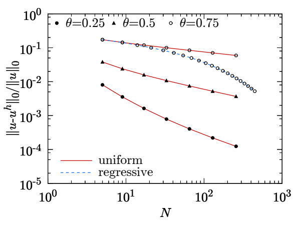
Due to the singularity at the left boundary the theoretical convergence rates for this problem are less than optimal for $\theta \neq 0$. In Figure 21 the convergence of the displacement errors in L_2-norm for different polynomial degrees and singularity strengths are shown. As can be seen, for uniform refinement the convergence order for $\theta = 0.25$, $\theta = 0.5$ and $\theta = 0.75$ is independent of the polynomial degree of the used b-spline. For $\theta = 0$ the solution (50) is a quadratic polynomial so that the convergence order for linear b-splines is quadratic (Figure 21a); and the solution can be exactly reproduced by the quadratic and cubic b-splines.



(a) Linear b-splines



(b) Quadratic b-splines



(c) Cubic b-splines

Figure 21: Bar with variable cross-section. Convergence of the relative displacement error in L_2 norm.

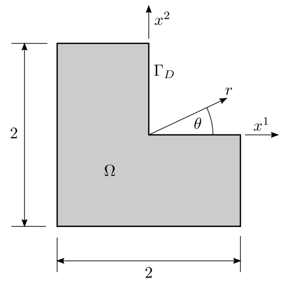


Figure 22: Geometry of the L-shaped domain.

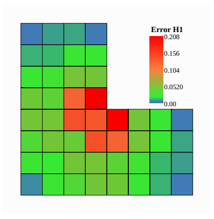
By choosing an appropriate non-uniform refinement the decrease in convergence rates with increasing θ can be mitigated. As depicted in Figure 21 this is demonstrated with the increase of convergence rates for singularity strength $\theta = 0.75$. The convergence curves associated with non-uniform refinement are referred to as 'regressive'. In the specific refinement strategy used, we successively refine near $x = 0$ and near $x = 1$ the refinement remains constant after the first step.

5.3. L-shaped domain

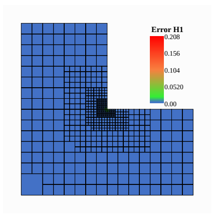
The Laplace equation on the L-shaped domain $\Omega = (-1, 1)^2 \setminus [0, 1]^2$ is a widely used benchmark example used for studying adaptive refinement, Figure 22. We demonstrate with this example the effectiveness of local hierarchical b-spline refinement in case of a two-dimensional domain with a singularity. The problem setup is chosen such that the exact solution is

$$u_r(r, \theta) = r^{2/3} \sin(2\theta/3 - \pi/3) \quad \text{for } r > 0 \text{ and } 0 < \theta \leq 2\pi \tag{51}$$

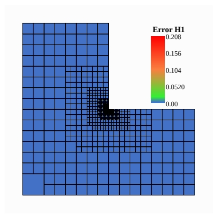
We computed this example with quadratic and cubic b-splines using uniform and hierarchical refinement. In all the computations the Dirichlet boundary conditions are applied with Nitsche's method and $\gamma = 10^{10}$. The parameter γ has been chosen large in order to sidestep the discussion about its optimal value. A possible estimation procedure for the parameter γ based on local eigenvalue problems can be found in [22]. The local refinement is carried out by successive refinement of integration cells which exhibit the largest error in the H^1-seminorm, see Figure 23. Or more precisely, the b-splines which are non-zero over the integration cells are refined. The error in the H^1-seminorm is determined with respect to the known exact solution (51).



(a) Initial.

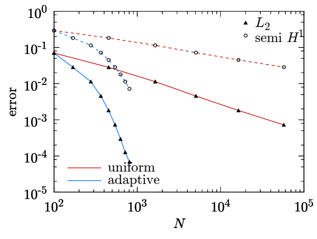


(b) 4th step.

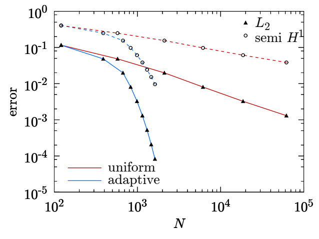


(c) 8th step.

Figure 23: L-shaped domain. Evolution of H^1-seminorm error isocontours for adaptive refinement with cubic hb-splines.



(a) Quadratic b-splines.



(b) Cubic b-splines.

Figure 24: L-shaped domain. Convergence of the solution errors in L_2-norm and H^1-seminorm.

The convergence of the L_2-norm and H^1-seminorm errors for uniform and adaptive refinement are compared in Figure 24. As expected the adaptive refinement strategy places fine hb-splines at the reentrant corner. The integration cells and H^1-seminorm errors for the corresponding computations with cubic b-splines are given in Figure 23. The loss of optimal convergence rate for the uniform case due to the presence of the reentrant corner is apparent. The local adaptive refinement is able to mitigate this drop in the convergence order and gives highly accurate solutions with significantly less degrees of freedom than in the uniform case.

5.4. Geometrically nonlinear three-dimensional solid

In this last example we present the discretisation of a geometrically nonlinear problem with hb-splines. The discretisation of nonlinear problems with hb-splines is straightforward as hb-splines are independent of the specific considered boundary value problem. In fact, as discussed in Section 4.2.2, using the proposed subdivision projection technique, existing nonlinear finite element implementations can readily be interfaced with the proposed hb-spline framework.

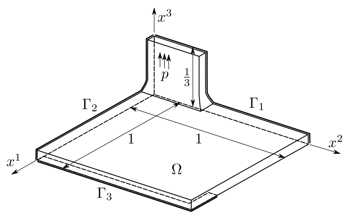
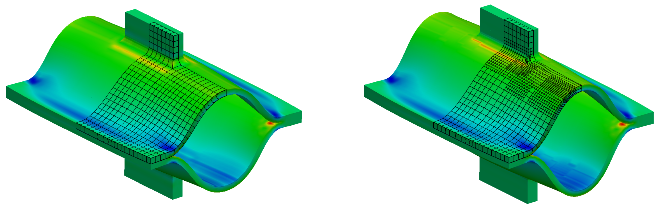


Figure 25: Geometrically nonlinear solid. Reference configuration of one-eighth of the geometry.



(a) Initial integration cells. (b) Locally refined integration cells at refinement step 15.

Figure 26: Geometrically nonlinear solid. Deflected shapes determined with quadratic hb-splines and integration cells.

The considered geometry consists of two plate-like solids in the x^1x^2-plane with two smaller attached plates in x^3-direction. An eighth of the system as shown in Figure 25 is analysed. Hence symmetric Dirichlet boundary conditions are applied to the boundaries Γ_1, Γ_2 and Γ_3. Each of the plates has a thickness of $1/18$ except at the fillet region at their intersection. The St. Venant–Kirchhoff material parameters Young modulus and Poisson ratio are chosen as $E = 10$ and $\nu = 0.3$, respectively. The vertical plate is subjected to a body load of $p = 1/2$ in x^3-direction per unit volume.

The computed deformed configuration of the complete system is shown in Figure 26. Quadratic b-splines are used as basis functions. Figure 26a shows the result obtained with the initial coarse grid and Figure 26b shows the result obtained with an adaptively refined grid. The isocontours in both plots indicate the x^1x^1-component of the second Piola–Kirchhoff stress tensor. As is to be expected, the maxima and minima of stress strongly correlate with locations of maximum curvature in the plates.

The local refinement in this example uses a residual based error indicator. The level of L_2-norm of the residual in equilibrium in each cell serves as a selection criterion for refinement, see e.g. [31]. It is noteworthy that in computing the residuals no jump terms across cell boundaries are present because of the C^1-continuity of quadratic hb-splines. The local refinement procedure increases the resolution of hb-splines which are non-zero over the integration cell with the highest residual.

6. Summary and conclusions

An isogeometric finite element analysis framework that uses the hierarchical b-splines in the form proposed by Kraft [16] was introduced. In hb-splines refinement proceeds by successively replacing selected b-splines with b-splines on finer grids. The refinement relation, or in other terms the two-scale relation of b-splines is a crucial ingredient of hb-spline refinement. Firstly, the refinement relation ensures that the basis can be isogeometrically refined without changing the object geometry. Secondly, the admissible refinement patterns given in [16] that lead to a linearly independent basis rely crucially on the refinement relation. An algorithmic interpretation of the refinement relation is given in form refinement rules in subdivision curves, surfaces and volumes. As demonstrated, the common link between hb-splines and subdivision can be exploited to facilitate the easy implementation of hb-splines.

The subdivision projection technique introduced for computing the element matrices and vectors enables to sidestep the complexities resulting from the hierarchic overlapping nature of hb-splines. In a straightforward implementation of hb-spline finite elements each element can have a different number of non-zero basis functions defined over several layers of the hierarchy. Subdivision projection allows to compute the element matrices and vectors with a fixed number of non-zero basis functions from the same level. The element matrices and vectors are multiplied with subdivision weights during the assembly stage of the global matrices and vectors. As a result subdivision projection allows the reuse of conventional finite element implementations by only using b-splines at a fixed level as basis functions. In practice, subdivision projection facilitates a modular software architecture in which the hb-splines provide only the projection matrices for each element.

The good performance of the introduced hb-spline finite elements was demonstrated with linear and geometrically nonlinear one-, two- and three-dimensional examples. As basis functions linear, quadratic and cubic hb-splines were used. The implementation discussed is able to handle arbitrary degree b-splines. In the considered test problems with singularities, local refinement based on an residual based error indicator facilitated significant increase in the convergence rates. As a result, sufficiently accurate solutions could be obtained with significantly less degrees of freedom compared to the uniform refinement. In case of nonlinear problems, an attractive property of hb-splines is the lossless transfer of field variables during the incremental/iterative solution procedure. As is known, the transfer of field variables between meshes requires special techniques if conventional finite elements are used, see e.g. [32, 33].

The proposed subdivision-based implementation of hierarchical b-splines was introduced with uniform b-splines. It is however possible to extend the presented approach to non-uniform rational b-splines (NURBS). It is particularly straightforward to consider rational b-splines by assigning each control point a weight and performing the subdivision in a space which has one more dimension than the embedding space. Subsequently, as usual rational b-splines are computed as the perspective projection from the higher-dimensional space to the embedding space [13, 14, 34]. In order to consider non-uniform b-splines it is instructive to consult previous work on non-uniform subdivision [7, 35, 36]. Specifically, only the presented data structures and algorithms have to be augmented so that knot intervals are stored and refined. In the introduced implementation we assumed that the knot intervals on each level are constant and are refined by bisection.

A further avenue for future research is the development of hb-splines for unstructured surface meshes. Grinspun et al. [20] have introduced a basis refinement technique for Loop subdivision surfaces similar to hb-splines. Recent results on higher-degree, non-uniform subdivision surfaces, see [36], make it feasible to develop NURBS-compatible hb-splines for unstructured surface meshes. Different from the surface case, the mathematical theory of subdivision on unstructured volume meshes is still very sparse. Therefore, immersed boundary type methods, also known as fictitious domain or embedded domain methods, in combination with hb-splines appear to be promising for problems with arbitrary topology [34].

Appendix A. Data structures and algorithms

In this appendix we discuss the implementation of hb-splines focusing on data structures and algorithms. As introduced in Section 3, hb-spline refinement proceeds in three sub-steps referred to as the *refinement*, *compilation* and *shrinkage* steps.

There are two different type of relations in the hb-spline basis which have to be managed. The first is the spatial relation between the supports of b-splines on the same level and the second is the parent-child relation between b-splines on different refinement levels. The relation between b-splines on the same level is effectively captured by the multi-index labelling of the tensor product grid and b-splines. For encoding the relation between b-splines on different levels a recursive tree data structure is the most straightforward choice. Each node of the hb-spline tree represents a b-spline and stores its multi-index $\boldsymbol{i} \in \mathbb{Z}^d$, refinement level ℓ and an activity tag. The nodes of the tree are linked according to the refinement relation (3). The set of all active b-splines $\mathcal{D}^\ell$, or in short $\{\mathcal{D}^\ell\}$, is collected by extracting all active b-splines in the complete index set (or tree) $\{\mathcal{I}^\ell\}$.

Before introducing the algorithms, we define two auxiliary index sets for describing the parent-child relation between b-splines on two consecutive levels.

- The index set of children $C^\ell_{\boldsymbol{i}} = \{\boldsymbol{k} \in \mathbb{Z}^d \,|\, \operatorname{supp} B^{\ell+1}_{\boldsymbol{k}} \subset \operatorname{supp} B^\ell_{\boldsymbol{i}}\}$ stores the $(\mu + 2)^d$ b-splines on the next finer level needed to represent $B^\ell_{\boldsymbol{i}}$.
- The index set of ancestors $\mathcal{A}^\ell_{\boldsymbol{i}} = \{\boldsymbol{k} \in \mathbb{Z}^d \,|\, \operatorname{supp} B^{\ell-1}_{\boldsymbol{k}} \supset \operatorname{supp} B^\ell_{\boldsymbol{i}}\}$ contains all b-splines on the coarser level $\ell - 1$ which cover completely the support of $B^\ell_{\boldsymbol{i}}$. Note, $|\mathcal{A}^\ell_{\boldsymbol{i}}|$ is only constant for even degree splines.

Moreover, we define the following set of update operations mimicking C-language syntax:

- $\mathcal{A} \cup\!= \mathcal{B}$ is short for $\mathcal{A} := \mathcal{A} \cup \mathcal{B}$; and
- $\mathcal{A} \setminus\!= \mathcal{B}$ is short for $\mathcal{A} := \mathcal{A} \setminus \mathcal{B}$.

The in the following listed algorithms cover refinement, unrefinement and methods for integration cells. The refinement algorithm 1 uses the three steps presented in algorithms 2, 3 and 4. Algorithm 5 shows how by parsing the active hb-splines $\{\mathcal{D}^\ell\}$, the integration cells can be collected. Algorithm 6 shows how to collect all active hb-splines on an integration cell. The unrefinement algorithm 7 contains algorithm 8 in which the hb-spline tree $\{\mathcal{I}^m\}$ is reduced in such a way that the targeted b-spline $B^\ell_{\boldsymbol{i}}$ is active again. The coefficients are transferred to the unrefined hb-splines with a least-squares fit in algorithm 9.

Algorithm 1 Refinement

```
Input: $\boxminus$, $\{\mathcal{I}^m\}$, $\{\mathcal{D}^m\}$, $(\boldsymbol{i}, \ell)$ with $\boldsymbol{i} \in \mathcal{D}^\ell$
1: Refine($(\boldsymbol{i}, \ell), \{\mathcal{I}^m\}$)
2: Compile($\{\mathcal{I}^m\}, \{\mathcal{D}^m\}$)
3: Shrink($\boxminus, \{\mathcal{D}^m\}$)
Output: $\{\mathcal{I}^m\}$, $\{\mathcal{D}^m\}$
```

Algorithm 2 Refine

```
Input: $(\boldsymbol{i}, \ell)$, $\{\mathcal{I}^m\}$
1: // expand indirectly domain at level $\ell + 1$ by adding children of $B^\ell_{\boldsymbol{i}}$
2: $\mathcal{I}^{\ell+1} \cup\!= C^\ell_{\boldsymbol{i}}$
3: // collect domain at level $\ell + 1$
4: $\square^{\ell+1} = \varnothing$
5: for each $\boldsymbol{i} \in \mathcal{I}^{\ell+1}$ do
6:     $\square^{\ell+1} \cup\!= \operatorname{supp} B^{\ell+1}_{\boldsymbol{j}}$
7: // complete index set at level $\ell + 1$
8: for each $\operatorname{supp} B^{\ell+1}_{\boldsymbol{i}} \subset \square^{\ell+1}$ do
9:     $\mathcal{I}^{\ell+1} \cup\!= \{\boldsymbol{i}\}$
Output: $\{\mathcal{I}^m\}$
```

Algorithm 3 Compile

```
Input: $\{\mathcal{I}^m\}$, $\{\mathcal{D}^m\}$
1: // construct linearly independent proceeding level-by-level
2: for each $m = 0, 1, \ldots$ do
3:     // discard b-splines which are refined
4:     $\mathcal{D}^m = \varnothing$
5:     for each $\boldsymbol{i} \in \mathcal{I}^m$ do
6:         if $C^m_{\boldsymbol{i}} \not\subset \mathcal{I}^{m+1}$ then
7:             $\mathcal{D}^m \cup\!= \{\boldsymbol{i}\}$
Output: $\{\mathcal{D}^m\}$
```

Algorithm 4 Shrink

```
Input: $\boxminus$, $\{\mathcal{D}^m\}$
1: for each $m = 0, 1, \ldots$ do
2:     // discard b-splines which are outside of effective domain
3:     for each $\boldsymbol{i} \in \mathcal{D}^m$ do
4:         if (supp $B^m_{\boldsymbol{i}} \cap \boxminus) = \varnothing$ then
5:             $\mathcal{D}^m \setminus= \{\boldsymbol{i}\}$
Output: $\{\mathcal{D}^m\}$
```

Algorithm 5 CollectIntegrationCells

```
Input: $\{\mathcal{T}^\ell\} = \varnothing$
1: for each $\ell = 0, 1, \ldots$ do
2:     // collect domain at level $\ell + 1$
3:     $\square^{\ell+1} = \varnothing$
4:     for each $\boldsymbol{k} \in \mathcal{D}^{\ell+1}$ do
5:         $\square^{\ell+1} \cup=$ supp $B^{\ell+1}_{\boldsymbol{k}}$
6:     // identify integration cells on level $\ell$
7:     $\mathcal{T}^\ell = \varnothing$
8:     for each $\boldsymbol{i} \in \mathcal{D}^\ell$ do
9:         for each $\square^\ell_{\boldsymbol{k}} \subset$ supp $B^\ell_{\boldsymbol{i}}$ do
10:            if $\square^\ell_{\boldsymbol{k}} \not\subset \square^{\ell+1}$ then
11:                $\mathcal{T}^\ell \cup= \{\boldsymbol{k}\}$
Output: $\{\mathcal{T}^\ell\}$
```

Algorithm 6 CollectHierarchicBSplinesOnCell

```
Input: $(\boldsymbol{k}, \ell)$, $\{\mathcal{J}^m\} = \varnothing$
1: // get shape same-level non-zero b-splines of cell contained in active hb-splines
2: $\mathcal{J}^\ell = \mathcal{G}^\ell_{\boldsymbol{k}} \cap \mathcal{D}^\ell$
3: // find all non-zero hb-splines at coarser levels
4: for each $m = \ell - 1, \ldots, 0$ do
5:     for each $\boldsymbol{i} \in \mathcal{J}^{m+1}$ do
6:         $\mathcal{J}^m \cup= (\mathcal{A}^{m+1}_{\boldsymbol{i}} \cap \mathcal{D}^m)$
Output: $\{\mathcal{J}^m\}$
```

Algorithm 7 Unrefinement

```
Input: $\boxminus$, $\{\mathcal{I}^m\}$, $\{\mathcal{D}^m\}$, $(\boldsymbol{i}, \ell)$ with $\boldsymbol{i} \in \mathcal{I}^\ell \setminus \mathcal{D}^\ell$, $\{u^m_{\boldsymbol{j}}\}$
1: Unrefine($(\boldsymbol{i}, \ell)$, $\{\mathcal{I}^m\}$)
2: Compile($\{\mathcal{I}^m\}$, $\{\tilde{\mathcal{D}}^m\}$)
3: Shrink($\boxminus$, $\{\tilde{\mathcal{D}}^m\}$)
4: UnrefineCoefficients($\boxminus$, $\{\mathcal{D}^m\}$, $\{u^m_{\boldsymbol{j}}\}$, $\{\tilde{\mathcal{D}}^m\}$, $\{\tilde{u}^m_{\boldsymbol{j}}\}$)
5: $\{\mathcal{D}^m\} = \{\tilde{\mathcal{D}}^m\}$; $\{u^m_{\boldsymbol{j}}\} = \{\tilde{u}^m_{\boldsymbol{j}}\}$
Output: $\{\mathcal{I}^m\}$, $\{\mathcal{D}^m\}$, $\{u^m_{\boldsymbol{j}}\}$
```

Acknowledgement

The support of the EPSRC through grant # EP/G008531/1 is gratefully acknowledged.

References

[1] 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.

[2] F. Cirak, M. J. Scott, E. K. Antonsson, M. Ortiz, P. Schröder, Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision, Computer-Aided Design 34 (2002) 137–148.

[3] L. Piegl, T. W., The NURBS book, Monographs in visual communication, Springer, 2 edn., 1997.

[4] J. M. Lane, R. F. Riesenfeld, A theoretical development for the computer generation and display of piecewise polynomial surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2 (1980) 35–46.

[5] D. Zorin, P. Schröder, Subdivision for Modeling and Animation, SIGGRAPH 2000 Course Notes, 2000.

[6] J. Warren, H. Weimer, Subdivision methods for geometric design: A constructive approach, Morgan Kaufmann, 2001.

[7] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, in: SIGGRAPH 2003 Conference Proceedings, San Diego, CA, 2003.

[8] D. R. Forsey, R. H. Bartels, Hierarchical b-spline refinement, Computer Graphics 22 (1988) 205–212.

[9] S. J. Gortler, M. F. Cohen, Hierarchical and variational geometric modeling with wavelets, in: Proceedings of the 1995 symposium on interactive 3D graphics, 1995.

[10] M. Lounsbery, T. D. DeRose, J. Warren, Multiresolution analysis for surfaces of arbitrary topological type, ACM Transactions of Graphics 16 (1997) 34–73.

[11] 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 (2010) 229–263.

[12] D. Schillinger, E. Rank, An unfitted hp-adaptive finite element method based on hierarchical B-splines for interface problems of complex geometry, Computer Methods in Applied Mechanics and Engineering 200 (2011) 3358–3380.

[13] A.-V. Vuong, C. Gianelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 200 (2011) 3554–3567.

[14] F. Cirak, Q. Long, Subdivision shells with exact boundary control and non-manifold geometry, International Journal for Numerical Methods in Engineering 88 (2011) 897–923.

[15] F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: A new paradigm for thin-shell finite-element analysis, International Journal for Numerical Methods in Engineering 47 (2000) 2039–2072.

[16] R. Kraft, Adaptive and linearly independent multilevel B-splines, in: A. L. Méhauté, C. Rabut, L. L. Schumaker (Eds.), Surface Fitting and Multiresolution Methods, Vanderbilt University Press, 209–218, 1997.

[17] C. de Boor, A practical guide to splines, Springer, 2001.

[18] G. Farin, Curves and Surfaces for CAGD, Academic Press, 2002.

[19] K. Höllig, Finite Element Methods with B-splines, SIAM Frontiers in Applied Mathematics, 2003.

[20] E. Grinspun, P. Krysl, P. Schröder, CHARMS: a simple framework for adaptive simulation, in: SIGGRAPH 2002 Conference Proceedings, San Antonio, TX, 281–290, 2002.

[21] J. Nitsche, Über ein Variationsverfahren zur Lösung von Dirichlet-Problemen bei der Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 36 (1971) 9–15.

[22] A. Embar, J. Dolbow, I. Harari, Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements, International Journal for Numerical Methods in Engineering 83 (2010) 877–898.

[23] T. Rüberg, F. Cirak, Subdivision-stabilised immersed b-spline finite elements for moving boundary flows, Computer Methods in Applied Mechanics and Engineering 209–212 (2012) 266–283.

[24] R. A. K. Sanches, P. B. Bornemann, F. Cirak, Immersed b-spline (i-spline) finite element method for geometrically complex domains, Computer Methods in Applied Mechanics and Engineering 200 (2011) 1432–1445.

[25] S. Fernandez-Mendez, A. Huerta, Imposing essential boundary conditions in mesh-free methods, Computer Methods in Applied Mechanics and Engineering 193 (2004) 1257–1275.

[26] M. J. Borden, M. A. Scott, J. A. Evans, T. J. R. Hughes, Isogeometric fi-

Algorithm 8 Unrefine

```
Input: $(\boldsymbol{i}, \ell)$, $\{\mathcal{I}^m\}$
 1: for each $m = \ell, \ell + 1, \ldots$ do
 2:     // collect refined domain at level m
 3:     $\Box^m = \varnothing$
 4:     for each $\boldsymbol{k} \in \mathcal{I}^m$ do
 5:         $\Box^m \cup= \operatorname{supp} B^m_{\boldsymbol{k}}$
 6:     // remove unrefinement domain
 7:     $\Box^m \setminus= \operatorname{supp} B^\ell_{\boldsymbol{i}}$
 8:     // collect all coarsened indices on level m
 9:     $\mathcal{I}^m = \varnothing$
10:     for each $\operatorname{supp} B^m_{\boldsymbol{k}} \subset \Box^m$ do
11:         $\mathcal{I}^m \cup= \{\boldsymbol{k}\}$
Output: $\{\mathcal{I}^m\}$
```

Algorithm 9 UnrefineCoefficients

```
Input: $\boxminus$, $\{\mathcal{D}^m\}$, $\{u^m_{\boldsymbol{j}}\}$, $\{\tilde{\mathcal{D}}^m\}$, $\{\tilde{u}^m_{\boldsymbol{j}}\})$
 1: // least-square fit
 2: get  $\min_{\tilde{u}^m_{\boldsymbol{j}}, \boldsymbol{j} \in \tilde{\mathcal{D}}^m, m \geq 0} \int_{\boxminus} \frac{1}{2} \Big( \sum_{\boldsymbol{j} \in \tilde{\mathcal{D}}^m, m \geq 0} B^m_{\boldsymbol{j}} \tilde{u}^m_{\boldsymbol{j}}$
            $- \sum_{\boldsymbol{l} \in \mathcal{D}^n, n \geq 0} B^n_{\boldsymbol{l}} u^n_{\boldsymbol{l}} \Big)^2 \, \mathrm{d}\Box$
Output: $\{\tilde{u}^m_{\boldsymbol{j}}\}$
```

nite element data structures based on Bézier extraction of NURBS, International Journal for Numerical Methods in Engineering 87 (2011) 15–47.

[27] M. A. Scott, B. M. J., 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.

[28] T. J. R. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 199 (2010) 301–303.

[29] I. Babuska, Pollution error in the finite element method, Ticam Forum Notes 4, Texas Institute for Computational and Applied Mathematics, The University of Texas at Austin, 1997.

[30] F. Cirak, Adaptive Finite-Element-Methoden bei der nichtlinearen Analyse von Flächentragwerken, Report nr. 26, University of Stuttgart, 1998.

[31] C. Johnson, P. Hansbo, Adaptive finite element methods in computational mechanics, Computer Methods in Applied Mechanics and Engineering 101 (1992) 143–181.

[32] D. Peric, J. Yu, D. R. J. Owen, On error estimates and adaptivity in elastoplastic solids: Applications to the numerical simulation of strain localization in classical and Cosserat continua, International Journal for Numerical Methods in Fluids 37 (1994) 1351–1379.

[33] F. Cirak, E. Ramm, A posteriori error estimation and adaptivity for elastoplasticity using the reciprocal theorem, International Journal for Numerical Methods in Engineering 47 (2000) 379–393.

[34] D. Schillinger, L. Dede, 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 In press.

[35] E. Cohen, T. Lyche, R. Riesenfeld, Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics, Computer Graphics and Image Processing 14 (1980) 87–111.

[36] T. Cashman, U. Augsdörfer, N. Dodgson, M. Sabin, NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes, in: SIGGRAPH 2009 Conference Proceedings, New Orleans, LA, 2009.