Multi-level Bézier extraction for hierarchical local refinement of Isogeometric Analysis

Davide D'Angella *[1,2], Stefan Kollmannsberger[1], Ernst Rank[1,2], and Alessandro Reali[2,3]

[1]Chair for Computation in Engineering, Technische Universität München, Germany
[2]Institute for Advanced Study, Technische Universität München, Germany
[3]Dipartimento di Ingegneria Civile e Architettura, Università degli Studi di Pavia, Italy

Abstract

One of the main topics of research on Isogeometric Analysis is local refinement. Among the various techniques currently studied and developed, one of the most appealing, referred to as hierarchical B-Splines, consists of defining a suitable set of basis functions on different hierarchical levels. This strategy can also be improved, for example to recover partition of unity, resorting to a truncation operation, giving rise to the so-called truncated hierarchical B-Splines. Despite its conceptual simplicity, implementing the hierarchical definition of shape functions into an existing code can be rather involved. In this work we present a simple way to bring the hierarchical isogeometric concept closer to a standard finite element formulation. Practically speaking, the hierarchy of functions and knot spans is flattened into a sequence of elements being equipped with a standard single-level basis. In fact, the proposed multi-level extraction is a generalization of the classical Bézier extraction and analogously offers a standard element structure to the hierarchical overlay of functions. Moreover, this approach is suitable for an extension to non-linear problems and for a parallel implementation. The multi-level extraction is presented as a general concept that can be applied to different kinds of refinements and basis functions. Finally, few basic algorithms to compute the local multi-level extraction operator for knot insertion on spline spaces are outlined and compared, and some numerical examples are presented.

Keywords: isogeometric analysis, local refinement, (truncated) hierarchical B-Splines, Bézier extraction

*davide.dangella@tum.de, Corresponding Author

Preprint submitted to *Computer Methods in Applied Mechanics and Engineering*
The final publication is available at doi.org/10.1016/j.cma.2017.08.017

1 Introduction

Isogeometric Analysis (IGA) [1, 2] is a powerful approach for the numerical analysis of problems governed by partial differential equations (PDEs). Its aim is to bridge the gap between geometry and simulation by performing finite element analysis using the same functions that represent the geometry in Computer Aided Design (CAD). Classically, these include tensor product B-Splines and Non-Uniform Rational B-Splines (NURBS). Thanks to the high-regularity of the adopted basis functions, IGA has shown better accuracy per-degree-of-freedom with respect to standard finite elements in many applications ranging from solids and structures (cf., e.g., [3, 4, 5, 6]) to fluids (cf., e.g., [7, 8]) and fluid-structure interaction (cf., e.g., [9, 10]), opening also the door to geometrically flexible discretizations of higher-order PDEs in primal form (cf., e.g., [11, 12]).

Within this context, adaptivity has become a fundamental topic of research, as the tensor product structure of B-Splines and NURBS precludes local refinement. Therefore, different techniques are being currently developed, including T-Splines (see, e.g., [13, 14]), LR-Splines [15] and hierarchical B-Splines ($\mathcal{HB}$) (see, e.g., [16, 17, 18]). $\mathcal{HB}$ are based on the definition of a suitable set of basis functions on different hierarchical levels and look like one of the most promising ways to effectively implement local refinement in IGA. This strategy has been recently improved under the name of truncated hierarchical B-Splines ($\mathcal{THB}$) [19, 20], with the aim to recover partition of unity and improve the bandwidth granted by standard $\mathcal{HB}$. Following its mathematical formulation, these techniques can be implemented with ad-hoc algorithms (see, e.g., [21]). However, despite its conceptual simplicity, implementing the hierarchical definition of shape functions into an existing code can be rather involved.

In this work, we present a simple way to bring the hierarchical concept closer to a standard finite element formulation. From the practical point of view, the hierarchy of functions and knot spans is locally flattened into a sequence of elements being equipped with a standard single-level basis. In fact, the proposed multi-level extraction is a generalization of the classical Bézier extraction [22] and analogously offers a standard element structure to the hierarchical overlay of functions. The proposed approach is suitable for an extension to non-linear problems and for a parallel implementation. In addition, the multi-level extraction is presented as a general concept that can be applied to different kinds of refinements and basis functions.

In the literature, several works moving along analogous research lines can be found. In [23] a similar approach is applied globally level-wise, where the active entries of the system matrices are computed in a standard way for each level. Then, each level system matrix is transformed and assembled to the hierarchical system. One advantage of this approach is that no connectivity information is needed. However, for large and parallel applications a local method is more efficient. Moreover, for non-linearities, the solution or its derivatives have to be evaluated at each integration point. This is naturally a local operation that would be less efficient when following a global strategy. Another similar global approach can be found in [24], where the active entries of the system matrices K^l are computed for each level l, they are inserted in the diagonal of a diagonal block matrix K and, finally, K is transformed in the hierarchical system.

The advantages and drawbacks are similar to the ones discussed for [23]. In [24, 25], Bézier extraction is applied level-wise in a standard fashion, while we propose to apply it just once per element, combined with the multi-level extraction. In this way we extract at once all the hierarchical functions supporting the element. A very similar framework is presented in [26, 27], but it is restricted to the overlay of uniform knot vectors obtained by bisection, no knot repetition (in particular, no open-knot vector) is permitted and the approach does not extend to truncated hierarchies. Moreover, the method in [26] is developed with the restriction of [18], i.e., refinements on the boundary are not allowed. Here, we consider more general refinements, truncated hierarchies, knot repetitions, open-knot vectors and we provide alternative algorithms to produce the extraction operators. Another closely related concept is outlined in [28, 29, 30], but also here truncation is not considered and we conceptually separate the multi-level extraction from the standard Bézier extraction, setting a more general independent framework applicable to any sequence of nested space. In particular, we consider possibly different target hierarchical spaces, i.e., $\mathcal{THB}$, and different standard functions, like B-Splines, Bernstein or Lagrange polynomials. In addition, we present and compare different algorithms to compute the multi-level extraction operator for knot insertion on spline spaces.

The structure of the paper is as follows. Section 2 introduces the $\mathcal{HB}$ and $\mathcal{THB}$ refinement strategies together with associated fundamental concepts. Section 3 discusses the multi-level extraction and its basic algorithms. Finally, Section 4 presents some numerical experiments.

2 Preliminaries

In this section, we concisely discuss some preliminary concepts we will use throughout the paper. In particular, we present B-Splines and NURBS, the idea of Bézier extraction, as well as hierarchical splines.

2.1 B-Splines and NURBS

We herein briefly introduce the basic definitions and notations about B-Splines and NURBS. For further details, readers are referred to [31, 1], and references therein.

A B-Spline basis function of degree p is generated starting from a non-decreasing sequence of real numbers referred to as knot vector

$$\Xi = \{\xi_1, ..., \xi_{m+p+1}\}$$

where m is the number of basis functions (equal to the number of the associated control points). A univariate B-Spline basis function $N_{i,p}(\xi)$ can be then constructed using the following Cox-de Boor recursion formula: starting from $p = 0$, where

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

the basis functions for $p > 0$ are obtained from

$$b_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} b_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} b_{i+1,p-1}(\xi)$$

where the convention $0/0 = 0$ is assumed. Given the multiplicity k of a knot, the smoothness of the B-Spline basis is C^{p-k} at that location, while it is C^{∞} between knots. In so-called open knot vectors, the first and the last knots have multiplicity $k = p + 1$ and the basis is interpolatory at the ends of its definition domain. A B-Spline curve can be then constructed as the linear combination of the basis functions

$$\mathbf{C}(\xi) = \sum_{i=1}^{m} b_{i,p} \mathbf{P}_i$$

where the coefficients $\mathbf{P}_i \in \mathbb{R}^{d_s}$ of the linear combination are the so-called control points, being d_s the dimension of the physical space. Multivariate B-Splines are generated through the tensor product of univariate B-Splines. If d_p denotes the dimension of the parametric space, d_p univariate knot vectors are needed:

$$\Xi^d = \left\{ \xi_1^d, ..., \xi_{m_d+p_d+1}^d \right\}$$

where $d = 1, ..., d_p$, p_d is the polynomial degree in the parametric direction d, and m_d is the associated number of basis functions. Denoting the univariate basis functions in each parametric direction ξ^d by b_{i_d,p_d}^d, the multivariate basis functions $B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})$ are obtained as

$$B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) = \prod_{d=1}^{d_p} b_{i_d,p_d}^d \left(\xi^d \right)$$

where the multi-index $\mathbf{i} = \left\{ i_1, ..., i_{d_p} \right\}$ denotes the position in the tensor product structure, $\mathbf{p} = \{p_1, ..., p_d\}$ indicates the polynomial degrees, and $\boldsymbol{\xi} = \left\{ \xi^1, ..., \xi^{d_p} \right\}$ is the vector of the parametric coordinates in each parametric direction d. B-Spline surfaces and solids are obtained, for $d_p = 2$ and $d_p = 3$, respectively, from a linear combination of multivariate B-Spline basis functions and control points as follows

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\mathbf{i}} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) \mathbf{P}_{\mathbf{i}}$$

where the summation is extended to all combinations of the multi-index $\mathbf{i}$.

NURBS basis functions in $\mathbb{R}^{d_s}$ are obtained from a projective transformation of their B-Spline counterparts in $\mathbb{R}^{d_s+1}$. Univariate NURBS basis functions $r_{i,p}(\xi)$ are given by

$$r_{i,p}(\xi) = \frac{b_{i,p}(\xi) w_i}{\sum_{j=1}^{m} b_{j,p}(\xi) w_j}$$

where $b_{i,p}$ are B-Spline basis functions and w_i are the corresponding weights (i.e., the

(d_s+1)-th coordinates of the B-Spline control points in $\mathbb{R}^{d_s+1}$). Finally, multivariate NURBS basis functions are obtained as

$$R_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) = \frac{B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})\, w_{\mathbf{i}}}{\sum_{\mathbf{j}} B_{\mathbf{j},\mathbf{p}}(\boldsymbol{\xi})\, w_{\mathbf{j}}}$$

and NURBS surfaces and solids are constructed as

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\mathbf{i}} R_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})\, \mathbf{P}_{\mathbf{i}}.$$

In the following, when indicating B-Splines and NURBS, the degree p will be omitted from the notation.

2.2 Bézier extraction

We now briefly introduce the idea of Bézier extraction [22]. Consider a knot vector Ξ with associated B-Spline basis functions b_i and a spline curve $\boldsymbol{\tau} = \sum_i \boldsymbol{P}_i b_i$, defined by a set of control points $\boldsymbol{P}_i$. The Bézier decomposition of $\boldsymbol{\tau}$ into piecewise Bézier curves is obtained by raising the multiplicity of each internal knot in Ξ to p. In this way, $\boldsymbol{\tau}$ can be represented in terms of Bernstein polynomials B_i, as $\boldsymbol{\tau} = \sum_i \bar{\boldsymbol{P}}_i B_i$. The Bernstein control points $\bar{\boldsymbol{P}}_i$ can be computed as linear combination of the original control points $\boldsymbol{P}_i$ [22], which we can represent by the matrix operation

$$\bar{\boldsymbol{P}} = \boldsymbol{C}^\top \boldsymbol{P}, \tag{1}$$

where $\boldsymbol{P} = \{\boldsymbol{P_i}\}$, $\bar{\boldsymbol{P}} = \{\bar{\boldsymbol{P}}_{\boldsymbol{i}}\}$. The matrix $\boldsymbol{C}^\top$ is called Bézier extraction operator [22]. Moreover, using Equation (1) and $\boldsymbol{\tau} = \boldsymbol{b}^\top \boldsymbol{P} = \boldsymbol{B}^\top \bar{\boldsymbol{P}}$, $\boldsymbol{b} = \{b_i\}$, $\boldsymbol{B} = \{B_i\}$, we obtain

$$\begin{aligned}\boldsymbol{\tau} &= \boldsymbol{b}^\top \boldsymbol{P} = \boldsymbol{B}^\top \bar{\boldsymbol{P}} \\ &= \boldsymbol{B}^\top (\boldsymbol{C}^\top \boldsymbol{P}) \\ &= (\boldsymbol{C}\boldsymbol{B})^\top \boldsymbol{P}.\end{aligned}$$

The arbitrarity of $\boldsymbol{P}$ yields

$$\boldsymbol{b} = \boldsymbol{C}\boldsymbol{B}. \tag{2}$$

Equations (1) and (2) show that the Bézier extraction operator relates B-Splines with the Bernstein basis of the Bézier decomposition and also the control points of a curve with respect to each of these basis.

2.3 (Truncated) hierarchical B-Splines

In this section we introduce the hierarchical B-Spline basis $\mathcal{HB}$ and its truncated variant $\mathcal{THB}$ following closely [20, 24].

2.3.1 Overlay of univariate B-Splines

Let $V^0 \subset V^1 \subset \cdots \subset V^N$ be a sequence of nested spaces of univariate splines defined on a domain Ω. Each space V^l, $l = 0, \ldots, N$ is spanned by the normalized B-Spline basis $\mathcal{B}^l$ of degree p. Let $\Xi^l = (\xi_0^l, \ldots, \xi_{n^l}^l)$ be the knot vector composed of non-decreasing real numbers defining the basis $\mathcal{B}^l$. The nested nature of the spaces V^l implies that also the knot vectors are nested, i.e., any knot in Ξ^l occurs in Ξ^{l+1} with at least the same multiplicity, for $l = 0 \ldots N-1$. We denote this by $\Xi^l \subset \Xi^{l+1}$. For example, see Figure 1, where three nested knot vectors and their associated overlays of B-Spline functions are shown. In particular, note that each knot in a level l is present in every knot vector of level $l^* > l$.

Following the isoparametric paradigm, our final aim is to identify a set of functions $\mathcal{N} \subset \bigcup_l \mathcal{B}^l$ to be used as basis functions for both analysis and geometry description. To this end, within the tree-structure of knot spans, we define as elements $\mathcal{E}$ for the finite element analysis a partition of Ω composed of knot spans of any level. More specifically, let $\hat{\Xi}^l = (\hat{\xi}_0^l, \ldots, \hat{\xi}_{\hat{n}^l}^l)$ be the knot vector composed of non-decreasing knots of Ξ^l without repetition, for some $\hat{n}^l \in \mathbb{N}$, and let $\mathcal{Q}^l = \left\{ \mathcal{Q}_i^l \mid \mathcal{Q}_i^l = (\hat{\xi}_i^l, \hat{\xi}_{i+1}^l), i = 0, \ldots, \hat{n}^l - 1 \right\}$ be the set of open intervals constituting the non-empty knot spans of Ξ^l. The elements of the multi-level mesh can be any partition $\mathcal{E} \subset \bigcup_l \mathcal{Q}^l$ of Ω. In particular,

- $\emptyset \notin \mathcal{E}$,
- $\epsilon \in \mathcal{E} \implies \epsilon \in \mathcal{Q}^l$, for some $0 \leq l \leq N$,
- $\bigcup_{\epsilon \in \mathcal{E}} \bar{\epsilon} = \Omega$,
- $\epsilon_1 \cap \epsilon_2 = \emptyset, \ \forall \epsilon_1, \epsilon_2 \in \mathcal{E}, \ \epsilon_1 \neq \epsilon_2$.

Here, $\bar{\epsilon}$ denotes the closure of ϵ. Note that we do not consider as elements all the non-empty spans of all levels, but a subset of them that partitions the domain. Moreover, let $\mathcal{E}^l = \mathcal{E} \cap \mathcal{Q}^l$ be the elements of level l, and let $\Omega^l = \bigcup_{\epsilon \in \mathcal{E}^l} \bar{\epsilon}$ be their domain. Furthermore, let $\Omega_+^l = \bigcup_{l^*=l+1}^N \Omega^l$ be the refined domain with respect to level l and, analogously, let $\Omega_-^l = \bigcup_{l^*=0}^{l-1} \Omega^l$ be the coarser domain. For example, Figure 2 shows a possible choice of elements for the knots depicted in Figure 1, where we have

$$
\begin{array}{lll}
\Omega^0 = [-1, 0], & \Omega^1 = [0, 0.25], & \Omega^2 = [0.25, 1], \\
\Omega_+^0 = [0, 1], & \Omega_+^1 = [0.25, 1], & \Omega_+^2 = \emptyset, \\
\Omega_-^0 = \emptyset, & \Omega_-^1 = [-1, 0], & \Omega_-^2 = [-1, 0.25].
\end{array}
$$

Given a set of elements $\mathcal{E}$, we still need to define a set of basis functions $\mathcal{N} \subset \bigcup_l \mathcal{B}^l$ suitable for analysis. To this end, we consider the set of functions with support on the elements as the set of *active* functions $\mathcal{B}_a^l = \left\{ b \,|b \in \mathcal{B}^l, \operatorname{supp}(b) \cap \Omega^l \neq \emptyset \right\} \subset \mathcal{B}^l$ (see Figure 2). Among these, a subset of linearly independent functions has to be chosen.

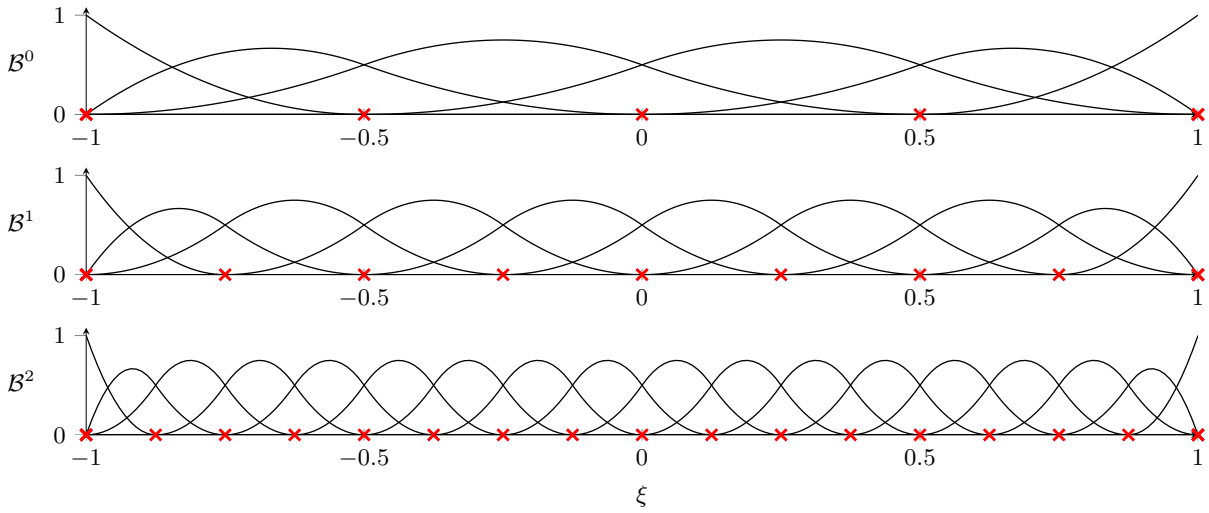


Figure 1: Example of hierarchical B-Spline functions. The knots of each level are marked by red crosses.

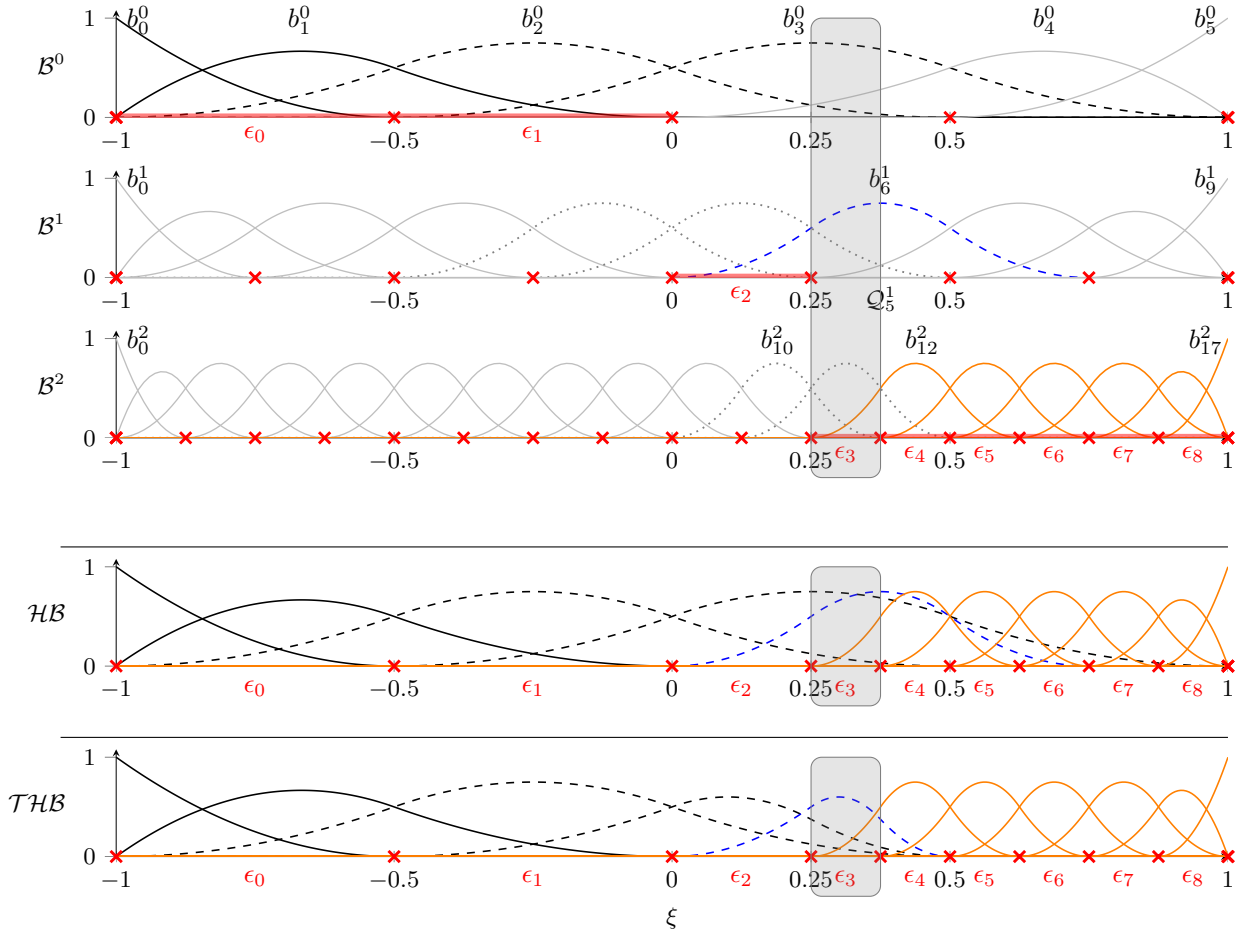


Figure 2: Example of element mesh with associated $\mathcal{HB}$- and $\mathcal{THB}$-Spline basis. The elements are marked by the thick red lines on the horizontal axes. $\mathcal{B}_a^l$ is composed of the colored and of the dotted functions. $\mathcal{B}_-^l$ is composed of the dotted functions. $\mathcal{B}_+^l$ is composed of the dashed functions. $\mathcal{B}_=^l$ is composed of the solid non-gray functions. The solid gray functions are inactive.

Therefore, we partition $\mathcal{B}_a^l$ into

$$\begin{aligned}
\mathcal{B}_-^l &= \left\{ b \mid b \in \mathcal{B}_a^l,\, \mathrm{supp}(b) \cap \Omega_-^l \neq \emptyset \right\} && \textit{(overlap with coarser elements),} \\
\mathcal{B}_+^l &= \left\{ b \mid b \in \mathcal{B}_a^l,\, \mathrm{supp}(b) \cap \Omega_+^l \neq \emptyset \right\} \setminus \mathcal{B}_-^l && \textit{(overlap with finer elements, but not with coarser),} \\
\mathcal{B}_=^l &= \left\{ b \mid b \in \mathcal{B}_a^l,\, \mathrm{supp}(b) \subset \Omega^l \right\} && \textit{(overlap just with elements of level } l\textit{),}
\end{aligned}$$

as shown in Figure 2. Such a classification is analogous to the one provided in [24].

We are now ready to introduce the Hierarchical B-Splines Basis ($\mathcal{HB}$) and the Truncated Hierarchical B-Splines Basis ($\mathcal{THB}$).

2.3.2 Hierarchical B-Spline basis

The hierarchical B-Splines basis $\mathcal{HB}$ [32, 18] is defined by

$$\begin{aligned}
\mathcal{HB} &= \bigcup_l \mathcal{HB}^l\,, \\
\mathcal{HB}^l &= \left(\mathcal{B}_=^l \cup \mathcal{B}_+^l \right).
\end{aligned} \tag{3}$$

Namely, $\mathcal{HB}$ is the set of B-Splines of each level l whose support covers just elements of level $l^* \geq l$ and at least one element of level l (See Figure 2). It was proven in [32] that this set is composed of linearly independent functions.

2.3.3 Truncated hierarchical B-Splines basis

The truncated hierarchical B-Spline basis $\mathcal{THB}$ [19] is very similar to $\mathcal{HB}$, with the only difference that the basis functions whose support overlaps finer elements are truncated, as described in [19]. This generates a basis that spans the same space as $\mathcal{HB}$ [19], but is composed of functions that

- have smaller support,
- form a partition of unity,
- have superior stability properties.

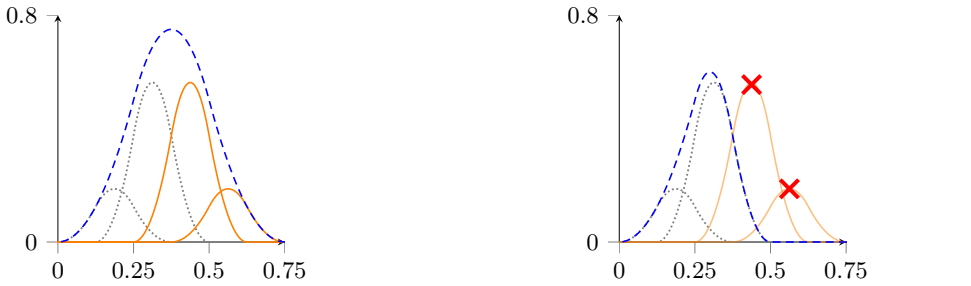
The above properties are all desirable in the context of numerical simulations. See [33] for further details.

The truncation operation is based on the following definition.

Definition 2.1 (Truncation operator [19]) *Let $\tau \in V^l$ and let*

$$\tau = \sum_{b \in \mathcal{B}^{l+1}} c_b^{l+1}(\tau)\, b, \qquad\qquad c_b^{l+1} \in \mathbb{R}\,, \tag{4}$$

be its representation with respect to the finer basis $\mathcal{B}^{l+1}$ of V^{l+1}. The truncation of τ



(a) $b_6^1 = \frac{1}{4}b_{10}^2 + \frac{3}{4}b_{11}^2 + \frac{3}{4}b_{12}^2 + \frac{1}{4}b_{13}^2$. (b) $trunc(b_6^1) = \frac{1}{4}b_{10}^2 + \frac{3}{4}b_{11}^2 + 0\,b_{12}^2 + 0\,b_{13}^2$.

Figure 3: The function b_6^1 of the example in Figure 2 (left, dashed blue) and its truncation $trunc(b_6^1)$ (right, dashed blue) expressed in terms of the finer functions (from left to right): b_{10}^2, b_{11}^2, b_{12}^2, b_{13}^2 (dotted gray and solid orange).

with respect to the level $l+1$ *is defined as*

$$trunc^{l+1}(\tau) = \sum_{\substack{b \in \mathcal{B}^{l+1}, \\ \text{supp}(b) \cap \Omega_-^{l+1} \neq \emptyset}} c_b^{l+1}(\tau)\, b.$$

Namely, the truncation takes the representation of τ in the finer basis and considers just the contributions of the basis functions whose support overlaps coarser elements (see an example in Figure 3). Note that Equation (4) is valid, as $V^l \subset V^{l+1}$.

Finally, we can define the recursive truncation operator as follows.

Definition 2.2 (Recursive truncation operator [34]) *Let* $\tau \in V^l$ *and let* $\mathcal{A} \subset \bigcup_l \mathcal{B}^l$. *The recursive truncation of* τ *and* $\mathcal{A}$ *with respect to the level* $l+1$ *is defined as*

$$\begin{aligned} Trunc^{l+1}(\tau) &= trunc^N(trunc^{N-1}(\dots\, trunc^{l+1}(\tau)\, \dots)), \\ Trunc^{l+1}(\mathcal{A}) &= \left\{ Trunc^{l+1}(a) \;|a \in \mathcal{A} \right\}. \end{aligned}$$

The definition of the $\mathcal{THB}$ basis is analogous to $\mathcal{HB}$, with the difference that active functions with support partly on finer elements will be truncated

$$\begin{aligned} \mathcal{THB} &= \bigcup_l \mathcal{THB}^l, \\ \mathcal{THB}^l &= \mathcal{B}_=^l \cup Trunc^{l+1}(\mathcal{B}_+^l) \\ &= Trunc^{l+1}(\mathcal{HB}^l). \end{aligned} \tag{5}$$

See the example in Figure 2.

3 Multi-level extraction operator

In this section we introduce the multi-level extraction operator and its variants. Before discussing the operator in a general setting, we start with an example that shows the key idea and its use in practical applications.

3.1 Basic concept

Consider the $\mathcal{HB}$ basis depicted in Figure 2 and the element $\epsilon_3 = (0.25, 0.375)$ of level 2 marked by the gray overlay box. The basis functions in $\mathcal{HB}$ restricted to ϵ_3 can be all obtained by a linear combination of functions in $\mathcal{B}^2$ with support on ϵ_3. Namely, the element-local hierarchical basis of ϵ_3 are a linear combination of the standard B-Splines defined by the knot vector Ξ^2 of level 2 . This is also depicted in Figure 4. This turns out to hold for each element ϵ of level l: the element-local hierarchical basis can be represented on ϵ as a linear combination of standard B-Spline functions $\mathcal{B}^l$ of level l. This concept will be further explained later in the section. Note that the B-Spline basis is taken of the same level as the element ϵ. As shown in Figure 4, this linear operation can be represented by a matrix local to each element called *multi-level extraction operator*.

In a certain sense, such an operator flattens the multi-level hierarchy of functions into a sequence of elements equipped with a single-level basis. Therefore, the multi-level extraction operator offers a classical finite element point of view on the hierarchical overlay of functions, that can simplify its implementation in an already existing finite element software. This benefit will become even more apparent, when the multi-level extraction is combined with the Bézier extraction [22], as shortly discussed in Section 3.7 and illustrated in Figure 4. Further advantages will be highlighted in Section 3.11.

Moreover, this tool can be defined not just for nested spline spaces produced by knot insertion, as in Figure 2, but in a more general sense for every sequence of nested spaces $V^0 \subset V^1 \subset \cdots \subset V^N$. In particular, different basis functions and different kind of refinements can be considered, e.g, (anisotropic) spline knot insertion, (anisotropic) spline degree elevation, (anisotropic) C^0-continuous linear polynomial and h-FEM refinement, or (anisotropic) C^0-continuous high-order polynomials and hp-FEM refinement (see, e.g., [35]).

3.2 Linear transformation between basis functions of nested spaces

In this section, we briefly discuss the relation between basis of nested spaces. This concept represent the core idea underlying the multi-level extraction operator.

Since the spaces $V^0 \subset V^1 \subset \cdots \subset V^N$ are nested, every basis function of V^{l_1} can be expressed as a linear combination of basis functions of V^{l_2} for any $l_2 \geq l_1$. In particular, let $\mathcal{B}^{l_1}$ and $\mathcal{B}^{l_2}$ be any two bases for V^{l_1} and V^{l_2}, respectively. Let $m^l = dim(V^l)$ and let $\boldsymbol{b}^l$ be the column vector of B-Splines composed of the m^l functions in $\mathcal{B}^l$ in some fixed order. Then, there exists a matrix $\boldsymbol{R}^{l_1,l_2}$ of dimension $m^{l_1} \times m^{l_2}$, called *refinement operator*, such that

$$\boldsymbol{b}^{l_1} = \boldsymbol{R}^{l_1,l_2} \boldsymbol{b}^{l_2}. \tag{6}$$

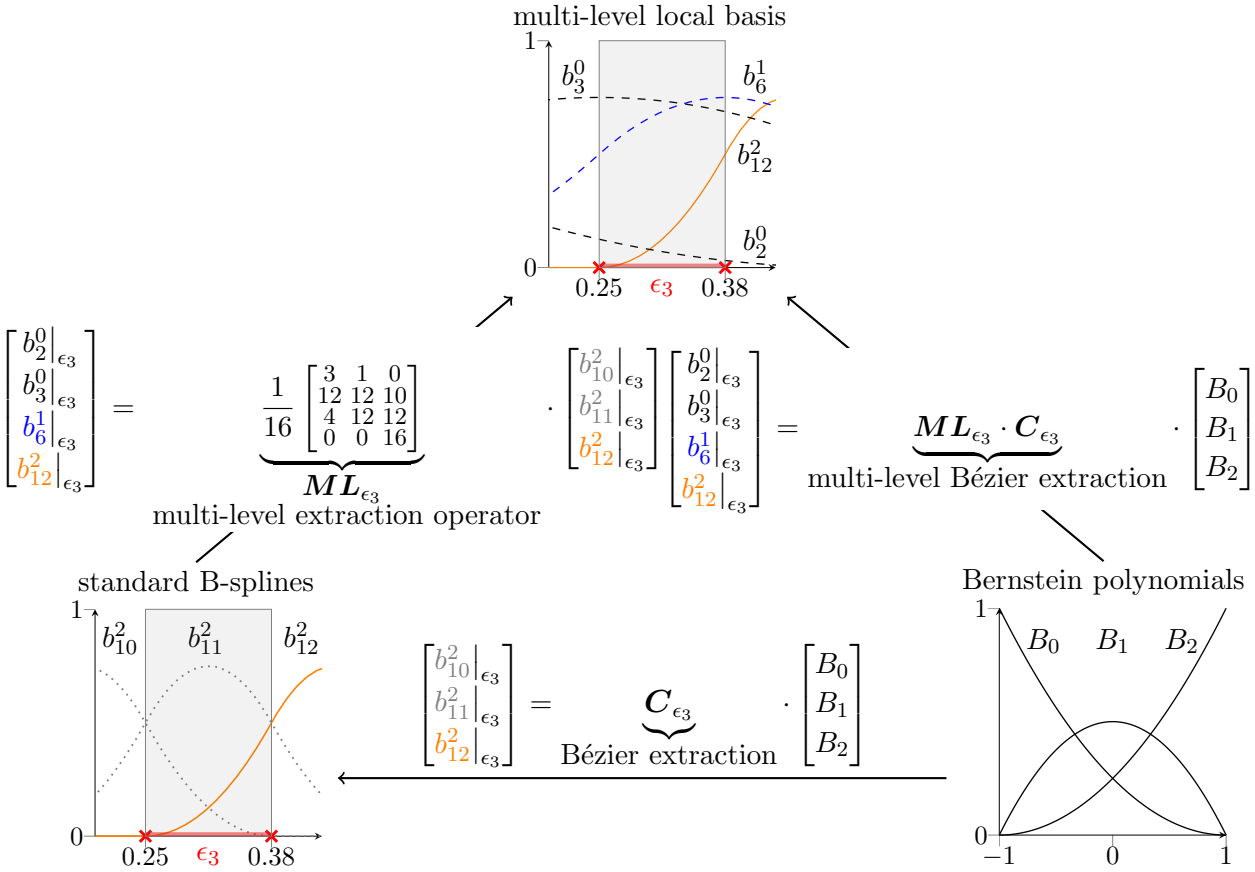


Figure 4: Basis functions of $\mathcal{HB}$ local to the element $\epsilon_3 = (0.25, 0.375)$ of the example in Figure 2,expressed as linear combination of standard functions in $\mathcal{B}^2$ with support on ϵ_3 or of Bernstein polynomials on the reference element.

Furthermore, given a spline $\boldsymbol{\tau} \in V^{l_1}$, $\boldsymbol{\tau} = \boldsymbol{b}^{l_1\top}\boldsymbol{P}^{l_1}$ for some coefficients $\boldsymbol{P}^{l_1}$. Again, $V^{l_1} \subset V^{l_2}$ implies that $\boldsymbol{\tau} = \boldsymbol{b}^{l_2\top}\boldsymbol{P}^{l_2}$ for some coefficients $\boldsymbol{P}^{l_2}$ and we can write

$$\boldsymbol{\tau} = \boldsymbol{b}^{l_2\top}\boldsymbol{P}^{l_2} = \boldsymbol{b}^{l_1\top}\boldsymbol{P}^{l_1} = \boldsymbol{b}^{l_2\top}\boldsymbol{R}^{l_1,l_2\top}\boldsymbol{P}^{l_1}.$$

The linear independence of $\boldsymbol{b}^{l_2}$ yields the dual relation

$$\boldsymbol{P}^{l_2} = \boldsymbol{R}^{l_1,l_2\top}\boldsymbol{P}^{l_1}. \tag{7}$$

Equations (6) and (7) generalize Equations (2) and (1), respectively.

3.3 The global multi-level extraction for $\mathcal{HB}$ and knot insertion

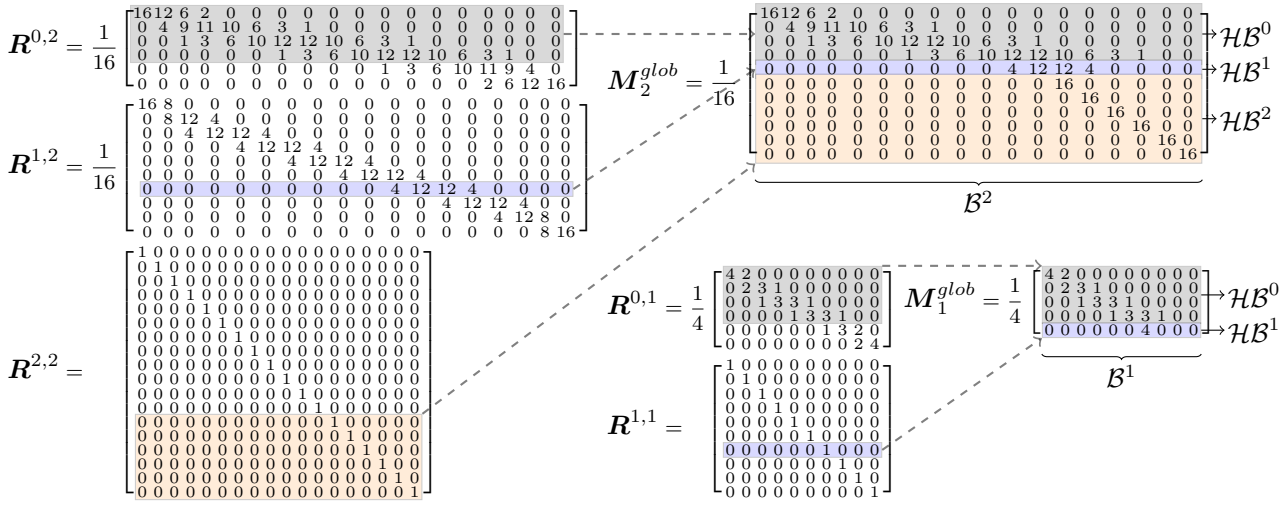
In this section, we introduce the concept of *global* multi-level extraction for the hierarchical basis $\mathcal{N} = \mathcal{HB}$ associated to a sequence of nested spline spaces $V^0 \subset V^1 \subset \cdots \subset V^N$

of fixed degree p obtained by knot insertion. Let $\mathcal{B}^l$ be the B-Spline basis of V^l. Then $\boldsymbol{R}^{l_1,l_2}$ of Equation (6) is known from the literature, e.g., see [36], and it will be referred as the knot insertion operator. It can be computed by standard techniques, e.g., by the Boehm's algorithm or the Oslo algorithm [36].

The global multi-level extraction operator $\boldsymbol{M}_N^{glob}$ is the linear operator that transforms $\mathcal{B}^N$ into the hierarchical basis $\mathcal{N}$. From Equation (6), it is possible to associate each row of $\boldsymbol{R}^{l_1,l_2}$ to a function in $\mathcal{B}^{l_1}$, and each column to a function in $\mathcal{B}^{l_2}$. For example, in the following picture we show the operator $\boldsymbol{R}^{0,2}$ for Figure 2, where the row and column labels indicate the associated functions.

$$\boldsymbol{R}^{0,2} = \frac{1}{16}\begin{bmatrix}
16 & 12 & 6 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 9 & 11 & 10 & 6 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 3 & 6 & 10 & 12 & 12 & 10 & 6 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 10 & 12 & 12 & 10 & 6 & 3 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 10 & 11 & 9 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6 & 12 & 16
\end{bmatrix}
\begin{matrix}
\leftarrow b_0^0 \\ \leftarrow b_1^0 \\ \leftarrow b_2^0 \\ \leftarrow b_3^0 \\ \leftarrow b_4^0 \\ \leftarrow b_5^0
\end{matrix}
\left.\begin{matrix}\\\\\\\\\\\\\end{matrix}\right\} \mathcal{B}^0 \left.\begin{matrix}\\\\\\\\\\\\\end{matrix}\right\} \mathcal{HB}^0$$
$$\underbrace{\begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ b_0^2 & b_1^2 & b_2^2 & b_3^2 & b_4^2 & b_5^2 & b_6^2 & b_7^2 & b_8^2 & b_9^2 & b_{10}^2 & b_{11}^2 & b_{12}^2 & b_{13}^2 & b_{14}^2 & b_{15}^2 & b_{16}^2 & b_{17}^2 \end{matrix}}_{\mathcal{B}^2}$$

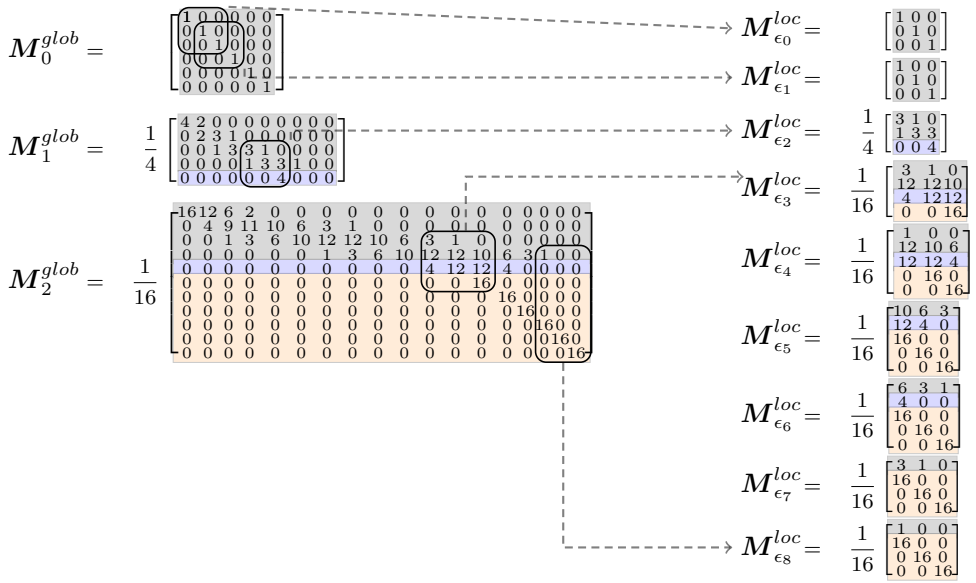
Therefore, $\boldsymbol{M}_N^{glob}$ can be simply obtained by joining the rows of the operators $\boldsymbol{R}^{l,N}$, $l = 0, \ldots, N$, associated to the basis functions in $\mathcal{N}$. For instance, considering again the example in Figure 2, we have:



The highlighted rows of $\boldsymbol{R}^{0,2}$ correspond to the functions $\mathcal{HB}^0 = \{b_0^0, b_1^0, b_2^0, b_3^0\}$, the highlighted row of $\boldsymbol{R}^{1,2}$ is associated to $\mathcal{HB}^1 = \{b_6^1\}$, and the highlighted rows of $\boldsymbol{R}^{2,2}$ correspond to $\mathcal{HB}^2 = \{b_{12}^2, b_{13}^2, \ldots, b_{17}^2\}$. Analogously, it is possible to define the operator $\boldsymbol{M}_L^{glob}$ that transforms $\mathcal{B}^L$ into $\bigcup_{l=0}^{L} \mathcal{N}^l$, $\mathcal{N}^l = \mathcal{B}^l \cap \mathcal{N}$, i.e., into the basis functions of $\mathcal{N}$ up to level L (see $\boldsymbol{M}_1^{glob}$ in the example above). This will be useful in the next section, where to each element of level L is assigned a localization of the global operator $\boldsymbol{M}_L^{glob}$ of the same level.

3.4 The local multi-level extraction for $\mathcal{HB}$ and knot insertion

The global multi-level extraction operator $\boldsymbol{M}_L^{glob}$ can then be localized to each element ϵ of level L by simply selecting the columns associated to the functions in $\mathcal{B}^L$ with support on ϵ and the rows corresponding to the functions in $\mathcal{N}$ with support on ϵ. Note that the multi-level extraction operator is always of the same level L as the element. Considering again the example in Figure 2, we have:



Note that the operator $\boldsymbol{M}_{\epsilon_3}^{loc}$ is the same as the one previously shown in Figure 4. The localization of the multi-level extraction operator is analogous to the localization of the Bézier extraction operator in [22].

3.5 The multi-level extraction operator

In this section, we define more formally the concept of multi-level extraction operator which was intuitively introduced in the previous sections. In the following, we denote with $\#A$ the number of elements in a finite set A. We start with the two following preliminary definitions:

Definition 3.1 (Restriction of a set of functions to a sub-domain) *Given $\Omega \subset \mathbb{R}$, let $\mathcal{A} = \{a_0, \dots, a_m\}$ be a set of functions $a_i : \Omega \to \mathbb{R}$. Given a sub-domain $\mathcal{Q} \subset \Omega$, the restriction $\mathcal{A}_\mathcal{Q}$ of $\mathcal{A}$ to $\mathcal{Q}$ is*

$$\mathcal{A}_\mathcal{Q} = \{a \,|\, a \in \mathcal{A}, \mathrm{supp}(a) \cap \mathcal{Q} \neq \emptyset\}\,.$$

Namely, it is the subset of functions with support on $\mathcal{Q}$.

Definition 3.2 (Restrictions of a refinement operator) *Let $\Omega \subset \mathbb{R}$. Let $\mathcal{B}^0 = \left\{b_0^0, \dots, b_{m^0}^0\right\}$ and $\mathcal{B}^1 = \left\{b_0^1, \dots, b_{m^1}^1\right\}$ be sets of functions $b_i^l : \Omega \to \mathbb{R}$. Let $\boldsymbol{b}^l =$*

$(b_0^l, \dots, b_{m^l}^l)^\top$ *be a column vector containing all the functions of* $\mathcal{B}^l$*. Let* $\boldsymbol{R}$ *be a real* $m^0 \times m^1$ *matrix such that*

$$\boldsymbol{b}^0 = \boldsymbol{R}^{0,1}\boldsymbol{b}^1.$$

Then, given a set $\mathcal{A}^0 \subset \mathcal{B}^0$ *we define the restriction* $\boldsymbol{R}_{\mathcal{A}^0}$ *of* $\boldsymbol{R}$ *to* $\mathcal{A}^0$*, as the* $\#\mathcal{A}^0 \times m^1$ *sub-matrix obtained by selecting the rows with indices* $r_0, \dots, r_{\#\mathcal{A}^0}$ *such that* $b_{r_i}^0 \in \mathcal{A}^0$ *,* $0 \le i \le \#\mathcal{A}^0$*. Namely,* $\boldsymbol{R}_{\mathcal{A}^0}^{0,1}$ *is the sub-matrix composed of rows associated to functions in* $\mathcal{A}^0$*.*

Moreover, given the sets $\mathcal{A}^0 \subset \mathcal{B}^0$*,* $\mathcal{A}^1 \subset \mathcal{B}^1$*, we define the restriction* $\boldsymbol{R}_{(\mathcal{A}^0,\mathcal{A}^1)}$ *of* $\boldsymbol{R}$ *to* $\mathcal{A}^0$ *and* $\mathcal{A}^1$*, as the* $\#\mathcal{A}^0 \times \#\mathcal{A}^1$ *sub-matrix obtained by selecting the rows with indices* $r_0, \dots, r_{\#\mathcal{A}^0}$ *and columns with indices* $c_0, \dots, c_{\#\mathcal{A}^1}$ *such that* $b_{r_i}^0 \in \mathcal{A}^0$ *and* $b_{c_j}^1 \in \mathcal{A}^1$*,* $0 \le i \le \#\mathcal{A}^0$*,* $0 \le j \le \#\mathcal{A}^1$*. Namely,* $\boldsymbol{R}_{(\mathcal{A}^0,\mathcal{A}^1)}^{0,1}$ *is the sub-matrix composed of rows associated to functions of* $\mathcal{A}^0$ *and columns associated to functions of* $\mathcal{A}^1$*.*

We are now ready to formally define the multi-level extraction operator as follows:

Definition 3.3 (The multi-level extraction operator) *Let* $V^0 \subset V^1 \subset \cdots \subset V^N$ *be a sequence of nested subspaces defined on a domain* Ω*. Let* $\mathcal{B}^l$ *be a basis of* V^l *and* $\boldsymbol{b}^l$ *a column vector of functions in* $\mathcal{B}^l$*. Furthermore, let* $\boldsymbol{R}^{l_1,l_2}$ *be the refinement operator such that* $\boldsymbol{b}^{l_1} = \boldsymbol{R}^{l_1,l_2}\boldsymbol{b}^{l_2}$*,* $l_2 \ge l_1$*. Given* $\mathcal{N} \subset \bigcup_l \mathcal{B}^l$*, e.g.,* $\mathcal{N} = \mathcal{HB}$ *or* $\mathcal{N} = \mathcal{THB}$*, let* $\mathcal{N}^l = \mathcal{N} \cap \mathcal{B}^l$ *be the functions in* $\mathcal{N}$ *of level* l*. Then, we define the global multi-level extraction operator* $\boldsymbol{M}_N^{glob}$ *as the matrix*

$$\boldsymbol{M}_N^{glob} = \begin{bmatrix} \boldsymbol{R}_{\mathcal{N}^0}^{0,N} \\ \vdots \\ \boldsymbol{R}_{\mathcal{N}^N}^{N,N} \end{bmatrix}.$$

Furthermore, given an element $\epsilon \subset \Omega$ *and a level* L*, we define the local multi-level extraction operator* $\boldsymbol{M}_{L,\epsilon}^{loc}$ *as*

$$\boldsymbol{M}_{L,\epsilon}^{loc} = \begin{bmatrix} \boldsymbol{R}_{\epsilon}^{0,L} \\ \vdots \\ \boldsymbol{R}_{\epsilon}^{L,L} \end{bmatrix},$$

where $\boldsymbol{R}_{\epsilon}^{l,L} = \boldsymbol{R}_{(\mathcal{N}_\epsilon^l,\, \mathcal{B}_\epsilon^L)}^{l,L}$*. Namely* $\boldsymbol{M}_{L,\epsilon}^{loc}$ *is the sub-matrix obtained from* $\boldsymbol{M}_L^{glob}$ *by selecting rows and columns associated to functions with support on* ϵ*.*

The global multi-level extraction operator $\boldsymbol{M}_L^{glob}$ is simply obtained by joining the rows of the operators $\boldsymbol{R}^{l,N}$, $l = 0, \dots, N$, associated to the basis functions in $\mathcal{N}$. It allows to represent the target basis $\mathcal{N}$ for analysis, in terms of the classical functions of the finest level. Indeed, denoting by $\boldsymbol{n}^l$ the column vector of functions in $\mathcal{N}^l$ sorted consistently

with $\boldsymbol{R}^{l,N}_{\mathcal{N}^l}$, we can write

$$\boldsymbol{n} = \begin{bmatrix} \boldsymbol{n}^0 \\ \vdots \\ \boldsymbol{n}^N \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{0,N}_{\mathcal{N}^0}\ \boldsymbol{b}^N \\ \vdots \\ \boldsymbol{R}^{N,N}_{\mathcal{N}^N}\ \boldsymbol{b}^N \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{0,N}_{\mathcal{N}^0} \\ \vdots \\ \boldsymbol{R}^{N,N}_{\mathcal{N}^N} \end{bmatrix} \boldsymbol{b}^N = \boldsymbol{M}^{glob}_N\, \boldsymbol{b}^N.$$

Furthermore, for a $0 \leq L \leq N$, it holds

$$\begin{bmatrix} \boldsymbol{n}^0 \\ \vdots \\ \boldsymbol{n}^L \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{0,L}_{\mathcal{N}^0} \\ \vdots \\ \boldsymbol{R}^{L,L}_{\mathcal{N}^L} \end{bmatrix} \boldsymbol{b}^L = \boldsymbol{M}^{glob}_L\, \boldsymbol{b}^L.$$

Denoting with $V = \operatorname{span}\mathcal{N} \subset V^N$ the final solution space and with $\boldsymbol{n}$ the column vector of functions in $\mathcal{N}$, $\boldsymbol{M}^{glob}_N$ provides the dual relation analogous to Equation (7). Indeed, given a spline $\boldsymbol{\tau} \in V$, with $\boldsymbol{\tau} = \boldsymbol{n}^\top \boldsymbol{P}_{\boldsymbol{n}}$ for some coefficients $\boldsymbol{P}_{\boldsymbol{n}}$, $\boldsymbol{\tau}$ can also be written as $\boldsymbol{\tau} = {\boldsymbol{b}^N}^\top \boldsymbol{P}_{\boldsymbol{b}^N}$ for some other coefficients $\boldsymbol{P}_{\boldsymbol{b}^N}$ and it holds

$$\boldsymbol{\tau} = {\boldsymbol{b}^N}^\top \boldsymbol{P}_{\boldsymbol{b}^N} = \boldsymbol{n}^\top \boldsymbol{P}_{\boldsymbol{n}} = {\boldsymbol{b}^N}^\top {\boldsymbol{M}^{glob}_N}^\top \boldsymbol{P}_{\boldsymbol{n}}.$$

Again, the linear independence of $\boldsymbol{b}^N$ yields the dual relation

$$\boldsymbol{P}_{\boldsymbol{b}^N} = {\boldsymbol{M}^{glob}_N}^\top \boldsymbol{P}_{\boldsymbol{n}}. \tag{8}$$

The local multi-level extraction operator associated to one element ϵ of level L is simply obtained by extracting the smallest sub-operator that interests ϵ. It allows to represent the local basis $\mathcal{N}_\epsilon$ in terms of the classical functions of the same level L as the element. In particular, letting $\boldsymbol{n}_\epsilon$, $\boldsymbol{n}^l_\epsilon$ and $\boldsymbol{b}^L_\epsilon$ be appropriate column vectors of functions in $\mathcal{N}_\epsilon$, $\mathcal{N}^l_\epsilon$ and $\mathcal{B}^L_\epsilon$, respectively, we have

$$\boldsymbol{n}_\epsilon(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{n}^0_\epsilon(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{n}^L_\epsilon(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{0,L}_\epsilon\ \boldsymbol{b}^L_\epsilon(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{R}^{L,L}_\epsilon\ \boldsymbol{b}^L_\epsilon(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{0,L}_\epsilon \\ \vdots \\ \boldsymbol{R}^{L,L}_\epsilon \end{bmatrix} \boldsymbol{b}^L_\epsilon(\boldsymbol{x}) = \boldsymbol{M}^{loc}_{L,\epsilon}\ \boldsymbol{b}^L_\epsilon(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \epsilon. \tag{9}$$

Note that the multi-level extraction operator is always of the same level L as the element. The dual relation (8) holds also locally. In particular, $\tau \in V$, $\tau(\boldsymbol{x}) = \boldsymbol{n}_\epsilon(\boldsymbol{x})^\top \boldsymbol{P}_{\boldsymbol{n}_\epsilon} = {\boldsymbol{b}^L_\epsilon(\boldsymbol{x})}^\top \boldsymbol{P}_{\boldsymbol{b}^L_\epsilon}$ for $\boldsymbol{x} \in \epsilon$ and some coefficients $\boldsymbol{P}_{\boldsymbol{n}_\epsilon}$, $\boldsymbol{P}_{\boldsymbol{b}^L_\epsilon}$. As before, we obtain

$$\boldsymbol{P}_{\boldsymbol{b}^L_\epsilon} = {\boldsymbol{M}^{loc}_{L,\epsilon}}^\top \boldsymbol{P}_{\boldsymbol{n}_\epsilon}. \tag{10}$$

Equation (10) can be used for translating degrees of freedom (for the solution description) or control points (for the geometry description) between the multi-level local basis $\boldsymbol{n}_\epsilon$ and the standard single-level basis $\boldsymbol{b}^L_\epsilon$. A similar interpretation can be drawn for

Equation (8). The above relations generalize the local and global version of the properties of the Bézier extraction given in Equations (2) and (1) and in [22]. As examples, see Figures 8 and 11.

3.6 The multi-level extraction for $\mathcal{THB}$ and knot insertion

Following [19, 20], it can be deduced directly from Definitions 2.1 and 2.2 that the refinement operators $\boldsymbol{R}^{\mathcal{THB};\, l_1,l_2}$, $l_2 \geq l_1$, for the truncated basis can be obtained from the standard knot insertion operators $\boldsymbol{R}^{l,l+1}$ of consecutive levels

$$\boldsymbol{R}^{\mathcal{THB};\, l_1,l_2} = \begin{cases} \boldsymbol{I} & \text{if } l_2 = l_1, \\ trunc(\boldsymbol{R}^{l_1,l_1+1}) \cdot \boldsymbol{R}^{\mathcal{THB};\, l_1+1,l_2} & \text{otherwise,} \end{cases}$$

$$\Big[trunc(\boldsymbol{R}^{l,l+1})\Big]_{ij} = \begin{cases} \big[\boldsymbol{R}^{l,l+1}\big]_{ij} & \text{if } b_j \in \mathcal{B}^{l+1} \text{ and } \mathrm{supp}(b_j) \cap \Omega_-^{l+1} \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

where $\boldsymbol{I}$ is the identity matrix of size $\#\mathcal{B}^{l1} \times \#\mathcal{B}^{l1}$.

The global and local multi-level extraction operator for the $\mathcal{THB}$ is then obtained by Definition 3.3 using the refinement operators $\boldsymbol{R}^{\mathcal{THB};\, l_1,l_2}$.

3.6.1 Local computation of the local multi-level extraction operator

Note that the local truncated operator $\boldsymbol{R}_\epsilon^{\mathcal{THB};\, l_1,l_2}$ of an element ϵ can be constructed directly from multiplication of the local truncated version of the operators $\boldsymbol{R}^{l,l+1}$ between consecutive levels. Namely,

$$\boldsymbol{R}_\epsilon^{\mathcal{THB};\, l_1,l_2} = \begin{cases} \boldsymbol{I} & \text{if } l_2 = l_1, \\ trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{l_1,l_1+1}) \cdot \boldsymbol{R}_\epsilon^{\mathcal{THB};\, l_1+1,l_2} & \text{otherwise,} \end{cases} \tag{11}$$

$$\Big[trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{l,l+1})\Big]_{ij} = \begin{cases} \Big[\boldsymbol{R}_{\mathcal{B}_\epsilon}^{l,l+1}\Big]_{ij} & \text{if } b_j \in \mathcal{B}_{\mathcal{B}_\epsilon}^{l+1} \cap \mathcal{B}_-^{l+1}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\boldsymbol{R}_{\mathcal{B}_\epsilon}^{l,l+1} = \boldsymbol{R}^{l,l+1}_{\left(\mathcal{B}_{\mathcal{B}_\epsilon}^{l},\mathcal{B}_{\mathcal{B}_\epsilon}^{l+1}\right)}. \tag{12}$$

Equation (12) indicates that $\boldsymbol{R}_{\mathcal{B}_\epsilon}^{l,l+1}$ is composed of rows and columns of $\boldsymbol{R}^{l,l+1}$, that are associated to functions supporting ϵ (see Definition 3.2). For example, considering again ϵ_3 in Figure 2, $\boldsymbol{R}_{\mathcal{B}_\epsilon}^{1,2}$ and $\boldsymbol{R}_{\mathcal{B}_\epsilon}^{0,1}$ can be seen directly as sub-matrices of the operators shown in Section 3.3.:

$$\boldsymbol{R}_{\mathcal{B}_\epsilon}^{0,1} = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 \\ 3 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \qquad \boldsymbol{R}_{\mathcal{B}_\epsilon}^{1,2} = \frac{1}{4}\begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{0,1}) = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{1,2}) = \frac{1}{4}\begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The local refinement operators can be computed and the multi-level extraction operator can be assembled the usual way

$$\boldsymbol{R}_{\epsilon_3}^{\mathcal{THB};\,2,2} = \boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R}_{\epsilon_3}^{\mathcal{THB};\,1,2} = trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{1,2}) = \frac{1}{16}\begin{bmatrix} 12 & 4 & 0 \\ 4 & 12 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{M}_{2,\epsilon_3}^{loc} = \frac{1}{16}\begin{bmatrix} 3 & 1 & 0 \\ 9 & 3 & 0 \\ 4 & 12 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

$$\boldsymbol{R}_{\epsilon_3}^{\mathcal{THB};\,0,2} = trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{0,1}) \cdot trunc(\boldsymbol{R}_{\mathcal{B}_\epsilon}^{1,2}) = \frac{1}{16}\begin{bmatrix} 3 & 1 & 0 \\ 9 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Finally, the recursion in the definition 11 can be exploited to define an algorithm to compute directly $M_{l,\epsilon}^{loc}$

1. $M_{0,\epsilon}^{loc} = J^0$,
2. $M_{l+1,\epsilon}^{loc} = \begin{bmatrix} M_{l,\epsilon}^{loc}\, trunc(R_{\mathcal{B}_\epsilon}^{l,l+1}) \\ J^{l+1} \end{bmatrix}$, $l = 0, \dots, N-1$

where N is the index of the biggest nested space (as in Section 3.5) and J^l is a matrix that selects the element active functions of level l. Namely, it has size $\#\mathcal{N}_\epsilon^l \times \#\mathcal{B}_\epsilon^l$ and $J^l(i,j) = 1$, if the j-th function in $\mathcal{B}_\epsilon^l$ is the i-th function in $\mathcal{N}_\epsilon^l$, otherwise $J^l(i,j) = 0$. Considering ϵ_3, we obtain

$$\boldsymbol{J}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \boldsymbol{J}^1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \quad \boldsymbol{J}^2 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$M_{0,\epsilon}^{loc} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad M_{1,\epsilon}^{loc} = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 \\ 3 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix} \quad M_{2,\epsilon}^{loc} = \frac{1}{16}\begin{bmatrix} 3 & 1 & 0 \\ 9 & 3 & 0 \\ 4 & 12 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

The algorithm is the local version of the one presented in [23], showing that the same algorithm can be used to build local operators. Finally, removing the truncation in Equation (11) and in the algorithm allows to build the local operators for $\mathcal{HB}$ by means of local computations.

3.6.2 Structure of the local multi-level extraction operator

Thanks to the truncation, it is straightforward to notice that $\boldsymbol{M}_{L,\epsilon}^{loc}$ will have (up to re-ordering) a block structure

$$\boldsymbol{M}_{L,\epsilon}^{loc} = \left[\begin{array}{c|c} \boldsymbol{M}_{\epsilon,\pm}^{loc} & 0 \\ \hline 0 & \boldsymbol{I} \end{array}\right]$$

$$\boldsymbol{M}_{\epsilon,\pm}^{loc} = \boldsymbol{M}_{\left(\mathcal{B}_\epsilon^l \cap \mathcal{B}_+^l\,,\,\mathcal{B}_\epsilon^L \cap \mathcal{B}_-^L\right)}^{loc}$$

that can be exploited for efficiency when using the operator in a computer implementation. In particular, the columns of $\boldsymbol{M}_{\epsilon,\pm}^{loc}$ are associated to functions in $\mathcal{B}_\epsilon^L \cap \mathcal{B}_-^L$, while its rows to functions in $\mathcal{B}_\epsilon^l \cap \mathcal{B}_+^l$, according to Definition 2.1 and 3.2. Instead, $\boldsymbol{I}$ is associated to the active functions of level L, i.e., $\mathcal{B}_\epsilon^L \cap \mathcal{THB}^L$. Practically, $\boldsymbol{M}_{\epsilon,\pm}^{loc}$ is the only non-trivial matrix that must be considered in the multiplication when applying the

operator. Note that in general $\boldsymbol{M}^{loc}_{\epsilon,\pm}$ can be considerably both smaller size and denser than $\boldsymbol{M}^{loc}_{L,\epsilon}$, especially in higher-dimensions. However, this structure is unfortunately lost once $\boldsymbol{M}^{loc}_{\epsilon}$ is combined with the Bézier extraction operator.

3.7 Multi-level Bézier extraction

The multi-level extraction operator can be combined directly with the Bézier extraction operator [22], as depicted in Figure 4. This creates a direct map from a standard set of reference basis functions equal for each element, to the multi-level local basis. As a consequence, hierarchical refinements can be treated in a way that is very similar to standard finite element implementations.

However, it should be noted that the hierarchical refinement introduces a non-constant number of degrees of freedom per element. For example, consider ϵ_0 (3 DOFs) and ϵ_4 (5 DOFs) in Figure 2. Therefore, although the multi-level Bézier extraction can significantly ease the introduction of hierarchical refinement in standard finite element implementations, the existing code should still allow for element matrices of non-constant size. This requirement seems to be inherent to the method.

3.8 The multi-level extraction for (truncated) hierarchical NURBS

Once the multi-level extraction operator for (truncated) hierarchical B-Splines is defined, following [22] it is possible to extend it to (truncated) hierarchical NURBS. Let $\boldsymbol{n} = (n_0, \dots, n_m)^\top$ be a vector of B-Splines, possibly of different levels l, $0 \leq l \leq L$. Let $\boldsymbol{b}^L = (b_0, \dots, b_t)^\top$ be a set of fine B-Splines of level L, so that $\boldsymbol{n} = \boldsymbol{M}^{glob}_L \boldsymbol{b}^L$. Then, given the weights $\boldsymbol{w} = (w_0, \dots, w_m)^\top$, and the diagonal matrix of weights

$$\boldsymbol{W} = \begin{bmatrix} w_0 & & & \\ & w_1 & & \\ & & \ddots & \\ & & & w_m \end{bmatrix}$$

the associated NURBS $\boldsymbol{r} = (r_0, \dots, r_m)^\top$ can be written as

$$\boldsymbol{r} = \frac{\boldsymbol{W}\boldsymbol{n}}{\boldsymbol{n}^\top \boldsymbol{w}} = \frac{\boldsymbol{W}\boldsymbol{M}^{glob}_L \boldsymbol{b}^L}{\boldsymbol{n}^\top \boldsymbol{w}}.$$

Analogously, defining the level-L weights $\boldsymbol{w}^L = (\boldsymbol{M}^{glob}_L)^\top \boldsymbol{w}$ allows to express a spline $\boldsymbol{S} = \boldsymbol{P}^\top \cdot \boldsymbol{r}$ in terms of the weights $\boldsymbol{w}^L$, the control points $\boldsymbol{P}^L$, and the functions $\boldsymbol{b}^L$ of

level L as

$$\boldsymbol{S} = \boldsymbol{P}^\top \frac{\boldsymbol{W}\boldsymbol{M}_L^{glob}\boldsymbol{b}^L}{\boldsymbol{n}^\top \boldsymbol{w}} = (\boldsymbol{P}^L)^\top \frac{\boldsymbol{W}^L\boldsymbol{b}^L}{(\boldsymbol{b}^L)^\top \boldsymbol{w}^L}$$

$$\boldsymbol{W}^L = \begin{bmatrix} w_0^L & & & \\ & w_1^L & & \\ & & \ddots & \\ & & & w_t^L \end{bmatrix}$$

$$\boldsymbol{P}^L = (\boldsymbol{W}^L)^{-1}(\boldsymbol{M}_L^{glob})^\top \boldsymbol{W}\boldsymbol{P}.$$

The above properties can be extended in the same way to local operators and to the (local or global) multi-level Bézier extraction.

3.9 Extension to higher dimensional spaces

The discussion and definitions of the previous sections are valid also for multi-dimensional spaces. In particular, for each level l, the functions $b_j^l \in \mathcal{B}^l$ of the basis $\mathcal{B}^l$ can be sorted in a fixed order (typically the tensor product order) into a vector $\boldsymbol{b}^l$. Then, the refinement operator $\boldsymbol{R}^{l_1,l_2}$, such that $\boldsymbol{b}^{l_1} = \boldsymbol{R}^{l_1,l_2}\boldsymbol{b}^{l_2}$, $l_2 \geq l_1$, exists and can be used directly in the definitions of Section 3.5 to compute the multi-level extraction operator.

In case of tensor product spaces with tensor-product order, the refinement operator $\boldsymbol{R}^{l_1,l_2}$ can be computed as Kronecker product of the univariate refinement operators. Moreover, according to Section 3.5, just some rows of $\boldsymbol{R}^{l_1,l_2}$ are needed. Therefore, the Kronecker product can be limited to the necessary rows. This holds also for the element-localizations $\boldsymbol{R}_\epsilon^{l_1,l_2}$. It is worth noting that the hierarchical basis $\mathcal{N}$ is, in general, not of tensor product structure. Therefore, it is clear that there is no possibility of applying the extraction operator to every single direction and constructing $\mathcal{N}$ by taking the tensor product of unidimensional extracted functions. This is not even possible for the basis local to one element. Such an optimization was possible with the standard Bézier extraction [22], but it is not applicable directly in the multi-level setting. However, it could be still possible to extract each direction singularly, and then compute just the tensor product of univariate functions producing $\mathcal{N}$.

3.10 Extension to other sequences of nested spaces

In general, the multi-level extraction operator can be constructed for every sequence of nested spaces. Indeed let $V^0 \subset V^1 \subset \cdots \subset V^N$ be a sequence of nested spaces with basis $\mathcal{B}^0, \ldots, \mathcal{B}^N$, respectively, then, for each $\mathcal{B}^{l_1}, \mathcal{B}^{l_2}$, such that $l_1 \leq l_2$, there exists the refinement operator $\boldsymbol{R}^{l_1,l_2}$, as described in Section 3.2. Given a suitable selection of functions for analysis $\mathcal{N} \subset \bigcup_l \mathcal{B}^l$ (as described, e.g., in Section 2.3.1), the associated multi-level extraction operator can be defined following Section 3.5.

Here we assert the existence of the refinement operator $\boldsymbol{R}$, however, its computation depends on the particular refinement considered. In the previous sections the whole

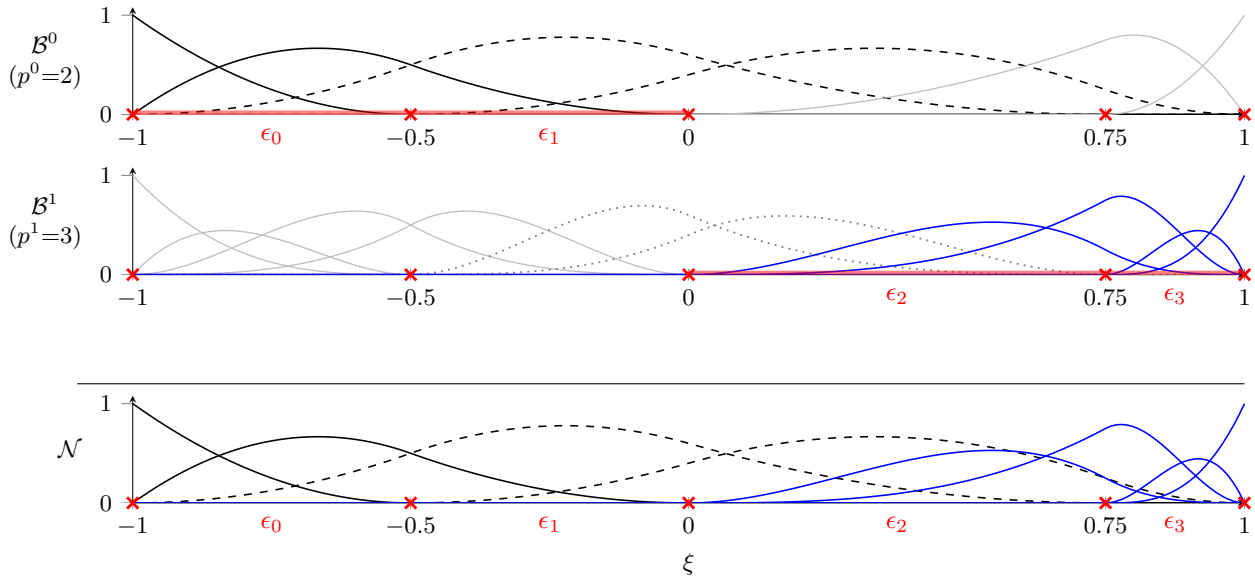


Figure 5: Example of hierarchical B-Spline functions. The knots of each level are marked by red crosses.

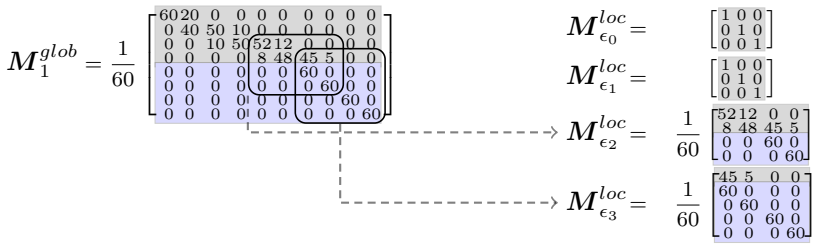
procedure was exemplified for knot insertion and the $\mathcal{HB}$ and $\mathcal{THB}$ bases. In this case, $\boldsymbol{R}$ can be computed by standard knot insertion techniques and Section 3.12 proposes a few algorithms to compute directly the *element-localized* operators. The next section shows a further example in the context of degree elevation.

3.10.1 The multi-level extraction for degree elevation

The refinement operator $\boldsymbol{R}^{l_1,l_2}$ for degree elevation can be constructed using standard degree elevation techniques (see, e.g., [31, 37]). For example, consider the B-Splines $\mathcal{B}^0 = \{b_0^0, \dots, b_5^0\}$ of degree $p^0 = 2$ and $\mathcal{B}^1 = \{b_0^1, \dots, b_9^1\}$ of degree $p^1 = 3$ defined by the knot vectors $\Xi^0 = [-1, -1, -1, -0.5, 0, 0.75, 1, 1, 1]$ and $\Xi^1 = [-1, -1, -1, -1, -0.5, -0.5, 0, 0, 0.75, 0.75, 1, 1, 1, 1]$, respectively. The overlay of B-Splines is shown in Figure 5, together with a possible choice for $\mathcal{N}$ (similar to Figure 4). For this example, the refinement operator $\boldsymbol{R}^{0,1}$ and multi-level extraction operator $\boldsymbol{M}_1^{glob}$ are:

$$\boldsymbol{R}^{0,1} = \frac{1}{60}\begin{bmatrix} 60 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 50 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 50 & 52 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 48 & 45 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 & 55 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 60 \end{bmatrix} \quad \boldsymbol{M}_1^{glob} = \frac{1}{60}\begin{bmatrix} 60 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 50 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 50 & 52 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 48 & 45 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 60 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60 \end{bmatrix}$$

Finally, the local operators read as follows.



Note that the operators of ϵ_0 and ϵ_1 have 3 columns, while the operators of ϵ_2 and ϵ_3 have 4 columns. This corresponds to the fact that the ϵ_0 and ϵ_1 belong to the 0-th level, where $\mathcal{B}^0$ has degree $p^0 = 2$, while ϵ_2 and ϵ_3 are of level 1, with degree $p^1 = 3$.

3.11 Advantages

The multi-level extraction operator offers the following advantages:

- When combined with Bézier extraction, it eases the implementation of hierarchical refinement in existing finite element codes, as it gives a total element point of view on the hierarchical refinement.
- More generally, the multi-level extraction operator eases the implementation of hierarchical refinement in existing "single-level" codes, as it flattens the multi-level hierarchy of functions into a sequence of elements equipped with a standard single-level basis. For example, it could ease the implementation of hierarchical refinement in existing IGA codes.
- It generalizes the Bézier extraction to multi-level refinements by presenting the same properties (9), (10) and usage of the standard Bézier extraction
- Once computed, it removes the need to navigate through the hierarchy of overlay functions, in contrast to, e.g., [20, 38, 39], but only a constant number of standard single-level shapes are evaluated once per element, e.g., Bernstein polynomials.
- Once computed, it removes concatenation of mappings in case of overlay of local reference spaces (e.g., [38, 39]):
  - just one direct mapping from element to physical space can be employed,
  - no coordinate conversion between levels is needed,
  - it is important for evaluating derivatives, especially for high-order PDEs.
- In comparison to global methods,
  - It substitutes large inefficient sparse multiplications by small dense multiplications, more efficient and suitable to parallelism.
  - It eases local evaluation of the solution field and its derivatives e.g., for material non-linearities, deformation gradient, etc. Indeed, a global method that maps all active functions (or degrees of freedom) simultaneously can be very inefficient to evaluate the solution locally at a point. Alternatively, it would be possible to evaluate the solution at all the integration point simultaneously, but this would still involve large inefficient sparse multiplications and increase the memory requirements.

  - It allows domain distribution in parallel codes, since it encodes local information about each element individually.
- This approach is valid for every overlay of nested spaces, namely it is applicable to different kinds of refinements and basis functions, e.g., (anisotropic) spline knot insertion, (anisotropic) spline degree elevation, (anisotropic) C^0-continuous linear polynomial and h-FEM refinement, or (anisotropic) C^0-continuous high-order polynomials and hp-FEM refinement (see, e.g., [35]). In particular, any polynomial basis can be used, i.e. Bernstein, Legendre or Lagrange (combining the multi-level extraction operator with the Legendre extraction [40]).

3.12 Knot insertion algorithms

In this section we present a few general algorithms inspired by [31, 41] to produce the knot insertion operators. The algorithms are expressed in the MATLAB® syntax. Given $\Xi^l = (\xi_0^l, \ldots, \xi_{n_l-p-1}^l)$, $l \in \{1,2\}$, $\Xi^1 \subset \Xi^2$, the local knot insertion operators $\boldsymbol{R}_\epsilon^{1,2}$ can be computed column-wise by the Oslo algorithm [36, 41], or row-wise by the Boehm's method [36, 41]. Algorithm 1 shows a version of the Oslo algorithm using vector operations, while Algorithm 2 presents its scalar version. The latter is suitable to traditional procedural programming languages. Finally, $\boldsymbol{R}_\epsilon^{1,2}$ can be produced by Algorithm 3. Analogously, the Boehm's method can be repeatedly used to produce the operators $\boldsymbol{R}_\epsilon^{1,2}$, as shown for example in Algorithm 5. Here, the main loop runs once through the knots Ξ^1 and the knots to be inserted. The associated multiplicities are cached to handle the overlapping columns between operators.

The Oslo algorithm computes the local coefficients of the knot insertion operator by means of a direct formula based on the coarse and fine local knots. Such a direct formula features no data dependencies between the computation of different columns of $\boldsymbol{R}_\epsilon^{1,2}$ and, therefore, Algorithm 3 is suitable for parallelization. A possible parallel version is presented in Algorithm 4. Instead, Algorithm 5 inserts a knot at a time from left to right and overwrites intermediate coefficients. This iterative procedure features inherent data dependencies between computations of coefficients of the same row, but different rows could be computed in parallel. However, this is not within the scope of this work. It should also be noted that Algorithm 5 is numerically more stable, as just convex combinations are used, while Algorithms 1 and 2 can perform also non-convex combinations. However, this becomes noticeable just for higher degrees.

The core operation of the presented algorithms are linear combinations of coefficients and their count is listed in Table 1. Table 1 compares also the Algorithm *bs2bs* presented in [42]. Algorithm 3 results to be more efficient than Algorithm 5 if $2 \cdot n_1 > n_2$, i.e., if the number of knots is at least doubled. Therefore, we prefer Algorithm 3, as this condition will be typically met, e.g., in case of refinement by bisection. Algorithm *bs2bs* is better than Algorithm 3 if $2 \cdot (1 + p/(p+1)) \cdot e < n_2$, i.e., when the number e of non-empty spans in the fine knot vector Ξ^2 is quite smaller than n_2. This can be the case when the multiplicity in Ξ^2 is high. Figure 6 compares the number of linear operations used by the considered algorithms while bisecting $\Xi = (-1,-1,-1,-1,-0.8,0.3,1,1,1,1)$ 11 times, for $p = 3$. The parallel Oslo algorithm (Algorithm 4) is executed on 4 processors.

	Boehm	Oslo	bs2bs
Operation Count	$p \cdot (p+1) \cdot (n_2 - n_1)$	$\frac{1}{2} p \cdot (p+1) \cdot n_2$	$p \cdot (2p+1) \cdot e$

Table 1: Number of linear combinations needed to produce the element local knot insertion operators. e = number of non-empty knot spans in the fine knot vector Ξ^2.

Note that Figure 6b shows that the parallel Oslo algorithm distributes the number of combinations uniformly among the processors as the number of inserted knots increases.

The multi-level extraction operator M_ϵ can be computed by composing rows of the operators $\boldsymbol{R}_\epsilon^{l_1,l_2}$, as described in Sections 3.4 and 3.4.

Algorithm 1 Oslo1 Algorithm (vector version)

1: **procedure** OSLO1(p, coarsekn, finekn, m)

Input:

p: spline degree
coarsekn: coarse knot vector
finekn = $\{\xi_1, \ldots, \xi_m\}$: fine knot vector
rf: arbitrary fine knot index, such that $1 \leq$ rf $\leq m - p - 1$
cf: coarse knot index such that *coarsekn*(cf) $\leq$ *finekn*(rf) $<$ *coarsekn*(cf+1)

Output:

b: vector knot insertion coefficients

2: ***b*** = 1;
3: **for** k=1, . . . , p **do**
4: ***t1*** = coarsekn(cf+1-k:cf);
5: ***t2*** = coarsekn(cf+1:cf+k);
6: x = finekn(rf+k);
7: ***w*** = (x-***t1***)./(***t2***-***t1***);
8: ***b*** = [(1-***w***) .* ***b***; 0] + [0; ***w*** .* ***b***];
9: **end for**
10: **end procedure**

Algorithm 2 Oslo1 Algorithm (scalar version)

1: **procedure** OSLO1(p, coarsekn, finekn, m)

Input:

p: spline degree

coarsekn: coarse knot vector

finekn = $\{\xi_1, \ldots, \xi_m\}$: fine knot vector

rf: arbitrary fine knot index such that $1 \leq \text{rf} \leq m - p - 1$

cf: coarse knot index such that coarsekn(cf) $\leq$ finekn(rf) $<$ coarsekn(cf+1)

Output:

b: vector knot insertion coefficients

```
2:     b(p+1)=1;
3:     for k=1, ..., p do
4:         x = finekn(rf+k);
5:         w2 = (coarsekn(cf+1)-x) / (coarsekn(cf+1) - coarsekn(cf+1-k));
6:         b(p-k+1) = w2 * b(p-k+2);
7:         for  a = 2,..., k  do
8:             w1 = w2;
9:             w2 = (coarsekn(cf+a)-x) / (coarsekn(cf+a) - coarsekn(cf-k+a) );
10:            b(p-k+a) = (1-w1) * b(p-k+a) + w2 * b(p-k+a+1);
11:        end for
12:        b(p+1) = (1-w2) * b(p+1);
13:    end for
14: end procedure
```

Algorithm 3 Element Knot Insertion Operators (using Oslo algorithm)

Input:

p: spline degree
coarsekn: coarse knot vector
finekn $= \{\xi_1, \ldots, \xi_m\}$: fine knot vector
m: length of fine knot vector

Output:

e: number of fine elements
R_{ϵ_a}, $a = 1, \ldots, e$: local element operators

```
1: cf=p+1;
2: rf=1;
3: e=1;
4: while rf ≤ m-p-1 do
5:     mult=1;
6:     while finekn(rf+mult) == finekn(rf) do
7:         mult=mult+1;
8:     end while
9:     lastcf = cf;
10:    while coarsekn(cf+1) ≤ finekn(rf) do
11:        cf = cf+1;
12:    end while
13:    if e>1 then
14:        offs = cf-lastcf;
15:        R^e( 1:p+1-offs, 1:p+1-mult ) = R^{e-1}( 1+offs:p+1, 1+mult:p+1 );
16:    end if
17:    for t= p+2-mult:p+1 do
18:        R^e(:, t) = oslo1(p, coarsekn, finekn, cf, rf);
19:        rf=rf+1;
20:    end for
21:    e=e+1;
22: end while
```

Algorithm 4 Element Knot Insertion Operators (parallel, using Oslo algorithm)

Input:

p: spline degree
coarsekn: coarse knot vector
glfinekn $= \{\xi_1, \ldots, \xi_m\}$: fine knot vector
m: length of fine knot vector
ithread: thread index (starting from 1)
nthreads: total number of threads

Output:

e: number of fine elements processed by the thread
R_{ϵ_a}, $a = 1, \ldots, e$: local element operators processed by the thread

```
1: chunk = floor((m-2*(p+1)+1)/nthreads);
2: rem = mod(m-2*(p+1)+1,nthreads);
3: knstart = max( (ithread-1)*chunk, 0 ) + min(ithread-1, rem)+1;
4: knend = 2*p+ ithread*chunk + min(ithread, rem)+1;
5: while knend-knstart+1 ≥ 2*(p+1) and glfinekn(knstart+p+1) == glfinekn(kn-
   start+p) do
6:     knstart = knstart+1;
7: end while
8: while knend-knstart+1 ≥ 2*(p+1) and glfinekn(knend-p-1) == glfinekn(knend-p)
   do
9:     knend = knend-1;
10: end while
11: if knend-knstart+1 < 2*(p+1) then return;
12: finekn = glfinekn( knstart:knend );
13: cf=p+1; rf=1; e=1;
14: while rf ≤length(finekn)-p-1 do
15:     mult=1;
16:     while finekn(rf+mult) == finekn(rf) do mult=mult+1;
17:     lastcf = cf;
18:     while coarsekn(cf+1) ≤ finekn(rf) do cf = cf+1;
19:     if e>1 then
20:         offs = cf-lastcf;
21:         R_e( 1:p+1-offs, 1:p+1-mult ) = R_{e-1}( 1+offs:p+1, 1+mult:p+1 );
22:     else
23:         while coarsekn(cf+1) <= finekn(p+1) do cf = cf+1;
24:         for t= 1:p+1-mult do
25:             R_e(1:p+1, t) = oslo1(p, coarsekn, finekn, cf, rf,ithread);
26:             rf=rf+1;
27:         end for
28:     end if
29:     for t= p+2-mult:p+1 do
30:         R_e(:, t) = oslo1(p, coarsekn, finekn, cf, rf);
31:         rf=rf+1;
32:     end for
33:     e=e+1;
34: end while
```

Algorithm 5 Element Knot Insertion Operators (using Boehm's method)

Input:

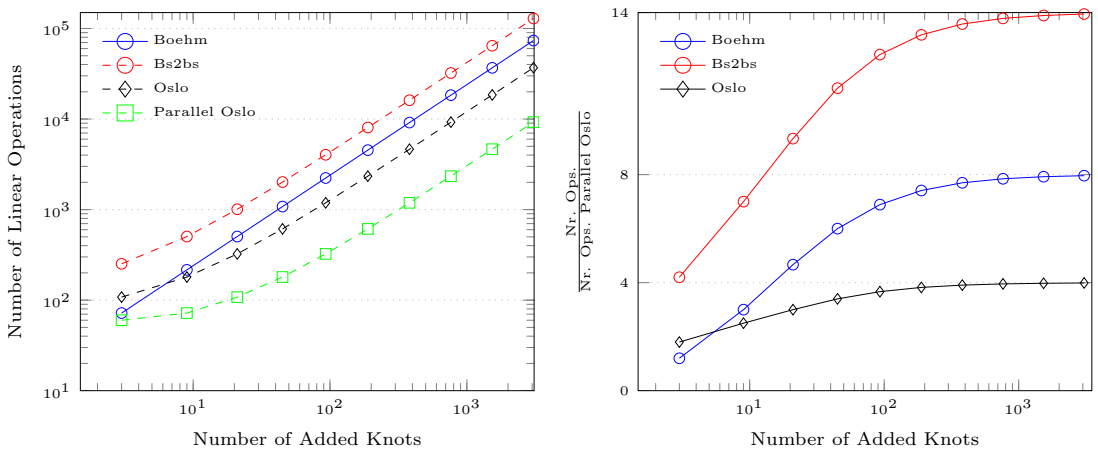
p: spline degree
old_kn = $\{\hat{\xi}_1, \ldots, \hat{\xi}_r\}$: : coarse knot vector
r: length of old_kn
ins_kn = $\{\xi_1, \ldots, \xi_s\}$: knots to insert
s: length of ins_kn

Output:

e: number of fine elements
R_{ϵ_a}, $a = 1, \ldots, e$: local element operators

```
1: i = p+1; k = p+1; j = 0; e=1; new_kn(1:p+1) = old_kn(1:p+1);
2: R_1 = I;
3: while i<r or j<s do
4:     nmv(e) = 0;
5:     om=1;
6:     while i+1+om≤r and old_kn(i+1) == old_kn(i+1+om) do
7:         om=om+1;
8:     end while
9:     while j<s and ins_kn(j+1) ≤ old_kn(i+1) do
10:        nm=0;
11:        while j+1+nm≤s and ins_kn(j+1) == ins_kn(j+1+nm) do
12:            nm=nm+1;
13:        end while
14:        nmv(e) = nm;
15:        if ins_kn(j+1) == old_kn(i+1) then omv(e) = om;
16:        else omv(e) = 0;
17:        end if
18:        for r = 0, ..., nm-1 do
19:            R_{e+1} (p+1-omv(e),p+1-nm-omv(e)+r+1)=1;
20:            for l=0, ..., -r+1 do
21:                alfa = (old_kn(i+p+l) - ins_kn(j+1))/(old_kn(i+p+l) - new_kn(k+l));
22:                ind2=p+1+l+r-nm-omv(e);
23:                R_{e+1}(:,ind2,e+1)= alfa*R_{e+1}(:,ind2-1) + (1-alfa)*R_{e+1}(:,ind2);
24:            end for
25:            for l=-r,...,-p+1 do
26:                alfa = (old_kn(i+p+l) - ins_kn(j+1))/(old_kn(i+p+l) - new_kn(k+l));
27:                ind2=p+1+l+r;
28:                R_e(:,ind2)= alfa*R_e(:,ind2-1) + (1-alfa)*R_e(:,ind2);
29:            end for
30:            R_{e+1}(1:p+1-omv(e),1:p+1-nm-omv(e))= R_e(1+omv(e):p+1,omv(e)+nm+1:p+1);
31:            new_kn(k+1) = ins_kn(j+1);
32:            k=k+1; j=j+1;
33:        end for
34:        if omv(e) == 0 then
35:            e = e+1;
36:            nmv(e) = 0;
37:        end if
38:    end while
```

```
    if i < r then
        for t=0:om-1 do
            new_kn(k+1) = old_kn(i+1);
            R_{e+1}(p+1-t,p+1-t)= 1;
            k = k + 1; i = i + 1;
        end for
        omv(e)=om;
        R_{e+1}(1:p+1-omv(e),1:p+1-nmv(e)-omv(e))= R_e(1+omv(e):p+1,omv(e)+nmv(e)+1:p+1);
        e = e+1;
    end if
end while
for f = e-1,...,2 do
    R_{f-1}(1+omv(f-1):p+1,2+nmv(f-1)+omv(f-1):p+1)=R_f(1:p+1-
omv(f-1),2:p+1-nmv(f-1)-omv(f-1));
end for
```



(a) Number of linear operations

(b) Number of linear operations divided by number of operations required by parallel Oslo

Figure 6: Comparison of knot insertion operator algorithms on a test consisting of 11 bisections of $\Xi = (-1, -1, -1, -1, -0.8, 0.3, 1, 1, 1, 1)$, for $p = 3$. The parallel Oslo algorithm (Algorithm 4) is executed on 4 processors.

4 Numerical Examples

In this section we present two illustrative numerical examples common in the literature, (see, e.g., [43, 32, 24]). Similarly to [24], the adaptive refinements will be driven by the contribution to the error in the energy norm η_ϵ local to one element ϵ. The formula for η_ϵ will be given in the following for each specific example. In particular, Algorithm 6 has been used to control refinement.

Algorithm 6 Multi-Level Error-Controlled Refinement

procedure MULTI-LEVEL ERROR-CONTROLLED REFINEMENT(α, max_iterations)
  Define an initial mesh of elements of level $l = 0$
  $k = 0$
  for $k <$ max_iterations **do**
    Solve problem on current mesh
    Compute η_ϵ for each element ϵ
    Mark $\alpha\%$ of the elements with highest error for refinement
    for each marked element $\tilde{\epsilon}$ **do**
      Deactivate element $\tilde{\epsilon}$
      Activate all the sub-domains of level $l + 1$ contained in $\tilde{\epsilon}$, where l is the level of $\tilde{\epsilon}$
    end for
    $k = k + 1$
  end for
end procedure

In the following, we consider overlay of B-Spline and NURBS functions, together with refinement by bi-section.

4.1 Heat Conduction on L-Shaped domain

As a first example, let us consider the Laplace equation defined over an L-shaped domain Ω. We seek a function $u : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} \Delta u &= 0 && \text{on } \Omega \\ u &= 0 && \text{on } \Gamma_D \\ \nabla u \cdot \boldsymbol{n} &= g && \text{on } \Gamma_N \end{aligned}$$

where $\Gamma_D \cap \Gamma_N = \emptyset$, $\overline{\Gamma_D \cup \Gamma_N} = \partial\Omega$, $\partial\Omega$ is the boundary of Ω, and $\boldsymbol{n}$ is the unit outward normal vector on $\partial\Omega$. The domain Ω and its boundaries Γ_N, Γ_D are depicted in Figure 7. We manufacture a solution for this problem considering as analytical solution the function $\bar{u} = r^{\frac{2}{3}} \sin(\frac{2}{3}\theta)$ satisfying the Dirichlet boundary conditions, with g computed from $\bar{u}$ as $g = \nabla\bar{u} \cdot \boldsymbol{n}$.

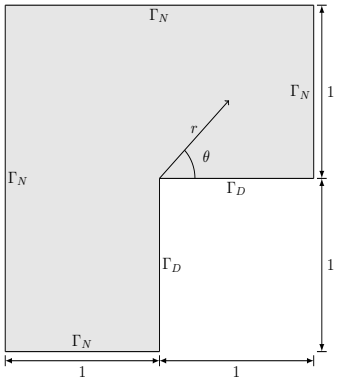


Figure 7: L-Shaped domain.

For an element ϵ, η_ϵ is defined as

$$\eta_\epsilon^2 = \int_\epsilon \nabla(u - \bar{u}) \cdot \nabla(u - \bar{u}) \mathrm{d}x.$$

An example of three refinement steps is shown in Figure 8, while the convergence for spline degree $p \in \{2, 3\}$ using Algorithm 6 with $\alpha = 0.2$ is shown in Figure 9. Here, the refinement is directed towards the singular re-entrant corner, improving the convergence of the uniform mesh refinement. In particular, the optimal convergence rate $p/2$ [44] is obtained, showing the capabilities of the adaptive method in capturing the singularity. The same result was obtained in, e.g., [32, 24]

4.2 Linear Elastic Circular Plate with a Hole

As a second example, we consider an infinite plate with a circular hole under constant in-plane tension. The infinite plate is modeled by a finite quarter of annulus Ω, as described in Figure 10. R_i is the radius of the hole, while R_o is the outer radius of the annulus. We solve the classical equations of linear elasticity with zero right hand side

$$\begin{aligned} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) &= \mathbf{0} && \text{on } \Omega \\ \boldsymbol{u} &= \mathbf{0} && \text{on } \Gamma_D \\ \nabla \boldsymbol{u} \cdot \boldsymbol{n} &= \boldsymbol{g} && \text{on } \Gamma_N \end{aligned}$$

where $\boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{C} : \boldsymbol{\epsilon}(\boldsymbol{u})$, $\boldsymbol{\epsilon}(\boldsymbol{u}) = (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)/2$, $\boldsymbol{C}_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})$. The constants λ and μ are the Lamè parameters and δ_{ij} is the standard Kronecker delta. For

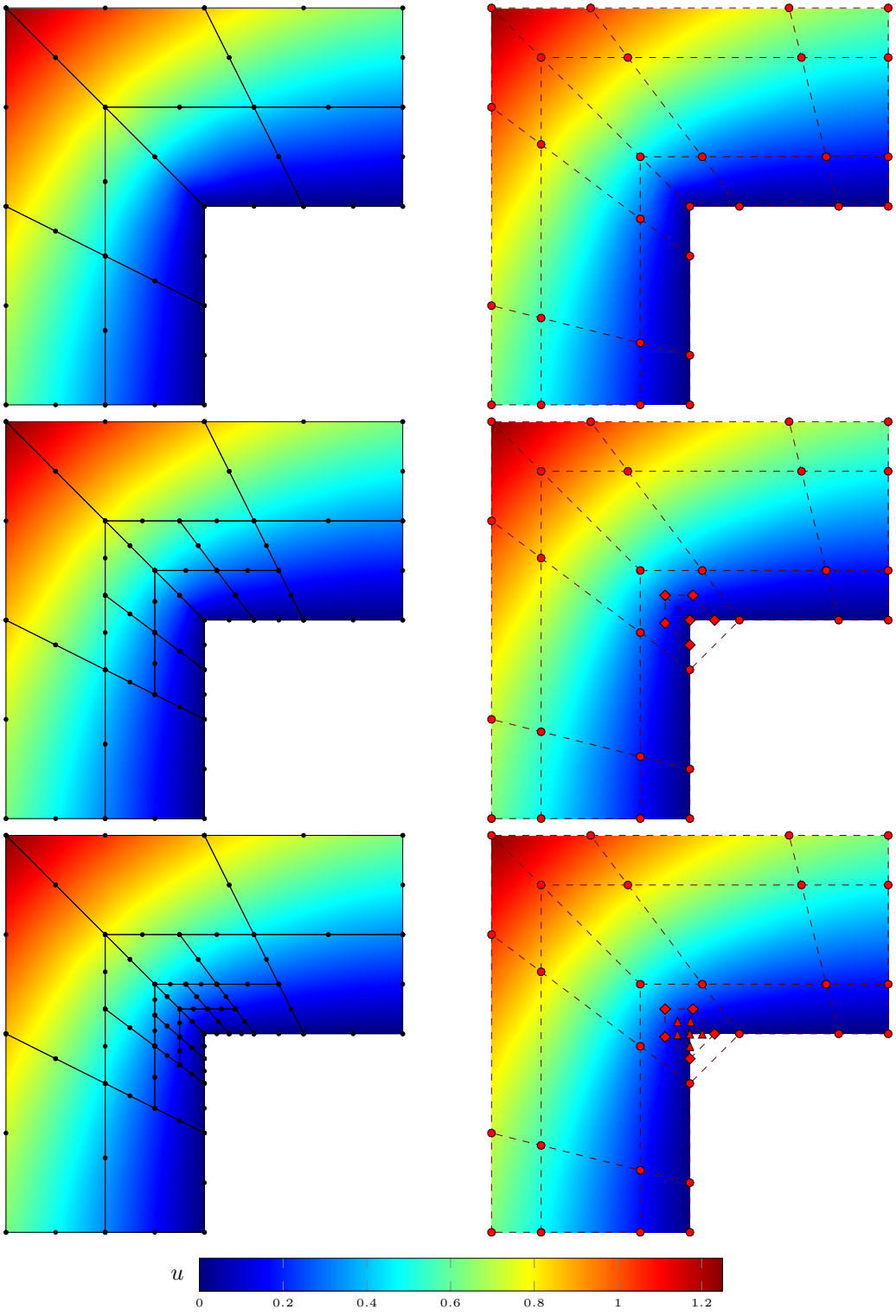
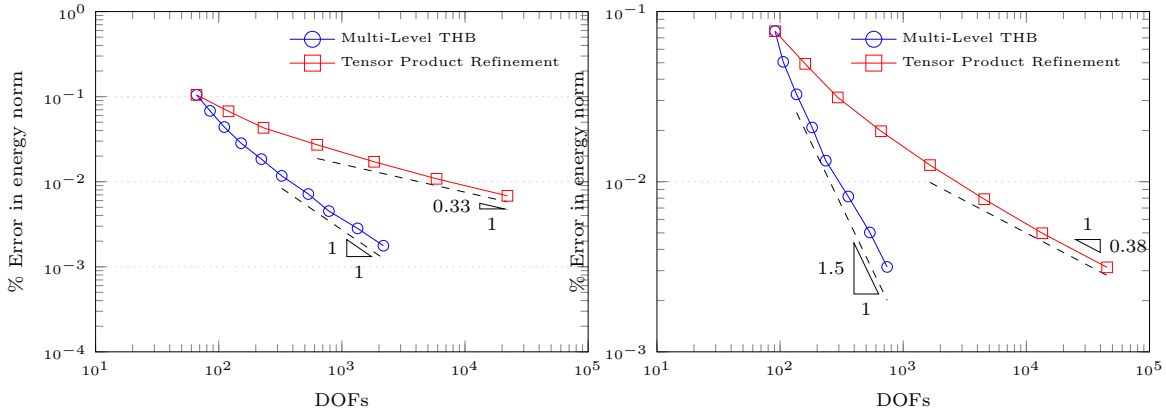


Figure 8: Example of three adaptive steps on the L-shaped domain problem. The color represents temperature. The left column shows the Bézier control mesh, i.e., the elements and the control points with respect of the Bernstein basis. The right column shows the control mesh. These two meshes are related by Equation (10). The control mesh is composed of control points associated to functions in $\mathcal{N}$. The control points relative to functions in $\mathcal{N}^l$ are represented by circles, for $l = 0$, diamonds for $l = 1$, triangles for $l = 2$.



(a) $p = 2$ (b) $p = 3$

Figure 9: Convergence rates of the multi-level refinement compared to standard tensor product refinement.

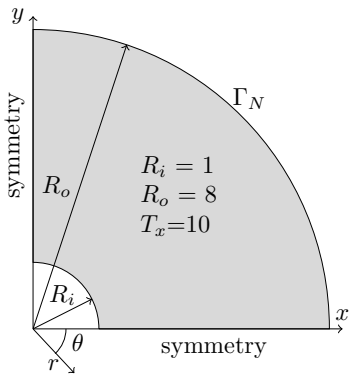


Figure 10: Elastic plate with circular hole.

this problem, the analytical stress [45] reads, in cylindrical coordinates,

$$\sigma_{rr}(r,\theta) = \frac{T_x}{2}\left[1 - \frac{R_i^2}{r^2} + \left(1 - 4\frac{R_i^2}{r^2} + 3\frac{R_i^4}{r^4}\right)\cos(2\theta)\right],$$
$$\sigma_{\theta\theta}(r,\theta) = \frac{T_x}{2}\left[1 + \frac{R_i^2}{r^2} - \left(1 + 3\frac{R_i^4}{r^4}\right)\cos(2\theta)\right],$$
$$\sigma_{r\theta}(r,\theta) = -\frac{T_x}{2}\left[1 + 2\frac{R_i^2}{r^2} - 3\frac{R_i^4}{r^4}\right]\sin(2\theta),$$

where T_x is the magnitude of the applied stress, imposed as Neumann boundary condition with the exact traction at the outer boundary Γ_N of the circular plate, see Figure 10. For an element ϵ, the measure for the true error η_ϵ is defined as

$$\eta_\epsilon^2 = \int_\epsilon \boldsymbol{\epsilon}(\boldsymbol{u} - \bar{\boldsymbol{u}}) : \boldsymbol{C} : \boldsymbol{\epsilon}(\boldsymbol{u} - \bar{\boldsymbol{u}})\mathrm{d}x,$$

where $\bar{u}$ is the analytical solution. An example of three refinement steps is shown in Figure 11, while the convergence for spline degree $p \in \{2, 3\}$ using Algorithm 6 with $\alpha = 0.4$ is depicted in Figure 12. Here, the solution is smooth but with a stress concentration close to the lower left corner. The local refinement is initially directed towards this area, yielding a superior pre-asymptotic convergence behaviour with respect to the uniform refinement. After this phase, the absence of singularity gradually drives the refinement to the whole domain. The error in the energy norm results to be slightly shifted down, but with the same convergence rate as the uniform refinement. The same results were obtained in, e.g., [43, 32, 24]

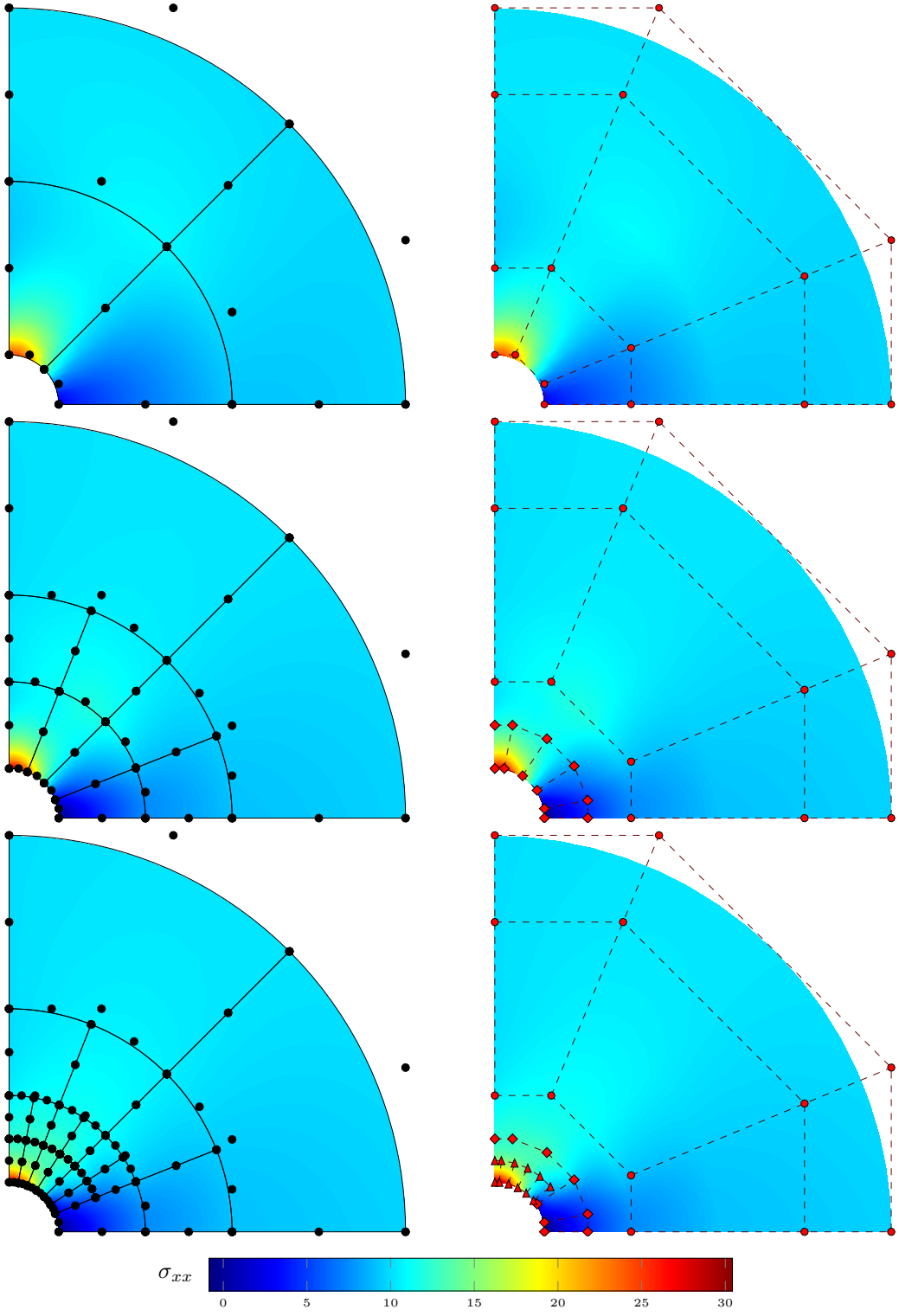
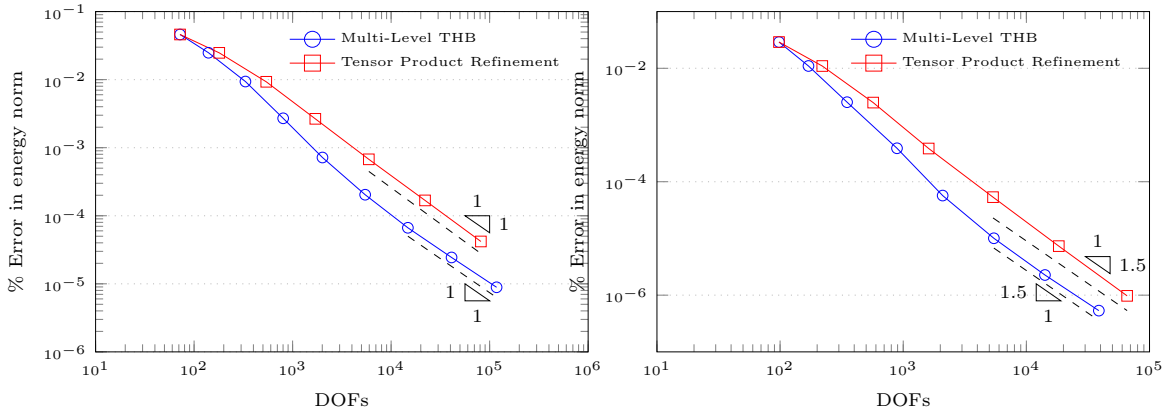


Figure 11: Example of three adaptive steps on the plate with circular hole. The color represent σ_{xx}. The left column shows the Bézier control mesh, i.e., the elements and the control points with respect of the Bernstein basis. The right column shows the control mesh. These two meshes are related by Equation (10). The control mesh is composed of control points associated to functions in $\mathcal{N}$. The control points relative to functions in $\mathcal{N}^l$ are represented by circles, for $l = 0$, diamonds for $l = 1$, triangles for $l = 2$.



(a) $p = 2$ (b) $p = 3$

Figure 12: Convergence rates of the multi-level refinement compared to standard tensor product refinement.

5 Conclusions

We have introduced the multi-level extraction as a way to implement hierarchically refined Isogeometric Analysis on existing finite element codes. Our main focus is on (Truncated) Hierarchical B-Splines and Isogeometric Analysis, but we have presented a general concept applicable to every sequence of nested spaces. The proposed local approach is suitable for big-scale, parallel and non-linear simulations, and in Section 3.11 we have reported a comprehensive list of its advantages and examples of applicable sequences of nested spaces. Moreover, the multi-level extraction incorporates the essential features of similar approaches present in the literature [26, 28, 25, 29, 24, 23, 30] and generalizes them in a unique framework. Furthermore, some basic algorithms to compute the multi-level extraction operator for knot insertion on spline spaces have been outlined and compared.

In conclusion, the proposed multi-level extraction appears to be a viable and practical tool for transforming an existing finite element software into an adaptive isogeometric code able to efficiently tackle problems where local refinement and adaptivity are necessary.

Acknowledgements

The authors gratefully acknowledge the support of the TUM Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement number 291763, as well as the support of the German Research Foundation (DFG) under Grant RA 624/22 and RA 624/27 of the priority programme SPP 1748. Moreover, A. Reali gratefully acknowledges the support of Fondazione Cariplo - Regione Lombardia through the project "Verso nuovi strumenti

di simulazione super veloci ed accurati basati sull'analisi isogeometrica", within the program RST - rafforzamento. Finally, the authors would like to thank Prof. T.J.R. Hughes (University of Texas at Austin) for the fruitful discussion on the subject of this paper.

References

[1] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs, *Isogeometric Analysis*. Chichester, UK: John Wiley & Sons, Ltd, Aug. 2009.

[2] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, pp. 4135–4195, Oct. 2005.

[3] M. J. Borden, C. V. Verhoosel, M. A. Scott, T. J. R. Hughes, and C. M. Landis, "A phase-field description of dynamic brittle fracture," *Computer Methods in Applied Mechanics and Engineering*, vol. 217, pp. 77–95, Apr. 2012. 00360.

[4] J. A. Cottrell, A. Reali, Y. Bazilevs, and T. J. R. Hughes, "Isogeometric analysis of structural vibrations," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, pp. 5257–5296, Aug. 2006. 00738.

[5] T. Elguedj, Y. Bazilevs, V. M. Calo, and T. J. R. Hughes, "$\bar B$ and $\bar F$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, pp. 2732–2762, June 2008. 00032.

[6] S. Morganti, F. Auricchio, D. J. Benson, F. I. Gambarin, S. Hartmann, T. J. R. Hughes, and A. Reali, "Patient-specific isogeometric structural analysis of aortic valve closure," *Computer Methods in Applied Mechanics and Engineering*, vol. 284, pp. 508–520, Feb. 2015. 00046.

[7] I. Akkerman, Y. Bazilevs, V. M. Calo, T. J. R. Hughes, and S. Hulshoff, "The role of continuity in residual-based variational multiscale modeling of turbulence," *Computational Mechanics*, vol. 41, pp. 371–378, Feb. 2008. 00211.

[8] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, and G. Scovazzi, "Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, pp. 173–201, Dec. 2007. 00662.

[9] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger, "3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades," *International Journal for Numerical Methods in Fluids*, vol. 65, pp. 236–253, Jan. 2011. 00277.

[10] M.-C. Hsu, D. Kamensky, F. Xu, J. Kiendl, C. Wang, M. C. H. Wu, J. Mineroff, A. Reali, Y. Bazilevs, and M. S. Sacks, "Dynamic and fluid–structure interaction simulations of bioprosthetic heart valves using parametric design with T-splines and Fung-type material models," *Computational Mechanics*, vol. 55, pp. 1211–1225, June 2015. 00053.

[11] H. Gómez, V. M. Calo, Y. Bazilevs, and T. J. R. Hughes, "Isogeometric analysis of the Cahn–Hilliard phase-field model," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, pp. 4333–4352, Sept. 2008. 00368.

[12] J. Kiendl, M.-C. Hsu, M. C. H. Wu, and A. Reali, "Isogeometric Kirchhoff–Love shell formulations for general hyperelastic materials," *Computer Methods in Applied Mechanics and Engineering*, vol. 291, pp. 280–303, July 2015. 00043.

[13] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes, "Isogeometric finite element data structures based on Bézier extraction of T-splines," *International Journal for Numerical Methods in Engineering*, vol. 88, pp. 126–156, Oct. 2011. 00193.

[14] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes, "Local refinement of analysis-suitable T-splines," *Computer Methods in Applied Mechanics and Engineering*, vol. 213, pp. 206–222, Mar. 2012. 00244.

[15] T. Dokken, T. Lyche, and K. F. Pettersen, "Polynomial splines over locally refined box-partitions," *Computer Aided Geometric Design*, vol. 30, pp. 331–356, Mar. 2013. 00174.

[16] D. R. Forsey and R. H. Bartels, "Hierarchical B-spline Refinement," in *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, (New York, NY, USA), pp. 205–212, ACM, 1988.

[17] G. Greiner and K. Hormann, "Interpolating and Approximating Scattered 3D-data with Hierarchical Tensor Product B-Splines," in *In Surface Fitting and Multiresolution Methods*, pp. 163–172, Vanderbilt University Press, 1997.

[18] R. Kraft, "Adaptive and linearly independent multilevel B-splines," in *Surface Fitting and Multiresolution Methods*, Vanderbilt University Press, 1997.

[19] C. Giannelli, B. Jüttler, and H. Speleers, "THB-splines: The truncated basis for hierarchical splines," *Computer Aided Geometric Design*, vol. 29, pp. 485–498, Oct. 2012.

[20] C. Giannelli, B. Jüttler, S. K. Kleiss, A. Mantzaflaris, B. Simeon, and J. Špeh, "THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 299, pp. 337–365, Feb. 2016.

[21] G. Kiss, C. Giannelli, and B. Jüttler, "Algorithms and Data Structures for Truncated Hierarchical B–splines," in *Mathematical Methods for Curves and Surfaces*, pp. 304–323, Springer, Berlin, Heidelberg, June 2012.

[22] M. J. Borden, M. A. Scott, J. A. Evans, and T. J. R. Hughes, "Isogeometric finite element data structures based on Bézier extraction of NURBS," *International Journal for Numerical Methods in Engineering*, vol. 87, pp. 15–47, July 2011. 00159.

[23] E. Garau and R. Vázquez, "Algorithms for the implementation of adaptive isogeometric methods using hierarchical splines," Techincal Report 16-08, IMATI-CNR, Pavia, June 2016.

[24] P. Hennig, S. Müller, and M. Kästner, "Bézier extraction and adaptive refinement of truncated hierarchical NURBS," *Computer Methods in Applied Mechanics and Engineering*, vol. 305, pp. 316–339, June 2016.

[25] A.-V. Vuong, "Finite Element Concepts and Bezier Extraction in Hierarchical Isogeometric Analysis," in *Progress in Industrial Mathematics at ECMI 2012*, pp. 385–390, Springer, Cham, 2014.

[26] P. B. Bornemann and F. Cirak, "A subdivision-based implementation of the hierarchical b-spline finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 253, pp. 584–598, Jan. 2013.

[27] C. Apprich, K. Höllig, J. Hörner, A. Keller, and E. N. Yazdani, "Finite Element Approximation with Hierarchical B-Splines," in *Curves and Surfaces*, pp. 1–15, Springer, Cham, June 2014.

[28] M. A. Scott, D. C. Thomas, and E. J. Evans, "Isogeometric spline forests," *Computer Methods in Applied Mechanics and Engineering*, vol. 269, pp. 222–264, Feb. 2014.

[29] E. J. Evans, M. A. Scott, X. Li, and D. C. Thomas, "Hierarchical T-splines: Analysis-suitability, Bézier extraction, and application as an adaptive basis for isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 284, pp. 1–20, Feb. 2015.

[30] G. Lorenzo, M. A. Scott, K. Tew, T. J. R. Hughes, and H. Gomez, "Hierarchically refined and coarsened splines for moving interface problems, with particular application to phase-field models of prostate tumor growth," *Computer Methods in Applied Mechanics and Engineering*, vol. 319, pp. 515–548, June 2017. 00001.

[31] L. Piegl and W. Tiller, *The NURBS Book*. Monographs in Visual Communications, Berlin, Heidelberg: Springer Berlin Heidelberg, 1995.

[32] A. V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon, "A hierarchical approach to adaptive local refinement in isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, pp. 3554–3567, Dec. 2011.

[33] C. Giannelli, B. Jüttler, and H. Speleers, "Strongly stable bases for adaptively refined multilevel spline spaces," *Advances in Computational Mathematics*, vol. 40, pp. 459–490, Sept. 2013.

[34] A. Buffa and C. Giannelli, "Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence," *Mathematical Models and Methods in Applied Sciences*, vol. 26, no. 01, pp. 1–25, 2016.

[35] P. Di Stolfo, A. Schröder, N. Zander, and S. Kollmannsberger, "An easy treatment of hanging nodes in -finite elements," *Finite Elements in Analysis and Design*, vol. 121, pp. 101–117, Nov. 2016.

[36] W. Boehm, "On the efficiency of knot insertion algorithms," *Computer Aided Geometric Design*, vol. 2, pp. 141–143, Sept. 1985.

[37] B.-G. Lee and Y. Park, "Degree elevation of B-spline curves and its matrix representation," *Journal of the Korean Society for Industrial and Applied Mathematics*, vol. 4, no. 2, pp. 1–9, 2000. 00001.

[38] N. Zander, T. Bog, M. Elhaddad, F. Frischmann, S. Kollmannsberger, and E. Rank, "The multi-level hp-method for three-dimensional problems: Dynamically changing high-order mesh refinement with arbitrary hanging nodes," *Computer Methods in Applied Mechanics and Engineering*, vol. 310, pp. 252–277, Oct. 2016.

[39] M. Nesme, F. Faure, and Y. Payan, "Hierarchical Multi-resolution Finite Element Model for Soft Body Simulation," in *2nd Workshop on Computer Assisted Diagnosis and Surgery*, pp. 40–47, Springer Berlin Heidelberg, Mar. 2006.

[40] D. Schillinger, P. K. Ruthala, and L. H. Nguyen, "Lagrange extraction and projection for NURBS basis functions: A direct link between isogeometric and standard nodal finite element formulations," *International Journal for Numerical Methods in Engineering*, vol. 108, pp. 515–534, Nov. 2016.

[41] T. Lyche and K. Morken, "Spline methods draft." www.uio.no/studier/emner/matnat/ifi/INF-MAT5340/v13/undervisningsmateriale/book.pdf, 2011 (Online; accessed 20.02.17)., 2008.

[42] G. Casciola and L. Romani, "A general matrix representation for non-uniform B-spline subdivision with boundary control," techincal report, ALMA-DL University of Bologna, 2007. amsacta.unibo.it/2532/.

[43] M. R. Dörfel, B. Jüttler, and B. Simeon, "Adaptive isogeometric analysis by local h-refinement with T-splines," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, pp. 264–275, Jan. 2010. 00332.

[44] Z. Yosibash, *Singularities in Elliptic Boundary Value Problems and Elasticity and Their Connection with Failure Initiation*, vol. 37 of *Interdisciplinary Applied Mathematics*. New York, NY: Springer New York, 2012. 00030.

[45] P. L. Gould, *Introduction to Linear Elasticity.* Springer-Verlag, 1999.