THB-splines: An Effective Mathematical Technology for Adaptive Refinement in Geometric Design and Isogeometric Analysis

Carlotta Giannelli[a,*], Bert Jüttler[b], Stefan K. Kleiss[b], Angelos Mantzaflaris[c], Bernd Simeon[d], Jaka Špeh[b]

[a] *Istituto Nazionale di Alta Matematica, Unità di Ricerca di Firenze c/o DiMaI 'U. Dini', Università di Firenze, Italy*
[b] *Institute of Applied Geometry, Johannes Kepler University Linz, Austria*
[c] *Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Linz, Austria*
[d] *Department of Mathematics, Technische Universität Kaiserslautern, Germany*

Abstract

Local refinement with hierarchical B-spline structures is an active topic of research in the context of geometric modeling and isogeometric analysis. By exploiting a multilevel control structure, we show that truncated hierarchical B-spline (THB-spline) representations support interactive modeling tools, while simultaneously providing effective approximation schemes for the manipulation of complex data sets and the solution of partial differential equations via isogeometric analysis. A selection of illustrative 2D and 3D numerical examples demonstrates the potential of the hierarchical framework.

Keywords: Adaptivity, Isogeometric analysis, Hierarchical B-splines, Truncated hierarchical B-splines, Local refinement

1. Introduction

Isogeometric Analysis (IgA) is a powerful approach to numerical simulation based on partial differential equations which has attracted substantial interest from the scientific community since its introduction in 2005 [9, 22]. It adopts the mathematical technology of spline functions for representing the unknown quantities that occur in the simulation and for describing the geometry of the computational domain.

The multivariate splines used in geometric design are often based on tensor-product constructions expressed in terms of B-spline representations. These constructions, however, preclude the possibility of adaptive local refinement, since the insertion of a knot in one of the defining univariate spline spaces necessarily introduces new degrees of freedom along an entire hyper-plane in the parameter domain. In order to provide more flexible solutions that may break the rigidity of classical tensor-product construction, an active area of research is currently devoted to the identification of suitable *adaptive spline bases* which allow to locally refine the numerical approximation of the solution without increasing the number of degrees of freedom in the area of the mesh where this is not necessary. The need of reliable adaptive refinement schemes has triggered the introduction of different generalizations of the B-spline model. Several relevant constructions are receiving particular attention: T-splines, hierarchical splines, PHT-splines, LR splines.

T-splines were introduced in [39, 40] as splines based on local knot vectors, which are defined by control meshes that may possess T-junctions. Their first application in IgA was reported in [1, 13]. In

*Corresponding author
Email addresses: carlotta.giannelli@unifi.it (Carlotta Giannelli), bert.juettler@jku.at (Bert Jüttler), stefan_ken.kleiss@jku.at (Stefan K. Kleiss), angelos.mantzaflaris@oeaw.ac.at (Angelos Mantzaflaris), simeon@mathematik.uni-kl.de (Bernd Simeon), jaka.speh@jku.at (Jaka Špeh)

order to obtain nested spaces and to guarantee linear independence, the restricted class of analysis-suitable T-splines was introduced [31]. Later, it was proposed to characterize them as dual-compatible T-splines [2, 3]. Recently, a refinement algorithm in the bivariate case for this class of T-splines with linear complexity has been described [33].

A classical approach to obtain local refinement in geometric modeling is provided by hierarchical B-splines [16, 20, 29]. The construction of the basis guarantees nested spaces and linear independence of the basis functions. The use of hierarchical constructions in isogeometric analysis is a very promising approach [14, 38, 43]. Unfortunately, the partition of unity property is not preserved by the standard hierarchical construction. For this reason, a new hierarchical basis — the truncated basis for hierarchical splines (THB-splines) — has recently been introduced [18]. THB-splines, defined as suitable linear combinations of refined B-splines, form a convex partition of unity, exhibit good stability and approximation properties [17, 42], and are suitable for applications in computer aided design [28]. By providing a way to define an adaptive extension of the B–spline framework which is also suitable for geometric modeling applications, THB-splines satisfy both the demands of adaptive numerical simulation and geometric design, making them well suited for isogeometric analysis. The generalization of THB-splines to the more general context of generating systems and also to geometries with arbitrary topologies was recently addressed [46, 48, 49].

Polynomial splines over hierarchical T-meshes [30] are based on a different paradigm to construct bases for the entire space of piecewise polynomials with a given smoothness on a certain subdivision of the domain. Consequently, nested meshes automatically generate nested spaces. However, the construction of the basis — which is especially tailored to each specific case — either assumes reduced regularity [11] or the satisfaction of certain constraints on the admissible mesh configurations [47]. Applications in isogeometric analysis were reported in [34, 44].

Finally, locally refined (LR) splines rely on the idea of splitting basis functions, and resolve the issue of nested spaces but create difficulties with linear independence [12] that have been further investigated [4, 5]. The use of LR splines in the isogeometric setting was also discussed [23]. A comparison between hierarchical splines, THB-spline, and LR splines with respect to sparsity and condition numbers was presented in [24]. Even if LR splines have smaller supports than THB-splines, that comparison did not reveal significant advantages with respect to sparsity patterns and condition numbers of mass and stiffness matrices.

The present paper is devoted to the truncated basis of hierarchical spline spaces. We show that THB-splines

- possess a firm theoretical foundation with regard to basis construction, nested spaces, partition of unity, stability and approximation properties;
- admit an efficient implementation using standard data structures and require only a modest increase in the computational effort for evaluation compared to standard hierarchical splines;
- are well-suited for geometric modeling and surface reconstruction, due to the convex hull property and the possibility of local refinement;
- are well-suited for adaptive refinement in isogeometric analysis and lead to discretizations that possess good numerical properties.

The structure of the paper is the following. Section 2 introduces the fundamental concepts related to the theory and implementation of THB-splines. Geometric design with THB-splines is discussed in Section 3, while Section 4 is devoted to applications in the context of isogeometric analysis.

2. THB-splines: theory and implementation

We recall the definition and the basic properties of THB-splines and discuss their efficient implementation.

2.1. Definition and basic properties

We define *hierarchical B-splines (HB-splines)* and *truncated hierarchical B-splines (THB-splines)* by describing an algorithm for their evaluation. For a detailed definition and an extensive discussion of (T)HB-splines, we refer to [18, 19] and the references therein. A detailed construction for the bivariate case of bidegree (p, p) has been presented in [28].

We consider a finite sequence of nested d-variate tensor-product spline spaces

$$V^0 \subset V^1 \subset \ldots \subset V^N \tag{1}$$

defined on the domain Ω^0 which is an axis aligned box in $\mathbb{R}^d$. Let

$$\{\beta_{\boldsymbol{i}}^\ell,\ \boldsymbol{i} \in \mathcal{I}^\ell\} \tag{2}$$

be the normalized tensor-product B-spline basis of the space V^ℓ of degree $\mathbf{p}^\ell = (p_1^\ell, \ldots, p_d^\ell)$. The set $\mathcal{I}^\ell$ of multi-indices is defined as

$$\mathcal{I}^\ell = \{\boldsymbol{i} = (i_1, \ldots, i_d),\ i_k = 1, \ldots, n_k^\ell,\ \text{for } k = 1, \ldots, d\},$$

where n_k^ℓ denotes the number of univariate B-spline basis functions in the k-th coordinate direction. We assume a fixed ordering of the index set $\mathcal{I}^\ell$. We can then rewrite the basis (2) as

$$\mathbf{b}^\ell(\mathbf{x}) = \left(\beta_{\boldsymbol{i}}^\ell(\mathbf{x})\right)_{\boldsymbol{i} \in \mathcal{I}^\ell} \tag{3}$$

and consider $\mathbf{b}^\ell(\mathbf{x})$ as a column vector of basis functions. Using this notation, a spline function $s : \Omega^0 \to \mathbb{R}^m$ defined by the basis $\mathbf{b}^\ell(\mathbf{x})$ and the coefficient matrix C^ℓ, whose rows are the coefficients $c_{\boldsymbol{i}}^\ell \in \mathbb{R}^m$, for $\boldsymbol{i} \in \mathcal{I}^\ell$, can be written as

$$s(\mathbf{x}) = \sum_{\boldsymbol{i} \in \mathcal{I}^\ell} \beta_{\boldsymbol{i}}^\ell(\mathbf{x})\, c_{\boldsymbol{i}}^\ell = \mathbf{b}^\ell(\mathbf{x})^T\, C^\ell, \tag{4}$$

where the dimension m of the image of $s(\mathbf{x})$ is determined by the dimension of the coefficients $c_{\boldsymbol{i}}^\ell$. Since $V^\ell \subset V^{\ell+1}$, we can express the basis functions $\mathbf{b}^\ell$ as a linear combination of the basis functions $\mathbf{b}^{\ell+1}$, namely

$$s^\ell(\mathbf{x}) = \mathbf{b}^\ell(\mathbf{x})^T C^\ell = \mathbf{b}^{\ell+1}(\mathbf{x})^T R^{\ell+1} C^\ell, \tag{5}$$

where the entries of the *refinement matrix* $R^{\ell+1}$ can be obtained from well-known B-spline refinement rules, see e.g., [35].

We also consider a corresponding sequence of nested domains

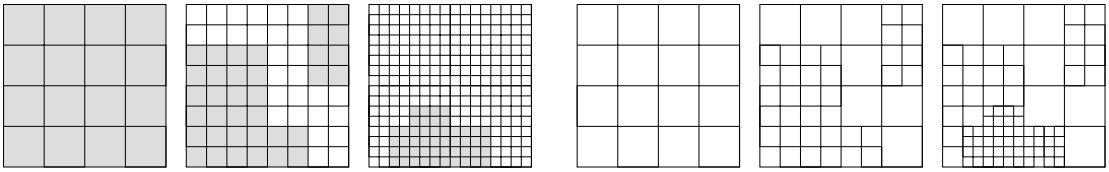
$$\Omega^0 \supseteq \Omega^1 \supseteq \ldots \supseteq \Omega^N, \tag{6}$$

where each Ω^ℓ represents the subset of $\mathbb{R}^d$ covered by a certain collection of cells with respect to the tensor-product grid of level ℓ. An example of a nested sequence of knot lines related to the bivariate case together with a corresponding hierarchical configuration composed of three nested domains is shown in Fig. 1.

Remark 2.1. The multivariate hierarchical model allows to consider different kinds of refinement, degrees, and smoothness as long as the nested nature of the spline spaces (1) is preserved. However, for simplicity, in our examples we will only focus on dyadic cell refinement for the bivariate and trivariate cases with uniform degrees for all levels and coordinates.

Analogously to [27], we define the *characteristic matrix* X^ℓ of $\mathbf{b}^\ell$ (with respect to Ω^ℓ and $\Omega^{\ell+1}$) as the diagonal matrix

$$X^\ell = \operatorname{diag}(x_{\boldsymbol{i}}^\ell)_{\boldsymbol{i} \in \mathcal{N}^\ell},$$



(a) knot lines and domains (shaded areas) for $\ell = 0, 1, 2$ (b) hierarchical meshes for $\ell = 0, 1, 2$

Figure 1: Example of a bivariate hierarchical configuration. The knot lines of the spaces V^ℓ, for $\ell = 0, 1, 2$, are shown from left to right (a) together with three nested domains $\Omega^0 \supseteq \Omega^1 \supseteq \Omega^2$ — shaded areas in (a). The hierarchical meshes at the three refinement levels are shown from left to right in (b).

where

$$x_{\boldsymbol{i}}^\ell = \begin{cases} 1, \text{ if } \operatorname{supp} \beta_{\boldsymbol{i}}^\ell \subseteq \Omega^\ell \wedge \operatorname{supp} \beta_{\boldsymbol{i}}^\ell \not\subseteq \Omega^{\ell+1}, \\ 0, \text{ otherwise.} \end{cases}$$

For each level ℓ, let $\mathcal{I}_*^\ell$ be the set of indices of *active functions* so that $x_{\boldsymbol{i}}^\ell = 1$, i.e.,

$$\mathcal{I}_*^\ell = \{\boldsymbol{i} \in \mathcal{I}^\ell : \ x_{\boldsymbol{i}}^\ell = 1\}.$$

The index set $\mathcal{I}$ of the (T)HB-spline basis functions contains the indices of all active functions at different hierarchical levels. It is defined as follows

$$\mathcal{I} = \left\{(\ell, \boldsymbol{i}) : \ \ell \in \{0, \dots, N\}, \ \boldsymbol{i} \in \mathcal{I}_*^\ell\right\}.$$

The (T)HB-spline basis related to the domain hierarchy (6) is defined as

$$\mathbf{t}(\mathbf{x}) = \left(\tau_{\boldsymbol{i}}^\ell(\mathbf{x})\right)_{(\ell,\boldsymbol{i})\in\mathcal{I}}, \tag{7}$$

where the hierarchical basis functions $\tau_{\boldsymbol{i}}^\ell(\mathbf{x})$ are given by

$$\tau_{\boldsymbol{i}}^\ell(\mathbf{x}) = \begin{cases} \beta_{\boldsymbol{i}}^\ell(\mathbf{x}), & \text{for HB-splines,} \\ \operatorname{trunc}^N(\operatorname{trunc}^{N-1}(\dots \operatorname{trunc}^{\ell+1}(\beta_{\boldsymbol{i}}^\ell(\mathbf{x}))\dots)), & \text{for THB-splines,} \end{cases}$$

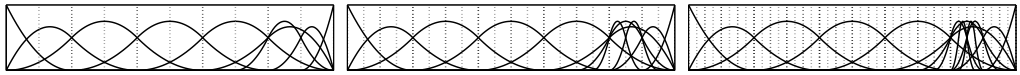
and the *truncation* of any $s(\mathbf{x}) \in V^\ell$ as in (4) with respect to level $\ell + 1$ is defined by

$$\operatorname{trunc}^{\ell+1} s(\mathbf{x}) = \mathbf{b}^{\ell+1}(\mathbf{x})^T \, (I^{\ell+1} - X^{\ell+1}) R^{\ell+1} \, C^\ell, \tag{8}$$

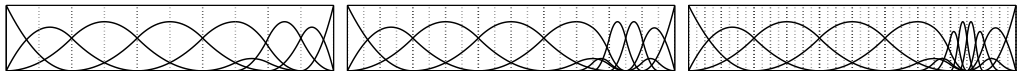
with $I^{\ell+1}$ indicating the identity-matrix of size $|\mathcal{I}^{\ell+1}| \times |\mathcal{I}^{\ell+1}|$. By multiplying the refinement matrix $R^{\ell+1}$ with the coarse coefficient matrix C^ℓ as in (5), we represent $s(\mathbf{x})$ with respect to the finer level $\ell + 1$. In (8), the additional multiplication with $(I^{\ell+1} - X^{\ell+1})$ realizes the truncation operation by setting all coefficients which correspond to active basis functions $\beta_{\boldsymbol{i}}^{\ell+1}$ to zero. A detailed discussion of the truncation operation $\operatorname{trunc}^\ell(\cdot)$ has been presented in [18, 19]. Figure 2 illustrates the effect of the truncation mechanism for a quadratic example in the univariate case. An example of hierarchical and truncated hierarchical B-splines for the bivariate mesh configuration presented in Figure 1 is shown in Figure 3, where the active B-splines of level 0 and 1 are shown before and after truncation.

A multilevel spline function $s(\mathbf{x})$ can be expressed in terms of the basis $\mathbf{t}(\mathbf{x})$ and a coefficient matrix $\widetilde{C} = (\widetilde{c}_{\boldsymbol{i}}^\ell)_{(\ell,\boldsymbol{i})\in\mathcal{I}}$, as

$$\tilde{s}(\mathbf{x}) = \sum_{(\ell,\boldsymbol{i})\in\mathcal{I}} \tau_{\boldsymbol{i}}^\ell(\mathbf{x}) \, \widetilde{c}_{\boldsymbol{i}}^\ell = \mathbf{t}(\mathbf{x})^T \, \widetilde{C}. \tag{9}$$



(a) HB-splines on 2, 3, 4 levels (from left to right)



(b) THB-splines on 2, 3, 4 levels (from left to right)

(c) THB-splines of level 1, 2, 3 influenced by the truncation (from left to right)

Figure 2: Comparison of HB-splines (a) and THB-splines (b) at different refinement levels. THB-splines of coarser levels influenced by the truncation mechanism are also shown (c).

We relate the coefficient matrix $\widetilde{C}$ to level-wise coefficient matrices $C^\ell,\ \ell = 0, \ldots, N$ as follows,

$$C^\ell = (c^\ell_{\boldsymbol{i}})_{\boldsymbol{i}\in\mathcal{I}^\ell}, \quad c^\ell_{\boldsymbol{i}} = \begin{cases} \widetilde{c}^\ell_{\boldsymbol{i}}, \text{ if } \boldsymbol{i} \in \mathcal{I}^\ell_*, \\ 0, \text{ otherwise.} \end{cases} \tag{10}$$

All entries of C^ℓ related to inactive basis functions at level ℓ are zero, while the entries corresponding to active functions are defined by the corresponding coefficients in $\widetilde{C}$. Consequently, the identity

$$C^\ell = X^\ell C^\ell \tag{11}$$

holds. We are now able to formulate Algorithm 1 for evaluating a multilevel spline function $\tilde{s}(\mathbf{x})$ as in (9) in terms of the tensor-product B-spline basis $\mathbf{b}^N(\mathbf{x})$ of the finest level N. It proceeds by iteratively evaluating intermediate spline coefficients D^ℓ using different rules for HB- and THB-splines.

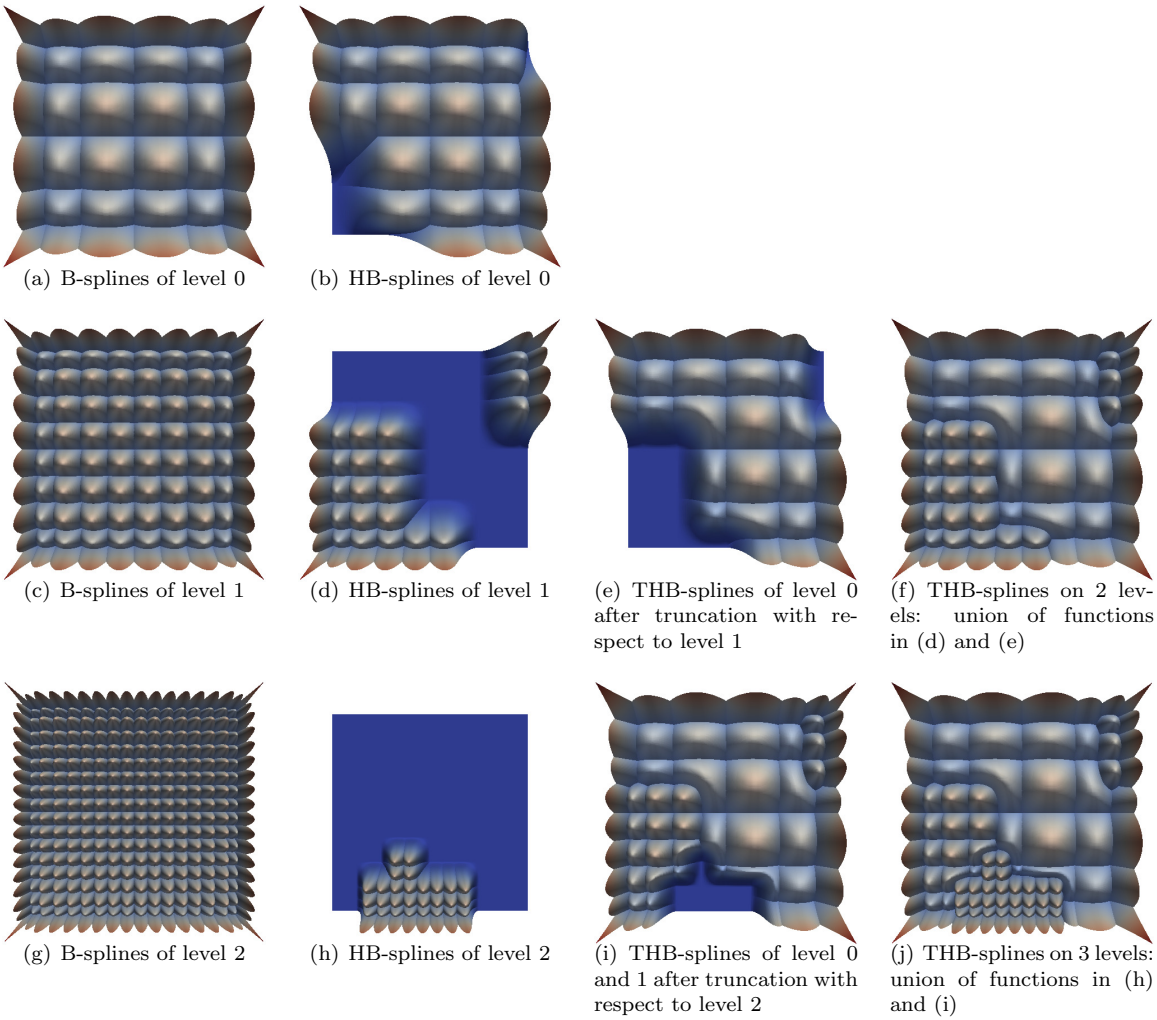
Data: C^ℓ for $\ell = 0, \ldots, N$;
R^ℓ, X^ℓ for $\ell = 1, \ldots, N$;
Result: D^N so that $v(\mathbf{x}) = \mathbf{b}^N(\mathbf{x})^T\, D^N$
$D^0 = C^0$;
for $\ell = 1$ **to** N **do**
  (a) HB-splines: $D^\ell = R^\ell D^{\ell-1} + C^\ell$;
  (b) THB-splines: $D^\ell = (I^\ell - X^\ell)R^\ell D^{\ell-1} + X^\ell C^\ell$;
end
return D^N

Algorithm 1: (T)HB-spline evaluation.

Once the refinement and characteristic matrices given by R^ℓ and X^ℓ, respectively, as well as the coefficient matrices C^ℓ are available, the algorithm is straightforward. The coefficients D^ℓ associated to the (T)HB-spline representation at a given level $\ell > 0$ is just a sum of two contributions. The first one determines the carry-over from the previous level $\ell - 1$, while the second one takes into account the new contributions arising from the current level ℓ. The resulting intermediate coefficient matrix D^ℓ at level ℓ represents the function $s(\mathbf{x})$ at level ℓ and at all coarser levels, i.e., the identity

$$s|_{\Omega^0 \setminus \Omega^{\ell+1}} = (\mathbf{b}^\ell)^T\, D^\ell \tag{12}$$

holds.



(a) B-splines of level 0 (b) HB-splines of level 0

(c) B-splines of level 1 (d) HB-splines of level 1 (e) THB-splines of level 0 after truncation with respect to level 1 (f) THB-splines on 2 levels: union of functions in (d) and (e)

(g) B-splines of level 2 (h) HB-splines of level 2 (i) THB-splines of level 0 and 1 after truncation with respect to level 2 (j) THB-splines on 3 levels: union of functions in (h) and (i)

Figure 3: Illustration of THB-spline basis construction on the domain hierarchy shown in Figure 1. The basis functions are displayed all at once in a top-down view. Note that, while HB-splines at level 2 are given by the union of functions in (b), (c), and (d), THB-splines at level 2 are given by the union of functions in (h) and (i), see also (j).

Note that in the THB-spline evaluation (b) in Algorithm 1 we explicitly use identity (11) to stress that each intermediate coefficient of level ℓ is either obtained by refinement from intermediate coefficients of the previous level, or it is a coefficient of the THB-spline representation (9). The algorithm implicitly defines the (T)HB-basis functions $\tau_{\boldsymbol{i}}^{\ell}$, $(\ell, \boldsymbol{i}) \in \mathcal{I}$, since one can obtain the value of $\tau_{\boldsymbol{i}}^{\ell}$ by applying the algorithm to a characteristic coefficient matrix. More precisely, if the algorithm is applied to evaluate a (T)HB-spline function s as in (9) at a point $\mathbf{x}$, the computations will be restricted only to the B-spline basis functions which contain $\mathbf{x}$ in their support. Furthermore, due to (12), the loop in Algorithm 1 will not necessarily go through all levels from 0 to N, but only up to level ℓ when $\mathbf{x} \in \Omega^{\ell} \setminus \Omega^{\ell+1}$.

2.2. Implementation in G+SMO

We describe an efficient C++ implementation of THB-splines, which is a module of the G+SMO library[1]. The implementation extends the prior 2D implementation presented in [27] to arbitrary spatial dimension d. This code is used for all examples presented in the present paper.

G+SMO is an open-source, object-oriented C++ library for isogeometric analysis. The library makes use of object polymorphism and inheritance techniques in order to support a variety of different discretization bases, namely B-spline, Bernstein, NURBS bases, hierarchical and truncated hierarchical B-spline bases of arbitrary polynomial order. The implementation of basis functions and geometries is dimension-independent, that is, curves, surfaces, volumes, bulks (in 4D) and other high–dimensional objects are instances of code templated with respect to the parameter domain dimension.

Three general guidelines have been set for the development process. Firstly, we promote both efficiency and ease of use; secondly, we ensure code quality and cross-platform compatibility and, thirdly, we always explore new strategies better suited for isogeometric analysis before adopting FEM practices.

The library is partitioned into modules that implement different functionalities. A basic module that is available is the *NURBS module*, which provides a dimension independent implementation of classical tensor-product B-splines and their rational counterpart. On top of the NURBS module we implemented the *hierarchical splines module*, which shall be described in more detail in the sequel.

The hierarchical domain. The backbone of the (T)HB-splines implementation is a suitable representation of the hierarchical domain. This is realized using a binary subdivision tree data structure, which is a generalization of the quad-tree implementation of [27]. The leaves of this tree form a partition of the domain into quadrilateral (in 2D) or cubical (in 3D) subdomains with the property that each subdomain (which is a collection of domain cells) is contained in the same hierarchical level. This allows for fast queries to be implemented, for instance identifying the level of an input cell, or reporting all cells that overlap a given quadrilateral.

In Figure 4 we show an instance of the hierarchical data structure corresponding to the domain of Figure 1. The domain data in this tree are represented by long integers. These refer to knot indices of the corresponding level. Each interior node (circles in Figure 4) stores a split position and the axis index along which the splitting is performed. Leaf nodes (that is, squares in Figure 4) do not store split position or axis, but they store a level index as well as the corner coordinates of the subdomain that it represents.

Note that the leaves are rectangular collections of elements of the same level. The representations of these rectangular subdomains are stored using distinct knot indices. This has the advantage of avoiding numerical errors, even on high levels. Also, note that knot indices on levels higher than zero have the form $2^\ell\kappa$ where κ is an index of a distinct knot in level zero. This allows to perform all related computations by low-level bit operations which are highly efficient. For instance, converting the knot indices between different hierarchical level requires multiplication or division by powers of two, since we restrict ourselves to dyadic refinement. Bit-shift operations provide an efficient way to implement these conversions.

The hierarchical data structure is initialized by the initial tensor-product mesh Ω^0 as in Figure 1(a). Then an insertion operation can be used to insert new subdomains in the hierarchy. Furthermore the set of active basis functions and the corresponding characteristic matrices X^ℓ for all levels ℓ can be efficiently extracted. This is done during a basis compilation (or initialization) step, after the box insertion.

Basis compilation step. After constructing the hierarchical domain, a basis compilation step takes place. In this step, the characteristic matrices X^ℓ for all levels ℓ are constructed and stored in a sparse format.

[1] *Geometry + Simulation Modules,* `gs.jku.at/gismo`

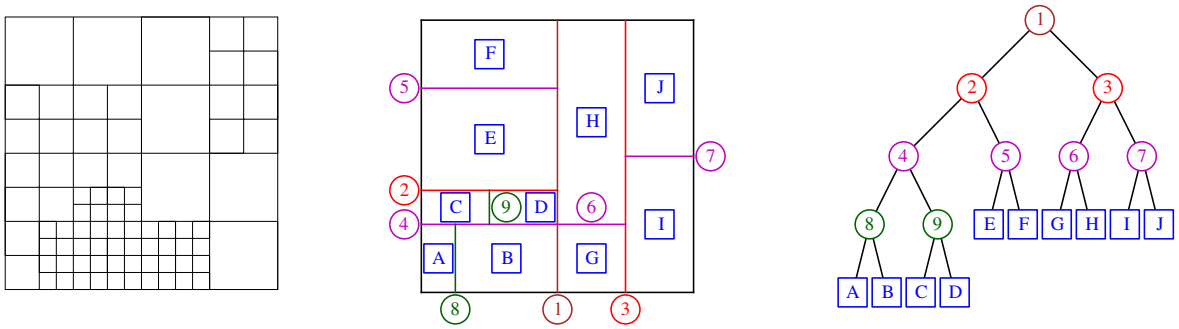


Figure 4: The partition of a domain and the binary subdivision tree for the example in Figure 1 (mesh on the left).

In the case of THB-splines, we identify the subset of basis functions that need to be truncated. This is done by a support overlap query on the tree structure. For an efficient evaluation procedure, we proceed as follows: for a truncated basis function, we identify the coarsest level ℓ_c such that it admits a representation in terms of B-spline functions of that level. This extra information is collected during the support overlap query. Consequently, we precompute and store this representation in level ℓ_c in order to speed up subsequent evaluations. This computation is done according to Algorithm 1, by setting all input coefficients C^ℓ to 0 except for the coefficient of the desired basis function which is set to 1 and then iterating for all levels $N \leq \ell_c$. The algorithm has then to be executed only once, and subsequent function evaluations are reduced to computing linear combinations of (typically few) tensor-product B-spline basis functions.

For storage and data exchange we do not need to store the precomputed B-spline representations of the truncated basis functions, as this information can always be recomputed from the domain hierarchy or from the characteristic matrices. Thus, using THB-splines does not increase the required memory for storage and data exchange. When performing computations and evaluations, however, the precomputation of THB-splines, which increases the efficiency, leads to additional memory consumption, see the experimental results reported below. For sufficient mesh grading, the memory still grows linearly with the number of degrees of freedom, see Fig. 7.

Point-wise evaluation of basis functions and (T)HB-splines. The evaluation of a tensor-product B-spline basis function in a HB-spline or THB-spline basis is done via the recursive definition. For a truncated basis function we use the precomputed coefficients to obtain its value as a linear combination of tensor-product B-splines at the representation level ℓ_c.

If all active basis functions at the given point are requested, it is often the case that several of them possess the same representation level. In this case, we exploit this fact by caching the values of active B-splines at the representation level, in order to increase efficiency.

The evaluation of a field at given point is performed by Algorithm 1. The implementation takes into account the level of a point and stops the iterations there. Without precomputation of the B-spline representation of the truncated basis functions, the cost of the THB–spline evaluation is equal to the application of the B-spline subdivision rule ("level of a point" times) plus the cost of the standard de Boor's algorithm, see also [27] for a detailed discussion. When using precomputation, this is reduced to the cost of the standard de Boor's algorithm.

Adaptive refinement: box insertion. Inserting boxes into the domain structure is the basic tool for performing adaptive refinement. It takes place as soon as subdomains need to be added in levels higher than zero. Firstly, quadrilateral subdomains are inserted at higher levels in the binary subdivision tree. Secondly, the resulting domain structure is used to update locally the characteristic matrices.

In the first step, new nodes are created with split positions and axes adapted to the input box. More precisely, the new box is tracked from the root of the tree until the leaves, and all overlapping

leaves are subdivided at a split positions and axes which are chosen according to a corner of the inserted box. If the inserted box is not aligned with the current hierarchical mesh, L-shaped cells could occur. To overcome this, affected cells of lower levels are locally subdivided to adapt to the inserted box, see Figure 5. Again, this operation breaks down to bit computations that can be efficiently implemented.

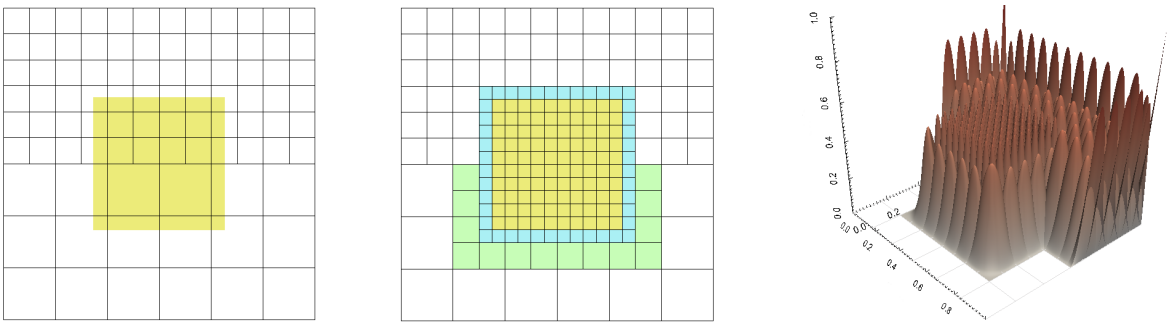


Figure 5: Left: the shaded box is to be inserted in level two. Middle: mesh after insertion; cells around the inserted area are subdivided to adapt the mesh. Right: the basis functions of the two finest levels of the resulting basis.

In practice a large number of small, possibly neighboring boxes will be inserted at a single step, for instance areas marked by a local error estimator. This can result in redundant nodes in the tree, that can be merged into a single leaf. Indeed, a "tree-compression" step is executed before the basis compilation step. In this step siblings that possess the same level are merged reccursively into their parent node, which becomes a leaf itself. This restores a compact tree representation.

The final step after the construction of the refined mesh is to update the characteristic matrices, and compute any new truncated coefficients, as described in the basis compilation step.

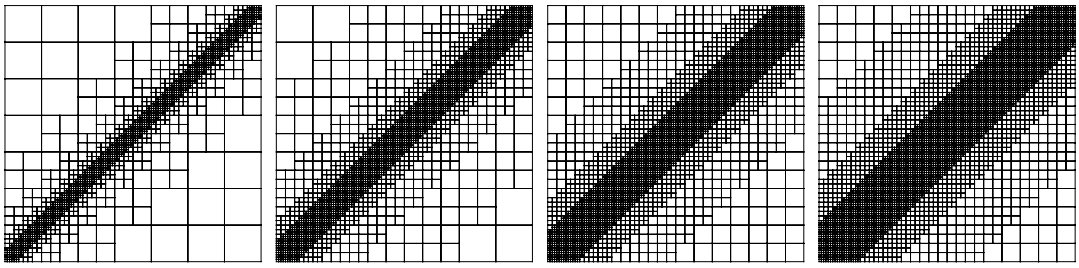
Storage format and data exchange. Data exchange is important in the design and analysis pipeline. THB-spline data can be exported to tensor-product patches and stored in standard CAD formats [28]. For the purpose of reliable exchange without numerical ambiguity, an XML format is also provided in G+SMO. This format stores the coarse tensor-product B-spline basis plus the leaves of the subdivision tree, i.e., the domain structure. In addition, the user can write (or generate) such a file by adding arbitrary "box" tags to an initial mesh, without respecting a tree structure. Upon loading the file, all input boxes are inserted in the tree structure, therefore reconstructing the hierarchical domain.

Our storage format supports robust transfer of the same basis between different hardware. Indeed, as already mentioned, the boxes (leaves) are represented by integer coordinates referring to distinct knot indices.

The local THB-spline evaluation in terms of B-spline patches can be combined with current CAD kernels in order to convert the multilevel spline representation into a standard CAD geometry. The software interface that may be used to connect the G+SMO library to commerical CAD systems, such as Parasolid^{TM} has been described in [28]. The conversion between hierarchical splines and T-splines was addressed in [45].

Experimental results. We compare the computational performance of HB- and THB-splines based on our current implementation. All computations are made on the domain with diagonal refinement shown in Figure 6. The four meshes realize different mesh grading. More precisely, the distance between cells of different levels increases from left to right.

The plots in Figure 7 compare three different aspects of our implementation: memory consumption, precomputation time and evaluation time. The required memory comprises the characteristic matrices and — in the case of THB-splines — the precomputed representations of the basis functions. Figure 7 (first row) shows that the memory consumption grows linearly for HB-splines and for THB-splines with sufficient mesh grading (meshes (c) and (d) for $p = 2$, and mesh (d) for $p = 3$). This is due to



(a) (b) (c) (d)

Config.	Number of levels								
	0	1	2	3	4	5	6	7	8
(a) p=2	81	164	324	638	1260	2498	4968	9902	19764
(b) p=2	81	222	524	1148	2416	4972	10104	20388	40976
(c) p=2	81	256	676	1586	3476	7326	15096	30706	61996
(d) p=2	81	256	770	1942	4430	9550	19934	40846	82814
(a) p=3	100	177	310	555	1024	1941	3754	7359	14548
(b) p=3	100	247	541	1129	2305	4657	9361	18769	37585
(c) p=3	100	289	709	1591	3397	7051	14401	29143	58669
(d) p=3	100	289	811	1963	4375	9307	19279	39331	79543

Figure 6: Top: Four different hierarchical mesh configurations for the diagonal refinement of the unit square with increasing distance between cells of different levels (a-d). Bottom: Number of degrees of freedom for degrees $p = 2$ and $p = 3$.

the fact that the repeated truncation leads to exponentional memory growth for the involved basis functions, while sufficient mesh grading limits the number of truncations for each basis functions.

The precomputation time (second row in Figure 7, in seconds) is needed for initializing the basis. For HB-splines, this includes the construction of the characteristic matrices, the knot vectors and the tree structure. For THB-splines, it also covers the time needed to evaluate the representation of each basis function. We observe that THB-splines require some overhead but the precomputation time for THB-splines does not increase faster than for HB-splines. Using sufficient mesh grading again reduces this overhead.

The evaluation time (bottom row in Figure 7, in seconds) comprises the time needed to evaluate all basis functions at the central point of all elements. We observe that the evaluation times for HB- and THB-splines are quite similar.

3. Geometric design with THB-splines

HB-splines are nonnegative, linearly independent [43], and their span contains all piecewise polynomial functions defined on a certain class of suitable underlying hierarchical mesh [17, 32]. However, these functions do not form a partition of unity, since the sum of hierarchical B-splines at different refinement levels may be greater than one.

THB-splines preserve the non-negativity and linear independence of HB-splines. Moreover, they also form a partition of unity [18] and provide improved stability properties [19]. Consequently, representations of curves and surfaces which are based on these functions prossess the convex hull property and are invariant under affine transformations. These are essential features for geometric modeling with adaptive spline bases.

3.1. Multilevel editing

A fundamental requirement in applications related to geometric modeling is to describe the geometry of a one or multi–dimensional parametric object in terms of a *control net* defined by a certain

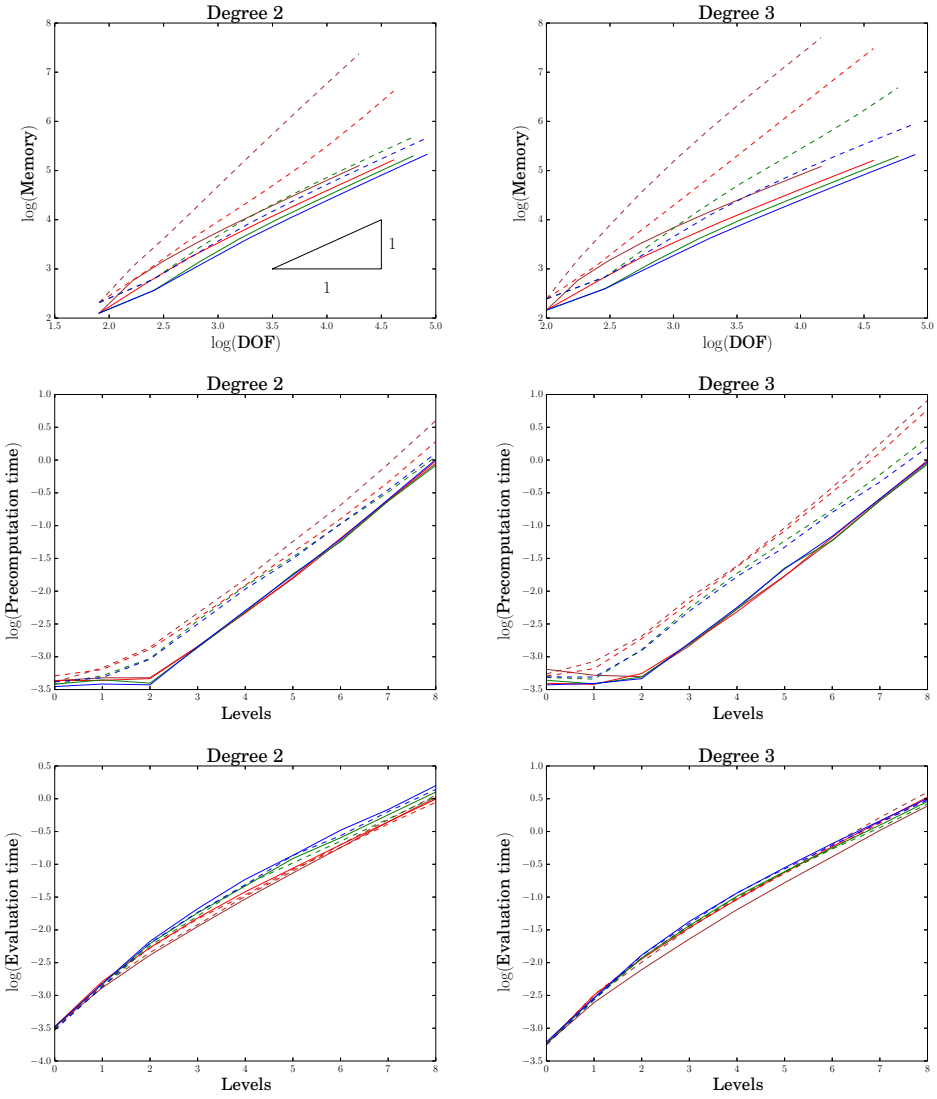


Figure 7: Experimental results for HB-splines (solid lines) and THB-splines (dashed lines) for the four hierarchical mesh configurations shown in Figure 6. The brown, red, green and blue color corresponds to the meshes (a-d), respectively.

collection of points. Consequently, manipulations of the parametric shape can be replaced by analogous (but simpler) operations on this modeling tool. In order to achieve this, any point along the parametric form p is represented as a convex linear combination of the considered control points $d_{\boldsymbol{i}}^{\ell}$ in terms of nonnegative basis functions which satisfy the partition of unity property,

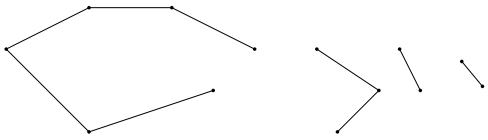
$$p(\mathbf{x}) = \sum_{(\boldsymbol{i},\ell)\in\mathcal{I}} \mathbf{d}_{\boldsymbol{i}}^{\ell} \tau_{\boldsymbol{i}}^{\ell}(\mathbf{x}).$$

The generalization to rational forms is also possible and analogous to NURBS curves and surfaces.

The convex hull property of THB-splines allows us to introduce the concept of a *multilevel control structure*, defined by the set of control points (and the edges connecting them) associated to the truncated basis function level by level. The three following examples show how this hierarchical control tool can be effectively used to perform interactive editing and design.

Example 1. In the univariate case, B-spline geometries can easily be manipulated through interactive and local editing of their control polygons without the need of additional extensions to hierarchical configurations. Nevertheless, in order to start with a simple example, we consider the successive modification of a B-spline curve according to a hierarchical control structure that consists of 4 different refinement levels in Figure 8. The control structure is shown in the top row. The HB- and THB-spline curves defined by considering the set of control points up to level 2, 3, and 4 are shown from left to right in Figure 8(b) and (c), respectively.

Since HB-splines do not form a convex partition of unity, the curve is not confined to the convex hull. Moreover, it is not affinely invariante, hence even a simple translation of the curve would produce a different curve. In contrast, the THB-spline representation is suitable for geometric modeling. The curve is contained in the convex hull of the corresponding multilevel control polygon and the representation is invariant under affine transformations.



(a) multilevel control polygon with four levels

(b) HB-spline curve on 1-4 levels (from left to right)

(c) THB-spline curve on 1-4 levels (from left to right)

Figure 8: Comparison of HB-spline (b) and THB-spline (c) representations at different refinement levels obtained by applying multilevel editing to the same initial B-spline curve and corresponding control polygon (a).

Example 2. We now present an example for local multilevel editing in the bivariate case. We start by considering the wine-glass-shaped B-spline surface shown on top the left of Figure 9. Several subsequent refinement levels are considered: the locally refined hierarchical meshes and the corresponding multilevel control nets defining the same THB-spline geometry are there shown (top center). In order to modify the bowl of the glass without propagation to the stem or the foot, we exploit the local nature of adaptive THB-spline refinements, which allows to insert additional control points only in the upper part of the model (middle row). Subsequently, these newly inserted control points can be interactively moved to change the shape of the bowl and insert an additional local feature (top right and bottom right). Using the standard tensor-product B-spline representation would have required a global refinement of the mesh.

Example 3. As second bivariate example of adaptive modeling, we present the editing of the bulbous bow at the front of a ship. Figures 10(a) and 10(b) illustrate the initial ship hull and the corresponding tensor-product grid, Figure 10(c) and 10(d) show the locally refined geometry and the hierarchical mesh after three refinement steps. In order to model the bulbous bow at the front of the ship, the control points on the left bottom corner are suitably changed, see Figures 10(e) and 10(f). Note that

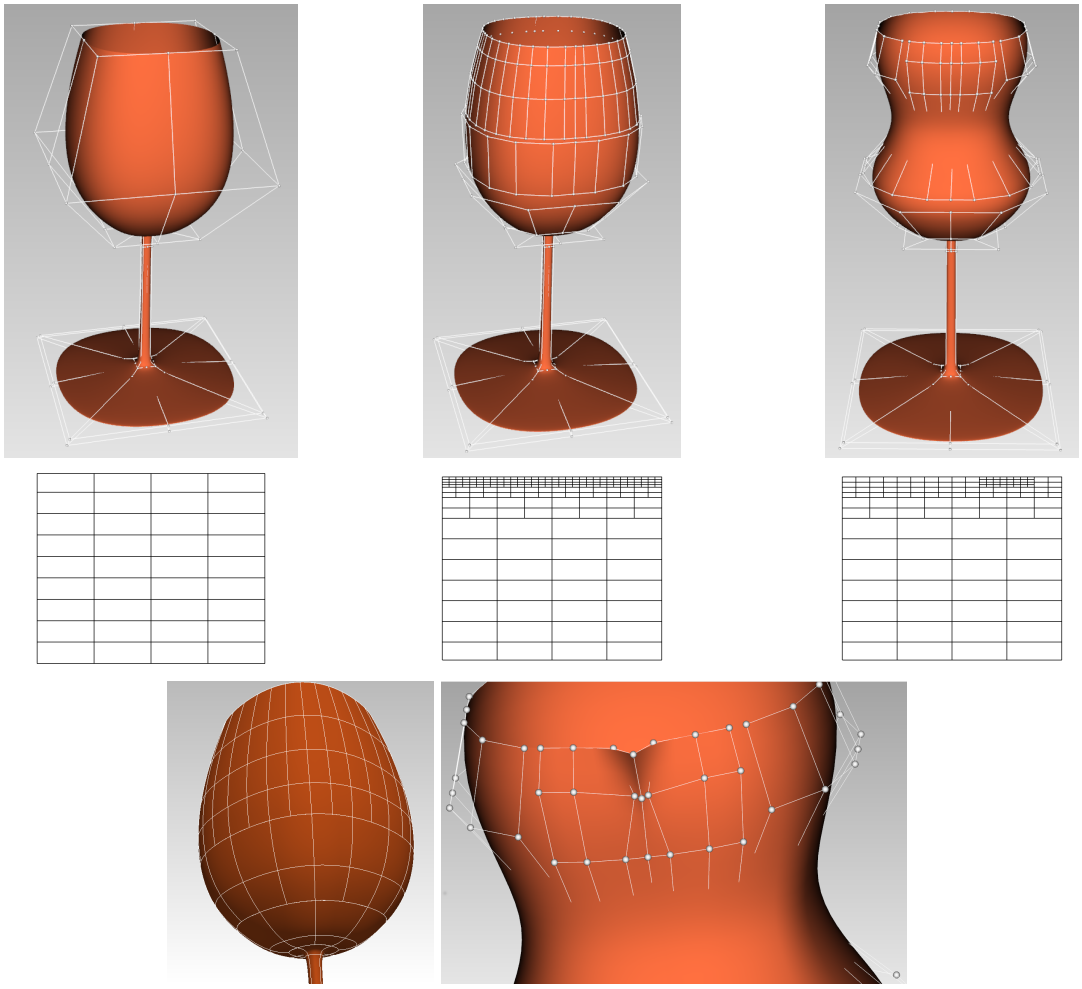


Figure 9: THB-spline representations of a wine glass (top row) using three different hierarchical meshes in the parameter domain (middle row). The images of the knot lines in the hierarchical mesh exhibit T-joints on the surface (bottom left). The THB-spline control mesh is suitable for interactive shape editing (bottom right).

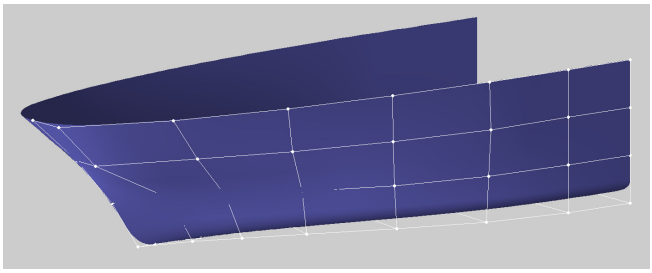
the control net shown in (c) and (e) has 193 control points. Three steps of global refinement would lead to 1300 degrees of freedom.

3.2. Reconstruction of geographic data

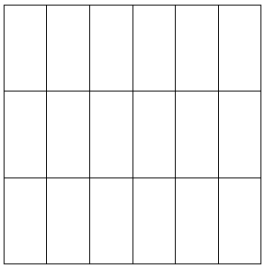
Adaptive spline models may be suitably exploited also for an effective reconstruction of complex models starting from large data sets. For example, the use of LR spline representation in the context of geographical data approximation has been recently addressed [41]. The next example demonstrates that adaptive refinement with hierarchical splines allows to obtain high accuracy with a reduced number of degrees of freedom when compared to classical tensor-product approximations.

Example 4. In order to compare the adaptive behaviour of hierarchical constructions with the uniform B-spline scheme, we compute a regularized least squares approximation in terms of THB–splines [28] for the Baltic sea data set `iowtopo1` consisting of 133,200 (spherical) gridded data points[2]. The

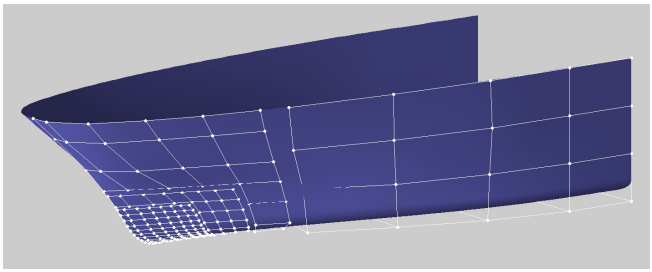
[2] The data set is available at `www.io-warnemuende.de/topography-of-the-baltic-sea.html` (Leibniz Institute for Baltic Sea Research Warnemünde, Digital Topography of the Baltic Sea).



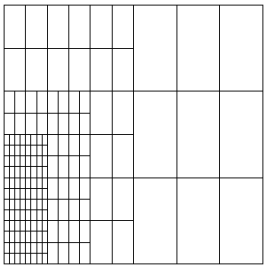
(a) initial model of the ship hull



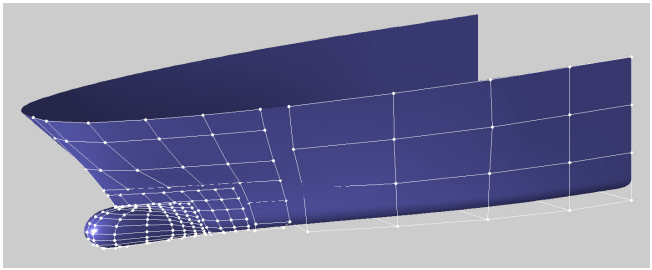
(b) initial tensor-product mesh



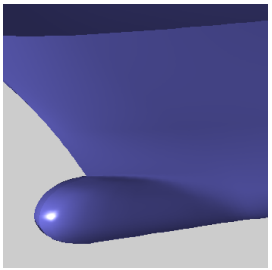
(c) ship hull model after 3 steps of local refinement



(d) hierarchical mesh



(e) ship hull model with local feature



(f) zoomed view of the bulbous bow

Figure 10: THB-spline representation of a ship hull (a) related to a tensor-product grid (b) — due to symmetry, only the control points for one side of the hull are considered. The geometry and the corresponding hierarchical mesh after three steps of local refinement are shown, (c) and (d), respectively. The bulbous bow shown in (e) and (f) is designed by local editing of the control points close to the left bottom corner of the model.

regularization term prevents the linear system associated with the approximation problem to become singular. In this way, even when the number of degrees of freedom exceeds the number of available data points, we are able to compute the resulting spline geometry. Table 1 compares the number of degrees of freedom and maximum error obtained by applying the adaptive fitting framework with THB-splines and tensor-product B-splines. Global and local degrees of freedom refer to the number of B-splines and THB-splines at the different levels, respectively. The percentage of points below an error threshold of 10^{-3} is also shown (the parameter domain and the data set are scaled to $[0,1]^2$ and $[-1,1]$, respectively, for better numerical accuracy, the zero value corresponds to the sea level). The reconstruction result is shown in Figures 11 and 12.

4. Isogeometric Analysis with THB-Splines

We study the numerical properties of HB-spline and THB-spline discretizations in isogeometric analysis. Special attention is paid to sparsity patterns and condition numbers of the matrices which

refinement step	no. of degrees of freedom		maximum error		% below threshold	
	local	global	local	global	local	global
0	100	100	1.02e-01	1.02e-01	10.44	10.44
1	324	324	9.59e-02	9.59e-02	16.58	16.58
2	1,156	1,156	8.68e-02	8.68e-02	27.62	27.62
3	4,356	4,356	6.82e-02	6.82e-02	41.23	41.23
4	16,398	16,900	6.38e-02	6.38e-02	57.62	57.63
5	57,034	66564	2.79e-02	2.79e-02	77.24	77.29
6	164,011	264,196	6.27e-03	6.27e-03	98.58	98.83

Table 1: Number of local (hierarchical) and global (tensor-product) degrees of freedom and related maximum error with respect to different refinement steps obtained by sampling 133,200 data points from the Baltic sea data set considered in Example 4. The percentage of points (%) below the error threshold of 10^{-3} is also shown.

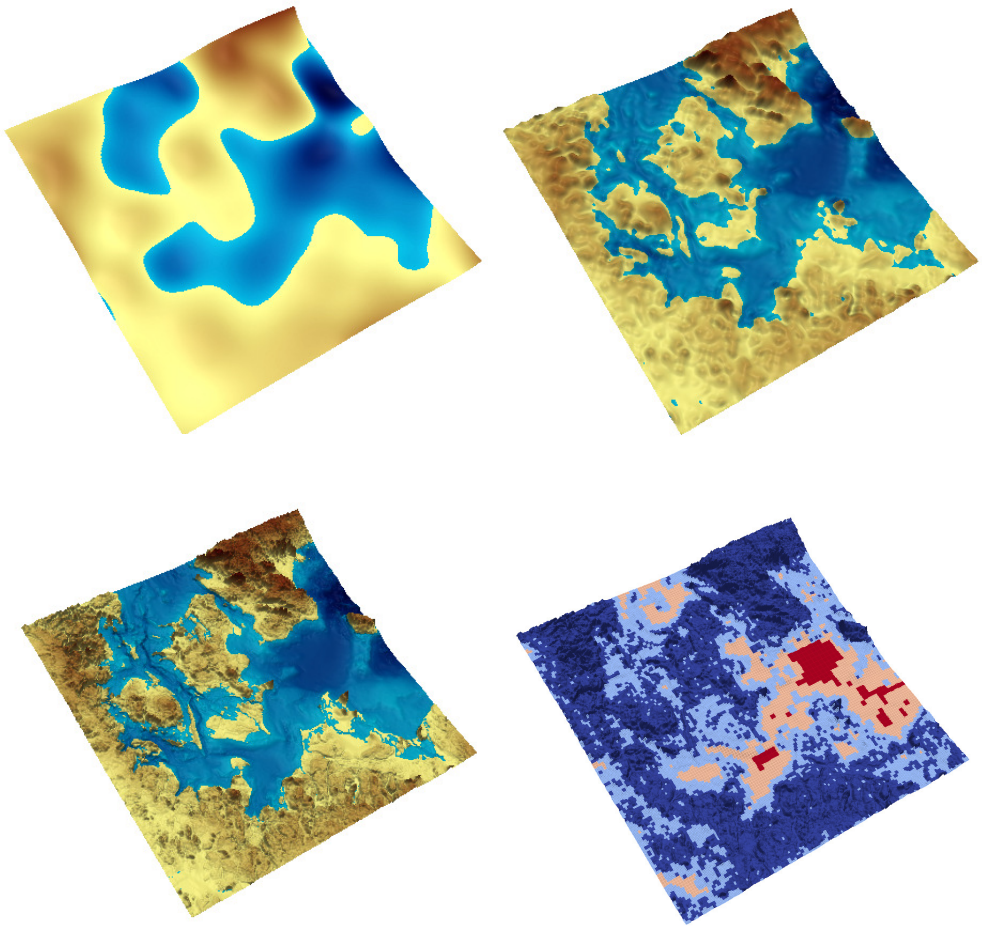


Figure 11: THB-spline approximations of the baltic sea data set considered in Example 4 at step 0, 3 and 6 (top row and bottom left). The corresponding multilevel control mesh at level 6 is also shown (bottom right). The different colors identify the control meshes of the different levels, where the color varies from red to dark blue as the level increases.

need to be generated when solving PDEs using Galerkin discretization. A comparison between HB-splines, THB-splines, and LR splines was recently presented in [24]. The results of the following

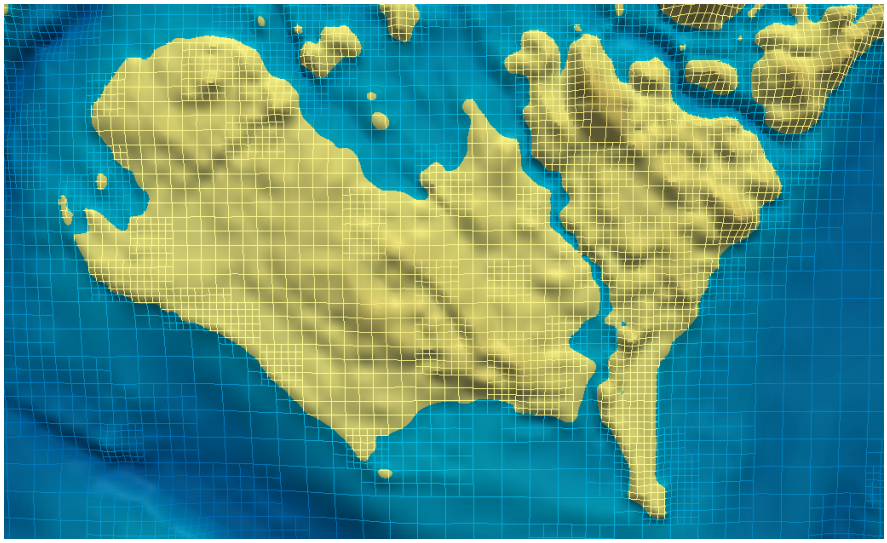


Figure 12: Close-up view at adaptive reconstruction of the Lolland and Falster islands, Denmark.

examples extend the analysis for the hierarchical case to more complex geometries, 3D configurations, and different model problems.

4.1. Model problem and discretization

We will consider several model problems, namely the Laplace equation, the Poisson equation, and advection-diffusion problems. These problems can be written in the following general form:

$$\text{Find } u \in V = C^2(\Omega) \cap C(\overline{\Omega}), \text{ such that } \quad \begin{cases} \mathcal{L}u = f & \text{in } \Omega, \\ u = g_D & \text{on } \partial\Omega, \end{cases} \tag{13}$$

where $\mathcal{L}$ is an elliptic second-order differential operator. In order to keep the presentation concise we restrict ourselves to Dirichlet boundary conditions.

By using standard techniques, we reformulate (13) in the following *variational form*:

$$\text{Find } u \in V_g \subset V = H^1(\Omega), \text{ such that } \quad a(u,v) = \langle f, v \rangle, \quad \forall v \in V_0 \subset V, \tag{14}$$

where $a(\cdot,\cdot)$ and $\langle f, \cdot \rangle$ are the bilinear form and the linear functional induced by the considered PDE, V_0 denotes the space of test functions which vanish on the Dirichlet boundary, and where V_g is the set of functions fulfilling the Dirichlet boundary conditions.

Using Galerkin discretization, we obtain from (14) the linear system of equations

$$L_h \, \mathbf{u}_h = \mathbf{f}_h. \tag{15}$$

We refer to L_h as *system matrix*, and to $\mathbf{f}_h$ as *load vector*. Solving system (15), we obtain the *coefficient vector* $\mathbf{u}_h$, which defines the discrete solution u_h. The system (15) is typically large and sparse, and it is solved by an iterative solver. The performance and convergence rate of such iterative solvers depends, amongst other factors, on the sparsity and on the condition number of the system matrix L_h (see, e.g., [37]).

The discretization space is constructed by employing the isogeometric approach [9]: Given a geometry mapping $F : \Omega' \to \Omega$ that maps the parameter domain Ω' to the physical domain Ω, the discretization space is spanned by the isogeometric functions

$$\hat{B}^\ell_{\boldsymbol{i}} = B^\ell_{\boldsymbol{i}} \circ F^{-1} \quad \text{i.e.} \quad \hat{B}^\ell_{\boldsymbol{i}}(F(\xi)) = B^\ell_{\boldsymbol{i}}(\xi) \quad \forall \xi \in \Omega', \tag{16}$$

where the basis functions $B_{\boldsymbol{i}}^{\ell}$ are (truncated) hierarchical B-splines (or rational versions thereof), $B_{\boldsymbol{i}}^{\ell} = \beta_{\boldsymbol{i}}^{\ell}$ or $B_{\boldsymbol{i}}^{\ell} = \tau_{\boldsymbol{i}}^{\ell}$, see Section 2. It is also assumed that the geometry mapping F admits a representation in the same space, thereby complying with the isoparametric principle.

In the numerical examples presented below we will report and compare the condition numbers and number of non-zero entries related to L_h, or the *mass matrix* M_h and the *stiffness matrix* K_h,

$$(M_h)_{(\boldsymbol{i},\ell),(\boldsymbol{j},k)} = (\hat{B}_{\boldsymbol{i}}^{\ell}, \hat{B}_{\boldsymbol{j}}^{k})_{L_2} = \int_{\Omega} \hat{B}_{\boldsymbol{i}}^{\ell} \hat{B}_{\boldsymbol{j}}^{k}\, dx, \quad (K_h)_{(\boldsymbol{i},\ell)(\boldsymbol{j},k)} = \int_{\Omega} \nabla \hat{B}_{\boldsymbol{i}}^{\ell} \cdot \nabla \hat{B}_{\boldsymbol{j}}^{k}\, dx, \quad (\boldsymbol{i},\ell),(\boldsymbol{j},k) \in \mathcal{I}, \tag{17}$$

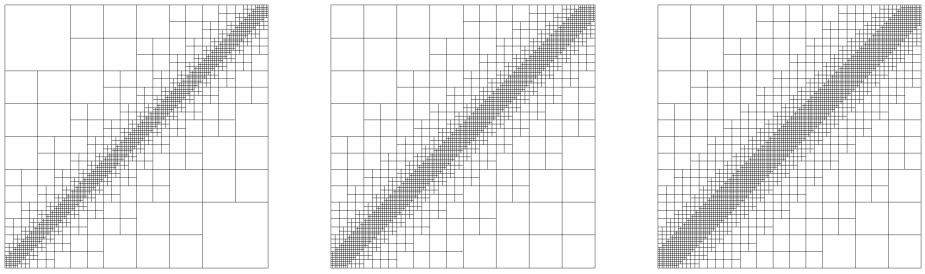
for HB- or THB-splines, namely for $B_{\boldsymbol{i}}^{\ell} = \beta_{\boldsymbol{i}}^{\ell}$ or $B_{\boldsymbol{i}}^{\ell} = \tau_{\boldsymbol{i}}^{\ell}$. Note that the values reported in the column labeled "# d.o.f." of the following tables refer to the total number of degrees of freedom of the corresponding discrete space. All other values (i.e., numbers of non-zero entries, bandwidth, and condition numbers) are computed after elimination of all degrees of freedom associated with the Dirichlet boundary.

4.2. Ad hoc refinement

In the first two examples, we apply ad hoc refinement on the unit hypercube, i.e., $\Omega = (0,1)^d$, $d = 2, 3$, which is discretized with HB- and THB-splines of degrees $p = 2, 3, 4$ using the identity as the geometry mapping.

Example 5 (Ad hoc refinement on unit square)**.** We consider the discretization of the unit square, i.e., $\Omega = (0,1)^2$, with C^{p-1}-smooth HB- and THB-splines of degrees $p = 2, 3, 4$. Fig. 13 shows the hierarchical meshes after 5 steps of refinement in a strip of $2p + 1$ cells centered at the diagonal.

Table 2 reports the numbers of hierarchical levels and degrees of freedom (d.o.f.) associated to the considered meshes in the columns labeled "L" and "#d.o.f.", respectively. The numbers of non-zero entries, the bandwidth, and the condition numbers of mass and stiffness matrices are presented together with their ratios, provided in the column labeled "$\frac{\text{THB}}{\text{HB}}$". These ratios are also visualized in the plots shown in Fig. 14. Note that these ratios are computed *before* rounding, i.e., the ratios of the presented numbers may slightly differ from the reported ratios. These results show that the use of THB-splines significantly reduces the number of non-zero entries. The condition numbers are also either improved or possess the same order of magnitude, except the case $p = 2$ related to the stiffness matrix.



(a) $p = 2$ (b) $p = 3$ (c) $p = 4$

Figure 13: Meshes after 5 steps of refinement in Example 5 (ad hoc refinement on unit square).

Example 6 (Ad hoc refinement on unit cube)**.** We consider the discretization of the unit cube $\Omega = (0,1)^3$ with HB- and THB-splines of degrees $p = 2, 3, 4$ and global C^{p-1} smoothness. We refine along a layer that is defined by a sphere of radius 0.5, centered at the origin. All cells overlapping this sphere, as well as one layer of cells around those, are refined in each step. The mesh after 3 steps of

		sparsity						condition numbers					
		non-zero entries			bandwidth			stiffness matrix			mass matrix		
L	#d.o.f.	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$
$p = 2$													
0	36	196	196	1.00	11	11	1.00	4.0e+00	4.0e+00	1.00	4.1e+01	4.1e+01	1.00
1	86	1208	1030	0.85	35	22	0.63	2.6e+01	1.1e+01	0.42	2.6e+02	6.0e+01	0.23
2	180	4580	3304	0.72	85	41	0.48	3.7e+01	1.8e+01	0.49	8.4e+02	1.8e+02	0.22
3	362	13856	8734	0.63	181	81	0.45	4.7e+01	4.7e+01	1.00	3.2e+03	7.2e+02	0.22
4	720	37252	20800	0.56	375	160	0.43	5.7e+01	1.1e+02	1.93	1.3e+04	2.9e+03	0.22
5	1430	93312	46462	0.50	767	310	0.40	6.7e+01	2.4e+02	3.60	5.2e+04	1.1e+04	0.22
6	2844	223348	99640	0.45	1555	599	0.39	7.7e+01	5.1e+02	6.61	2.1e+05	4.6e+04	0.22
$p = 3$													
0	49	529	529	1.00	19	19	1.00	3.0e+01	3.0e+01	1.00	4.0e+02	4.0e+02	1.00
1	121	2601	2601	1.00	45	45	1.00	3.0e+01	3.0e+01	1.00	4.4e+02	4.4e+02	1.00
2	253	10195	8477	0.83	90	74	0.82	8.6e+02	3.6e+02	0.42	1.4e+04	5.7e+03	0.40
3	505	32173	22701	0.71	196	145	0.74	1.1e+03	4.3e+02	0.39	4.6e+04	1.8e+04	0.40
4	997	89243	54365	0.61	406	274	0.67	1.2e+03	4.6e+02	0.38	1.8e+05	7.2e+04	0.39
5	1969	228653	121197	0.53	826	483	0.58	1.2e+03	4.6e+02	0.38	7.3e+05	2.9e+05	0.39
6	3901	556419	258365	0.46	1666	1066	0.64	1.3e+03	5.8e+02	0.46	2.9e+06	1.2e+06	0.39
$p = 4$													
0	64	1156	1156	1.00	29	29	1.00	2.7e+02	2.7e+02	1.00	4.1e+03	4.1e+03	1.00
1	144	4900	4900	1.00	64	64	1.00	2.4e+02	2.4e+02	1.00	3.8e+03	3.8e+03	1.00
2	316	20528	17356	0.85	190	118	0.62	9.2e+04	3.9e+04	0.42	1.1e+06	4.9e+05	0.46
3	640	66032	47968	0.73	329	230	0.70	1.1e+05	4.3e+04	0.39	2.8e+06	9.0e+05	0.32
4	1268	184656	118252	0.64	657	446	0.68	1.1e+05	4.3e+04	0.38	1.3e+07	3.4e+06	0.27
5	2504	475216	272536	0.57	1341	809	0.60	1.1e+05	4.3e+04	0.37	4.5e+07	1.4e+07	0.31
6	4956	1160016	599620	0.52	2709	1502	0.55	1.1e+05	4.3e+04	0.37	2.6e+08	5.5e+07	0.21

Table 2: Sparsity properties and condition numbers in Example 5 (ad hoc refinement on unit square).

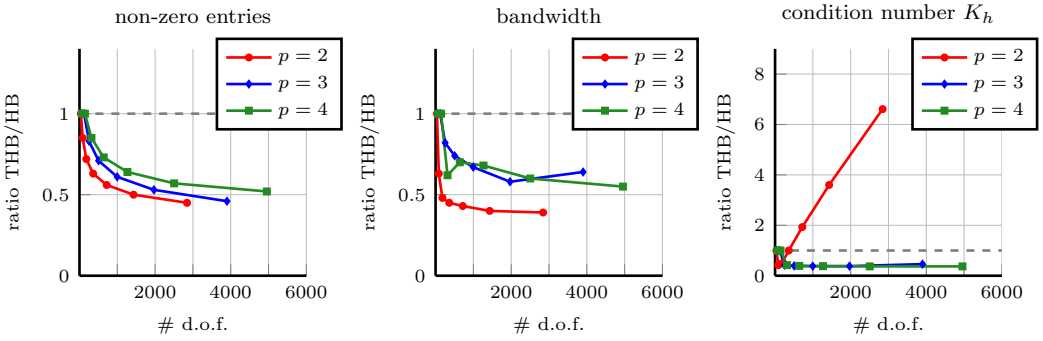


Figure 14: Comparison of number of non-zero entries, bandwidths, and condition numbers of the stiffness matrix K_h in Example 5 (ad hoc refinement on unit square).

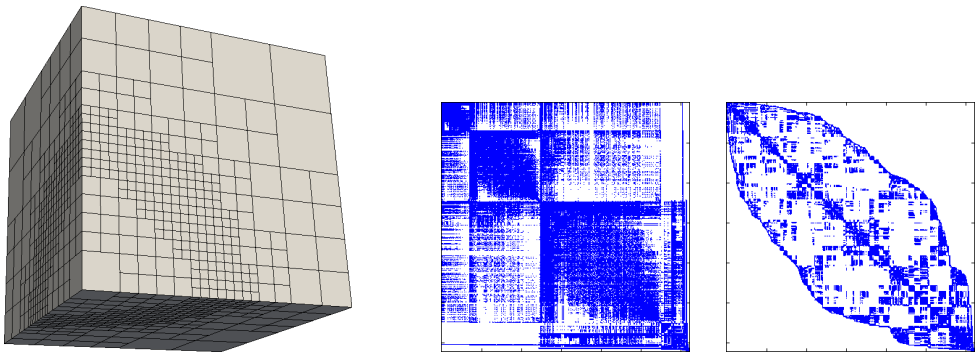
refinement is shown in Fig. 15(a). The comparison of HB- and THB-splines shows a similar behaviour as in Example 5, except the case $p = 2$ that in this example exhibits a ratio for the condition number of the stiffness matrix on the line of other degrees. The ratios of non-zero entries, bandwidths and condition numbers of the stiffness matrix are presented in Fig. 16. Fig. 15(b) and 15(c) show the corresponding sparsity patterns of the mass and stiffness matrix after Cuthill-McKee-reordering at step 3 with $p = 2$.

4.3. Adaptive refinement guided by error estimates

In the following examples, we apply adaptive h-refinement based on a well-known residual-based error indicator (see, e.g., [25]). Assuming we have computed a discrete solution u_h of the model problem (13), the estimate η_K of the local error on a cell K is defined as

$$\eta_K^2 = h_K^2 \| f - \mathcal{L} u_h \|_{L_2(K)}^2, \tag{18}$$

where h_K denotes the diameter of the d-dimensional cell K.



(a) mesh of the physical domain (b) HB-splines (c) THB-splines

Figure 15: Mesh (a) and sparsity patterns of mass and stiffness matrix (after Cuthill-McKee-reordering) related to HB- (b) and THB-splines (c) in Example 6 for $p = 2$ (ad hoc refinement on unit cube) at step 3.

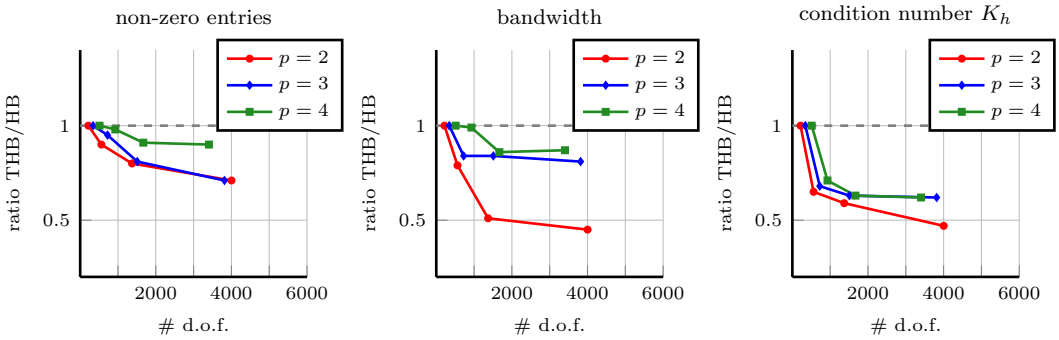


Figure 16: Comparison of number of non-zero entries, bandwidths, and condition numbers of the stiffness matrix K_h in Example 6 (ad hoc refinement on unit cube).

Remark 4.1. Note that residual-based error indicators also involve a term corresponding to the jumps of ∇u_h across cell interfaces, and a term involving Neumann boundary conditions. Both terms vanish when considering at least C^1-continuous discrete solutions and Dirichlet boundary conditions.

A cell K is marked for refinement, if the criterion

$$\eta_K \geq \Theta \tag{19}$$

is fulfilled for a threshold Θ. We use two different strategies for choosing this threshold, which we describe in the following. These strategies depend on a scalar parameter $\psi \in [0, 1]$, where the choices $\psi \approx 0$ and $\psi \approx 1$ result in global and no refinement, respectively.

(i) "Absolute threshold": We compute Θ as

$$\Theta = \psi \cdot \max_K \{\eta_K\}, \quad \psi \in [0, 1]. \tag{20}$$

Note that the percentage of marked cells may vary in each step, since the threshold computation only takes into account the magnitude of the largest local error, without considering the distribution of the estimated error.

(ii) "Relative threshold": We fix a percentage of all cells that should be marked in each step, by choosing Θ such that

$$|\{K : \eta_K > \Theta\}| \approx (1 - \psi) \cdot |\{K\}|$$

For example, if $\psi = 0.8$, the threshold Θ is chosen such that (19) is fulfilled for $20\% = 100 \cdot (1 - \psi)\%$ of all cells. Formally, we express this as

$$\Theta = (100 \cdot (1 - \psi))\text{-percentile}\{\eta_K\}_K, \quad \psi \in [0, 1]. \tag{21}$$

Remark 4.2. Both strategies have advantages and disadvantages. On the one hand, if the local error in a small area dominates the remainder of the error in terms of magnitude, other, possibly larger areas will not be marked when using the first strategy, even if they contribute significantly to the total error. On the other hand, using the second strategy may lead to over-refinement if the error is concentrated in a small portion of the domain.

We only consider dyadic cell refinement. Due to the definition of (T)HB-splines, new basis functions and thus new degrees of freedom are added only if the refined area contains the support of at least one tensor-product B-spline of the next finer level. This, however, is not guaranteed, if dyadic refinement is applied to single cells only. In this case, the refined area might be as small as 2×2 cells (in 2D), which, in general, is smaller than the support of a tensor-product B-spline basis function with degree $p > 1$. To guarantee that the number of degrees of freedom increases, we refine not only the marked cells, but also the ring of one cell around any marked cell. Consequently, the refined area consists of at least 6^d cells of the finer mesh, which is sufficient for basis functions degree $p \leq 5$.

Example 7 (Laplace equation on L-shaped domain)**.** In our first exampe for a posteriori error estimation and adaptive refinement, we consider the Laplace equation

$$\Delta u = 0$$

with Dirichlet boundary conditions on the L-shaped domain shown in Fig. 17 (the bounding box of the domain is given by $[-1, 1]^2$). The domain is represented by globally C^1-continuous splines of degree $p = 2$, using a singular point at the concave corner. The boundary conditions are determined by the exact solution (see Fig. 18)

$$u(r, \phi) = r^{\frac{2}{3}} \sin\left(\frac{2\phi - \pi}{3}\right),$$

which is given in polar coordinates and possesses a singularity at the origin.

This well-known example illustrates the differences between cell marking strategies (i) and (ii), previously introduced. Fig. 19(a) and Fig. 19(b) show the error convergence plots for adaptive refinement with strategies (i) and (ii), respectively, with various values of ψ. In both figures, the error convergence using uniform (global) refinement is indicated by the dashed black line.

Fig. 19(a) shows the results for strategy (i) with parameter values $\psi = 0.1, 0.3, 0.6, 0.9$. Initially, a fast convergence on a degrees-of-freedom-basis can be observed, but the convergence soon stagnates. This is due to the fact that, in this example, the magnitude of the local error estimate in the vicinity of the singularity increases as the mesh around the singularity becomes finer. Once this value dominates the error estimates in other areas of the domain (in terms of magnitude), only few cells around the corner are marked for refinement, and the refined area is not sufficient for a further reduction of the global error.

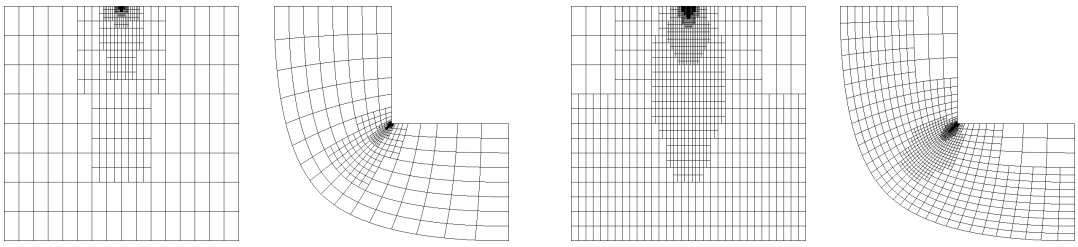
The error convergence for refinement using strategy (ii) with parameter values $\psi = 0.3, 0.6, 0.9$ is presented in Fig. 19(b). Using a small parameter ψ leads to a large number of cells being marked for refinement. In this particular example, this leads to an over-refinement of the domain. With all three choices of ψ, the error is reduced by approximately the same amount in each iteration. However, when small values of ψ are used, many superfluous degrees of freedom are added, thus reducing the convergence on a degrees-of-freedom-basis.

Note that, for strategy (i), the best result, in the sense that the total error is reduced the most, is obtained with the smallest of the tested values, namely $\psi = 0.1$. When using strategy (ii), however,

the fastest convergence on a degrees-of-freedom-basis is achieved by the largest value of ψ, namely $\psi = 0.9$. For this reason, we will compare the results obtained with these two configurations: strategy (i), $\psi = 0.1$, and strategy (ii), $\psi = 0.9$.

The adaptively refined meshes obtained with these two strategies are compared in Fig. 17, and the corresponding sparsity patterns (after Cuthill-McKee-reordering) of mass and stiffness matrices are shown in Fig. 20. The computed numbers of non-zero matrix entries, bandwidths, and condition numbers are reported in Table 3, while the THB vs HB ratios are presented in Fig. 21 and 22. We also present the numbers for discretizations of degrees 3 and 4, i.e., for the computations carried out after applying one or two steps of p-refinement on the geometry mapping and thus on the initial mesh (see, e.g., [9, 10]).

In all tested settings, the use of THB-splines results in a (in some cases substantial) reduction of the number of non-zero matrix entries, and of the bandwidth, compared to HB-splines. The condition numbers of the stiffness and mass matrix are either improved or increased only moderately.



(a) parameter and physical domains — strategy (i) (b) parameter and physical domains — strategy (ii)

Figure 17: Adaptively refined meshes in Example 7 (L-shaped domain). The hierarchical configurations obtained after 8 refinements with strategy (i), $\psi = 0.1$ (a) and after 5 refinements with strategy (ii), $\psi = 0.9$ (b) are shown.

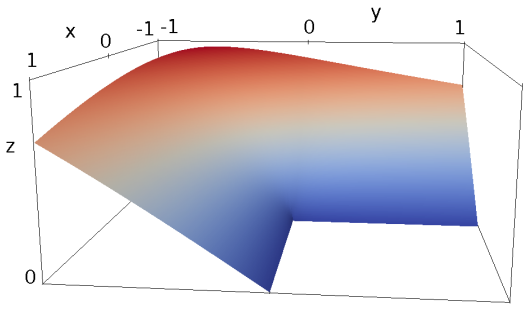
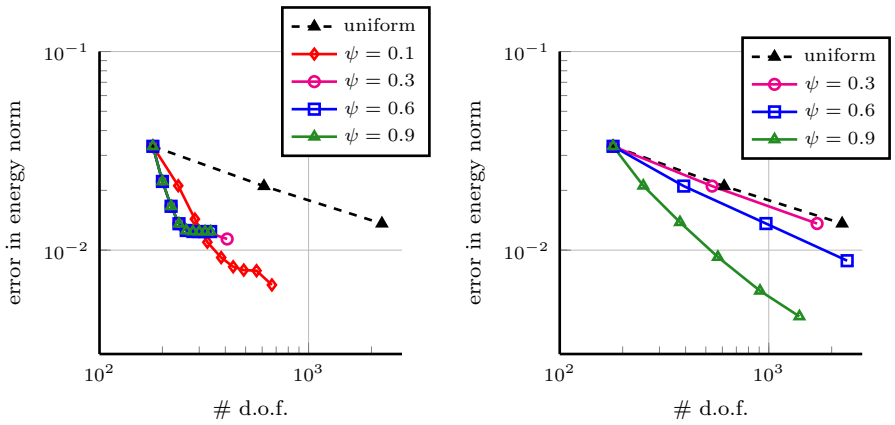


Figure 18: Computed solution in Example 7 (L-shaped domain). A comparison of initial and final solutions is omitted, because the differences are too small to be visible.

Example 8 (Advection-diffusion problem on unit square)**.** This example is also well-known in the context of adaptive refinement and has already been used in numerous publications on isogeometric analysis, e.g., [9, 22]. We consider an advection-diffusion-problem of the form

$$-\kappa\,\Delta u + b \cdot \nabla u = 0, \tag{22}$$

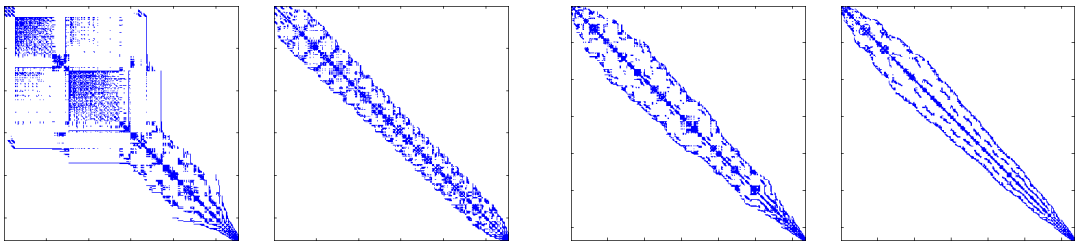
where $\kappa = 10^{-6}$ is the diffusion coefficient, and the advection velocity is given by $b = (\cos\frac{\pi}{4}, \sin\frac{\pi}{4})^T$. The Peclet number Pe is defined by $Pe = L|b|/\kappa$, where L is the side length of the domain. If $Pe \gg 1$, the advection dominates the diffusion, which is clearly the case in our example with $Pe \approx 10^6$. The Streamline Upwinding Petrov Galerkin (SUPG) stabilization method is used for stabilization, i.e., we



(a) stragtegy (i), $\psi = 0.1, 0.3, 0.6, 0.9$.

(b) strategy (ii), $\psi = 0.3, 0.6, 0.9$.

Figure 19: Convergence of error in energy norm with marking strategies (i) and (ii) for different values of ψ in Example 7.



(a) HB- and THB-splines — strategy (i)

(b) HB- and THB-splines — strategy (ii)

Figure 20: Sparsity patterns of system matrix after Cuthill-McKee-reordering in Example 7 (L-shaped domain). The matrices obtained after 8 refinements with strategy (i), $\psi = 0.1$ (a) and after 5 refinements with strategy (ii), $\psi = 0.9$ (b) are shown. In both case the sparsity pattern obtained with HB-splines and THB-spline are presented on the left and right, respectively.

replace the test functions v in the variational problem (14) by test functions $v + \omega\, b \cdot \nabla v$, which have an additional weight in direction of the advection (see, e.g., [22, 36]). The stabilization parameter ω for a cell K is set to

$$\omega(K) = \frac{h_b(K)}{2|b|},$$

where $h_b(K)$ is the length of the cell K in direction of the flow b, and $|b|$ denotes the magnitude of b. We prescribe the following Dirichlet boundary conditions

$$g_D = \begin{cases} 1, & \text{if } y \leq (1-x)/5, \\ 0, & \text{otherwise.} \end{cases}$$

These discontinuous boundary conditions in combination with the strong advection will result in sharp layers. Their expected position is indicated by dashed lines in Fig. 23(a). The mesh after six steps of adaptive refinement with strategy (i), $\psi = 0.1$, is presented in Fig. 23(b). The meshes after three and six steps with strategy (ii), $\psi = 0.7$, are presented in Fig. 23(c) and 23(d), respectively (the intermediate mesh is shown as a reference for the discrete solutions presented in Fig. 26 below). Note that, analogously to Example 7, the meshes are obtained by using different parameters ψ, namely $\psi = 0.1$ for strategy (i), and $\psi = 0.7$ for strategy (ii). The computed non-zero entries, bandwidths and condition numbers are similar for both strategies, and for brevity, we report only the numbers

		sparsity			
		non-zero entries		bandwidth	
L	#d.o.f.	HB	THB	HB	THB
$p = 2$					
0	180	2516	2516	30	30
1	238	4188	3896	57	45
2	286	5632	5000	71	54
3	328	6874	5942	78	69
4	382	8538	7142	101	58
5	436	10266	8342	147	58
6	490	12098	9542	193	58
7	564	14730	11318	244	70
8	668	18424	13812	314	69
$p = 3$					
0	220	5712	5712	48	48
1	250	7224	7048	64	59
2	282	8850	8402	90	70
3	314	10976	9844	116	85
4	346	13670	11238	142	93
5	378	16872	12632	168	96
6	410	20582	14026	194	96
7	462	25680	16640	240	113
8	544	33722	21042	215	108
$p = 4$					
0	264	10780	10780	68	68
1	284	11904	11904	68	68
2	316	14086	14054	94	94
3	348	16996	16860	120	120
4	380	20654	20314	146	146
5	412	25000	24336	172	172
6	444	30034	28926	198	198
7	476	35756	34084	224	224
8	528	44578	41310	270	248

		sparsity			
		non-zero entries		bandwidth	
L	#d.o.f.	HB	THB	HB	THB
$p = 2$					
0	180	2516	2516	30	30
1	251	4611	4259	63	48
2	376	8493	7257	117	64
3	571	13753	11829	108	93
4	908	22564	19968	149	126
5	1402	35778	31790	203	176
$p = 3$					
0	220	5712	5712	48	48
1	291	9831	9137	87	69
2	406	17642	14978	139	102
3	617	30785	24369	223	135
4	929	50639	39931	337	203
5	1443	77633	63491	506	241
$p = 4$					
0	264	10780	10780	68	68
1	335	17281	16485	111	92
2	466	30847	27761	172	141
3	677	55635	44873	259	163
4	1031	88223	71713	361	228
5	1569	142969	114971	527	360

Table 3: Sparsity properties and condition numbers in Example 7 (L-shaped domain). Adaptive refinement with strategy (i) and $\psi = 0.1$ (left), and strategy (ii) and $\psi = 0.9$ (right).

for strategy (ii), $\psi = 0.7$ in Table 4. The ratios are plotted in Fig. 24 and the corresponding sparsity patterns are visualized in Fig. 25. The results show a similar behaviour as in the previous examples, namely a reduction of non-zero entries and bandwidths, and, in almost all cases, a significant reduction of the condition numbers.

In Fig 26, we present the initial, intermediate, and final solutions, both computed with tensor-product B-splines, and with THB-splines. When comparing Fig. 26(b) and Fig. 26(d), the solution computed with THB-splines shows more oscillations near the corner $(1, 0)$, than the solution computed with tensor-product B-splines. This is due to the fact that, at this step, the cells at this corner have not yet been refined to level 3, see in Fig. 23(c). This difference is no more evident after 6 refinements, see Fig. 26(c) and 26(e), respectively.

4.4. Further examples

We present two examples with PDEs solved on more challenging domains.

Example 9 (Advection-diffusion problem on Indiana)**.** In this example, we solve a PDE on the domain representing the state of Indiana, which has been constructed in [15]. The bounding box of the domain is given by $[-0.6, 370.1] \times [-1.6, 577.2]$. In this situation, the mesh referred to as "step 0" is the result of the geometry approximation process, i.e., it is already a hierarchical mesh with seven levels which are caused by the need to capture the geometric details. The numbers of refinement steps reported here refer to the *additional, error estimator-driven refinement steps*, starting from the initial mesh presented in Fig. 27(a) and (b).

We consider the advection-diffusion problem (22) with $\kappa = 1$ and $b = (0.2, -0.1)^T$, i.e., with only mild advection. We prescribe homogenous Dirichlet boundary conditions and the following right-

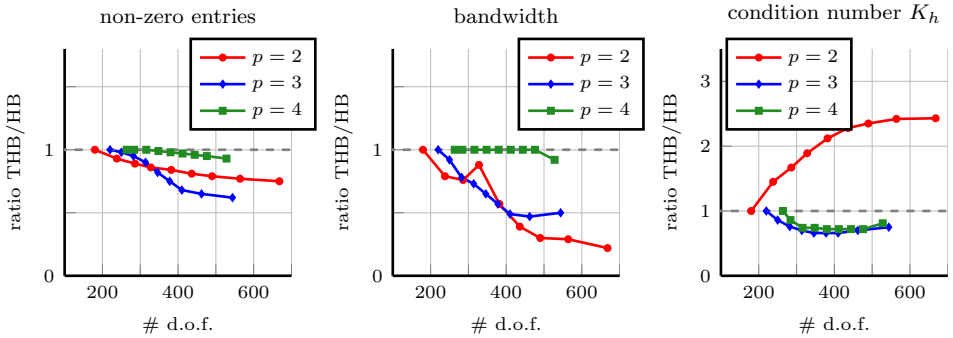


Figure 21: Comparison of number of non-zero entries, bandwidths, and condition numbers of the stiffness matrix K_h in Example 7 (L-shaped domain). Adaptive refinement with strategy (i), $\psi = 0.1$.

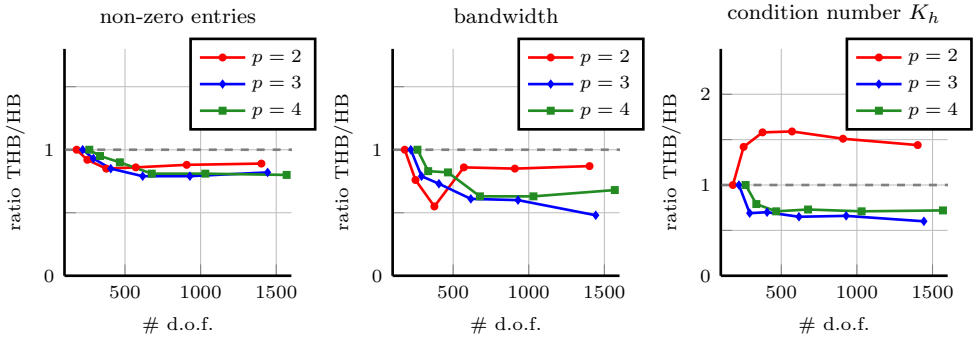
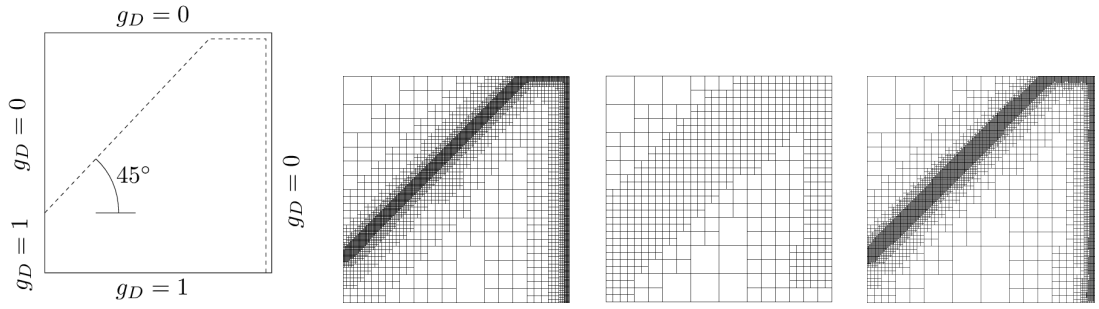


Figure 22: Comparison of number of non-zero entries, bandwidths, and condition numbers of the stiffness matrix K_h in Example 7 (L-shaped domain). Adaptive refinement with strategy (ii), $\psi = 0.9$.



(a) boundary conditions and expected positions of layers

(b) mesh after 6 refinements with strategy (i), $\psi = 0.1$

(c) mesh after 3 refinements with strategy (ii), $\psi = 0.7$

(d) mesh after 6 refinements with strategy (ii), $\psi = 0.7$

Figure 23: Problem setting and adaptively refined meshes for $p = 2$ in Example 8 (advection-diffusion on unit square).

hand-side

$$f(x, y) = \begin{cases} 1, & \text{if } |(x, y) - (186, 500)| < 30, \\ 0, & \text{else.} \end{cases}$$

In this setting, we have a source which is located approximately at the position of Lake Maxinkuckee,

step	#d.o.f.	sparsity: non-zero entries HB	THB	$\frac{\text{THB}}{\text{HB}}$	bandwidth HB	THB	$\frac{\text{THB}}{\text{HB}}$
		$p = 2$					
0	36	196	196	1.00	11	11	1.00
1	86	1208	1030	0.85	35	22	0.63
2	247	5011	4417	0.88	72	49	0.68
3	650	17170	14048	0.82	176	98	0.56
4	1590	50849	38377	0.75	700	262	0.37
5	3804	134537	95385	0.71	1874	448	0.24
6	9114	333007	231669	0.70	4180	837	0.20
		$p = 3$					
0	49	529	529	1.00	19	19	1.00
1	106	2515	2329	0.93	51	36	0.71
2	253	9844	8342	0.85	124	70	0.57
3	601	33473	25645	0.77	256	152	0.59
4	1411	102686	69954	0.68	659	348	0.53
5	3301	292312	176446	0.60	1641	725	0.44
6	7636	770659	411681	0.53	4134	1113	0.27
		$p = 4$					
0	64	1156	1156	1.00	29	29	1.00
1	144	4900	4900	1.00	64	64	1.00
2	287	16671	15077	0.90	164	133	0.81
3	623	56391	45487	0.81	383	215	0.56
4	1424	182259	129971	0.71	1052	544	0.52
5	3317	552138	346608	0.63	2717	1403	0.52
6	7623	1523631	832399	0.55	6599	2737	0.41

Table 4: Sparsity properties and condition numbers in Example 8 (advection-diffusion on unit square). Adaptive refinement with strategy (ii), $\psi = 0.7$.

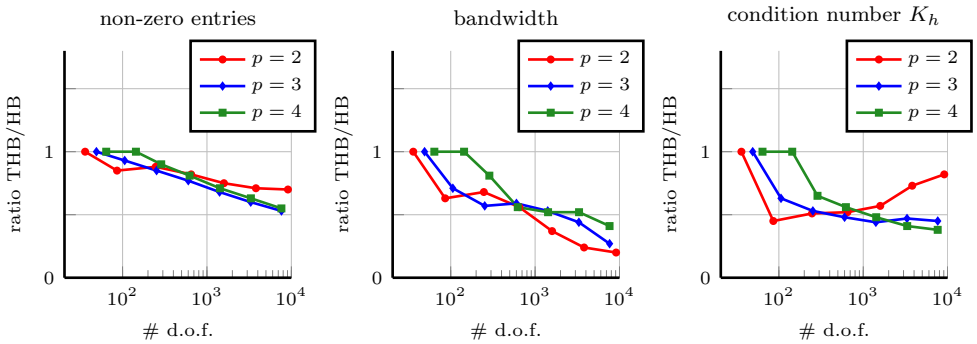


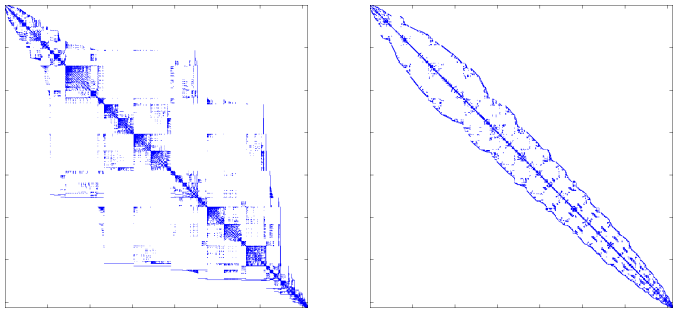
Figure 24: Comparison of number of non-zero entries, bandwidths, and condition numbers of the stiffness matrix K_h in Example 8 (advection-diffusion on unit square). Adaptive refinement with strategy (ii), $\psi = 0.7$.

and the advection is pointing from there towards the position where Wabash River crosses from Indiana to Ohio. The computed solution on the inital mesh is Fig. 27(c), the solution after 5 adaptive refinement steps in Fig. 28(c).

Fig. 28(a) and 28(b) show the mesh after five steps of adaptive refinement with strategy (i), $\psi = 0.1$ on the parameter domain and the physical domain, respectively. The area around Lake Maxinkuckee and the area where the advection meets the boundary are refined, as are the areas near the more complicated parts of the domain boundary. This example illustrates the combination of geometry-driven, and PDE-solver-driven adaptive local refinement using (T)HB-splines. The sparsity patterns of the system matrix at the final refinement level are visualized in Fig. 29. The behaviour with respect to sparsity and condition numbers is similar to previous examples and is thus omitted here.

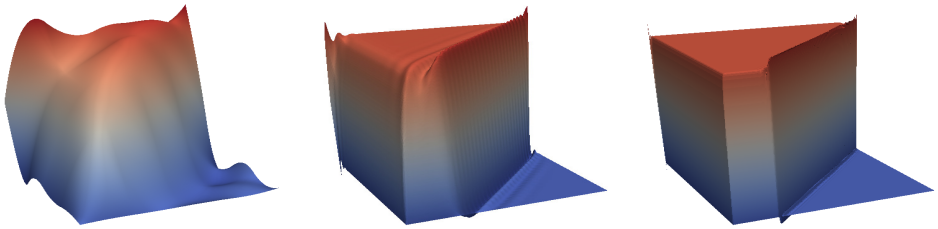
Example 10 (Poisson equation on G-shaped volume)**.** We consider the Poisson's equation

$$-\Delta u = f$$



(a) HB-splines (b) THB-splines

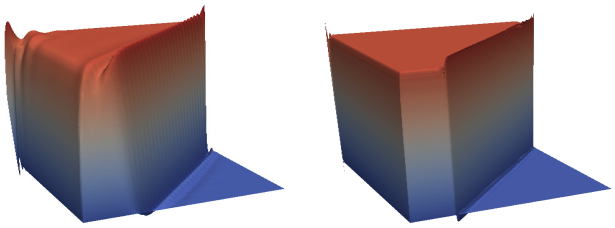
Figure 25: Sparsity pattern of system matrix after Cuthill-McKee-reordering at step 6, refinement strategy (ii), $\psi = 0.7$ in Example 8, $p = 2$ (advection-diffusion on unit square).



(a) solution on initial mesh

(b) tensor-B-spline solution after 3 refinements

(c) tensor-B-spline solution after 6 refinements



(d) THB-spline solution after 3 refinements

(e) THB-spline solution after 6 refinements

Figure 26: Initial (top left), intermediate (center), and final (right) scalar solution in Example 8, $p = 2$ (advection-diffusion on unit square), computed with tensor-B-splines (top) and THB-splines (bottom). Refinement strategy (ii), $\psi = 0.7$.

on the three-dimensional domain representing the character "G", as shown in Fig. 30.

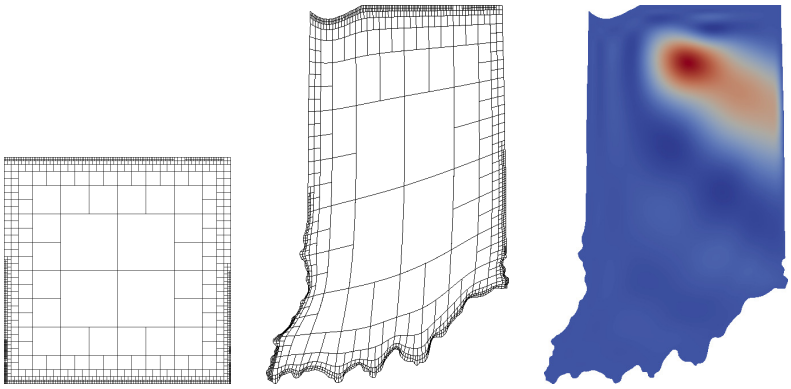
The source term f and the Dirichlet boundary conditions are determined by the exact solution

$$u(x, y, z) = \tanh(1 - 100(x + 2y + 4z - 4)).$$

Fig. 30 shows the parametric domain and the hierarchical mesh at level 6. The initial coarse mesh, which describes the geometry, has five interior knots along the sweep (depth) direction but no interior knots in the thickness direction, and 13 interior knots along the side that forms the shape of the G. This explains the rectangular shape of the parametric elements of the coarse elements (in red), which is also inherited by the elements of all higher levels, due to the dyadic refinement.

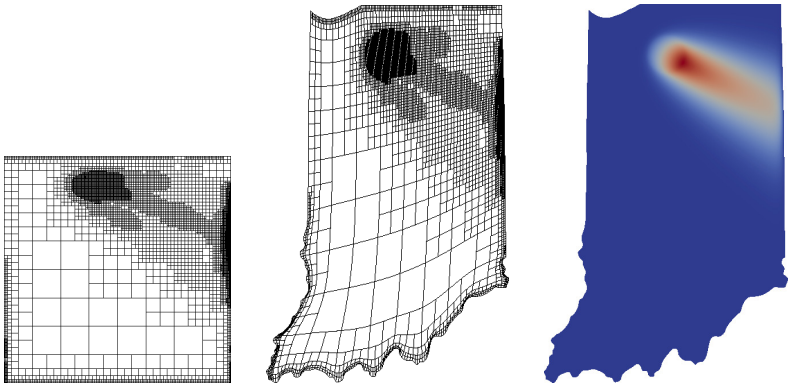
The corresponding numbers of non-zero entries and condition numbers are reported in Table 5.

Example 11 (Linear elasticity on support structure)**.** In our last example, we consider a linear elasticity problem on the domain depicted in Fig. 31, representing an aluminum support structure.



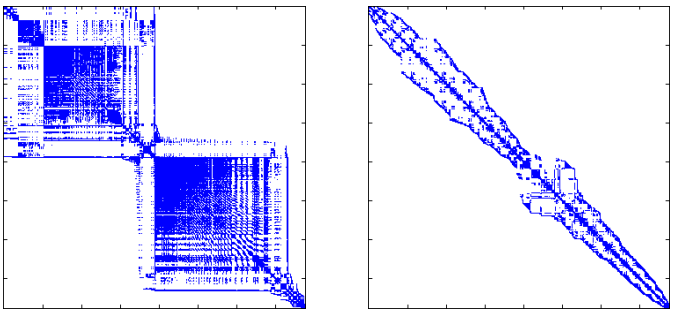
(a) parameter domain (b) physical domain (c) initial solution

Figure 27: Parameterization of Indiana (middle) in Example 9 (advection-diffusion on Indiana) with respect to the hierarchical knot configuration shown on the left.



(a) parameter domain (b) physical domain (c) solution at step 5

Figure 28: Meshes and computed solution after 5 refinement steps with strategy (i), $\psi = 0.1$ in Example 9 (advection-diffusion on Indiana).



(a) HB-splines (b) THB-splines

Figure 29: Sparsity pattern of system matrix after Cuthill-McKee-reordering at step 5, Example 9 (advection-diffusion on Indiana). Adaptive refinement with strategy (i), $\psi = 0.1$.

The domain is represented by a total of 10 subdomains which are fully matching in the sense that

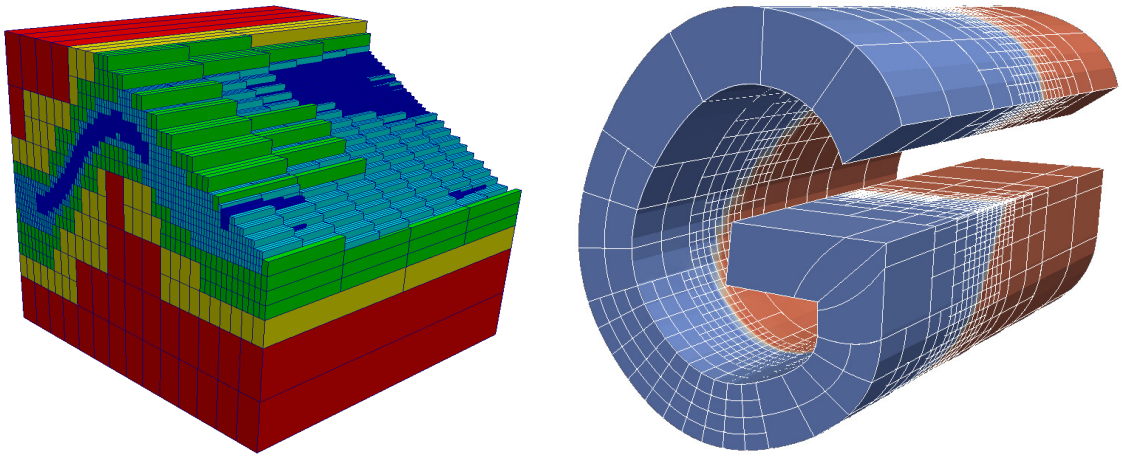


Figure 30: Example 10 (G-shaped volume). Left: parametric domain (sliced on the top), with elements colored according to the refinenent level. Right: geometry and mesh after four refinement steps with strategy (ii), with $\psi = 0.95$. The color shows the value of the solution.

		sparsity						condition numbers		
		non-zero entries			bandwidth			stiffness matrix		
L	#d.o.f.	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$
$p = 2$										
0	384	1536	1536	1.00	22	22	1.00	2.5e+01	2.5e+01	1.00
1	674	7482	6786	0.91	64	60	0.94	6.0e+01	3.4e+01	0.56
2	1553	56619	46653	0.82	297	177	0.60	1.9e+02	1.3e+02	0.69
3	3852	300007	231529	0.77	988	752	0.76	4.8e+02	2.1e+02	0.43
4	9205	1055336	753760	0.71	2771	1493	0.54	9.6e+02	3.3e+02	0.34

Table 5: Sparsity properties and condition numbers in Example 10 (G-shaped volume). Note that the stiffness matrix is also the system matrix in this example.

they are geometrically conforming and that the corresponding meshes match. Its bounding box is given by $[0, 0.1] \times [0, 0.05] \times [0, 0.1]$.

The structure is fixed to a wall and subject to a constant downward-pointing force at the top (see Fig 31(a)). The total applied force is 60N, the material parameters are given by $E = 6.9 \cdot 10^{10}$Pa (Young's modulus), $\nu = 0.334$ (Poisson's ratio), and $\rho = 2700\text{kg/m}^3$ (density). Due to the shape of the structure and the applied boundary conditions, stress peaks appear in several areas of the domain. The mesh after three refinement steps is shown in Fig. 32(c). In Fig. 33, the convergence of the energy norm of the computed solution is plotted and compared with the results obtained with uniform refinement. The faster convergence on a degrees-of-freedom-basis is clearly visible. The non-zero entries and condition numbers for this example are reported in Table 6.

		sparsity						condition numbers		
		non-zero entries			bandwidth			system matrix		
L	#d.o.f.	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$	HB	THB	$\frac{\text{THB}}{\text{HB}}$
$p = 2$										
0	1728	153797	153797	1.00	368	368	1.00	25292	25292	1.00
1	4164	615174	584119	0.95	820	716	0.87	112725	103177	0.92
2	6564	1272241	1117400	0.88	1941	1265	0.65	134270	94591	0.70
3	10728	2772293	2208819	0.80	3269	2975	0.91	142146	104041	0.73

Table 6: Sparsity properties and condition numbers in Example 11 (linear elasticity on support structure).

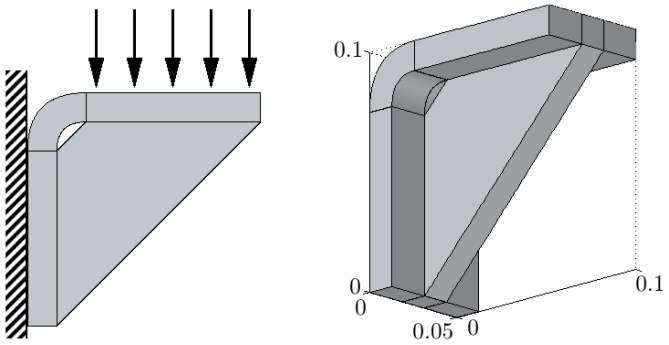


Figure 31: Problem setting and geometry in Example 11 (linear elasticity on support structure). The structure is fixed at $x = 0$ and subject to a uniform downward-pointing force at the top.

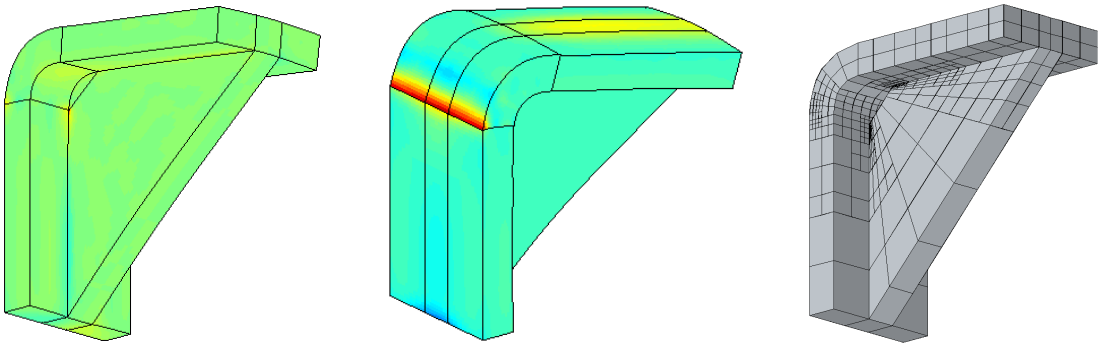


Figure 32: Distribution of computed stress component σ_{22} (left and middle) in Example 11 (linear elasticity on support structure) — the displacement is scaled up for visualization by a factor of 10^4. The hierarchical mesh after three adaptive refinement steps is also shown (right).

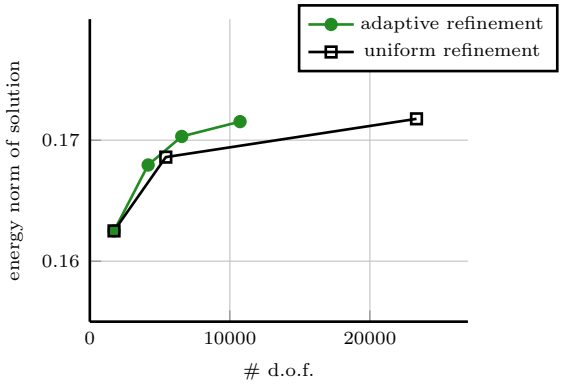


Figure 33: Convergence of the energy norm of the computed solution in Example 11 (linear elasticity on support structure).

5. Conclusion

The paper was devoted to THB-splines, which are a useful generalization of tensor-product splines that provides the possibility for adaptive refinement. THB-splines possess a firm theoretical basis, including results on linear independence, partition of unity, approximation power, completeness and stability [18, 19, 32, 42], and they can be exported as standard CAD geometries/models, thereby ensuring the compatibility with the existing technology [28]. They have been efficiently implemented in

the frame of the G+SMO Open Source C++ library [26]. As described in the paper, the implementation benefited substantially from the regular structure of the underlying tensor-product spline spaces.

Based on these theoretical and practical results, we explored the suitability of THB-splines for applications in geometric and numerical simulation. It was shown that these representations are well suited to perform adaptive refinement in related application scenarios.

The advantageous properties of THB-splines can be also fully exploited to derive suitable extensions to multi-patch domains [6] and manifold structures [46, 49], and to perform the convergence analysis of adaptive isogeometric methods [7]. The critical role of suitable error estimators, together with the influence of the marking and refinement strategies, require deep investigations in order to design reliable adaptive schemes for demanding application settings. Obviously, in order to overcome the difficulties related to the high condition numbers of isogeometric mass and stiffness matrices, efficient preconditioners are needed, see e.g., [8]. THB-splines have been recently exploited also to establish isogeometric multigrid methods [21]. The locality of THB-spline refinement, suitably combined with the mathematical foundations of the basis construction, provide a promising concept for the future development of adaptive methods for geometric modeling and isogeometric analysis.

Acknowledgements

The authors would like to thank Dr. Anh-Vu Vuong for collaborating with us at an earlier stage of this research. The support by the European Commission (project EXAMPLE, GA No. 324340), by the Austrian Science Fund (NFN S117 "Geometry + Simulation") and by MIUR (project DREAMS, programme "Futuro in Ricerca" RBFR13FBI3) is gratefully acknowledged.

References

[1] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using T-Splines. *Comput. Methods Appl. Mech. Engrg.*, 199:229–263, 2010.

[2] L. Beirão da Veiga, A. Buffa, D. Cho, and G. Sangalli. Analysis-Suitable T-splines are Dual-Compatible. *Comput. Methods Appl. Mech. Engrg.*, 249–252:42–51, 2012.

[3] L. Beirão da Veiga, A. Buffa, G. Sangalli, and R. Vázquez. Analysis-suitable T-splines of arbitrary degree: definition, linear independence and approximation properties. *Math. Models Methods Appl. Sci.*, 23:1979–2003, 2013.

[4] A. Bressan. Some properties of LR-splines. *Comput. Aided Geom. Design*, 30:778–794, 2013.

[5] A. Bressan and B. Jüttler. A Hierarchical Construction of LR Meshes in 2D. *G+S Report No. 23, Johannes Kepler University Linz*, 2015.

[6] A. Buchegger, B. Jüttler, and A. Mantzaflaris. Adaptively refined multi-patch B-splines with enhanced smothness. *G+S Report No. 26, Johannes Kepler University Linz*, 2015.

[7] A. Buffa and C. Giannelli. Adaptive isogeometric methods with hierarchical splines: error estimator and convergence. *Math. Models Methods Appl. Sci.*, to appear, 2015. `http://dx.doi.org/10.1142/S0218202516500019`.

[8] A. Buffa, H. Harbrecht, and G. Kunoth, A. Sangalli. BPX-preconditioning for isogeometric analysis. *Comput. Methods Appl. Mech. Engrg.*, 265:63–70, 2013.

[9] J. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, Chichester, 2009.

[10] J.A. Cottrell, T.J.R. Hughes, and A. Reali. Studies of refinement and continuity in isogeometric structural analysis. *Comput. Methods Appl. Mech. Engrg.*, 196:4160–4183, 2007.

[11] J. Deng, F. Chen, X. Li, Ch. Hu, W. Tong, Z. Yang, and Y. Feng. Polynomial splines over hierarchical T-meshes. *Graph. Models*, 70:76–86, 2008.

[12] T. Dokken, T. Lyche, and K. F. Pettersen. Polynomial splines over locally refined box–partitions. *Comput. Aided Geom. Design*, 30:331–356, 2013.

[13] M.R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Comput. Methods Appl. Mech. Engrg.*, 199(5-8):264–275, 2010.

[14] 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. *Comput. Methods Appl. Mech. Engrg.*, 284:1–20, 2015.

[15] A. Falini, J. Špeh, and B. Jüttler. Planar domain parameterization with THB-splines. *Comput. Aided Geom. Design*, 35–36:95–108, 2015.

[16] D.R. Forsey and R.H. Bartels. Hierarchical B-spline refinement. *Comput. Graphics*, 22:205–212, 1988.

[17] C. Giannelli and B. Jüttler. Bases and dimensions of bivariate hierarchical tensor–product splines. *J. Comput. Appl. Math.*, 239:162–178, 2013.

[18] C. Giannelli, B. Jüttler, and H. Speleers. THB-splines: The truncated basis for hierarchical splines. *Comput. Aided Geom. Design*, 29(7):485–498, 2012.

[19] C. Giannelli, B. Jüttler, and H. Speleers. Strongly stable bases for adaptively refined multilevel spline spaces. *Adv. Comp. Math.*, 40(2):459–490, 2014.
[20] G. Greiner and K. Hormann. Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines. In A. Le Méhauté, C. Rabut, and L. L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 163–172. Vanderbilt University Press, Nashville, TN, 1997.
[21] C. Hofreither, B. Jüttler, G. Kiss, and W. Zulehner. Multigrid methods for isogeometric analysis with THB-splines. In preparation, 2015.
[22] T.J.R. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194(39-41):4135–4195, 2005.
[23] K.A. Johannessen, T. Kvamsdal, and T. Dokken. Isogeometric analysis using LR B-splines. *Comput. Methods Appl. Mech. Engrg.*, 269:471–514, 2014.
[24] K.A. Johannessen, F. Remonato, and T. Kvamsdal. On the similarities and differences between Classical Hierarchical, Truncated Hierarchical and LR B-splines. *Comput. Methods Appl. Mech. Engrg.*, 291:64–101, 2015.
[25] V. John. A numerical study of a posteriori error estimators for convection–diffusion equations. *Comput. Methods Appl. Mech. Engrg.*, 190(5-7):757–781, 2000.
[26] B. Jüttler, U. Langer, A. Mantzaflaris, S.E. Moore, and W. Zulehner. Geometry + Simulation Modules: Implementing Isogeometric Analysis. *PAMM*, 14:961–962, 2014.
[27] G. Kiss, C. Giannelli, and B. Jüttler. Algorithms and data structures for truncated hierarchical B–splines. In M. Floater et al., editors, *Mathematical Methods for Curves and Surfaces*, volume 8177, pages 304–323. Lecture Notes in Computer Science, 2014.
[28] G. Kiss, C. Giannelli, U. Zore, B. Jüttler, D. Großmann, and J. Barner. Adaptive CAD model (re–)construction with THB–splines. *Graph. Models*, 76:273–288, 2014.
[29] R. Kraft. Adaptive and linearly independent multilevel B–splines. In A. Le Méhauté, C. Rabut, and L.L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 209–218. Vanderbilt University Press, Nashville, 1997.
[30] X. Li, J. Deng, and F. Chen. Surface modeling with polynomial splines over hierarchical T-meshes. *Visual Comput.*, 23:1027–1033, 2007.
[31] X. Li, J. Zheng, T.W. Sederberg, T.J.R. Hughes, and M.A. Scott. On linear independence of T–spline blending functions. *Comput. Aided Geom. Design*, 29:63–76, 2012.
[32] D. Mokriš, B. Jüttler, and C. Giannelli. On the completeness of hierarchical tensor-product B-splines. *J. Comput. Appl. Math.*, 271:53–70, 2014.
[33] P. Morgenstern and D. Peterseim. Analysis-suitable adaptive T-mesh refinement with linear complexity. *Comput. Aided Geom. Design*, 34:50–66, 2015.
[34] N. Nguyen-Thanh, H. Nguyen-Xuan, S.P.A. Bordas, and T. Rabczuk. Isogeometric analysis using polynomial splines over hierarchical T-meshes for two-dimensional elastic solids. *Comput. Methods Appl. Mech. Engrg.*, 200:1892–1908, 2011.
[35] L. Piegl and W. Tiller. *The NURBS book*. Springer Berlin Heidelberg, 2 edition, 1997.
[36] O. Pironneau. *Finite element methods for fluids*. Wiley, 1989.
[37] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.
[38] D. Schillinger, L. Dedé, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, and 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. *Comput. Methods Appl. Mech. Engrg.*, 249–252:116 – 150, 2012.
[39] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM Trans. Graphics*, 23:276 – 283, 2004.
[40] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCS. *ACM Trans. Graphics*, 22:477–484, 2003.
[41] V. Skytt, O. Barrowclough, and T. Dokken. Locally refined spline surfaces for representation of terrain data. *Comput. Graphics*, 49:58–68, 2015.
[42] H. Speleers and C. Manni. Effortless quasi-interpolation in hierarchical spaces. *Numer. Math.*, to appear, 2015. `http://dx.doi.org/10.1007/s00211-015-0711-z`.
[43] 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*, 200(49-52):3554–3567, 2011.
[44] P. Wang, J. Xu, J. Deng, and F. Chen. Adaptive isogeometric analysis using rational PHT-splines. *Computer-Aided Design*, 43(11):1438–1448, 2011.
[45] Y. Wang, J. Zheng, and H. S. Seah. Conversion between T-splines and hierarchical B-splines. In M. H. Hamza, editor, *Computer Graphics and Imaging*, pages 8–13. IASTED/ACTA Press, 2005.
[46] X. Wei, Y. Zhang, T.J.R. Hughes, and M.A. Scott. Truncated hierarchical Catmull–Clark subdivision with local refinement. *Comput. Methods Appl. Mech. Engrg.*, 291:1–20, 2015.
[47] M. Wu, J. Xu, R. Wang, and Z. Yang. Hierarchical bases of spline spaces with highest order smoothness over hierarchical T–subdivisions. *Comput. Aided Geom. Design*, 29:499–509, 2012.
[48] U. Zore and B. Jüttler. Adaptively refined multilevel spline spaces from generating systems. *Comput. Aided Geom. Design*, 31:545–566, 2014.
[49] U. Zore, B. Jüttler, and J. Kosinka. On the linear independence of (truncated) hierarchical subdivision splines. *G+S Report No. 17, Johannes Kepler University Linz*, 2014.