Efficient Matrix Assembly in Isogeometric Analysis with Hierarchical B-splines

Maodong Pan, Bert Jüttler, Angelos Mantzaflaris

Efficient Matrix Assembly in Isogeometric Analysis with Hierarchical B-splines

Maodong Pan[a,*], Bert Jüttler[a,b], Angelos Mantzaflaris[c]

[a] *Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Linz, Austria*
[b] *Institute of Applied Geometry, Johannes Kepler University, Linz, Austria*
[c] *Inria Sophia Antipolis - Méditerranée, Université Côte d'Azur, France*

Abstract

Hierarchical B-splines that allow local refinement have become a promising tool for developing adaptive isogeometric methods. Unfortunately, similar to tensor-product B-splines, the computational cost required for assembling the system matrices in isogeometric analysis with hierarchical B-splines is also high, particularly if the spline degree is increased. To address this issue, we propose an efficient matrix assembly approach for bivariate hierarchical B-splines based on the previous work [42]. The new algorithm consists of three stages: approximating the integrals by quasi-interpolation, building three compact look-up tables and assembling the matrices via sum-factorization. A detailed analysis shows that the complexity of our method has the order $\mathcal{O}(Np^3)$ under a mild assumption about mesh admissibility, where N and p denote the number of degrees of freedom and spline degree respectively. Finally, several experimental results are demonstrated to verify the theoretical results and to show the performance of the proposed method.

Keywords: Matrix formation; Isogeometric analysis; Hierarchical B-splines; Quasi-interpolation, look-up and sum-factorization; Computational cost

1. Introduction

Introduced to improve the interoperability across the design and analysis pipeline, isogeometric analysis (IgA) [16, 34] has been recognized as a promising tool in several disciplines such as nonlinear solid mechanics [21], electromagnetic problems [12], shape optimization[43] and plate analysis [17]. The core idea of IgA is to employ the same basis functions (e.g., non-uniform rational B-splines (NURBS)) both for the representation of geometry in computer aided design (CAD) and for the approximation of solution fields in finite element method (FEM), allowing from the beginning the elimination of geometry errors [41]. Another important feature of IgA is that it allows for higher global regularity of the basis functions, up to

*Corresponding author.
Email addresses: mdpan@mail.ustc.edu.cn (Maodong Pan), bert.juettler@jku.at (Bert Jüttler), angelos.mantzaflaris@inria.fr (Angelos Mantzaflaris)

C^{p-1} across mesh elements for splines of degree p, which leads to an increased accuracy per degree of freedom [7, 18] and improved spectrum properties [22, 35]. We refer the interested reader to the reference [19] for a detailed mathematical analysis of the IgA approach.

However, the use of high-degree basis functions in IgA raises the issue of expensive computational effort required to assemble the system matrices arising in isogeometric Galerkin discretizations [20]. For a d-dimensional spline space with spline degree p and N degrees of freedom, standard assemblers based on element-wise Gauss quadrature require $\mathcal{O}(Np^{3d})$ operations to compute the system matrix, which is far from being optimal. For this reason, several important contributions have been made towards the development of efficient quadrature rules for tensor-product-based isogeometric discretizations. The first line of research addressing this problem aims at developing reduced or specialized quadrature rules [2, 4–6, 15, 31, 32, 36, 46] that are more efficient than the standard Gauss quadrature with $\mathcal{O}(p)$ nodes per knot span. The second category of quadrature rules are the isogeometric collocation methods [3, 30, 45], which can be interpreted as one-point quadrature schemes in the IgA context. Another approach aims at exploiting the tensor-product structure, which includes the low-rank tensor approximation [33, 40, 47] and sum-factorization [1, 8, 15, 42] techniques. Finally, let us mention two quadrature-free approaches [39, 42] that are the starting points of this paper. In the interpolation and look-up approach [39], the common factor occurring in the integrand is firstly transformed into tensor-product splines, and subsequently the resulting integrals are evaluated exactly using the precomputed look-up tables. Based on this method, the recent work [42] further reduced the computational cost of matrix assembly by exploiting the tensor-product structure via the sum-factorization technique. Instead of repeating the description of these prior works, we refer to the papers [14, 42], whose introductory sections contain an overview of the state of the art.

In the framework of IgA, hierarchical B-splines have attracted increasing attention [10, 26, 27, 50] since they possess local refinement abilities, which is impossible for tensor-product splines. Unfortunately, similar to the tensor-product case, isogeometric simulations based on hierarchical B-splines face a great computational burden at the point of matrix assembly, especially for high polynomial degrees. However, to the best of our knowledge, so far few works on this topic have been proposed [28].

In this paper, we extend the previous work [42] to bivariate hierarchical B-splines. Although both methods consist of three stages: spline projection, building look-up tables and assembling the matrices via sum-factorization, performing each step of the new proposed method is more technical and challenging. A detailed complexity analysis demonstrates that the computational cost for matrix assembly has order $\mathcal{O}(Np^3)$ under a mesh admissibility assumption. Several numerical tests in terms of the N-dependence and p-dependence behavior are provided, which confirm the theoretical result of the new algorithm. In addition, the convergence behavior and accuracy of our method are also verified by applying it to solving a Poisson problem with an adaptive refinement strategy. It should be noted that the extension of the new algorithm to trivariate hierarchical B-splines is not a trivial task, since the last stage – matrix assembly via sum factorization presented in current paper – is not suitable for the trivariate case. This will be discussed in another paper.

The remainder of this paper is structured as follows. Section 2 recalls the basic concepts

of hierarchical B-splines and admissible meshes followed by introducing the isogeometric discretizations of an elliptic problem. Section 3–6 describe the outline and three stages of the proposed method. A detailed theoretical analysis of the computational cost of the new algorithm is given in Section 7. Section 8 presents several numerical examples to demonstrate the performance and to verify the theoretical results of our method. Finally, we conclude the paper with a summary and future work in Section 9.

2. Preliminaries

This section includes three parts. The first part recalls the basic concepts and defines the global index space of hierarchical B-splines (HB-splines). Then the definition and some important properties of the admissible HB-splines are presented. Finally, we use the introduced hierarchical spline space to derive the isogeometric Galerkin discretizations of an elliptic boundary value problem.

2.1. Hierarchical B-splines

We consider two sequences (one per coordinate direction, i.e., $d = 1$ or $d = 2$) of spaces $\mathcal{B}_d^\ell$ composed of univariate splines of degree p on $[0, 1]$. The spaces are indexed by the level $\ell = 0, \ldots, L$.

For each level ℓ, the two spaces $\mathcal{B}_1^\ell$ and $\mathcal{B}_2^\ell$ of level ℓ are both defined by 2^ℓ knot spans. Therefore, they are spanned by bases

$$B_d^\ell = \{\beta_{d,i_d}(\xi_d) \ : \ i_d \in \mathbb{I}^\ell\} \tag{1}$$

that consist of $(p + 2^\ell)$ B-splines $\beta_{d,i_d}(\xi_d)$.

In order to simplify the notation, we shall employ the *global index space* $\mathbb{Z}$, where B-splines of level ℓ have indices i_d in the subspace

$$\mathbb{I}^\ell = \{2^\ell + \ell p, \ldots, 2^{\ell+1} + (\ell + 1)p - 1\}. \tag{2}$$

Note that the index subspaces associated with different levels are mutually disjoint. We denote with $\text{level}(i_d)$ the level of an index i_d, i.e., $i_d \in \mathbb{I}^{\text{level}(i_d)}$.

Additionally, we assume that the knot spans of any level $\ell + 1$ are created by performing non-uniform dyadic refinement to the ones of the previous level. More precisely, the knot spans of level ℓ are split into two knot spans of level $\ell + 1$, where non-uniform splits are allowed. This assumption implies

- the nestedness of the spline spaces, $\mathcal{B}_d^\ell \subset \mathcal{B}_d^{\ell+1}$,
- the maximum smoothness C^{p-1}, and
- a level-wise bound on the number of B-splines that possess a support intersection with a given B-spline.

The latter fact is made precise in the following lemma.

Lemma 1. *For any given B-spline β_{d,i_d} of level ℓ, there are at most $\lceil 2^{\ell'-\ell}(p+1)\rceil + p + 1$ B-splines of level ℓ' that possess a support intersection[1] with it.*

[1]These B-splines β_{d,j_d} satisfy $\text{supp}(\beta_{d,i_d}\beta_{d,j_d}) \neq \emptyset$.

The tensor-product spline bases $B^\ell = B_1^\ell \otimes B_2^\ell$ consist of all products of univariate B-splines,

$$B^\ell = \{\beta_{i_1,i_2}(\xi_1,\xi_2) = \beta_{1,i_1}(\xi_1)\beta_{2,i_2}(\xi_2) \ : \ i_1 \in \mathbb{I}^\ell,\ i_2 \in \mathbb{I}^\ell\}. \tag{3}$$

They span nested tensor-product spline spaces

$$\mathcal{B}^\ell = \text{span } B^\ell, \quad \mathcal{B}^\ell \subset \mathcal{B}^{\ell+1}. \tag{4}$$

The arguments of the tensor-product splines are $\boldsymbol{\xi} = (\xi_1, \xi_2)$. After introducing multi-indices $\boldsymbol{i} = (i_1, i_2)$ which is an element of the global index space $\mathbb{Z}^2$, we can simply write

$$\beta_{\boldsymbol{i}}(\boldsymbol{\xi}) = \beta_{1,i_1}(\xi_1)\beta_{2,i_2}(\xi_2), \tag{5}$$

where B-splines of level ℓ have indices $\boldsymbol{i}$ in the subspace $\mathbb{I}^\ell \times \mathbb{I}^\ell$.

The two consequences of the assumption about dyadic refinement of the univariate spline spaces carry over to the bivariate setting, implying again

- the nestedness of the spline spaces, $\mathcal{B}^\ell \subset \mathcal{B}^{\ell+1}$, and
- a level-wise bound on the number of B-splines that possess a support intersection with a given B-spline, as described in the following lemma.

Lemma 2. *For any given B-spline $\beta_{\boldsymbol{i}}$ of level ℓ, there are at most $(\lceil 2^{\ell'-\ell}(p+1)\rceil + p + 1)^2$ B-splines of level ℓ' that possess a support intersection with it.*

In order to define hierarchical B-splines (HB-splines), we need to consider a sequence of nested subdomains

$$[0,1]^2 = \Omega^0 \supseteq \Omega^1 \supseteq \ldots \supseteq \Omega^L, \tag{6}$$

where we assume that the boundary of Ω^ℓ is aligned to the knot lines of $B^{\ell-1}$. The HB-splines

$$H = \{\beta_{\boldsymbol{i}} \ : \ \boldsymbol{i} \in \mathbb{H}\}, \tag{7}$$

which are defined by the index set

$$\mathbb{H} = \{\boldsymbol{i} \in \bigcup_{\ell=0}^{L} \mathbb{I}^\ell \times \mathbb{I}^\ell \ : \ \Omega^\ell \supseteq \text{supp } \beta_{\boldsymbol{i}} \not\subseteq \Omega^{\ell+1}\} \tag{8}$$

with $\Omega^{L+1} = \emptyset$, span the hierarchical spline space

$$\mathcal{H} = \text{span } H. \tag{9}$$

Under the definition (8), the indices of the HB-splines H possess the block structure shown in Figure. 1, since only univariate B-splines of the same level are considered in the individual tensor-product bases.

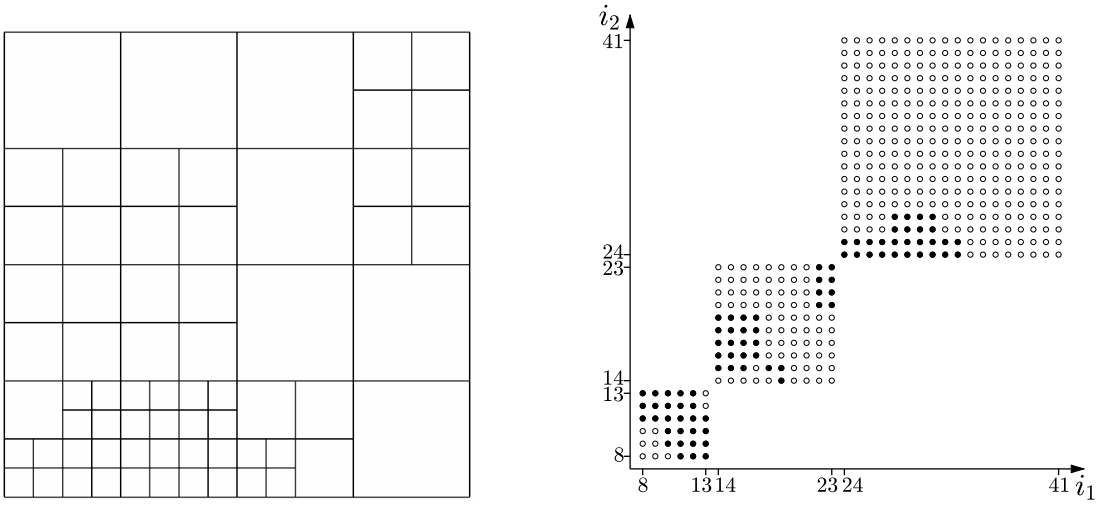


Figure 1: The global indices (coordinates of the solid dots on the right) of the HB-splines (left) of degree 2 with three levels. Here the coordinates of all the dots represent the subspaces $\mathbb{I}^\ell \times \mathbb{I}^\ell$ $(\ell = 2, 3, 4)$ of the global index system $\mathbb{Z}^2$, where $\mathbb{I}^2 = \{8, \ldots, 13\}$, $\mathbb{I}^3 = \{14, \ldots, 23\}$ and $\mathbb{I}^4 = \{24, \ldots, 41\}$.

2.2. Ensuring admissibility

We recall the notion of admissible mesh configurations from [10]: The HB-splines H, which are defined by a given subdomain hierarchy, are said to be admissible of class $r \geq 2$ if the following condition holds:

$$\operatorname{supp} \beta_{\boldsymbol{i}} \cap \Omega^{\ell+r} = \emptyset, \quad \forall \boldsymbol{i} \in (\mathbb{I}^\ell \times \mathbb{I}^\ell) \cap \mathbb{H}, \ \ell = 0, 1, \ldots, L - r. \tag{10}$$

Figure 2 illustrates this definition through a simple example of biquadratic hierarchical splines of class 2.

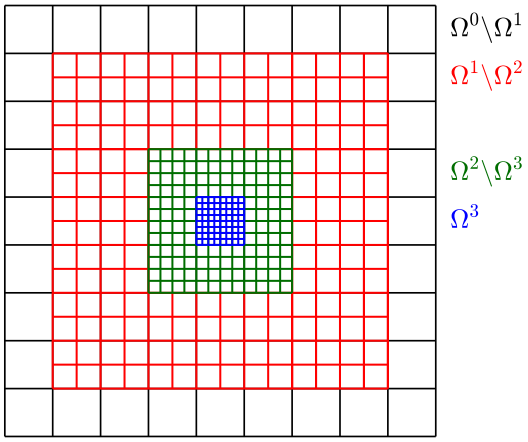


Figure 2: An admissible mesh of class 2 for biquadratic HB-splines with $N = 3$. The subdomains $\Omega^0 \setminus \Omega^1$, $\Omega^1 \setminus \Omega^2$, $\Omega^2 \setminus \Omega^3$ and Ω^3 are highlighted in black, red, green and blue respectively.

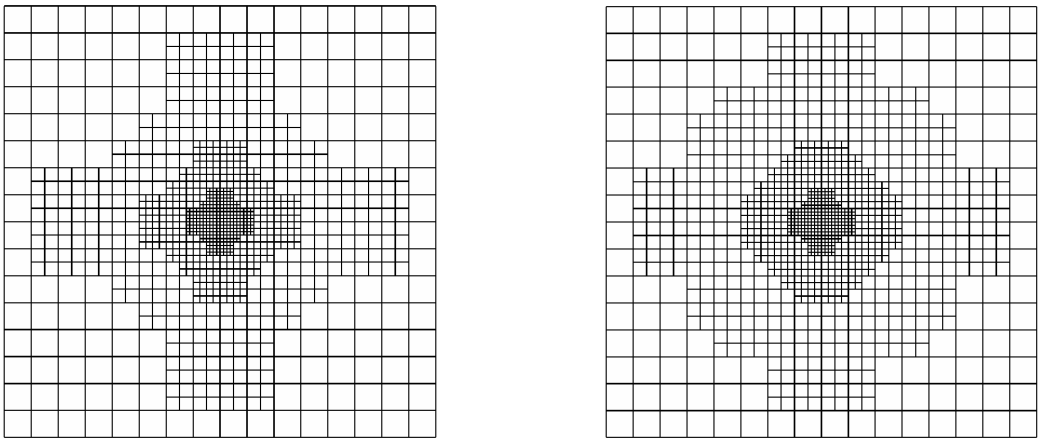
Admissible HB-splines possess three important properties:

- the basis functions in H that take non-zero values over any element of the hierarchical mesh belong to at most r successive levels;
- the number of basis functions acting on each element of the hierarchical mesh does not exceed $r(p+1)^2$, independently of the overall number of hierarchical levels;
- under the assumption of dyadic refinement strategy for creating the sequence of nested spaces, the number of basis functions in H that possess a support intersection with any given HB-spline $\beta_{\boldsymbol{i}}$ in H does not exceed

$$\sum_{k=1-r}^{r-1} \left(\lceil \frac{p+1}{2^k} \rceil + 1 + p \right)^2 = \mathcal{O}(p^2).$$

These properties play an important role in the analysis of adaptive isogeometric analysis [10, 11]. In particular, the last one is directly related to the sparsity properties of the isogeometric Galerkin matrices.

In the remainder of this paper we shall assume admissibility of class $r = 2$. Adaptive mesh refinement subject to this constraint can be performed using the algorithms presented in [10]. Given an input mesh, the admissibility can be restored by suitably enlarging the nested subdomains Ω^ℓ. An example is shown in Figure 3.



(a) Input: non-admissible mesh (b) Output: admissible mesh

Figure 3: An admissible mesh (right) of class 2 for biquadratic HB-splines obtained from a given input mesh (left).

2.3. Isogeometric Galerkin discretizations

To describe the proposed method for assembling the system matrices arising in isogeometric Galerkin discretizations of partial differential equations (PDEs), we consider a general second-order linear elliptic boundary value problem

$$\begin{cases} -\mathrm{div}(K\nabla u) + \boldsymbol{\lambda}^T \nabla u + \sigma u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{11}$$

on the bounded domain $\Omega \subset \mathbb{R}^2$. The matrix-valued function $K \in L^\infty(\Omega, \mathbb{R}^{2\times 2})$ is symmetric and uniformly positive definite, the scalar function $\sigma \in L^\infty(\Omega)$ is non-negative, the vector-valued function $\boldsymbol{\lambda}$ and the source function f satisfy $\boldsymbol{\lambda} \in L^\infty(\Omega, \mathbb{R}^2)$, $f \in L^2(\Omega)$.

Considering the Sobolev space $V = H_0^1(\Omega)$, we recall the weak formulation of the problem (11): Find $u \in V$ satisfying

$$a(u, v) = F(v) \text{ for all } v \in V, \tag{12}$$

where the bilinear and linear forms are

$$a(u, v) = \int_\Omega \nabla u^T(\boldsymbol{x}) K(\boldsymbol{x}) \nabla v(\boldsymbol{x}) + \boldsymbol{\lambda}^T(\boldsymbol{x}) \nabla u(\boldsymbol{x}) v(\boldsymbol{x}) + \sigma(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

and

$$F(v) = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

respectively.

In the framework of isogeometric analysis, the physical domain Ω is parameterized using spline functions. More precisely, we have a diffeomorphism $\boldsymbol{G}$ from an axis-aligned parametric domain $\hat{\Omega} := [0, 1]^2$ satisfying

$$\Omega = \boldsymbol{G}(\hat{\Omega}).$$

For the sake of simplicity, we assume that the parameterization $\boldsymbol{G}$ is given by single-patch hierarchical splines

$$\boldsymbol{G}(\boldsymbol{\xi}) = \sum_{\boldsymbol{i} \in \mathbb{H}} \boldsymbol{c_i} \, \beta_{\boldsymbol{i}}(\boldsymbol{\xi}), \tag{13}$$

which are represented as a linear combination of HB-splines $\beta_{\boldsymbol{i}}(\boldsymbol{\xi})$ with associated global indices $\boldsymbol{i}$ and coefficients $\boldsymbol{c_i} \in \mathbb{R}^2$. See [9, 24] and the references therein for the extension to the multi-patch case.

When using the isogeometric framework, the discretization of (12) utilizes the given parameterization of the domain Ω and is obtained by employing the same basis used for defining the parameterization (13). Specifically, the discretization space V_h is spanned by the isogeometric functions

$$\varphi_{\boldsymbol{i}} = \beta_{\boldsymbol{i}} \circ \boldsymbol{G}^{-1}, \quad \beta_{\boldsymbol{i}} \in H.$$

Applying the Galerkin approach to the variational problem (12), we arrive at the following discrete formulation: Find u_h in V_h satisfying

$$a(u_h, v_h) = F(v_h) \text{ for all } v_h \in V_h. \tag{14}$$

Solving the discrete variational problem (14) requires the formation of several matrices and vectors. More precisely, we need to assemble the stiffness matrix S with the elements

$$S_{\boldsymbol{i},\boldsymbol{i}'} = \int_\Omega \nabla {\varphi_{\boldsymbol{i}}}^T K \nabla \varphi_{\boldsymbol{i}'} \mathrm{d}\boldsymbol{x} = \int_{\hat{\Omega}} \nabla_{\boldsymbol{\xi}} {\beta_{\boldsymbol{i}}}^T \nabla_{\boldsymbol{\xi}} \boldsymbol{G}^{-1} \hat{K} \nabla_{\boldsymbol{\xi}} \boldsymbol{G}^{-T} \nabla_{\boldsymbol{\xi}} \beta_{\boldsymbol{i}'} | \det \nabla_{\boldsymbol{\xi}} \boldsymbol{G}| \mathrm{d}\boldsymbol{\xi}, \tag{15}$$

the advection matrix A with the elements

$$A_{\boldsymbol{i},\boldsymbol{i}'} = \int_{\Omega} \boldsymbol{\lambda}^T \nabla \varphi_{\boldsymbol{i}} \varphi_{\boldsymbol{i}'} \mathrm{d}\boldsymbol{x} = \int_{\hat{\Omega}} \hat{\boldsymbol{\lambda}}^T \nabla_{\boldsymbol{\xi}} \boldsymbol{G}^{-T} \nabla_{\boldsymbol{\xi}} \beta_{\boldsymbol{i}} \beta_{\boldsymbol{i}'} |\det \nabla_{\boldsymbol{\xi}} \boldsymbol{G}| \mathrm{d}\boldsymbol{\xi}, \tag{16}$$

the mass matrix M with the elements

$$M_{\boldsymbol{i},\boldsymbol{i}'} = \int_{\Omega} \sigma \varphi_{\boldsymbol{i}} \varphi_{\boldsymbol{i}'} \mathrm{d}\boldsymbol{x} = \int_{\hat{\Omega}} \hat{\sigma} \beta_{\boldsymbol{i}} \beta_{\boldsymbol{i}'} |\det \nabla_{\boldsymbol{\xi}} \boldsymbol{G}| \mathrm{d}\boldsymbol{\xi}, \tag{17}$$

and the load vector $\boldsymbol{b}$ with the elements

$$b_{\boldsymbol{i}} = \int_{\Omega} f \varphi_{\boldsymbol{i}} \mathrm{d}\boldsymbol{x} = \int_{\hat{\Omega}} \hat{f} \beta_{\boldsymbol{i}} |\det \nabla_{\boldsymbol{\xi}} \boldsymbol{G}| \mathrm{d}\boldsymbol{\xi}. \tag{18}$$

Note that the hat symbol $\hat{}$ denotes the pull-back of a function from the physical domain Ω to the parametric domain $\hat{\Omega}$, i.e.,

$$\hat{K} = K \circ \boldsymbol{G}, \quad \hat{\boldsymbol{\lambda}} = \boldsymbol{\lambda} \circ \boldsymbol{G}, \quad \hat{\sigma} = \sigma \circ \boldsymbol{G}, \quad \hat{f} = f \circ \boldsymbol{G}.$$

In order to distinguish it from the gradient on the physical domain, we use the symbol $\nabla_{\boldsymbol{\xi}}$ to denote the gradient in the parametric domain $\hat{\Omega}$.

3. Framework overview

In order to keep the presentation concise, we focus on the evaluation of the elements of the stiffness matrix, which can be rewritten as

$$S_{\boldsymbol{i},\boldsymbol{i}'} = \int_{\hat{\Omega}} \nabla_{\boldsymbol{\xi}} \beta_{\boldsymbol{i}}{}^T W \nabla_{\boldsymbol{\xi}} \beta_{\boldsymbol{i}'} \mathrm{d}\boldsymbol{\xi}, \quad W = |\det \nabla_{\boldsymbol{\xi}} \boldsymbol{G}| \, \nabla_{\boldsymbol{\xi}} \boldsymbol{G}^{-1} \hat{K} \nabla_{\boldsymbol{\xi}} \boldsymbol{G}^{-T}. \tag{19}$$

The approach can be easily adapted to other relevant matrices in (16), (17) and (18).

The elements of the kernel W in (19) are generally rational functions possessing high degrees. To make the calculation more efficient, we propose a quadrature-free method via spline projection, look-up and sum-factorization techniques. The overall framework of our approach is outlined as follows:

• Step 1. *Approximating the integrals via spline projection.* We first approximate the elements of W by hierarchical spline functions in $\mathcal{H}$,

$$W(\boldsymbol{\xi}) \approx \sum_{\boldsymbol{i}'' \in \mathbb{H}} W_{\boldsymbol{i}''} \beta_{\boldsymbol{i}''}(\boldsymbol{\xi}), \tag{20}$$

where $W_{\boldsymbol{i}''} = (w_{\boldsymbol{i}''}^{\mu,\nu})_{\mu,\nu=1,2}$ are the associated coefficient matrices. This transforms the stiffness matrix elements (19) into sums of integrals of tensor-product B-splines

$$S_{\boldsymbol{i},\boldsymbol{i}'} \approx \sum_{\mu=1}^{2} \sum_{\nu=1}^{2} \sum_{\boldsymbol{i}'' \in \mathbb{H}} w_{\boldsymbol{i}''}^{\mu,\nu} \int_{\hat{\Omega}} \partial_{\mu} \beta_{\boldsymbol{i}} \, \partial_{\nu} \beta_{\boldsymbol{i}'} \, \beta_{\boldsymbol{i}''} \mathrm{d}\boldsymbol{\xi}, \tag{21}$$

where the partial derivatives are defined as $\partial_d \beta = \partial \beta / \partial \xi_d$. This is the only approximation step in the entire method.

• Step 2. *Building compact look-up tables.* In order to perform an exact evaluation of the sums (21), we define the following tri-product integrals

$$T^{\theta,\vartheta}_{i_d,i'_d,i''_d} = \int_0^1 \beta^{\theta}_{d,i_d} \beta^{\vartheta}_{d,i'_d} \beta_{d,i''_d} \mathrm{d}\xi_d, \tag{22}$$

where β_{d,i_d}, β_{d,i'_d} and β_{d,i''_d} are univariate B-splines of degree p (maybe from different levels), and $\theta, \vartheta \in \{0, 1\}$ represent the order of derivatives. We collect all these values in a look-up table. Consequently, the quantities (21) can be rewritten as

$$S_{\boldsymbol{i},\boldsymbol{i}'} \approx \sum_{\mu=1}^{2} \sum_{\nu=1}^{2} \sum_{\boldsymbol{i}'' \in \mathbb{H}} w^{\mu,\nu}_{\boldsymbol{i}''} \prod_{d=1}^{2} T^{\delta(\mu,d),\delta(\nu,d)}_{i_d,i'_d,i''_d}, \tag{23}$$

where $\delta(\cdot,\cdot)$ is the Kronecker delta function. The second step of our approach is to generate three compact look-up tables for the exact values of the resulting tri-product integrals (22).

• Step 3: *Assembling the matrices.* In order to obtain a computationally efficient method, we employ sum-factorization to assemble the matrix, based on the previously established look-up tables.

The three steps will be described in more detail in the following three sections.

4. Approximating the integrals via spline projection

The first step of our approach is to project the matrix-valued function W that appears in the integrals (19) into a suitable hierarchical spline space. This can be accomplished by the global fitting or quasi-interpolation methods. We assume that the spline space used for the projector (20) is the same as the one for isogeometric discretizations described in Section 2.3. In this case, the size of the required look-up tables in the subsequent steps is relatively small and independent of the mesh size of the HB-splines. Also in the tensor-product case, it was noted that this choice gives optimal performance [39].

4.1. Global fitting

The most straightforward method for approximating a function g defined on $\hat{\Omega}$ with a hierarchical spline s in $\mathcal{H}$ is based on global L^2 fitting. This is done by solving the minimization problem

$$\min_{s \in \mathcal{H}} \int_{\hat{\Omega}} (s - g)^2 \mathrm{d}\boldsymbol{\xi}, \tag{24}$$

where the unknown spline function

$$s = \sum_{\boldsymbol{j} \in \mathbb{H}} c_{\boldsymbol{j}} \beta_{\boldsymbol{j}}$$

depends on the unknown coefficients $c_{\boldsymbol{j}}$ of the HB-splines $\beta_{\boldsymbol{j}} \in H$. In order to deal with the above problem, one may proceed by using Gaussian quadrature, arriving at the discrete optimization problem

$$\min_{c_{\boldsymbol{j}}} \sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \left(\sum_{\boldsymbol{j} \in \mathbb{H}} c_{\boldsymbol{j}} \beta_{\boldsymbol{j}}(\boldsymbol{\xi}_{\boldsymbol{i}}) - g(\boldsymbol{\xi}_{\boldsymbol{i}}) \right)^2, \tag{25}$$

where the points $\boldsymbol{\xi}_{\boldsymbol{i}}$ and positive weights $\gamma_{\boldsymbol{i}}$ are derived from the Gaussian nodes and weights, respectively. The unknowns $c_{\boldsymbol{j}}$ are determined by the normal equations

$$X^T X \boldsymbol{c} = X^T \boldsymbol{v} \tag{26}$$

with

$$X_{\boldsymbol{ij}} = \sqrt{\gamma_{\boldsymbol{i}}} \beta_{\boldsymbol{j}}(\boldsymbol{\xi}_{\boldsymbol{i}}), \quad \boldsymbol{v}_{\boldsymbol{i}} = \sqrt{\gamma_{\boldsymbol{i}}} g(\boldsymbol{\xi}_{\boldsymbol{i}}), \quad \boldsymbol{c} = (c_{\boldsymbol{j}})_{\boldsymbol{j} \in \mathbb{H}}. \tag{27}$$

It is known that the linear system (26) always has a unique solution. The solutions of the two problems (24) and (25) are identical if the Gauss quadrature exactly integrates the products $\beta_{\boldsymbol{i}} \beta_{\boldsymbol{j}}$ and $\beta_{\boldsymbol{i}} g$. Otherwise, the solution of the discrete problem is an approximation of the exact one.

Although global fitting is a simple method for spline projection, it requires evaluating the given function and the HB-splines at numerous Gauss points and solving a linear system. In fact, the matrix $X^T X$ is equal to the mass matrix (17) of the HB-splines in the case $\hat{\sigma} |\det \nabla_{\boldsymbol{\xi}} \boldsymbol{G}| = 1$. We conclude that this simple method is computationally too expensive.

4.2. Quasi-interpolation

Quasi-interpolation (QI) is another well-established methodology to construct approximations of a given function by splines. Typically it leads to lower computational costs than global fitting. Recently, the QI method has also been used to deal with the integrals arising in Isogeometric Boundary Element Methods (IGABEM), see [13, 23, 25].

Given a function g with domain $\hat{\Omega}$ and a set of blending functions $\Phi = \{\phi_{\boldsymbol{j}} : \boldsymbol{j} \in \mathcal{J}\}$, a quasi-interpolant of g takes the general form

$$Qg = \sum_{\boldsymbol{j} \in \mathcal{J}} \lambda_{\boldsymbol{j}}(g) \phi_{\boldsymbol{j}} \tag{28}$$

with certain linear coefficient functionals

$$\lambda_{\boldsymbol{j}}(g) = \int_{\operatorname{supp} \phi_{\boldsymbol{j}}} g(\boldsymbol{\xi}) \psi_{\boldsymbol{j}}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \tag{29}$$

that are determined by suitably chosen weight functions $\psi_{\boldsymbol{j}}$. The linear operator Q is called the quasi-interpolation operator. This operator is required to reproduce polynomials of degree p, in order to possess good approximation properties. Moreover, certain types of QI operators even reproduce the functions in $\operatorname{span} \Phi$, which makes them *projectors*.

A popular choice for the coefficient functionals is to use linear combinations of sampled values, corresponding to weight functions that are linear combinations of translates of Dirac delta functions. Here we consider only coefficient functionals of this type.

Method	applicable to HB	applicable to THB	Spline projector	#smpl per dof	#flops per dof
Kraft [38]	Yes	No	Yes	$\mathcal{O}(p^2)$	$\mathcal{O}(p^2)$
Speleers and Manni [49]	No	Yes	Yes	$\mathcal{O}(p^2)$	$\mathcal{O}(p^2)$
Speleers [48]	No	Yes	No	$\mathcal{O}(1)$	$\mathcal{O}(p^2)$
Giust et al. [29]	Yes	Yes	Yes	$\mathcal{O}(1)$	$\mathcal{O}(p^2)$

Table 1: Comparison of various hierarchical spline QI schemes for the bivariate case with respect to their applicability to HB- or THB-splines, the spline projector property, and the computational costs per degree of freedom (dof).

QI operators in hierarchical spline space were studied already by Kraft [38]. He introduced an HB-spline projector that is constructed in a recursive way. Later, these results have been extended to truncated hierarchical B-splines (THB-splines, see [26, 27]) by Speleers and Manni in [49]. In a subsequent paper [48], Speleers showed how to construct general QI operators for hierarchical spline spaces, which are not projectors but still possess good approximation properties. Recently, Giust et al. [29] established a new quasi-interpolation scheme for both HB-splines and THB-splines by employing local least-square fitting in restricted hierarchical spline spaces. This provides more efficient spline projectors, while obtaining the same accuracy as the previous quasi-interpolants.

The computational costs of QI operators depend on the required number of sampled values (#smpl) and on the number of floating point operations (#flops) needed to evaluate the coefficient functionals. Note that the evaluations of these functionals is perfectly suited for parallel computing. Table 1 summarizes the properties of various quasi-interpolants for bivariate hierarchical splines. It should be noted that these properties are based on certain assumptions the admissibility of the hierarchical meshes.

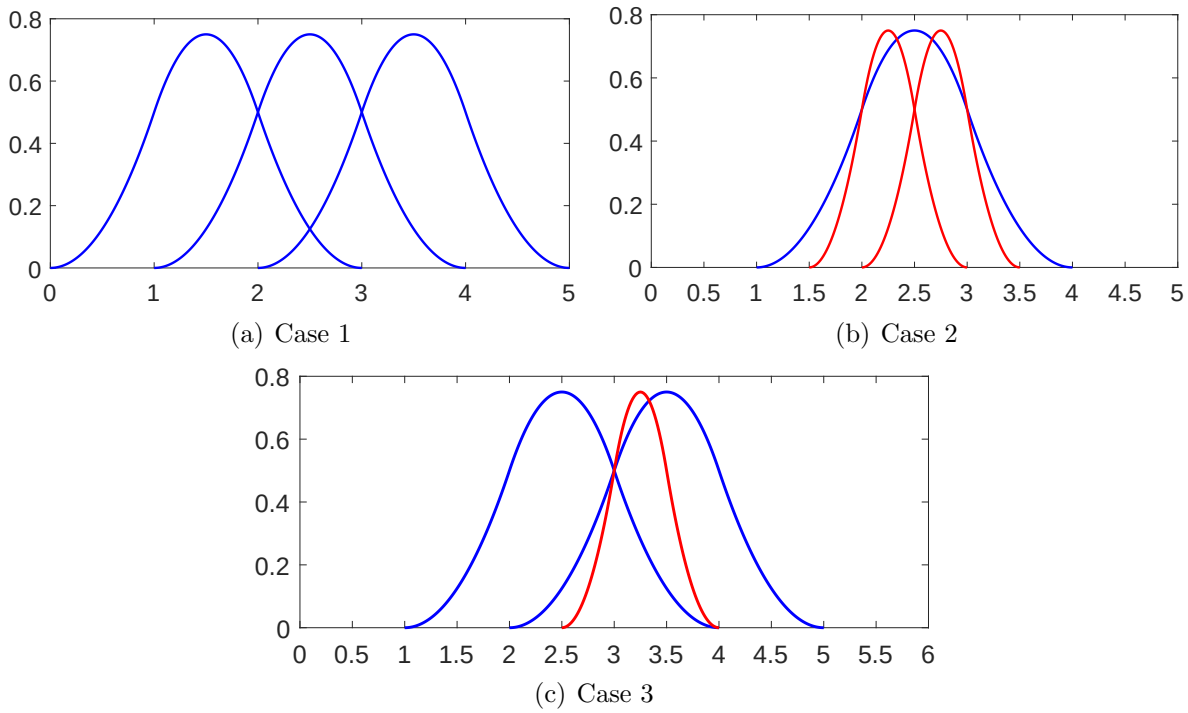
5. Building compact look-up tables

After replacing the elements of W in (19) by hierarchical spline functions via HB-spline projection, the integrals are transformed into sums of integrals of tensor-product B-splines (21). The next step of our approach is to evaluate those tri-product integrals, which are defined in the look-up tables (22), thereby preparing the accurate evaluation of (21).

Owing to the small supports of the B-splines, the majority of the entries in the look-up tables vanish. Non-zero entries are created only by B-spline-triplets $(\beta_{d,i_d}, \beta_{d,i'_d}, \beta_{d,i''_d})$ with overlapping supports. As noted before, we assume that the HB-splines used for isogeometric discretizations and spline projection are the same and that they are admissible of class 2. In this situation, the number of non-zero elements in (22) is relatively small and the B-splines β_{d,i_d}, β_{d,i'_d} and β_{d,i''_d} belong to the same level or to two successive levels (see Figure 4).

For uniform B-splines, the tri-product integrals (22) that do not involve boundary B-splines[2] can be classified into the following three cases:

[2]The remaining integrals, which involve boundary B-splines, only account for a relatively small portion of



(a) Case 1 (b) Case 2

(c) Case 3

Figure 4: The B-splines in the tri-product integrals (22) belong to at most two successive levels. They all belong to the same level (a), two of them belong to the finer level (b), or two of them belong to the coarser level (c). Note that the knots of the B-splines in (22) may be different from the ones shown in these figures.

• Case 1 – Figure 4(a): *β_{d,i_d}, β_{d,i'_d} and β_{d,i''_d} belong to the same level.* We define the normalized B-spline tri-product integrals

$$\Lambda_{ij}^{\sigma,\sigma',\sigma''} = \int_{\mathbb{R}} \hat{\beta}^{p,\sigma}_{[0,1,\ldots,p+1]}(\hat{\xi})\ \hat{\beta}^{p,\sigma'}_{[i,i+1,\ldots,i+p+1]}(\hat{\xi})\ \hat{\beta}^{p,\sigma''}_{[j,j+1,\ldots,j+p+1]}(\hat{\xi})\ \mathrm{d}\hat{\xi} \tag{30}$$

with $\sigma,\sigma',\sigma'' \in \{0,1\}, \sigma\sigma'\sigma'' = 0$ and $0 \le i,j \le p$, where $\hat{\beta}^{p,\sigma}_{\Xi}$ denotes the σth order derivative of the B-spline $\hat{\beta}$ of degree p and with knot vector Ξ. Note that the case $\sigma = \sigma' = \sigma'' = 1$ is not considered. These normalized integrals (30) can be precomputed and stored in a look-up table with $7(p+1)^2$ elements. As an example, Table 2 reports these values for B-splines of degree 2.

The integrals (22) required for assembling the system matrices can be evaluated via

$$T^{\mu,\nu}_{i_d,i'_d,i''_d} = \frac{1}{h_d^{\mu+\nu-1}} \begin{cases} \Lambda^{\mu,\nu,0}_{i'_d-i_d,i''_d-i_d} & \text{if} \quad i_d = \min\{i_d,i'_d,i''_d\} \\ \Lambda^{\nu,\mu,0}_{i_d-i'_d,i''_d-i'_d} & \text{if} \quad i'_d = \min\{i_d,i'_d,i''_d\} \\ \Lambda^{0,\mu,\nu}_{i_d-i''_d,i'_d-i''_d} & \text{if} \quad i''_d = \min\{i_d,i'_d,i''_d\} \end{cases} . \tag{31}$$

the overall computations. They can be evaluated via Gauss quadrature without compromising the efficiency of the approach.

i,j	$\Xi_{ij}^{0,0,0}$	$\Xi_{ij}^{0,0,1}$	$\Xi_{ij}^{0,1,0}$	$\Xi_{ij}^{0,1,1}$	$\Xi_{ij}^{1,0,0}$	$\Xi_{ij}^{1,0,1}$	$\Xi_{ij}^{1,1,0}$
$0,0$	$12/35$	0	0	$2/5$	0	$2/5$	$2/5$
$0,1$	$43/420$	$31/120$	$-31/240$	$-7/40$	$-31/240$	$-7/40$	$17/60$
$0,2$	$1/840$	$1/120$	$-1/240$	$-1/40$	$-1/240$	$-1/40$	$1/60$
$1,0$	$43/420$	$-31/240$	$31/120$	$-7/40$	$-31/240$	$17/60$	$-7/40$
$1,1$	$43/420$	$31/240$	$31/240$	$17/60$	$-31/120$	$-7/40$	$-7/40$
$1,2$	$1/168$	$7/240$	0	$1/120$	$-7/240$	$-7/60$	$1/120$
$2,0$	$1/840$	$-1/240$	$1/120$	$-1/40$	$-1/240$	$1/60$	$-1/40$
$2,1$	$1/168$	0	$7/240$	$1/120$	$-7/240$	$1/120$	$-7/60$
$2,2$	$1/840$	$1/240$	$1/240$	$1/60$	$-1/120$	$-1/40$	$-1/40$

Table 2: The look-up table for tri-product integrals of quadratic uniform B-splines with respect to Case 1.

i	$\Gamma\Xi_{i3}^{0,0,0}$	$\Gamma\Xi_{i3}^{0,0,1}$	$\Gamma\Xi_{i3}^{0,1,0}$	$\Gamma\Xi_{i3}^{0,1,1}$	$\Gamma\Xi_{i3}^{1,0,0}$	$\Gamma\Xi_{i3}^{1,0,1}$	$\Gamma\Xi_{i3}^{1,1,0}$
0	0	0	0	0	0	0	0
1	8	70	21	196	-91	-644	-224
2	359	$1,085$	581	$2,352$	$-1,666$	$-2,268$	$-1,988$
3	$1,209$	-441	882	$1,596$	-441	$7,896$	$1,596$
4	526	$-2,002$	0	728	$2,002$	$-3,472$	728
5	19	-182	-14	168	196	$-1,512$	-112
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

Table 3: A slice of the look-up table for tri-product integrals (22) of quadratic uniform B-splines with respect to Case 2, scaled by the factor $\Gamma = 6,720$.

i,j	$\Gamma\Xi_{ij}^{0,0,0}$	$\Gamma\Xi_{ij}^{0,0,1}$	$\Gamma\Xi_{ij}^{0,1,0}$	$\Gamma\Xi_{ij}^{0,1,1}$	$\Gamma\Xi_{ij}^{1,0,0}$	$\Gamma\Xi_{ij}^{1,0,1}$	$\Gamma\Xi_{ij}^{1,1,0}$
$0,0$	1	7	7	56	-14	-84	-84
$0,1$	0	0	0	0	0	0	0
$0,2$	0	0	0	0	0	0	0
$1,0$	125	427	427	$1,624$	-854	$-2,436$	$-2,436$
$1,1$	0	0	0	0	0	0	0
$1,2$	0	0	0	0	0	0	0
$2,0$	$1,683$	$3,129$	$3,129$	$6,888$	$-6,258$	$-6,972$	$-6,972$
$2,1$	19	182	14	168	-196	$-1,512$	-112
$2,2$	0	0	0	0	0	0	0

Table 4: Three slices of the look-up table for tri-product integrals (22) of quadratic uniform B-splines with respect to Case 3, scaled by the factor $\Gamma = 26,880$.

This can be confirmed by exploiting the properties of B-splines plus index shifting and swapping operations. Here h_d represents the length of the knot span of the basis functions β_{d,i_d}, β_{d,i'_d} and β_{d,i''_d}.

• Case 2 – Figure 4(b): *Two of the B-splines β_{d,i_d}, β_{d,i'_d} and β_{d,i''_d} belong to the finer level, the remaining one belongs to the coarser level.* Suppose these B-splines are at level ℓ and

$\ell+1$ respectively. We shift them such that the one at level ℓ is transformed to the B-spline with global index $2^\ell+(\ell+1)p+\lfloor(p+1)/2\rfloor$. We then consider the standardized tri-product integrals

$$\Lambda_{ij}^{\sigma,\sigma',\sigma''}=\int_{\mathbb{R}}\hat{\beta}^{p,\sigma}_{[\frac{i}{2},\frac{i+1}{2},\ldots,\frac{i+p+1}{2}]}(\hat{\xi})\ \hat{\beta}^{p,\sigma'}_{[\lfloor\frac{p+1}{2}\rfloor,\lfloor\frac{p+1}{2}\rfloor+1,\ldots,\lfloor\frac{p+1}{2}\rfloor+p+1]}(\hat{\xi})\cdot\hat{\beta}^{p,\sigma''}_{[\frac{j}{2},\frac{j+1}{2},\ldots,\frac{j+p+1}{2}]}(\hat{\xi})\,\mathrm{d}\hat{\xi} \tag{32}$$

with $\sigma,\sigma',\sigma''\in\{0,1\},\sigma\sigma'\sigma''=0$ and $0\le i,j\le 2\lfloor 3(p+1)/2\rfloor-1$, which can be pre-calculated and constitute a look-up table with $7(2\lfloor 3(p+1)/2\rfloor)^2$ entries (see Table 3 for a slice of it). We then evaluate integrals (22) via

$$T^{\mu,\nu}_{i_d,i'_d,i''_d}=\left(\frac{2}{\tilde{h}_d}\right)^{\mu+\nu-1}\begin{cases}\Lambda^{\nu,\mu,0}_{i'_d-2i_d+\rho,i''_d-2i_d+\rho} & \text{if}\quad i_d=\min\{i_d,i'_d,i''_d\}\\ \Lambda^{\mu,\nu,0}_{i_d-2i'_d+\rho,i''_d-2i'_d+\rho} & \text{if}\quad i'_d=\min\{i_d,i'_d,i''_d\}\\ \Lambda^{0,\mu,\nu}_{i_d-2i''_d+\rho,i'_d-2i''_d+\rho} & \text{if}\quad i''_d=\min\{i_d,i'_d,i''_d\}\end{cases} \tag{33}$$

with $\rho=2\lfloor(p+1)/2\rfloor+\ell p$, where $\tilde{h}_d$ represents the length of the knot span of B-splines at level $\ell+1$ in the d–th coordinate direction.

• Case 3 – Figure 4(c): *Two of the B-splines β_{d,i_d}, β_{d,i'_d} and β_{d,i''_d} belong to the coarser level, the other one belongs to the finer level.* Again we assume these B-splines are at level ℓ and $\ell+1$ respectively, and shift them such that the first B-spline of level ℓ (i.e., the one with the lower index) has the global index $(\ell+1)p+2^\ell+\lfloor(p+1)/2\rfloor$. Consequently, the integrals (22) can be derived from the tri-product integrals

$$\Lambda_{ij}^{\sigma,\sigma',\sigma''}=\int_{\mathbb{R}}\hat{\beta}^{p,\sigma}_{[\frac{i}{2},\frac{i+1}{2},\ldots,\frac{i+p+1}{2}]}(\hat{\xi})\ \hat{\beta}^{p,\sigma'}_{[\lfloor\frac{p+1}{2}\rfloor,\lfloor\frac{p+1}{2}\rfloor+1,\ldots,\lfloor\frac{p+1}{2}\rfloor+p+1]}(\hat{\xi})\cdot\hat{\beta}^{p,\sigma''}_{[j+\lfloor\frac{p+1}{2}\rfloor,j+\lfloor\frac{p+1}{2}\rfloor+1,\ldots,j+\lfloor\frac{p+1}{2}\rfloor+p+1]}(\hat{\xi})\,\mathrm{d}\hat{\xi} \tag{34}$$

via the formula

$$T^{\mu,\nu}_{i_d,i'_d,i''_d}=\left(\frac{2}{\tilde{h}_d}\right)^{\mu+\nu-1}\begin{cases}\Lambda^{\mu,\nu,0}_{i_d-2i'_d+\rho,i''_d-i'_d} & \text{if}\quad i'_d\le i''_d<i_d\\ \Lambda^{\mu,0,\nu}_{i_d-2i''_d+\rho,i'_d-i''_d} & \text{if}\quad i''_d\le i'_d<i_d\\ \Lambda^{\nu,\mu,0}_{i'_d-2i_d+\rho,i''_d-i_d} & \text{if}\quad i_d\le i''_d<i'_d\\ \Lambda^{\nu,0,\mu}_{i'_d-2i''_d+\rho,i_d-i''_d} & \text{if}\quad i''_d\le i_d<i'_d\\ \Lambda^{0,\mu,\nu}_{i''_d-2i_d+\rho,i'_d-i_d} & \text{if}\quad i_d\le i'_d<i''_d\\ \Lambda^{0,\nu,\mu}_{i''_d-2i'_d+\rho,i_d-i'_d} & \text{if}\quad i'_d\le i_d<i''_d\end{cases}, \tag{35}$$

where $\tilde{h}_d$ and ρ are as in the previous case. The indices in (34) and (35) satisfy $\sigma,\sigma',\sigma''\in\{0,1\},\sigma\sigma'\sigma''=0$, $0\le i\le 2\lfloor 3(p+1)/2\rfloor-1, 0\le j\le p$ and $\rho=2\lfloor(p+1)/2\rfloor+\ell p$ respectively, and all the integrals (34) can be precomputed and stored in a look-up table with $7(p+1)(2\lfloor 3(p+1)/2\rfloor)$ entries. Table 4 presents three slices of this table.

By exploiting the symmetry and compact supports of the B-splines, the look-up tables can be further compressed. Consequently, the duplicated entries and most of the zero entries in Tables 2, 3 and 4 can be omitted. On the other hand, the symmetric structure of these tables simplifies the subsequent look-up procedure.

Our method is also applicable to non-uniform B-splines. In this case, we evaluate all the non-zero tri-product integrals (22) in each of the 2 coordinate directions via standard Gauss quadrature, forming 6 look-up tables. To estimate the sizes of these look-up tables, we take the number of levels of the HB-splines into account and use the quantity (2) and Lemma 1 introduced in the preliminary section. Consequently the numbers of nonzero entries in Tables 2, 3 and 4 in each coordinate direction are $\mathcal{O}(Lp^3 + 2^{L+1}p^2)$, $\mathcal{O}(Lp^3 + 2^L p^2)$ and $\mathcal{O}(Lp^3 + 2^L p^2)$ respectively. As a difference to the case of uniform B-splines, the sizes of the look-up tables for non-uniform case depend on both the polynomial degree and on the number of levels of the HB-splines.

6. Efficient matrix assembly

With the established look-up tables at hand, the remaining step of our approach is to perform the sum and product operations in (23) more efficiently. To this end, we further employ the sum-factorization technique to accelerate the matrix assembly procedure.

When used in isogeometric analysis, the method of sum factorization [1, 8, 15, 42] achieves its efficiency by exploiting the tensor-product structure of multivariate splines. Essentially, a univariate quadrature is performed in each stage of the algorithm realizing the multivariate quadrature. Applying this technique in our framework, we rewrite the sums in (23) as

$$S_{\boldsymbol{i},\boldsymbol{i}'} \approx \underbrace{\sum_{\mu=1}^{2}\sum_{\nu=1}^{2}\sum_{i_2''\in\mathbb{H}_2} T_{i_2,i_2',i_2''}^{\delta(\mu,2),\delta(\nu,2)} \underbrace{\sum_{i_1''\in\mathbb{H}_1[i_2'']} w_{i_1''i_2''}^{\mu,\nu} T_{i_1,i_1',i_1''}^{\delta(\mu,1),\delta(\nu,1)}}_{=\,S_{i_1 i_1' i_2''}^{\mu,\nu,1}}}_{=\,S_{i_1 i_2 i_1' i_2'}}, \tag{36}$$

where we use the index sets

$$\mathbb{H}_2 = \{i_2 \in \mathbb{Z} \mid \exists i_1 \in \mathbb{Z} \,:\, (i_1, i_2) \in \mathbb{H}\} \quad \text{and} \quad \mathbb{H}_1[i_2] = \{i_1 \in \mathbb{Z} \mid (i_1, i_2) \in \mathbb{H}\}.$$

Here we introduce two auxiliary sparse tensors $\boldsymbol{S}^{(1)} := \left(S_{i_1 i_1' i_2''}^{\mu,\nu,1}\right), \boldsymbol{S}^{(2)} := \left(S_{i_1 i_2 i_1' i_2'}\right)$.

The sums for all pairs $(\boldsymbol{i}, \boldsymbol{i}')$ are evaluated in a recursive way: The innermost loop computes and stores the first tensor $\boldsymbol{S}^{(1)}$ with at most $\mathcal{O}(Np^2)$ non-zero elements, which is subsequently used for evaluating the tensor $\boldsymbol{S}^{(2)}$ with $\mathcal{O}(Np^2)$ non-zero entries in the second stage of the recursion.

Note that we visit the elements of the index set $\mathbb{H}$ in a particular way: The outer loop runs through all indices i_2'' that have at least one associated multi-index $(i_1'', i_2'') \in \mathbb{H}$, and the inner loop visits all indices i_1'' that are relevant for the given value of i_2''. Consequently, referring to Figure 1 (right), the outer loop runs through all the rows with at least one index

in $\mathbb{H}$, while the inner loop visits all elements of the row under consideration. To implement this efficiently, a row-wise ordering of $\mathbb{H}$ would be needed.

We will show that it is possible to perform the assembly without such an ordering. In fact, an arbitrary enumeration of the index set $\mathbb{H}$ can be used instead. The assembly is realized by the following algorithm:

Algorithm StiffnessMatrixAssembly ▷ Stiffness matrix assembly via sum-factorization
for $\mu = 1, 2$ **do**
  for $\nu = 1, 2$ **do**
    for $\boldsymbol{i}'' \in \mathbb{H}$ **do**
      for $i_1 \in \mathcal{I}_1(i_1'')$ **do** ▷ indices of B-splines that overlap $\beta_{1,i_1''}$
        for $i_1' \in \mathcal{I}_1(i_1) \cap \mathcal{I}_1(i_1'')$ **do** ▷ indices of B-splines that overlap β_{1,i_1} and $\beta_{1,i_1''}$
          $S^{\mu,\nu,1}_{i_1 i_1' i_2''} = 0$ ▷ initialization of the first tensor
  for $\boldsymbol{i}'' \in \mathbb{H}$ **do**
    for $i_1 \in \mathcal{I}_1(i_1'')$ **do**
      for $i_1' \in \mathcal{I}_1(i_1) \cap \mathcal{I}_1(i_1'')$ **do**
        $S^{\mu,\nu,1}_{i_1 i_1' i_2''} \mathrel{+}= w^{\mu,\nu}_{i_1'' i_2''} T^{\delta(\mu,1),\delta(\nu,1)}_{i_1,i_1',i_1''}$ ▷ evaluation of the first tensor
for $\boldsymbol{i} \in \mathbb{H}$ **do**
  for $\boldsymbol{i}' \in \mathcal{N}(\boldsymbol{i})$ **do** ▷ indices of B-splines in H that overlap $\beta_{\boldsymbol{i}}$
    $S_{i_1 i_2 i_1' i_2'} = 0$ ▷ initialization of the stiffness matrix
    for $\mu = 1, 2$ **do**
      for $\nu = 1, 2$ **do**
        for $i_2'' \in \mathcal{I}_2(i_2) \cap \mathcal{I}_2(i_2')$ **do** ▷ indices of B-splines that overlap β_{2,i_2} and $\beta_{2,i_2'}$
          $S_{i_1 i_2 i_1' i_2'} \mathrel{+}= T^{\delta(\mu,2),\delta(\nu,2)}_{i_2,i_2',i_2''} S^{\mu,\nu,1}_{i_1 i_1' i_2''}$ ▷ evaluation of the stiffness matrix
return S

The index sets $\mathcal{I}_d(i_d)$ and $\mathcal{N}(\boldsymbol{i})$ are defined by

$$\mathcal{I}_d(i_d) = \{i_d' \in \mathbb{I}^{\text{level}(i_d)-1} \cup \mathbb{I}^{\text{level}(i_d)} \cup \mathbb{I}^{\text{level}(i_d)+1} \ : \ \text{supp}(\beta_{d,i_d}\beta_{d,i_d'}) \neq \emptyset\}$$

and

$$\mathcal{N}(\boldsymbol{i}) = \{\boldsymbol{i}' \in \mathbb{H} \ : \ \text{supp}(\beta_{\boldsymbol{i}}\beta_{\boldsymbol{i}'}) \neq 0\} \tag{37}$$

respectively, where $\mathbb{I}^{-1} = \emptyset$ and $\mathbb{I}^{L+1} = \emptyset$. According to Lemma 1 and 2, the sizes of these index sets ($\mathcal{I}_d(i_d)$ and $\mathcal{N}(\boldsymbol{i})$) do not exceed

$$\lceil (p+1)/2 \rceil + 6(p+1) = \mathcal{O}(p)$$

and

$$10(p+1)^2 + (\lceil (p+1)/2 \rceil + (p+1))^2 = \mathcal{O}(p^2)$$

respectively.

For simplifying the implementation and reducing the complexity of the above algorithm, we evaluate some extra elements $S^{\mu,\nu,1}_{i_1 i_1' i_2''}$ that are not needed for computing the tensor $\boldsymbol{S}^{(2)}$. The sub-indices i_1, i_1' of these elements do not belong to the set

$$\mathbb{H}_1 = \{i_1 \in \mathbb{Z} \mid \exists i_2 \in \mathbb{Z} \ : \ (i_1, i_2) \in \mathbb{H}\}.$$

Otherwise, we have to check if the indices i_1, i_1' exist in $\mathbb{H}_1$ in each loop, which results in a non-constant cost per function. Also in the second stage, some elements $S^{\mu,\nu,1}_{i_1 i_1' i_2''}$ required for evaluating the final stiffness matrix are not pre-computed in the first stage and are zero by default. The sub-indices i_2'' of these elements do not belong to the set $\mathbb{H}_2$.

Remark 1. Compared with the case of tensor-product B-splines [42], the Algorithm StiffnessMatrixAssembly for HB-splines becomes more sophisticated due to the non-trivial task of generating the index sets $\mathcal{I}_d(i_d)$ and $\mathcal{N}(\boldsymbol{i})$ in each loop. However, as analyzed in the next section, the complexity of the new algorithm remains unchanged under a mild assumption. ◇

7. Complexity analysis

In order to discuss the computational complexity of our algorithm and compare it with Gauss quadrature, we denote the dimension and degree of the hierarchical spline space $\mathcal{H}$ by N and p, respectively. The number of non-zero entries in the system matrix, which equals $\mathcal{O}(Np^2)$ by Lemma 2 and under the assumption of admissible meshes, is a lower bound for the computational costs of matrix assembly.

A certain portion of the computation time is spent for memory operations (e.g., sparse matrix allocation and memory write), and for finding all the basis functions that possess a support intersection with a given B-spline $\beta_{\boldsymbol{i}}$ in H, i.e., for computing the index sets $\mathcal{N}(\boldsymbol{i})$. This is needed by any approach towards matrix assembly for HB-splines. We do not include these costs in the analysis presented below, since they depend heavily on the data structures used for the sparse matrix representation and on the HB-spline implementation. We will assume that all the index sets $\mathcal{N}(\boldsymbol{i})$, $\boldsymbol{i} \in \mathbb{H}$, which have size $|\mathcal{N}(\boldsymbol{i})| = \mathcal{O}(p^2)$ due to the admissibility assumption, have been precomputed and can be enumerated in $\mathcal{O}(p^2)$ time. We do, however, include these costs in the experimental part, which will demonstrate that they do not compromise the good behavior of the new method.

We now discuss the complexity of stiffness matrix assembly based on Gauss quadrature and the new method, respectively.

For the standard element-wise Gauss quadrature, one needs to iterate over $\mathcal{O}(N)$ elements and evaluate $\mathcal{O}(p^4)$ values and derivatives of basis functions at each of the $\mathcal{O}(q^2)$ Gauss nodes per element ($q = p + 1$ is a reasonable choice for maintaining the optimal convergence rate [39]), resulting in a total complexity of $\mathcal{O}(Np^6)$.

The computational effort of the new proposed method is described by the following theorem.

Theorem 1. *We consider an admissible HB-spline discretization of degree p and with N degrees of freedom. As N increases, the number of flops per dof required for assembling the stiffness matrix S (15) via the proposed method tends to $\mathcal{O}(p^3)$.*

Proof. Our algorithm, which is outlined in Section 3, consists of three stages: Approximating the integrals via spline projection, building three compact look-up tables and assembling the matrix via sum-factorization.

In the first step, we choose the local (T)HB-spline projector [29] as the projection operator. As stated in Section 4 (cf. Table 1), this type of quasi-interpolant requires asymptotically $\mathcal{O}(p^2)$ flops per dof.

We already noticed that the sizes of the built look-up tables for uniform B-splines and non-uniform B-splines are $\mathcal{O}(p^2)$ and no more than $\mathcal{O}(Lp^3 + 2^{L+1}p^2)$, respectively. The number of Gauss quadrature nodes needed for calculating each of the entries is at most

$$\lceil \frac{3p+1}{2} \rceil (p+1) = \mathcal{O}(p^2),$$

hence the total numbers of quadrature nodes per dof (i.e., the numbers of flops per dof) required for generating the look-up tables for uniform and non-uniform B-splines are asymptotically equal to

$$\lim_{N\to\infty} \frac{1}{N} \cdot \mathcal{O}(p^2) \cdot \mathcal{O}(p^2) = 0$$

and

$$\lim_{N\to\infty} \frac{1}{N} \cdot \mathcal{O}(Lp^3 + 2^{L+1}p^2) \cdot \mathcal{O}(p^2) = 0,$$

respectively. Therefore, the computational cost for building the look-up tables is negligible compared to the overall effort.

The task of the matrix assembly stage is to compute two auxiliary tensors in a recursive way. To estimate the cost of Algorithm StiffnessMatrixAssembly, it suffices to consider only the non-zero elements and terms in (36). The first loop, which evaluates the first tensor, computes

$$4 \cdot N \cdot \mathcal{O}(p) \cdot \mathcal{O}(p)$$

non-zero elements $S^{\mu,\nu,1}_{i_1 i_1' i_2''}$, requiring 2 flops per element. The computation of the second tensor with

$$4 \cdot N \cdot \mathcal{O}(p^2)$$

non-zero elements proceeds by summing over $\mathcal{O}(p)$ indices i_2'' and needs 2 flops per term. Thus the cost of the matrix assembly step tends to $\mathcal{O}(p^3)$ per dof as N increases.

Summing up, the computational effort per dof for the proposed algorithm asymptotically amounts to

$$\mathcal{O}(p^2) + 0 + \mathcal{O}(p^3) = \mathcal{O}(p^3). \qquad \square$$

Remark 2. Although we restricted the presentation to the assembly of stiffness matrices, the advection matrix A (16) and the mass matrix M (17) can be dealt with analogously. The computational complexities of assembling these three kinds of matrices possess the same order, i.e., $\mathcal{O}(Np^6)$ for Gauss quadrature and $\mathcal{O}(Np^3)$ for our method. $\diamond$

8. Numerical experiments

In this section, we demonstrate the performance of the proposed algorithm by solving a 2D Poisson problem with Dirichlet boundary conditions on four different domains, which

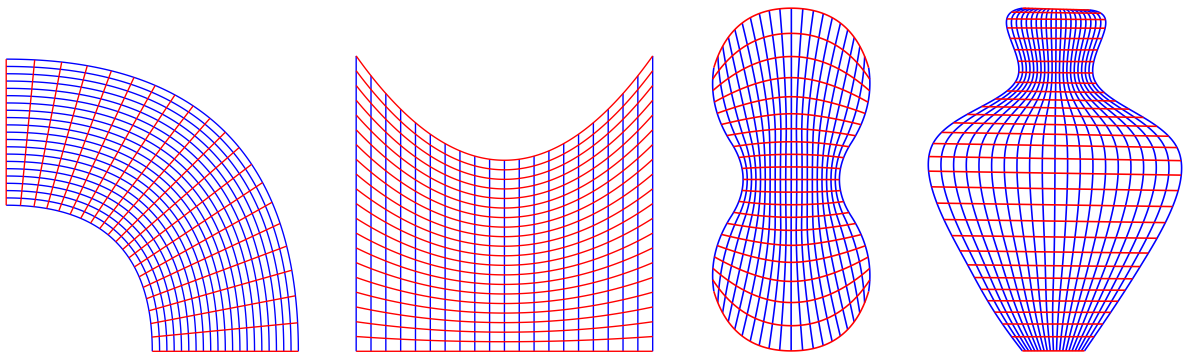


Figure 5: Four different computational domains used in the experiments. Each domain is parameterized by single-patch B-splines, and the corresponding iso-parametric curves are demonstrated.

are illustrated in Figure 5. First, the theoretical complexity of our method in Theorem 1 is verified by exploring the dependence of the computational time on the number of degrees of freedom and on the polynomial degree. We show the superiority of our method by comparing it with the standard Gauss quadrature in terms of the computational time. Second, the convergence behavior for adaptive refinement with the new method is investigated.

The algorithms are implemented in C++ using the `G+Smo` software [37], and all the tests are run on a desktop PC with 32GB of RAM and an octa-core Intel i7 CPU @3.4GHz. For the purpose of obtaining a more accurate result, in each example we take the average of the computational times of 10 repeated experiments. In addition, although we do not consider the costs of memory access and computing the index set $\mathcal{N}(\boldsymbol{i})$ for any basis $\beta_{\boldsymbol{i}}$ in the complexity analysis, the reported computational time in this section does include them. However, we do not consider the costs of the spline projection (first step of the algorithm) since the available implementation is not yet sufficiently optimized. We feel that it is still justified to present these experimental results since the overall computational complexity is dominated by the last step of the assembly procedure.

8.1. N-dependence of the computational time

We firstly investigate the relation between the complexity and the number of degrees of freedom N. The tests are performed on four computational domains (see Figure 5(a)–5(d)) using different HB-splines and polynomial degrees ($p = 3, 4, 5$).

On each domain, we report the assembly time for six hierarchical meshes with monotonically increasing numbers of degrees of freedom. These meshes are generated by repeatedly performing a dyadic refinement to all the cells of the initial hierarchical meshes depicted in Figure 6. Note that the post-processing step is indispensable so that all the hierarchical meshes are admissible of class 2.

Figure 7(a)–7(d) report the time needed for the system matrix formation on different domains with various N and degrees p. All these log–log plots reveal that the computational time of our method nearly grows linearly as the number of degrees of freedom increases (note that the slope is almost 1), which is consistent with the theoretical result presented

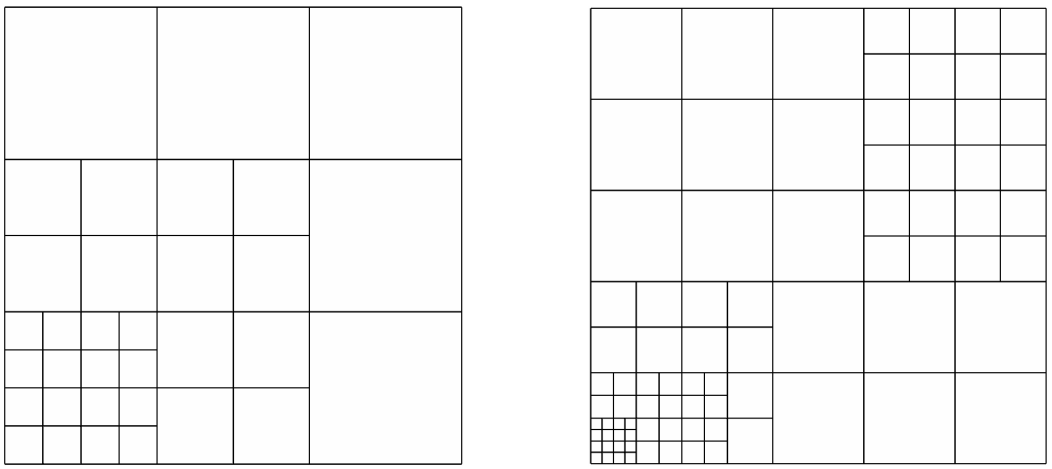


Figure 6: The initial hierarchical meshes for different domains. The left resp. right mesh is used for domain (a) and (c) resp. (b) and (d).

in Theorem 1. It also shows that the costs of memory access and computing the index sets $\mathcal{N}(\boldsymbol{i})$ do not have an obvious impact on the N-dependence behavior.

8.2. Degree-dependence of the computational time

Next we explore the dependence of the complexity on the spline degree p both for Gauss quadrature and the new method. As suggested in [39, 42], we choose $p+1$ nodes per element for Gauss quadrature, which suffices to obtain the optimal convergence rate.

The tests are conducted on four different computational domains. For each domain, we report the assembly time for a sequence of hierarchical meshes with four levels and spline degrees varying from 2 to 10. For each degree p, we choose the same hierarchical mesh for all the domains, and the corresponding hierarchical mesh is generated in the following way: a tensor-product mesh consisting of $4p \times 4p$ cells is repeatedly refined by selecting the left-bottom $2p \times 2p$ cells of each level to form the subdomain at the next level. Obviously, these meshes are admissible of class 2. As an example, we show the mesh for case $p = 2$ in Figure 8.

Figure 9(a)–9(d) report the required time per dof of these approaches, which is needed for assembling the system matrix. The slopes depicted in these log-log plots demonstrate that our method achieves the theoretical asymptotic behavior analyzed in Theorem 1. In contrast, the growth of the time of Gauss quadrature with respect to the degree p is lower than predicted by the theoretical analysis, i.e., $\mathcal{O}(Np^6)$, at least for $p \leq 10$. This is probably related to the implementation of Gauss quadrature provided in the `G+Smo` library [37], which has been particularly optimized for the low degree cases.

8.3. Convergence behavior

Besides the N-dependence and degree-dependence behavior, we also verify the convergence performance of the proposed method embedded into an adaptive refinement framework, which is fulfilled by solving a Poisson problem on the quarter annulus-shaped domain

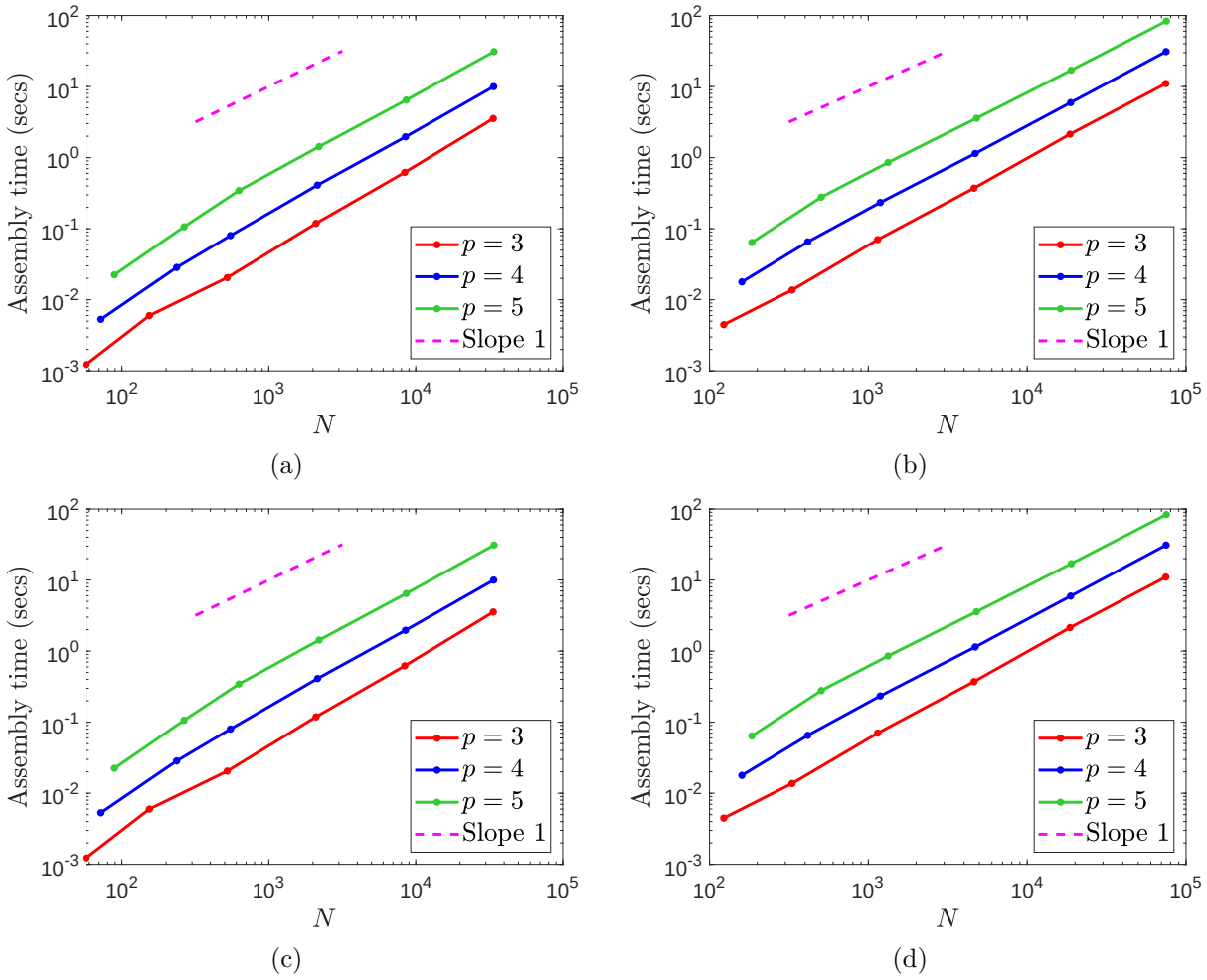


Figure 7: N-dependence of the assembly time on four computational domains with different spline degrees ($p = 3, 4, 5$). (a)–(d) correspond to domain (a)–(d) (cf. Figure 5(a)–5(d)) respectively.

(Figure 5(a)) with Dirichlet boundary condition and known exact solution

$$u(x, y) = e^{-100((x-1)^2+(y-1)^2)}.$$

With a uniform tensor-product B-spline as the initialization, an adaptive strategy for numerically solving the Poisson problem by our method repeats the following steps:

- Solve the discrete problem (14). The boundary coefficients of the approximate solution u_h are determined by least square fitting of the exact solution at boundary points with B-splines, the remaining coefficients are computed by solving a linear system using the preconditioned conjugate gradient method with incomplete Cholesky factorization.
- Estimate the error distribution. An element-based posteriori error estimator is employed and the error on each cell $\mathcal{C} \in \Omega$ is defined in (38).

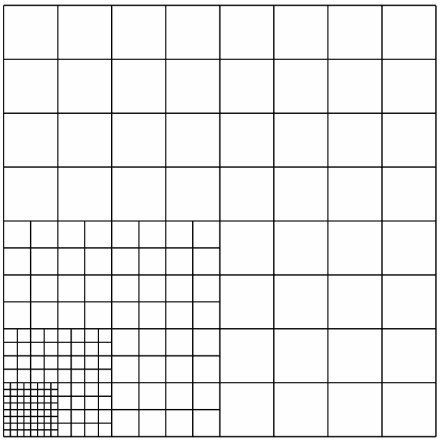


Figure 8: An admissible mesh of class 2 for biquadratic HB-splines with four levels.

- Mark the cells with large errors and refine them. The criterion for marking cells is introduced below.
- Apply the algorithms presented in [10] to ensure the admissibility of the hierarchical mesh, which is a prerequisite for the proposed approach.

The procedure runs until the maximal number of levels $L_{\max}$ is achieved.

To estimate the error, we consider at least C^1-continuous discrete solutions u_h (i.e., $p \geq 2$). For a posteriori error estimator [10], the estimate $\varepsilon_{\mathcal{C}}$ of the local error on a cell $\mathcal{C} \in \Omega$ is defined as

$$\varepsilon_{\mathcal{C}}(u_h) = h_{\mathcal{C}} \| f + \Delta u_h \|_{L^2(\mathcal{C})}, \tag{38}$$

where $h_{\mathcal{C}}$ denotes the diameter of cell $\mathcal{C}$.

A cell $\mathcal{C}$ is marked for refinement provided the criterion

$$\varepsilon_{\mathcal{C}} \geq \tau$$

is satisfied for a threshold τ. We use the second strategy that is discussed in [26] for choosing τ, i.e., a fixed percentage (denoted as η) of all cells should be marked in each refinement step. The parameter η is set as 25% in our experiment.

In the numerical test, we start with a uniform mesh consisting of 8×8 elements and repeat the refinement procedure five times, i.e., $L_{\max} = 5$. Figure 10 demonstrates the convergence plots both for quadratic and cubic splines. With respect to the degrees of freedom used to achieve a certain accuracy, the adaptive hierarchical approach is, as expected, superior to a uniform refinement. This is due to the fact that, the exact solution of the Poisson problem is a smooth function with a peak at the location $(1, 1)$, once the error estimates are dominated by the ones around the peak, only few cells around the peak are marked for refinement in the adaptive approach (see the second row of Figure 10). In contrast, the uniform method refines all the cells, which results in an over-refinement of the domain. We also compare the adaptive refinement by the new method with the one by Gauss quadrature, which indicates

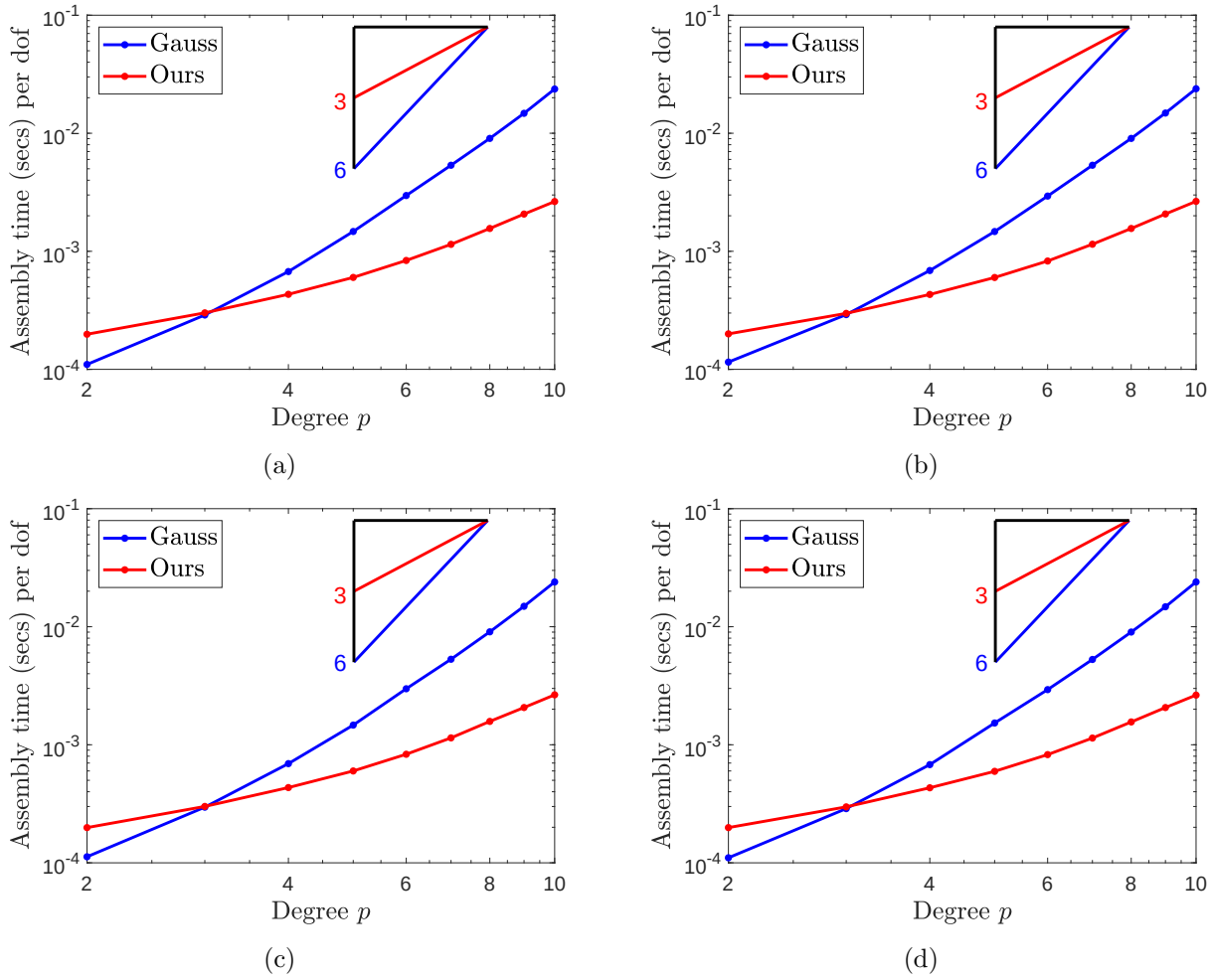
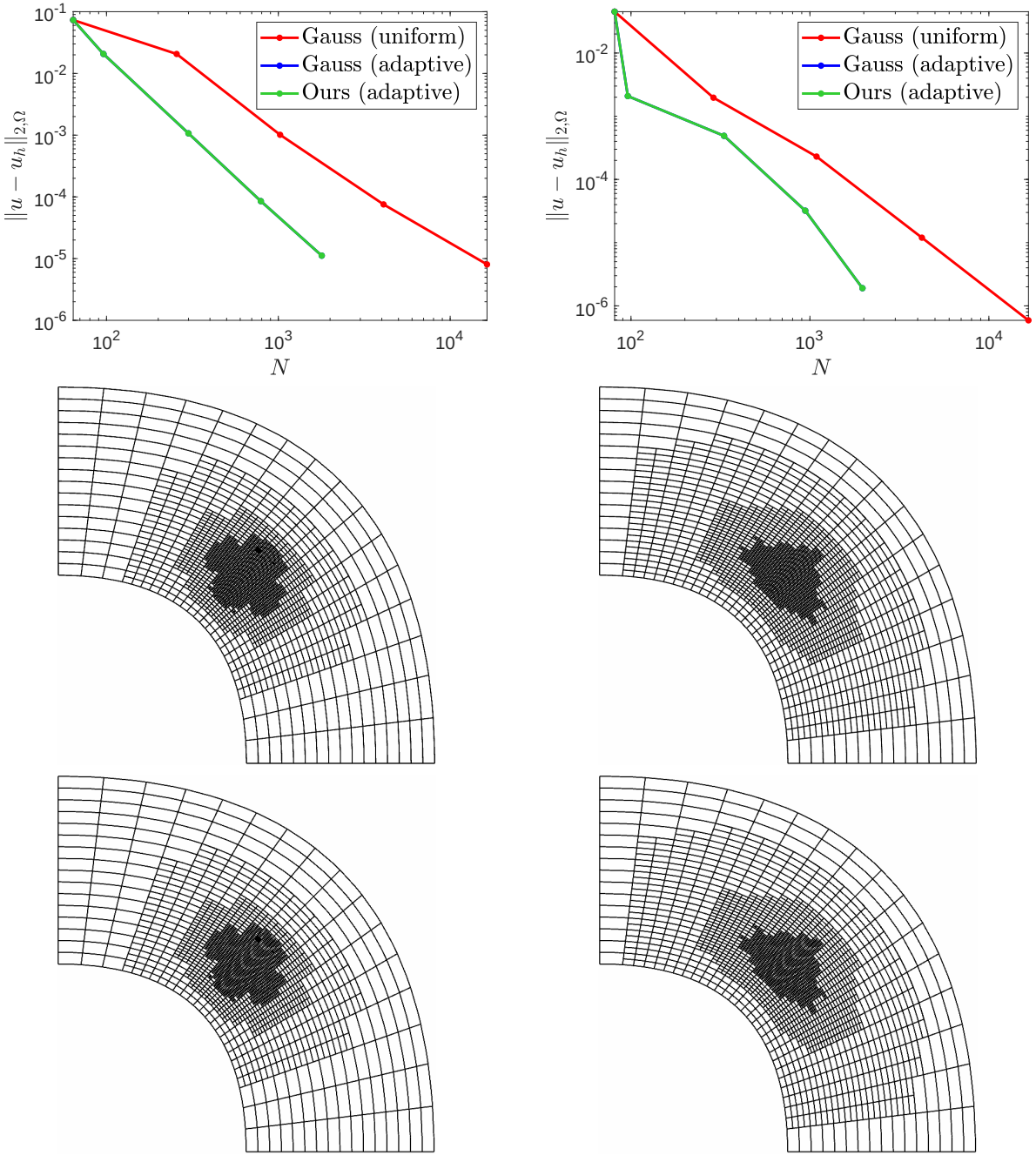


Figure 9: p-dependence of the assembly time of Gauss quadrature and the proposed method on different computational domains. (a)–(d) correspond to domain (a)–(d) (cf. Figure 5(a)–5(d)) respectively.

that the accuracy of our method is maintained provided the spline projection is sufficiently accurate. Indeed, we cannot visually detect any difference between the two methods, which is due to the fact that the spline projections used by our methods are very accurate. For this reason, we provide Table 5 that shows the subtle differences.

9. Conclusions and future work

Based on the previous paper [42], this work introduces an efficient approach for assembling the matrices in IgA with bivariate HB-splines. The novel approach relies on three key ingredients: (1) projecting the common matrix-valued function into a suitable hierarchical spline space via quasi-interpolation; (2) building three compact look-up tables; (3) assembling the matrices by employing the sum-factorization technique. We present a detailed complexity analysis, which shows that the number of flops per dof required for assembling the system matrices via the proposed method tends to $\mathcal{O}(p^3)$ as N increases. In addition, the



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

Figure 10: Convergence of error in L^2 norm with respect to degree $p = 2$ (left) and $p = 3$ (right) for the quarter annulus-shaped domain. The first row shows the error plots for uniform refinement by Gauss quadrature, adaptive refinements by Gauss quadrature and our method. Note that the latter two curves are visually indistinguishable. The second and last row present the hierarchical meshes on the computational domain obtained via adaptive refinement by Gauss quadrature and our method, respectively.

Step	0	1	2	3	4
Gauss	7.3226779e-2	2.1718522e-2	1.0762212e-3	1.0607612e-4	1.8194260e-05
Ours	7.3226785e-2	2.1718860e-2	1.0762424e-3	1.0607620e-4	1.8194257e-05

Table 5: The L^2 errors of the solution obtained using adaptive refinement combined with assembly via Gauss quadrature and our method, respectively. Here each number is displayed with 8 digits to highlight the slight differences between these two approaches.

accuracy of our method is maintained provided the spline projection is sufficiently accurate. Finally, these theoretical results are verified by several numerical examples.

There are some major problems that are worthy of further investigation. Firstly, since the exposition in current work is restricted to bivariate HB-splines, we will consider the extension of the proposed approach to d-variate ($d > 2$) case. Already for $d = 3$, the problem gets more complicated, since the assembly procedure requires another auxiliary (intermediate) tensor, the construction and efficient evaluation of which would need further study, in particular for generalizing our computational complexity estimates.

Secondly, the present implementation has not been fully optimized, especially for the low-degree splines, see Figure 9. We will explore some options to accelerate the code. Thirdly, the extension to THB-splines [26, 27] is also a possible topic for future work, which is especially challenging since the factorization into univariate integrals (as used in this paper) is not possible. Finally, for large-scale problems, the solution of the linear system tends to dominate the total computational cost and matrix-free techniques [44] might be more efficient. Thus we will consider the extension of the present approach to matrix-free applications.

Acknowledgement

This work was supported by the Austrian Science Fund through the project "Geometry + Simulation" (NFN S11708), and by the European Research Council via the project "CHANGE" (694515). Maodong Pan was also supported by the Natural Science Foundation of China (No. 61972368) and the USTC Research Funds of the Double First-Class Initiative (No. YD0010002003). These supports are gratefully acknowledged.

References

[1] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli. Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization. *Computer Methods in Applied Mechanics and Engineering*, 285:817–828, 2015.

[2] F. Auricchio, F. Calabrò, T.J.R. Hughes, A. Reali, and G. Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 249:15–27, 2012.

[3] F. Auricchio, L. Beirão da Veiga, T.J.R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107, 2010.

[4] M. Bartoň and V.M. Calo. Gaussian quadrature for splines via homotopy continuation: rules for C^2 cubic splines. *Journal of Computational and Applied Mathematics*, 296:709–723, 2016.

[5] M. Bartoň and V.M. Calo. Optimal quadrature rules for odd-degree spline spaces and their application to tensor-product-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 305:217–240, 2016.

[6] M. Bartoň and V.M. Calo. Gauss–Galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis. *Computer-Aided Design*, 82:57–67, 2017.

[7] Y. Bazilevs, L. Beirão da Veiga, J.A. Cottrell, T.J.R. Hughes, and G. Sangalli. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences*, 16(07):1031–1090, 2006.

[8] A. Bressan and S. Takacs. Sum factorization techniques in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 352:437–460, 2019.

[9] F. Buchegger and B. Jüttler. Planar multi-patch domain parameterization via patch adjacency graphs. *Computer-Aided Design*, 82:2–12, 2017.

[10] A. Buffa and C. Giannelli. Adaptive isogeometric methods with hierarchical splines: error estimator and convergence. *Mathematical Models and Methods in Applied Sciences*, 26(01):1–25, 2016.

[11] A. Buffa, C. Giannelli, P. Morgenstern, and D. Peterseim. Complexity of hierarchical refinement for a class of admissible mesh configurations. *Computer Aided Geometric Design*, 47:83–92, 2016.

[12] A. Buffa, G. Sangalli, and R. Vázquez. Isogeometric methods for computational electromagnetics: B-spline and T-spline discretizations. *Journal of Computational Physics*, 257:1291–1320, 2014.

[13] F. Calabrò, A. Falini, M.L. Sampoli, and A. Sestini. Efficient quadrature rules based on spline quasi-interpolation for application to IGA-BEMs. *Journal of Computational and Applied Mathematics*, 338:153–167, 2018.

[14] F. Calabrò, G. Loli, G. Sangalli, and M. Tani. Quadrature Rules in the Isogeometric Galerkin Method: State of the Art and an Introduction to Weighted Quadrature. In *Advanced Methods for Geometric Modeling and Numerical Simulation*, pages 43–55. Springer, 2019.

[15] F. Calabrò, G. Sangalli, and M. Tani. Fast formation of isogeometric Galerkin matrices by weighted quadrature. *Computer Methods in Applied Mechanics and Engineering*, 316:606–622, 2017.

[16] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[17] L. Beirão da Veiga, A. Buffa, C. Lovadina, M. Martinelli, and G. Sangalli. An isogeometric method for the Reissner–Mindlin plate bending problem. *Computer Methods in Applied Mechanics and Engineering*, 209:45–53, 2012.

[18] L. Beirão da Veiga, A. Buffa, J. Rivas, and G. Sangalli. Some estimates for h–p–k-refinement in isogeometric analysis. *Numerische Mathematik*, 118(2):271–305, 2011.

[19] L. Beirão da Veiga, A. Buffa, G. Sangalli, and R. Vázquez. Mathematical analysis of variational isogeometric methods. *Acta Numerica*, 23:157–287, 2014.

[20] D. Drzisga, B. Keith, and B. Wohlmuth. The surrogate matrix methodology: Low-cost assembly for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 361:112776, 2020.

[21] T. Elguedj, Y. Bazilevs, V.M. Calo, and T.J.R. Hughes. $\bar{B}$ and $\bar{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2732–2762, 2008.

[22] J.A. Evans and T.J.R. Hughes. Discrete spectrum analyses for various mixed discretizations of the Stokes eigenproblem. *Computational Mechanics*, 50(6):667–674, 2012.

[23] A. Falini, C. Giannelli, T. Kanduč, M.L. Sampoli, and A. Sestini. An adaptive IgA-BEM with hierarchical B-splines based on quasi-interpolation quadrature schemes. *International Journal for Numerical Methods in Engineering*, 117(10):1038–1058, 2019.

[24] A. Falini and B. Jüttler. THB-splines multi-patch parameterization for multiply-connected planar domains via Template Segmentation. *Journal of Computational and Applied Mathematics*, 349:390–402, 2019.

[25] A. Falini and T. Kanduč. A study on spline quasi-interpolation based quadrature rules for the isogeometric Galerkin BEM. In *Advanced Methods for Geometric Modeling and Numerical Simulation*, pages 99–125. Springer, 2019.

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

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

[28] C. Giannelli, T. Kanduč, M. Martinelli, G. Sangalli, and M. Tani. Efficient matrix assembly for hierarchical B-splines. Talk at VII International Conference on Isogeometric Analysis, 2019.

[29] A. Giust, B. Jüttler, and A. Mantzaflaris. Local (T)HB-spline projectors via restricted hierarchical spline fitting. *Computer Aided Geometric Design*, page 101865, 2020.

[30] H. Gomez and L. De Lorenzis. The variational collocation method. *Computer Methods in Applied Mechanics and Engineering*, 309:152–181, 2016.

[31] R.R. Hiemstra, F. Calabrò, D. Schillinger, and T.J.R. Hughes. Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:966–1004, 2017.

[32] R.R. Hiemstra, G. Sangalli, M. Tani, F. Calabrò, and T.J.R. Hughes. Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity. *Computer Methods in Applied Mechanics and Engineering*, 355:234–260, 2019.

[33] C. Hofreither. A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 333:311–330, 2018.

[34] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005.

[35] T.J.R. Hughes, A. Reali, and G. Sangalli. Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS. *Computer Methods in Applied Mechanics and Engineering*, 197(49-50):4104–4124, 2008.

[36] T.J.R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):301–313, 2010.

[37] B. Jüttler, U. Langer, A. Mantzaflaris, S.E. Moore, and W. Zulehner. Geometry + Simulation modules: Implementing isogeometric analysis. *Proceedings in Applied Mathematics and Mechanics*, 14(1):961–962, 2014.

[38] R. Kraft. *Adaptive und linear unabhängige Multilevel B-Splines und ihre Anwendungen*. PhD thesis, Universität Stuttgart, 1998.

[39] A. Mantzaflaris and B. Jüttler. Integration by interpolation and look-up for Galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:373–400, 2015.

[40] A. Mantzaflaris, B. Jüttler, B.N. Khoromskij, and U. Langer. Low rank tensor methods in Galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:1062–1085, 2017.

[41] V.P. Nguyen, C. Anitescu, S.P.A. Bordas, and T. Rabczuk. Isogeometric analysis: an overview and computer implementation aspects. *Mathematics and Computers in Simulation*, 117:89–116, 2015.

[42] M. Pan, B. Jüttler, and A. Giust. Fast formation of isogeometric Galerkin matrices via integration by interpolation and look-up. *Computer Methods in Applied Mechanics and Engineering*, 366:113005, 2020.

[43] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 199(29-32):2059–2071, 2010.

[44] G. Sangalli and M. Tani. Matrix-free weighted quadrature for a computationally efficient isogeometric k-method. *Computer Methods in Applied Mechanics and Engineering*, 338:117–133, 2018.

[45] D. Schillinger, J.A. Evans, A. Reali, M.A. Scott, and T.J.R. Hughes. Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Computer Methods in Applied Mechanics and Engineering*, 267:170–232, 2013.

[46] D. Schillinger, S.J. Hossain, and T.J.R. Hughes. Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*,

277:1–45, 2014.

[47] F. Scholz, A. Mantzaflaris, and B. Jüttler. Partial tensor decomposition for decoupling isogeometric Galerkin discretizations. *Computer Methods in Applied Mechanics and Engineering*, 336:485–506, 2018.

[48] H. Speleers. Hierarchical spline spaces: quasi-interpolants and local approximation estimates. *Advances in Computational Mathematics*, 43(2):235–255, 2017.

[49] H. Speleers and C. Manni. Effortless quasi-interpolation in hierarchical spaces. *Numerische Mathematik*, 132(1):155–184, 2016.

[50] 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.