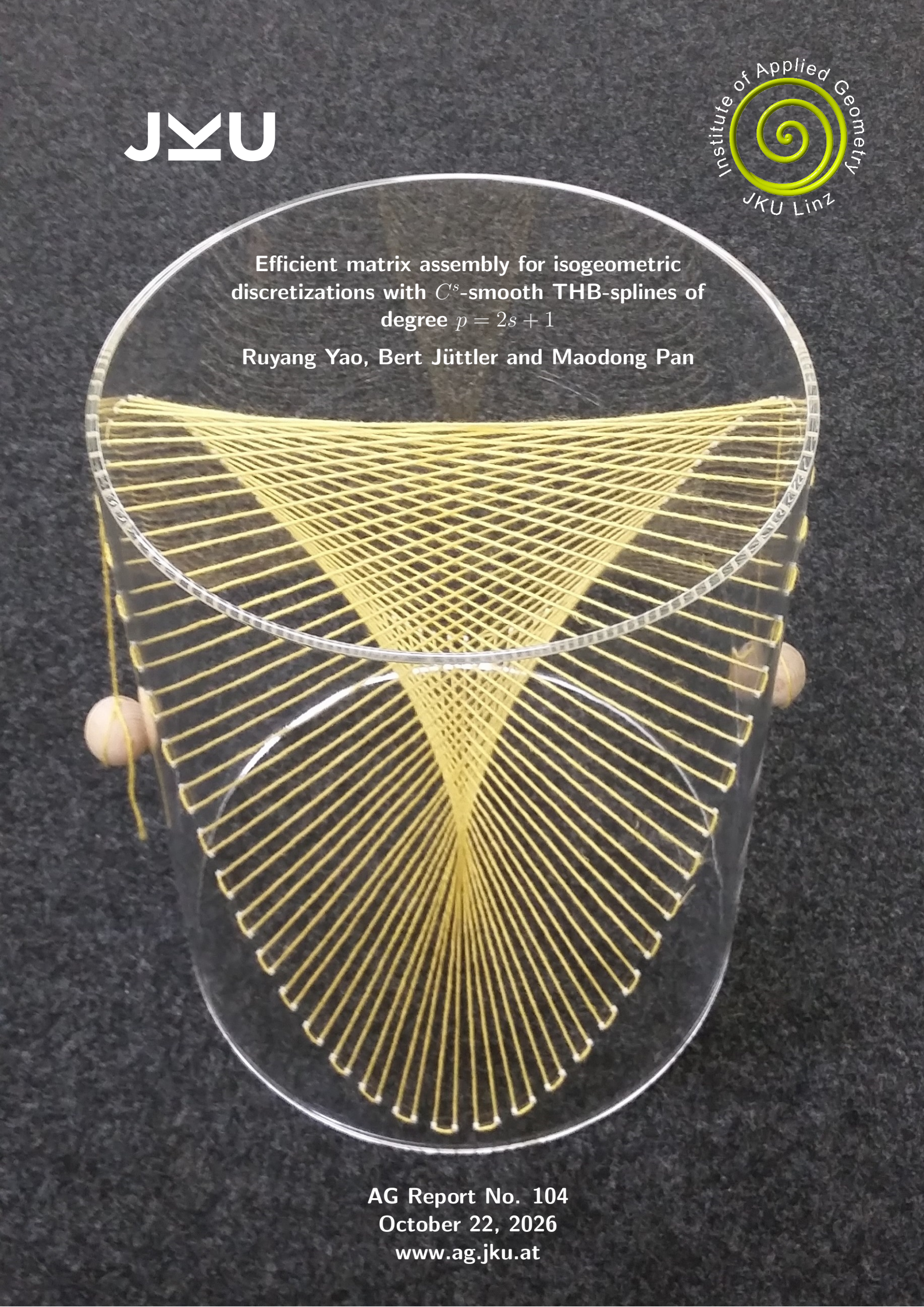


Efficient matrix assembly for isogeometric discretizations with C^s-smooth THB-splines of degree $p = 2s + 1$

Ruyang Yao, Bert Jüttler and Maodong Pan

AG Report No. 104
October 22, 2026
www.ag.jku.at

Efficient matrix assembly for isogeometric discretizations with C^s-smooth THB-splines of degree $p = 2s + 1$

Ruyang Yao[a], Bert Jüttler[b], Maodong Pan[a,*]

[a]*School of Mathematics, Nanjing University of Aeronautics and Astronautics, Nanjing, China*
[b]*Institute of Applied Geometry, Johannes Kepler University Linz, Austria*

Abstract

The efficient assembly of system matrices for THB-spline discretizations is challenging due to truncation, which breaks the tensor-product structure of basis functions and prevents the direct use of sum-factorization. We address this problem for C^s-smooth THB-splines of degree $p = 2s + 1$ by decomposing each truncated basis function on an active cell into a tensor-product B-spline component minus a correction term that is itself separable. The order of smoothness is chosen to be less than maximal ($\frac{p-1}{2}$ rather than $p - 1$) to enable sum-factorization, which significantly speeds up the matrix assembly process, especially for higher polynomial degrees.

The correction terms are organized into systems that have a proven tensor-product structure. This structure enables the sum-factorized evaluation of all resulting quantities. For a d-dimensional problem, the proposed assembly algorithm achieves a computational complexity of $\mathcal{O}(p^{2d+1})$ per element, compared to $\mathcal{O}(p^{3d})$ for standard Gaussian quadrature. Numerical experiments in three dimensions confirm the theoretical complexity analysis and demonstrate substantial speedups over Gaussian quadrature—approximately by a factor of 10 for degree $p = 3$—with the improvement becoming more pronounced as the spline degree increases.

Keywords: isogeometric analysis, matrix assembly, THB-splines, sum-factorization, computational complexity, Gaussian quadrature

1. Introduction

Isogeometric analysis (IgA) has established itself as a powerful paradigm for numerical simulation, enabling the seamless integration of computer-aided design and finite element analysis. This is achieved by employing spline-based discretizations that exactly represent the geometry [10, 22]. Among the various spline technologies, truncated hierarchical B-splines (THB-splines), introduced by Giannelli et al. [14], have attracted increasing attention because they support local refinement while preserving fundamental properties such as

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

linear independence and partition of unity. These features make THB-splines particularly attractive for adaptive isogeometric simulations [9, 13, 41].

A major challenge in large-scale IgA simulations is efficiently assembling system matrices, such as mass and stiffness matrices [20, 27, 33]. Significant progress has been made for tensor-product B-spline discretizations. Relevant techniques include *specialized or reduced quadrature rules* [2, 4, 5, 12, 17, 18, 21, 23, 24, 40, 42], *quadrature-free methods* [19, 31, 32, 35, 38], *quadrature methods employing sum-factorization techniques* [1, 6, 8, 35] and *isogeometric collocation approaches* [3, 16, 29, 30, 34, 39].

For hierarchical B-spline (HB-spline) discretizations, Pan et al. [36, 37] extended sum-factorization by combining quasi-interpolation with precomputed look-up tables. This approach achieves efficient assembly under mild mesh admissibility conditions. More recently, Giannelli et al. [15] proposed a weighted quadrature method for HB-splines that reduces the number of quadrature points by exploiting the hierarchical structure. However, these approaches rely on properties specific to the HB-spline basis and do not directly apply to THB-splines. The reason is that truncation restores the partition of unity but destroys the tensor product structure of the basis functions on the active cells. To the best of our knowledge, few existing works have focused on efficient matrix assembly for THB-splines.

In this paper, we address the aforementioned gap by developing a novel matrix assembly algorithm for a specific yet practically relevant class of THB splines, namely, those of degree $p = 2s + 1$ with C^s smoothness across cell interfaces. The order of smoothness is chosen to be less than maximal ($\frac{p-1}{2}$ rather than $p-1$) to enable sum-factorization, which significantly speeds up the matrix assembly process, especially for higher polynomial degrees.

The first key idea is to represent each truncated basis function on an active cell as a tensor-product B-spline minus a correction term that also has a tensor-product structure. The second key idea is grouping the THB-splines into THB-systems with closely related correction terms. We formalize this combination of decomposition and grouping using the so-called C-systems (correcting systems) and prove their cell-wise separability.

Based on this decomposition, we break the cell-wise matrix contribution into a small number of terms, each of which can be efficiently evaluated via sum-factorization. The computational complexity of the proposed algorithm is $\mathcal{O}(p^{2d+1})$ per element, compared to $\mathcal{O}(p^{3d})$ for standard Gaussian quadrature.

The remainder of this paper is organized as follows: In Section 2 we formulate the mass matrix assembly problem. Section 3 recalls the construction of C^s-smooth THB-splines of degree $p = 2s + 1$, which form the basis of our discretization. Section 4 analyzes the restriction of THB-splines to an active cell and introduces the *C-system* decomposition, a key ingredient for enabling sum-factorization. Section 5 presents the complete assembly algorithm, including auxiliary matrices, decomposition of cell-wise matrix entries, and sum-factorization. Section 6 provides numerical experiments that demonstrate the efficiency and accuracy of the proposed method, followed by concluding remarks in Section 7.

2. Problem formulation: The mass matrix

IGA uses the same spline basis functions to describe both the geometry and the relevant physical quantities, ensuring an exact geometric representation. The latter is based on a d-dimensional geometry mapping

$$\mathbf{G}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in\mathcal{I}} \mathbf{c}_\mathbf{i} T_\mathbf{i}(\boldsymbol{\xi}),$$

which is specified by spline basis functions $T_\mathbf{i}$ with associated control points $\mathbf{c}_\mathbf{i}$, where $\mathcal{I}$ is the index set. The mapping transforms the parameter domain, e.g., $\hat{\Omega} = [0,1]^d$, into the physical domain Ω.

The spline basis functions are defined via an adaptively refined box mesh with a minimum cell size of h that covers the parameter domain $\hat{\Omega}$. The resulting isogeometric discretization space,

$$V_h = \operatorname{span}\{T_\mathbf{i} \circ \mathbf{G}^{-1} \ : \ \mathbf{i} \in \mathcal{I}\} \ ,$$

is used to find numerical approximations of the physical fields required for the specific application.

The implementation of Galerkin-based isogeometric discretization of partial differential equations involves assembling various system matrices. The mass and stiffness matrices are the most fundamental among these. To keep the presentation concise, we will focus on the entries of the mass matrix

$$m_{\mathbf{i},\mathbf{j}} = \int_{[0,1]^d} T_\mathbf{i}(\boldsymbol{\xi})\, T_\mathbf{j}(\boldsymbol{\xi})\, W(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} \ , \tag{1}$$

where we use

$$W(\boldsymbol{\xi}) = |\det \nabla \mathbf{G}(\boldsymbol{\xi})|$$

to denote the absolute determinant of the Jacobian of the geometry mapping, which accounts for the change of variables between the physical and parametric domains. Methods similar to those used for the mass matrix can be applied to other matrices and vectors, such as the stiffness matrix and load vector. See, e.g., [36] for details.

To efficiently assemble the global mass matrix, a standard, cell-based quadrature approach is adopted. The hierarchical mesh partitions the parameter domain into quadrature cells, which we denote by c.

The contribution of each cell to the mass matrix is given by

$$m^c_{\mathbf{i},\mathbf{j}} = \int_c T_\mathbf{i}(\boldsymbol{\xi})\, T_\mathbf{j}(\boldsymbol{\xi})\, W(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi}.$$

As is standard in numerical simulations, we approximate these integrals via Gaussian quadrature,

$$m^c_{\mathbf{i},\mathbf{j}} \approx \tilde{m}^c_{\mathbf{i},\mathbf{j}} = \sum_\mathbf{q} \omega_\mathbf{q}\, T_\mathbf{i}(\boldsymbol{\xi}^\mathbf{c}_\mathbf{q})\, T_\mathbf{j}(\boldsymbol{\xi}^\mathbf{c}_\mathbf{q})\, W(\boldsymbol{\xi}^\mathbf{c}_\mathbf{q}) \ , \tag{2}$$

using quadrature weights $\omega_{\mathbf{q}}$ and Gauss points $\boldsymbol{\xi}_{\mathbf{q}}^{c}$ associated with the cell c. The approximation of the full mass matrix is then assembled by accumulating contributions from all active cells, the set of which is denoted by C,

$$\tilde{m}_{\mathbf{i},\mathbf{j}} = \sum_{c \in C} \tilde{m}_{\mathbf{i},\mathbf{j}}^{c} \ .$$

This assembly procedure is conceptually straightforward. However, its computational cost becomes substantial for high-degree splines and adaptively refined meshes when using a naive implementation. This is primarily due to the large number of non-zero basis function interactions and the repeated evaluation of basis functions at each quadrature point.

Sum-factorization has been used to speed up the process; however, it is not directly applicable to adaptively refined spline discretizations. The remainder of this paper presents a method to adapt it to a specific class of THB-splines.

3. C^s-smooth THB-splines of degree $p = 2s + 1$

In this section, we recall the construction of truncated hierarchical B-splines (THB-splines). Specifically, we consider splines of degree $p = 2s + 1$ that are C^s continuous across cell interfaces. These splines are defined over an adaptively refined box mesh in the parameter domain $\hat{\Omega} = [0, 1]^d$.

3.1. The spline hierarchy

Let $\ell = 0, 1, \ldots, L$ denote the refinement level. The cells of level ℓ are the elements of the set

$$C^{\ell} = \{2^{-\ell}\big(\mathbf{k} + (0, 1)^d\big) \ : \ \mathbf{k} \in \mathbb{Z}^d\},$$

and the corresponding grid lines are defined as

$$\Gamma^{\ell} = \bigcup_{c \in C^{\ell}} \partial c \ .$$

For each level, we consider the tensor-product B-spline space consisting of piecewise polynomial functions of degree $2s + 1$ on each cell of C^{ℓ} that are $\mathcal{C}^s$ smooth.

We denote the univariate B-spline of degree $p = 2s + 1$ associated with the $p + 2$ non-decreasing knots Ξ, which form a local knot vector, by $N_{\Xi}^{p}(\xi)$. The B-spline depends on a single variable, which we denote by ξ.

Taking the smoothness into account, we consider local knot vectors of the form

$$\Xi_{i,u} = (\ \underbrace{i-1}_{s+1-u \text{ times}}\ ,\ \underbrace{i}_{s+1 \text{ times}}\ ,\ \underbrace{i+1}_{u+1 \text{ times}}\ )\ , i, u \in \mathbb{Z}\ ,$$

which have repeated knots. Each local knot vector $\Xi_{i,u}$ defines a B-spline, denoted by

$$\beta_{i,u}(\xi) = N_{\Xi_{i,u}}^{p}(\xi) \ .$$

For each integer i, we generate the *B-spline vector*

$$[\beta_{i,u}(\xi)]_{u=0,\ldots,s} \ .$$

This yields a collection of B-splines that are locally supported on the same interval $(i-1, i+1)$ and C^s-smooth.

Given a multi-index $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{Z}^d$, the tensor product of the univariate B-spline vectors gives the *B-systems*

$$\mathbf{B}_{\mathbf{i}}(\boldsymbol{\xi}) = \bigotimes_{k=1}^{d} [\beta_{i_k,u}(\xi^k)]_{u=0,\ldots,s}$$

that contain multivariate B-splines. These functions depend on d variables, which we collect into the vector $\boldsymbol{\xi} = (\xi^1, \ldots, \xi^d)$. The B-systems are tensors

$$\mathbf{B}_{\mathbf{i}}(\boldsymbol{\xi}) = \left[B_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi})\right]_{\mathbf{u}\in\{0,\ldots,s\}^d}$$

whose elements are products of univariate B-splines

$$B_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \beta_{i_k,u_k}(\xi^k) \ . \tag{3}$$

The multi-index $\mathbf{u} = (u_1, \ldots, u_d) \in \{0, \ldots, s\}^d$ is used to identify the tensor elements.

So far we have created B-systems for B-splines with integer knots, which are now declared to be the B-systems of level 0. The B-systems for higher levels ℓ are obtained by dyadic dilations,

$$\mathbf{B}^{\ell}_{\mathbf{i}}(\boldsymbol{\xi}) = \mathbf{B}_{\mathbf{i}}(2^{\ell}\boldsymbol{\xi}) \ ,$$

and they form a basis for a tensor-product spline space. Again, these B-systems are tensors

$$\mathbf{B}^{\ell}_{\mathbf{i}}(\boldsymbol{\xi}) = \left[B^{\ell}_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi})\right]_{\mathbf{u}\in\{0,\ldots,s\}^d}$$

whose elements are products of univariate B-splines

$$B^{\ell}_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \beta^{\ell}_{i_k,u_k}(\xi^k) \tag{4}$$

with

$$\beta^{\ell}_{i,u}(\xi) = \beta_{i,u}(2^{\ell}\xi) \ .$$

To keep the presentation simple, we now consider splines on the unbounded domain $\mathbb{R}^d$. In a subsequent step, we will extract the splines (and their basis) using the restriction to the compact domain $\hat{\Omega}$.

3.2. THB-splines

Recall that a regular open set is equal to the interior of its closure. The hierarchical mesh is defined by a nested sequence of subdomains

$$\mathbb{R}^d = \Omega^0 \supseteq \Omega^1 \supseteq \Omega^2 \supseteq \cdots \supseteq \Omega^L \ ,$$

which are regular open sets. It is assumed that the closure $\bar{\Omega}^\ell$ of each subdomain consists of the closure of some set of cells of level $\ell - 1$. Equivalently, its boundary

$$\partial\Omega^\ell \subset \Gamma^{\ell-1}$$

is contained in the grid of level $\ell - 1$.

We create the hierarchical spline basis by applying Kraft's selection procedure [28] to B-systems. The result is described with the help of index sets

$$\mathcal{S}^\ell = \{\mathbf{i} \in \mathbb{Z}^d : \bar{\Omega}^\ell \supseteq \operatorname{supp} \mathbf{B}_\mathbf{i}^\ell(\boldsymbol{\xi}) \not\subseteq \bar{\Omega}^{\ell+1}\} \ .$$

The hierarchical spline basis is found by collecting the selected functions,

$$\bigcup_{\ell=0}^{L} \{\mathbf{B}_\mathbf{i}^\ell(\boldsymbol{\xi}) : \mathbf{i} \in \mathcal{S}^\ell\}.$$

We apply the truncation mechanism of Giannelli et al. [14] to secure the partition of unity property, which yields the THB-systems

$$\bigcup_{\ell=0}^{L} \{\mathbf{T}_\mathbf{i}^\ell(\boldsymbol{\xi}) : \mathbf{i} \in \mathcal{S}^\ell\} \ .$$

More precisely, we denote the tensor containing the truncated hierarchical B-splines obtained from the tensor-product B-splines that make up the B-system $\mathbf{B}_\mathbf{i}^\ell(\boldsymbol{\xi})$ by $\mathbf{T}_\mathbf{i}^\ell(\boldsymbol{\xi})$. The tensor

$$\mathbf{T}_\mathbf{i}^\ell(\boldsymbol{\xi}) = \left[T_{\mathbf{i},\mathbf{u}}^\ell(\boldsymbol{\xi})\right]_{\mathbf{u}\in\{0,\ldots,s\}^d}$$

consists of $(s+1)^d$ functions. However, these functions no longer have a (tensor-) product structure.

When extracting the splines on the compact domain $\hat{\Omega}$ it should be noted that, near the boundary, these functions are compatible with the spline spaces that have been specified by the hierarchy. For instance, even if the entire domain has been refined to the first level by specifying $\Omega^1 = \hat{\Omega}$, the behaviour of the splines near the boundary is still determined by the level-0 basis functions. This issue can easily be addressed by suitably extending the subdomains to include the layer of cells surrounding $\hat{\Omega}$.

4. The THB-splines on an active cell

A cell $c \in C^\ell$ of level ℓ is said to be *active* if $c \subseteq \hat{\Omega}$ and

$$\Omega^\ell \supseteq c \not\subseteq \Omega^{\ell+1} \ .$$

By considering only the active cells, we extract the splines (and their basis) on the compact domain $\hat{\Omega}$.

4.1. Index sets of active THB-splines

We analyze the structure of the THB-splines restricted to an active cell. Clearly, each individual functions is simply a multivariate polynomial of degree p with respect to each variable $\xi^1, \ldots, \xi^d$.

We assume a mild *mesh grading* property, which is typically enforced in hierarchical refinement to ensure stability. More precisely, we consider strictly admissible meshes of class $m = 2$ introduced by Buffa et al. [7], for which the truncated basis functions taking non-zero values on any active cell belong to at most two successive levels. As shown in [7, Theorem 13], such meshes can be generated at linear cost, i.e., the number of elements introduced to restore admissibility remains proportional to the number of marked elements throughout the refinement procedure. Consequently, enforcing the grading condition at each step of the adaptive loop does not become a bottleneck in the assembly pipeline.

Therefore, each active cell c of level ℓ is included in the supports of basis functions from at most two consecutive levels: level $\ell-1$ and level ℓ. This property is crucial for the efficiency of the assembly algorithm because it limits the number of basis functions interactions per cell.

Under the mesh grading assumption, there are 2^d THB-systems taking non-zero values on this cell, with one associated to each vertex. This implies that *the THB-systems possess the property of local linear independence if mesh grading is assumed*. This fact was first noted in [25] for bivariate splines. The property and its proof can be extended to d-variate splines. However, we will not discuss this topic in more detail in order to stay focused on the central topic of the paper: efficient matrix assembly.

Firstly, without loss of generality we consider the cell $c = [0, \frac{1}{2}]^d$ of level $\ell = 1$, and assume that it is active. It is one of the 2^d cells obtained by bisecting the cell $[0, 1]^d$ of level $\ell - 1 = 0$ in every coordinate direction. The vertices of c belong to the set

$$\{\mathbf{i}/2 \ : \ \mathbf{i} \in \{0, 1\}^d\}$$

There are two types of vertices and associated THB-systems:

(i) The vertex $\mathbf{i}/2$ is contained within the interior of the subdomain Ω^1. The associated THB-system restricted to the cell c is then the B-system $\mathbf{B}^1_{\mathbf{i}}(\boldsymbol{\xi})$ of level 1.

(ii) The vertex $\mathbf{i}/2$ is not contained within the interior of the subdomain Ω^1. The THB-system restricted to the cell c is then the truncated B-system $\mathbf{T}^0_{\mathbf{i}}(\boldsymbol{\xi})$ of level 0. It is constructed by truncating the B-system $\mathbf{B}^0_{\mathbf{i}}(\boldsymbol{\xi})$ with respect to all B-systems $\mathbf{B}^1_{\mathbf{i}'}(\boldsymbol{\xi})$ of the type (i).

We collect the relevant multi-indices $\mathbf{i}$ in two index sets, $\mathcal{I}^1_c$ and $\mathcal{I}^0_c$, for the first and second type, respectively. The first set contains at least the multi-index $(1, \ldots, 1)$, while the second set may be empty.

Secondly, we extend these observations to the general setting. Any active cell c of level ℓ is associated with 2^d THB-systems that take non-zero values on it. We denote the index set of the THB-systems of level ℓ' whose supports contain the cell c by $\mathcal{I}^{\ell'}_c$. Due to mesh

grading, we have that $\mathcal{I}_c^{\ell'} = \emptyset$ whenever $\ell' \notin \{\ell-1, \ell\}$. Consequently, the set of the non-zero THB-systems on the cell c, which has cardinality 2^d, is given by

$$\{\mathbf{T}_{\mathbf{i}}^{\ell-1}(\boldsymbol{\xi}) : \mathbf{i} \in \mathcal{I}_c^{\ell-1}\} \cup \{\mathbf{T}_{\mathbf{i}}^{\ell}(\boldsymbol{\xi}) : \mathbf{i} \in \mathcal{I}_c^{\ell}\} \ .$$

4.2. The correcting systems

The next step is to express the THB-systems in terms of underlying B-systems.

First we consider the THB-systems of level ℓ. The truncation does not affect their representation on the active cell, i.e.,

$$\mathbf{T}_{\mathbf{i}}^{\ell}(\boldsymbol{\xi}) = \mathbf{B}_{\mathbf{i}}^{\ell}(\boldsymbol{\xi}) \ , \quad \boldsymbol{\xi} \in c \ , \quad \mathbf{i} \in \mathcal{I}_c^{\ell}$$

Second we analyze the THB-systems of level $\ell-1$. They experience truncation on the active cell, and we represent them as follows:

$$\mathbf{T}_{\mathbf{i}}^{\ell-1}(\boldsymbol{\xi}) = \mathbf{B}_{\mathbf{i}}^{\ell-1}(\boldsymbol{\xi}) - \sum_{\mathbf{i}' \in \mathcal{I}_c^{\ell}} \mathbf{H}_{\mathbf{i},\mathbf{i}'}^{\ell-1}(\boldsymbol{\xi}) \ , \quad \boldsymbol{\xi} \in c, \quad \mathbf{i} \in \mathcal{I}_c^{\ell-1} \ . \tag{5}$$

Here we use *correcting systems (C-systems)*, denoted by

$$\mathbf{H}_{\mathbf{i},\mathbf{i}'}^{\ell-1}(\boldsymbol{\xi}) = \left[H_{\mathbf{i},\mathbf{i}',\mathbf{u}}^{\ell-1}(\boldsymbol{\xi})\right]_{\mathbf{u} \in \{0,\dots,s\}^d} \ ,$$

to encode the contribution of fine-level basis functions on c. For later reference we rewrite Eq. (5) for the tensor elements (i.e., the individual basis functions),

$$T_{\mathbf{i},\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) = B_{\mathbf{i},\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) - \sum_{\mathbf{i}' \in \mathcal{I}_c^{\ell}} H_{\mathbf{i},\mathbf{i}',\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) \ , \quad \boldsymbol{\xi} \in c, \quad \mathbf{i} \in \mathcal{I}_c^{\ell-1} \ .$$

The following observation is essential for the efficiency of the assembly algorithm:

Lemma 1. *The C-systems* $\mathbf{H}_{\mathbf{i},\mathbf{i}'}^{\ell-1}$ *have tensor-product structure,*

$$\mathbf{H}_{\mathbf{i},\mathbf{i}'}^{\ell-1}(\boldsymbol{\xi}) = \bigotimes_{k=1}^{d} [\eta_{i_k,i'_k,u}^{\ell-1}(\xi^k)]_{u=0,\dots,s} \ .$$

More specifically, the tensor elements are products of univariate functions,

$$H_{\mathbf{i},\mathbf{i}',\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \eta_{i_k,i'_k,u_k}^{\ell-1}(\xi^k) \ , \quad \mathbf{u} = (u_1, \dots, u_d) \in \{0, \dots, s\}^d \ . \tag{6}$$

The specific form of the tensor elements is derived in the proof:

Proof. The univariate B-splines satisfy the two-scale relation

$$\beta_{i,u}^{\ell-1}(\xi) = \sum_{i'=2i-1}^{2i+1} \sum_{v=0}^{s} r_{i'-2i}^{u,v} \beta_{i',v}^{\ell}(\xi)$$

p	s	$(r_{-1}^{u,v})_{u,v=0,\dots,s}$	$(r_{0}^{u,v})_{u,v=0,\dots,s}$	$(r_{1}^{u,v})_{u,v=0,\dots,s}$
1	0	$\begin{pmatrix} \frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2} \end{pmatrix}$
3	1	$\begin{pmatrix} \frac{1}{4} & \frac{5}{8} \\ 0 & \frac{1}{8} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{8} & 0 \\ \frac{5}{8} & \frac{1}{4} \end{pmatrix}$
5	2	$\begin{pmatrix} \frac{1}{8} & \frac{7}{16} & \frac{19}{32} \\ 0 & \frac{1}{16} & \frac{1}{4} \\ 0 & 0 & \frac{1}{32} \end{pmatrix}$	$\begin{pmatrix} \frac{9}{16} & \frac{3}{16} & \frac{1}{16} \\ \frac{3}{8} & \frac{5}{8} & \frac{3}{8} \\ \frac{1}{16} & \frac{3}{16} & \frac{9}{16} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{32} & 0 & 0 \\ \frac{1}{4} & \frac{1}{16} & 0 \\ \frac{19}{32} & \frac{7}{16} & \frac{1}{8} \end{pmatrix}$
7	3	$\begin{pmatrix} \frac{1}{16} & \frac{9}{32} & \frac{33}{64} & \frac{65}{128} \\ 0 & \frac{1}{32} & \frac{5}{32} & \frac{43}{128} \\ 0 & 0 & \frac{1}{64} & \frac{11}{128} \\ 0 & 0 & 0 & \frac{1}{128} \end{pmatrix}$	$\begin{pmatrix} \frac{27}{64} & \frac{9}{64} & \frac{3}{64} & \frac{1}{64} \\ \frac{27}{64} & \frac{33}{64} & \frac{19}{64} & \frac{9}{64} \\ \frac{9}{64} & \frac{19}{64} & \frac{33}{64} & \frac{27}{64} \\ \frac{1}{64} & \frac{3}{64} & \frac{9}{64} & \frac{27}{64} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{128} & 0 & 0 & 0 \\ \frac{11}{128} & \frac{1}{64} & 0 & 0 \\ \frac{43}{128} & \frac{5}{32} & \frac{1}{32} & 0 \\ \frac{65}{128} & \frac{33}{64} & \frac{9}{32} & \frac{1}{16} \end{pmatrix}$

Table 1: B-spline refinement coefficients.

with positive refinement coefficients $r_{i'-2i}^{u,v}$, which can be computed using the knot insertion algorithm. Table 1 lists these coefficients for $p \in \{1, 3, 5, 7\}$.

The two-scale relation yields the following result for the product (4) of the univariate B-splines:

$$B_{\mathbf{i},\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \sum_{i'_k=2i_k-1}^{2i_k+1} \sum_{v_k=0}^{s} r_{i'_k-2i_k}^{u_k,v_k} \beta_{i'_k,v_k}^{\ell}(\xi_k) \ .$$

We expand the sum and collect the contributions for each multi-index $\mathbf{i}' = (i'_1, \dots, i'_d)$, resulting in

$$B_{\mathbf{i},\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) = \sum_{\mathbf{2i-1} \le \mathbf{i}' \le \mathbf{2i+1}} \underbrace{\prod_{k=1}^{d} \sum_{v_k=0}^{s} r_{i'_k-2i_k}^{u_k,v_k} \beta_{i'_k,v_k}^{\ell}(\xi_k)}_{=H_{\mathbf{i},\mathbf{i}',\mathbf{u}}^{\ell-1}} \ ,$$

where we define $\mathbf{1} = (1, \dots, 1)$. Each contribution is a C-system if the associated B-system is selected using Kraft's procedure. It possesses the required tensor-product structure (6) with the auxiliary functions

$$\eta_{i,i',u}^{\ell-1}(\xi) = \sum_{v=0}^{s} r_{i'-2i}^{u,v} \beta_{i',v}^{\ell}(\xi) \ .$$

This completes the proof. □

5. The Assembly Algorithm

Standard cell-wise Gaussian quadrature with sum-factorization is highly efficient for tensor-product spline discretizations in isogeometric analysis [1]. However, its direct applicability to truncated hierarchical B-splines (THB-splines) is prevented by the truncation

mechanism. The truncation operator recovers the partition of unity by subtracting fine-level basis functions from coarse-level ones. This breaks the pure tensor-product structure, which blocks conventional sum-factorization.

Recall the cell-wise mass matrix entry

$$\tilde{m}^c_{(\hat{\ell},\mathbf{i},\mathbf{u}),(\bar{\ell},\mathbf{j},\mathbf{v})} = \sum_{\mathbf{q}} T^{\hat{\ell}}_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi}^c_{\mathbf{q}})\, T^{\bar{\ell}}_{\mathbf{j},\mathbf{v}}(\boldsymbol{\xi}^c_{\mathbf{q}})\, \underbrace{\omega_{\mathbf{q}}\, W(\boldsymbol{\xi}^c_{\mathbf{q}})}_{=\nu^c_{\mathbf{q}}} \tag{7}$$

which has been approximated by Gaussian quadrature. Note that the multi-indices $\mathbf{i},\mathbf{j}$ in the previous version (2) have evolved into triplet indices $(\hat{\ell},\mathbf{i},\mathbf{u})$, $(\bar{\ell},\mathbf{j},\mathbf{v})$ that reflect the structure of the truncated hierarchical B-systems. These triplet indices combine information about the level, the vertex associated with the THB-system and the individual THB-spline in that system.

Furthermore, to simplify the presentation, we introduce the quantities

$$\nu^c_{\mathbf{q}} = \omega_{\mathbf{q}} |\det \nabla \mathbf{G}(\boldsymbol{\xi}^c_{\mathbf{q}})| \ ,$$

which combine the effects of the Jacobian determinant and the quadrature weights.

According to the mesh grading assumption, an active cell c of level ℓ only belongs to the supports of basis functions from levels $\ell-1$ and ℓ. Thus, the levels $\hat{\ell}$, $\bar{\ell}$ of the THB-splines can only take two possible values. Without loss of generality, we assume that the levels fulfill the ordering condition

$$\ell \geq \hat{\ell} \geq \bar{\ell} \ .$$

Consequently, there are three possible combinations of levels,

$$(\hat{\ell},\bar{\ell}) \in \{(\ell-1,\ell-1),(\ell,\ell-1),(\ell,\ell)\} \ .$$

5.1. Auxiliary matrices

For a given cell c of level ℓ we introduce five families of auxiliary matrices, which are organized in three groups: The individual matrices have dimension $(s+1)^d \times (s+1)^d$ and contain elements identified by pairs $(\mathbf{u},\mathbf{v})$ of multi-indices $\mathbf{u},\mathbf{v} \in \{0,\dots,s\}^d$, both enumerated lexicographically.

- The first group, which covers the interactions of B-systems, consists of three families

$$\begin{aligned}
\Upsilon^{\ell-1,\ell-1}_{\mathbf{i},\mathbf{j}} &= (\upsilon^{\ell-1,\ell-1}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}})_{\mathbf{u},\mathbf{v}\in\{0,\dots,s\}^d} \ , \quad \mathbf{i},\mathbf{j} \in \mathcal{I}^{\ell-1}_c \ , \\
\Upsilon^{\ell,\ell-1}_{\mathbf{i},\mathbf{j}} &= (\upsilon^{\ell,\ell-1}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}})_{\mathbf{u},\mathbf{v}\in\{0,\dots,s\}^d} \ , \quad \mathbf{i} \in \mathcal{I}^{\ell}_c,\ \mathbf{j} \in \mathcal{I}^{\ell-1}_c \ , \\
\Upsilon^{\ell,\ell}_{\mathbf{i},\mathbf{j}} &= (\upsilon^{\ell,\ell}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}})_{\mathbf{u},\mathbf{v}\in\{0,\dots,s\}^d} \ , \quad \mathbf{i},\mathbf{j} \in \mathcal{I}^{\ell}_c \ .
\end{aligned}$$

  The matrix entries are defined as

$$\upsilon^{\hat{\ell},\bar{\ell}}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}} = \sum_{\mathbf{q}} B^{\hat{\ell}}_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi}^c_{\mathbf{q}}) B^{\bar{\ell}}_{\mathbf{j},\mathbf{v}}(\boldsymbol{\xi}^c_{\mathbf{q}})\, \nu^c_{\mathbf{q}} \ . \tag{8}$$

dimension:		$d=2$				$d=3$							
index sets:	$\lvert\mathcal{I}_c^{\ell-1}\rvert$	0	1	2	3	0	1	2	3	4	5	6	7
	$\lvert\mathcal{I}_c^{\ell}\rvert$	4	3	2	1	8	7	6	5	4	3	2	1
first group:	$\Upsilon_{\mathbf{i},\mathbf{j}}^{\ell-1,\ell-1}$	0	1	4	9	0	1	4	9	16	25	36	49
	$\Upsilon_{\mathbf{i},\mathbf{j}}^{\ell,\ell-1}$	0	3	4	3	0	7	12	15	16	15	12	7
	$\Upsilon_{\mathbf{i},\mathbf{j}}^{\ell,\ell}$	16	9	4	1	64	49	36	25	16	9	4	1
second group:	$\Phi_{\mathbf{i},\mathbf{j}}^{\ell-1}$	0	1	4	9	0	1	4	9	16	25	36	49
	$\Phi_{\mathbf{i},\mathbf{j}}^{\ell}$	0	3	4	3	0	7	12	15	16	15	12	7
third group:	$\Psi_{\mathbf{i},\mathbf{j}}$	0	1	4	9	0	1	4	9	16	25	36	49
total number of matrices:		16	18	24	34	64	66	72	82	96	114	136	162

Table 2: Number of auxiliary matrices.

- The second group, which covers the interactions of B-systems and C-systems, consists of two families

$$\begin{aligned}\Phi_{\mathbf{i},\mathbf{j}}^{\ell-1} &= (\phi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}^{\ell-1})_{\mathbf{u},\mathbf{v}\in\{0,\dots,s\}^d}\ , \quad \mathbf{i},\mathbf{j}\in\mathcal{I}_c^{\ell-1}\ ,\\ \Phi_{\mathbf{i},\mathbf{j}}^{\ell} &= (\phi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}^{\ell})_{\mathbf{u},\mathbf{v}\in\{0,\dots,s\}^d}\ , \quad \mathbf{i}\in\mathcal{I}_c^{\ell},\ \mathbf{j}\in\mathcal{I}_c^{\ell-1}\ .\end{aligned}$$

  The matrix entries are defined as

$$\phi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}^{\hat{\ell}} = \sum_{\mathbf{j}'\in\mathcal{I}_c^{\ell}}\sum_{\mathbf{q}} B_{\mathbf{i},\mathbf{u}}^{\hat{\ell}}(\boldsymbol{\xi}_{\mathbf{q}}^c) H_{\mathbf{j},\mathbf{j}',\mathbf{v}}^{\ell-1}(\boldsymbol{\xi}_{\mathbf{q}}^c)\,\nu_{\mathbf{q}}^c\ . \tag{9}$$

- The third group, which covers the interactions of C-systems, consists of only one family

$$\Psi_{\mathbf{i},\mathbf{j}} = (\psi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}})_{\mathbf{u},\mathbf{v}\in\{0,\dots,s\}^d}\ , \quad \mathbf{i},\mathbf{j}\in\mathcal{I}_c^{\ell-1}.$$

  The matrix entries are defined as

$$\psi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}} = \sum_{\mathbf{i}'\in\mathcal{I}_c^{\ell}}\sum_{\mathbf{j}'\in\mathcal{I}_c^{\ell}}\sum_{\mathbf{q}} H_{\mathbf{i},\mathbf{i}',\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}_{\mathbf{q}}^c) H_{\mathbf{j},\mathbf{j}',\mathbf{v}}^{\ell-1}(\boldsymbol{\xi}_{\mathbf{q}}^c)\,\nu_{\mathbf{q}}^c\ . \tag{10}$$

The size of the index sets $\mathcal{I}_c^{\ell-1}$ and $\mathcal{I}_c^{\ell}$ is bounded by 2^d. Consequently, the total number of matrices is bounded by

$$6\cdot 2^d\cdot 2^d,$$

independently of the degree p. More precise bounds can be found by noting that the cardinalities of the two index sets sum to 2^d. We present a detailed analysis for dimensions $d=2$ and $d=3$ in Table 2. The total number of matrices does not exceed 34 and 162 for dimensions $d=2$ and $d=3$, respectively, in this case, significantly less than the upper bounds of $6\cdot 2^2\cdot 2^2 = 96$ and $6\cdot 2^3\cdot 2^3 = 384$.

Lemma 2. *The assembly of all the auxiliary matrices $\Upsilon_{\mathbf{i},\mathbf{j}}, \Phi_{\mathbf{i},\mathbf{j}}, \Psi_{\mathbf{i},\mathbf{j}}$ using sum-factorization requires $\mathcal{O}(p^{2d+1})$ flops.*

Proof. We need to assemble $\mathcal{O}(1)$ auxiliary matrices, each of which requires $\mathcal{O}(p^{2d+1})$ flops to compute. The assembly of these matrices is described in Appendices A-D for dimension $d = 3$. Other dimensions can be dealt with analogously. □

Remark 1. In our implementation, the auxiliary matrices $\Upsilon_{\mathbf{i},\mathbf{j}}, \Phi_{\mathbf{i},\mathbf{j}}, \Psi_{\mathbf{i},\mathbf{j}}$ are computed on-the-fly for each cell within the cell loop. They cannot be reused, even for cells with identical level configurations and local refinement patterns, since all their entries involve the weights $\nu_{\mathbf{q}}^c = \omega_{\mathbf{q}} |\det \nabla \mathbf{G}(\boldsymbol{\xi}_{\mathbf{q}}^c)|$, which depend on the specific geometry mapping $\mathbf{G}$ evaluated at the quadrature points of the cell. Nevertheless, since the number of auxiliary matrices per cell is constant (independent of p), this does not affect the asymptotic complexity stated in Theorem 1. ◇

5.2. Decomposition of the cell-wise mass matrix entry

We decompose the matrix entry (7) for the three resulting cases, based on the THB-spline representations

$$T_{\mathbf{i},\mathbf{u}}^{\ell}(\boldsymbol{\xi}) = B_{\mathbf{i},\mathbf{u}}^{\ell}(\boldsymbol{\xi})$$

and

$$T_{\mathbf{i},\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) = B_{\mathbf{i},\mathbf{u}}^{\ell-1}(\boldsymbol{\xi}) - \sum_{\mathbf{i}' \in \mathcal{I}_c^{\ell}} H_{\mathbf{i},\mathbf{i}',\mathbf{u}}^{\ell-1}(\boldsymbol{\xi})$$

on the active cell c. These representations reflect the fact that truncation is not necessary for the finer level ℓ, but is needed for the coarser level $\ell - 1$.

Case 1: $\ell = \hat{\ell} = \bar{\ell}$. For THB-splines on the same level as the active cell, the cell-wise contribution to the matrix entry is

$$\begin{aligned} \tilde{m}^c_{(\hat{\ell},\mathbf{i},\mathbf{u}),(\bar{\ell},\mathbf{j},\mathbf{v})} &= \sum_{\mathbf{q}} B_{\mathbf{i},\mathbf{u}}^{\ell}(\boldsymbol{\xi}_{\mathbf{q}}^c) B_{\mathbf{j},\mathbf{v}}^{\ell}(\boldsymbol{\xi}_{\mathbf{q}}^c)\, \nu_{\mathbf{q}}^c \\ &= \upsilon_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}^{\ell,\ell} \ . \end{aligned} \tag{11}$$

No further computation is necessary. We just copy the values from the auxiliary matrix.

Case 2: $\ell = \hat{\ell} > \bar{\ell} = \ell - 1$. One of the THB-splines is at a coarser (i.e., lower) level than the active cell. Expanding the truncated coarse-level THB-spline $T_{\mathbf{j},\mathbf{v}}^{\ell-1}(\boldsymbol{\xi})$ yields the decomposition

$$\begin{aligned} \tilde{m}^c_{(\hat{\ell},\mathbf{i},\mathbf{u}),(\bar{\ell},\mathbf{j},\mathbf{v})} &= \sum_{\mathbf{q}} B_{\mathbf{i},\mathbf{u}}^{\ell}(\boldsymbol{\xi}_{\mathbf{q}}^c) \left(B_{\mathbf{j},\mathbf{v}}^{\ell-1}(\boldsymbol{\xi}_{\mathbf{q}}^c) - \sum_{\mathbf{j}' \in \mathcal{I}_c^{\ell}} H_{\mathbf{j},\mathbf{j}',\mathbf{v}}^{\ell-1}(\boldsymbol{\xi}_{\mathbf{q}}^c) \right) \nu_{\mathbf{q}}^c \\ &= \upsilon_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}^{\ell,\ell-1} - \phi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}^{\ell} \end{aligned}$$

of the cell-wise mass matrix into a tensor-product B-spline term and a correction term involving C-systems $H_{\mathbf{j},\mathbf{j}',\mathbf{v}}^{\ell-1}(\boldsymbol{\xi})$.

Case 3: $\ell > \hat{\ell} = \bar{\ell} = \ell - 1$. Both THB-splines are at a coarser (i.e., lower) level than the active cell. Expanding them yields the decomposition

$$\begin{aligned}\tilde{m}^c_{(\hat{\ell},\mathbf{i},\mathbf{u}),(\bar{\ell},\mathbf{j},\mathbf{v})} &= \sum_{\mathbf{q}} \left(B^{\ell-1}_{\mathbf{i},\mathbf{u}}(\boldsymbol{\xi}^c_{\mathbf{q}}) - \sum_{\mathbf{i}' \in \mathcal{I}^\ell_c} H^{\ell-1}_{\mathbf{i},\mathbf{i}',\mathbf{u}}(\boldsymbol{\xi}^c_{\mathbf{q}}) \right) \left(B^{\ell-1}_{\mathbf{j},\mathbf{v}}(\boldsymbol{\xi}^c_{\mathbf{q}}) - \sum_{\mathbf{j}' \in \mathcal{I}^\ell_c} H^{\ell-1}_{\mathbf{j},\mathbf{j}',\mathbf{v}}(\boldsymbol{\xi}^c_{\mathbf{q}}) \right) \nu^c_{\mathbf{q}} \\ &= \upsilon^{\ell-1,\ell-1}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}} - \phi^{\ell-1}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}} - \phi^{\ell-1}_{\mathbf{j},\mathbf{i},\mathbf{v},\mathbf{u}} + \psi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}} \ .\end{aligned} \tag{12}$$

Lemma 3. *The computation of a single cell-wise mass matrix entry from the auxiliary matrices requires no more than 3 additions and subtractions.*

5.3. The mass matrix assembly algorithm

Algorithm 1 presents a complete, executable procedure for assembling the global mass matrix for $d = 3$. Some comments are in order:

1. In addition to $\mathcal{I}^{\ell-1}_c$ and $\mathcal{I}^\ell_c$, we use three further index sets $\mathcal{G}, \mathcal{S}, \mathcal{E}$ to keep the notation compact:
   - In each cell, we have $(p+1)^3$ Gauss quadrature nodes, denoted by $\boldsymbol{\xi}^c_q = (\xi^{c,1}_{q_1}, \xi^{c,2}_{q_2}, \xi^{c,3}_{q_3})$, where the indices vary in
   $$\mathcal{G} = \{0, \dots, p\}^3 = \{(0,0,0), (0,0,1), \dots, (p,p,p)\} \ .$$
   - Each (TH)B-system consists of $(s+1)^3$ functions, which are identified by indices $\mathbf{u}$ that vary in
   $$\mathcal{S} = \{0, \dots, s\}^3 = \{(0,0,0), (0,0,1), \dots, (s,s,s)\} \ .$$
   - To keep the presentation compact, we use the empty string ε and
   $$\mathcal{E} = \{(\varepsilon, \varepsilon, \varepsilon)\}$$
   to omit indices.
2. Moreover, we use the projection operators P_k and $\mathsf{P}_{kk'}$ to extract index sets with fewer dimensions by projection. When applied to an index triplet $\mathbf{i} = (i_1, i_2, i_3)$,
   $$\mathsf{P}_k(i_1, i_2, i_3) = i_k \quad \text{and} \quad \mathsf{P}_{kk'}(i_1, i_2, i_3) = (i_k, i_{k'}) \ .$$
   The extension to index sets is defined by applying the operator to each element and collecting the results. For instance,
   $$\mathsf{P}_k\mathcal{G} = \{0, \dots, p\} \quad \text{and} \quad \mathsf{P}_{kk'}\mathcal{G} = \{(0,0), (0,1), \dots, (p,p)\} \ .$$
   We will also apply to the projection operators to $\mathcal{I}^{\ell-1}_c$ and $\mathcal{I}^\ell_c$, which will produce mesh-dependent results.

Algorithm 1 ASSEMBLY OF THE MASS MATRIX

1: **Input:** Set of active cells C
2: **Output:** Global mass matrix $\tilde{M}$
3: $\tilde{M} \leftarrow 0$
4: **for** $c \in C$ **do** ▷ Loop over all cells
5: $\quad \tilde{M}^c \leftarrow 0$, $\ell \leftarrow$ cell level, $\mathcal{I}_c^{\ell-1}$ and $\mathcal{I}_c^{\ell} \leftarrow$ index sets
6: $\quad$ **for** $k = 1, 2, 3$ **do** ▷ Precomputation of function values
7: $\quad\quad$ **for** $i_k \in \mathsf{P}_k\mathcal{I}_c^{\ell-1}$, $u_k \in \mathsf{P}_k\mathcal{S}$, $q_k \in \mathsf{P}_k\mathcal{G}$ **do**
8: $\quad\quad\quad \beta^{\ell-1,k}_{i_k,\varepsilon,u_k,q_k} \leftarrow \beta^{\ell-1}_{i_k,u_k}(\xi^{c,k}_{q_k})$
9: $\quad\quad\quad$ **for** $j_k \in \mathsf{P}_k\mathcal{I}_c^{\ell}$ **do**
10: $\quad\quad\quad\quad \eta^{\ell-1,k}_{i_k,j_k,u_k,q_k} \leftarrow \eta^{\ell-1}_{i_k,j_k,u_k}(\xi^{c,k}_{q_k})$
11: $\quad\quad$ **for** $i_k \in \mathsf{P}_k\mathcal{I}_c^{\ell}$, $u_k \in \mathsf{P}_k\mathcal{S}$, $q_k \in \mathsf{P}_k\mathcal{G}$ **do**
12: $\quad\quad\quad \beta^{\ell,k}_{i_k,\varepsilon,u_k,q_k} \leftarrow \beta^{\ell}_{i_k,u_k}(\xi^{c,k}_{q_k})$
13: ▷ first group of auxiliary matrices
14: $(\Upsilon^{\ell-1,\ell-1}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I}_c^{\ell-1},\mathbf{j}\in\mathcal{I}_c^{\ell-1}} \leftarrow \textsc{AuxMat}(\ (\beta^{\ell-1,k}_{i_k,\varepsilon,u_k,q_k})_{k,i_k,\varepsilon,u_k,q_k}, (\beta^{\ell-1,k}_{j_k,\varepsilon,v_k,q_k})_{k,j_k,\varepsilon,v_k,q_k}, \mathcal{I}_c^{\ell-1}, \mathcal{E}, \mathcal{I}_c^{\ell-1}, \mathcal{E})$
15: $(\Upsilon^{\ell,\ell-1}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I}_c^{\ell},\mathbf{j}\in\mathcal{I}_c^{\ell-1}} \leftarrow \textsc{AuxMat}(\ (\beta^{\ell,k}_{i_k,\varepsilon,u_k,q_k})_{k,i_k,\varepsilon,u_k,q_k}, (\beta^{\ell-1,k}_{j_k,\varepsilon,v_k,q_k})_{k,j_k,\varepsilon,v_k,q_k}, \mathcal{I}_c^{\ell}, \mathcal{E}, \mathcal{I}_c^{\ell-1}, \mathcal{E})$
16: $(\Upsilon^{\ell,\ell}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I}_c^{\ell},\mathbf{j}\in\mathcal{I}_c^{\ell}} \leftarrow \textsc{AuxMat}(\ (\beta^{\ell,k}_{i_k,\varepsilon,u_k,q_k})_{k,i_k,\varepsilon,u_k,q_k}, (\beta^{\ell,k}_{j_k,\varepsilon,v_k,q_k})_{k,j_k,\varepsilon,v_k,q_k}, \mathcal{I}_c^{\ell}, \mathcal{E}, \mathcal{I}_c^{\ell}, \mathcal{E})$
17: ▷ second group
18: $(\Phi^{\ell}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I}_c^{\ell},\mathbf{j}\in\mathcal{I}_c^{\ell-1}} \leftarrow \textsc{AuxMat}(\ (\beta^{\ell,k}_{i_k,\varepsilon,u_k,q_k})_{k,i_k,\varepsilon,u_k,q_k}, (\eta^{\ell-1,k}_{j_k,j'_k,v_k,q_k})_{k,j_k,j'_k,v_k,q_k}, \mathcal{I}_c^{\ell}, \mathcal{E}, \mathcal{I}_c^{\ell-1}, \mathcal{I}_c^{\ell})$
19: $(\Phi^{\ell-1}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I}_c^{\ell-1},\mathbf{j}\in\mathcal{I}_c^{\ell-1}} \leftarrow \textsc{AuxMat}(\ (\beta^{\ell-1,k}_{i_k,\varepsilon,u_k,q_k})_{k,i_k,\varepsilon,u_k,q_k}, (\eta^{\ell-1,k}_{j_k,j'_k,v_k,q_k})_{k,j_k,j'_k,v_k,q_k}, \mathcal{I}_c^{\ell-1}, \mathcal{E}, \mathcal{I}_c^{\ell-1}, \mathcal{I}_c^{\ell})$
20: ▷ third group
21: $(\Psi_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I}_c^{\ell-1},\mathbf{j}\in\mathcal{I}_c^{\ell-1}} \leftarrow \textsc{AuxMat}(\ (\eta^{\ell-1,k}_{i_k,i'_k,u_k,q_k})_{k,i_k,i'_k,u_k,q_k}, (\eta^{\ell-1,k}_{j_k,j'_k,v_k,q_k})_{k,j_k,j'_k,v_k,q_k}, \mathcal{I}_c^{\ell-1}, \mathcal{I}_c^{\ell}, \mathcal{I}_c^{\ell-1}, \mathcal{I}_c^{\ell})$
22: $\quad$ **for** $\mathbf{i} \in \mathcal{I}_c^{\ell-1}$, $\mathbf{j} \in \mathcal{I}_c^{\ell-1}$ **do** ▷ local matrix, first block of submatrices
23: $\quad\quad \tilde{M}^c_{(\ell-1,\mathbf{i}),(\ell-1,\mathbf{j})} \leftarrow \Upsilon^{\ell-1,\ell-1}_{\mathbf{i},\mathbf{j}} - \Phi^{\ell-1}_{\mathbf{i},\mathbf{j}} - (\Phi^{\ell-1}_{\mathbf{j},\mathbf{i}})^{\mathrm{T}} + \Psi_{\mathbf{i},\mathbf{j}}$
24: $\quad$ **for** $\mathbf{i} \in \mathcal{I}_c^{\ell-1}$, $\mathbf{j} \in \mathcal{I}_c^{\ell}$ **do** ▷ second block of submatrices
25: $\quad\quad \tilde{M}^c_{(\ell-1,\mathbf{i}),(\ell,\mathbf{j})} \leftarrow (\Upsilon^{\ell,\ell-1}_{\mathbf{j},\mathbf{i}} - \Phi^{\ell}_{\mathbf{j},\mathbf{i}})^{\mathrm{T}}$
26: $\quad$ **for** $\mathbf{i} \in \mathcal{I}_c^{\ell}$, $\mathbf{j} \in \mathcal{I}_c^{\ell-1}$ **do** ▷ third block of submatrices
27: $\quad\quad \tilde{M}^c_{(\ell,\mathbf{i}),(\ell-1,\mathbf{j})} \leftarrow \Upsilon^{\ell,\ell-1}_{\mathbf{i},\mathbf{j}} - \Phi^{\ell}_{\mathbf{i},\mathbf{j}}$
28: $\quad$ **for** $\mathbf{i} \in \mathcal{I}_c^{\ell}$, $\mathbf{j} \in \mathcal{I}_c^{\ell}$ **do** ▷ fourth block of submatrices
29: $\quad\quad \tilde{M}^c_{(\ell,\mathbf{i}),(\ell,\mathbf{j})} \leftarrow \Upsilon^{\ell,\ell}_{\mathbf{i},\mathbf{j}}$
30: $\quad \tilde{M} \mathrel{+}= \tilde{M}^c$ ▷ contribution to global matrix
31: **return** $\tilde{M}$

3. The algorithm builds on an assembly algorithm for the auxiliary matrices (used in lines 13-21), which is described in Appendix D. The latter algorithm is structured into three main loops, one for each coordinate direction.
4. The approximate local mass matrix is structured into $(|\mathcal{I}_c^{\ell-1}| + |\mathcal{I}_c^{\ell}|)^2 = (2^3)^2 = 64$ sub-matrices

$$\tilde{M}^c_{(\hat{\ell},\mathbf{i}),(\bar{\ell},\mathbf{j})} = (\tilde{m}^c_{(\hat{\ell},\mathbf{i},\mathbf{u}),(\bar{\ell},\mathbf{j},\mathbf{v})})_{\mathbf{u},\mathbf{v}\in\mathcal{S}} \ .$$

This fact is exploited in lines 22-28.

Theorem 1. *The proposed algorithm for assembling the mass matrices* (1) *arising from isogeometric discretizations with C^s-smooth THB-splines of degree $p = 2s + 1$ requires no more than $\mathcal{O}(p^{2d+1})$ flops per element.*

Proof. For each element, we assemble $\mathcal{O}(1)$ auxiliary matrices (one per pair of THB-systems). By Lemma 2, each matrix costs $\mathcal{O}(p^{2d+1})$ flops. □

In contrast, the standard Gaussian quadrature for the same mass matrix assembly requires $\mathcal{O}(p^{3d})$ flops per element: Each d-dimensional integral evaluation requires $\mathcal{O}(p^d)$ Gauss nodes, and each node contributes $\mathcal{O}(p^{2d})$ flops for THB-spline product evaluations.

6. Numerical experiments

We validate the theoretical complexity of the proposed assembly algorithm and assess its practical performance. We conducted two sets of experiments: the first examined the computational cost of assembling mass matrices using the proposed method. The second embedded the algorithm into an adaptive refinement framework for solving elliptic PDEs. In the latter experiment, we compared C^1 tricubic THB-splines with C^2 tricubic HB- and THB-splines in terms of accuracy and matrix properties.

All tests were performed on the computational domains shown in Figure 1. The algorithms were implemented in C++ using the `G+Smo` library [26]. All the experiments are conducted on a laptop computer with a 12th Gen Intel Core i7-12700H (2.70 GHz) and 16.0 GB DDR5 4800MHz RAM. To alleviate random fluctuations, all reported CPU times were averaged over five runs.

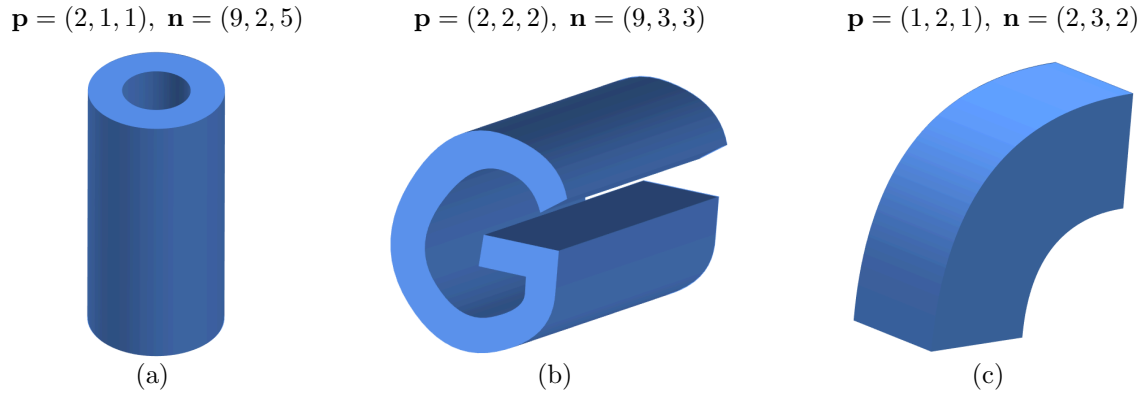


Figure 1: The computational domains of the numerical experiments, represented by B-splines of degree $\mathbf{p}$ with $\mathbf{n}$ control points. .

6.1. Computational complexity

First, we investigate the relationship between the computational cost of the proposed algorithm and the number of cells. We performed the test on the single-patch domain depicted in Figure 1(a) using six THB-spline bases of degrees $p = 1, 3, 5$. The number of degrees of freedom possessed by the bases (that we consider on the same hierarchical mesh) varies for each degree p.

Starting from an initial tensor-product mesh with $4 \times 4 \times 4$ cells, the ith ($i = 0, \ldots, 5$) hierarchical mesh is generated using four distinct refinement strategies:

- *Random refinement*: In each step, 60% of the cells at the finest level are randomly selected for refinement.
- *Central refinement*: The eight cells surrounding the central point of the initial mesh are progressively refined.
- *Curve-driven refinement*: All cells traversed by the following parabolic curve are refined:

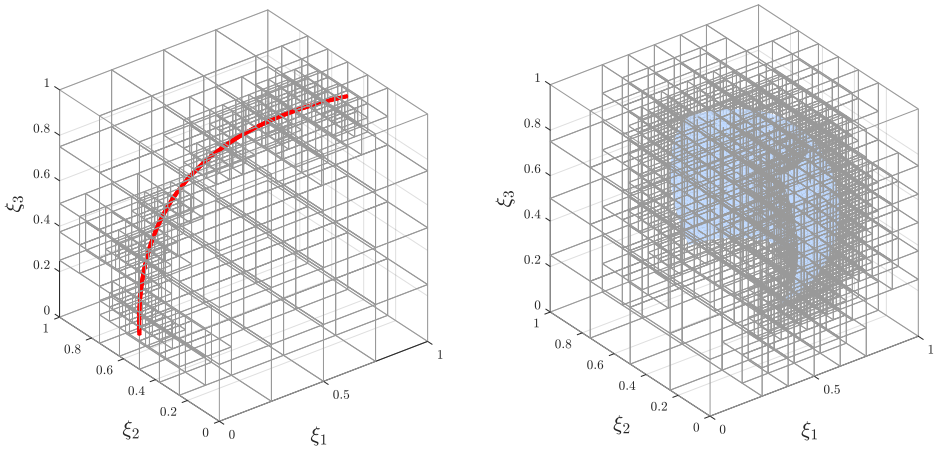
$$\begin{aligned}\xi_1(t) &= t,\\ \xi_2(t) &= 0.5 + 1.4t(1-t), \quad t \in [0,1]\\ \xi_3(t) &= 0.15 + 0.7t.\end{aligned}$$

- *Surface-driven refinement*: All cells that intersect the following spherical patch are refined:

$$\begin{aligned}\alpha(s) &= \frac{\pi}{2}(s-0.5),\\ \beta(t) &= \pi(t-0.5),\\ \xi_1(s,t) &= 0.5 + 0.4\cos\alpha\cos\beta, \quad s,t \in [0,1]\\ \xi_2(s,t) &= 0.5 + 0.4\cos\alpha\sin\beta,\\ \xi_3(s,t) &= 0.5 + 0.4\sin\alpha.\end{aligned}$$

The postprocessing step discussed in [7, 36] is employed to ensure the mesh grading property of all hierarchical meshes.

Figures 2(a) and 2(b) illustrate hierarchical meshes with three levels generated by the curve-driven and surface-driven refinement strategies, respectively.



(a) (b)

Figure 2: Hierarchical meshes with three levels generated using (a) curve-driven and (b) surface-driven refinement strategies.

Figure 3 reports the total assembly time, i.e., the time required by all the loops, as a function of the number of cells for the four refinement strategies and the three polynomial

degrees. It is worth noting that the new algorithm involves a large number of sparse matrix accessing operations, which affect the required time to some extent. Nevertheless, this does not significantly impact the overall performance. All the plots indicate that the asymptotic computational cost roughly scales linearly with the number of cells, confirming the theoretical result stated in Theorem 1.

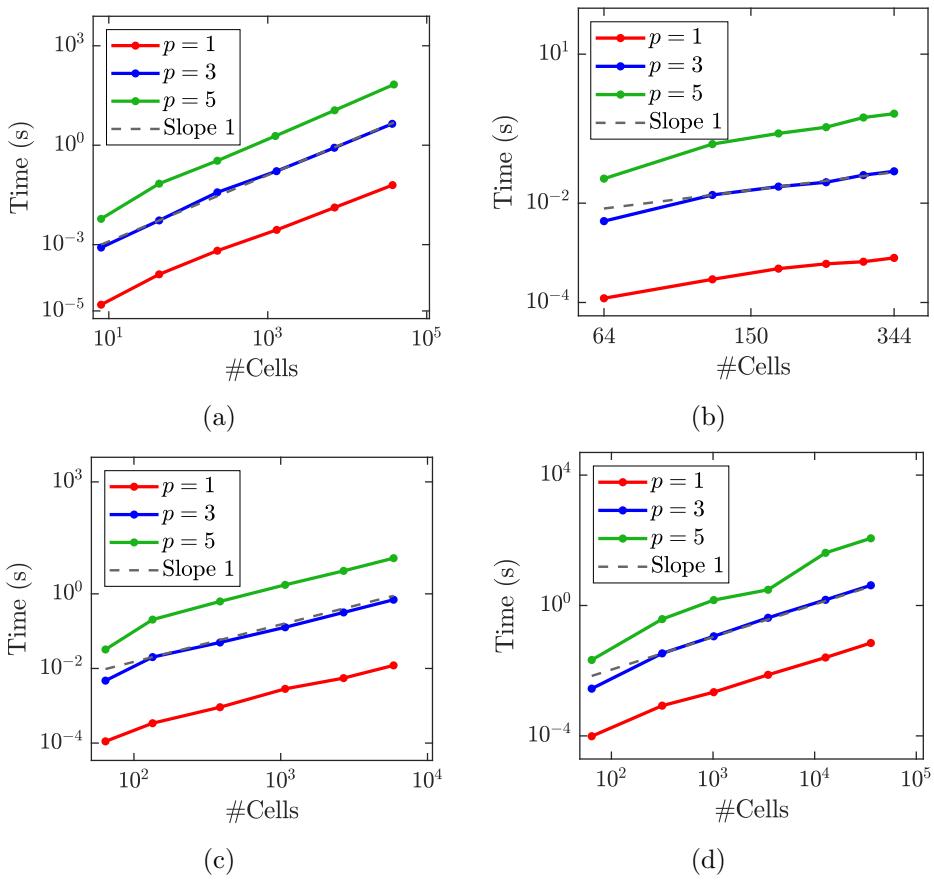


Figure 3: Assembly time (all loops) versus number of cells for hierarchical meshes generated via (a) random refinement, (b) central refinement, (c) curve-driven refinement and (d) surface-driven refinement strategies, for spline degrees $p = 1, 3, 5$.

Second, we study the relationship between the complexity and spline degrees. Before presenting the results, we clarify the terminology used in the subsequent figures:

As described in the appendix, the auxiliary matrices $\Upsilon_{\mathbf{i},\mathbf{j}}$, $\Phi_{\mathbf{i},\mathbf{j}}$, and $\Psi_{\mathbf{i},\mathbf{j}}$ are assembled by invoking three nested summation loops, one for each coordinate direction. Loop 1 corresponds to the innermost level of summation, which evaluates the intermediate tensors I by summing over the third coordinate direction. Loop 2 evaluates the intermediate tensors II based on the results of Loop 1 by summing over the second coordinate direction. Loop 3 corresponds to the outermost level of summation, which completes the assembly of the final auxiliary matrices.

The figures labeled "Loop 1", "Loop 2" and "Loop 3" report the cumulative CPU time

spent in each stage. The figures labeled "All loops" report the total time across all stages.

We assemble the mass matrices on the single-patch domain depicted in Figure 1(b) using THB-spline bases with varying degrees ($p = 1, 3, 5, 7, 9$). For each degree, the hierarchical meshes are generated following the refinement strategies described above. The only difference is that all meshes consist of exactly three levels.

Figure 4 demonstrates the relationship between computational cost and spline degrees on various hierarchical meshes. The figure shows that the cost of assembling the auxiliary matrices increases progressively from the innermost to the outermost loop. More importantly, the observed growth rates have not yet reached the theoretically predicted asymptotic values as $p \to \infty$: 9 for Gaussian quadrature and 7 for our method.

6.2. Solving PDEs with tricubic splines

We embed the proposed assembly algorithm into an adaptive refinement framework to investigate the performance of different tricubic spline discretizations when solving elliptic problems. Specifically, we compare three spline bases:

(i) C^1 tricubic THB-splines
(ii) C^2 tricubic HB-splines, and
(iii) C^2 tricubic THB-splines.

The comparison focuses on two aspects: approximation accuracy versus the number of degrees of freedom and the algebraic properties of the resulting linear systems, including sparsity, bandwidth, and conditioning.

We consider the following elliptic problem with Dirichlet boundary conditions

$$\begin{cases} -\Delta u + u = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega \end{cases} \tag{13}$$

on the bounded domain depicted in Figure 1(c). The exact solution is chosen as

$$u_{\text{exact}}(x, y, z) = e^{-100((x-0.5)^2+(y-1.5)^2+(z-0.5)^2)} + e^{-100((x-1.5)^2+(y-0.5)^2+(z-0.5)^2)},$$

which exhibits two steep peaks centered at $(0.5, 1.5, 0.5)$ and $(1.5, 0.5, 0.5)$. This solution is chosen to test the capability of adaptive refinement to resolve localized features.

The adaptive mesh refinement is guided by an error estimator that considers the difference between the computed and exact solutions. This estimator uses the Dörfler marking strategy [11] with parameter $\theta = 0.4$.

We compare the performance of the three different spline discretizations (i-iii). Figure 5 reports the convergence behavior of adaptive refinement with these three spline bases, where the L^2-error is plotted against the number of degrees of freedom. Among them, the C^2 HB-splines and C^2 THB-splines yield nearly identical errors for a given number of degrees of freedom, as expected, since they both span the same space. However, the C^1 THB-splines require slightly more degrees of freedom to reach a given accuracy. This is a direct consequence of reduced continuity: C^1 splines provide more basis functions than C^2 splines on the same mesh.

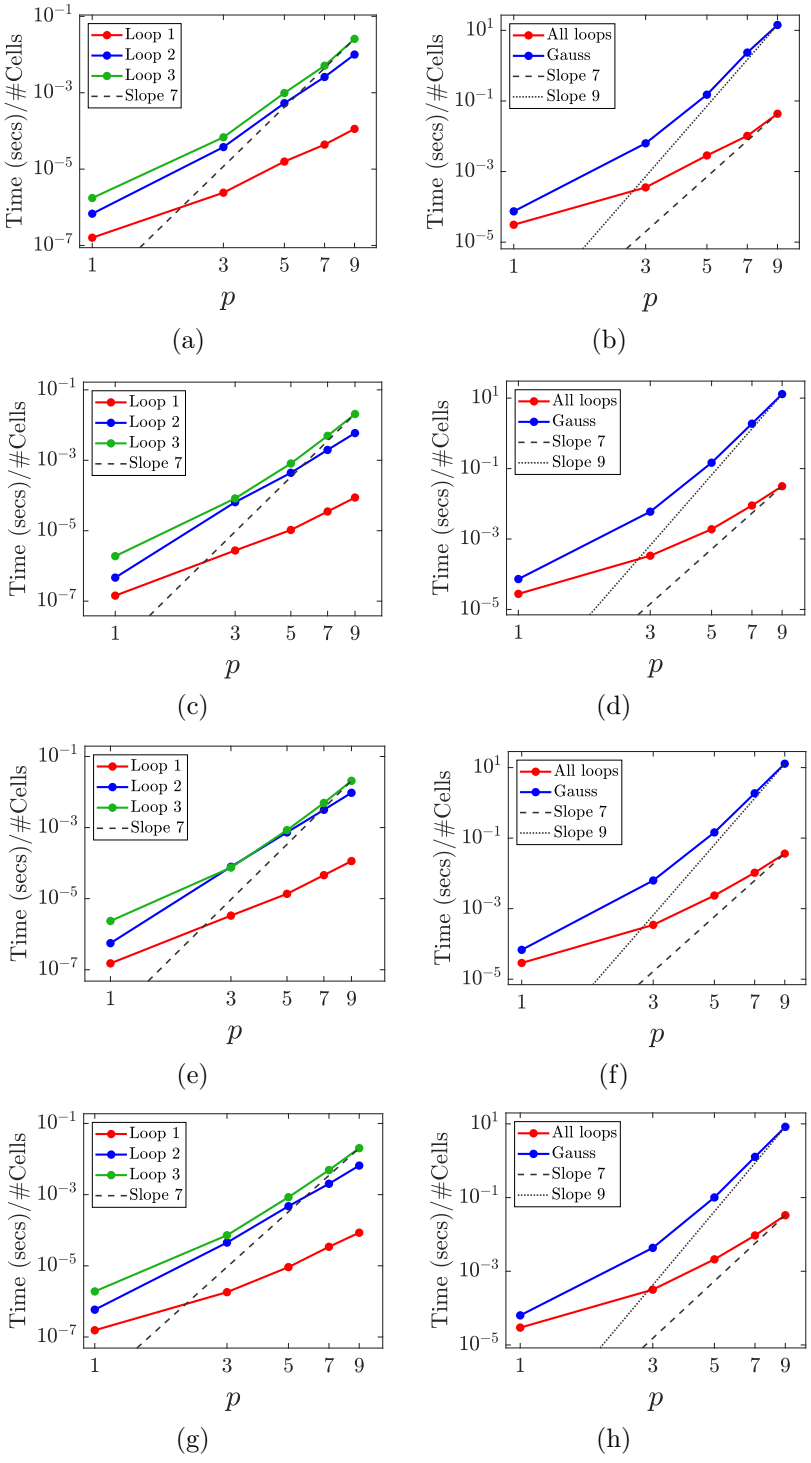


Figure 4: Dependence of the computational cost on the spline degree for hierarchical meshes with three refinement levels. Left column (a,c,e,g): assembly time of the auxiliary matrices for each summation loops. Right column (b,d,f,h): total assembly time (all loops and Gaussian quadrature). Rows: (a,b) random, (c,d) central, (e,f) curve-driven, and (g,h) surface-driven refinement.

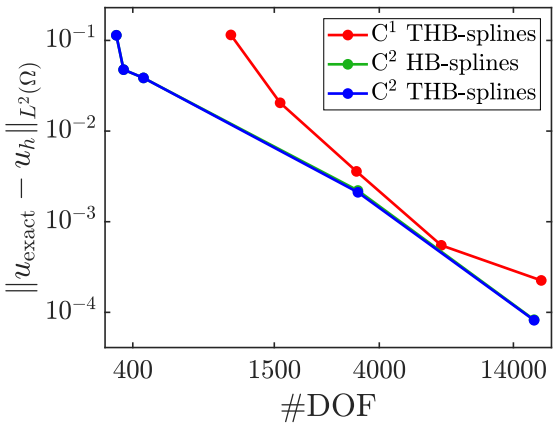


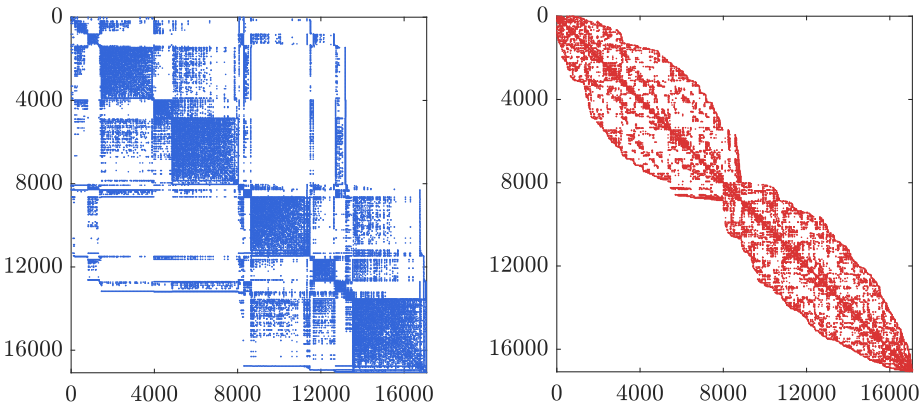
Figure 5: Convergence behaviour for adaptive refinement with (i) C^1 THB-splines, (ii) C^2 HB-splines and (iii) C^2 THB-splines of degree 3.

In addition to the accuracy of the solutions, we also investigate the matrix properties of the resulting linear systems. Figure 6 depicts the sparsity patterns of the system matrices at the fourth refinement level obtained for C^1 (i) tricubic THB-splines and

(iv) C^1 tricubic HB-splines

after Cuthill-McKee reordering. We compare THB splines with HB splines because they span the same space, and an efficient assembly algorithm for HB splines has already been reported [37].

The sparsity pattern of THB-splines is noticeably more compact than that of HB-splines, indicating a more localized coupling between basis functions. This observation is quantified in Table 3, which reports the number of nonzero entries, bandwidth and condition numbers for both discretizations across five refinement levels.



(a) C^1 tricubic HB-splines (b) C^1 tricubic THB-splines

Figure 6: Sparsity patterns of system matrix at the 4th step of refinements after Cuthill-McKee-reordering.

Several useful observations can be drawn from Table 3. First, the number of nonzero

entries in the THB-based system matrices is consistently lower than in the HB matrices, with the ratio THB/HB decreasing from 1.00 at level 0 to 0.56 at level 4. Second, the bandwidth of THB-spline matrices is substantially smaller than that of HB-spline matrices, with the ratio THB/HB dropping from 1.00 to 0.30 at level 4. Third, the condition numbers of THB-based matrices are slightly smaller than those of HB-based matrices across all refinement levels, with the ratio THB/HB ranging from 0.89 to 0.97.

Level ℓ	#DOF	Sparsity						Condition number		
		number of nonzero entries			bandwidth					
		HB	THB	THB/HB	HB	THB	THB/HB	HB	THB	THB/HB
0	512	54,872	54,872	1.00	232	232	1.00	7.22e02	7.22e02	1.00
1	960	140,712	130,056	0.92	507	355	0.70	8.71e02	8.45e02	0.97
2	2,416	603,064	416,664	0.69	1,544	706	0.46	1.38e03	1.28e03	0.93
3	6,240	2,105,336	1,245,528	0.59	4,264	1,750	0.41	3.10e03	2.75e03	0.89
4	17,084	6,548,632	3,649,624	0.56	11,809	3,517	0.30	6.77e03	6.13e03	0.91

Table 3: Sparsity and condition numbers of the system matrices generated in solving problem (13) with C^1 tricubic HB-splines and THB-splines.

Overall, these results demonstrate that C^1 tricubic THB-splines, while requiring slightly more degrees of freedom to achieve a given accuracy than C^2 splines, offer superior matrix properties in terms of sparsity, bandwidth, and conditioning. This makes them an attractive choice for large-scale adaptive isogeometric simulations, especially when combined with the efficient assembly algorithm developed in this paper, that does not apply to C^2 tricubic THB-splines.

6.3. Speedup over Gaussian quadrature

Finally we evaluate the computational efficiency of the proposed algorithm by comparing it with standard Gaussian quadrature in the context of solving the elliptic problem (13). Table 4 reports the stiffness matrix assembly time for both methods, along with the resulting speedup factors, for various spline degrees and refinement levels. To ensure a valid comparison, both methods are applied to the same hierarchical meshes with mesh grading enforced at each level and identical system matrices assembled.

Two main observations can be drawn from Table 4. First, the proposed algorithm consistently outperforms Gaussian quadrature across all configurations, with speedup factors ranging from 1.83 for $p = 1$ at level 1 to 76.34 for $p = 7$ at level 2. Second, the speedup becomes more pronounced as the spline degree increases. For $p = 1$, the speedup is modest (around 2), while for $p = 3$ it reaches approximately 12, and for $p = 5$ and $p = 7$ it increases to between 26 and 76. This trend aligns with the theoretical complexity analysis in Theorem 1: the proposed method scales as $\mathcal{O}(p^7)$ per element, whereas Gaussian quadrature scales as $\mathcal{O}(p^9)$, resulting in an asymptotic speedup of $\mathcal{O}(p^2)$.

Degree	Level	#DOF	#Cell	Assembly time (s)		Speedup
				GQ	Ours	
1	0	125	64	5.93e-3	2.56e-3	2.31
1	1	201	176	2.06e-2	1.13e-2	1.83
1	2	783	764	7.99e-2	4.05e-2	1.97
1	3	2,374	2,773	3.33e-1	1.59e-1	2.10
3	0	1,000	64	2.99e-1	2.29e-2	13.05
3	1	1,584	176	2.17e0	1.73e-1	12.59
3	2	4,488	596	1.47e1	1.20e0	12.19
3	3	14,656	2,136	5.13e1	4.33e0	11.86
5	0	3,375	64	7.28e0	2.78e-1	26.22
5	1	5,562	176	9.91e1	2.16e0	45.48
5	2	15,741	624	2.16e2	7.35e0	29.40
5	3	48,114	2,108	7.19e2	2.61e1	27.51
7	0	8,000	64	3.79e1	7.29e-1	52.03
7	1	13,184	176	7.57e2	1.05e1	71.90
7	2	37,440	624	2.84e3	3.72e1	76.34
7	3	116,992	2,192	1.03e4	1.62e2	63.41

Table 4: Assembly time (in seconds) and speedup factors of the proposed method over standard Gaussian quadrature for different spline degrees and refinement levels.

7. Conclusion

We developed an efficient matrix assembly algorithm for isogeometric discretizations using C^s-smooth THB-splines of degree $p = 2s + 1$. The order of smoothness was chosen to be less than maximal ($\frac{p-1}{2}$ rather than $p - 1$) to enable sum-factorization.

The key idea is to decompose each truncated basis function in an active cell as a tensor-product B-spline component minus a correction term that preserves the tensor-product structure. This decomposition, formalized through C-systems, allows us to use sum-factorization to assemble three groups of auxiliary matrices.

The proposed algorithm achieves a computational complexity of $\mathcal{O}(p^{2d+1})$ per element, compared to $\mathcal{O}(p^{3d})$ for standard Gaussian quadrature. Numerical experiments confirm the theoretical complexity analysis and demonstrate substantial speedups over Gaussian quadrature. The improvement becomes more pronounced as the spline degree increases.

Regarding the generalizability of the proposed framework, two directions deserve further discussion.

- First, the extension to maximally smooth THB-splines with C^{p-1} continuity is not straightforward. Although the decomposition into tensor-product B-spline components and correction terms remains valid, the truncation mechanism for maximally smooth splines involves a more complicated coupling between coarse-level and fine-level basis functions. In particular, on an active cell the truncated coarse-level functions receive contributions from a larger stencil of fine-level B-systems, and the resulting correction terms are no longer associated solely with the vertices of the cell. A direct application of the present C-system decomposition would therefore require additional assumptions

on the hierarchical mesh, such as macro-element refinement, together with special quadrature rules designed for the enlarged interaction stencil.

- Second, for lower smoothness with fixed degree $p > 2s+1$, the proposed decomposition framework and the sum-factorization assembly algorithm can be extended, but the structure of the functions on an active cell becomes more complicated. In the present setting the relevant B-systems are associated with the vertices of the cell. For $p > 2s+1$ in three dimensions, additional B-systems associated with 12 edges, 6 faces, and the cell itself have to be taken into account. Nevertheless, the main observation concerning the tensor-product structure of the correction terms remains valid, which suggests that a generalized C-system decomposition and a corresponding sum-factorization algorithm can be expected.

Another promising direction is the integration of the proposed assembly algorithm with iterative solvers and preconditioning techniques for large-scale adaptive simulations. Finally, we also note the possibility of parallelization, since computations on the cells may run in parallel.

Acknowledgement. Maodong Pan was funded by the National Key R&D Program of China (No. 2024YFA1016300), the National Natural Science Foundation of China (No. 12471359) and the Basic Research Program of Jiangsu Province, China (No. BK20250176). Bert Jüttler was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/I6779 and 10.55776/PIN6740223.

Appendices

Appendix A: Assembling the auxiliary matrices of the first group

We will describe the assembly process for dimension $d = 3$. The algorithm works in a similar way for other dimensions. With the help of the tensor-product structure (3) we can rewrite the matrix entries (8) as

$$v^{\hat{\ell},\bar{\ell}}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}} = \underbrace{\sum_{q_1} \beta^{\hat{\ell}}_{i_1,u_1}(\xi^{c,1}_{q_1})\beta^{\bar{\ell}}_{j_1,v_1}(\xi^{c,1}_{q_1}) \underbrace{\sum_{q_2} \beta^{\hat{\ell}}_{i_2,u_2}(\xi^{c,2}_{q_2})\beta^{\bar{\ell}}_{j_2,v_2}(\xi^{c,2}_{q_2}) \underbrace{\sum_{q_3} \beta^{\hat{\ell}}_{i_3,u_3}(\xi^{c,3}_{q_3})\beta^{\bar{\ell}}_{j_3,v_3}(\xi^{c,3}_{q_3})\nu^{c}_{(q_1,q_2,q_3)}}_{= \mathrm{I}_{(q_1,q_2),i_3,j_3,u_3,v_3}}}_{= \mathrm{II}_{q_1,(i_2,i_3),(j_2,j_3),(u_2,u_3),(v_2,v_3)}}}_{= \mathrm{III}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}} .$$

We note that the indices q_r, i_r, j_r, u_r and v_r, where $r \in \{1, 2, 3\}$, take

$$p + 1 = \mathcal{O}(p) \ , \ \mathcal{O}(1) \ , \ \mathcal{O}(1) \ , \ s + 1 = \mathcal{O}(p) \ , \text{ and } s + 1 = \mathcal{O}(p)$$

different values, respectively. The matrix entries are evaluated in three steps:

I. We evaluate the
$$\mathcal{O}(p^2)\cdot\mathcal{O}(1)\cdot\mathcal{O}(1)\cdot\mathcal{O}(p)\cdot\mathcal{O}(p)=\mathcal{O}(p^4)$$
quantities
$$\mathrm{I}_{(q_1,q_2),i_3,j_3,u_3,v_3}$$
requiring $\mathcal{O}(p)$ flops for each one of them.

II. We evaluate the
$$\mathcal{O}(p)\cdot\mathcal{O}(1^2)\cdot\mathcal{O}(1^2)\cdot\mathcal{O}(p^2)\cdot\mathcal{O}(p^2)=\mathcal{O}(p^5)$$
quantities
$$\mathrm{II}_{q_1,(i_2,i_3),(j_2,j_3),(u_2,u_3),(v_2,v_3)}$$
requiring $\mathcal{O}(p)$ flops for each one of them.

III. We evaluate the
$$\mathcal{O}(1^3)\cdot\mathcal{O}(1^3)\cdot\mathcal{O}(p^3)\cdot\mathcal{O}(p^3)=\mathcal{O}(p^6)$$
quantities
$$\mathrm{III}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}=\mathrm{III}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}=\upsilon^{\hat{\ell},\bar{\ell}}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}=\upsilon^{\hat{\ell},\bar{\ell}}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}$$
requiring $\mathcal{O}(p)$ flops for each one of them.

The total effort, dominated by the third step, amounts to $\mathcal{O}(p^7)$ flops.

Appendix B: Assembling the auxiliary matrices of the second group

We will describe the assembly process for dimension $d=3$. The algorithm works in a similar way for other dimensions. With the help of the tensor-product structure (3) and (6) we can rewrite the matrix entries (9) as

$$\phi^{\hat{\ell}}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}=\sum_{(j'_1,j'_2,j'_3)\in\mathcal{I}_c^\ell}\sum_{q_1}\beta^{\hat{\ell}}_{i_1,u_1}(\xi^{c,1}_{q_1})\eta^{\ell-1}_{j_1,j'_1,v_1}(\xi^{c,1}_{q_1})\underbrace{\sum_{q_2}\beta^{\hat{\ell}}_{i_2,u_2}(\xi^{c,2}_{q_2})\eta^{\ell-1}_{j_2,j'_2,v_2}(\xi^{c,2}_{q_2})\underbrace{\sum_{q_3}\beta^{\hat{\ell}}_{i_3,u_3}(\xi^{c,3}_{q_3})\eta^{\ell-1}_{j_3,j'_3,v_3}(\xi^{c,3}_{q_3})\nu^c_{(q_1,q_2,q_3)}}_{=\mathrm{I}_{(q_1,q_2),i_3,j_3,j'_3,u_3,v_3}}}_{=\mathrm{II}_{q_1,(i_2,i_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}}}_{=\mathrm{III}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(j'_1,j'_2,j'_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}}$$
$$=\phi^{\hat{\ell}}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}$$

We note that the indices q_r, i_r, j_r, j'_r, u_r and v_r, where $r\in\{1,2,3\}$, take
$$p+1=\mathcal{O}(p)\ ,\ \mathcal{O}(1)\ ,\ \mathcal{O}(1)\ ,\ s+1=\mathcal{O}(p)\ ,\ \text{and}\ s+1=\mathcal{O}(p)$$
different values, respectively. The matrix entries are evaluated in four steps:

I. We evaluate the

$$\mathcal{O}(p^2)\cdot\mathcal{O}(1)\cdot\mathcal{O}(1)\cdot\mathcal{O}(1)\cdot\mathcal{O}(p)\cdot\mathcal{O}(p)=\mathcal{O}(p^4)$$

quantities

$$\mathrm{I}_{(q_1,q_2),i_3,j_3,j'_3,u_3,v_3}$$

requiring $\mathcal{O}(p)$ flops for each one of them.

II. We evaluate the

$$\mathcal{O}(p)\cdot\mathcal{O}(1^2)\cdot\mathcal{O}(1^2)\cdot\mathcal{O}(1^2)\cdot\mathcal{O}(p^2)\cdot\mathcal{O}(p^2)=\mathcal{O}(p^5)$$

quantities

$$\mathrm{II}_{q_1,(i_2,i_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}$$

requiring $\mathcal{O}(p)$ flops for each one of them.

III. We evaluate the

$$\mathcal{O}(1^3)\cdot\mathcal{O}(1^3)\cdot\mathcal{O}(1^3)\cdot\mathcal{O}(p^3)\cdot\mathcal{O}(p^3)=\mathcal{O}(p^6)$$

quantities

$$\mathrm{III}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(j'_1,j'_2,j'_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}=\mathrm{III}_{\mathbf{i},\mathbf{j},\mathbf{j}',\mathbf{u},\mathbf{v}}$$

requiring $\mathcal{O}(p)$ flops for each one of them.

IV. We evaluate the

$$\mathcal{O}(1^3)\cdot\mathcal{O}(1^3)\cdot\mathcal{O}(p^3)\cdot\mathcal{O}(p^3)=\mathcal{O}(p^6)$$

quantities

$$\phi^{\hat{\ell}}_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}=\phi^{\hat{\ell}}_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}$$

requiring $\mathcal{O}(1)$ flops for each one of them.

The total effort, dominated by the third step, amounts to $\mathcal{O}(p^7)$ flops.

Appendix C: Assembling the auxiliary matrices of the third group

We will describe the assembly process for dimension $d=3$. The algorithm works in a similar way for other dimensions. With the help of the tensor-product structure (6) we can rewrite the matrix entries (10) as

$$\psi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}=\underbrace{\underbrace{\underbrace{\underbrace{\sum_{(i'_1,i'_2,i'_3)\in\mathcal{I}_c^\ell}\sum_{(j'_1,j'_2,j'_3)\in\mathcal{I}_c^\ell}\sum_{q_1}\eta^{\ell-1}_{i_1,i'_1,u_1}(\xi^{c,1}_{q_1})\eta^{\ell-1}_{j_1,j'_1,v_1}(\xi^{c,1}_{q_1})}_{=\tilde{\psi}_{(i_1,i_2,i_3),(i'_1,i'_2,i'_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}}\sum_{q_2}\eta^{\ell-1}_{i_2,i'_2,u_2}(\xi^{c,2}_{q_2})\eta^{\ell-1}_{j_2,j'_2,v_2}(\xi^{c,2}_{q_2})}_{=\mathrm{III}_{(i_1,i_2,i_3),(i'_1,i'_2,i'_3),(j_1,j_2,j_3),(j'_1,j'_2,j'_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}}\sum_{q_3}\eta^{\ell-1}_{i_3,i'_3,u_3}(\xi^{c,3}_{q_3})\eta^{\ell-1}_{j_3,j'_3,v_3}(\xi^{c,3}_{q_3})\,\nu^c_{(q_1,q_2,q_3)}}_{=\mathrm{II}_{q_1,(i_2,i_3),(i'_2,i'_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}}}_{=\mathrm{I}_{(q_1,q_2),i_3,i'_3,j_3,j'_3,u_3,v_3}}\cdot$$

$$=\psi_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)}$$

We note that the indices q_r, i_r, i'_r, j_r, j'_r, u_r and v_r, where $r \in \{1,2,3\}$, take

$$p+1 = \mathcal{O}(p)\ ,\ \mathcal{O}(1)\ ,\ \mathcal{O}(1)\ ,\ \mathcal{O}(1)\ ,\ s+1 = \mathcal{O}(p)\ ,\ \text{and}\ s+1 = \mathcal{O}(p)$$

different values, respectively. The matrix entries are evaluated in five steps:

I. We evaluate the

$$\mathcal{O}(p^2) \cdot \mathcal{O}(1) \cdot \mathcal{O}(1) \cdot \mathcal{O}(1) \cdot \mathcal{O}(1) \cdot \mathcal{O}(p) \cdot \mathcal{O}(p) = \mathcal{O}(p^4)$$

quantities

$$\mathrm{I}_{(q_1,q_2),i_3,i'_3,j_3,j'_3,u_3,v_3}$$

requiring $\mathcal{O}(p)$ flops for each one of them.

II. We evaluate the

$$\mathcal{O}(p) \cdot \mathcal{O}(1^2) \cdot \mathcal{O}(1^2) \cdot \mathcal{O}(1^2) \cdot \mathcal{O}(1^2) \cdot \mathcal{O}(p^2) \cdot \mathcal{O}(p^2) = \mathcal{O}(p^5)$$

quantities

$$\mathrm{II}_{q_1,(i_2,i_3),(i'_2,i'_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}$$

requiring $\mathcal{O}(p)$ flops for each one of them.

III. We evaluate the

$$\mathcal{O}(1^3) \cdot \mathcal{O}(1^3) \cdot \mathcal{O}(1^3) \cdot \mathcal{O}(1^3) \cdot \mathcal{O}(p^3) \cdot \mathcal{O}(p^3) = \mathcal{O}(p^6)$$

quantities

$$\mathrm{III}_{(i_1,i_2,i_3),(i'_1,i'_2,i'_3),(j_1,j_2,j_3),(j'_1,j'_2,j'_3),(u_1,u_2,u_3),(v_1,v_2,v_3)} = \mathrm{III}_{\mathbf{i},\mathbf{i'},\mathbf{j},\mathbf{j'},\mathbf{u},\mathbf{v}}$$

requiring $\mathcal{O}(p)$ flops for each one of them.

IV. We evaluate the

$$\mathcal{O}(1^3) \cdot \mathcal{O}(1^3) \cdot \mathcal{O}(1^3) \cdot \mathcal{O}(p^3) \cdot \mathcal{O}(p^3) = \mathcal{O}(p^6)$$

quantities

$$\tilde{\psi}_{(i_1,i_2,i_3),(i'_1,i'_2,i'_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)} = \tilde{\psi}_{\mathbf{i},\mathbf{i'},\mathbf{j},\mathbf{u},\mathbf{v}}$$

requiring $\mathcal{O}(1)$ flops for each one of them.

V. We evaluate the

$$\mathcal{O}(1^3) \cdot \mathcal{O}(1^3) \cdot \mathcal{O}(p^3) \cdot \mathcal{O}(p^3) = \mathcal{O}(p^6)$$

quantities

$$\psi_{(i_1,i_2,i_3),(j_1,j_2,j_3),(u_1,u_2,u_3),(v_1,v_2,v_3)} = \psi_{\mathbf{i},\mathbf{j},\mathbf{u},\mathbf{v}}$$

requiring $\mathcal{O}(1)$ flops for each one of them.

The total effort, dominated by the third step, amounts to $\mathcal{O}(p^7)$ flops.

output	parameters $(\vartheta^k_{i_k,i'_k,u_k,q_k})$	$(\delta^k_{j_k,j'_k,v_k,q_k})$	$\mathcal{I}$	$\mathcal{I}'$	$\mathcal{J}$	$\mathcal{J}'$
$\Upsilon^{\ell-1,\ell-1}_{\mathbf{i},\mathbf{j}}$	$(\beta^{\ell-1}_{i_k,u_k}(\xi^{c,k}_{q_k}))_{k,i_k,\varepsilon,u_k,q_k}$	$(\beta^{\ell-1}_{j_k,v_k}(\xi^{c,k}_{q_k}))_{k,j_k,\varepsilon,v_k,q_k}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{E}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{E}$
$\Upsilon^{\ell,\ell-1}_{\mathbf{i},\mathbf{j}}$	$(\beta^{\ell}_{i_k,u_k}(\xi^{c,k}_{q_k}))_{k,i_k,\varepsilon,u_k,q_k}$	$(\beta^{\ell-1}_{j_k,v_k}(\xi^{c,k}_{q_k}))_{k,j_k,\varepsilon,v_k,q_k}$	$\mathcal{I}_c^{\ell}$	$\mathcal{E}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{E}$
$\Upsilon^{\ell,\ell}_{\mathbf{i},\mathbf{j}}$	$(\beta^{\ell}_{i_k,u_k}(\xi^{c,k}_{q_k}))_{k,i_k,\varepsilon,u_k,q_k}$	$(\beta^{\ell}_{j_k,v_k}(\xi^{c,k}_{q_k}))_{k,j_k,\varepsilon,v_k,q_k}$	$\mathcal{I}_c^{\ell}$	$\mathcal{E}$	$\mathcal{I}_c^{\ell}$	$\mathcal{E}$
$\Phi^{\ell-1}_{\mathbf{i},\mathbf{j}}$	$(\beta^{\ell-1}_{i_k,u_k}(\xi^{c,k}_{q_k}))_{k,i_k,\varepsilon,u_k,q_k}$	$(\eta^{\ell-1}_{j_k,j'_k,v_k}(\xi^{c,k}_{q_k}))_{k,j_k,j'_k,v_k,q_k}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{E}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{I}_c^{\ell}$
$\Phi^{\ell}_{\mathbf{i},\mathbf{j}}$	$(\beta^{\ell}_{i_k,u_k}(\xi^{c,k}_{q_k}))_{k,i_k,\varepsilon,u_k,q_k}$	$(\eta^{\ell-1}_{j_k,j'_k,v_k}(\xi^{c,k}_{q_k}))_{k,j_k,j'_k,v_k,q_k}$	$\mathcal{I}_c^{\ell}$	$\mathcal{E}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{I}_c^{\ell}$
$\Psi_{\mathbf{i},\mathbf{j}}$	$(\eta^{\ell-1}_{i_k,i'_k,u_k}(\xi^{c,k}_{q_k}))_{k,i_k,i'_k,u_k,q_k}$	$(\eta^{\ell-1}_{j_k,j'_k,v_k}(\xi^{c,k}_{q_k}))_{k,j_k,j'_k,v_k,q_k}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{I}_c^{\ell}$	$\mathcal{I}_c^{\ell-1}$	$\mathcal{I}_c^{\ell}$

Table 5: The parameters of Algorithm 2 (AuxMat), cf. lines 13–21 of the matrix assembly algorithm. The indices $\mathbf{i},\mathbf{j},k,i_k,i'_k,j_k,j'_k,u_k,v_k$, and q_k vary in the index sets $\mathcal{I},\mathcal{J},\{1,2,3\},\mathsf{P}_k\mathcal{I},\mathsf{P}_k\mathcal{I}',\mathsf{P}_k\mathcal{J},\mathsf{P}_k\mathcal{J}',\mathsf{P}_k\mathcal{S}$, and $\mathsf{P}_k\mathcal{Q}$, respectively.

Algorithm 2 AuxMat – Auxiliary matrices assembly

Input: pre-computed function values:
$(\vartheta^k_{i_k,i'_k,u_k,q_k})_{k\in\{1,2,3\},i_k\in\mathsf{P}_k\mathcal{I},i'_k\in\mathsf{P}_k\mathcal{I}',u_k\in\mathsf{P}_k\mathcal{S},q_k\in\mathsf{P}_k\mathcal{G}}$,
$(\delta^k_{j_k,j'_k,v_k,q_k})_{k\in\{1,2,3\},j_k\in\mathsf{P}_k\mathcal{J},j'_k\in\mathsf{P}_k\mathcal{J}',v_k\in\mathsf{P}_k\mathcal{S},q_k\in\mathsf{P}_k\mathcal{G}}$;
index sets: $\mathcal{I}$, $\mathcal{I}'$, $\mathcal{J}$, $\mathcal{J}'$
for $(q_1,q_2)\in\mathsf{P}_{12}\mathcal{G}$ **do** ▷ Step I - first loop
  for $i_3\in\mathsf{P}_3\mathcal{I}$, $j_3\in\mathsf{P}_3\mathcal{J}$ **do**
    for $i'_3\in\mathsf{P}_3\mathcal{I}'$ $j'_3\in\mathsf{P}_3\mathcal{J}'$ **do**
      for $u_3\in\mathsf{P}_3\mathcal{S}$, $v_3\in\mathsf{P}_3\mathcal{S}$ **do**
        $\mathrm{I}_{q_1,q_2,i_3,i'_3,j_3,j'_3,u_3,v_3}\leftarrow 0$
        for $q_3\in\mathsf{P}_3\mathcal{G}$ **do**
          $\mathrm{I}_{q_1,q_2,i_3,i'_3,j_3,j'_3,u_3,v_3}\mathrel{+}=\vartheta^3_{i_3,i'_3,u_3,q_3}\,\delta^3_{j_3,j'_3,v_3,q_3}\,\nu^c_{(q_1,q_2,q_3)}$
for $q_1\in\mathsf{P}_1\mathcal{G}$ **do** ▷ Step II - second loop
  for $(i_2,i_3)\in\mathsf{P}_{23}\mathcal{I}$, $(j_2,j_3)\in\mathsf{P}_{23}\mathcal{J}$ **do**
    for $(i'_2,i'_3)\in\mathsf{P}_{23}\mathcal{I}'$, $(j'_2,j'_3)\in\mathsf{P}_{23}\mathcal{J}'$ **do**
      for $(u_2,u_3)\in\mathsf{P}_{23}\mathcal{S}$, $(v_2,v_3)\in\mathsf{P}_{23}\mathcal{S}$ **do**
        $\mathrm{II}_{q_1,(i_2,i_3),(i'_2,i'_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}\leftarrow 0$
        for $q_2\in\mathsf{P}_2\mathcal{G}$ **do**
          $\mathrm{II}_{q_1,(i_2,i_3),(i'_2,i'_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}\mathrel{+}=\vartheta^2_{i_2,i'_2,u_2,q_2}\,\delta^2_{j_2,j'_2,v_2,q_2}\,\mathrm{I}_{q_1,q_2,i_3,i'_3,j_3,j'_3,u_3,v_3}$
for $\mathbf{i}\in\mathcal{I}$, $\mathbf{j}\in\mathcal{J}$ **do** ▷ Step III - third loop
  for $\mathbf{i}'\in\mathcal{I}'$, $\mathbf{j}'\in\mathcal{J}'$ **do**
    for $\mathbf{u}\in\mathcal{S}$, $\mathbf{v}\in\mathcal{S}$ **do**
      $\mathrm{III}_{\mathbf{i},\mathbf{i}',\mathbf{j},\mathbf{j}',\mathbf{u},\mathbf{v}}\leftarrow 0$
      for $q_1\in\mathsf{P}_1\mathcal{G}$ **do**
        $\mathrm{III}_{\mathbf{i},\mathbf{i}',\mathbf{j},\mathbf{j}',\mathbf{u},\mathbf{v}}\mathrel{+}=\vartheta^1_{i_1,i'_1,u_1,q_1}\,\delta^1_{j_1,j'_1,v_1,q_1}\,\mathrm{II}_{q_1,(i_2,i_3),(i'_2,i'_3),(j_2,j_3),(j'_2,j'_3),(u_2,u_3),(v_2,v_3)}$
for $\mathbf{i}\in\mathcal{I}$, $\mathbf{j}\in\mathcal{J}$ **do** ▷ Steps IV and V
  $\mathrm{X}_{\mathbf{i},\mathbf{j}}\leftarrow(0)_{\mathbf{u},\mathbf{v}\in\mathcal{S}}$
  for $\mathbf{i}'\in\mathcal{I}'$, $\mathbf{j}'\in\mathcal{J}'$ **do**
    $\mathrm{X}_{\mathbf{i},\mathbf{j}}\mathrel{+}=(\mathrm{III}_{\mathbf{i},\mathbf{i}',\mathbf{j},\mathbf{j}',\mathbf{u},\mathbf{v}})_{\mathbf{u},\mathbf{v}\in\mathcal{S}}$
return $(\mathrm{X}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}\in\mathcal{I},\mathbf{j}\in\mathcal{J}}$ ▷ **Output:** Auxiliary Matrices

Appendix D: The algorithm for assembling the auxiliary matrices

All the auxiliary matrix entries can be computed by a single procedure listed in Algorithm 2. This algorithm is called with the parameters listed in Table 5.

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. 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] M. Carraturo, M. Torre, C. Giannelli, and A. Reali. An isogeometric approach to coupled thermomechanics in 3D via hierarchical adaptivity. *Computers & Mathematics with Applications*, 162:133–144, 2024.

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

[11] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM Journal on Numerical Analysis*, 33(3):1106–1124, 1996.

[12] F. Fahrendorf, L. De Lorenzis, and H. Gomez. Reduced integration at superconvergent points in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 328:390–410, 2018.

[13] 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.

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

[15] C. Giannelli, T. Kanduč, M. Martinelli, G. Sangalli, and M. Tani. Weighted quadrature for hierarchical B-splines. *Computer Methods in Applied Mechanics and Engineering*, 400:115465, 2022.

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

[17] 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.

[18] 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.

[19] 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.

[20] S. Hou, D. Wang, A. Shen, and Z. Wu. A superconvergent point quadrature with optimized weights for efficient Galerkin isogeometric analysis of thin plates and shells. *Computer Methods in Applied Mechanics and Engineering*, 461:119235, 2026.

[21] S. Hou, X. Xu, D. Wang, A. Shen, and Z. Sun. Equivalent and augmented explicit Greville quadrature for isogeometric analysis. *Engineering with Computers*, 42(1):36, 2026.

[22] 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.

[23] 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.

[24] K.A. Johannessen. Optimal quadrature for univariate and tensor product splines. *Computer Methods in Applied Mechanics and Engineering*, 316:84–99, 2017.

[25] B. Jüttler, H. Kang, D. Mokriš, and M. Pan. Bases for C^r-smooth polynomial splines of degree $p = 2r + 1$ over hierarchical T-meshes refined by cross insertion. In *INdAM Meeting: Fast Methods for Isogeometric Analysis*. Springer, 2026. accepted for publication.

[26] B. Jüttler, U. Langer, A. Mantzaflaris, S.E. Moore, and W. Zulehner. Geometry+ simulation modules: Implementing isogeometric analysis. *PAMM*, 14(1):961–962, 2014.

[27] A Karatarakis, P Karakitsios, and M Papadrakakis. GPU accelerated computation of the isogeometric analysis stiffness matrix. *Computer Methods in Applied Mechanics and Engineering*, 269:334–355, 2014.

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

[29] G. Li and H. Lin. Collocation and mass matrix in least-squares isogeometric analysis. *Computer-Aided Design*, 192:104026, 2025.

[30] L. De Lorenzis, J.A. Evans, T.J.R. Hughes, and A. Reali. Isogeometric collocation: Neumann boundary conditions and contact. *Computer Methods in Applied Mechanics and Engineering*, 284:21–54, 2015.

[31] 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.

[32] 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.

[33] B. Marussig, R. Hiemstra, and D. Schillinger. Fast immersed boundary method based on weighted quadrature. *Computer Methods in Applied Mechanics and Engineering*, 417:116397, 2023.

[34] M. Montardini, G. Sangalli, and L. Tamellini. Optimal-order isogeometric collocation at Galerkin superconvergent points. *Computer Methods in Applied Mechanics and Engineering*, 316:741–757, 2017.

[35] 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.

[36] M. Pan, B. Jüttler, and A. Mantzaflaris. Efficient matrix assembly in isogeometric analysis with hierarchical B-splines. *Journal of Computational and Applied Mathematics*, 390:113278, 2021.

[37] M. Pan, B. Jüttler, and F. Scholz. Efficient matrix computation for isogeometric discretizations with hierarchical B-splines in any dimension. *Computer Methods in Applied Mechanics and Engineering*, 388:114210, 2022.

[38] T.-C. Riemer and M. Stoll. A low-rank tensor framework for hierarchical refinement in IgA. *Computer Methods in Applied Mechanics and Engineering*, 455:118921, 2026.

[39] 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.

[40] T. Teijeiro, J.M. Taylor, A. Hashemian, and D. Pardo. Machine learning discovery of optimal quadrature rules for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 416:116310, 2023.

[41] L. Venta-Viñuela, H.M. Verhelst, A. Mantzaflaris, C. Giannelli, and A. Reali. Effective phase-field modeling of the Cahn–Hilliard equation in 2D and 3D using a quasi-interpolation approach to adaptive isogeometric analysis with THB-splines. *Computer Methods in Applied Mechanics and Engineering*, 455:118855, 2026.

[42] Z. Zou, T.J.R. Hughes, M.A. Scott, Di Miao, and R.A. Sauer. Efficient and robust quadratures for isogeometric analysis: Reduced Gauss and Gauss-Greville rules. *Computer Methods in Applied Mechanics and Engineering*, 392:114722, 2022.