Sum-factorization techniques in Isogeometric Analysis

Andrea Bressan, Stefan Takacs

Sum-factorization techniques in Isogeometric Analysis

A. Bressan*and S. Takacs†

September 14, 2018

Abstract

The fast assembling of stiffness and mass matrices is a key issue in isogeometric analysis, particularly if the spline degree is increased. If the assembling is done in a naive way, the computational complexity grows with the spline degree to a power of $3d$, where d is the spacial dimension. Recently much progress was achieved in improving the assembling procedures, particularly in the methods of sum factorization, low rank assembling, and weighted quadrature. A few years ago, it was shown that the computational complexity of the sum factorization approach grows with the spline degree to a power of $2d+1$. We show that it is possible to decrease this to a power of $d+2$ without loosing generality or accuracy.

1 Introduction

Isogeometric Analysis, [10], was proposed around a decade ago as a new approach for the discretization of partial differential equations (PDEs) and has gained much interest since than. Spline spaces, such as spaces spanned by tensor product B-splines or NURBS, are typically used for geometry representation in standard CAD systems. In Isogeometric Analysis, one uses such spaces for the geometry representation of the computational domain and as space of ansatz functions for solution of the PDE.

The fast assembling of stiffness and mass matrices is a key issue in Isogeometric Analysis, particularly if the spline degree is increased. If the assembling is done in a naive way, and p^d quadrature points are used per element, the computational complexity of assembling a standard mass or stiffness matrix has order Np^{3d} , where N is the number of unknowns, p is the spline order (degree + 1), d is the domain dimension. In recent years, much effort was set on proposing

*Department of MathematicsUniversity of Oslo, andbres@math.uio.no

†RICAM, Austrian Academy of Sciences, Linz, Austriastefan.takacs@ricam.oeaw.ac.at

faster assembling schemes; we want to name particularly the methods of *sum factorization*, *low rank assembling*, and *weighted quadrature*.

Sum factorization was originally proposed for spectral methods and later applied to high-order finite element methods [13, 1]. Antolín, Buffa, Calabro, Martinelli, and Sangalli [2] have carried over this approach to the case of Isogeometric Analysis and have shown that the computational complexity of assembling a standard mass of stiffness matrix can be reduced to order Np^{2d+1}. The idea of this approach is to perform the required computational steps in a smart way and to reuse already computed quantities accordingly.

Authors from the same group have then further reduced the computational complexity by *weighted quadrature*, cf. the publication by Calabrò, Sangalli, and Tani [7]. Here, the idea is to reduce the number of quadrature points by setting up appropriately adjusted quadrature rules. This allows to assemble a standard mass or stiffness matrix with a computational complexity of order Np^{d+1}. This, however, comes with the cost that the resulting matrix is non-symmetric and a careful analysis is necessary to show that the overall discretization satisfies the error bounds that are typically expected for isogeometric discretizations.

Low rank assembling is based on a completely different idea. It is observed that for practical problems, evaluation of the geometry function on the quadrature points does not have the maximum possible rank. Note that often the geometry is already represented exactly on coarse grid levels and the grid is only refined for a better resolution of the solution of the PDE. In this case, the rank of the geometry function is unchanged during refinement and is small compared to the maximum possible rank. It was observed that low rank geometry functions yield to mass and stiffness matrices which typically have a small tensor rank. Mantzaflaris, Jüttler, Khoromskij, and Langer [12] have discussed how to set up an assembling algorithm based on this approach. Hofreither [9] has shown that by rewriting the problem accordingly, standard adaptive cross approximation algorithms can be used as black-box methods to determine the mass or stiffness matrices. For these approaches, the overall computational complexity has order NRp^d, where R is the (unknown) tensor-rank of the resulting matrix. If the coefficients of the matrix are not actually needed, it is possible to compute a low-tensor rank representation of the matrix and to compute the corresponding matrix-vector products with a computational complexity of order NRp.

In this note, we want to have again a look onto the approach of sum factorization. We will show that by performing a *global variant of sum factorization,* the computational complexity is reduced to order Np^{d+2}. Still, this algorithm yields exactly the same matrix which would be obtained with straight-forward assembling and the algorithm is as general and as extendable to other differential equation as straight-forward assembling. We will then see that the computational complexity is still preserved for a localized approach. We will observe that also here, like for the low rank assembling, the formation of the matrix is the most expensive part and that the corresponding matrix-vector products has a computational complexity of order Np^{d+1}.

This article is organized as follows. In Section 2, we state the abstract formulation of the problem and give examples of bilinear forms falling into the class. We introduce a fast assembling procedure in Section 3. In Section 4, we discuss the localization of sum factorization and its applications. Then, in Section 5, we give an algebraic description of the proposed algorithm. In Section 6, we explain why the evaluation of the basis functions and the geometry function is non trivial. We comment on matrix-free implementations in Section 7. Finally, in Section 8 we give numerical experiments and draw conclusions.

2 Preliminaries

In the following L^2, L^∞ and H^1 denote the standard Lebesgue and Sobolev spaces with standard scalar products. Consider a bilinear form of the type

$$\begin{aligned} &a : L^2([0,1]^d) \times L^2([0,1]^d) \to \mathbb{R}, \\ &a(u,v) := (\mathcal{F}u, v)_{L^2([0,1]^d)} = \int_{[0,1]^d} \mathcal{F}(\mathbf{x}) u(\mathbf{x}) v(\mathbf{x})\, d\mathbf{x}, \end{aligned} \tag{1}$$

where $d \in \mathbb{N}$ is the domain dimension and $\mathcal{F} \in L^\infty([0,1]^d)$ is a given coefficient function. At a first glance, this model problem looks rather restrictive as it is only an L^2-like scalar product. However, we will see in Section 2.1 that we can write the variational form of any PDE, like Poisson, or system, like Stokes, as a sum of bilinear operators of the form (1).

The goal of this article is to formulate fast algorithms that compute the matrix corresponding to a Petrov-Galerkin discretization of a. (This, of course, also covers the special case of standard Galerkin discretizations.) Let $\Phi = (\phi_n)_{n=1}^N, \Psi = (\psi_m)_{m=1}^M$ be two generating systems of functions that span corresponding subspaces of $L^2([0,1]^d)$. Following the Petrov-Galerkin approach, the restriction of a to $\operatorname{span}\Phi \times \operatorname{span}\Psi$ is represented by the matrix

$$A := [a(\phi_n, \psi_m)]_{n=1,\dots,N}^{m=1,\dots,M}. \tag{2}$$

In practice, the coefficients of A are not computed exactly, but they are approximated using a quadrature rule. This means that A is approximated by

$$\mathtt{A} := [\mathtt{a}(\phi_n, \psi_m)]_{n=1,\dots,N}^{m=1,\dots,M}, \tag{3}$$

with

$$\mathtt{a}(\phi_n, \psi_m) := \sum_{\mathbf{x}\in\mathbb{X}} \omega(\mathbf{x})\, \mathcal{F}(\mathbf{x})\, \phi_n(\mathbf{x})\, \psi_m(\mathbf{x}), \tag{4}$$

where $\mathbb{X}$ is the set of quadrature points and $\omega(\mathbf{x})$ are the corresponding quadrature weights. The techniques in this note require a tensor-product structure.

- *Tensor-product discretization:* For each direction $\delta = 1, \dots, d$, there exist Φ_δ, Ψ_δ such that

$$\begin{cases} \Phi = \Phi_1 \otimes \ldots \otimes \Phi_d, & \Phi_\delta = (\phi_n^{\overline{z}\delta})_{n=1}^{N_\delta}, \\ \Psi = \Psi_1 \otimes \ldots \otimes \Psi_d, & \Psi_\delta = (\psi_m^{\overline{z}\delta})_{m=1}^{M_\delta}, \end{cases} \tag{5}$$

- *Tensor-product quadrature:* For each direction $\delta = 1, \dots, d$, there exist $\mathbb{X}_\delta$ and ω_δ such that

$$\begin{cases} \mathbb{X} = \mathbb{X}_1 \times \ldots \times \mathbb{X}_d, \\ \omega(\mathbf{x}) = \omega_1(x_1) \ldots \omega_d(x_d). \end{cases} \tag{6}$$

The meaning of (5) is that the d-variate functions in Φ are the products of the univariate functions in $\Phi_1, \dots, \Phi_d$ (and analogously the functions in Ψ). By convention, we assume the lexicographic ordering. For a rigorous definition, let $\pi : \{1, \dots, N\} \to \{1, \dots, N_1\} \times \cdots \times \{1, \dots, N_d\}$ be a lexicographic ordering, i.e., the bijection defined by

$$\pi := (\pi_1, \dots, \pi_d), \quad \pi_\delta(n) := \left\lfloor \frac{(n-1) \bmod N^{\lesssim \delta}}{N^{\lesssim \delta - 1}} \right\rfloor + 1, \quad N^{\lesssim \delta} := \prod_{i=1}^{\delta} N_i.$$

Define $\sigma : \{1, \dots, M\} \to \{1, \dots, M_1\} \times \cdots \times \{1, \dots, M_d\}$ analogously. Using these orderings, we denote the generating functions for Φ and Ψ as follows:

$$\begin{cases} \phi_n(x_1, \dots, x_d) = \phi_{\pi_1(n)}^{\overline{z}1}(x_1) \cdots \phi_{\pi_d(n)}^{\overline{z}d}(x_d), & n = 1, \dots, N, \\ \psi_m(x_1, \dots, x_d) = \psi_{\sigma_1(m)}^{\overline{z}1}(x_1) \cdots \psi_{\sigma_d(m)}^{\overline{z}d}(x_d), & m = 1, \dots, M. \end{cases}$$

Analogous to $N^{\lesssim \delta}$, it is convenient to define

$$M^{\lesssim \delta} := \prod_{i=1}^{\delta} M_i \qquad \text{and} \qquad \mathbb{X}^{\lesssim \delta} := \bigtimes_{i=1}^{\delta} \mathbb{X}_i.$$

Moreover, for the complexity analysis, we need some bounds on the supports of the functions in Φ_δ and Ψ_δ. As usual in the context of Sobolev spaces, supports are always closed, i.e., we define $\operatorname{supp} f := \overline{\{x : f(x) \neq 0\}}$.

- *Bounded number of active functions*: for all $\delta \in \{1, \dots, d\}$ and $x_\delta \in \mathbb{X}_\delta$:

$$\begin{cases} \#\{n \,:\, x_\delta \in \operatorname{supp} \phi_n^{\overline{z}\delta}\} \leq p_\delta, \\ \#\{m \,:\, x_\delta \in \operatorname{supp} \psi_m^{\overline{z}\delta}\} \leq p_\delta. \end{cases} \tag{7}$$

- *Bounded overlap*: for all $\delta \in \{1, \dots, d\}$ and all $\phi_n^{\overline{z}\delta} \in \Phi_\delta$:

$$\#\{m : \operatorname{supp} \phi_n^{\overline{z}\delta} \cap \operatorname{supp} \psi_m^{\overline{z}\delta} \cap \mathbb{X}_\delta \neq \emptyset\} \leq k_\delta \, p_\delta. \tag{8}$$

For the convenience of the reader, we will often express the main results in simplified form using $p := \max\{p_1, \dots, p_d\}$.
Remark 1. The condition (8) bounds the number of test functions that interact with any of the trial functions; the presented complexity analysis could also be done if the roles of the trial functions and the test functions as well as of N and M are interchanged.

To keep the notation tight, here and in what follows, $a \lesssim b$ means that there is a constant $c > 0$ (which does not depend on any of the quantities discussed in this paper) such that $a \leq c\, b$ and $a \eqsim b$ means that $a \lesssim b$ and $b \lesssim a$.

2.1 Bilinear forms that depend on derivatives

The described algorithms are not limited to the computation of the mass matrix, but can be employed for Isogeometric Analysis of linear and non-linear PDEs or systems of PDEs. In this subsection we show how an assembling procedure for Petrov-Galerkin discretization of bilinear form of (1) extends naturally to bilinear forms that depend on the derivatives of their arguments.

Any bilinear form $a : H^r([0,1]^d) \times H^r([0,1]^d) \to \mathbb{R}$ can be written as:

$$a(\phi, \psi) = \sum_{\theta,\eta \in \Theta_r} (\mathcal{F}_{\theta,\eta} \partial_\theta \phi, \partial_\eta \psi)_{L^2} =: \sum_{\theta,\eta \in \Theta_r} a_{\theta,\eta}(\partial_\theta \phi, \partial_\eta \psi), \tag{9}$$

where $\Theta_r \subset \mathbb{N}^d$ is the set of multiindices $\theta = (\theta_1, \dots, \theta_d)$ corresponding to partial derivatives of order up to r. Analogously, the matrix $\mathtt{A}$ representing its restriction to $\Phi \times \Psi$ decomposes as

$$\mathtt{A} = \sum_{\theta,\eta \in \Theta_r} \mathtt{A}_{\theta,\eta},$$

where $\mathtt{A}_{\theta,\eta}$ is the matrix representing the restriction of $a_{\theta,\eta}$ to $\partial_\theta \Phi \times \partial_\eta \Psi$ with

$$\begin{cases} \partial_\theta \Phi := (\frac{\partial^{\theta_1}}{\partial x^{\theta_1}} \cdots \frac{\partial^{\theta_d}}{\partial x^{\theta_d}} \phi_n)_{n=1}^N, \\ \partial_\eta \Psi := (\frac{\partial^{\eta_1}}{\partial x^{\eta_1}} \cdots \frac{\partial^{\eta_d}}{\partial x^{\eta_d}} \psi_m)_{m=1}^M. \end{cases}$$

To extend any assembling procedure from a as in (1) to a as in (9), it is sufficient to prove that the assumptions on Φ and Ψ hold for the associated spaces $\partial_\theta \Phi$ and $\partial_\eta \Psi$ so that the procedure can be applied on each $a_{\theta,\eta}$. This is done in the next lemma.

Lemma 1. *If $\Phi, \Psi \subset H^r([0,1]^d)$ satisfy any of the conditions* (5), (7), (8) *then also $\partial_\eta \Phi, \partial_\theta \Psi$ satisfies that condition for all $\theta, \eta \in \Theta_r$.*

Proof. First of all the tensor product structure carries over because

$$\partial_\theta \Phi = (\tfrac{\partial^{\theta_1}}{\partial x^{\theta_1}} \cdots \tfrac{\partial^{\theta_d}}{\partial x^{\theta_d}} \phi_n)_{n=1}^N = \partial_{\theta_1} \Phi_1 \otimes \cdots \otimes \partial_{\theta_d} \Phi_d,$$

where $\partial_{\theta_\delta}\Phi_\delta := (\frac{\partial^{\theta_\delta}}{\partial x^{\theta_\delta}}\phi_n^{\vec{\ }\delta})_{n=1}^{N_\delta}$. The same applies to $\partial_\eta\Psi$. For a sufficiently smooth function $f : [0,1] \to \mathbb{R}$ and using that the supports are closed, we have $\operatorname{supp}\frac{\partial}{\partial x}f \subseteq \operatorname{supp} f$. In particular, we deduce

$$\operatorname{supp}\partial_{\theta_\delta}\phi^{\vec{\ }\delta} \subseteq \operatorname{supp}\phi^{\vec{\ }\delta}, \qquad \operatorname{supp}\partial_{\eta_\delta}\psi^{\vec{\ }\delta} \subseteq \operatorname{supp}\psi^{\vec{\ }\delta},$$

so that (7) and (8) carry over to $\partial_\eta\Phi, \partial_\theta\Psi$ with the same values of p_δ and k_δ. □

Remark 2. Note that considering a Petrov-Galerkin approach (different spaces for test and trial functions) is required for the application of the algorithms to higher order problems *even if* $\Phi = \Psi$ because for any $\theta \neq \eta$, typically the two spaces $\partial_\theta\Phi$ and $\partial_\eta\Psi$ differ.

2.2 Examples: B-spline based discretizations

Lemma 2. *If $\Phi_\delta := (\phi_n^{\vec{\ }\delta})_{n=1}^{N_\delta}$ and $\Psi_\delta := (\psi_m^{\vec{\ }\delta})_{m=1}^{M_\delta}$ are univariate B-spline bases of order not more than p_δ (degree not more than $p_\delta - 1$) and sharing the same knot vectors such that no knot coincides with a quadrature point in $\mathbb{X}_\delta$, then* (7) *and* (8) *hold with $k_\delta = 2$.*

The same holds if the knot vector of Φ_δ is a refinement of the knot vector of Ψ_δ, i.e., it contains additional knots or some knots have a larger multiplicity.

Proof. Let $q_{\Phi,\delta}$ and $q_{\Psi,\delta}$ the orders of Φ_δ and Ψ_δ, respectively. Denote the knot vector of Φ_δ by $\Xi := (\xi_{-q_{\Phi,\delta}+1}, \dots, \xi_{N_\delta+q_{\Phi,\delta}-1})$. Note that $\operatorname{supp}\phi_n^{\vec{\ }\delta} = [\xi_n, \xi_{n+q_{\Phi,\delta}}]$. Consider some quadrature point $x \in (\xi_j, \xi_{j+1})$ and observe that the supports only of the functions $\phi_{j+1-q_{\Phi,\delta}}^{\vec{\ }\delta}, \dots, \phi_j^{\vec{\ }\delta}$ include the quadrature point x, which are $q_{\Phi,\delta}$ functions. The same arguments hold for Ψ_δ. This shows (7). Observe that the supports only of the functions $\psi_{n-q_{\Psi,\delta}+1}^{\vec{\ }\delta}, \dots, \psi_{n+q_{\Phi,\delta}}^{\vec{\ }\delta}$ contribute to $(\xi_n, \xi_{n+q_{\Phi,\delta}}) \supseteq \operatorname{supp}\phi_n^{\vec{\ }\delta} \cap \mathbb{X}_\delta$. This shows condition (8) with $k_\delta = 2$.

If now the functions in Φ_δ are based on a refinement of Ξ, we know that the supports of the functions in Φ_δ shrink. More precisely, we have $\operatorname{supp}\phi_n^{\vec{\ }\delta} \cap \mathbb{X}_\delta \subseteq (\xi_{n'}, \xi_{n'+q_{\Phi,\delta}})$ for some accordingly chosen n'. So, (8) stays true. □

Remark 3. If some quadrature points coincide with a knot, we obtain an analogous result with $k_\delta = 3$ if only B-splines with orders up to $p_\delta - 1$ are considered.

Example 1 (Convection diffusion equation)**.** We consider a standard single-patch isogeometric discretization, so we assume that the computational domain Ω is parameterized by a diffeomorphism

$$\mathbf{G} : \widehat{\Omega} := [0,1]^d \to \Omega := \mathbf{G}(\widehat{\Omega}) \subset \mathbb{R}^d.$$

We assume that a source function $f \in L^2(\Omega)$ and coefficient-functions $A \in L^\infty(\Omega, \mathbb{R}^{d\times d})$, $\mathbf{b} \in L^\infty(\Omega, \mathbb{R}^d)$ and $c \in L^\infty(\Omega)$ are given. The boundary value problem reads as follows. Find $u \in H^1(\Omega)$ such that

$$-\nabla\cdot(A\nabla u) + \mathbf{b}\cdot\nabla u + cu = f \quad \text{in } \Omega, \qquad \frac{\partial u}{\partial n} = 0 \quad \text{on } \partial\Omega.$$

The variational formulation reads as follows. Find $u \in H^1(\Omega)$ such that

$$\mathrm{a}(u,v) := \Big(\begin{bmatrix} c & 0 \\ \mathbf{b} & A \end{bmatrix} \begin{bmatrix} u \\ \nabla u \end{bmatrix}, \begin{bmatrix} v \\ \nabla v \end{bmatrix} \Big)_{L_2(\Omega)} = (f,v)_{L_2(\Omega)} \text{ for all } v \in H^1(\Omega).$$

A standard isogeometric discretization is set up on the parameter domain, i.e., we first define the spline space

$$\widehat{V}_h := \operatorname{span}\{\widehat{B}_1, \dots, \widehat{B}_N\} := \bigotimes_{\delta=1}^{d} \operatorname{span}\{\widehat{B}_1^{(\delta)}, \dots, \widehat{B}_{N_\delta}^{(\delta)}\} \subset H^1(\widehat{\Omega}),$$

where $\widehat{B}_n^{(\delta)}$ are the standard B-spline basis functions as given by the Cox-de Boor formula. Then, the ansatz functions are transferred to the physical domain using the pull-back principle

$$V_h := \widehat{V}_h \circ \mathbf{G}^{-1}, \quad B_n := \widehat{B}_n \circ \mathbf{G}^{-1}.$$

For the computation of the stiffness matrix $\mathtt{A}$, we transfer the functions of interest to the parameter domain and obtain

$$\begin{aligned}
\mathtt{A} &= [\mathtt{a}(B_n, B_m)]_{n=1\dots,N}^{m=1,\dots,N} \\
&= \Big[\Big(\underbrace{|J_{\mathbf{G}}| \begin{bmatrix} I & \\ & J_{\mathbf{G}}^{-\top} \end{bmatrix} \begin{bmatrix} \widehat{c} & 0 \\ \widehat{\mathbf{b}} & \widehat{A} \end{bmatrix} \begin{bmatrix} I & \\ & J_{\mathbf{G}}^{-1} \end{bmatrix}}_{\mathcal{F} :=} \underbrace{\begin{bmatrix} 1 \\ \nabla \end{bmatrix}}_{\mathcal{D} :=} \widehat{B}_n, \begin{bmatrix} 1 \\ \nabla \end{bmatrix} \widehat{B}_m \Big)_{L_2(\widehat{\Omega})} \Big]_n^m \\
&= \sum_{\theta=0}^{d} \sum_{\eta=0}^{d} \Big[\underbrace{(\mathcal{F}_{\theta,\eta} \mathcal{D}_\theta \widehat{B}_n, \mathcal{D}_\eta \widehat{B}_m)_{L_2(\widehat{\Omega})}}_{\mathtt{a}_{\theta,\eta}(\mathcal{D}_\theta \widehat{B}_n, \mathcal{D}_\eta \widehat{B}_m) :=} \Big]_{n=1,\dots,N}^{m=1,\dots,N},
\end{aligned} \tag{10}$$

where $J_{\mathbf{G}}$ is the Jacobi matrix of the geometry function, $|J_{\mathbf{G}}|$ the absolute value of its determinant and

$$\mathcal{D}_\alpha := \begin{cases} 1 & \text{for } \alpha = 0 \\ \frac{\partial}{\partial x_\alpha} & \text{for } \alpha \in \{1, \dots, d\}. \end{cases}$$

Note that for each (θ, η), $\partial_\theta \widehat{B}_n$ and $\partial_\eta \widehat{B}_m$ are tensor-products of B-spline basis functions and/or their derivatives. Using Lemmas 1 and 2, we obtain that $(\partial_\theta \widehat{B}_n, \partial_\eta \widehat{B}_m)$ satisfies (7) and (8). So, we immediately observe that the computation of $A_{\theta,\eta,h}$ belongs to the abstract problem (1) – (2) with $\mathcal{F} := \mathcal{F}_{\theta,\eta}$.

Example 2 (Stokes equation)**.** As a second example, consider the Stokes system, which reads in variational form as follows

$$\begin{cases} (\nabla u, \nabla w)_{L^2(\Omega)} + (\nabla p, w)_{L^2(\Omega)} = (f, w)_{L^2(\Omega)} \\ (\nabla \cdot u, q)_{L^2(\Omega)} \qquad\qquad\qquad\quad = 0 \end{cases}. \tag{11}$$

A Galerkin discretization of such a system yields a block-matrix, which looks for two dimensions as follows

$$\mathtt{A} = \begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & B_1^\top \\ A_{21} & A_{22} & B_2^\top \\ B_1 & B_2 & 0 \end{bmatrix}.$$

Each of the blocks A_{ij} and B_i is obtained by a Galerkin discretization of a scalar valued differential equation, which can be treated analogously to Example 1.

The discretization has to be chosen such that inf-sup stability holds; in [4], inf-sup stability has been shown for the isogeometric Taylor-Hood and the isogeometric sub-grid method.

In the *isogeometric Taylor-Hood method*, the pressure variable is discretized with B-splines of some order p with maximum smoothness. Each of the components of the velocity variable is discretized with B-splines of order $p+1$ and reduced smoothness, $p-2$, obtained by taking each knot in the knot vector twice.

In the *isogeometric sub-grid method*, the pressure variable is again discretized with B-splines of some order p with maximum smoothness. Each of the components of the velocity variable is discretized with B-splines of order $p+1$ with maximum smoothness on a grid obtained from the pressure grid by one uniform refinement step.

Here, the idea of sum factorization can be employed for each of the blocks of the overall system, where due to symmetry it suffices to discretize the Laplace operator A and the divergence operator B. The conditions of Lemmas 1 and 2 hold for A because $\Phi = \Psi$, i.e., the spaces of the trial and the test functions coincide; they hold for B because the knot vector of the trial function space Φ is a refinement of that of the space of test functions Ψ.

3 Sum factorization

The method of sum factorization is based on the observation that expanding each term in (4) with respect to its components yields

$$\begin{aligned}
\mathtt{a}(\phi_n, \psi_m) &= \sum_{x_1 \in \mathbb{X}_1} \cdots \sum_{x_d \in \mathbb{X}_d} \prod_{\delta=1}^{d} \omega_\delta(x_\delta) \prod_{\delta=1}^{d} \phi^{\bar{\delta}}_{\pi_\delta(n)}(x_\delta) \prod_{\delta=1}^{d} \psi^{\bar{\delta}}_{\sigma_\delta(m)}(x_\delta)\, \mathcal{F}(\mathbf{x}) \\
&= \sum_{x_d \in \mathbb{X}_d} \phi^{\bar{d}}_{\pi_d(n)}(x_d)\, \psi^{\bar{d}}_{\sigma_d(m)}(x_d) \omega_d(x_d) \\
&\qquad \underbrace{\sum_{x_1 \in \mathbb{X}_1} \cdots \sum_{x_{d-1} \in \mathbb{X}_{d-1}} \prod_{\delta=1}^{d-1} \omega_\delta(x_\delta) \prod_{\delta=1}^{d-1} \phi^{\bar{\delta}}_{\pi_\delta(n)}(x_\delta) \prod_{\delta=1}^{d-1} \psi^{\bar{\delta}}_{\sigma_\delta(m)}(x_\delta)\, \mathcal{F}(\mathbf{x})}_{=:\, \mathtt{a}^{<d-1}_{x_d}(\phi_n^{<d-1}, \psi_m^{<d-1})},
\end{aligned}$$

that is

$$\mathtt{a}(\phi_n, \psi_m) = \sum_{x_d \in \mathbb{X}_d} \phi^{\overline{\leq} d}_{\pi_d(n)}(x_d)\, \psi^{\overline{\leq} d}_{\sigma_d(m)}(x_d)\omega_d(x_d)\ \mathtt{a}^{\leq d-1}_{x_d}(\phi^{\leq d-1}_n, \psi^{\leq d-1}_m), \qquad (12)$$

where

$$\phi^{\leq \delta}_n := \prod_{i=1}^{\delta} \phi^{\overline{\leq} i}_{\pi_i(n)} \qquad \text{and} \qquad \psi^{\leq \delta}_m := \prod_{i=1}^{\delta} \phi^{\overline{\leq} i}_{\sigma_i(m)}.$$

The term $\mathtt{a}^{\leq d-1}_{x_d}(\phi^{\leq d-1}_n, \psi^{\leq d-1}_m)$ is independent of the d-th components of Φ, Ψ, ω and it appears for many (ϕ_n, ψ_m) pairs. The advantage of (12) over (4) is that it shows that $\mathtt{a}^{\leq d-1}_{x_d}(\phi^{\leq d-1}_n, \psi^{\leq d-1}_m)$ can be computed once and used many times. By rewriting $\mathtt{a}^{\leq d-1}_{x_d}(\phi^{\leq d-1}_n, \psi^{\leq d-1}_m)$ as

$$\mathtt{a}^{\leq d-1}_{x_d}(\phi^{\leq d-1}_m, \psi^{\leq d-1}_n) = \sum_{\mathbf{x} \in \mathbb{X}^{\leq d-1}} \omega^{\leq d-1}(\mathbf{x})\phi^{\leq d-1}_n(\mathbf{x})\psi^{\leq d-1}_m(\mathbf{x})\mathcal{F}(\mathbf{x}, x_d), \qquad (13)$$

we see that, analogously to (4), it is an approximation with the quadrature $(\mathbb{X}^{\leq d-1}, \omega^{\leq d-1})$ of the bilinear form on $L^2([0,1]^{d-1})$ defined by

$$a^{\leq d-1}_{x_d}(\phi, \psi) = \int_{[0,1]^{d-1}} \phi(\mathbf{x})\psi(\mathbf{x})\mathcal{F}(\mathbf{x}, x_d)\, d\mathbf{x}.$$

Let $\mathtt{A}^{\leq d-1}_{x_d}$ be the matrix representing the Petrov-Galerkin restriction of $a^{\leq d-1}_{x_d}$ to span $\Phi^{\leq d-1} \times$ span $\Psi^{\leq d-1}$, i.e.,

$$\mathtt{A}^{\leq d-1}_{x_d} = [\mathtt{a}^{\leq d-1}_{x_d}(\phi^{\leq d-1}_n, \psi^{\leq d-1}_m)]^{n=1,\ldots,N^{\leq d-1}}_{m=1,\ldots,M^{\leq d-1}}.$$

According to (12), the entries of $\mathtt{A}$ are linear combinations of the entries in $\mathtt{A}^{\leq d-1}_{x_d}$ for different $x_d \in \mathbb{X}_d$, but for the same $\phi^{\leq d-1}_n$ and $\psi^{\leq d-1}_n$. This suggests the decomposition of $\mathtt{A}$ into blocks corresponding to the d-th components of ϕ_n and ψ_m, i.e. $\phi^{\overline{\leq} d}_{\pi_d(n)}$ and $\psi^{\overline{\leq} d}_{\sigma_d(m)}$. They identify a block of $\mathtt{A} = \mathtt{A}^{\leq d}$ of size $M^{\leq d-1} \times N^{\leq d-1}$. Let $B^{\overline{\leq} d}_{i,j}$ denote these blocks as in

$$\mathtt{A}^{\leq d} = \begin{pmatrix} B^{\overline{\leq} d}_{1,1} & \cdots & B^{\overline{\leq} d}_{1,N_d} \\ \vdots & & \vdots \\ B^{\overline{\leq} d}_{M_d,1} & \cdots & B^{\overline{\leq} d}_{M_d,N_d} \end{pmatrix}. \qquad (14)$$

Combining (14) and (12) we obtain the following recursive assembling procedure.

Algorithm 1 Recursive sum-factorization

1: **procedure** ASSEMBLE($[\mathbb{X}_\delta]_{\delta=1}^d, [\omega_\delta]_{\delta=1}^d, [\Phi_\delta]_{\delta=1}^d, [\Psi_\delta]_{\delta=1}^d, [\mathcal{F}(\mathbf{x})]_{x\in\mathbb{X}}$)
2: **if** d=0 **then**
3: **return** $[\mathcal{F}(\mathbf{x})]_{x\in\mathbb{X}}$
4: **end if**
5: $\mathtt{A}^{\leq d} \leftarrow 0$ ▷ start with null matrix
6: **for all** $x_d \in \mathbb{X}_d$ **do** ▷ sum all $\mathtt{A}_{x_d}^{\leq d-1}$
7: $\mathtt{A}_{x_d}^{\leq d-1} \leftarrow$ ASSEMBLE($[\mathbb{X}_\delta]_{\delta=1}^{d-1}, [\omega_\delta]_{\delta=1}^{d-1}, [\Phi_\delta]_{\delta=1}^{d-1}, [\Psi_\delta]_{\delta=1}^{d-1}, [\mathcal{F}(\mathbf{x}, x_d)]_{\mathbf{x}\in\mathbb{X}^{\leq d-1}}$)
8: **for all** n_d **with** $\phi_{n_d}^{\bar{d}}(x_d) \neq 0$ **do**
9: **for all** m_d **with** $\psi_{m_d}^{\bar{d}}(x_d) \neq 0$ **do**
10: $B_{m_d,n_d}^{\bar{d}} \leftarrow B_{m_d,n_d}^{\bar{d}} + \omega_d(x_d)\phi_{n_d}^{\bar{d}}(x_d)\psi_{m_d}^{\bar{d}}(x_d)\mathtt{A}_{x_d}^{\leq d-1}$ ▷ cf. (14)
11: **end for**
12: **end for**
13: **end for**
14: **return** $\mathtt{A} = \mathtt{A}^{\leq d}$
15: **end procedure**

For the complexity analysis, we assume that the values of $\mathcal{F}$, ϕ_n and ψ_m are already available; cf. Section 6 on evaluating them and the related computational complexity. The recursive assembling procedure provides a recursive formula for the computational costs for assembling:

$$\mathfrak{C}^{\leq d} \eqsim \#\mathbb{X}_d\big(\mathfrak{C}^{\leq d-1} + p_d^2 \ \mathrm{nnz}(\mathtt{A}_{x_d}^{\leq d-1})\big),$$

where $\mathfrak{C}^{\leq d-1}$ is the cost of assembling $\mathtt{A}_{x_d}^{\leq d-1}$ and $\mathrm{nnz}(\mathtt{A}_{x_d}^{\leq d-1})$ is the number of non-zero entries of $\mathtt{A}_{x_d}^{\leq d-1}$. Here, the factor p_d^2 comes from the number of iterations at line 8 and 9 that are bounded using (7) . Letting $\mathtt{A}_{x_{d-1},x_d}^{\leq d-2}$ be the analogous of $\mathtt{A}_{x_d}^{\leq d-1}$ and so on till $\mathtt{A}_{\mathbf{x}}^{\leq 0} = \mathcal{F}(\mathbf{x})$ and expanding the recursive cost formula yields

$$\mathfrak{C}^{\leq d} \eqsim \sum_{i=1}^{d} \Big(\prod_{\delta=i+1}^{d} \#\mathbb{X}_\delta \Big)(p_i^2 \#\mathbb{X}_i) \ \mathrm{nnz}(\mathtt{A}_{x_i,\dots,x_d}^{\leq i-1}). \tag{15}$$

Note that (15) was obtained using (7) alone. The assumption (8) allow us to bound the number of non-zero entries as follows:

$$\mathrm{nnz}(\mathtt{A}_{x_i,\dots,x_d}^{\leq i-1}) \leq \prod_{\delta=1}^{i-1} k_\delta p_\delta N_\delta,$$

which yields

$$\begin{aligned} \mathfrak{C}^{\leq d} &\lesssim \sum_{i=1}^{d} p_i^2 \Big(\prod_{\delta=i}^{d} \#\mathbb{X}_\delta \Big) \Big(\prod_{\delta=1}^{i-1} k_\delta p_\delta N_\delta \Big) \\ &\leq \sum_{i=1}^{d} p_i^2 \#\mathbb{X}_d \prod_{\delta=1}^{d-1} \max\{\#\mathbb{X}_\delta, k_\delta p_\delta N_\delta\}. \end{aligned} \tag{16}$$

Depending on the maximum, we obtain

$$\forall \delta,\ \#\mathbb{X}_\delta \leq k_\delta p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}^{\lesssim d} \lesssim d\, 2^d\, p^{d+2} N, \tag{17}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \leq k_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}^{\lesssim d} \lesssim d\, 2^d\, p^{d+1} N, \tag{18}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \geq k_\delta p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}^{\lesssim d} \lesssim d\, p^2 \#\mathbb{X}, \tag{19}$$

where we assume in all cases that $k_\delta = 2$.

Remark 4. To achieve the above cost, it is necessary to implement Algorithm 1 properly. From the tensor product structure of the spaces, it follows that the sparsity pattern of $\mathtt{A}_{x_d}^{\lesssim d-1}$ and that of the non-zero blocks $B^{\bar{=}\delta}_{m_\delta,n_\delta}$ are the same. By pre-computing the sparsity pattern, the sums at line 10 of the algorithm can be performed at a cost proportional to the number of non-zeros entries in $\mathtt{A}_{x_d}^{\lesssim d-1}$.

Example 3 (Gauss quadrature)**.** In Isogeometric Analysis, typically spline spaces are used for discretization (cf. Examples 1 and 2). The breakpoints of these spline spaces introduce a decomposition of the parameter domain into

$$K_\delta \leq N_\delta - p_\delta + 1$$

elements on which the splines are polynomial. For assembling, typically Gauss quadrature with p_δ quadrature nodes per element and direction is performed, so using (17) we obtain

$$\#\mathbb{X}_\delta = p_\delta K_\delta \leq p_\delta N_\delta \quad \text{and} \quad \mathfrak{C}^{\lesssim d} \lesssim d\, 2^d\, p^{d+2} N.$$

Example 4 (Weighted quadrature)**.** A way to reduce the computational costs is to reduce the number of quadrature points. In [7] it was shown, that this can be done without loosing accuracy using test-function dependent weights:

$$\omega(\mathbf{x}, m) = \prod_{\delta=1}^{d} \omega_\delta(x_\delta, \sigma_\delta(m)).$$

The adaptation of Algorithm 1 to test-function-dependent quadrature formulas is straightforward. Indeed, it can equivalently be thought as a quadrature with constant weights $\omega(\mathbf{x}) = 1$ and a different test space $\widetilde{\Psi}$ whose functions are pre-multiplied by the quadrature weights. It is interesting to note that the computational cost reported in [7] is explained by (18). Indeed with their construction $\#\mathbb{X}_\delta \lesssim N_\delta$. Note however that this strategy breaks the symmetry between test and trial functions and consequently the symmetry of the assembled matrix.

4 Localized sum factorization

We have described sum factorization as a global assembling procedure that requires the pre-computation of the weights $\mathcal{F}$ on the whole domain. This is

in contrast to the usual FEM assembling procedure which is performed element by element. The idea of this section is to consider a domain that is a union of disjoint boxes and to apply sum factorization on each box.

Such a localization makes sense even if a global tensor-product discretization is chosen because the localization reduces the memory footprint required to store precomputed values of $\mathcal{F}$ and improves locality of the data which might lead to positive caching effects. The case of localizing a global tensor-product discretization is discussed in Section 4.1. It is worth noting that the possibilities opened by localized sum factorization go beyond that case and allow interesting applications, like multipatch domains and adaptive methods, see Section 4.2.

We start our discussion with a given collection $\mathcal{P}$ of pairwise disjoined, axis aligned boxes $\mathcal{D} = \times_{\delta=1}^{d} \mathcal{D}_\delta \subset \mathbb{R}^d$. For each box $\mathcal{D}$, let $\Phi_\mathcal{D} = (\phi_{\mathcal{D},n})_{n=1}^{N_\mathcal{D}}$ and $\Psi_\mathcal{D} = (\psi_{\mathcal{D},m})_{m=1}^{M_\mathcal{D}}$ be generating systems of functions $\mathcal{D} \to \mathbb{R}$ and $\mathbb{X}_\mathcal{D}$ a quadrature on $\mathcal{D}$ fulfilling (5), (6), (7) and (8).

Let $\Phi = (\phi_n)_{n=1}^{N}$ and $\Psi = (\psi_m)_{m=1}^{M}$ be two generating systems of functions defined on the union of the boxes in $\mathcal{P}$, which satisfy the following assumptions.

- *Representability by box-local generators:* For each box $\mathcal{D} \in \mathcal{P}$, we have

$$\begin{cases} \forall\, n = 1, \dots, N, & \phi_n|_\mathcal{D} \in \Phi_\mathcal{D} \cup \{0\}, \\ \forall\, m = 1, \dots, M, & \psi_m|_\mathcal{D} \in \Psi_\mathcal{D} \cup \{0\}. \end{cases} \tag{20}$$

- *Bounded number of box-local generators:* There is some $R > 0$ such that

$$\sum_{\mathcal{D}\in\mathcal{P}} N_\mathcal{D} \le RN \quad \text{where} \quad N_\mathcal{D} = \#\,\Phi_\mathcal{D}, \tag{21}$$

$$\sum_{\mathcal{D}\in\mathcal{P}} \#\{n : \operatorname{supp}\phi_n \cap \mathbb{X}_\mathcal{D} \neq \emptyset\} \le RN. \tag{22}$$

Analogously to Lemma 1, we obtain the following statement.

Lemma 3. *If $\Phi, \Psi \subset H^r([0,1]^d)$ and $\Phi_\mathcal{D}, \Psi_\mathcal{D} \subset H^r(\mathcal{D}^d)$ satisfy any of the conditions* (20)*,* (21) *or* (22) *then also $\partial_\eta\Phi, \partial_\theta\Psi, \partial_\eta\Phi_\mathcal{D}, \partial_\theta\Psi_\mathcal{D}$ satisfy that condition for all $\theta, \eta \in \Theta_r$.*

Remark 5. Note that (22) is equivalent to

$$\sum_{n=1}^{N} \#\{\mathcal{D} \in \mathcal{P} : \operatorname{supp}\phi_n \cap \mathbb{X}_\mathcal{D} \neq \emptyset\} \le RN,$$

which follows from the stronger condition

$$\forall\, \phi_n \in \Phi,\ \#\{\mathcal{D} \in \mathcal{P} : \operatorname{supp}\phi_n \cap \mathcal{D} \neq \emptyset\} \le R. \tag{23}$$

Remark 6. Provided that (20) holds and that for all $\mathcal{D} \in \mathcal{P}$ we have

$$m \neq n \quad \Rightarrow \quad \phi_n|_\mathcal{D} \neq \phi_m|_\mathcal{D} \quad \vee \quad \phi_n|_\mathcal{D} = \phi_m|_\mathcal{D} = 0,$$

then the condition (21) implies the condition (22) with the same value of R.

Let $a : L^2([0,1]^d) \times L^2([0,1]^d) \to \mathbb{R}$ be a bilinear form of the form (1) and $\mathtt{A}$ be the matrix associated to the restriction of a to $\operatorname{span}\Phi \times \operatorname{span}\Psi$. Then, it follows from (20) that $\mathtt{A}$ can be decomposed as in

$$\mathtt{A} = \sum_{\mathcal{D}\in\mathcal{P}} M_{\Psi,\mathcal{D}}^{\top} \mathtt{A}_{\mathcal{D}} N_{\Phi,\mathcal{D}}. \tag{24}$$

where the matrices $N_{\Phi,\mathcal{D}}$ and $M_{\Psi,\mathcal{D}}$ are selection matrices whose columns contain at most a non zero coefficient. The cost decomposes in two parts:

$$\mathfrak{C} = \mathfrak{C}_{ass} + \mathfrak{C}_{acc}, \tag{25}$$

where $\mathfrak{C}_{ass}$ is the cost of assembling the matrices $\mathtt{A}_{\mathcal{D}}$ for all $\mathcal{D} \in \mathcal{P}$ and $\mathfrak{C}_{acc}$ is the cost of accumulating the $\mathtt{A}_{\mathcal{D}}$ into $\mathtt{A}$.

The matrix products in (24) realize only a selection and relabeling of the non zero entries to sum to $\mathtt{A}$, thus the cost $\mathfrak{C}_{acc}$ of accumulating the result in $\mathtt{A}$ is bounded by the number of selected non zero coefficients. As the number of non zero columns in $\mathtt{A}_{\mathcal{D}} N_{\Phi,\mathcal{D}}$ is $\leq \#\{n : \operatorname{supp}\phi_n \cap \mathbb{X}_{\mathcal{D}} \neq \emptyset\}$ and each column contains at most $\prod_{\delta=1}^{d} k_\delta p_\delta$ non zeros because of (8), we obtain

$$\mathfrak{C}_{acc} \lesssim \sum_{\mathcal{D}\in\mathcal{P}} \prod_{\delta=1}^{d} k_\delta p_\delta \#\{n : \operatorname{supp}\phi_n \cap \mathbb{X}_{\mathcal{D}} \neq \emptyset\}.$$

Using (22) and assuming $k_\delta = 2$, we further obtain

$$\mathfrak{C}_{acc} \lesssim 2^d p^d R N.$$

If $\#\mathbb{X}_{\mathcal{D},\delta} \leq k_\delta p_\delta N_{\mathcal{D},\delta}$ for all $\mathcal{D} \in \mathcal{P}$ and $\delta = 1, \ldots, d$, then (17) and (21) yield

$$\mathfrak{C}_{ass} \lesssim \sum_{\mathcal{D}\in\mathcal{P}} d 2^d p^{d+2} N_{\mathcal{D}} \leq d 2^d R p^{d+2} N. \tag{26}$$

Similarly, under the stronger hypothesis $\#\mathbb{X}_{\mathcal{D},\delta} \leq k_\delta N_\delta$, (18) and (21), we obtain

$$\mathfrak{C}_{ass} \lesssim \sum_{\mathcal{D}\in\mathcal{P}} \leq d 2^d R p^{d+1} N. \tag{27}$$

Concluding, $\mathfrak{C}$ is dominated by $\mathfrak{C}_{ass}$ and thus we have the following analogues to (17), (18) and (19):

$$\forall \mathcal{D},\ \delta,\ \#\mathbb{X}_{\mathcal{D},\delta} \leq N_{\mathcal{D},\delta} k_\delta p_\delta \qquad \Rightarrow \qquad \mathfrak{C} \lesssim d\, 2^d\, R\, p^{d+2} N, \tag{28}$$

$$\forall \mathcal{D},\ \delta,\ \#\mathbb{X}_{\mathcal{D},\delta} \leq N_{\mathcal{D},\delta} k_\delta \qquad \Rightarrow \qquad \mathfrak{C} \lesssim d\, 2^d\, R\, p^{d+1} N, \tag{29}$$

$$\forall \mathcal{D},\ \delta,\ \#\mathbb{X}_{\mathcal{D},\delta} \geq N_{\mathcal{D},\delta} k_\delta p_\delta \qquad \Rightarrow \qquad \mathfrak{C} \lesssim d\, R\, p^2 \#\mathbb{X}. \tag{30}$$

4.1 Localized sum factorization for tensor-product B-spline bases

In this subsection, we again consider the discretization from Example 3: Φ is a B-spline basis defined on $[0,1]^d$.

Again, the breakpoints of Φ define a partitioning of the domain $[0,1]^d$ into elements on which the basis functions are polynomials. On each element, tensor product Gauss quadrature of order p_δ is employed.

Now, we localize sum-factorization to a partition $\mathcal{P}$ in which each box $\mathcal{D}$ is aligned with the element boundaries and we study the dependence of the assembling cost on the size of the boxes measured by the number of contained elements. The main advantages of partitioning the assembling process are reducing the memory requirement and allow for parallelization. By assembling a box at at time only $\mathcal{F}(\mathbb{X}_\mathcal{D})$ needs to be pre-computed and stored instead of $\mathcal{F}(\mathbb{X})$. Assembling on different boxes can be done in parallel provided that data-races are avoided when the result is gathered to the final matrix.

First, we observe that the restriction to axes-aligned boxes does not destroy the tensor-product structure.

Lemma 4. *Assume that Φ and Ψ satisfy* (5) *and $\mathcal{P}$ is a partition of $[0,1[^d$ into boxes of the type $[a_1,b_1[\times\cdots\times[a_d,b_d[$, then*

$$\begin{cases} \Phi_\mathcal{D} := \{\phi_n|_\mathcal{D} : n = 1,\dots,N\}\backslash\{0\}, \\ \Psi_\mathcal{D} := \{\psi_m|_\mathcal{D} : m = 1,\dots,M\}\backslash\{0\} \end{cases}$$

satisfies (5) *for all $\mathcal{D}$. For this choice,* (22) *implies* (21).

Proof. We have $\phi_n(x) = \prod_{\delta=1}^d \phi^{=\delta}_{\pi_\delta(n)}(x_\delta)$, and $\mathcal{D} = \mathcal{D}_1\times\cdots\times\mathcal{D}_d$. So, we obtain $(\phi_n|_\mathcal{D})(x) = \prod_{\delta=1}^d \left(\phi^{=\delta}|_{\mathcal{D}_\delta}\right)(x_\delta)$ and, therefore,

$$\phi_n|_\mathcal{D} \in \Phi_1|_{\mathcal{D}_1}\otimes\cdots\otimes\Phi_d|_{\mathcal{D}_d},$$

and if $\phi_n|_\mathcal{D}\neq 0$

$$\phi_n|_\mathcal{D} \in (\Phi_1|_{\mathcal{D}_1}\backslash\{0\})\otimes\cdots\otimes(\Phi_d|_{\mathcal{D}_d}\backslash\{0\}),$$

which shows (20). The same is possible for Ψ.

If $\operatorname{supp}\phi_n\cap\mathbb{X}_\mathcal{D}\neq\emptyset$ then $0\neq\phi_n|_\mathcal{D}$ and $\phi_n|_\mathcal{D}\in\Phi_\mathcal{D}$. Then, we have $\Phi_\mathcal{D} = \{\phi_n|_\mathcal{D} \,:\, \phi_n|_\mathcal{D}\neq 0\}$, so (22) implies (21). □

As next step, we show (22) for B-spline bases.

Lemma 5. *If Φ is a tensor-product B-spline basis of order $p_1, \dots, p_d$, then taking $\mathcal{P}$ such that each $\mathcal{D} \in \mathcal{P}$ contains at least $s_1 \times \dots \times s_d \geq 1$ elements, yields* (21) *and* (22) *with*

$$R = \prod_{\delta=1}^{d} \left\lceil \frac{s_\delta + p_\delta - 1}{s_\delta} \right\rceil.$$

Proof. The support of $\phi_n^{\bar{z}\delta}$ contains at most p_δ elements in direction δ. The the number of boxes containing s_δ elements intersecting $\operatorname{supp} \phi_n^{\bar{z}\delta}$ is maximized if the end of the leftmost box coincide with the end of the leftmost element. In this case the total number of boxes intersecting $\operatorname{supp} \phi_n^{\bar{z}\delta}$ is

$$1 + \left\lceil \frac{p_\delta - 1}{s_\delta} \right\rceil = \left\lceil \frac{s_\delta + p_\delta - 1}{s_\delta} \right\rceil.$$

Using Remark 5, we obtain (22) and using Lemma 4 further (21). □

Finally, we give an estimate for the number of quadrature points per box.

Lemma 6. *If Φ is a tensor-product B-spline basis of order $p_1, \dots, p_d$, and we perform Gauss-quadrature with p_δ quadrature points per element and direction, then we have for each box $\mathcal{D}$ of size $s_1 \times \dots \times s_d \geq 1$ elements that*

$$\#\mathbb{X}_{\mathcal{D},\delta} \leq \underbrace{\frac{s_\delta\, p_\delta}{s_\delta + p_\delta - 1}}_{\in [1, p_\delta[} N_{\mathcal{D},\delta}.$$

Proof. We have on the one hand for Gauss quadrature $\#\mathbb{X}_{\mathcal{D},\delta} = s_\delta p_\delta$ and on the other hand $s_\delta + p_\delta - 1 \leq N_{\mathcal{D},\delta}$, where equality is obtained for B-splines of maximum smoothness. □

Lemmas 5 and 6 show two competing effects on the cost of localized sum factorization. On one hand, the upper bound for R from Lemma 5 deteriorates for small boxes. On the other hand, Lemma 6 states that the ratio between quadrature points and basis function decreases for small boxes allowing one to apply the bound in (29) instead of that in (28). This observation allows to conclude as follows.

Example 5 (Per-element sum-factorization)**.** Let Φ be a B-spline space of order $p_1, \dots, p_d$. In this example, $\mathcal{P}$ is the collection of the elements associated to Φ. The local spaces $\Phi_\mathcal{D}, \Psi_\mathcal{D}$ and the local quadrature $\mathbb{X}_\mathcal{D}$ are defined by

$$\begin{cases} \Phi_{\mathcal{D},\delta} = \{\phi_n^{\bar{z}\delta}|_{\mathcal{D}_\delta} : \phi_n^{\bar{z}\delta} \in \Phi_\delta\} \backslash \{0\}, \\ \Psi_{\mathcal{D},\delta} = \{\psi_m^{\bar{z}\delta}|_{\mathcal{D}_\delta} : \psi_m^{\bar{z}\delta} \in \Psi_\delta\} \backslash \{0\}, \\ \mathbb{X}_\mathcal{D} \text{ is the standard Gauss quadrature on the box } \mathcal{D}. \end{cases} \tag{31}$$

Lemma 5 yields (22) with $R = p^d$ and, by Lemma 4, we have (21). In this case $\#\mathbb{X}_{\mathcal{D},\delta} = N_{\mathcal{D},\delta} = p_\delta$ so that (29) holds and, as proved in [2],

$$\mathfrak{C} \lesssim d\, 2^d\, p^{2d+1} N.$$

Note that this example employs the sum factorization approach which was proposed in [2].

Example 6 (Per macro-element sum-factorization)**.** Let Φ be again a B-spline space of order $p_1, \dots, p_d$. In this example, we now consider a partition $\mathcal{P}$ of $[0,1]^d$ such that all boxes have size of at least $p_1 \times \cdots \times p_\delta$ elements. Again, we assume (31). For this case, we obtain from Lemmas 5 and 6 and (28)

$$\mathfrak{C} \lesssim d\, 4^d\, p^{d+2}\, N, \tag{32}$$

i.e., asymptotically the same costs as for the global approach.

This description can be refined leading to a compromise that allows for more freedom in the choice of the partitioning $\mathcal{P}$. The following Lemma is a refinement of Lemma 5 that allows for macro-elements that have size 1 in one direction, possibly different for each macro-element. It will be used in Example 7.

Lemma 7. *For all $2 \le p_1, \dots, p_d \in \mathbb{N}$ and covering $\mathcal{P}$ of $\times_{\delta=1}^d [0, p_\delta[$ such that for all $\mathcal{D} \in \mathcal{P}$*

$$\mathcal{D} \cap \left(\bigtimes_{\delta=1}^{d} [0, p_\delta[\right) \neq \emptyset, \tag{33}$$

$$\mathcal{D} = [a_1, b_1[\times \cdots \times [a_d, b_d[, \qquad a_i, b_i \in \mathbb{Z}, \tag{34}$$

$$b_\delta - a_\delta = \begin{cases} p_\delta & \delta \neq j_{\mathcal{D}} \\ 1 & \delta = j_{\mathcal{D}} \end{cases} \tag{35}$$

for some $j_{\mathcal{D}} \in \{1, \dots, d\}$ that depends on $\mathcal{D}$, the following sharp inequality holds

$$\#\mathcal{P} \le 2^{d-1} \sum_{\delta=1}^{d} (p_\delta - 2) + 2^d.$$

Proof. We prove the bound by induction on the dimension d. For $d = 1$ any partition of $[0, p]$ in segments of length 1 has exactly p elements. Let $\mathcal{P}$ be a partition realizing the maximum cardinality in dimension d and define $\mathcal{P}_B$ to be the subset of $\mathcal{P}$ containing the boxes that intersect some box B. Then for $L = [0,1[\times \times_{\delta=2}^d [0, p_\delta[$, $R = [p_1 - 1, p_1[\times \times_{\delta=2}^d [0, p_\delta[$ we have

$$\#\mathcal{P} = \#\mathcal{P}_L + \#\mathcal{P}_R + \#(\mathcal{P} \setminus \mathcal{P}_{L \cup R}).$$

A box $\mathcal{D} \in \mathcal{P} \setminus \mathcal{P}_{L \cup R}$ can not have length p_1 in the first direction. Thus, $\mathcal{D}$ has size $1 \times p_2 \times \cdots \times p_d$ and is contained in $[i, i+1[\times \mathbb{R}^{d-1}$ for some $i \in \{1, \dots, p_1 - 2\}$.

Applying Lemma 5 yields that for each $i = 1, \dots, p_1 - 2$, there are at most 2^{d-1} such boxes so that we have

$$\#(\mathcal{P}_S \setminus \mathcal{P}_{L\cup R}) \leq 2^{d-1}(p_1 - 2).$$

The projection $(x_1, \dots, x_d) \to (x_2, \dots, x_d)$ maps the boxes in $\mathcal{P}_L$ to disjoint boxes in $\mathbb{R}^{d-1}$ that cover $\times_{\delta=2}^{d}[0, p_\delta[$. By induction hypothesis they are at most $2^{d-2}\sum_{\delta=2}^{d}(p_\delta - 2) + 2^{d-1}$. The same reasoning applies to $\mathcal{P}_R$ so that we obtain

$$\begin{aligned}\#\mathcal{P}_S &\leq 2[2^{d-2}\sum_{\delta=2}^{d}(p_\delta - 2) + 2^{d-1}] + 2^{d-1}(p_1 - 2)\\ &= 2^{d-1}\sum_{\delta=1}^{d}(p_\delta - 2) + 2^d.\end{aligned}$$

For $d = 1$ the result is sharp as we can cover $[0, p[$ with p segments of length 1. Given a covering $\widetilde{\mathcal{P}}$ of $\times_{\delta=2}^{d}[0, p_\delta[$ with cardinality $2^{d-2}\sum_{\delta=2}^{d}(p_\delta - 2) + 2^{d-1}$ we construct a maximal cardinality covering of $\times_{\delta=1}^{d}[0, p_\delta[$ by setting

$$\begin{aligned}\mathcal{P}_L &= \{[1 - p_1, 1[\times\tilde{\mathcal{D}} : \widetilde{\mathcal{D}} \in \widetilde{\mathcal{P}}\}\ ,\\ \mathcal{P}_R &= \{[p_1 - 1, 2p_1 - 1[\times\widetilde{\mathcal{D}} : \widetilde{\mathcal{D}} \in \widetilde{\mathcal{P}}\}\ ,\end{aligned}$$

and completing $\mathcal{P}_L \cup \mathcal{P}_R$ to a covering $\mathcal{P}$ of $\times_{\delta=1}^{d}[0, p_\delta[$ using the boxes

$$\mathcal{D} = [i, i+1[\times \bigtimes_{\delta=2}^{d}[t_i p_\delta + 1, (t_i + 1)p_\delta + 1[$$

for $i = 1, \dots, p_1 - 2$ and $t_i \in \{-1, 0\}$. □

Example 7 (Narrow macro-elements)**.** Using the last lemma, we obtain that macro-elements $\mathcal{D} \in \mathcal{P}$ of size $s_1 \times \cdots \times s_d$ with $s_\delta \geq p_\delta$ for $\delta \neq j_{\mathcal{D}}$ and $s_{j_{\mathcal{D}}} \geq 1$ achieve the same asymptotic costs in p as we have obtained in Example 6.

From Lemma 7 and Remark 5, we have (21) and (22) with $R = d2^{d-1}p$. On each macro-element $\mathcal{D}$ the cost $\mathfrak{C}_{\mathcal{D}} \lesssim d2^d p^{d+1} N_{\mathcal{D}}$ can be achieved by permuting the coordinates so that $j_{\mathcal{D}}$, the direction of size one element, comes last. Indeed, if $j_{\mathcal{D}} = d$, we have $\#\mathbb{X}_{\mathcal{D},d} = \#N_{\mathcal{D},d} = p_\delta$ and the cost in (18) is achieved. Since the estimate holds for all boxes, we have (29) and

$$\mathfrak{C} \lesssim d^2 4^d p^{d+2} N.$$

Narrow macro-elements are particularly of interest for the construction of hierarchical LR splines, [3], where local refinement is done in alternating directions. Note that additional costs are caused by the reordering of $\mathcal{F}$ and by the permutation of the coefficients in $\mathtt{A}_{\mathcal{D}}$.

4.2 Localized sum factorization for some non-tensor-product bases

Example 8 (Multipatch domains)**.** Another example of localized sum factorization is Isogeometric Analysis on multipatch geometries. The domain Ω of the PDE is the union of the subdomains Ω_ℓ, $\ell = 1, \dots, L$ each parameterized by a map $\mathbf{G}_\ell$ defined on $\widehat{\Omega}_\ell = [0,1]^d$:

$$\Omega = \bigcup_{\ell=1}^{L} \mathbf{G}_\ell(\widehat{\Omega}_\ell), \quad \text{where} \quad \widehat{\Omega}_\ell = [0,1]^d.$$

This means that the assembling is done on L copies of the parametric domain $[0,1]^d$. This can be expressed on the framework form the beginning of this section by writing

$$\widehat{\Omega} := \{1, \dots, L\} \times [0,1]^d.$$

The partition $\mathcal{P}$ contains the parametric patches

$$\mathcal{D}_\ell := \{\ell\} \times [0,1]^d,$$

where we have a tensor product spline space $\Phi_{\mathcal{D}_\ell}$ on each of those patches. The global space Φ is obtained by identifying matching functions that are non-zero on the glued boundaries of the patches.

The condition (21) and (22) are fulfilled according to Remark 5 with R that is the maximum number of patches on which a function $\phi \in \Phi$ is active. In cases of practical interest the patches are big compared to the supports of the basis functions and their angles at the vertices (and edges in 3D) are close to $\pi/2$. Consequently the value of R is small $\approx 2^d$. However, in contrived examples, it can be made arbitrarily big by constructing a multipatch in which many patches contain a common vertex.

Example 9 (HB-splines)**.** The Hierarchical B-spline (HB) is a basis that breaks the global tensor product structure and allows for adaptive methods in which only a part of the domain is refined, see [6]. The basis is obtained by *selecting* functions from different tensor product B-spline bases on different regions of the domain.

Let $\mathfrak{B}_1, \dots, \mathfrak{B}_L$ be the tensor product B-spline bases of the same degree defined on the domain Ω that generate nested spaces

$$i < j \Rightarrow \operatorname{span} \mathfrak{B}_i \subset \operatorname{span} \mathfrak{B}_j. \tag{36}$$

Let $\Omega \supseteq \Omega_1 \supseteq \cdots \supseteq \Omega_L = \emptyset$ be corresponding closed domains. For simplicity we further assume that the $\mathfrak{B}_i$ are a sequence of dyadically refined spaces and that each of the Ω_i is a union of elements associated to $\mathfrak{B}_i$. The *hierarchical basis* (HB-splines) is defined by Kraft's *selection* criteria [11]:

$$\mathfrak{H} = \bigcup_{i=1}^{L} \{\gamma \in \mathfrak{B}_i : \operatorname{supp} \gamma \subseteq \Omega_i \text{ and } \operatorname{supp} \gamma \cap (\Omega_i \setminus \Omega_{i+1}) \neq \emptyset\} \ . \tag{37}$$

A HB basis is said to be β-*admissible* if for all $\mathbf{x} \in \Omega$,

$$\max\{i : \exists \gamma \in \mathfrak{B}_i \cap \mathfrak{H} : \mathbf{x} \in \operatorname{supp} \gamma\} - \min\{i : \exists \gamma \in \mathfrak{B}_i \cap \mathfrak{H} : \mathbf{x} \in \operatorname{supp} \gamma\} \leq \beta + 1. \tag{38}$$

The localized sum factorization applies to β-admissible (usually 2-admissible) HB bases Φ and Ψ if the collection $\mathcal{P}$ is chosen such that each box $\mathcal{D}$ is contained in one of the *rings*

$$\Delta_i := \overline{\Omega_i \setminus \Omega_{i+1}}.$$

In this case, the restriction of the functions in $\mathfrak{H}$ to any of the boxes $\mathcal{D}$ is contained in the union of β tensor product spaces. If each ring Δ_i allows a partition into macro-elements of size p_δ elements per direction, then (21) and (22) hold with $R = 3^d$. For Gauss quadrature, we obtain using (28) the following bound for the costs $\mathfrak{C}$, bounded as follows:

$$\mathfrak{C} \lesssim d 6^d p^{d+2} N.$$

Note that narrow macro-elements are also possible as a consequence of Lemma 7.

Remark 7. Note that localized sum factorization cannot be applied directly to Truncated Hierarchical B-spline bases, cf. [8], even if it spans the same space. The problem is that truncation breaks the tensor structure, i.e. that the $\Phi_{\mathcal{D}}$ corresponding to the truncated basis do not fulfill (5). Nevertheless, it is possible to express the matrix $\mathtt{A}_{\text{trunc}}$ corresponding to the truncated basis as

$$\mathtt{A}_{\text{trunc}} = M_\Psi^\top \mathtt{A} N_\Phi$$

where $\mathtt{A}$ is the system matrix for the hierarchical basis and M_Ψ, N_Φ are the change of basis. Since the number of non zero entries in each column of M_Ψ, N_Φ is bounded by $2^{\beta d+1} p^d$ for β-admissible hierarchical splines we obtain that

$$\mathfrak{C}_{trunc} = \mathfrak{C} + \mathfrak{C}_{post}, \quad \text{and} \quad \mathfrak{C}_{post} \lesssim 2^{\beta d+1} p^{2d} N_\Phi.$$

Interestingly it has the same asymptotic complexity as the cost for hierarchical splines for two dimensional domains ($2d = d + 2$).

Remark 8. The extension to trimmed domains would be a challenge. So far, there is no clear solution for such discretizations. A straight-forward approach would be to compute the discretization ignoring the trimming first and recompute the matrix elements corresponding to the basis functions whose support intersects with the trimming curve. In reasonable cases, one could assume that the number of those matrix elements is small compared to the total number.

Remark 9. The presented examples are not exhaustive. Surely, different generating systems fulfill the assumptions, for instance the hierarchical LR splines [3] do. Moreover, multipatch domains (Example 8) and adaptivity (Example 9) can be combined effortless.

5 Two algebraic descriptions

We called the described recursive algorithm *sum factorization*, but it differ from the original description in [2]. The unifying idea is that the map from the value of the coefficients on the quadrature points to the system matrix is linear and can be represented as a tensor. Moreover it has a Kronecker-like structure that is inherited from the tensor structure of the discrete spaces and of the quadrature.

Let $\mathbb{X}$ be lexicographically ordered and its points be $\mathbf{x}_{i_1,\dots,i_d}$ with $i_\delta = 1,\dots,\#\mathbb{X}_\delta$ and F be the d tensor defined as

$$F_{i_1,\dots,i_d} = \mathcal{F}(\mathbf{x}_{i_1,\dots,x_{i_d}}).$$

Then the $2d$ tensor T that corresponds to $\mathtt{A}$ through π and σ

$$T_{\sigma_1(m),\dots,\sigma_d(m),\pi_1(n),\dots,\pi_d(n)} = \mathsf{a}(\phi_n,\psi_m) = \mathtt{A}_{m,n}$$

can be written as

$$T_{m_1,\dots,m_d,n_1,\dots,n_d} = F_{i_1,\dots,i_d} Q_{m_1,\dots,m_d,n_1,\dots,n_d,i_1,\dots,i_d}$$

where, using Einstein notation, repeated indices denote sums and Q is the $3d$ tensor corresponding to the linear maps from F to T_h, i.e.,

$$Q_{\sigma_1(m),\dots,\sigma_d(m),\pi_1(n),\dots,\pi_d(n),i_1,\dots,i_d} = \phi_n(\mathbf{x}_{i_1,\dots,i_d})\psi_m(\mathbf{x}_{i_1,\dots,i_d})\omega(\mathbf{x}_{i_1,\dots,i_d}).$$

The Cartesian structure of the quadrature points, together with the tensor structure of the generators allow for a factorisation of the above as

$$T_{m_1,\dots,m_d,n_1,\dots,n_d} = F_{i_1,\dots,i_d} Q^{\bar{}1}_{m_1,n_1,i_1} \dots Q^{\bar{}d}_{m_d,n_d,i_d}.$$

The original description of sum factorisation corresponds to the partial evaluations

$$T^{\bar{<}\delta}_{m_1,\dots,m_\delta,n_1,\dots,n_\delta,i_{\delta+1},\dots,i_d} = F_{i_1,\dots,i_d} Q^{\bar{}1}_{m_1,n_1,i_1} \dots Q^{\bar{}\delta}_{m_\delta,n_\delta,i_\delta}$$

where the temporaries for all $i_{\delta+1},\dots,i_d$ are computed at each step. This achieves a cost reduction because $\operatorname{nnz}(Q^{\bar{}\delta}) \approx k_\delta p_\delta N_\delta$ while $\operatorname{nnz}(Q) \approx \prod_{\delta=1}^{d} k_\delta p_\delta N_\delta$ and this impact the cost of computing the tensor contraction. The recursive description corresponds to the computation of a slice of $T^{\bar{<}\delta}_{m_1,\dots,m_\delta,n_1,\dots,n_\delta,i_{\delta+1},\dots,i_d}$ corresponding to fixed indices $i_{\delta+1},\dots,i_d$ at a time.

They do exactly the same operation, although in a different order, but the recursive implementation requires less temporary memory. The complexity gain with respect to a standard per-element quadrature is given by the fact that in standard per-element assembling, the coefficients of the composition of the $Q^{\bar{}\delta}$ are recomputed for each element.

The same factorisation can be explained, without recurring tensors, using a Kronecker construction. This second description correspond to the first by the flattening of the indices $i_1, \ldots, i_d$. Let $\mathbb{X}$ be lexicographically ordered and define

$$Q_\Phi = [\phi_n(\mathbf{x})]_{n=1,\ldots,N}^{\mathbf{x}\in\mathbb{X}}, \qquad Q_\Psi = [\omega(\mathbf{x})\psi_m(\mathbf{x})]_{m=1,\ldots,M}^{\mathbf{x}\in\mathbb{X}}$$

and F as the diagonal matrix containing $\mathcal{F}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{X}$. Then

$$\mathtt{A} = (Q_\Psi)^\top F Q_\Phi. \tag{39}$$

The factorization can be now expressed by decomposing Q_Φ and Q_Ψ as Kronecker products:

$$\begin{aligned} Q_\Phi &= \bigKr_{\delta=1}^{d} Q_\Phi^{\bar{\delta}}, \qquad Q_\Phi^{\bar{\delta}} = [\phi_n^{\bar{\delta}}(x_\delta)]_{n=1,\ldots,N_\delta}^{x_\delta\in\mathbb{X}_\delta} \\ Q_\Psi &= \bigKr_{\delta=1}^{d} Q_\Psi^{\bar{\delta}}, \qquad Q_\Psi^{\bar{\delta}} = [\psi_m^{\bar{\delta}}(x_\delta)\omega_\delta(x_\delta)]_{m=1,\ldots,M_\delta}^{x_\delta\in\mathbb{X}_\delta}. \end{aligned}$$

Note that we have the following correspondence with the previous representation: $\forall m_\delta, n_\delta, i_\delta$

$$Q^{\bar{\delta}+1}_{m_\delta,n_\delta,i_\delta} = Q^{\bar{\delta}}_{\Psi\, m_\delta,i_\delta} Q^{\bar{\delta}}_{\Phi\, m_\delta,i_\delta}.$$

Since for all matrices C, D

$$C \Kr D = (C \Kr I)(I \Kr D)$$

we have

$$\mathtt{A} = (I \Kr \ldots \Kr Q_\Psi^{\bar{d}})^\top \ldots (Q_\Psi^{\bar{1}} \Kr \ldots \Kr I)^\top F (Q_\Phi^{\bar{1}} \Kr \ldots \Kr I) \ldots (I \Kr \ldots \Kr Q_\Phi^{\bar{d}}). \tag{40}$$

Note that

$$(Q_\Psi^{\bar{1}} \Kr \ldots \Kr I)^\top F (Q_\Phi^{\bar{1}} \Kr \ldots \Kr I) = \begin{pmatrix} \ddots & & \\ & \mathtt{A}^{\bar{1}}_{x_2,\ldots,x_d} & \\ & & \ddots \end{pmatrix}_{x_i\in\mathbb{X}_i}$$

where the right hand side is the block diagonal matrix with blocks $\mathtt{A}^{\bar{1}}_{x_2,\ldots,x_d}$ for $x_i \in \mathbb{X}_i$, $i = 2, \ldots, d$. The blocks are lexicographically ordered along the diagonal. More generally after δ products we obtain

$$(I \Kr \ldots Q_\Psi^{\bar{\delta}} \ldots \Kr I)^\top \ldots F \ldots (I \Kr \ldots Q_\Phi^{\bar{\delta}} \ldots \Kr I) = \begin{pmatrix} \ddots & & \\ & \mathtt{A}^{\bar{\delta}}_{x_{\delta+1},\ldots,x_d} & \\ & & \ddots \end{pmatrix}_{x_i\in\mathbb{X}_i}.$$

The improved performance can then easily explained by the cost of matrix-matrix multiplication:

$$\operatorname{nnz}(I \circledast \dots Q_{\Phi}^{\bar{\delta}} \dots \circledast I) \leq \#\mathbb{X} p_{\delta}$$

while

$$\operatorname{nnz}(Q_{\Phi}) \leq \#\mathbb{X} \prod_{\delta=1}^{d} p_{\delta}.$$

By computing the product step by step the cost $\eqsim \sum_{\delta} p_{\delta}$, while if the product is computed in one step the cost is $\eqsim \prod_{\delta} p_{\delta}$.

6 Pre-assembling costs

The overall costs presented in the last sections only covers the costs of assembling the matrix, assuming that the values of the functions in the generating sets Φ_{δ} and Ψ_{δ} and the values of the coefficient function $\mathcal{F}$ are already known or can be computed in constant time. This follows the presentation in the literature, however it is worth to have a closer look onto the corresponding computational costs. Our aim is to show algorithms for which the computation of Φ_{δ}, Ψ_{δ} and $\mathcal{F}$ can be performed at a cost that is inferior to the assembling cost. For simplicity, we focus on the global sum-factorization approach, but the results hold also for the localized version as they apply to each box separately.

6.1 Evaluation of the functions in the generating sets Φ and Ψ

The evaluation of a B-spline function ϕ_n or ψ_m cannot be done in constant time. As we have a tensor-product structure, an efficient approach is to pre-compute the function values for the corresponding univariate functions $\phi_n^{\bar{\delta}}$ or $\psi_m^{\bar{\delta}}$.

For a standard B-spline basis, this yields costs $\mathfrak{C}$, which are bounded as follows

$$\mathfrak{C} \lesssim \sum_{\delta=1}^{d} p_{\delta}^2 \#\mathbb{X}_{\delta} \leq 2p^3 \sum_{\delta=1}^{d} N_{\delta}, \tag{41}$$

where the second estimate holds for $\#\mathbb{X}_{\delta} \leq k_{\delta} p_{\delta} N_{\delta}$ with $k_{\delta} = 2$ as in (17) and (18). The resulting costs are always much smaller than $p^{d+2}N$, the costs of assembling.

6.2 Evaluation of the coefficient function $\mathcal{F}$

As mentioned in Section 2, we have to evaluate the function $\mathcal{F}$ for every quadrature point. The computational costs related to this task are often not addressed

in standard papers on assembling in Isogeometric Analysis. If the computational costs for evaluating $\mathcal{F}$ at one quadrature point, grows faster than p^2, then a direct approach for evaluating $\mathcal{F}$ would lead to computational costs which exceed the costs of the assembling strategy.

In classical Isogeometric Analysis, we assume that the computational domain Ω is parameterized by a diffeomorphism

$$\mathbf{G} : \widehat{\Omega} := [0,1]^d \to \Omega = \mathbf{G}(\widehat{\Omega}) \subset \mathbb{R}^s,$$

which is an element of span Φ and has the form

$$\mathbf{G} = \sum_{n=1}^{N} \mathbf{c}_n \phi_n.$$

In this case, also the evaluation of $\mathcal{F}$ involves the evaluation of $\mathbf{G}$ and/or its derivatives, cf. (10). For ease of the presentation, in the following we only present the evaluation of $\mathbf{G}$, the evaluation of its derivatives is completely analogous.

For the evaluation of $\mathbf{G}$ at the quadrature points $\mathbb{X}$, we can again exploit the tensor product structure of Φ. This yields the following algorithm:

Algorithm 2 Recursive function computation

```
1:  procedure EVAL([X_δ]_{δ=1}^d, [Φ_δ]_{δ=1}^d, [c_n]_{n=1}^N)
2:      if d=0 then
3:          return c_1                                   ▷ Here, N = 1
4:      end if
5:      G^{≤d} ← 0
6:      for all n_d ∈ {1, ..., N_d} do
7:          G^{≤d-1}_{n_d} ← EVAL([X_δ]_{δ=1}^{d-1}, [Φ_δ]_{δ=1}^{d-1}, [c_n]_{n:π_d(n)=n_d})
8:          for all x_d ∈ X_d : φ^{-d}_{n_d}(x_d) ≠ 0 do
9:              G^{≤d}(x, x_d) ← G^{≤d}(x, x_d) + φ^{-d}_{n_d}(x_d) G^{≤d-1}_{n_d}(x)
10:         end for
11:     end for
12:     return G^{≤d}
13: end procedure
```

Again, we derive the number of floating point operations, assuming that the functions in the generating set Φ_δ have already been evaluated. By counting the number of invocations of lines 7 and 9 and using (7), we obtain that the costs $\mathfrak{C}^{\leq d}$ satisfy

$$\mathfrak{C}^{\leq d} \lesssim N_d \mathfrak{C}^{\leq d-1} + s p_d \ \#\mathbb{X}^{\leq d},$$

where s comes from the fact that the $\mathbf{c}_i$ are vectors in $\mathbb{R}^s$. Recursively plugging in yields

$$\mathfrak{C}^{\leq d} \lesssim s \sum_{\delta=1}^{d} \left(\prod_{i=\delta+1}^{d} N_i \right) p_\delta \ \#\mathbb{X}^{\leq \delta}.$$

Depending on the ratio between $\#\mathbb{X}_\delta$ and N_δ and assuming $k_\delta = 1$, as with Gauss quadrature, we obtain the following costs

$$\forall \delta,\ \#\mathbb{X}_\delta \leq k_\delta p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}^{\lesssim d} \lesssim sd\ p^{d+1} N, \tag{42}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \leq k_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}^{\lesssim d} \lesssim sd\, p\, N, \tag{43}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \geq k_\delta p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}^{\lesssim d} \lesssim sd\, p\, \#\mathbb{X}. \tag{44}$$

Here, d is the dimension of parametric domain and s is the dimension of the target space. So, for a surface embedded in $\mathbb{R}^3$, we have $d = 2$ and $s = 3$.

The computation of the map is not sufficient to IGA applications: as shown in Example 1, it is often the case that the Jacobian of $\mathbf{G}$, its pseudoinverse and the determinant of the Jacobian are required. The i-th column of $J_{\mathbf{G}}$ is computed using the presented algorithm by replacing Φ_i with $\partial\Phi_i$. Remembering that $J_{\mathbf{G}}$ has d columns, we have the following costs:

$$\forall \delta,\ \#\mathbb{X}_\delta \leq p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{jac} \lesssim sd^2\ p^{d+1} N, \tag{45}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \leq N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{jac} \lesssim sd^2\, p\, N, \tag{46}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \geq p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{jac} \lesssim sd^2\, p\, \#\mathbb{X}. \tag{47}$$

For second order PDEs, the coefficients $\mathcal{F}$ are computed as in (10). If $s = d$, i.e. $\mathbf{G} : \mathbb{R}^d \to \mathbb{R}^d$ then $J_{\mathbf{G}}^{-1}$ can be computed in $\approx d^3$ operations per quadrature point using LU factorization. If $\mathbf{G} : \mathbb{R}^d \to \mathbb{R}^s$, $s > d$ then the formula in (10) is still applicable by denoting with $J_{\mathbf{G}}^{-1}$ the left pseudoinverse of $J_{\mathbf{G}}$ and with $|J_{\mathbf{G}}|$ the area density, i.e.

$$J_{\mathbf{G}}^{-1} := (J_{\mathbf{G}}^\top J_{\mathbf{G}})^{-1} J_{\mathbf{G}}^\top, \tag{48}$$

$$|J_{\mathbf{G}}| := \det(J_{\mathbf{G}}^\top J_{\mathbf{G}})^{\frac{1}{2}}. \tag{49}$$

Both can be computed with $\approx sd^2$ operations per quadrature point using *LU* factorization to invert $J_{\mathbf{G}}^\top J_{\mathbf{G}}$ giving

$$\mathfrak{C}_{inv} \lesssim sd^2\, \#\mathbb{X}.$$

The costs $\mathfrak{C}_{mult}$ of evaluating the product in (10) is of the same order as the costs for evaluating the block $J_{\mathbf{G}}^{-T} A J_{\mathbf{G}}^{-1}$. This has costs of order $\approx s^2 d$ per quadrature point. Thus the costs of computing $\mathcal{F}$ according to (10) decomposes as

$$\mathfrak{C}_{\mathcal{F}} = \mathfrak{C}_{jac} + \mathfrak{C}_{inv} + \mathfrak{C}_{mult} \lesssim \mathfrak{C}_{jac} + s^2 d\, \#\mathbb{X}$$

and, since $s \geq d$, we conclude

$$\forall \delta,\ \#\mathbb{X}_\delta \leq p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{\mathcal{F}} \lesssim s^2 d\ p^{d+1}\, N, \tag{50}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \leq N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{\mathcal{F}} \lesssim s^2 d\, p\, N, \tag{51}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \geq p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{\mathcal{F}} \lesssim s^2 d\, p\, \#\mathbb{X}. \tag{52}$$

7 A note about matrix-free implementations

In this section, we remark on matrix-free implementation. Matrix-free methods are often combined with iterative solvers. Iterative solvers typically require only the ability to compute matrix-vector products $\mathbf{v} = \mathtt{A}\mathbf{u}$, while a direct access to the entries of $\mathtt{A}$ is not required. By definition

$$\mathtt{A}\mathbf{u} = \Big(\int_{[0,1]^d} \psi_m(\mathbf{x}) u(\mathbf{x}) \Big)_{m=0}^{M},$$

where

$$u(\mathbf{x}) = \sum_{n=1}^{N} \mathbf{u}_n \phi_n(\mathbf{x}).$$

For this case, unfortunately, the cost of matrix-free setting must be counted based on M, i.e., on the dimension of the test space. So, we assume $N = M$ for simplicity.

Algorithm 1 can be changed to a matrix-free setting by removing the loop at line 8 and pre-multiplying $\mathcal{F}(\mathbb{X})$ by $u(\mathbb{X})$. This gives the following algorithm, where the blocks $\mathbf{w}_{m_d}^{\lesssim d}$ are the blocks of $M^{\lesssim d-1}$ coefficients in $\mathbf{v} = \mathbf{v}^{\lesssim d}$:

$$\mathbf{v} = (\underbrace{\mathbf{v}_1, \dots, \mathbf{v}_{M^{\lesssim d-1}}}_{\mathbf{w}_1}, \dots, \underbrace{\mathbf{v}_{M^{\lesssim d} - M^{\lesssim d-1}+1}, \dots, \mathbf{v}_{M^{\lesssim d}}}_{\mathbf{w}_{M_d}}).$$

Algorithm 3 Matrix-free sum-factorization

1: **procedure** APPLY($[\mathbb{X}_\delta]_{\delta=1}^d, [\omega_\delta]_{\delta=1}^d, [\Psi_\delta]_{\delta=1}^d, [\mathcal{F}(\mathbf{x})u(\mathbf{x})]_{x\in\mathbb{X}},$)
2: **if** d=0 **then**
3: **return** $[\mathcal{F}(\mathbf{x})u(\mathbf{x})]_{x\in\mathbb{X}}$
4: **end if**
5: $\mathbf{v}^{\lesssim d} \leftarrow 0$ ▷ start with null vector
6: **for all** $x_d \in \mathbb{X}_d$ **do** ▷ sum all $\mathbf{v}_{x_d}^{\lesssim d-1}$
7: $\mathbf{v}_{x_d}^{\lesssim d-1} \leftarrow$APPLY$([\mathbb{X}_\delta]_{\delta=1}^{d-1}, [\omega_\delta]_{\delta=1}^{d-1}, [\Psi_\delta]_{\delta=1}^{d-1}, [\mathcal{F}(\mathbf{x}, x_d)u(\mathbf{x}, x_d)]_{\mathbf{x}\in\mathbb{X}^{\lesssim d-1}})$
8: **for all** m_d **with** $\psi_{m_d}^{=d}(x_d) \neq 0$ **do**
9: $\mathbf{w}_{m_d}^{\lesssim d} \leftarrow \mathbf{w}_{m_d}^{\lesssim d} + \omega_d(x_d)\psi_{m_d}^{=d}(x_d)\mathbf{v}_{x_d}^{\lesssim d-1}$
10: **end for**
11: **end for**
12: **return** $\mathbf{v} = \mathbf{v}^{\lesssim d}$
13: **end procedure**

The above yields a cost that, excluding the computation of $u(\mathbb{X})$, is

$$\mathfrak{C}_{app}^{\lesssim d} \eqsim \#\mathbb{X}_d \big(\mathfrak{C}_{app}^{\lesssim d-1} + p_d \; M^{\lesssim d-1} \big).$$

This, remembering the simplification $M_\delta = N_\delta$, leads analogously to (15) to the following bound for the costs:

$$\mathfrak{C}_{app}^{\lesssim d} \eqsim \sum_{i=1}^{d} \Big(\prod_{\delta=i+1}^{d} \#\mathbb{X}_\delta \Big) (p_i \#\mathbb{X}_i)\ N^{\lesssim i-1}. \tag{53}$$

Depending on the relative ratio between N_δ and $\#\mathbb{X}_\delta$, we obtain

$$\forall \delta,\ \#\mathbb{X}_\delta \leq p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{app}^{\lesssim d} \lesssim d\, p^{d+1} N, \tag{54}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \leq N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{app}^{\lesssim d} \lesssim d\, p\, N, \tag{55}$$

$$\forall \delta,\ \#\mathbb{X}_\delta \geq p_\delta N_\delta \qquad \Rightarrow \qquad \mathfrak{C}_{app}^{\lesssim d} \lesssim d\, p\, \#\mathbb{X}. \tag{56}$$

By comparing the estimate (55) with (18) and (51), we observe that a matrix free approach can achieve significant saving if weighted quadrature is employed as proposed in [14]. Indeed in this case, both the computation of the coefficients and the application of the matrix are computed with a cost $\approx pN$. Note that this is a factor of p^d less than the assembling of the matrix and the same factor less than the cost of the multiplication of the assembled matrix with some given vector.

However, by comparing the estimates estimate (54) and (56) with (17), (19), (50) and (52), we observe that for standard Gauss quadrature, the potential saving is "only" for a factor of p.

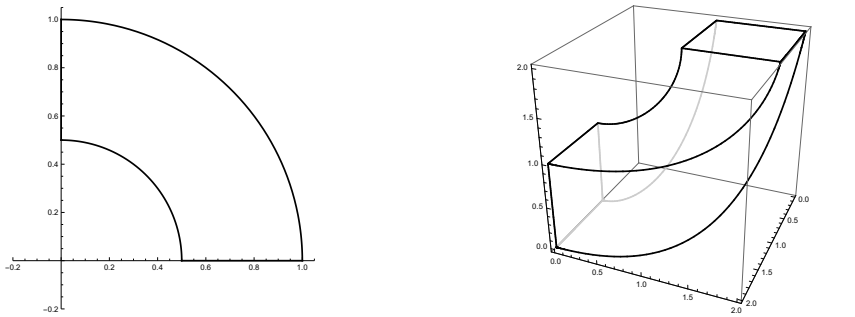
8 Numerical experiments and conclusions

We have implemented the global strategy and the per-(macro)-element strategies and have tested their behavior for a few sample problems. Our implementation is a C++ code which has been carefully optimized. It is available online[1] as an stand-alone assembling library.

In the numerical experiments, we assemble a standard stiffness matrix for the 2D and 3D domains depicted in Fig. 1. The 2D domain is decomposed into 200×200 elements, the 3D domain into $22 \times 22 \times 22$ elements. On each domain, we assemble the stiffness matrix for splines of several orders p using Gauss quadrature of the same order and we compare the time used by different algorithms. This was done on a single socket machine with an Intel(R) Core(TM) i3-8100 CPU running at 3.60GHz.

In Fig. 2 and 3, we report the assembling times for standard per-element assembling (`standard`), per-element sum factorization (`element`), per-macro-element sum factorization (`macroS`) and global sum-factorization (`global`). The macro-elements have size $p \times p$ in $2D$ and $p \times p \times p$ in $3D$. Standard per-element

[1] `https://github.com/IgASF/IgASF`



(a) 2D domain: quarter annulus

(b) 3D domain: bended and twisted box

Figure 1: Computational domains

assembling uses the same Gauss quadrature and was performed using the freely available IGA library G+Smo [5].

In Fig. 4 and 5, we focus on the effect of the macro-element size. The considered approaches are: standard macro-elements (`macroS`) corresponding to a size of $p \times p$ in 2D and $p \times p \times p$ in 3D, narrow macro-elements (`macroN`) having size $p \times 1$ in 2D and $p \times p \times 1$ in 3D and rotated narrow macro-elements (`macroR`) having size $1 \times p$ in 2D and $1 \times p \times p$ in 3D. According to Example 7, the narrow dimension should come last in sum factorization. As our code does not reorder the dimensions we expect the same behavior for `macroS` and `macroN`, but a cost higher by a factor p for `macroR`. This means that `macroR` behaves as `element` in $2D$ and as $p^6 N$ in 3D.

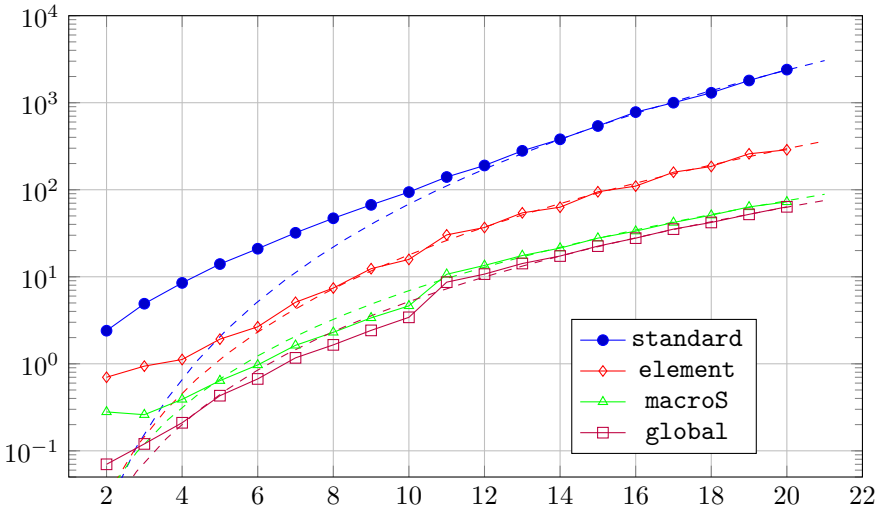


Figure 2: Time in seconds for assembling the stiffness matrix on the 2D domain depending on the polynomial order p.

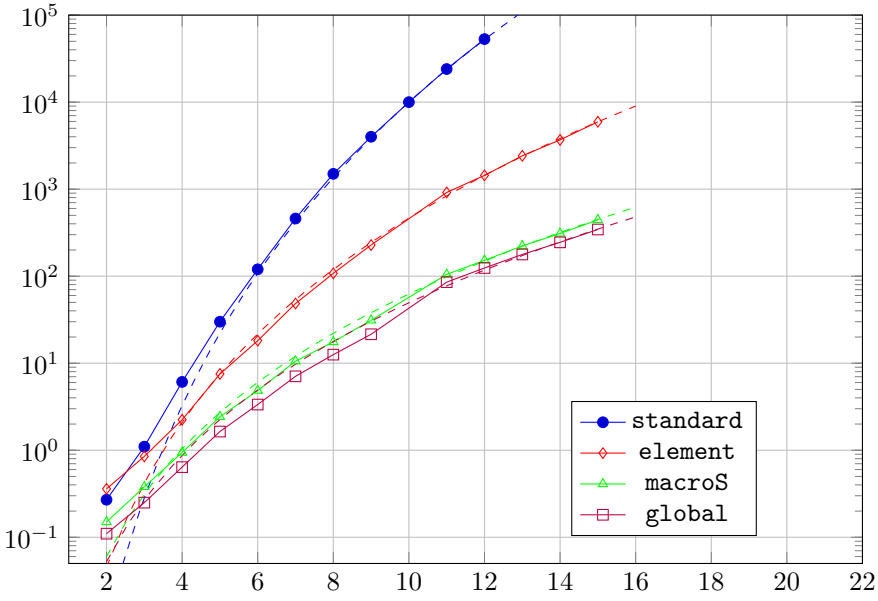


Figure 3: Time in seconds for assembling the stiffness matrix on the 3D domain depending on the polynomial order p.

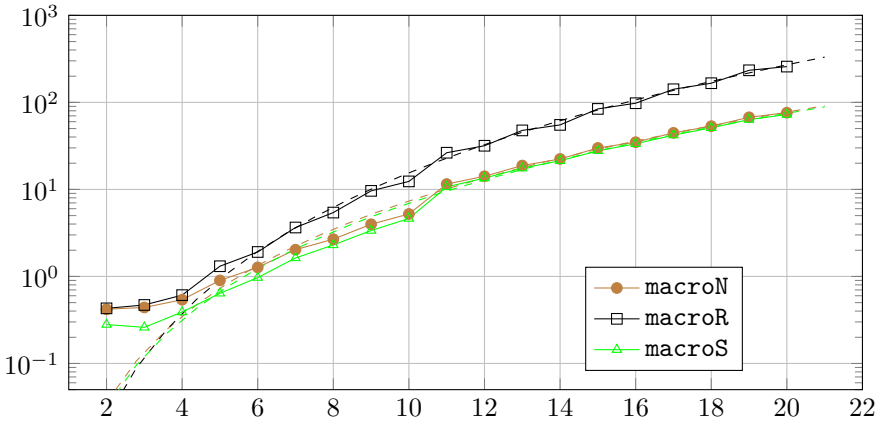


Figure 4: Time in seconds for assembling the stiffness matrix on the 2D domain with macro-elements depending on the polynomial order p.

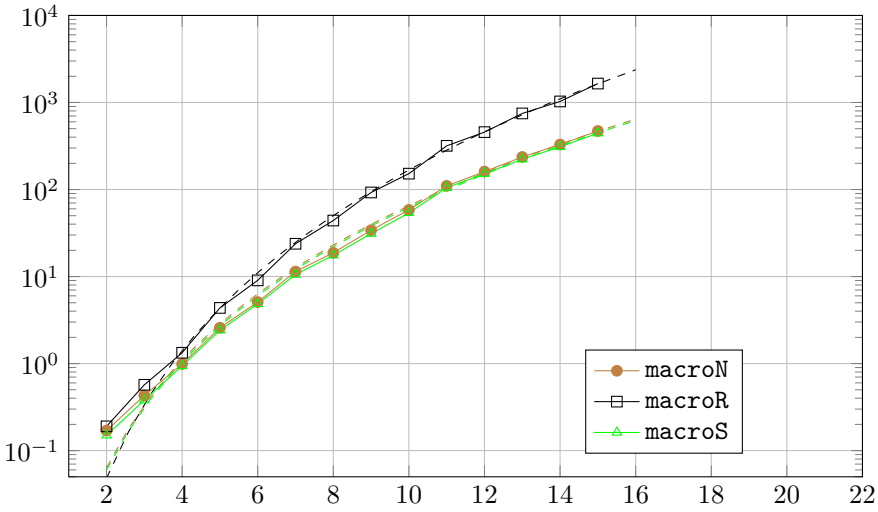


Figure 5: Time in seconds for assembling the stiffness matrix on the 3D domain using macro-elements depending on the polynomial order p.

We observe that in any case, the sum factorization approaches are faster than the standard approach. Conforming with the theory, we see that `global` is significantly faster than `element`. Moreover, we obtain that the macro-element approaches `macro` and `macroN` are indeed almost as fast as the global sum-factorization approach `global`.

The results are used for fitting the parameters c and e in the formula

$$t = cp^e N(p),$$

where t is the measured time, $N(p) = \prod_{\delta=1}^{d}(p+K_\delta-1)$ is the number of degrees of freedom and K_δ is the number of elements in the corresponding direction, i.e., $K_1 = K_2 = 200$ for $d = 2$ and $K_1 = K_2 = K_3 = 22$ for $d = 3$. The fitted curves are dashed in Fig. 2–5. The fitting procedure yields the following values for the exponent e:

	2D domain		3D domain	
	theory	experiments	theory	experiments
`standard`	6	4.98	9	8.05
`element`	5	3.92	7	5.18
`macroN`	4	3.26	5	3.75
`macroR`	5	3.99	6	4.51
`macroS`	4	3.30	5	3.76
`global`	4	3.47	5	3.70

		threads											
dim	p	1	2	4	6	8	12	16	20	24	28	32	36
2	3	1	1.9	3.5	4.7	5.9	7.6	9.3	10.6	11.3	12.4	13.5	12.8
2	5	1	2.0	3.6	4.9	6.2	8.5	10.3	11.9	14.0	15.2	15.8	16.1
2	7	1	2.0	3.8	5.1	6.6	8.9	11.0	13.2	15.2	16.9	18.4	19.8
3	3	1	1.9	3.7	5.2	6.6	8.9	11.0	12.9	14.2	16.7	15.0	17.4
3	5	1	2.0	3.7	5.2	6.6	9.0	10.9	12.8	14.3	15.2	15.6	17.6
3	7	1	2.0	3.7	5.0	6.6	8.3	10.9	11.4	13.6	13.6	17.4	17.5

Table 1: Speed-up factors for assembling the stiffness matrix using a parallel macro-element implementation.

We observe that the exponents e obtained in our experiments are significantly lower than those predicted by the theory. We believe that the huge difference between the speed in performing computation and the speed in accessing memory in modern processors, is masking one order in p. For the 3D example, there is also another reason: the fitting is distorted by the small number of elements, i.e., the approximation $p^d N \approx \#\mathbb{X}$ does not apply: $L_\delta = 22$ so that $pN_\delta = 22p + p^2 - p$ is twice bigger than $\#\mathbb{X}_\delta = 22p$ for $p = 20$.

In Fig. 6 and 7, we report the assembling times using a proof-of-concept multi-threaded approach based on macro-elements of size $p \times p$ in 2D and $p \times p \times p$ in 3D (`macroS`). To have a sufficient number of macro-elements the 2D domain is split in 2000×2000 elements and the 3D domain is split in $64 \times 64 \times 64$ elements. The lines correspond to the polynomial orders $p = 4, 6$ and 8, and the abscissa is the number of parallel threads. The tests have been executed on a dual socket machine with Intel(R) Xeon(R) CPU E5-2695 v4 running at 2.10GHz. The figures show that the macro-element approach is viable to parallelization. Table 1 shows the corresponding speed-up factors, i.e. the ratio between the time for the single-threaded execution and the time for the multi-threaded execution.

In this paper, we have developed a unified complexity analysis for sum factorization approaches. The theory shows that in Isogeometric Analysis, the computational costs can be reduced significantly by using the technique of sum factorization. This might be of interest because sum factorization can use the standard Gauss quadrature yielding exactly the same matrix as the standard approach.

We show that sum factorization does not yield its optimal complexity if it is applied on each element separately. However, we do not need to consider the global approach to obtain its optimal complexity: it is sufficient to apply it to blocks of at least p elements in each direction.

Moreover, we have shown that parallel implementations of localized sum-factorization are a viable strategy for fast assembling of the system matrix in IgA applications.

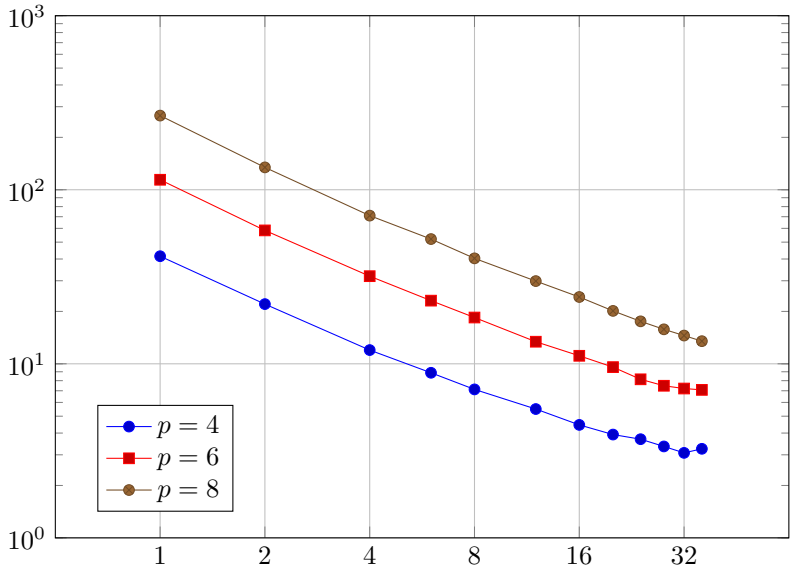


Figure 6: Times in seconds for assembling the stiffness matrix on the 2D domain using a parallel macro-element implementation.

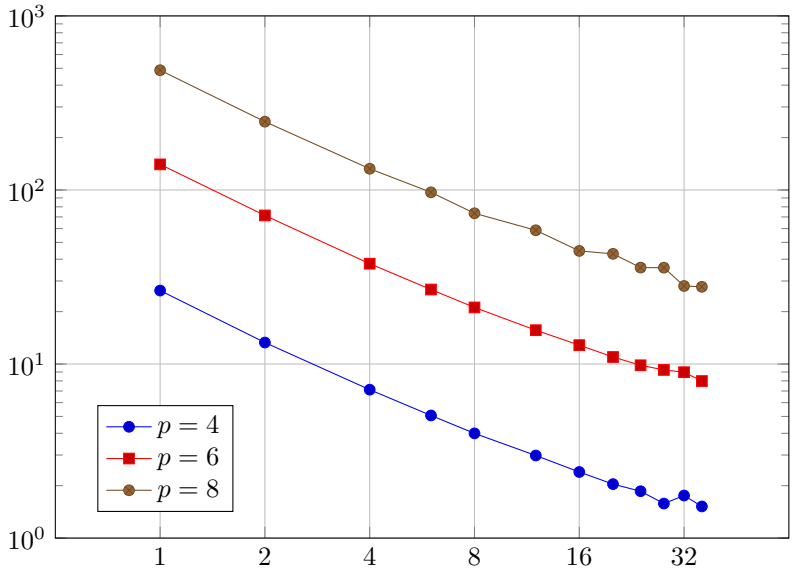


Figure 7: Times in seconds for assembling the stiffness matrix on the 3D domain using a parallel macro-element implementation.

Acknowledgments

The first author has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement 339643. The second author was partially funded by the Austrian Science Fund (FWF) under grant NFN S117-03

References

[1] N. Ainsworth, G. Andriamaro, and O. Davydov, *Bernstein-Bézier finite elements of arbitrary order and optimal assembly procedures*, SIAM J. Sci. Comput. **33** (2011), no. 6, 3087–3109.

[2] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli, *Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization*, Comput. Methods Appl. Mech. Eng. **285** (2015), 817–828.

[3] A. Bressan and B. Jüttler, *A hierarchical construction of LR meshes in 2D*, Comput. Aided Geom. Design **37** (2015), 9–24.

[4] A. Bressan and G. Sangalli, *Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique*, IMA J. Numer. Anal. **33** (2013), no. 2, 629–651.

[5] A. Bressan, S. Takacs, A. Mantzaflaris, et al., *G+Smo*, `http://gs.jku.at/gismo`, 2018.

[6] A. Buffa and C. Giannelli, *Adaptive isogeometric methods with hierarchical splines: error estimator and convergence*, Math. Models Methods Appl. Sci. **26** (2016), no. 1, 1–25.

[7] F. Calabrò, G. Sangalli, and M. Tani, *Fast formation of isogeometric Galerkin matrices by weighted quadrature*, Comput. Methods Appl. Mech. Eng. **316** (2017), 606–622.

[8] C. Giannelli, B. Jüttler, and H. Speleers, *THB-splines: the truncated basis for hierarchical splines*, Comput. Aided Geom. Design **29** (2012), no. 7, 485–498.

[9] C. Hofreither, *A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis*, Comput. Methods Appl. Mech. Eng. **333** (2018), 311–330.

[10] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Methods Appl. Mech. Eng. **194** (2005), no. 39-41, 4135–4195.

[11] R. Kraft, *Adaptive and linearly independent multilevel B–splines*, Surface Fitting and Multiresolution Methods (A. Le Méhauté, C. Rabut, and L. L. Schumaker, eds.), Vanderbilt University Press, Nashville, 1997, pp. 209–218.

[12] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer, *Low rank tensor methods in Galerkin-based isogeometric analysis*, Comput. Methods Appl. Mech. Eng. **316** (2017), 1062–1085.

[13] J.M. Melenk, K. Gerdes, and C. Schwab, *Fully discret hp-finite elements: Fast quadrature*, Comput. Methods Appl. Mech. Eng. **190** (2001), no. 32–33, 4339–4364.

[14] G. Sangalli and M. Tani, *Matrix-free weighted quadrature for a computationally efficient isogeometric k-method*, Comput. Methods Appl. Mech. Eng. **338** (2018), 117–133.