Partial tensor decomposition for decoupling isogeometric Galerkin discretizations

Felix Scholz[a], Angelos Mantzaflaris[a,b], Bert Jüttler[a,b]

[a] *Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences, Linz, Austria*
[b] *Institute of Applied Geometry, Johannes Kepler University Linz, Austria*

Abstract

System matrix assembly for isogeometric (i.e., spline-based) discretizations of partial differential equations is more challenging than for classical finite elements, due to the increased polynomial degrees and the larger (and hence more overlapping) supports of the basis functions. The global tensor-product structure of the discrete spaces employed in isogeometric analysis can be exploited to accelerate the computations, using sum factorization, precomputed look-up tables, and tensor decomposition. We generalize the third approach by considering partial tensor decompositions. We show that the resulting new method preserves the global discretization error and that its computational complexity compares favorably to the existing approaches. Moreover, the numerical realization simplifies considerably since it relies on standard techniques from numerical linear algebra.

Keywords: isogeometric analysis, tensor decomposition, numerical integration, low-rank approximation, matrix assembly, singular value decomposition

1. Introduction

Isogeometric discretizations (see [1]) possess significant advantages for the numerical solution of partial differential equations (PDEs). These include the higher smoothness of the obtained numerical solution (when using higher polynomial degree), the compatibility of the representation with models coming from computer-aided design (CAD) systems, as well as the reduction of the number of degrees of freedom required to reach a prescribed accuracy level. However, these advantages come at the price of increased computation costs per degree of freedom, and this effect becomes even more pronounced as the dimension increases [2]. The computational efficiency challenges manifest themselves both in the matrix assembly, which is in the focus of this work, and in the solution of the resulting linear systems, see e.g. [3, 4] for recent advances in that direction.

There are several lines of work which aim at improving the computational efficiency of isogeometric computations. First, quadrature rules with a reduced (or even minimal)

Email addresses: felix.scholz@oeaw.ac.at (Felix Scholz), angelos.mantzaflaris@oeaw.ac.at (Angelos Mantzaflaris), bert.juettler@jku.at (Bert Jüttler)

number of nodes have been studied [5, 6, 7, 8]. These rules are defined for univariate spline spaces and require less quadrature points compared to standard Gaussian quadrature, taking into account the higher regularity of the space. Specialized reduced quadrature rules have also emerged [9, 10, 11] recently. The generalization to multivariate integrals is achieved by a tensor-product approach.

Second, the collocation approach to discretization of PDEs has been extensively explored in isogeometric analysis [12, 13]. Intuitively, collocation may be thought as one-point quadrature, since approximately one evaluation per element is required for highly-smooth B-spline discretizations. In [14] the efficiency of the collocation approach is increased by identifying collocation points with optimal approximation properties. However, the method acts on the strong form of the PDE at hand, thereby sacrificing some of the benefits of the Galerkin method.

Third, the built-in tensor-product structure of isogeometric discretizations has been exploited to improve efficiency. The evaluation of mass and stiffness matrices was addressed in [15, 16], based on small look-up tables for univariate B-spline integral. These are used in conjunction with an interpolation approach that transforms the integrands into spline functions. Building on these ideas, singular value decomposition (SVD) is used in [17] for the decomposition of bivariate integrals into univariate integrals. This has been generalized to higher dimensions in [18], using a similar decomposition approach for tensors.

A different idea to exploit the tensor-product structure has been explored in [19]: the authors use quadrature rules which are exact for all integrals that involve a fixed basis function. This approach is combined with sum factorization [20]. More recently, the tensor-product structure has been employed for constructing fast preconditioners for the linear systems that arise from isogeometric discretisation [21].

Tensor methods are quite effective when dealing with high-dimensional operators [22, 23, 24] and have found applications in several fields where structured data are present [25, 26]. Recently, tensor decomposition has been proposed as a general approach to reduce the complexity of simulations in isogeometric analysis [18]. While in 2D this approach requires solely standard linear algebra tools, its extension to higher dimensions requires advanced decomposition algorithms for tensors, such as higher-order singular value decomposition (HOSVD) and alternating least squares (ALS). Those methods are not supported by all standard scientific computing libraries. Moreover, they require non-linear optimization and their properties are not yet fully understood. In contrast to this, singular value decomposition (SVD) of matrices – which is the main tool required for the 2D case – is well established and highly optimized implementations are provided by linear algebra packages.

The present approach improves on [18]. We employ SVD in order to decouple isogeometric discretizations partially, while maintaining a quasi-optimal complexity for the task of matrix formation. This demonstrates that for efficient 3D isogeometric matrix computations, sophisticated tools such as HOSVD/ALS can be replaced by standard SVD. More precisely, we split the domain variables into two groups and perform SVD with respect to this partition. Consequently, our initial 3D matrix computation problem is replaced by a set of univariate and bivariate integration problems, which are solved independently, and provide a Kronecker product representation of the original matrix. We show that the overall asymptotic complexity remains quasi-optimal, and the obtained rank values for this decomposition are in the worst case equal to the case of a full 3D

decomposition and can be much lower in practice.

The rest of the paper is organized as follows: First we recall in Section 2 the theory of singular value decomposition of functions, in particular in finite dimensional function spaces. We also present our model problem and recall the isogeometric Galerkin method. We then apply the theory of singular value decomposition to decouple the isogeometric discretisation in Section 3. The following section examines the assembly of the system matrices using the decoupled discretisation. Based on this we analyze the consistency errors and their contribution to the total error in Section 5. We complete the analysis by discussing the complexity of our method and by comparing it to the full decomposition method from [18] in Section 6. Finally, we present the results of our numerical experiments.

2. Preliminaries

This section has three parts. First we recall existing results concerning singular value decomposition of functions in tensor-product spaces. The remaining two parts describe the model problem and its isogeometric discretization.

2.1. Singular value decomposition in finite dimensional function spaces

Let $\Omega_1 \subset \mathbb{R}^{d_1}$ and $\Omega_2 \subset \mathbb{R}^{d_2}$ be domains of dimensions $d_1, d_2 \in \mathbb{N}$. We will use the singular value decomposition (SVD) of matrices to compute low-rank approximations of functions

$$f : \Omega_1 \times \Omega_2 \longrightarrow \mathbb{R}$$

in a tensor–product space $\Phi \otimes \Psi$ generated by two finite–dimensional spaces $\Phi \subset C(\Omega_1)$ and $\Psi \subset C(\Omega_2)$. Analogously to the rank of a matrix, the rank of a function $g \in \Phi \otimes \Psi$, denoted by $\operatorname{rank}(g)$, is defined to be the smallest integer such that

$$g(\xi) = \sum_{r=1}^{\operatorname{rank}(g)} g_r^1(\xi_1) g_r^2(\xi_2)$$

with $\{g_r^1\} \subset \Phi$ and $\{g_r^2\} \subset \Psi$.

First we recall the SVD of a matrix and some of its properties, see e.g. [25, Section 2.5.3]: For any matrix $C \in \mathbb{R}^{\mu \times \nu}$, there exist orthogonal matrices $U \in \mathbb{R}^{\mu \times \mu}$, $V \in \mathbb{R}^{\nu \times \nu}$ and a rectangular diagonal matrix $\Sigma \in \mathbb{R}^{\mu \times \nu}$, such that

$$C = U \Sigma V^T = \sum_{r=1}^{\min(\mu,\nu)} \sigma_r \mathbf{u}_r \mathbf{v}_r^T,$$

where the diagonal entries of Σ, which are called the singular values, satisfy $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_{\min(\mu,\nu)} \geq 0$ and $\mathbf{u}_r$ and $\mathbf{v}_r$ are the columns of the matrices U and V. By defining the truncated diagonal matrix

$$(\Sigma_R)_{ij} = \begin{cases} \sigma_i & \text{for } i = j \leq R \\ 0 & \text{otherwise} \end{cases}$$

for any positive integer R, we obtain the *rank-R approximation*

$$C_R = U\Sigma_R V^T = \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^T \tag{1}$$

of the given matrix C.

In order to apply SVD to the function approximation problem, we choose two bases

$$\boldsymbol{\phi} = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_\mu \end{pmatrix} \text{ and } \boldsymbol{\psi} = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_\nu \end{pmatrix}$$

for the two finite-dimensional spaces $\Phi \subset C(\Omega_1)$ and $\Psi \subset C(\Omega_2)$ of dimensions μ and ν, respectively. The Euclidean inner product of the coefficients then defines the inner product

$$\langle h, h' \rangle_\Phi = \mathbf{h}^T \mathbf{h}' = \sum_{i=1}^{\mu} h_i h_i'$$

on Φ (and analogously for Ψ), where the functions are represented as

$$h = \langle \mathbf{h}, \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} = \sum_{i=1}^{\mu} h_i \phi_i \quad \text{and} \quad h' = \langle \mathbf{h}', \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} = \sum_{i=1}^{\mu} h_i' \phi_i$$

with coefficient vectors $\mathbf{h}$ and $\mathbf{h}'$. These inner products on Φ and Ψ induce an inner product on

$$\Phi \otimes \Psi \subset C(\Omega_1 \times \Omega_2),$$

which is equal to the inner product defined by the tensor–product basis $\boldsymbol{\phi} \otimes \boldsymbol{\psi}$. We find it convenient to express it via the Frobenius inner product $\langle ., . \rangle_F$ of matrices,

$$\langle g, g' \rangle_{\Phi \otimes \Psi} = \langle G, G' \rangle_F = \sum_{i=1}^{\mu} \sum_{j=1}^{\nu} G_{ij} G_{ij}',$$

since the functions $g, g' \in \Phi \otimes \Psi$ have coefficient matrices G and G' with respect to the basis $\boldsymbol{\phi} \otimes \boldsymbol{\psi}$. An analogous relation connects the Euclidean coefficient norm $\|.\|_{\Phi \otimes \Psi}$ with the Frobenius norm $\|.\|_F$ of the coefficient matrices.

The Frobenius inner product also furnishes the compact notation

$$f = \langle C, \boldsymbol{\phi} \otimes \boldsymbol{\psi} \rangle_F = \sum_{i=1}^{\mu} \sum_{j=1}^{\nu} C_{ij} \phi_i \psi_j, \tag{2}$$

where $C \in \mathbb{R}^{\mu \times \nu}$ is the coefficient matrix, for the representation of any function $f \in \Phi \otimes \Psi$ with respect to the tensor-product basis.

Combining (2) with the SVD of the coefficient matrix C leads to the decomposition

$$f = \sum_{r=1}^{\min(\mu,\nu)} \sigma_r \langle \mathbf{u}_r, \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} \langle \mathbf{v}_r, \boldsymbol{\psi} \rangle_{\mathbb{R}^\nu} . \tag{3}$$

Considering again the truncated diagonal matrix, we obtain the *rank-R approximation*

$$f_R = \langle C_R, \boldsymbol{\phi} \otimes \boldsymbol{\psi} \rangle_F = \sum_{r=1}^{R} \sigma_r \, \langle \mathbf{u}_r, \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} \, \langle \mathbf{v}_r, \boldsymbol{\psi} \rangle_{\mathbb{R}^\nu} \ , \tag{4}$$

of the function f with respect to the tensor product basis $\boldsymbol{\phi} \otimes \boldsymbol{\psi}$.

Lemma 1. *The rank-R approximation* (4) *solves the best approximation problem*

$$\min_{\{g \in \Phi \otimes \Psi \colon \operatorname{rank}(g) \leq R\}} \|f - g\|_{\Phi \otimes \Psi}$$

and the resulting approximation error equals

$$\|f - f_R\|_{\Phi \otimes \Psi} = \|C - C_R\|_F = \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2}.$$

Proof. It is known that the approximation error of the rank-R approximation C_R of the matrix C in the Frobenius norm is given by

$$\|C - C_R\|_F = \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2} \tag{5}$$

and it is the best approximation of C by a matrix of rank R in the Frobenius norm, see [25, Lemma 2.30]. With our choice of norm on $\Phi \otimes \Psi$ these properties are transferred to the function f and its rank-R approximation f_R. □

We establish a simple bound on the approximation error in the L^∞ norm:

Lemma 2. *Given f as in* (2) *and its rank-R approximation f_R as in* (4) *we have*

$$\|f - f_R\|_{L^\infty(\Omega_1 \times \Omega_2)} \leq \big(\max_{\xi \in \Omega_1 \times \Omega_2} \|(\boldsymbol{\phi} \otimes \boldsymbol{\psi})(\xi)\|_F \big) \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2} \, .$$

Proof. We rewrite the L^∞ norm of the approximation error and use the Cauchy-Schwarz inequality to obtain

$$\begin{aligned} \|f - f_R\|_{L^\infty} &= \|\langle C, \boldsymbol{\phi} \otimes \boldsymbol{\psi} \rangle_F - \langle C_R, \boldsymbol{\phi} \otimes \boldsymbol{\psi} \rangle_F\|_{L^\infty} \\ &= \max_{\xi \in \Omega_1 \times \Omega_2} |\langle C - C_R, (\boldsymbol{\phi} \otimes \boldsymbol{\psi})(\xi) \rangle_F| \\ &\leq \big(\max_{\xi \in \Omega_1 \times \Omega_2} \|(\boldsymbol{\phi} \otimes \boldsymbol{\psi})(\xi)\|_F \big) \, \|C - C_R\|_F. \end{aligned}$$

The result now follows from (5). □

This result is useful if we are able to control the constant that appears in front of the square root. For instance, this constant can be evaluated easily when using tensor-product B-splines.

Another result can be derived for the approximation error in the L^2 norm if one uses L^2–orthonormal bases $\{\phi_i\}_{i=1}^{\mu}$ and $\{\psi_j\}_{j=1}^{\nu}$. Indeed, the coefficient-based inner products defined on Φ, Ψ and $\Phi \otimes \Psi$ are then equal to the corresponding L^2 products. Consequently, the rank-R approximation is the best approximation with respect to the L^2-norm and the error satisfies

$$\|f - f_R\|_{L^2} = \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2}. \tag{6}$$

In this case, the representation (3) is equal to the SVD of an L^2 function, which exists for any function in $L^2(\Omega_1 \times \Omega_2)$, see [25, Corollary 4.115]. In the infinite dimensional case, any function $f \in L^2(\Omega_1 \times \Omega_2)$ has a decomposition

$$f = \sum_{r=1}^{\infty} \sigma_r u_r v_r$$

where $\{u_r\}_{r=1}^{\infty}$ is an orthonormal system of $L^2(\Omega_1)$ and $\{v_r\}_{r=1}^{\infty}$ is an orthonormal system of $L^2(\Omega_2)$. The singular values are known to decay like

$$\sigma_r \lesssim r^{-\frac{s}{\min(d_1,d_2)}},$$

if $f \in H^s(\Omega_1 \times \Omega_2)$, see [27, Theorem 3.1].

2.2. Model problem and isogeometric formulation

Let L be a differential operator

$$Lu = -\nabla \cdot (A(x)\nabla u) + \boldsymbol{b}(x) \cdot \nabla u + c(x)u. \tag{7}$$

We assume L to be elliptic, the coefficients to be in L^∞ and A to be symmetric.

Our goal is to approximate the weak solution of the boundary value problem (BVP)

$$\begin{cases} Lu &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma_D, \end{cases} \tag{8}$$

on a domain $\Omega \subset \mathbb{R}^3$ which we assume to be parameterized by a regular diffeomorphism

$$G : \hat{\Omega} \longrightarrow \Omega$$

on the parameter domain $\hat{\Omega} = [0,1]^3$.

Since we only want to work on the parameter domain and not on the physical domain, the next step is to transform the differential operator L to functions $\hat{u} = u \circ G$ on the parameter domain. A short computation confirms that

$$\hat{L}\left(\hat{u} \circ G^{-1}\right) = \frac{-1}{|\det J_G|} \hat{\nabla} \cdot \left(|\det J_G| J_G^{-1} A J_G^{-T} \hat{\nabla} \hat{u}\right) + (J_G^{-1}\boldsymbol{b}) \cdot \hat{\nabla}\hat{u} + c\hat{u}.$$

Thus, we transform the original BVP to an equivalent BVP

$$\begin{cases} \hat{L}\hat{u} &= \hat{f} \quad \text{in } \hat{\Omega}, \\ \hat{u} &= 0 \quad \text{on } \hat{\Gamma}_D, \end{cases} \tag{9}$$

on the parameter domain $\hat{\Omega}$ by setting $\hat{f} = |\det J_G| f \circ G$ and

$$\hat{L}\hat{u} = |\det J_G| L\left(\hat{u} \circ G^{-1}\right) = -\hat{\nabla} \cdot (K\hat{\nabla}\hat{u}) + \boldsymbol{\ell} \cdot \hat{\nabla}\hat{u} + m\hat{u},$$

where

$$K = |\det(J_G)|(J_G)^{-1} A (J_G)^{-T}, \tag{10}$$

$$\boldsymbol{\ell} = |\det(J_G)|(J_G)^{-1}\boldsymbol{b}, \tag{11}$$

$$m = |\det(J_G)|c. \tag{12}$$

The associated bilinear form of $\hat{L}$ on $V \times V$, where $V = H_0^1(\hat{\Omega})$, is defined as

$$\hat{b}(\hat{u}, \hat{v}) = \int_{\hat{\Omega}} \hat{\nabla}\hat{u} \cdot (K\hat{\nabla}\hat{v}) + (\boldsymbol{\ell} \cdot \hat{\nabla}\hat{u})\hat{v} + m\hat{u}\hat{v} \,\mathrm{d}\xi \tag{13}$$

and the weak formulation of the transformed problem on the parameter domain is

$$\hat{b}(\hat{u}, \hat{v}) = \hat{\lambda}(\hat{v}) \text{ for all } \hat{v} \in V, \tag{14}$$

where the right hand side is given by the linear functional

$$\hat{\lambda}(\hat{v}) = \int_{\hat{\Omega}} \hat{f}\hat{v} \,\mathrm{d}\xi. \tag{15}$$

Finally we note that if $\hat{u} \in V$ is a weak solution to the BVP on the parameter domain, then

$$u = \hat{u} \circ G^{-1} \in H_0^1(\Omega)$$

is a weak solution to the BVP on the physical domain.

2.3. Isogeometric discretisation

Under certain conditions on the coefficients, the weak problem (14) has a unique solution. Assuming that a unique solution exists, we approximate it using an isogeometric Galerkin method, which means that we perform a discretization based on tensor–product spline functions.

More precisely, we choose three univariate spline spaces $S^{p_\ell}_{\boldsymbol{\tau}_\ell}$ of degree p_ℓ with open knot vectors $\boldsymbol{\tau}_\ell$, $\ell = 1, 2, 3$, and consider their tensor-product

$$\mathbb{S} = S^{p_1}_{\boldsymbol{\tau}_1} \otimes S^{p_2}_{\boldsymbol{\tau}_2} \otimes S^{p_3}_{\boldsymbol{\tau}_3}.$$

To simplify notation, we shall omit the indices for the polynomial degrees. We denote by

$$\{\beta^1_i\}_{i\in\{1,\dots,n_1\}}, \{\beta^2_j\}_{j\in\{1,\dots,n_2\}}, \text{ and } \{\beta^3_k\}_{k\in\{1,\dots,n_3\}}$$

the basis functions of the univariate spline spaces. We assume that homogeneous Dirichlet boundary conditions have been implemented in each of the spaces. The basis functions of the tensor–product spline space $\mathbb{S}$ are given as the tensor product of the univariate bases. We denote them by

$$\beta_{ijk}(\xi) = \beta^1_i(\xi_1)\beta^2_j(\xi_2)\beta^3_k(\xi_3).$$

In addition we will use the basis functions

$$\beta_{ij}(\xi_{12}) = \beta_{ij}(\xi_1, \xi_2) = \beta_i^1(\xi_1)\beta_j^2(\xi_2)$$

of the bivariate spline space $S_{\tau_1}^{p_1} \otimes S_{\tau_2}^{p_2}$, were we use the notation $\xi_{12} = (\xi_1, \xi_2)$. Whenever it is clear from context we will omit the upper indices on the univariate basis functions.

In order to assemble the system matrix for the Galerkin method we need to evaluate the bilinear form (13) for all pairs of basis functions of $\mathbb{S}$. In particular we have to compute the stiffness matrix S with elements

$$\begin{aligned} S_{(ijk),(i'j'k')} &= \int_{(0,1)^3} \hat{\nabla}\beta_{ijk}(\xi) \left(K(\xi)\hat{\nabla}\beta_{i'j'k'}(\xi) \right) \mathrm{d}\xi \\ &= \sum_{p,q=1}^{3} \int_{(0,1)^3} \frac{\partial}{\partial \xi_p}\beta_{ijk} K_{pq} \frac{\partial}{\partial \xi_q}\beta_{i'j'k'} \,\mathrm{d}\xi, \end{aligned} \tag{16}$$

the advection matrix C with elements

$$C_{(ijk),(i'j'k')} = \int_{(0,1)^3} (\boldsymbol{\ell}(\xi) \cdot \hat{\nabla}\beta_{ijk}(\xi))\beta_{i'j'k'}(\xi) \,\mathrm{d}\xi = \sum_{p=1}^{3} \int_{(0,1)^3} \boldsymbol{\ell}_p \frac{\partial}{\partial \xi_p}\beta_{ijk}\beta_{i'j'k'} \,\mathrm{d}\xi \tag{17}$$

and the mass matrix M with elements

$$M_{(ijk),(i'j'k')} = \int_{(0,1)^3} m(\xi)\beta_{ijk}(\xi)\beta_{i'j'k'}(\xi) \,\mathrm{d}\xi. \tag{18}$$

Here, we use (ijk) to denote a lexicographical ordering of the components.

The approximation u_h of the exact solution to the weak problem (14) is then

$$u_h = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} u_{ijk}\beta_{ijk} \,,$$

where the coefficients u_{ijk} are obtained by solving the linear problem

$$Bu = F.$$

The matrix representation B of the bilinear form $\hat{b}(\cdot,\cdot)$ with respect to the B-spline basis is the sum of S, C and M. The right hand side vector F is computed accordingly and its entries are inner products of the form $(\hat{f}, \beta_{ijk})_{L^2}$.

The spline discretization $u_h \in \mathbb{S}$ of the transformed problem (9) is an *isogeometric discretization* of the original boundary value problem (8) if the regular diffeomorphism G, which is called the geometry mapping, is an element of the space $\mathbb{S}^3$. In this situation one uses the same space for discretizing the problem and for representing the geometry.

More generally, one considers geometry mappings of the form G/w, where $w \in \mathbb{S}$ is a non-negative denominator. In this situation, the transformation of the original problem (8) to an equivalent BVP on the parameter domain needs to use the substitution $u = (\hat{u}/w) \circ G^{-1}$. We omit any details, since this leads to slightly involved formulas for the quantities K, $\boldsymbol{\ell}$ and m that define the equivalent BVP (9). The remaining results of this paper apply to the more general situation also. Again, one obtains an isogeometric discretization by considering spline functions $u_h \in \mathbb{S}$.

3. Decoupling the isogeometric Galerkin discretisation

In order to evaluate the trivariate integrals in the components of the system matrices in an efficient way, we will replace the components of K, $\boldsymbol{\ell}$ and m as well as the right hand side $\hat{f}$ by sums of products of bivariate and univariate functions. These components define *weight functions* that are present in the trivariate integrals. By performing the replacement step we are able to replace the expensive trivariate numerical integration by a number of univariate and bivariate ones. Clearly, the error introduced by this operation needs to be controlled carefully.

As the first step, we project each weight function into an appropriate spline space. In some cases, the weight function is already contained in some spline space, hence the projection does not introduce any error. Otherwise we choose the spline space such that the error ϵ_Π does not exceed a given error tolerance. The resulting approximate weight function is represented by tensor–product splines.

In the second step, we decompose the approximate weight function using the results presented in Section 2.1. To this end, we consider the factorization of its domain as the Cartesian product $[0,1]^2 \times [0,1]$ (or any of the three possible splittings of $[0,1]^3$) and use the associated low-rank approximation. Given a tolerance ϵ_Λ, we obtain a rank-R approximation of the approximate weight function.

3.1. Spline projection

Let g be one of the weight functions, i.e., any component K_{pq} of K, any component $\boldsymbol{\ell}_p$ of $\boldsymbol{\ell}$ or either of the functions m and $\hat{f}$. We project g into some tensor–product spline space

$$\bar{\mathbb{S}} = S^{\bar{p}_1}_{\gamma_1} \otimes S^{\bar{p}_2}_{\gamma_2} \otimes S^{\bar{p}_3}_{\gamma_3} = \left(S^{\bar{p}_1}_{\gamma_1} \otimes S^{\bar{p}_2}_{\gamma_2}\right) \otimes S^{\bar{p}_3}_{\gamma_3}. \tag{19}$$

Note that in (19) we can choose each of the three possible splittings of the tensor-product space by combining two of the univariate spline spaces. In the complexity analysis performed in Section 6.1 we will observe that it is beneficial to choose the one resulting in the lowest rank. In general, this space will be different from the discretisation space $\mathbb{S}$. Since the weight functions depend on the geometry mapping, we keep the knots of its spline representation. The choice of $\bar{\mathbb{S}}$ will be discussed later on in this section.

Using the notation of Section 2.1, the splitting (19) of $\bar{\mathbb{S}}$ corresponds to

$$\bar{\mathbb{S}} = \Phi \otimes \Psi, \quad \text{where} \quad \Phi = S^{q_1}_{\gamma_1} \otimes S^{q_2}_{\gamma_2}, \quad \Psi = S^{q_3}_{\gamma_3}. \tag{20}$$

Instead of separating the last variable of the tensor–product spline space from the first two, we could have selected each of the three variables. The potential benefits will be analyzed experimentally in Section 7.

Choosing bases $\boldsymbol{\phi} = \{\bar{\beta}_{ij}\}$ of $\Phi = S^{\bar{p}_1}_{\gamma_1} \otimes S^{\bar{p}_2}_{\gamma_2}$ and $\boldsymbol{\psi} = \{\bar{\beta}_k\}$ of $\Psi = S^{\bar{p}_3}_{\gamma_3}$ results in the usual tensor-product basis functions

$$\bar{\beta}_{ijk}(\xi) = \bar{\beta}_{ij}(\xi_1, \xi_2)\bar{\beta}_k(\xi_3)$$

of the spline space $\Phi \otimes \Psi = \bar{\mathbb{S}}$. The particular choice of the basis will be determined by the error norm that we want to control. In particular we will use either B-splines or an L^2-orthonormal basis of the spline space, see next section.

We now consider a spline projection operator

$$\Pi : C([0,1]^3) \to \bar{\mathbb{S}},$$

which will be either interpolation at the Greville points or orthogonal L^2 projection. Both can be implemented efficiently by exploiting the tensor-product structure $\Pi = \Pi_1 \otimes \Pi_2 \otimes \Pi_3$, since one can realize them by sequentially applying the corresponding univariate operators. The particular choice of the projector is again related to the error norm that we want to control.

Applying the operator Π to the weight function g gives a tensor-product spline function

$$(\Pi g)(\xi) = \sum_{i=1}^{\bar{n}_1} \sum_{j=1}^{\bar{n}_2} \sum_{k=1}^{\bar{n}_3} G_{ijk} \bar{\beta}_{ij}(\xi_{12}) \bar{\beta}_k(\xi_3),$$

where the G_{ijk} form the coefficient tensor with respect to the chosen basis. We assume that the projection error satisfies

$$\|g - \Pi g\| \leq \epsilon_\Pi$$

for a given tolerance ϵ_Π. We will use the L^∞ norm if g is one of the components of K, $\boldsymbol{\ell}$ or m and the L^2 norm if $g = \hat{f}$.

The standard error bounds for spline functions can be used to analyze the asymptotic behavior of the error as the number of knots of $\bar{\mathbb{S}}$ increases. These will be used later for the theoretical analysis. In the implementation, we rely on a simple sampling-based approach in order to estimate the norm of the approximation error for any specific choice of $\bar{\mathbb{S}}$ and Π.

There is no error in some cases: The function $m = |\det(J_G)|c$, which is needed when computing the mass matrix M, is itself a spline function if c is a spline function, too. One may choose the space $\bar{\mathbb{S}}$ such that m can be represented exactly. This also applies to the components of $\boldsymbol{\ell} = |\det(J_G)|(J_G)^{-1}\boldsymbol{b}$ if the elements of b are spline functions, too.

For the stiffness matrix, every component of K is a rational function with differentiability reduced by one compared to G if the components of A are sufficiently smooth spline functions, too. We then use a high-degree tensor–product spline space for the approximation, see [18, Section 6.1], since it provides a highly accurate approximation while simultaneously requiring only very few knots.

3.2. Partial tensor decomposition

We apply the theory presented in Section 2.1 to decompose the spline function Πg in the space $\Phi \otimes \Psi = \bar{\mathbb{S}}$. Since we only split one of the three directions of the coefficient tensor instead of computing a full (canonical) decomposition, we call it *partial tensor decomposition*.

Taking the considered factorization (20) into account, the coefficients G_{ijk} of Πg form the matrix $(C_{(ij)k})$ with elements $C_{(ij)k} = G_{ijk}$, the notation (ij) again indicates that several indices are combined into a single one by a lexicographic ordering. By computing the SVD of this coefficient matrix, as described in Section 2.1, we obtain the decomposition (3) of Πg into a sum of products of bivariate and univariate functions,

$$(\Pi g)(\xi) = \sum_{r=1}^{\min(\bar{n}_1 \cdot \bar{n}_2, \bar{n}_3)} \mathcal{U}^r(\xi_{12}) \mathcal{V}^r(\xi_3),$$

with the bivariate and univariate factors

$$\mathcal{U}^r(\xi_{12}) = \sqrt{\sigma^r} \sum_{i=1}^{\bar{n}_1} \sum_{j=1}^{\bar{n}_2} u_{ij}^r \bar{\beta}_{ij}(\xi_{12}) \quad \text{and} \quad \mathcal{V}^r(\xi_3) = \sqrt{\sigma^r} \sum_{k=1}^{\bar{n}_3} v_k^r \bar{\beta}_k(\xi_3).$$

We truncate the sum to obtain an low-rank approximation of Πg. For a given rank value R we define the *partial rank-R approximation*

$$(\Lambda g)(\xi) = \sum_{r=1}^{R} \mathcal{U}^r(\xi_{12}) \mathcal{V}^r(\xi_3). \tag{21}$$

We present two error bounds for the truncation. First we consider the L^∞ norm, which is closely related to using B-splines:

Lemma 3. *The truncation error of the partial rank-R approximation satisfies*

$$||\Pi g - \Lambda g||_{\mathrm{L}^\infty} \leq \sqrt{\sum_{r=R+1}^{\min(\bar{n}_1 \cdot \bar{n}_2, \bar{n}_3)} (\sigma^r)^2}$$

if the basis functions $\{\bar{\beta}_{ijk}\}$ are tensor–product B-splines.

Proof. Recall that the tensor-product splines form a non-negative partition of unity. Consequently, the constant in Lemma 2 does not exceed 1, since

$$\|(\boldsymbol{\phi} \otimes \boldsymbol{\psi})(\xi)\|_F = \sum_{i=1}^{\bar{n}_1} \sum_{j=1}^{\bar{n}_2} \sum_{k=1}^{\bar{n}_3} (\bar{\beta}_{ijk}(\xi))^2 \leq \sum_{i=1}^{\bar{n}_1} \sum_{j=1}^{\bar{n}_2} \sum_{k=1}^{\bar{n}_3} \bar{\beta}_{ijk}(\xi) = 1.$$

□

We use the estimate in Lemma 3 to select the smallest rank R, such that the truncation error does not exceed a given tolerance ϵ_Λ. This can be done by computing the full SVD (in this case we know all singular values σ_r) or by computing a truncated SVD incrementally (in this case we exploit the fact that the singular values are decreasing). The first method was found to be sufficient for our experimental results.

Combining the effects of approximation and truncation gives the bound

$$\|g - \Lambda g\|_{L^\infty} \leq \epsilon = \epsilon_\Pi + \epsilon_\Lambda.$$

Clearly, the L^∞-norm also provides an upper bound on the L^2-norm.

Second, we obtain an optimal bound on the L^2-norm of the truncation error by considering orthonormal bases. Such bases can be obtained by applying Gram-Schmidt orthogonalization to B-splines. An alternative construction is described in [28].

Lemma 4. *If $\{\bar{\beta}_{ij}\}$ and $\{\bar{\beta}_k\}$ are L^2-orthonormal bases, then the truncated function Λg is the best approximation with respect to the L^2-norm among all functions of rank R with respect to the factorization $\bar{\mathbb{S}} = \Phi \otimes \Psi$. Moreover, the truncation error satisfies*

$$||\Pi g - \Lambda g||_{\mathrm{L}^2} = \sqrt{\sum_{r=R+1}^{\min(\bar{n}_1 \cdot \bar{n}_2, \bar{n}_3)} (\sigma^r)^2}.$$

Proof. Since we consider L^2-orthonormal bases, the Frobenius norm of the coefficient matrix of a function is equal to its L^2-norm. Thus we can prove the result by applying Lemma 1(i). □

Again, we select the smallest rank R, such that the approximation error does not exceed the given tolerance ϵ_Λ, now considering the L^2-norm. Combining the effects approximation and truncation gives the bound

$$\|g - \Lambda g\|_{L^2} \leq \epsilon = \epsilon_\Pi + \epsilon_\Lambda.$$

4. Matrix assembly

The decomposition of the integrands in (16), (17) and (18) of the components of the system matrix results in a decomposition of the system matrix itself. We represent the approximate mass, advection and stiffness matrix in the general form

$$\sum_{r=1}^{\varrho} X_r \otimes Y_r \tag{22}$$

with $X_r \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ and $Y_r \in \mathbb{R}^{n_3 \times n_3}$. We refer to the integer ϱ as the Kronecker rank of the matrix.

The low-rank approximation of the components of K makes it possible to assemble the stiffness matrix by computing bivariate and univariate integrals. Replacing K_{pq} by

$$\Lambda K_{pq} = \sum_{r=1}^{R_{pq}} \mathcal{U}_{pq}^r(\xi_{12}) \mathcal{V}_{pq}^r(\xi_3)$$

in (16), where R_{pq} is the rank value used for generating the approximation of the matrix element K_{pq}, gives the approximate stiffness matrix

$$\tilde{S}_{(ijk),(i'j'k')} =$$

$$\sum_{p,q=1}^{3} \sum_{r=1}^{R_{pq}} \underbrace{\int_{[0,1]^2} \mathcal{U}_{pq}^r \left((1-\delta_{p3}) \frac{\partial \beta_{ij}}{\partial \xi_p} + \delta_{p3} \beta_{ij} \right) \left((1-\delta_{q3}) \frac{\partial \beta_{i'j'}}{\partial \xi_q} + \delta_{q3} \beta_{i'j'} \right) \mathrm{d}\xi_{12}}_{= X^r_{pq,(ij)(i'j')}} \tag{23}$$

$$\cdot \underbrace{\int_0^1 \mathcal{V}_{pq}^r \left(\delta_{p3} \frac{\partial \beta_k}{\partial \xi_p} + (1-\delta_{p3}) \beta_k \right) \left(\delta_{q3} \frac{\partial \beta_{k'}}{\partial \xi_q} + (1-\delta_{q3}) \beta_{k'} \right) \mathrm{d}\xi_3}_{= Y^r_{pq,kk'}} . \tag{24}$$

This can be verified by using the identity

$$\frac{\partial}{\partial \xi_p} \beta_{ijk} = \left((1-\delta_{p3}) \frac{\partial \beta_{ij}}{\partial \xi_p} + \delta_{p3} \beta_{ij} \right) \left(\delta_{p3} \frac{\partial \beta_k}{\partial \xi_p} + (1-\delta_{p3}) \beta_k \right),$$

with the Kronecker delta δ_{p3}.

The approximate stiffness matrix can thus be written as a sum of $\varrho = \sum_{p,q=1}^{3} R_{pq}$ Kronecker products

$$\tilde{S} = \sum_{p,q=1}^{3} \sum_{r=1}^{R_{pq}} X_{pq}^r \otimes Y_{pq}^r$$

where the elements of the $n_1 n_2 \times n_1 n_2$ matrices X_{pq}^r and of the $n_3 \times n_3$ matrices Y_{pq}^r have been defined in (23) and (24), respectively. Consequently, we obtain an approximate representation of the stiffness matrix in the form (22) with Kronecker rank ϱ. The other two matrices can be dealt with analogously.

5. Error analysis

Next, we investigate the convergence of the discretized solution of the weak formulation using the approximate bilinear form and right hand side. Similarly to Section 2.2, we set $V = H_0^1(\hat{\Omega})$ and denote by $\hat{u} \in V$ the solution to the transformed problem (14).

Furthermore, we consider a sequence of discretizations, see Section 2.3, which are based on spline spaces $V_h = \mathbb{S}$ with decreasing diameter of the elements $h \to 0$, and denote by $u_h \in \mathbb{S}$ the solution of

$$\hat{b}_h(u_h, v_h) = \hat{\lambda}_h(v_h) \text{ for all } v_h \in V_h,$$

where

$$\hat{b}_h(u_h, v_h) = \int_{\hat{\Omega}} \hat{\nabla} u_h \cdot ((\Lambda_h K)\hat{\nabla} v_h) + ((\Lambda_h \boldsymbol{\ell}) \cdot \hat{\nabla} u_h) v_h + (\Lambda_h m) u_h v_h \, \mathrm{d}\xi \tag{25}$$

and

$$\hat{\lambda}_h(v_h) = \int_{\hat{\Omega}} \left(\Lambda_h \hat{f}\right) w_h \, \mathrm{d}\xi. \tag{26}$$

We shall see that convergence can be guaranteed only by considering an associated sequence of partial rank-R approximations with h-dependent values of the rank value R. We thus use Λ_h to denote the operator defined in (21) that transforms a weight function into its approximation.

The analysis is based on Strang's first lemma, see [29]: Since $\hat{b}_h$ is uniformly elliptic, there is a constant C such that the overall approximation error can be estimated by

$$\begin{aligned} \|\hat{u} - u_h\|_V \leq C \Bigg(\inf_{v_h \in V_h} \left(\|\hat{u} - v_h\|_V + \sup_{w_h \in V_h} \frac{|\hat{b}(v_h, w_h) - \hat{b}_h(v_h, w_h)|}{\|w_h\|_V} \right) + \\ + \sup_{w_h \in V_h} \frac{|\lambda(w_h) - \lambda_h(w_h)|}{\|w_h\|_V} \Bigg). \end{aligned} \tag{27}$$

We control the consistency error, i.e. the second and third term of the right hand side in Strang's lemma, by suitably choosing the error tolerance as described in Section 3.2. First we consider the second term, which corresponds to the approximation of the bilinear form $\hat{b}$ in (13) by the approximate bilinear form $\hat{b}_h$ in (25):

Lemma 5. *The approximate bilinear form satisfies the inequality*

$$\sup_{w_h \in V_h} \frac{|\hat{b}(v_h, w_h) - \hat{b}_h(v_h, w_h)|}{\|w_h\|_V} \leq \epsilon_h \|v_h\|_V$$

for all $v_h \in V_h$, where the parameter ϵ_h on the right hand-side is determined by the error of the low rank approximation Λ_h,

$$\epsilon_h = 3 \cdot \max\{\|K_{pq} - (\Lambda_h K)_{pq}\|_{L^\infty(\hat{\Omega})}, \ \|\boldsymbol{\ell}_p - (\Lambda_h \boldsymbol{\ell})_p\|_{L^\infty(\hat{\Omega})}, \ \|m - \Lambda_h m\|_{L^\infty(\hat{\Omega})}\}.$$

Proof. Considering the first term of $|\hat{b}(v_h, w_h) - \hat{b}_h(v_h, w_h)|$, using the Cauchy-Schwarz inequality confirms that

$$\left| \int_{\hat{\Omega}} \hat{\nabla} v_h \cdot K \hat{\nabla} w_h - \hat{\nabla} v_h \cdot (\Lambda_h K) \hat{\nabla} w_h \, \mathrm{d}\xi \right| = \left| \sum_{pq=1}^{3} \int_{\hat{\Omega}} (K_{pq} - (\Lambda_h K)_{pq}) \frac{\partial v_h}{\partial \xi_p} \frac{\partial w_h}{\partial \xi_q} \, \mathrm{d}\xi \right|$$

$$\leq \sum_{p,q=1}^{3} \|K_{pq} - \Lambda_h K_{pq}\|_{L^\infty(\hat{\Omega})} \left\| \frac{\partial v_h}{\partial \xi_p} \right\|_{L^2(\hat{\Omega})} \left\| \frac{\partial w_h}{\partial \xi_q} \right\|_{L^2(\hat{\Omega})} \leq \frac{\epsilon_h}{3} \|v_h\|_V \|w_h\|_V.$$

Since we can estimate the L^2-norm by the H^1-norm, the same bound applies to the two remaining terms. □

Second we consider the third term, which corresponds to the approximation of the linear form $\hat{\lambda}$ in (15) by the approximate linear form $\hat{\lambda}_h$ in (26):

Lemma 6. *The approximate linear form satisfies the inequality*

$$\sup_{w_h \in V_h} \frac{|\hat{\lambda}(w_h) - \hat{\lambda}_h(w_h)|}{\|w_h\|_V} \leq \epsilon'_h,$$

where the right-hand side is given by

$$\epsilon'_h = \|\hat{f} - \Lambda_h \hat{f}\|_{L^2(\hat{\Omega})}.$$

Here, $\hat{f} = |\det J_G| f \circ G$ is the right hand side of the transformed problem on the parameter domain.

Proof. By using the Cauchy-Schwarz inequality,

$$\begin{aligned} |\hat{\lambda}(w_h) - \hat{\lambda}_h(w_h)| &= \int_{\hat{\Omega}} \left(\hat{f} - \Lambda_h \hat{f} \right) w_h \, \mathrm{d}\xi \\ &\leq \|\hat{f} - \Lambda_h \hat{f}\|_{L^2(\hat{\Omega})} \|w_h\|_{L^2(\hat{\Omega})} \leq \epsilon'_h \|w_h\|_V. \end{aligned}$$

□

Thus, when approximating the right-hand side, it can be beneficial to use an L^2-orthonormal basis for the projection space since this guarantees the best approximation in the L^2-norm and thus the minimal rank value. Moreover, since the right hand side f may only be in L^2 and not in L^∞ we might not be able to measure the projection error

in the L^∞-norm. While the L^2-orthonormal basis guarantees the minimal rank value, we certainly can also use simply the B-spline basis for the low partial rank approximation, using the fact that the L^2-norm of functions on the unit cube can be bounded by the L^∞ norm.

When using a spline space $V_h\mathbb{S}$ of degree $p_1 = p_2 = p_3 = p$, the first term in (27), which represents the discretization error, can be estimated as Ch^p, where the constant C depends only on $\hat{u}$. In order to obtain an optimal rate of convergence of the overall problem, we need to make sure that the consistency error possesses the same order of convergence. This is guaranteed by choosing an appropriate sequence of operators Λ_h:

The operators Λ_h are said to be order p-convergent if there exists an h-independent constant C, such that the error bounds in Lemmas 5 and 6 satisfy $\epsilon_h \leq Ch^p$ and $\epsilon'_h \leq Ch^p$. We can now state the desired convergence result:

Theorem 7. *The solution obtained using the approximated bilinear form $\hat{b}_h$ attains the optimal rate of convergence*

$$\|\hat{u} - \hat{u}_h\|_V \leq Ch^p,$$

if the low–rank approximation operators Λ_h are order p-convergent.

The proof is obtained by combining the previous observation concerning the two sources of error (discretization and consistency error).

We conclude this section by describing a particular choice of the discretization and projection spaces that guarantees optimal rate of convergence. Let p and $\bar{p}$ denote the polynomial degrees of the three univariate factors of the tensor-product spaces $\mathbb{S}$ and $\bar{\mathbb{S}}$, respectively (for simplicity we choose uniform degrees). Similarly, we denote by n and $\bar{n}$ the numbers of degrees of freedom of these univariate factors. The total number of degrees of freedom equals n^3 and $\bar{n}^3$, respectively.

We consider an h-dependent sequence of discretization and projection spaces and the associated operators Π_h, Λ_h, where the element size h of the discretization space satisfies $h = O(1/n)$. Note that the element size $\bar{h}$ of the projection space is generally different. Nevertheless we use h to parameterize the operators.

When choosing the polynomial degree of the projection spaces as

$$\bar{p} = \mu p \tag{28}$$

for some constant integer $\mu \geq 1$, we obtain order p-convergent interpolation operators Π_h if

$$\bar{n} = O(n^{1/\mu}). \tag{29}$$

Indeed, since $\bar{h} = O(1/\bar{n})$, we may use the approximation properties of splines (see [30]) to conclude that

$$\|g - \Pi g\|_{L^\infty(\hat{\Omega})} \leq C_g \left(\frac{1}{\bar{n}}\right)^{\bar{p}+1} \leq C'_g \left(\frac{1}{n}\right)^{p} \leq C''_g h^p.$$

Furthermore, *we obtain order p-convergent low–rank approximation operators Λ_h by choosing the rank value $R = R(g, h)$ such that the truncation error has the same order of magnitude as the interpolation error.* A trivial upper bound is

$$R \leq \bar{n} = O(n^{1/\mu}), \text{ hence also } \varrho \leq O(n^{1/\mu}), \tag{30}$$

where ϱ is the total rank in (22), since the right-hand side is the maximum rank of the coefficient matrix, which has dimension $\bar{n}^2 \times \bar{n}$. We will later see that a much smaller rank can be used for most geometries. In particular, it is possible to use less degrees of freedom of the projection space if additional information on the properties of the weight function is available, see Section 3.1.

6. Computational complexity

We analyze the computational complexity of our method with respect to n, the number of degrees of freedom in each direction, and p, the polynomial degree. We compare with the complexity of the classical element-wise Gauss quadrature and the complexity of the low-rank tensor decomposition method based on HOSVD, which is described in [18]. For all methods, the computational effort is bounded from below by the number of its non-zero entries, which is $O(n^3 p^3)$.

6.1. Partial tensor decomposition method (PDM)

We briefly recall the algorithm. The input consists of the geometry map, the discretisation basis, the coefficients in (7), and a consistency tolerance. In order to perform the matrix assembly, we execute the following steps:

1. Perform the projection of the weight functions into the spline space $\bar{\mathbb{S}}$ chosen as described in Section 5. When exploiting the tensor-product structure, the complexity of this step is equal to
$$O(\bar{n}\bar{p}^2) = \mu^2 O(n^{1/\mu} p^2),$$
cf. [18].

2. Perform partial low rank approximation of the weight functions by applying partial SVD on the resulting spline function. The complexity of computing the SVD including the relevant singular vectors equals
$$O(\bar{n}^4) = O(n^{4/\mu}),$$
see [31]. this can be reduced to $O(\varrho \bar{n}^3) = O(\varrho n^{3/\mu})$ if either the total rank ϱ (or, more precisely, the rank of all weight functions) is known in advance or it is estimated based on the error bound in Lemma 3 during the computation.

3. Assemble the matrices X_r and Y_r for $r = 1, \dots, \varrho$ using bivariate (tensor-product) and univariate Gauss quadrature, respectively. This step is dominated by the bivariate Gauss quadrature[1], whose total complexity is
$$O(\varrho n^2 p^6).$$

[1] We use Gauss quadrature with $p + 1$ Gauss nodes per knot span in each coordinate direction. Strictly speaking, this corresponds to another approximation of the bilinear and linear forms. The overall accuracy of the simulation is again preserved, since the assumptions of Strang's first lemma are again satisfied.

4. Generate the full system matrix by evaluating the sum of Kronecker products (22). Since each entry is the sum of ϱ products, the complexity of this step equals

$$O(\varrho n^3 p^3).$$

As analyzed in the end of the previous section we can choose $\bar{n} = O(n^{\frac{1}{\mu}})$, where the positive integer μ determines the degree $\bar{p} = \mu p$. Therefore, in cases where the number of degrees of freedom n is much bigger than the polynomial degree p, the generation of the global matrix is dominated by the last step, that is, the computation of the sum of Kronecker products. The total complexity of PDM is then

$$O(\varrho n^3 p^3).$$

Consequently, our method is optimal if the rank ϱ does not increase with h-refinement. Even if we do not truncate the partial tensor decomposition, we get $\varrho = \bar{n} = O(n^{\frac{1}{\mu}})$ and thus the computational complexity equals $O(n^{3+\frac{1}{\mu}} p^3)$. This means that PDM is especially efficient in connection with h-refinement.

Note that there are three different split directions for the partial tensor decomposition, each resulting in a potentially different rank value for the given accuracy. Since the overall complexity is governed by the computation of the sum of Kronecker products, we choose the split direction that provides the lowest rank even if it is not the one which results in the smallest number of bivariate integrals. In some cases it is obvious which direction will result in the lowest rank, e.g. if the domain is created by a linear sweep. Otherwise, all three splits are computed as computing the SVD is a problem of low complexity relative to the algorithm.

6.2. Element-wise Gauss quadrature

For an element-wise Gauss quadrature we need to iterate over $O(n^3)$ elements and compute $O(p^6)$ values and derivatives of basis functions on each quadrature point. Since the number of quadrature points in each direction is in $O(p)$, we arrive at a total complexity of

$$O(n^3 p^9).$$

6.3. Comparison with the full tensor decomposition method (FDM)

In FDM, which is described in [18], the case of arbitrary dimensions is considered. The weight functions are projected to a spline space in the same way as described above and then decomposed into products of d univariate components using higher order SVD.

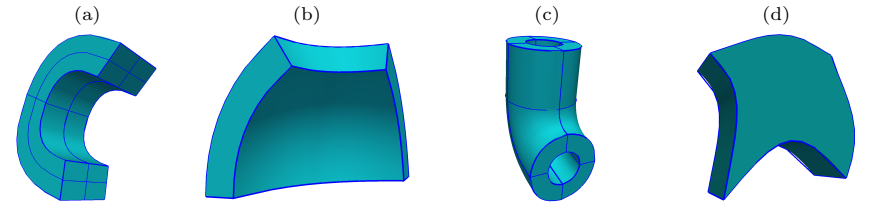
In the same way as in PDM, this decomposition is then truncated with respect to a rank value $\tilde{\varrho}$ depending on the given error tolerance ϵ. In general, the rank $\tilde{\rho}$ in the full decomposition is larger than the rank ρ obtained in the partial tensor decomposition.

In the 3-dimensional case, FDM leads to a complexity of $O(\bar{n}\bar{p}^2 + \bar{n}^4 + \tilde{\varrho} n p^3)$ for the assembly in the Kronecker format. The assembly of the full matrix from the Kronecker format is the same as in PDM and thus its complexity is in

$$O(\tilde{\varrho} n^3 p^3).$$

The quadrature in FDM is less expensive as we only have to integrate univariate integrals but the dominating step of computing the Kronecker products remains of the

Figure 1: The patches of the test suite.



same complexity and is proportional to the rank needed to fulfill the given tolerance for the consistency error.

One advantage of PDM over FDM is the optimality of the rank with respect to the Euclidean norm of the coefficient tensor as described in Section 2.1. If we find a full rank $\tilde{\varrho}$ tensor decomposition

$$\bar{\Lambda} g = \sum_{r=1}^{\tilde{\varrho}} \mathcal{U}_r \otimes \mathcal{V}_r \otimes \mathcal{W}_r = \sum_{r=1}^{\tilde{\varrho}} (\mathcal{U}_r \otimes \mathcal{V}_r) \otimes \mathcal{W}_r,$$

we also have a rank $\tilde{\varrho}$ partial tensor decomposition by combining the first two vectors into a matrix. Since the rank value obtained by using partial SVD is optimal, it follows that

$$\varrho \leq \tilde{\varrho},$$

where ϱ is the rank needed to satisfy the same error tolerance in the partial decomposition.

Summing up, we conclude that – for $n \gg p$ – the complexity of PDM is at most of the same order as the one of FDM and has the potential to be much smaller. Additionally, the total rank ϱ in the partial tensor decomposition is bounded by $O(\bar{n})$, while the rank $\tilde{\varrho}$ that appears in the full decomposition is bounded by $O(\bar{n}^2)$.

7. Experiments and numerical results

We present results of numerical experiments, in order to demonstrate the behavior of PDM. We choose a test suite of different patches that possess different rank values for the Jacobian of the geometry mapping as well as the matrix-valued factor in the stiffness matrix. Figure 1 visualizes the patches that we used to test our method, as well as the degrees of freedom and polynomial degrees of the geometry mapping[2].

We will compare the rank values and the computing times with those of FDM. As an additional benchmark, we use the computing times of a classical element-wise Gauss quadrature. All numerical experiments were performed on a standard PC using the G+Smo C++ library [32, 33, 34].

[2]The parametrizations used in this section can be found in https://github.com/gismo/data/.

7.1. Rank comparison

Consider the operator (7) with $A = I$, $b = 0$ and $c = 1$. We analyze the ranks of the weight functions defined by the Jacobian determinant m and the matrix K, see (10) and (12). In particular, we will study the stability of the rank as the prescribed truncation error ϵ_Λ tends to zero.

The experiments were done using the B-spline basis of the spline space $\bar{\mathbb{S}}$ which is used for interpolating the weight functions. Consequently, as discussed in Section 3.2, we measure the approximation error in the L^∞-norm using Lemma 3.

Table 1 shows the rank values that are needed to approximate the Jacobian determinant for varying accuracy ϵ_Λ, using the three different split directions. No projection error is present since the Jacobian determinant can be represented exactly in a tensor–product spline space $\bar{\mathbb{S}}$. In addition we provide information about the full tensor rank.

Table 1: Ranks needed to approximate the Jacobian determinant m with varying accuracy. The bold font indicates the lowest ranks among the different split directions.

(a)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	**1**	**1**	**1**
rank for split direction 2	**1**	**1**	**1**
rank for split direction 3	**1**	**1**	**1**
tensor rank	**1**	**1**	**1**

(b)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	1	2	3
rank for split direction 2	1	3	3
rank for split direction 3	**1**	**2**	**2**
tensor rank	1	3	4

(c)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	2	2	3
rank for split direction 2	**2**	**2**	**2**
rank for split direction 3	2	2	3
tensor rank	4	4	5

(d)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	6	7	7
rank for split direction 2	6	7	7
rank for split direction 3	**1**	**1**	**1**
tensor rank	6	7	7

As analyzed in Section 6.3, the rank obtained by the partial decomposition in all directions never exceeds the tensor rank. The Jacobian determinant of model (a) has tensor rank 1 which is recognized by both methods. For the volumes (b) and (d), which were generated by approximate offsetting, the rank obtained by decomposing the parametrization with respect to the offset direction is the lowest one. This effect is particularly strong for (d).

We now proceed to the ranks of the elements of the matrix K, see (10). This matrix does not admit an exact representation in the spline space $\bar{\mathbb{S}}$ (except for trivial geometries). To minimize the influence of the spline approximation error, we choose the interpolation spaces $\bar{\mathbb{S}}$ such that the L^∞-norm of the interpolation error does not exceed $\epsilon_\Pi = 10^{-10}$.

Table 2 shows the *total rank* for different prescribed tolerances in the low-rank approximation, i.e. the sum of the ranks of the components of K. Again, we compare the rank values of the full decomposition with the partial decomposition in all three directions.

Since we consider the total rank, the difference in the rank values between the full and partial decompositions is more prominent in the case of the stiffness matrix. Similar

Table 2: The total ranks needed to approximate the matrix-valued function K with varying accuracy. The bold font indicates the lowest ranks among the different split directions.

(a)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	**5**	**5**	**5**
rank for split direction 2	**5**	**5**	**5**
rank for split direction 3	**5**	**5**	**5**
tensor rank	**5**	**5**	**5**

(b)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	13	22	31
rank for split direction 2	13	28	38
rank for split direction 3	**9**	**16**	**16**
tensor rank	13	29	42

(c)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	27	31	35
rank for split direction 2	18	22	24
rank for split direction 3	**18**	**18**	**19**
tensor rank	38	44	49

(d)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	48	85	101
rank for split direction 2	48	85	101
rank for split direction 3	**5**	**5**	**8**
tensor rank	48	85	101

to the case of the mass matrix, we observe that the splitting with respect to the direction of the sweep gives the lowest values of the total rank for volumes (b) and (d).

For the mass matrix, both methods result in quite stable rank values. For the stiffness matrix, however, the rank may increase as the error tolerance goes to zero. However, while the worst-case upper bound for the tensor rank is $\bar{n}^2$, it is only $\bar{n}$ for the rank generated by the partial decomposition.

In the following sections, when comparing PDM with FDM and with the element-wise Gauss quadrature, we always choose the split direction which provides the smallest rank value. This is motivated by the observation that the overall complexity is dominated by this rank.

7.2. Decomposition step

We investigate the computational costs of the decomposition step (HOSVD or SVD), in relation to the total cost of the matrix assembly. These costs are reported in Table 3 for the mass and the stiffness matrix.

Table 3: Efficiency of the decomposition step when assembling the mass matrix (top) and the stiffness matrix (bottom) for the coarsest discretization.

mass matrix

Model	Disc. DOF	Intpl. DOF	HOSVD time(s)	Assembly time(s)	Ratio	SVD time(s)	Assembly time(s)	Ratio
(a)	$4 \times 4 \times 4$	$11 \times 11 \times 11$	$1.6 \cdot 10^{-3}$	$8.1 \cdot 10^{-3}$	20%	$2.7 \cdot 10^{-4}$	$8.0 \cdot 10^{-3}$	3%
(b)	$3 \times 3 \times 2$	$6 \times 6 \times 3$	$6.3 \cdot 10^{-4}$	$2.1 \cdot 10^{-3}$	30%	$7.5 \cdot 10^{-5}$	$1.5 \cdot 10^{-3}$	5%
(c)	$9 \times 2 \times 5$	$24 \times 3 \times 12$	$3.2 \cdot 10^{-3}$	$1.3 \cdot 10^{-2}$	25%	$6.1 \cdot 10^{-4}$	$1.0 \cdot 10^{-2}$	6%
(d)	$5 \times 5 \times 2$	$12 \times 12 \times 3$	$1.0 \cdot 10^{-3}$	$7.4 \cdot 10^{-3}$	14%	$9.5 \cdot 10^{-5}$	$4.8 \cdot 10^{-3}$	2%

stiffness matrix

Model	Disc. DOF	Intpl. DOF	HOSVD time(s)	Assembly time(s)	Ratio	SVD time(s)	Assembly time(s)	Ratio
(a)	$4 \times 4 \times 4$	$31 \times 31 \times 31$	$1.2 \cdot 10^{-1}$	$3.7 \cdot 10^{-1}$	32%	$5.5 \cdot 10^{-2}$	$2.8 \cdot 10^{-1}$	20%
(b)	$3 \times 3 \times 2$	$13 \times 13 \times 9$	$8.2 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	37%	$2.0 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$	13%
(c)	$9 \times 2 \times 5$	$64 \times 12 \times 32$	$1.6 \cdot 10^{-1}$	$2.9 \cdot 10^{-1}$	55%	$7.4 \cdot 10^{-2}$	$3.8 \cdot 10^{-1}$	19%
(d)	$5 \times 5 \times 2$	$32 \times 32 \times 20$	$1.3 \cdot 10^{-1}$	$4.0 \cdot 10^{-1}$	33%	$4.1 \cdot 10^{-2}$	$3.1 \cdot 10^{-1}$	13%

We choose the projection spaces $\bar{\mathbb{S}}$ for the mass matrix and the stiffness matrix such that the interpolation error accuracy is below 10^{-8}. The resulting numbers of degrees of freedom are reported in the third column of both tables. The same accuracy is used when performing the rank truncation.

We then compare the decomposition time with the total time needed for the matrix assembly, for the coarsest possible approximation as determined by the geometry mapping (reported in the second column of the tables). Consequently, the number $\bar{n}$ is always larger than n in this experiment. Even in this situation, the total cost is dominated by the assembly step.

More precisely, the tables report the decomposition time needed for both methods (columns 4 and 7), the associated assembly times (columns 5 and 8), and their ratio. For the partial decomposition, the decomposition time never contributes more than 20% to the total time, even for these very sparse discretizations. In addition we observe that the SVD is always significantly faster than HOSVD.

Even for this experiment, where $\bar{n} \geq n$, the decomposition step only takes up a relatively small part of the total computation time. As we will see in the next section, the assembly time increases linearly with the number of degrees of freedom of the discretisation space, while the refinement of $\mathbb{S}$ has no effect on the decomposition. Indeed, the latter step only depends on the dimension of the projection space $\bar{\mathbb{S}}$. Thus, the contribution of the decomposition step becomes even less significant as n is increased.

7.3. Order of convergence

The consistency error in Strang's first lemma – and consequently the overall error of the isogeometric discretization – can be controlled by the combined error of spline projection and rank truncation, $\epsilon = \epsilon_\Pi + \epsilon_\Lambda$. Fig. 2 reports the convergence rates for the solution of the Poisson problem

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega, \\ u &= u_0 && \text{on } \partial\Omega, \end{aligned}$$

with known exact solution

$$u(x, y, z) = \sin(\pi x) \sin(\pi y) \sin(\pi z).$$

on the domain Ω represented by patch (b) in Fig. 1, using PDM. We consider discretizations of degree $p = 2, 3, 4$ and an error tolerance $\epsilon = 10^{-10}$, which results in rank-16 approximations of the stiffness matrices. These realize the optimal rates of convergence for the considered range of n, both with respect to the L^2 norm (left) and H^1 seminorm (right).

7.4. Computational complexity of matrix assembly

We investigate the dependence of the computation times on n (the number of degrees of freedom per direction used for the discretization) and p (the corresponding polynomial degrees). The experiments were performed on the computational domain represented by patch (c), using different polynomial degrees for the discretization. The overall accuracy was chosen to be $\epsilon = 10^{-8}$, and the degrees of the projection spline space $\bar{\mathbb{S}}$ were kept constant, $(5, 2, 5)$ in the mass case and $(11, 7, 11)$ in the stiffness case.

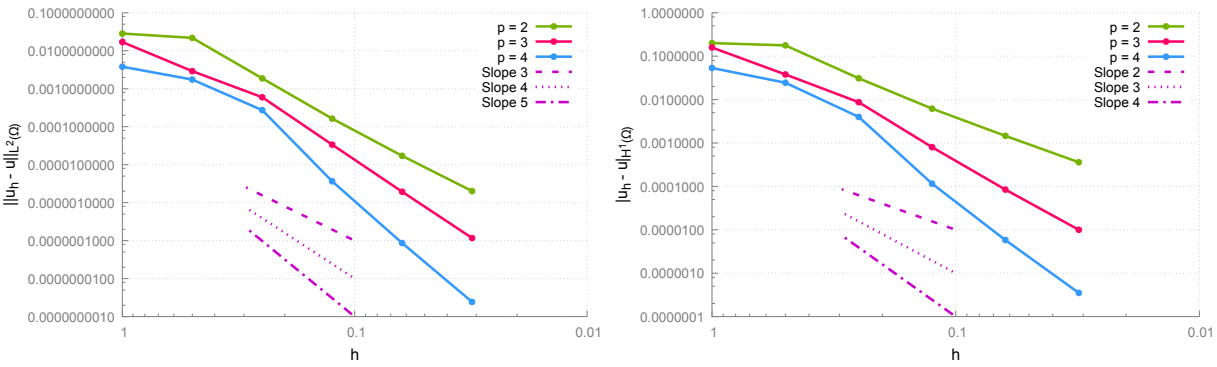


Figure 2: Numerical errors for the solution to a Poisson problem on model (b) for degrees $p = 2, 3, 4$..

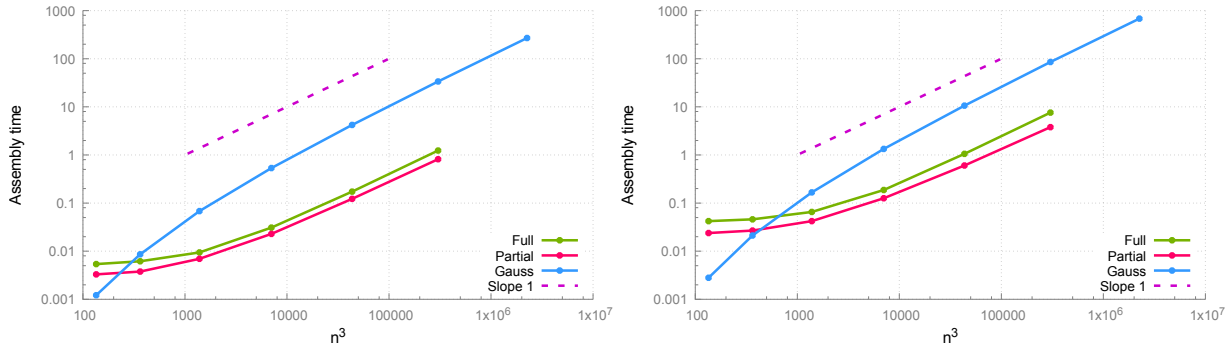
First we explore the dependence on the number n^3 of degrees of freedom. Figure 3 reports the computation times (including interpolation, decomposition, numerical quadrature and sum of Kronecker products) required for assembling the mass and stiffness matrices for various values of n and degrees $p = 2, 3, 4$.

All three methods scale linearly with the dimension n^3 of the discretisation space $\mathbb{S}$. Both decomposition-based methods are much faster than the element-wise Gauss quadrature, even for small polynomial degrees p. PDM is faster than FDM for sufficiently large values of n, since the rank is smaller. For small values of n and larger degrees p, however, the overall effort of PDM is dominated by the bivariate quadrature, and hence FDM performs slightly better.

Now we continue with the dependence on the polynomial degree p. Figure 4 reports the computation times required for assembling the mass and stiffness matrices for three different values of n and degrees $p = 1, \ldots, 8$.

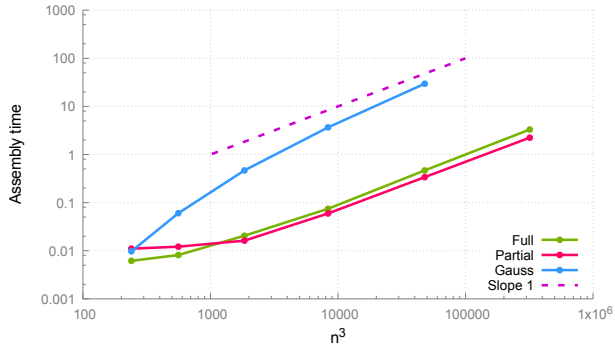
These plots confirm the expected asymptotic scaling with p^3 for both of the decomposition-based methods, while the computational costs of the element-wise Gauss quadrature grow much faster. We also note that the complexity (of order 6 with respect to the degree) of the bivariate quadrature dominates the overall effort for large values of p, hence FDM becomes slightly faster than PDM. In fact, if n has lower order of magnitude than p^3, the overall complexity is dominated by the bivariate quadrature and FDM is potentially more efficient. Consequently, PDM and FDM are better suited for h- and p-refinement respectively.

Finally we address the Kronecker format. For certain applications (for instance, when using iterative solvers that require only matrix-vector multiplications), it suffices to represent the matrices in Kronecker format (22). This can be achieved simply by omitting the last step of the algorithm. The computation time is then dominated by the Gauss quadrature needed when evaluating the bivariate and univariate integrals. Consequently, PDM and FDM have complexity $O(\rho n^2 p^6)$ and $O(\tilde{\rho} n p^3)$, respectively. This is confirmed by the numerical experiments reported in Fig. 5. However, the cost of matrix-vector multiplications using the Kronecker format grows linearly with the rank of the representation, and hence PDM (which typically generates a much lower rank value) has an advantage when considering the overall computational effort.

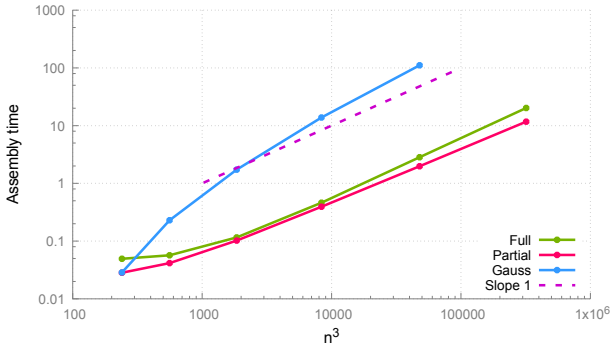


(a) Mass, $p = 2$

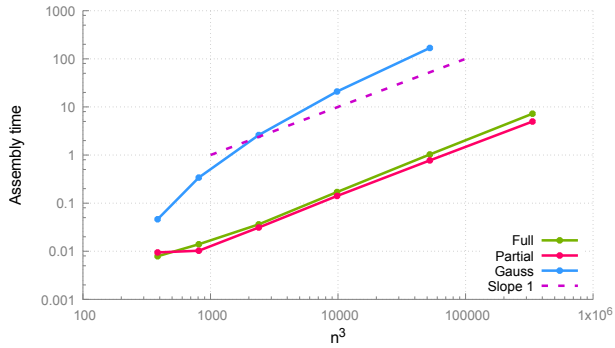
(b) Stiffness, $p = 2$



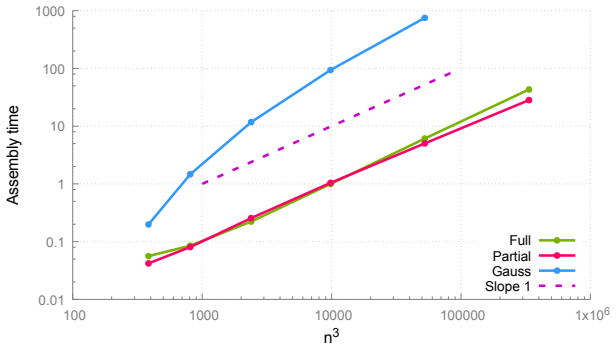
(c) Mass, $p = 3$



(d) Stiffness, $p = 3$

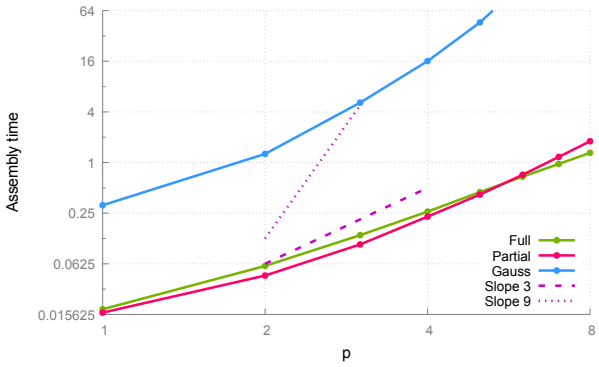


(e) Mass, $p = 4$

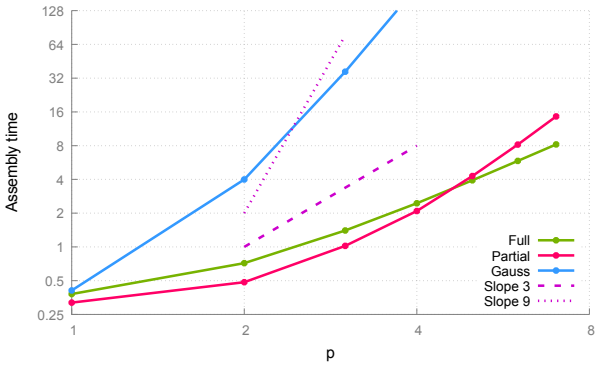


(f) Stiffness, $p = 4$

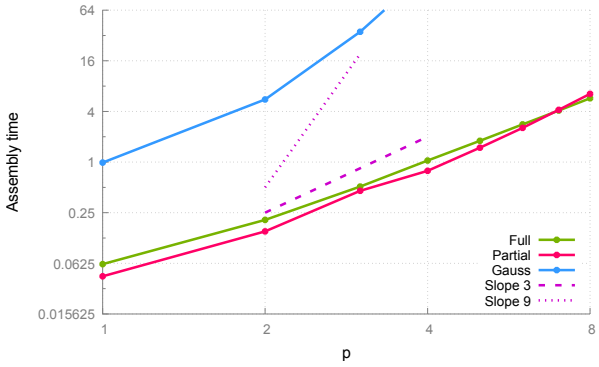
Figure 3: n-dependence of the assembly times of the mass and stiffness matrices on model (c) for different polynomial degrees p.



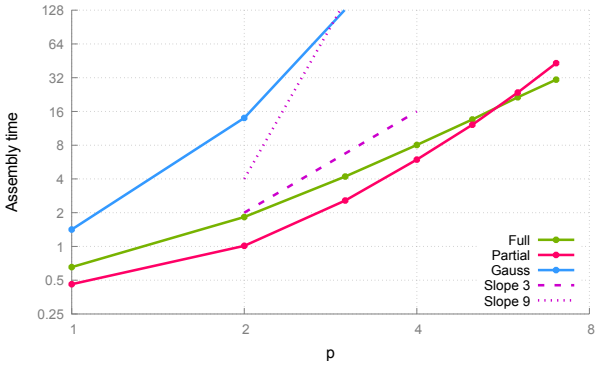
(a) Mass, $25 \times 25 \times 25$



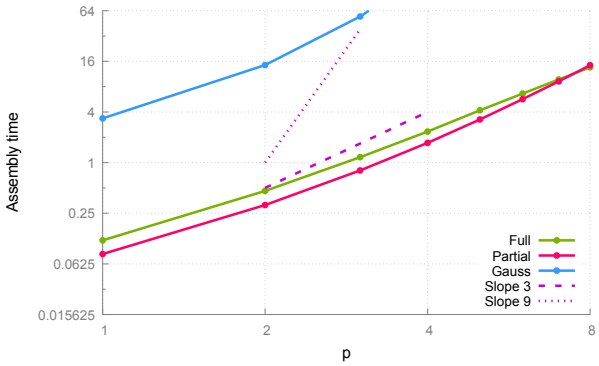
(b) Stiffness, $25 \times 25 \times 25$



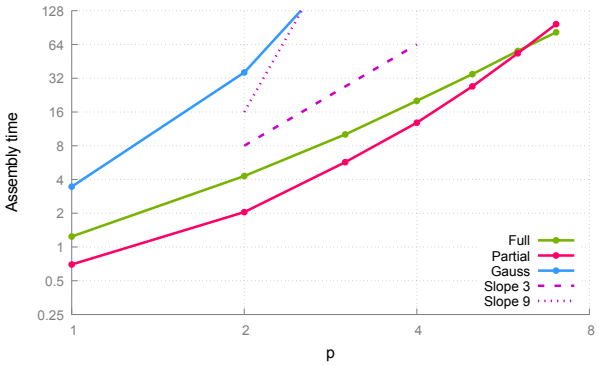
(c) Mass, $37 \times 37 \times 37$



(d) Stiffness, $37 \times 37 \times 37$

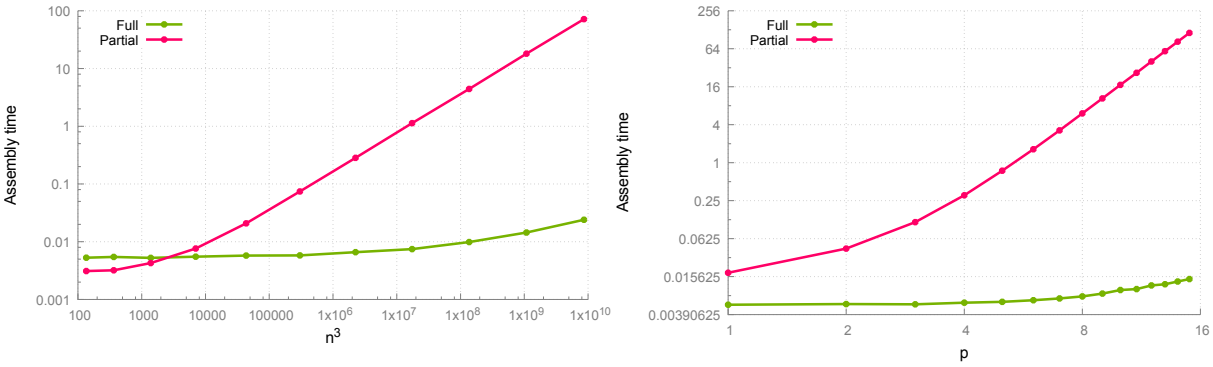


(e) Mass, $50 \times 50 \times 50$



(f) Stiffness, $50 \times 50 \times 50$

Figure 4: p-dependence of the assembly times of the mass and stiffness matrices on model (c) for different values of n.



(a) n-dependence for $p = 2$

(b) p-dependence for $50 \times 50 \times 50$ DOF

Figure 5: n- and p-dependence of the computation time for the Kronecker format of the mass matrix on model (c).

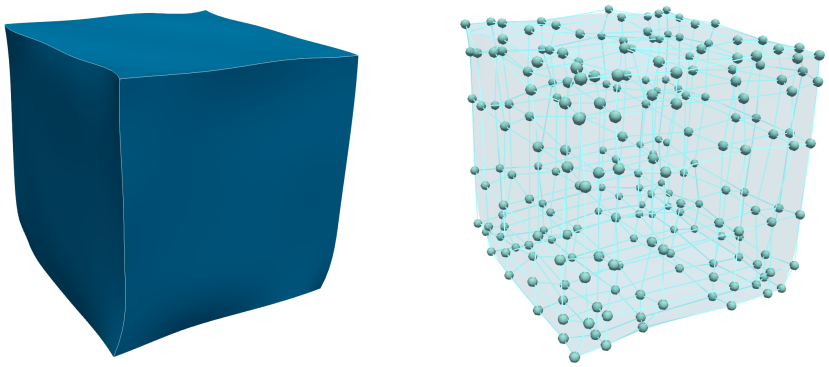


Figure 6: The randomly perturbed cube (left) and its control grid (right), $n = 6$ and $p = 3$.

7.5. Worst-case comparison with FDM

In Section 6.3 we concluded that the rank obtained by PDM is bounded by $O(\bar{n})$ and is always lower than the rank obtained by FDM which is bounded by $O(\bar{n}^2)$. This means that even for very irregular patches PDM will still have an advantage over the classical element-wise Gauss quadrature while FDM might perform poorly, especially since the cost of the nonlinear optimization in the decomposition step increases with the rank.

In order to demonstrate this advantage experimentally, we perform another experiment on a patch which is expected to exhibit maximal rank for both methods. We use a cubic parametrization of a cube with two inner knots in each direction and disturb its control net using pseudo-random numbers. The displacement of each control point was bounded to maintain the regularity of the parametrization. Figure 6 shows the resulting perturbed cube. First we analyze the behavior of the ranks in both methods depending on the chosen tolerance ϵ_Λ. Figure 7 shows the resulting rank when approximating the Jacobian determinant for decreasing tolerance in both FDM and PDM. In this experiment we chose the spline space $\bar{\mathbb{S}}$ such that the Jacobian determinant is represented

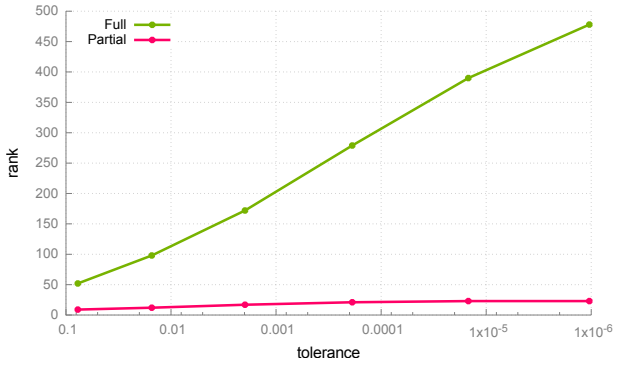


Figure 7: Ranks of the Jacobian determinant of the perturbed cube

exactly, result in an interpolation space of dimensions $23 \times 23 \times 23$. This means that the maximum rank in PDM is 23 while the maximum rank in FDM is 529. The results comply with these bounds.

Second we compare the computation times for evaluating the mass matrix using FDM, PDM, and the classical element-wise Gauss quadrature on the perturbed cube. We choose the tolerance depending on the number of degrees of freedom of the discretization space such that it decays as fast as the theoretical discretization error of the L^2-projection, thereby maintaining the overall accuracy of the numerical simulation. Table 4 shows the resulting total computation times as well as the time needed for the (full or partial) tensor decomposition. We observe that the decomposition time in PDM is constant while the

Table 4: Computation times for assembling the mass matrix on the perturbed cube using FDM, PDM and Gauss quadrature.

		rank		decomposition time (s)		total time (s)		
#DOF	tolerance	FDM	PDM	FDM	PDM	FDM	PDM	Gauss
216	$7.7 \cdot 10^{-2}$	52	9	0.16	0.0017	0.32	0.16	**0.02**
729	$1.5 \cdot 10^{-2}$	98	12	1.84	0.0017	2.09	0.18	**0.13**
3375	$1.9 \cdot 10^{-3}$	172	17	27.36	0.0017	28.85	**0.37**	1.05
19683	$1.9 \cdot 10^{-4}$	279	21	250.67	0.0017	267.88	**1.56**	8.39
132651	$1.5 \cdot 10^{-5}$	390	23	896.87	0.0017	1083.72	**9.02**	67.75
970299	$1.0 \cdot 10^{-6}$	478	23	2288.38	0.0017	4526.56	**58.35**	566.81

decomposition time in FDM grows significantly as the rank increases. The comparison with the classical element-wise Gauss quadrature shows that PDM still achieves a large speed-up even in this worst-case example. On the other hand, FDM performs worse than Gauss quadrature in this case. This is not in contradiction to the complexity result presented in Section 6; a higher degree p would be required to realize the asymptotic advantage of FDM with respect to Gauss quadrature.

8. Conclusion

We introduced partial tensor decomposition method (PDM) as a new approach to matrix assembly in isogeometric analysis. As a major advantage, the implementation

of PDM is considerably simpler than the one of the full decomposition method (FDM) introduced in [18]. This is due to the fact that PDM relies on standard singular value decomposition (SVD) and does not require HOSVD/ALS or similar techniques. After discussing theoretical aspects (convergence and computational complexity), we presented several numerical experiments to test the performance of the new method. The new method provides a further reduction of the computational costs when considering standard matrix representations and h-refinement. Additionally, we observed that PDM is much more robust than FDM in cases where the domain is ill-suited for tensor decomposition methods.

The exposition in the present paper was restricted to three-dimensional domains. Current work is devoted to the extension to higher dimensions, such as time-dependent problems on four-dimensional space-time domains [35].

Acknowledgment

The authors gratefully acknowledge the support provided by the Austrian Science Fund (FWF) through project NFN S11708 and by the European Research Council (ERC), project GA 694515.

References

[1] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, Chichester, England, 2009.

[2] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, V. Calo, The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 353 – 361.

[3] D. Garcia, M. Bartoň, D. Pardo, Optimally refined isogeometric analysis, Procedia Computer Science 108 (2017) 808 – 817.

[4] M. Woźniak, M. Paszyński, D. Pardo, L. Dalcin, V. M. Calo, Computational cost of isogeometric multi-frontal solvers on parallel distributed memory machines, Computer Methods in Applied Mechanics and Engineering 284 (2015) 971 – 987.

[5] F. Auricchio, F. Calabrò, T. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 249-252 (2012) 15–27.

[6] M. Bartoň, 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 (2016) 217 – 240.

[7] M. Bartoň, V. M. Calo, Gauss–galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis, Computer-Aided Design 82 (2017) 57 – 67.

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

[9] C. Adam, T. Hughes, S. Bouabdallah, M. Zarroug, H. Maitournam, Selective and reduced numerical integrations for NURBS-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 284 (2015) 732–761.

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

[11] M. Hillman, J. Chen, Y. Bazilevs, Variationally consistent domain integration for isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 284 (2015) 521–540.

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

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

[14] H. Gomez, L. D. Lorenzis, The variational collocation method, Computer Methods in Applied Mechanics and Engineering 309 (2016) 152 – 181.
[15] A. Mantzaflaris, B. Jüttler, Exploring matrix generation strategies in isogeometric analysis, in: M. Floater, et al. (Eds.), Mathematical Methods for Curves and Surfaces, Vol. 8177 of Lecture Notes in Computer Science, Springer, 2014, pp. 364–382.
[16] A. Mantzaflaris, B. Jüttler, Integration by interpolation and look-up for Galerkin-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 284 (2015) 373 – 400.
[17] A. Mantzaflaris, B. Jüttler, B. Khoromskij, U. Langer, Matrix generation in isogeometric analysis by low rank tensor approximation, in: J.-D. Boissonnat, et al. (Eds.), Curves and Surfaces, Vol. 9213 of LNCS, Springer, 2015, pp. 321–340.
[18] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, U. Langer, Low rank tensor methods in Galerkin-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1062 – 1085.
[19] F. Calabrò, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, Computer Methods in Applied Mechanics and Engineering 316 (2017) 606 – 622.
[20] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, G. Sangalli, Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization, Computer Methods in Applied Mechanics and Engineering 285 (2015) 817–828.
[21] G. Sangalli, M. Tani, Isogeometric preconditioners based on fast solvers for the Sylvester equation, SIAM Journal on Scientific Computing 38 (6) (2016) A3644–A3671.
[22] M. Bachmayr, W. Dahmen, Adaptive low-rank methods: Problems on sobolev spaces, SIAM Journal on Numerical Analysis 54 (2) (2016) 744–796.
[23] E. Kieri, C. Lubich, H. Walach, Discretized dynamical low-rank approximation in the presence of small singular values, SIAM Journal on Numerical Analysis 54 (2) (2016) 1020–1038.
[24] C. Lubich, I. Oseledets, B. Vandereycken, Time integration of tensor trains, SIAM Journal on Numerical Analysis 53 (2) (2015) 917–941.
[25] W. Hackbusch, Tensor spaces and numerical tensor calculus, Springer, Berlin, 2012.
[26] V. Khoromskaia, B. N. Khoromskij, Tensor numerical methods in quantum chemistry: From Hartree-Fock to excitation energies, Physical Chemistry Chemical Physics 17 (2015) 31491–31509.
[27] M. Griebel, H. Harbrecht, Approximation of bi-variate functions: singular value decomposition versus sparse grids, IMA Journal of Numerical Analysis 34 (1) (2014) 28–54.
[28] Y. Wei, G. Wang, P. Yang, Legendre-like orthogonal basis for spline space, Computer-Aided Design 45 (2) (2013) 85 – 92, solid and Physical Modeling 2012.
[29] G. Strang, Approximation in the finite element method, Numerische Mathematik 19 (1972) 81–98.
[30] L. Schumaker, Spline Functions: Basic Theory, 3rd Edition, Cambridge University Press, 2007.
[31] G. Golub, C. Van Loan, Matrix Computations, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, 2013.
[32] B. Jüttler, U. Langer, A. Mantzaflaris, S. E. Moore, W. Zulehner, Geometry + Simulation Modules: Implementing Isogeometric Analysis, PAMM 14 (1) (2014) 961–962.
[33] U. Langer, A. Mantzaflaris, S. Moore, I. Toulopoulos, Multipatch discontinuous Galerkin isogeometric analysis, in: Isogeometric Analysis and Applications, Lecture Notes in Computational Science and Engineering, Springer International Publishing, 2015, pp. 1–32.
[34] A. Mantzaflaris, F. Scholz, others (see website), G+smo (Geometry plus Simulation modules) v0.8.1, http://gs.jku.at/gismo (2017).
[35] A. Mantzaflaris, F. Scholz, I. Toulopoulos, Low-rank space-time decoupled isogeometric analysis for parabolic problems with varying coefficients, Computational Methods in Applied Mathematics (provisionally accepted).