SIAM J. Sci. Comput.
Vol. 42, No. 1, pp. A140–A161

A LOW-RANK TENSOR METHOD FOR PDE-CONSTRAINED OPTIMIZATION WITH ISOGEOMETRIC ANALYSIS[*]

ALEXANDRA BÜNGER[†], SERGEY DOLGOV[‡], AND MARTIN STOLL[†]

Abstract. Isogeometric analysis (IgA) has become one of the most popular methods for the discretization of PDEs, motivated by the use of nonuniform rational B-splines for geometric representations in industry and science. A crucial challenge lies in the solution of the discretized equations, which we discuss in this paper with a particular focus on PDE-constrained optimization discretized using IgA. The discretization results in a system of large mass and stiffness matrices, which are typically very costly to assemble. To reduce the computation time and storage requirements, low-rank tensor methods as proposed in [A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer, *Comput. Methods Appl. Mech. Engrg.*, 316 (2017), pp. 1062–1085] have become a promising approach. We present a framework for the assembly of these matrices in low-rank form using tensor train approximations. Furthermore, our framework allows for the exploitation of the resulting low-rank structure of the mass and stiffness matrices, and it can be used to solve a PDE-constrained optimization problem without assembling the actual system matrices and carries the low-rank format over to the solution. We use the block alternating minimal energy method to efficiently solve the corresponding KKT system of the optimization problem. We show several numerical experiments with three-dimensional geometries to demonstrate that the low-rank assembly and solution drastically reduce the memory demands and computing times, depending on the approximation ranks of the domain.

Key words. isogeometric analysis, optimal control, low-rank decompositions, tensor train format

AMS subject classifications. 65F10, 65F50, 15A69, 93C20

DOI. 10.1137/18M1227238

1. Motivation.

Isogeometric analysis (IgA) is a relatively new discretization technique to give an approximate solution to a problem posed by a partial differential equation (PDE) on a given domain Ω. It was introduced by Hughes, Cottrell, and Bazilevs in 2005 [19].

In IgA the physical domain Ω and the solution space for solving a PDE via the Galerkin method [36] are parameterized by the same spline functions, typically B-splines or nonuniform rational B-splines (NURBS). These basis functions are globally defined and may have large overlapping supports depending on their degrees. This leads to a global representation of the physical domain. Hence, the PDE discretization has a high computational complexity increasing exponentially with respect to the dimension of the problem [24].

Recently, a lot of effort has been made to find strategies to overcome this drawback and efficiently assemble the arising system matrices. Here, a big focus lies on exploiting the tensor product structure of the basis functions and lowering the overall computational cost of the basis function quadrature, e.g., via sum factorization [1] or

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section November 16, 2018; accepted for publication (in revised form) October 2, 2019; published electronically January 8, 2020.
https://doi.org/10.1137/18M1227238

Funding: The work of the first and third authors was partially supported by the German Science Foundation (DFG) through grant 1742243256 - TRR 96 and DAAD-MIUR Joint Mobility Program 2018-2020 through grant 5739665.

[†]Technische Universität Chemnitz, Department of Mathematics, Chair of Scientific Computing, 09107 Chemnitz, Germany (alexandra.buenger@mathematik.tu-chemnitz.de, martin.stoll@mathematik.tu-chemnitz.de).
[‡]University of Bath, Claverton Down, BA2 7AY, Bath, United Kingdom (s.dolgov@bath.ac.uk).

finding new quadrature rules [15, 20].

To reduce the complexity of the integration and ultimately the overall computation time and storage requirements, Mantzaflaris et al. [23, 22] developed a low-rank tensor method, which exploits the tensor structure of the basis functions and separates the variables of the integrals. The arising system matrices can then be represented in a compact manner as a sum of Kronecker products of smaller matrices which are assembled via univariate integration, lifting the curse of dimensionality from the integration. This is accomplished via an interpolation step and a low-rank representation of the resulting coefficient tensor.

For two-dimensional settings, the low-rank approximation can be easily realized by a singular value decomposition of the coefficient matrix. In higher dimensions, however, we need to decompose a high-order tensor which is more challenging, both computationally and conceptually, as there exist many different decompositions and definitions of ranks in the dimensions greater than two.

In this paper we combine the low-rank method of Mantzaflaris et al. with low-rank tensor train (TT) calculations [25, 27]. Exploiting the tensor product nature of the arising interpolation, we can calculate a low-rank TT approximation without prior assembly of the full coefficient tensor by means of the alternating minimal energy (AMEn) method [10]. We further utilize this method to ultimately solve a large-scale optimal control problem in a compact low-rank block format, exploiting the Kronecker product structure of the system.

We consider an optimal control problem with a parabolic PDE constraint of the form

$$\min_{y,u} \quad \frac{1}{2}\int_0^T \int_\Omega (y-\hat{y})^2 \,\mathrm{d}x\,\mathrm{d}t + \frac{\alpha}{2}\int_0^T \int_\Omega u^2 \,\mathrm{d}x\,\mathrm{d}t \tag{1.1}$$

$$\text{such that (s.t.)} \qquad y_t - \Delta y = u \qquad \text{in } (0,T)\times\Omega, \tag{1.2}$$

$$y = 0 \qquad \text{on } (0,T)\times\partial\Omega, \tag{1.3}$$

$$y = y_0 \qquad \text{on } \Omega \text{ for } t = 0, \tag{1.4}$$

with a desired state $\hat{y}$ and control u on a given domain Ω parameterized by B-splines or NURBS, as described later on. The discretization of (1.1)–(1.4) in this paper will be performed by IgA, and the workhorse is the representation of two bilinear forms, the mass term a_m and the stiffness term a_s in a discretized low-rank format. The goal of this work is to illustrate that the use of NURBS in the finite element context is a particularly convenient choice, both from the perspective of being able to discretize complex domains as well as leading to nicely structured linear systems. Alternatively, one can rely on the discretization of the optimization problem using other finite element spaces, and we refer to [17, 37] for general introductions and a coherent list of references.

We will briefly review the basic ingredients for IgA in section 2 to clarify the terminology and notations used throughout the paper. In section 3 we review the previously mentioned low-rank approach of Mantzaflaris et al. and present a way to exploit the tensor product structure to quickly find a low-rank approximation using the TT format. We then show how an optimal control problem of the form (1.1)–(1.4) is discretized using IgA and state the resulting discrete saddle point problem in section 4. In section 5 we exploit the derived low-rank representation from section 3 to solve the resulting large-scale linear system in a compact format, making use of the iterative block AMEn method [2, 8].

The performance of the low-rank scheme is illustrated by various examples in section 6. First, we show the performance for approximating both mass and stiffness matrices in the low-rank format for domains with different ranks. We then use these approximations to solve computationally challenging PDE-constrained optimization problems in a low-rank TT format.

2. Basics for IgA.

In IgA, a geometry is represented exactly using a set of B-splines or NURBS functions. The same basis functions are then used to build the solution space to solve a PDE on the geometric domain [19]. The term *B-spline* is short for basis spline and denotes a special type of recursively defined splines. Every spline function with a chosen degree, smoothness, and domain partition can be uniquely represented as a linear combination of B-splines with the same degree, smoothness, and domain partition [5].

A set of n B-splines is uniquely defined by its degree and knot vector. Choosing a degree $p \in \mathbb{N}_0$, we define a vector $\xi \in [0,1]^{(n+p+1)}$ called the open knot vector as $\xi = [\xi_1, \dots \xi_{n+p+1}]^\top$ with

$$0 = \xi_1 = \cdots = \xi_{p+1} < \xi_{p+2} \leq \cdots \leq \xi_n < \xi_{n+1} = \cdots = \xi_{n+p+1} = 1, \tag{2.1}$$

where the end knots appear $p+1$ times and for all other knots, duplicate appearances are allowed up to multiplicity p. The parameter $n \in \mathbb{N}$ determines the number of resulting B-splines $\beta_{i,p}$ with $i = 1, \dots, n$.

For each knot vector ξ as in (2.1), the according B-splines $\beta_{i,p}$ of degree p, with $i = 1, \dots, n$, are uniquely defined by the recursion

$$\beta_{i,0}(\hat{x}) = \begin{cases} 1 & \text{if } \xi_i \leq \hat{x} < \xi_{i+1}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.2}$$

$$\beta_{i,j}(\hat{x}) = \frac{\hat{x} - \xi_i}{\xi_{i+j} - \xi_i} \beta_{i,j-1}(\hat{x}) + \frac{\xi_{i+j+1} - \hat{x}}{\xi_{i+j+1} - \xi_{i+1}} \beta_{i+1,j-1}(\hat{x}), \tag{2.3}$$

where $j = 1, 2, \dots, p$ and $i = 1, \dots, n$. Each resulting B-spline $\beta_{i,p}$ has the local support $[\xi_i, \xi_{i+p+1}]$.

We use $\mathbb{S}_\xi^p$ to denote the spline space spanned by the B-splines with degree p and knot vector ξ. Given a B-spline space $\mathbb{S}_\xi^p$ and n control points $C_i \in \mathbb{R}^D$, a D-dimensional B-spline curve of degree p is defined by

$$F : \mathbb{R} \to \mathbb{R}^D, \quad F(\hat{x}) = \sum_{i=1}^{n} C_i \beta_{i,p}(\hat{x}). \tag{2.4}$$

Unfortunately, conic shapes can not be represented exactly with B-splines [30]. Therefore, a generalization of the B-splines was developed, called NURBS [31]. The term *nonuniform* refers to the fact that NURBS usually are defined by a knot vector with nonuniformly sized knot spans. NURBS are used in a wide spectrum of computational application, especially in computer-aided design or computer-generated imagery environments, where they became the standard tool to model any kind of required shape [12]. By multiplying the B-spline functions with weights and rationalizing the curve, a NURBS curve is defined as

$$N(\hat{x}) = \sum_{i=1}^{n} C_i \frac{\beta_i(\hat{x}) w_i}{\sum_{j=1}^{n} \beta_j(\hat{x}) w_j}. \tag{2.5}$$

Note that we will omit the index p of the basis functions in the remainder of this paper and refer to the basis functions as $\beta_i \in \mathbb{S}_\xi^p$ for simplicity. To represent arbitrary D-dimensional geometries with B-splines or NURBS as necessary in isogeometric analysis, multivariate spaces are constructed via tensor products of univariate spline spaces.

Consider D different univariate spline spaces $\mathbb{S}_{\xi_d}^{p_d}$, each having its own degree p_d and knot vector ξ_d, with $d = 1, \ldots, D$. We use $\hat{x}^{(d)} \in [0,1]$ to denote the one-dimensional variables for each spline space and $\beta_1^{(d)}, \ldots, \beta_{n_d}^{(d)}$ to denote its basis functions.

We obtain a D-variate tensor product spline space $\mathbb{S}^D = \mathbb{S}_{\xi_1}^{p_1} \otimes \cdots \otimes \mathbb{S}_{\xi_D}^{p_D}$ with variables $\hat{x} = (\hat{x}^{(1)}, \ldots, \hat{x}^{(D)})^T$ as a space of piecewise polynomial functions. Its basis functions are denoted by

$$\beta_{\mathbf{i}}(\hat{x}) = \prod_{d=1}^{D} \beta_{i_d}^{(d)}(\hat{x}^{(d)}), \tag{2.6}$$

with multi-index $\mathbf{i} \in \mathbf{I}$ with an index set

$$\mathbf{I} = \{(i_1, \ldots, i_D) \,|\, i_d \in \{1, \ldots, n_d\} \text{ for all } d = 1, \ldots, D\}. \tag{2.7}$$

Let $\Omega \subset R^D$ be an arbitrary geometric shape. A B-spline (or NURBS) geometry mapping $G : \hat{\Omega} \to \Omega$ from the D-dimensional unit cube $\hat{\Omega} := [0,1]^D$ onto Ω can be defined as

$$G(\hat{x}) = \sum_{\mathbf{i} \in \mathbf{I}} C_{\mathbf{i}} \beta_{\mathbf{i}}(\hat{x}) = C : B(\hat{x}) \tag{2.8}$$

with control points $C_{\mathbf{i}} \in \mathbb{R}^D$. Here $C \in \mathbb{R}^{D \times n_1 \times \cdots \times n_D}$ and $B(\hat{x}) \in \mathbb{R}^{n_1 \times \cdots \times n_D}$ denote the tensors holding all control points $C_{\mathbf{i}}$ and basis functions $\beta_{\mathbf{i}}(\hat{x})$, respectively, and $C : B(\hat{x})$ denotes their Frobenius product.

Now that we have a spline representation of the geometry Ω, we can use the same spline functions to parameterize the solution space of a PDE on the geometry. For the discretization of the optimal control problem (1.1)–(1.4) we need to look at two bilinear forms,

$$a_m(u, v) = \langle u, v \rangle_2 = \int_\Omega uv \, \mathrm{d}x, \tag{2.9}$$

$$a_s(u, v) = -\int_\Omega (\Delta u) v \, \mathrm{d}x = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}x, \tag{2.10}$$

called the mass and stiffness terms.

We want to produce approximations to the solutions $u \in H_0^1(\Omega)$ with discrete functions $u_h \in V_h \subset H_0^1(\Omega)$ constructed with B-splines. In IgA we use the same splines from the geometry mapping (2.8) to parameterize the solution space V_h,

$$V_h = \operatorname{span}\{\beta_{\mathbf{i}} \circ G^{-1} \ : \ \mathbf{i} \in \mathbf{I}\}, \tag{2.11}$$

with basis functions $\beta_{\mathbf{i}}$ and index set $\mathbf{I}$ as in (2.7). The functions in V_h are linear combinations of the basis functions with coefficients $u_{\mathbf{i}} \in \mathbb{R}$,

$$u_h = \sum_{\mathbf{i} \in \mathbf{I}} u_{\mathbf{i}} (\beta_{\mathbf{i}} \circ G^{-1}). \tag{2.12}$$

The space V_h is now used for the Galerkin discretization of the mass and stiffness terms, resulting in the discrete mass and stiffness terms

$$a_m(u_h, v_h) = \int_\Omega u_h(x) v_h(x)\,\mathrm{d}x = \int_{\hat\Omega} \sum_{\mathbf{i}\in\mathbf{I}} u_{\mathbf{i}}\beta_{\mathbf{i}}(\hat x) \sum_{\mathbf{j}\in\mathbf{I}} v_{\mathbf{j}}\beta_{\mathbf{j}}(\hat x)\omega(\hat x)\,\mathrm{d}\hat x, \tag{2.13}$$

$$a_s(u_h, v_h) = \int_\Omega (\nabla u_h(x))\cdot\nabla v_h(x)\,\mathrm{d}x = \int_{\hat\Omega} (Q(\hat x) \sum_{\mathbf{i}\in\mathbf{I}} u_{\mathbf{i}}\nabla\beta_{\mathbf{i}}(\hat x)) \cdot \sum_{\mathbf{j}\in\mathbf{I}} v_{\mathbf{j}}\nabla\beta_{\mathbf{j}}(\hat x)\,\mathrm{d}\hat x \tag{2.14}$$

with the additional terms stemming from the domain transformation,

$$\omega(\hat x) = |\det\nabla G(\hat x)| \qquad \in \mathbb{R}, \tag{2.15}$$

$$Q(\hat x) = \left(\nabla G(\hat x)^T \nabla G(\hat x)\right)^{-1} |\det\nabla G(\hat x)| \qquad \in \mathbb{R}^{D\times D}, \tag{2.16}$$

as introduced in [23].

Assembling the coefficients u_i and v_j in vectors, we can rewrite the bilinear forms into a system of linearly independent equations. The resulting system matrix for the mass term is called the mass matrix M with elements

$$M_{\mathbf{i},\mathbf{j}} = \int_{\hat\Omega} \beta_{\mathbf{i}}\beta_{\mathbf{j}}\omega\,\mathrm{d}\hat x, \tag{2.17}$$

and for the stiffness term we get the stiffness matrix K with elements

$$K_{\mathbf{i},\mathbf{j}} = \int_{\hat\Omega} (Q\nabla\beta_{\mathbf{i}})\cdot\nabla\beta_{\mathbf{j}}\,\mathrm{d}\hat x = \sum_{k,l=1}^{D} \int_{\hat\Omega} q_{k,l} \frac{\partial}{\partial \hat x_l}\beta_{\mathbf{i}} \frac{\partial}{\partial \hat x_k}\beta_{\mathbf{j}}\,\mathrm{d}\hat x. \tag{2.18}$$

In the derivation of the mass and stiffness matrices, we did not exploit the tensor product structure of $\mathbb{S}^D$ yet. We can arrange M and K either as matrices or as tensors of size $(\mathbf{n},\mathbf{n}) = (n_1,\dots,n_D,n_1,\dots,n_D)$, which produces a more compact representation in high-dimensional settings.

Let B again denote the tensor of order D and size $\mathbf{n} = (n_1,\dots,n_D)$ holding every $\beta_{\mathbf{i}} \in \mathbb{S}^D$ as in (2.8). All combinations of elements $\beta_{\mathbf{i}}\beta_{\mathbf{j}}$ which make up the integrands of (2.17) are included in the tensor product $B\otimes B$. With this consideration, we can write the mass tensor as

$$M = \int_{\hat\Omega} \omega B\otimes B\,\mathrm{d}\hat x \quad \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}. \tag{2.19}$$

Similarly, we can write the stiffness tensor with integrands from (2.18) as

$$K = \sum_{k,l=1}^{D} \int_{\hat\Omega} q_{k,l} \frac{\partial}{\partial \hat x_l} B \otimes \frac{\partial}{\partial \hat x_k} B\,\mathrm{d}\hat x. \tag{2.20}$$

We will exploit these tensor structures in a low-rank scheme to reduce the complexity of the assembly procedure.

3. Low-rank IgA.

Looking at the mass and stiffness matrices (2.17) and (2.18), we see that their entries are the product of univariate B-splines and a D-variate weight function, $\omega(\hat{x})$ or $Q(\hat{x})$. The scalar $\omega(\hat{x}) = |\det \nabla G(\hat{x})|$ and the matrix $Q(\hat{x}) = (\nabla G(\hat{x})^{-1})(\nabla G(\hat{x}))^{-T}\omega(t) \in \mathbb{R}^{D\times D}$ are determined by the geometry mapping. As Mantzaflaris et al. suggest in [23], we can approximate these weight functions by some combination of univariate functions via interpolation.

For the low-rank approximation of the mass and stiffness matrix, we then approximate the arising multidimensional integrals as products of univariate integrals. The integrands are separable after interpolating the weight functions. To further reduce the computation time and storage requirements of the mass and stiffness matrix calculation, the resulting interpolating function is approximated with low-rank methods giving low-rank approximations of the system matrices [22, 23].

To do so, we interpolate the weight functions in a multivariate spline space $\tilde{\mathbb{S}}^D$ of higher order with suitable knot vectors $\tilde{\xi}_d$ and degrees $\tilde{p}_d > p_d$ for $d = 1, \ldots, D$. The weight function $\omega(\hat{x})$ of (2.17) is interpolated as

$$\omega(\hat{x}) \approx \sum_{\mathbf{j}\in\mathbf{J}} W_{\mathbf{j}}\tilde{\beta}_{\mathbf{j}}(\hat{x}) = W : \tilde{B}(\hat{x}), \tag{3.1}$$

where $\tilde{\beta}_{\mathbf{j}}(\hat{x})$ are the basis functions of the spline space $\tilde{\mathbb{S}}^D$ and $\tilde{B}(\hat{x})$ is the tensor holding all $\tilde{\beta}_{\mathbf{j}}$ with the index set $\mathbf{J}$. The weight tensor W has the same dimension as the spline space $\tilde{\mathbb{S}}^D$, being $(\tilde{n}_1, \ldots, \tilde{n}_D)$, and we obtain its entries by interpolating the weight function in a sufficient number of points, i.e., $\tilde{n} = \tilde{n}_1 \cdots \tilde{n}_D$.

As the derivatives of a B-spline or NURBS are again B-splines or NURBS, the weight function $\omega(\hat{x}) = |\det \nabla G(\hat{x})|$ is again the absolute value of a spline of univariate degrees $Dp_d - 1$ [23]. Thus we can get an exact interpolation if we choose basis functions of degree $Dp - 1$.

Now assume we have a low-rank representation of the weight tensor,

$$W \approx \sum_{r=1}^{R} \bigotimes_{d=1}^{D} w_r^{(d)} =: W_R \tag{3.2}$$

with $w_r^{(d)} \in \mathbb{R}^{n_d}$. With this we can get a low-rank representation of the weight function,

$$\omega(\hat{x}) \approx W_R : \tilde{B}(\hat{x}) = \sum_{r=1}^{R} \prod_{d=1}^{D} w_r^{(d)} \cdot \tilde{\beta}^{(d)}(\hat{x}^{(d)}). \tag{3.3}$$

Here $\tilde{\beta}^{(d)}(\hat{x}^{(d)}) \in \mathbb{R}^{n_d}$ denotes the vector holding all univariate basis functions evaluated in $\hat{x}^{(d)}$, and "$\cdot$" is the scalar product.

The entries of the mass matrix can be approximated using this low-rank representation, where we can calculate each entry as the sum of products of univariate integrals,

$$\begin{aligned} M_{\mathbf{i},\mathbf{j}} &= \int_{\hat{\Omega}} \prod_{d=1}^{D} \beta_{i_d}^{(d)} \beta_{j_d}^{(d)} \sum_{r=1}^{R} \prod_{d=1}^{D} w_r^{(d)} \cdot \tilde{\beta}^{(d)} \,\mathrm{d}\hat{x} \\ &= \sum_{r=1}^{R} \prod_{d=1}^{D} \int_0^1 \beta_{i_d}^{(d)} \beta_{j_d}^{(d)} w_r^{(d)} \cdot \tilde{\beta}^{(d)} \,\mathrm{d}\hat{x}^{(d)}. \end{aligned} \tag{3.4}$$

For these univariate integrals, we define the univariate mass matrix of some weight function $\omega^{(d)}$ as

$$M^{(d)}(\omega^{(d)}) = \int_0^1 B^{(d)} \otimes B^{(d)} \omega^{(d)} \, \mathrm{d}\hat{x}^{(d)}, \tag{3.5}$$

where $B^{(d)} \in \mathbb{R}^{n_d}$ is the vector holding all n_d univariate B-splines of $\mathbb{S}^{p_d}_{\xi_d}$. According to the tensor representation in (2.19), we can finally write the mass matrix as a sum of Kronecker products of small univariate mass matrices (3.5) with $\omega^{(d)} = w_r^{(d)} \cdot \tilde{\beta}^{(d)}$,

$$M = \sum_{r=1}^{R} \bigotimes_{d=1}^{D} M^{(d)}(w_r^{(d)} \cdot \tilde{\beta}^{(d)}). \tag{3.6}$$

Note that we get an exact interpolation for the weight function $\omega(\hat{x})$. Depending on the geometry, this may require only a very low number of interpolatory coefficients. Thus, depending on the geometry and truncation tolerance, the tensor decomposition and low-rank approximation may still yield an exact representation for the mass matrix M.

The same procedure can be applied to the weight function of the stiffness matrix, $Q(\hat{x})$. Note that $Q(\hat{x}) \in \mathbb{R}^{D\times D}$; thus we have to apply the interpolation to each entry of Q. Similarly to (3.3), we get the low-rank representations

$$q_{k,l}(\hat{x}) \approx V_{k,l,R} : \tilde{B}(\hat{x}) = \sum_{r=1}^{R} \prod_{d=1}^{D} v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)}(\hat{x}^{(d)}) \qquad \text{for all } k,l = 1,\dots,D, \tag{3.7}$$

with $v_{k,l,r}^{(d)} \in \mathbb{R}^{n_d}$. Using this low-rank method, we approximate the entries of the stiffness matrix as

$$\begin{aligned} K_{\mathbf{i},\mathbf{j}} &= \sum_{k,l=1}^{D} \int_{\hat{\Omega}} \Big(\prod_{d=1}^{D} \delta(l,d)\beta_{i_d}^{(d)} \delta(k,d)\beta_{j_d}^{(d)} \Big) \sum_{r=1}^{R} \prod_{d=1}^{D} v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)} \, \mathrm{d}\hat{x} \\ &= \sum_{k,l=1}^{D} \sum_{r=1}^{R} \prod_{d=1}^{D} \int_0^1 \delta(l,d)\beta_{i_d}^{(d)} \delta(k,d)\beta_{j_d}^{(d)} v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)} \, \mathrm{d}\hat{x}^{(d)}, \end{aligned} \tag{3.8}$$

where $\delta(k,d)$ denotes the operator acting on f as

$$\delta(k,d)f = \begin{cases} \frac{\partial f}{\partial \hat{x}_d} & \text{if } k = d, \\ f & \text{otherwise.} \end{cases} \tag{3.9}$$

To get a stiffness matrix representation corresponding to the mass matrix representation in (3.6), we define the D^2 univariate stiffness matrices of some weight function $q^{(d)}$ as

$$K_{k,l}^{(d)}(q^{(d)}) = \int_0^1 (\delta(l,d)B) \otimes (\delta(k,d)B)\, q^{(d)} \, \mathrm{d}\hat{x}^{(d)} \qquad \text{for } k,l = 1,\dots,D. \tag{3.10}$$

With $q^{(d)} = v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)}$, the final low-rank tensor representation of the stiffness matrix is

$$K = \sum_{k,l=1}^{D} \sum_{r=1}^{R} \bigotimes_{d=1}^{D} K_{k,l}^{(d)}(v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)}). \tag{3.11}$$

Both (3.6) and (3.11) rely on efficient low-rank representations of W and $V_{k,l}$, and we need suitable strategies to perform this task. For a two-dimensional setting, W and $V_{k,l}$ are matrices, and a singular value decomposition (SVD) can be applied easily to find a low-rank representation [22]. We approximate the matrix $W \in \mathbb{R}^{\tilde{n}_1 \times \tilde{n}_2}$ as

$$W = U\Sigma V^T \approx \sum_{r=1}^{R} u_r \sigma_r v_r^T = \sum_{r=1}^{R} (u_r \sqrt{\sigma_r}) \otimes (v_r \sqrt{\sigma_r}) \tag{3.12}$$

with $U \in \mathbb{R}^{\tilde{n}_1 \times \tilde{n}_1}$, $V \in \mathbb{R}^{\tilde{n}_2 \times \tilde{n}_2}$, and $\Sigma \in \mathbb{R}^{\tilde{n}_1 \times \tilde{n}_2}$ is the rectangular matrix holding the sorted singular values σ_i, $i = 1, \dots, \min(\tilde{n}_1, \tilde{n}_2)$ on its main diagonal. The low-rank approximation is derived by truncating all but the R largest singular values and the corresponding rows of U and V.

In higher-dimensional settings, the decomposition becomes more challenging, and different types of low-rank tensor approximations can be applied as well as considering only a partial decomposition, e.g., into a univariate and a bivariate integration in three-dimensional settings [33].

For the low-rank tensor approximation of a D-dimensional tensor as in (3.2), there exists a multitude of possible approximations, e.g., the higher-order SVD (HOSVD) [6], or a canonical polyadic (CP) decomposition [34]. However, the approximation problem in the CP format is ill-posed [7] and might be numerically unstable, more pressingly so if we are seeking a low-rank solution to a linear system. The HOSVD (also known as the Tucker format) still suffers from the curse of dimensionality, which makes it numerically expensive and unsuitable due to its D-dimensional core tensor. Thus, we switch to the more robust TT decomposition [25] in the rest of the paper, also because of the availability of tailored solvers. Note that the hierarchical Tucker format [14] could also be used.

A tensor W is said to be in TT format if it can be written as

$$W(i_1, \dots, i_D) = W_1(i_1) \cdots W_D(i_D), \tag{3.13}$$

where $W_d(i_d)$ is an $R_{d-1} \times R_d$ matrix for each fixed i_d, $1 \le i_d \le n_d$ and $R_0 = R_D = 1$ [25].

By rearranging the matrices $W_d(i_d)$ into D tensors of sizes $R_{d-1} \times n_d \times R_d$ we can rewrite the TT format into a low-rank representation as desired in (3.2),

$$W = \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^{D} W_d(r_{d-1}, :, r_d). \tag{3.14}$$

To interpolate each weight function, we inherently need to solve a large system of equations $\omega(\hat{X}) = W : \tilde{B}(\hat{X})$, where $\hat{X}$ denotes the set of $\tilde{n} = \tilde{n}_1 \cdots \tilde{n}_D$ interpolation points. This equation can be rewritten into

$$\operatorname{vec}(\omega(\hat{X})) = \underbrace{\left(B^{(1)}(\hat{X}^{(1)}) \otimes \cdots \otimes B^{(D)}(\hat{X}^{(D)})\right)}_{A} \operatorname{vec}(W). \tag{3.15}$$

The matrix $A \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$ in (3.15) can be very large for a direct solution. However, we can first approximate the tensor holding the values of ω evaluated in all interpolation points $\hat{X}$, $\omega(\hat{X})$, with a tensor decomposition, and then use the Kronecker structure of A for an efficient computation of a tensor decomposition of W without assembling

A. Indeed, assuming that we have the tensor of interpolation values $\omega(\hat{X})$ in the TT format

$$\omega(\hat{X}) = \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^{D} \operatorname{vec}(\omega_d(r_{d-1},:,r_d)), \tag{3.16}$$

we can write the TT format (3.14) for W in the form

$$W = \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^{D} \left[B^{(d)}(\hat{X}^{(d)})\right]^{-1} \operatorname{vec}(\omega_d(r_{d-1},:,r_d)), \tag{3.17}$$

which requires solving d linear systems of sizes $\tilde{n}_1, \ldots, \tilde{n}_D$, respectively. The TT approximation for $\omega(\hat{X})$ could be precomputed by the TT-SVD [25] or the TT-Cross [28] methods to further improve the efficiency. This strategy and the efficient representation of M and K allow us to tackle PDE-constrained optimal control problems next.

4. A PDE-constrained optimization model problem.

We recall the optimal control problem,

$$\min_{y,u} \quad \frac{1}{2}\int_0^T \int_\Omega (y-\hat{y})^2 \,\mathrm{d}x\,\mathrm{d}t + \frac{\alpha}{2}\int_0^T \int_\Omega u^2 \,\mathrm{d}x\,\mathrm{d}t \tag{4.1}$$

$$\text{s.t.} \quad y_t - \Delta y = u \quad \text{in } (0,T)\times\Omega, \tag{4.2}$$

$$y = 0 \quad \text{on } (0,T)\times\partial\Omega, \tag{4.3}$$

$$y = y_0 \quad \text{on } \Omega \text{ for } t=0 \tag{4.4}$$

to a desired state $\hat{y}$ with control u and state y defined on a given geometry Ω and time frame $[0,T]$.

We want to solve this by discretizing in both time and space, resulting in a large saddle point problem [3, 11]. Using an implicit Euler scheme for the time discretization of the PDE and the rectangle rule for the objective function leads to the time-discrete problem

$$\min_{y,u} \quad \sum_{k=1}^{N_t} \frac{\tau}{2}\Big(\int_\Omega (y_k-\hat{y}_k)^2\,\mathrm{d}x + \alpha\int_\Omega u_k^2\,\mathrm{d}x\Big) \tag{4.5}$$

$$\text{s.t.} \quad \frac{y_k - y_{k-1}}{\tau} - \Delta y_k = u_k \quad \text{in } \Omega, \text{ for } k=1,\ldots,N_t, \tag{4.6}$$

$$y_k = 0 \quad \text{on } \partial\Omega, \text{ for } k=1,\ldots,N_t, \tag{4.7}$$

with the number of time steps N_t corresponding to the time step size $\tau = T/N_t$ and continuous solution y_k in each time step k and initial condition y_0.

Using the Galerkin-based spatial discretization as described in section 2 leads to the discrete quadratic problem

$$\min_{y,u} \quad \sum_{k=1}^{N_t} \frac{\tau}{2}\big((y_k-\hat{y}_k)^T M (y_k-\hat{y}_k) + \alpha u_k^T M u_k\big) \tag{4.8}$$

$$\text{s.t.} \quad \frac{My_k - My_{k-1}}{\tau} + Ky_k = Mu_k \quad \text{for } k=1:N_t, \tag{4.9}$$

where M and K are the mass and stiffness matrix of the chosen discretization. Here the zero boundary conditions (4.3) are integrated in M and K by omitting the boundary nodes. Disregarding the notation from the time-continuous problem (4.5), the states are collected in the vector $y = [y_1, \dots, y_{N_t}]^T$ with each state y_k being the vector of the corresponding coefficients (2.12). The same notation is used for the control parameters u_k. Note that y_k and u_k are vectors of appropriate dimensionality.

The *Karush–Kuhn–Tucker* (KKT) conditions [4] for optimization problems with linear constraints state that if (y^*, u^*) is a minimum, then there exists a multiplier vector λ^* such that the *Lagrangian* of the problem, here

$$(4.10)\quad \mathcal{L}(y,u,\lambda) = \sum_{k=1}^{N_t} \bigg(\frac{\tau}{2}\big((y_k - \hat{y}_k)^T M (y_k - \hat{y}_k) + \alpha u_k^T M u_k\big) \\ + \lambda_k^T \big(My_k - My_{k-1} + \tau K y_k - \tau M u_k\big)\bigg),$$

has a saddle point in (y^*, u^*, λ^*),

$$(4.11)\qquad \nabla \mathcal{L}(y^*, u^*, \lambda^*) = 0.$$

For our discrete optimization problem this results in the conditions

$$(4.12)\qquad 0 = \nabla_{y_k}\mathcal{L}(y,u,\lambda) = \tau M(y_k - \hat{y}_k) - M\lambda_{k+1} + (M + \tau K^T)\lambda_k,$$

$$(4.13)\qquad 0 = \nabla_{u_k}\mathcal{L}(y,u,\lambda) = \tau\alpha M u_k - \tau M \lambda_k,$$

$$(4.14)\qquad 0 = \nabla_{\lambda_k}\mathcal{L}(y,u,\lambda) = M(y_k - y_{k-1}) + \tau K y_k - \tau M u_k$$

for all $k = 1, \dots, N_t$.

We can rewrite these equations into an equation system

$$(4.15)\qquad \begin{bmatrix} \tau\mathcal{M} & 0 & \mathcal{K}^T \\ 0 & \tau\alpha\mathcal{M} & -\tau\mathcal{M} \\ \mathcal{K} & -\tau\mathcal{M} & 0 \end{bmatrix} \begin{bmatrix} y \\ u \\ \lambda \end{bmatrix} = \begin{bmatrix} \tau\mathcal{M}\hat{y} \\ 0 \\ 0 \end{bmatrix},$$

where $\mathcal{M} = \mathcal{I}_{N_t} \otimes M$ and $\mathcal{K} = \mathcal{I}_{N_t} \otimes \tau K + C \otimes M$, where $\mathcal{I}_{N_t}$ is the $N_t \times N_t$ identity matrix and C is

$$(4.16)\qquad C = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix}.$$

Note that in this derivation we used the same spline spaces for the state and control, but it is also possible to have a different discretization for the control. Assume we discretized the geometry as $G(\hat{x}) = \sum_{\mathbf{i}\in\mathbf{I}} C_{\mathbf{i}}\beta_{\mathbf{i}}(\hat{x})$ and used the same basis functions to discretize y as in (2.11),

$$(4.17)\qquad V_h = \operatorname{span}\{\beta_{\mathbf{i}} \circ G^{-1} : \mathbf{i} \in \mathbf{I}\}.$$

Instead of this spline space we can use another set of D-dimensional B-splines $\hat{\beta}_{\mathbf{j}}$, $\mathbf{j} \in \mathbf{J}$ and use the discretization

$$(4.18)\qquad U_h = \operatorname{span}\{\hat{\beta}_{\mathbf{j}} \circ G^{-1} : \mathbf{j} \in \mathbf{J}\}$$

for the control. This can be a spline space with different degree or a subset for local control, e.g., boundary control. With this we can use the same weight function for the mass matrices $\omega(\hat{x})$ and get the discretized problem

$$\min_{y,u} \quad \sum_{k=1}^{N_t} \frac{\tau}{2}\left((y_k - \hat{y}_k)^T M (y_k - \hat{y}_k) + \alpha u_k^T M_1 u_k\right) \tag{4.19}$$

$$\text{s.t.} \quad \frac{My_k - My_{k-1}}{\tau} + Ky_k = M_2 u_k \qquad \text{for } k = 1 : N_t, \tag{4.20}$$

where M_1 is a square mass matrix of different size than M and M_2 is a rectangular matrix, both assembled in the same low-rank structure as M.

The resulting equation system (4.15) is a saddle point problem as described in [3, 35, 29]. Different techniques to solve such a system include a variational discretization concept as introduced in [16] that avoids discretizing the control explicitly. This would reduce the dimensionality of the system, but the complexity of designing a suitable iterative solver would remain similar due to the nonzero block that would arise and the Schur complement being unchanged. Nevertheless, this is an attractive option, and we believe that all the techniques presented here are applicable in that case. Additionally, one could solve a system with $\tau\alpha\mathcal{M} + \tau^3\mathcal{M}\mathcal{K}^{-T}\mathcal{M}\mathcal{K}^{-1}\mathcal{M}$. In our case this system is now symmetric and positive definite, but an accurate evaluation of the matrix vector multiplication would require resolving $\mathcal{K}^{-T}$ and $\mathcal{K}^{-1}$ very accurately, which is often not feasible. In [13] the authors replace $\mathcal{K}^{-1}$ and $\mathcal{K}^{-T}$ by their preconditioners resulting in inexact matrix vector products. While this seems to work well for the elliptic problem considered there, we want to resolve the matrix vector product as accurately as possible to reflect the optimality conditions for state, control, and adjoint state here. Therefore we stick with the full saddle point system in (4.15) for the remainder of this work where we only need to multiply with the PDE blocks.

For preconditioning the saddle point system, very crude approximations of $\mathcal{K}^{-T}$ and $\mathcal{K}^{-1}$ are often sufficient. At convergence we satisfy the optimality conditions to the desired accuracy independent of how accurately the preconditioner was chosen. But due to the structure of our system matrices, i.e., they are composed of the sum of many terms, it is not trivial to provide a good preconditioner. We will investigate this in future research and test this for the saddle point formulation as well as the reduced formulation.

5. Low-rank solvers for the PDE-constrained optimization problem.

The saddle point problem (4.15) typically becomes very large, depending on the number of time steps and refinement in the spatial discretization. By exploiting the tensor product structure for both the solution and the coefficients from section 3, we can reduce the problem to smaller linear systems on the elements of individual TT blocks.

As we can represent the low-rank mass and stiffness matrices as sums of Kronecker products, we can rewrite

$$\mathcal{M} = \mathcal{I}_{N_t} \otimes \left(\sum_{r=1}^{R} \bigotimes_{d=1}^{D} M_r^{(d)}\right), \tag{5.1}$$

$$\mathcal{K} = \mathcal{I}_{N_t} \otimes \left(\sum_{k,l=1}^{D} \sum_{r=1}^{R} \bigotimes_{d=1}^{D} K_{k,l,r}^{(d)}\right) + C \otimes \left(\sum_{r=1}^{R} \bigotimes_{d=1}^{D} M_r^{(d)}\right). \tag{5.2}$$

With this, each block of (4.15) becomes a sum of Kronecker products of small matrices. This structure can be preserved and exploited in appropriate linear solvers, such as alternating linear scheme (ALS) [18], density matrix renormalization group [32, 21], and AMEn [10]. However, the fact that the matrix in (4.15) is indefinite might yield instabilities in the vanilla versions of these algorithms. We use an extended block ALS method, which preserves the block structure in (4.15) and hence the numerical stability.

This algorithm aims to approximate all solution components y, u, λ in the same representation, called block TT decomposition [9]. Let us collect y, u, λ as a matrix

$$f = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix} = \begin{bmatrix} y & u & \lambda \end{bmatrix}, \tag{5.3}$$

where the components are referred to as f_ℓ, $\ell = 1, 2, 3$. The block TT decomposition incorporates the ℓ-index into one of the factors. Instead of (3.13), we write

$$f_\ell(i_1, \ldots, i_D) = F_1(i_1) \cdots F_{d-1}(i_{d-1}) \cdot \hat{F}_d(i_d, \ell) \cdot F_{d+1}(i_{d+1}) \cdots F_D(i_D) \tag{5.4}$$

for some $1 \le d \le D$. We can put the component enumerator ℓ into an arbitrary TT factor using the SVD. Suppose we want to move ℓ from the factor d to $d+1$. Consider $\hat{F}_d$ as an $R_{d-1}n_d \times 3R_d$ matrix, $\hat{F}_d(r_{d-1}, i_d;\ \ell, r_d)$, and compute the truncated SVD

$$\hat{F}_d \approx U \Sigma V^\top.$$

Now we call U the dth TT factor instead of $\hat{F}_d$ and multiply $\Sigma V^\top$ with the $(d+1)$th factor,

$$F_d(r_{d-1}, i_d, r'_d) = U(r_{d-1}, i_d;\ r'_d), \tag{5.5}$$

$$\hat{F}_{d+1}(r'_d, i_{d+1}, \ell, r_{d+1}) = \sum_{r_d=1}^{R_d} \Sigma V^\top(r'_d;\ \ell, r_d) F_{d+1}(r_d, i_{d+1}, r_{d+1}). \tag{5.6}$$

We have obtained the same representation as (5.4) with ℓ sitting in the $(d+1)$th factor. This process can be continued further or reversed in order to place ℓ in an arbitrary factor. Note that the TT ranks might change during this procedure; in particular, R_d might increase up to $\min(R_{d-1}n_d, 3R_d)$.

The ALS algorithm replaces a (difficult) task of finding all elements of a TT decomposition simultaneously by a (easier) task of computing only one TT factor at a time. We can observe that the TT representation is linear with respect to the elements of each factor. Indeed, introduce the following $n^D \times R_{d-1}n_dR_d$ *frame* matrix:

$$\begin{aligned} F_{\neq d}(i_1, \ldots, i_D;\ r_{d-1}, j_d, r_d) = F_1(i_1) &\cdots F_{d-1}(i_{d-1}, r_{d-1}) \\ &\cdot \mathcal{I}_{i_d, j_d} \\ &\cdot F_{d+1}(r_d, i_{d+1}) \cdots F_D(i_D), \end{aligned} \tag{5.7}$$

where $\mathcal{I}_{i_d,j_d}$ is the identity matrix with respect to the indices i_d, j_d. In case of the block TT decomposition (5.4), we assume that we choose the same d for both the position of ℓ in (5.4) and the position of the identity matrix in (5.7). We can then observe that

$$f_\ell = F_{\neq d} \cdot \mathrm{vec}(\hat{F}_d(\ell)). \tag{5.8}$$

This linearity allows us to project the original problem into a subspace spanned by the columns of $F_{\neq d}$. Iterating over all $d = 1, \ldots, D$, we obtain the ALS algorithm. This method starts from some initial guess in the low-rank TT representation, and hence it never encounters the original (prohibitively large) tensors.

The block ALS method [2, 8] projects each of the *submatrices* of (4.15) onto the frame matrix individually. For each selected $d = 1, \ldots, D$, we compute the elements of $\hat{F}_d$ from the following reduced KKT system:

$$\begin{bmatrix} \tau F_{\neq d}^T \mathcal{M} F_{\neq d} & 0 & F_{\neq d}^T \mathcal{K}^T F_{\neq d} \\ 0 & \tau\alpha F_{\neq d}^T \mathcal{M} F_{\neq d} & -\tau F_{\neq d}^T \mathcal{M} F_{\neq d} \\ F_{\neq d}^T \mathcal{K} F_{\neq d} & -\tau F_{\neq d}^T \mathcal{M} F_{\neq d} & 0 \end{bmatrix} \begin{bmatrix} \text{vec } \hat{F}_d(1) \\ \text{vec } \hat{F}_d(2) \\ \text{vec } \hat{F}_d(3) \end{bmatrix} = \begin{bmatrix} \tau F_{\neq d}^T \mathcal{M} \hat{y} \\ 0 \\ 0 \end{bmatrix}. \tag{5.9}$$

This system is small (each submatrix is now of size $R_{d-1} n_d R_d$), and can be solved efficiently by, e.g., the minimal residual (MINRES) method. Moreover, since $F_{\neq d}$ inherits the TT decomposition of f, and the system matrices $\mathcal{M}$ and $\mathcal{K}$ have the tensor product structure (5.1), (5.2), the reduced matrices $F_{\neq d}^T \mathcal{M} F_{\neq d}$ and $F_{\neq d}^T \mathcal{K} F_{\neq d}$ can be assembled efficiently using the multiplication of TTs factor by factor [25, 18]. Having solved (5.9), we plug the new factor $\hat{F}_d$ back into the block TT decomposition (5.4), which provides an updated approximation to y, u, and λ through (5.3). In order to prepare the next ALS step we move the enumerator ℓ to the next factor using SVD (e.g., (5.5)–(5.6) in the forward sweep $d \to d+1$) and recompute the corresponding frame matrix using the new F_d factor. The SVD makes the appropriate matricization of F_d orthogonal such that the whole frame matrix $F_{\neq d}$ is orthogonal in each step. This ensures invertibility of the projected matrix in (5.9).

In principle, the block ALS method described above is already rank adaptive, since the TT ranks may change when we switch to the next step (5.5)–(5.6). However, one can facilitate the convergence further by an explicit expansion of the TT factors with a TT approximation of the residual, as in the AMEn method [10]. This general block AMEn is implemented in the *amen_block_solve.m* function in the TT-Toolbox [26], which we use in our computations.

6. Numerical experiments.

The performance of the low-rank TT method highly depends on the geometry, as the interpolation becomes more challenging and the ranks grow with increasing complexity of the geometry. We conduct some numerical experiments of different complexity to show the advantages of the low-rank assembly compared to the full assembly of the stiffness matrix, before combining the assembly with optimal control problems to show the performance of our method for solving large-scale saddle point problems in a low-rank TT format.

For our numerical experiments we used MATLAB R2018b with the TT-Toolbox [26] on a desktop computer with an Intel Core i7-4770 Quad-Core processor running at 4×3400 MHz with 32 GB of RAM.

For each geometry, we first compare the assembly of the stiffness matrix in low-rank format with a full standard assembly performed by the IgA toolbox GeoPDEs 3.0 [38] in MATLAB. The assembly is compared for different levels of refinement, and for each refinement we insert one additional knot per knot section in each spatial dimension. We use the same Gauss–Legendre quadrature rule with five quadrature nodes in each spatial direction for both assemblies.

Additionally, we examine the performance of our low-rank scheme to solve an optimal control problem, comparing the results to a full solution and analyzing the overall performance of the method on large-scale systems.

6.1. Quarter annulus domain.

We first look at a very simple domain which exhibits low-rank properties, a quarter annulus. The domain is depicted in Figure 1(a). The weight function ω can be interpolated exactly by only one combination of univariate spline functions, thus giving us the rank 1. The entries of Q

Table 1
Ranks for the weight function tensor approximations of Figure 1(a).

Degree $\tilde{p}$	TT-ranks	$q_{1,1}$	$q_{1,2}$	$q_{1,3}$	$q_{2,2}$	$q_{2,3}$	$q_{3,3}$	ω
5	R_1	1	0	0	1	0	1	1
	R_2	1	0	0	1	0	1	1
7	R_1	1	0	0	2	0	2	1
	R_2	1	0	0	2	0	2	1

Table 2
Relative errors to the full solution for different residual thresholds in the block AMEn method.

Threshold ε	Relative error	Iterations	Max. solution rank
1e-04	3.3871e-04	3	25
1e-06	9.3282e-06	3	36
1e-08	8.3753e-08	4	45
1e-10	4.0481e-09	4	46
1e-12	1.0935e-11	8	44

can also be approximated with a very small number of basis functions. The resulting ranks are outlined in Table 1. We set the truncation tolerance for solving (3.17) to 10^{-14}, and our method detects these low ranks without prior knowledge of the nature of the geometry.

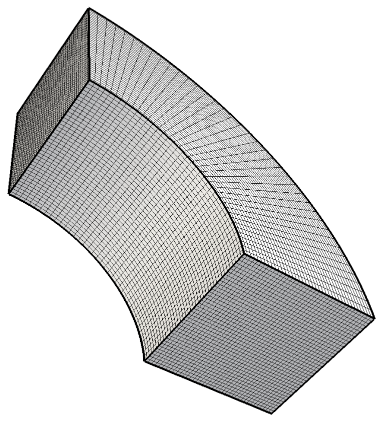
Figure 1(b) shows that the TT method is a lot faster than the full assembly, especially for a high number of degrees of freedom the computation time is decreased to about 1% of the full computation time. The graph in Figure 1(c) depicts the mean difference of the matrices in the Frobenius norm,

$$\text{diff} = \frac{\|S - \tilde{S}\|_F}{\|S\|_F}. \tag{6.1}$$

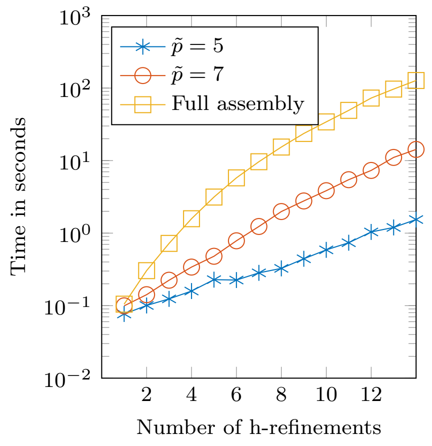
We used two different spline spaces for the interpolation, one with splines of degree $\tilde{p} = 5$ and one with degree $\tilde{p} = 7$ in each spatial direction. Enlarging the degrees leads to a great accuracy gain even for coarse discretizations at a very low cost. Figure 1(d) shows the advantages in regard of storage requirements. For the low-rank method we only need to store a small number of small sparse matrices instead of storing the large stiffness matrix. This reduces the required memory drastically, e.g., for 12 refinements memory requirements are decreased to roughly 0.5% of the full matrix almost without any loss in accuracy. If a high refinement is desired, this effect is amplified.

We now illustrate the performance of the block AMEn method from section 5 on a control example with the quarter annulus domain. We first compare the method with a full solution of the KKT system for different residual tolerances ε for the block AMEn method. As a right-hand side we chose an isogeometric B-spline interpolation of the time-constant function $\hat{y}(x, y, z) = \exp(-4((x-1)^2 - (y-1)^2 - (z-0.4)^2))$ with forced zero boundary values by setting the coefficient for the first and last B-spline basis function in each spatial direction to zero. We divide the time frame into 10 time steps. Note that we will not pay any regard to the variation of the time discretization in this work. However, our experiments showed that increasing the number of time steps does not affect the number of iterative steps for most setups.

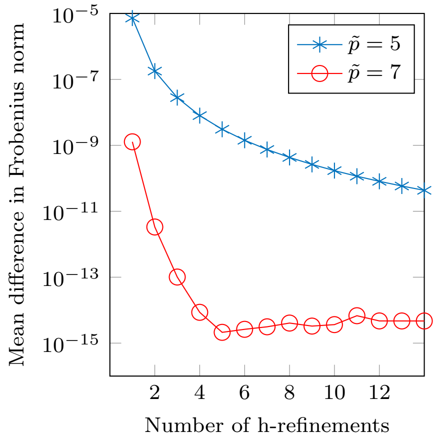
Table 2 exemplarily shows a comparison of our method to the full solution with full matrices derived from GeoPDEs for a low refinement level of 18 degrees of freedom per spatial direction and the regularization parameter $\alpha = 10^{-3}$. The KKT system for



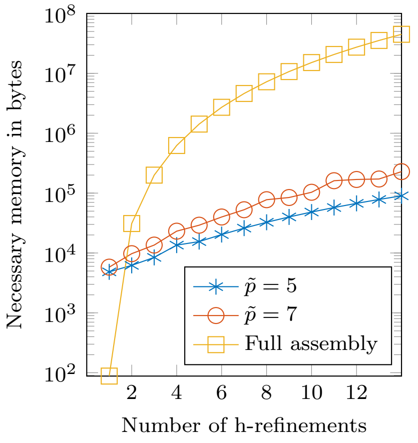
(a) Quarter annulus domain



(b) Time comparision for stiffness matrix assembly



(c) Difference to full assembly stiffness matrix



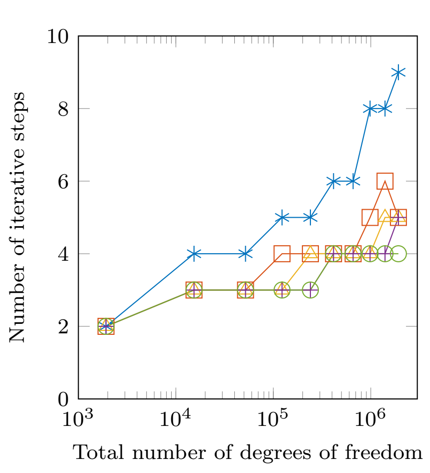
(d) Storage requirements

FIG. 1. *Comparison between full assembly and low-rank assembly with different interpolation spline degrees $\tilde p$ for the stiffness matrix on the quarter annulus domain.*

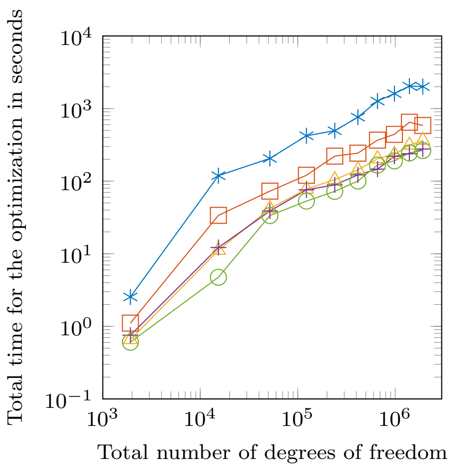
the full matrices was solved using the MATLAB *backslash* operator. The low-rank solution was computed by the block AMEn method with different residual thresholds, ranging from $\varepsilon = 10^{-4}$ to $\varepsilon = 10^{-12}$ using a low-rank stiffness matrix approximation with interpolatory spline degree of $\tilde p = 7$ resulting in an accuracy of roughly 10^{-15}. Our TT method converges to a solution close to the desired residual difference within a low number of iterative steps for each accuracy as can be seen in Table 2. Here the relative error was computed considering all three variables as

$$e = \frac{\|[y,u,\lambda] - [\tilde y, \tilde u, \tilde \lambda]\|}{\|[y,u,\lambda]\|} \tag{6.2}$$

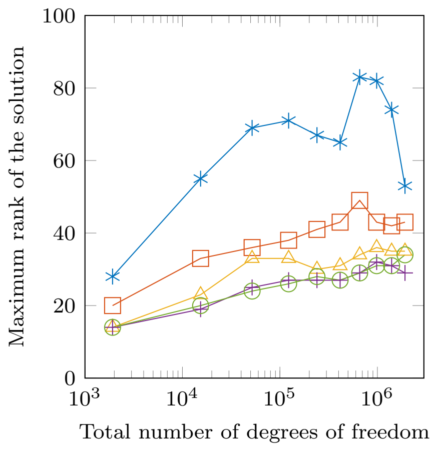
with the full solution $[y,u,\lambda]$ and approximation $[\tilde y, \tilde u, \tilde \lambda]$.



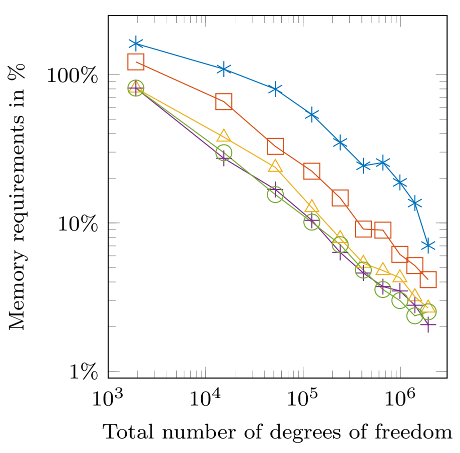
(a) Iterations for Block AMEn solver for different refinements



(b) Computation time for the optimization



(c) Ranks of the solution for different refinements



(d) Memory compression of the low-rank solution in % of the full solution

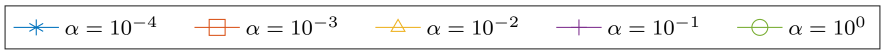
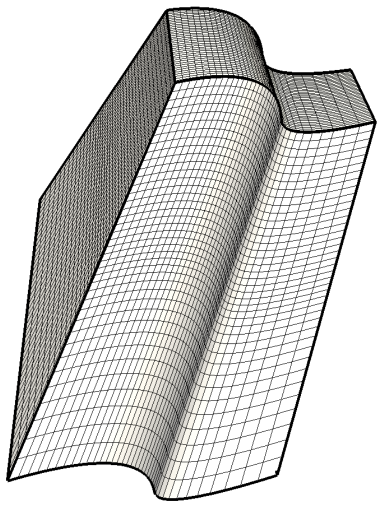


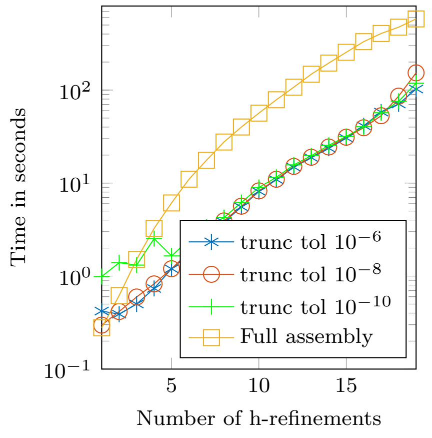
FIG. 2. *Performance for different refinements and control parameters on quarter annulus geometry from Figure* 1(a) *with* $\varepsilon = 10^{-8}$.

Additionally we show the behavior and performance of the low-rank scheme for high numbers of degrees of freedom. We again use equidistant spatial knot insertion to refine the geometric representation and thus the solution space. Here, the desired accuracy for the block AMEn solver was set to $\varepsilon = 10^{-8}$, and the accuracy for the stiffness matrix again is 10^{-15}.

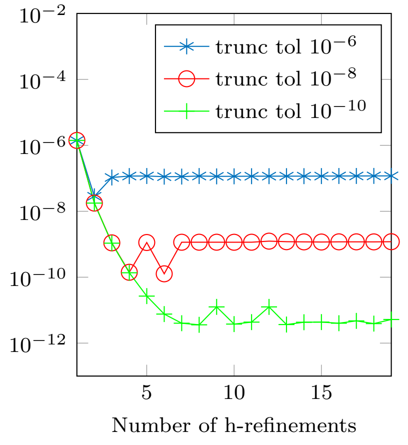
In Figure 2 we see the performance throughout different refinements for different control parameters α. Even for a high number of degrees of freedom the method converges after a small number of iterative steps, as seen in Figure 2(a).



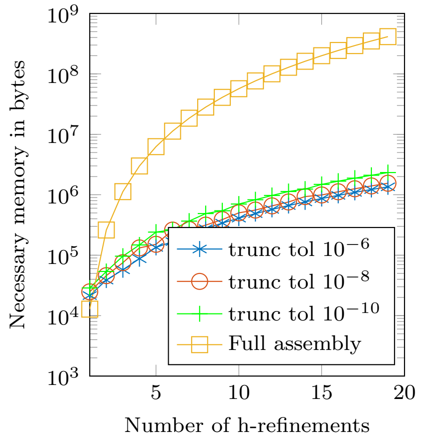
(a) Deformed cuboid domain



(b) Time comparision for stiffness matrix assembly



(c) Difference to full assembly stiffness matrix



(d) Storage requirements

FIG. 3. *Comparison between full assembly and low rank assembly of the stiffness matrix assembly for the domain* 3(a) *for different truncation tolerances.*

Using the block AMEn method to solve the optimal control problem in a low-rank format returns the solution in a low-rank TT format as well. Figure 2(c) illustrates the maximum TT-rank of the solution. Note that due to the adaptive rank determination the ranks for the stiffness matrix entries fluctuate between 1 and 2 for different refinements. This appears to result in a bump in the solution ranks. Nevertheless, even though the ranks become quite large the memory consumption of the solution is reduced drastically. Figure 2(d) displays the storage requirements of the solution in relation to the full solution vector.

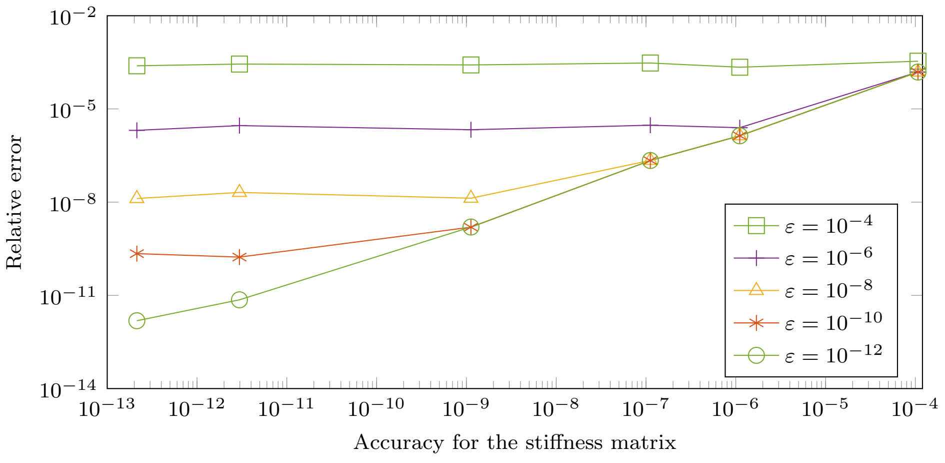


FIG. 4. *Relative error to full solution on deformed cuboid domain from Figure* 3(a) *depending on different accuracies for the stiffness matrix and different residual tolerances* ε *for the block AMEn solver.*

6.2. Deformed cuboid domain. For the quarter annulus domain the stiffness matrix is assembled with a very low rank almost without any loss of accuracy. For most geometries this will not be the case, and a truncation error has to be accepted to obtain low-rank representations. We will examine the relation of truncation errors to the matrix assembly and to the solution of the optimal control problem in this numerical experiment.

The domain considered is a deformed cuboid, depicted in Figure 3(a). This geometry still possesses a low-rank structure, except for the weight function $q_{3,3}$ where we get different high ranks depending on the truncation tolerance. The ranks for different desired accuracies, i.e., truncation tolerances, are displayed in Table 3. These ranks stay constant throughout various levels of refinements once the weight function is interpolated with the requested accuracy. We see the comparison of the full assembly and the TT method with different truncation tolerances being 10^{-6}, 10^{-8}, and 10^{-10} and in Figure 3. We reach the desired accuracies quickly after some refinement steps, as depicted in Figure 3(c). Again, the TT method is faster than the full assembly and has an advantage with respect to the storage requirements especially for high refinements, as seen in Figure 3(b) and 3(d).

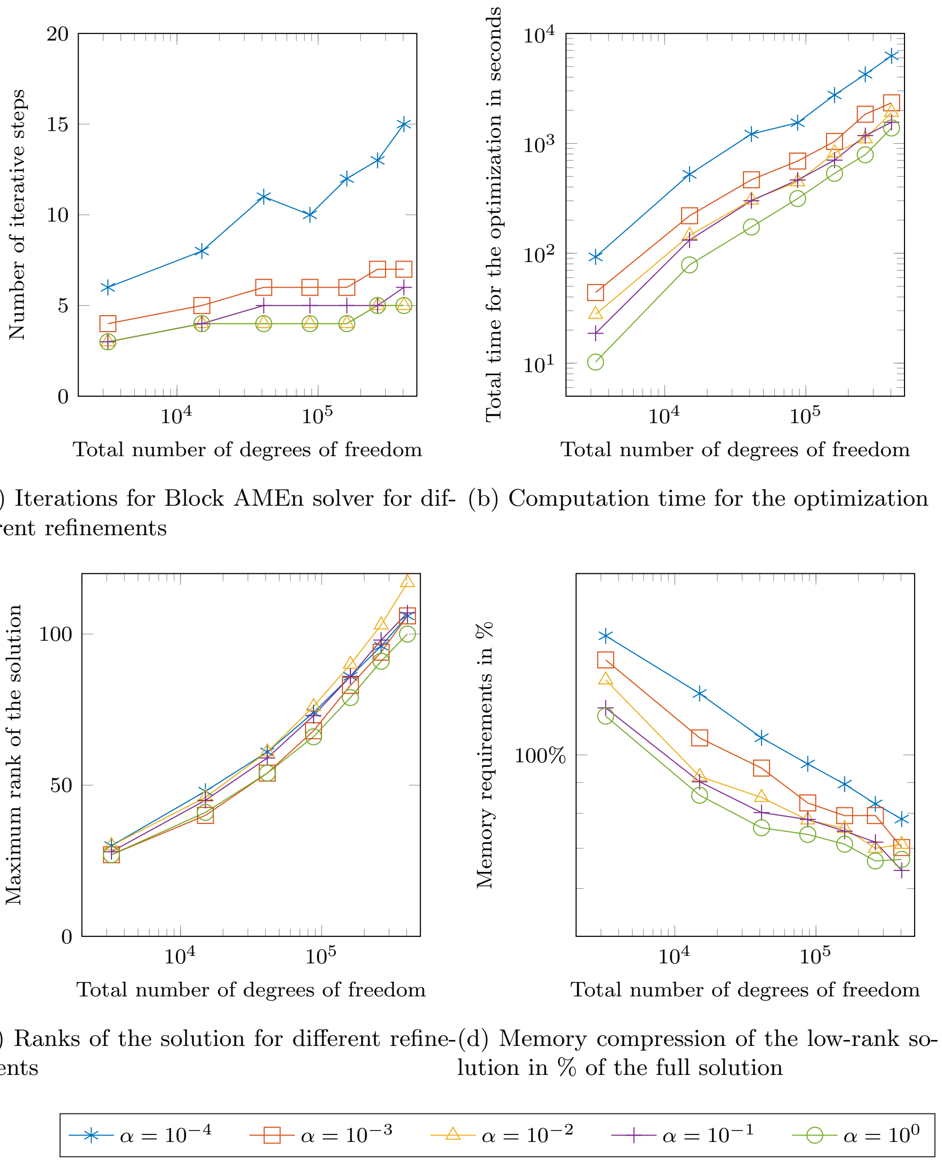
Note that the *bumps* in the error in Figure 3(c) only result from small numerical inaccuracies which lead the method to truncate $q_{3,3}$ with ranks ± 1 here.

We now evaluate the low-rank method on this domain for an optimal control problem. We use the same setup as in section 6.1, shifting the nonzero domain of $\hat{y}(x, y, z)$ to be inside this geometric domain. The corresponding results for the error over the full solution are shown in Figure 4. We see that the accuracy for the stiffness matrix directly translates to the accuracy we can obtain for our optimal control problem. If we have an approximation of the stiffness matrix up to some accuracy, we can expect to obtain a solution of the optimal control problem with similar accuracy.

For this geometry the stiffness term in (5.2) is the sum of 49 Kronecker products. Even with this high rank we achieve promising results, depicted in Figure 5. The number of iterative steps stays quite small throughout refinements as seen in Figure 5(a) whereas the rank of the solution rises with each refinement in Figure 5(c).

Table 3
Ranks for the weight function tensor approximation of Figure 3(a).

Tolerance	TT-ranks	$q_{1,1}$	$q_{1,2}$	$q_{1,3}$	$q_{2,2}$	$q_{2,3}$	$q_{3,3}$	ω
10^{-6}	R_1	2	2	1	2	1	6	2
	R_2	1	1	1	1	1	3	1
10^{-8}	R_1	2	2	1	2	1	8	2
	R_2	1	1	1	1	1	2	1
10^{-10}	R_1	2	2	1	2	2	11	2
	R_2	1	1	1	1	1	3	1



(a) Iterations for Block AMEn solver for different refinements

(b) Computation time for the optimization

(c) Ranks of the solution for different refinements

(d) Memory compression of the low-rank solution in % of the full solution

Fig. 5. *Performance for different refinements and control parameters on deformed cuboid geometry from Figure* 3(a) *with* $\varepsilon = 10^{-8}$.

Even though these ranks get very large we can still report a reduction in memory requirements, as seen in Figure 5(d).

7. Conclusion.

In this paper, we combined the low-rank method presented by Mantzaflaris et al. with TT calculations to obtain a powerful method for solving large equation systems arising from IgA-discretized PDEs. Furthermore, we successfully applied the developed scheme to efficiently solve large PDE-constrained optimal control problems in a low-rank format.

We can reduce the storage requirements and calculation time for the mass and stiffness matrix assembly drastically by finding low-rank approximations and splitting the matrices into a Kronecker product of smaller matrices. Our scheme efficiently finds low-rank approximations for given desired accuracies without any prior knowledge about the geometry. We can exploit the resulting low-rank structures, keeping the memory consumption low throughout further computations. The iterative block AMEn method allows us to solve large systems like a PDE-constrained optimal control problem without assembling the whole equation system. In combination with this iterative method, the low-rank format gives a great advantage, and we can solve very large systems within a reasonably short time and up to high accuracies.

Various numerical experiments show the high potential of the method. However, there might be even further efficiency gains if we find a suitable preconditioner for the reduced linear systems (5.9) in the block AMEn method.

Acknowledgment. The authors would like to thank Angelos Mantzaflaris for his helpful insights.

REFERENCES

[1] P. Antolin, A. Buffa, F. Calabró, M. Martinelli, and G. Sangalli, *Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization*, Comput. Methods Appl. Mech. Engrg., 285 (2015), pp. 817–828, https://doi.org/10.1016/j.cma.2014.12.013.

[2] P. Benner, S. Dolgov, A. Onwunta, and M. Stoll, *Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data*, Comput. Methods Appl. Mech. Engrg., 304 (2016), pp. 26–54, https://doi.org/10.1016/j.cma.2016.02.004.

[3] M. Benzi, H. G. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[4] J. Brinkhuis and V. Tikhomirov, *Optimization: Insights and Applications*, Princeton Ser. Appl. Math., Princeton University Press, Princeton, NJ, 2005.

[5] C. De Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, NY, 1978.

[6] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[7] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127, https://doi.org/10.1137/06066518x.

[8] S. Dolgov and M. Stoll, *Low-rank solution to an optimization problem constrained by the Navier–Stokes equations*, SIAM J. Sci. Comput., 39 (2017), pp. A255–A280, https://doi.org/10.1137/15M1040414.

[9] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, *Computation of extreme eigenvalues in higher dimensions using block tensor train format*, Comput. Phys. Commun., 185 (2014), pp. 1207–1216, https://doi.org/10.1016/j.cpc.2013.12.017.

[10] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J. Sci. Comput., 36 (2014), pp. A2248–A2271.

[11] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative solvers: With Applications in Incompressible Fluid Dynamics*, 2nd ed., Oxford University Press, Oxford, UK, 2014, https://doi.org/10.1093/acprof:oso/9780199678792.001.0001.

[12] G. E. Farin, *NURBS: From Projective Geometry to Practical Use*, AK Peters, Ltd., Natick, MA, 1999.

[13] S. Garreis and M. Ulbrich, *Constrained optimization with low-rank tensors and applications to parametric problems with pdes*, SIAM J. Sci. Comput., 39 (2017), pp. A25–A54.

[14] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.

[15] R. R. Hiemstra, F. Calabró, D. Schillinger, and T. J. Hughes, *Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis*, Comput. Methods Appl. Mech. Engrg., 316 (2017), pp. 966–1004, https://doi.org/10.1016/j.cma.2016.10.049.

[16] M. Hinze, *A variational discretization concept in control constrained optimization: The Linear-quadratic case*, Comput. Optim. Appl., 30 (2005), pp. 45–61, https://doi.org/10.1007/s10589-005-4559-5.

[17] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Springer-Verlag, New York, NY, 2009.

[18] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713, https://doi.org/10.1137/100818893.

[19] T. Hughes, J. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 4135–4195.

[20] T. Hughes, A. Reali, and G. Sangalli, *Efficient quadrature for nurbs-based isogeometric analysis*, Comput. Engrg. Methods Appl. Mech. Engrg., 199 (2010), pp. 301–313, https://doi.org/10.1016/j.cma.2008.12.004.

[21] E. Jeckelmann, *Dynamical density–matrix renormalization–group method*, Phys. Rev. B, 66 (2002), 045114, https://doi.org/10.1103/PhysRevB.66.045114.

[22] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer, *Matrix generation in isogeometric analysis by low rank tensor approximation*, in Curves and Surfaces: 8th International Conference, Paris, France, June 12-18, 2014, Revised Selected Papers, Springer, New York, NY, 2015, pp. 321–340.

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

[24] A. Mantzaflaris, F. Scholz, and I. Toulopoulos, *Low-rank space-time decoupled isogeometric analysis for parabolic problems with varying coefficients*, Comput. Methods Appl. Mech. Engrg., 19 (2018), pp. 123–136.

[25] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

[26] I. V. Oseledets, S. Dolgov, V. Kazeev, D. Savostyanov, O. Lebedeva, P. Zhlobich, T. Mach, and L. Song, *TT-Toolbox*, 2011, https://github.com/oseledets/TT-Toolbox.

[27] I. V. Oseledets and S. V. Dolgov, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739, https://doi.org/10.1137/110833142.

[28] I. V. Oseledets and E. E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88, https://doi.org/10.1016/j.laa.2009.07.024.

[29] J. W. Pearson, M. Stoll, and A. J. Wathen, *Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1126–1152, https://doi.org/10.1137/110847949.

[30] L. Piegl, *On NURBS, a survey*, IEEE Comput. Graph. Appl., 11 (1991), pp. 55–71.

[31] L. Piegl and W. Tiller, *The NURBS Book*, Springer, Berlin, 1996.

[32] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys., 326 (2011), pp. 96–192, https://doi.org/10.1016/j.aop.2010.09.012.

[33] F. Scholz, A. Mantzaflaris, and B. Jüttler, *Partial tensor decomposition for decoupling isogeometric Galerkin discretizations*, Comput. Methods Appl. Mech. Engrg., 336 (2018), pp. 485–506.

[34] M. Sorensen, D. Lathauwer, P. Comon, S. Icart, and L. Deneire, *Canonical polyadic decomposition with a columnwise orthonormal factor matrix*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1190–1213.

[35] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci. Comput., 37 (2015), pp. B1–B29, https://doi.org/10.1137/130926365.

[36] G. Strang and G. Fix, *An Analysis of the Finite Element Method*, Wellesley-Cambridge Press, Wellesley, MA, 2008.

[37] F. Tröltzsch, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, American Mathematical Society, Providence, RI, 2010.

[38] R. Vázquez, *A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs* 3.0, Comput. Math. Appl., 72 (2016), pp. 523–554.