DISSERTATION

Low-rank Tensor Methods for PDE-constrained Optimization

Submitted on September 14, 2021 by

Alexandra Bünger, M. Sc.

for the fulfillment of the requirements for the degree
doctor rerum naturalium (Dr. rer. nat.)

at Technische Universität Chemnitz, Faculty of Mathematics
under the supervision of Prof. Dr. Martin Stoll

Reviewers: Prof. Dr. Martin Stoll, Technische Universität Chemnitz
Prof. Dr. Valeria Simoncini, University of Bologna
Prof. Dr. Howard C. Elman, University of Maryland

Date of defense: December 6, 2021

Year of publication: 2021

Publication link: <https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-769374>

Abstract

Optimization problems governed by Partial Differential Equations (PDEs) arise in various applications of science and engineering. If we formulate a discretization of a PDE problem, it may become infeasible to treat the problem with conventional methods due to its size. Solving an optimization problem on top of the forward problem poses additional difficulties. Often, we can formulate the PDE problem in a more compact format by paying attention to the underlying Kronecker product structure between the space and time dimension of the discretization. When the PDE is discretized with Isogeometric Analysis we can additionally formulate a low-rank representation with Kronecker products between its individual spatial dimensions. This low-rank formulation gives rise to a fast and memory efficient assembly for the system matrices. The PDE problem represented as a sum of Kronecker products can then be interpreted as a low-rank tensor train formulation, which can be efficiently solved in a low-rank format. We illustrate this for several challenging PDE-constrained problems.

Contents

List of Figures	iii
List of Symbols	v
1 Introduction	1
2 Tensor Train Format	5
2.1 Basics for Tensor calculations	5
2.2 Tensor decompositions	10
2.3 The Alternating Minimal Energy solver	15
3 Isogeometric Analysis	19
3.1 B-splines, NURBS, and geometric representations	20
3.2 Isogeometric Galerkin method	25
3.3 Tensor Isogeometric Analysis	30
3.4 Low rank method for system matrix assembly	32
3.5 Approximation of Weight Functions	35
3.6 Example: matrix assembly with low-rank Isogeometric Analysis	39
4 PDE-constrained optimization	43
4.1 Low rank PDE-constrained optimization	47
5 Bayesian inverse problems	49
5.1 Low-rank computation for the posterior covariance	53
5.2 Example: Low-rank variances for a machine tool	54
6 A low-rank Tensor Method for PDE-constrained Optimization with Iso- geometric Analysis	61
6.1 Manuscript [15]	63

7	A low-rank matrix equation method for solving PDE-constrained optimization problems	85
7.1	Manuscript [16]	87
8	A low-rank tensor method to reconstruct sparse initial states for PDEs with Isogeometric Analysis	105
8.1	Manuscript [17]	107
9	Theses & Summary	131
	Bibliography	133

List of Figures

1.1	Exemplary 3D models	2
2.1	Tensor of order 3	6
3.1	Spline spaces with $p = 2$. In Figure 3.1b an additional knot repetition is inserted, rendering the splines non-differentiable at $\hat{x} = 1/3$	21
3.2	Two dimensional B-spline curve with basis functions from Figure 3.1b.	22
3.3	The model in Figure 1.1a is composed of $2D$ slices of different size with curved surfaces making up a volumetric domain.	39
5.1	Prior variance computed from (5.25) with $a = 0.0593$ and $b = 1.8541$. .	57
5.2	The computed spectrum for the Hessenberg matrix with 50 Arnoldi vectors	58
5.3	Computed posterior variance for the machine too using low-rank com- putations.	58

List of Symbols

Symbol	meaning	additional information
A	Tensors or matrices are denoted by upper-case letters	
D	Dimension of space $\mathbb{R}^D$ or order of a tensor	
$\mathbf{n}$	size of a tensor	$\mathbf{n} = (n_1, \dots, n_D)$
$\mathbb{W}_{\mathbf{n}}$	space of tensors of size $\mathbf{n}$	
I	index set for tensor indices	
$\mathbf{i}$	multi-index for tensor elements	$\mathbf{i} = [i_1, \dots, i_D]$
$[A]_{\mathbf{i}}$	element of A with index $\mathbf{i}$	
$A \simeq B$	the tensor B is a permutation of A	
$\langle A, B \rangle$	inner product of two tensors, matrices or vectors	
$\mathcal{O}(n)$	Landau symbol for runtime or storage estimations	
$\hat{x}$	variable on reference domain	$\hat{x} \in [0, 1]^D$
$x^{(d)}$	component of x associated with dimension d	
$\beta(\hat{x})$	single B-spline	
ξ	knot vector for B-spline basis	$\xi = \{x_1, \dots, x_{n+p+1}\}$
$\mathbb{S}_{\xi}^p$	Space spanned by B-splines with knot vector ξ and degree p	
$B(\hat{x})$	B-spline basis holding all elements of $\mathbb{S}_{\xi}^p$	$B(\hat{x}) : [0, 1]^D \rightarrow \mathbb{R}^D$
N_t	number of time steps	
I_{N_t}	identity matrix of size $\mathbb{R}^{N_t \times N_t}$	
M	mass matrix	
K	stiffness matrix	
TT	tensor train	
IGA	Isogeometric Analysis	
FEM	Finite Element method	
CAD	Computer aided design	
NURBS	non-uniform rational B-spline	
ALS	Alternating linear scheme	
AMEn	Alternating minimal energy	

1 Introduction

The modeling, analysis and optimization of physical processes is of high interest in various scientific and industrial fields. Most of these problems are modeled with Partial Differential Equations (PDEs), depending on space and time. Modeling the temperature in a mechanical system, medical processes like blood flow or the growth of a tumor, the potential impact of a car crash before its manufacturing – the applications are numerous and diverse across all areas of science and engineering. With this variety, the complexity of their solutions and the potential for improvements in their mathematical modeling and numerical computation are vast. Over the last decades many developments were made in the theoretical analysis of the properties of the problems or proving the existence of solutions, as well as in their applications by finding and enhancing discretization methods and suitable solvers (cf. [32, 29]).

Commonly, questions even surpass the plain solution of a PDE towards more qualitative problems: Finding the most efficient way to keep a steady system temperature, reconstructing the most likely origin of tumor growth [96], or locating the most meaningful sensor positions to monitor system parameters [46]. All of these applications and more call for mathematical optimization. For this, we formulate an objective and couple it with constraints such as the governing PDE and limitations on the variables. Finding the optimal solution then is usually associated with solving a large discrete linear equation system, which stems from the problem's first order optimality conditions [52]. For linear PDEs, such as the Poisson equation [1], convection diffusion equations [75] or Stokes equation [83], the optimality conditions directly result in a linear equation system. In this work, we only consider linear PDEs but many other applications call for non-linear PDE-constrained optimization. Such methods include Lagrange-Newton methods [52, 78], SQP schemes [104], or interior point methods [101, 72]. However, these solvers also require solving a linear system

Figure 1.1: Exemplary 3D models

of similar type at their core [93] to which the presented methods can potentially be applied.

In this work, we want to present a strategy developed for a certain type of PDE discretization and their efficient application to optimization problems. Let us consider the model problem of heat distribution in some structural component. This setup can be described by a (PDE), e.g.

$$\frac{\partial}{\partial t}y(x, t) = \Delta y(x, t) + u(x, t) \quad \text{in } \Omega \times (0, T], \quad (1.1)$$

$$y(x, 0) = y_0 \quad \text{in } \Omega, \quad (1.2)$$

$$y(x, t) = 0 \quad \text{on } \partial\Omega \times (0, T]. \quad (1.3)$$

Here, y is the physical state depending on time $t \in (0, T]$ and space $x \in \Omega$ and u is a source term. The structural domain Ω has to be described and discretized for the computations. Consider the PDE problem on a complex three dimensional domain such as the ones displayed in Figure 1.1. The model in Figure 1.1a is a single piece

with a smooth and curved surface. Figure 1.1b displays a full machine tool consisting of many individual parts. Solving a PDE or even an optimization problem on those three dimensional domains is a challenging task due to the computational resources required.

A very common choice to analyze a problem on these models is approximating their shape with a large number of small elements, e.g. tetrahedra, on which the problems are easily discretized via the Finite Element Method (FEM) [29]. Together with the dimension of time we arrive at 4-dimensional problems which quickly leads to a very high number of variables. Depending on the discretization method, we easily arrive at millions of degrees of freedom and their system matrices and solutions become difficult to store. Breaking or reducing this *curse of dimensionality* by e.g. decoupling the problem dimensions of space and time with a low-rank approach has recently become a common approach for memory and time efficient solution methods [7, 94, 30, 59]. In this work we will focus on a discretization method that allows for additional low-rank approximations in its spatial dimensions.

This discretization method is called Isogeometric Analysis (IGA) [22]. Here, spline functions used to construct CAD (computer-aided design) models are reused for the discretization of the solution space. This approach avoids the loss in geometric accuracy typically associated with discretizing a CAD model for analysis as in the traditional FEM. The underlying functions in IGA are globally defined and highly overlapping which makes their system matrices costly to assemble and hard to work with. This drawback can be overcome by decoupling part of the system in its spatial dimensions.

We propose a method to efficiently assemble the arising system matrices in a low-rank tensor based format. This format can subsequently be used to solve PDE based problems such as PDE-constrained optimization or uncertainty quantification. Special focus lies on the exploitation of Kronecker product structured systems that can be viewed as low-rank Tensor Trains (TT) and arise in IGA discretizations.

This leads to three distinct topics which we present in this thesis and will be discussed in the following chapters:

- Tensor decompositions and efficient computations in Section 2,

- Isogeometric Analysis and possible low-rank approximations in Section 3,
- Application to PDE-constrained optimization and Bayesian inverse problems in Sections 4 and 5.

Section 3.1 is accompanied by a computational example on the domain displayed in Figure 1.1a. In Section 5 we additionally showcase the application of a low-rank technique for an FEM discretization of the machine tool in Figure 1.1b. Following these fundamental sections, we provide the completed research papers of the author which are either published or under review.

Each paper is introduced with a preface explaining the novelty of the work and the author's contribution. First, the published work titled *A low-rank tensor method for PDE-constrained optimization with Isogeometric Analysis* [15] is presented in Section 6. In this paper, the developed low-rank method is applied to large-scale three dimensional PDE-constrained optimization problems. Second, in 7 *A low-rank matrix equation method for solving PDE-constrained optimization problems* [16] explores an alternative method for solving the previously examined saddle point system for discretizations which are not low-rank in space but only admit a low-rank structure decoupling space and time. This work was peer-reviewed and accepted for publication in June 2021. Following this work, we present the paper titled *A low-rank tensor method to reconstruct sparse initial states for PDEs with Isogeometric Analysis* [17] in Section 8. This work examines the low-rank IGA method developed previously for reconstruction of initial temperature fields in a Bayesian setting with a sparsity constraint. It was submitted for review in September 2021.

2 Tensor Train Format

We will now introduce the basic framework of tensors, their notations, and mathematical operations to facilitate their usage in the following sections. Additionally, we will give an overview of tensor formats and decompositions. A special focus in this work lies on the tensor train (TT) format which we will discuss in detail together with a popular algorithm to solve problems in TT format.

2.1 Basics for Tensor calculations

There exist multiple ways for denoting tensors and notation differs across the literature. We will loosely follow the formulations in [61]. Additional resources with thorough explanations and analysis can be found in [42, 58]. We can view a tensor as a multidimensional generalization of matrices with D indices.

Definition 2.1 (Tensor). We call an array T of dimension D and size $n_1 \times n_2 \times \dots \times n_D$,

$$T \in \mathbb{R}^{n_1 \times \dots \times n_D}, \quad (2.1)$$

a *tensor* of order D . For a more compact notation we will denote its size as $\mathbf{n} = (n_1, n_2, \dots, n_D)$.

Note that in literature $\mathbf{n}$ is often referred to as the dimension of the tensor. As we will later work with geometric domains, we reserve the term *dimension* for the dimension D , i.e., D is the dimension of the space $\mathbb{R}^D$. This often coincides with the order of the associated tensor. We will therefore call $\mathbf{n}$ the *size* of the tensor.

We use $\mathbb{W}_{\mathbf{n}}$ to denote the space holding all tensors of size $\mathbf{n}$. To describe a tensor A of size $(n_1, \dots, n_{D_1}, m_1, \dots, m_{D_2})$, resulting, e.g., from an operation on two smaller

Figure 2.1: A tensor A of order 3 with size $(3, 3, 3)$.

tensors, we will use $A \in \mathbb{W}_{(\mathbf{n}, \mathbf{m})}$ with $\mathbf{n} = (n_1, \dots, n_{D_1})$ and $\mathbf{m} = (m_1, \dots, m_{D_2})$. The tensor then has the order $D_1 + D_2$.

We will denote tensors with capital letters and refer to their elements with square brackets with subscript indices or as lowercase letters with the same subscript indices. With this notation, the tensor T has the elements

$$t_{\mathbf{i}} = [T]_{i_1, i_2, \dots, i_D}, \text{ with } 1 \leq i_d \leq n_d \text{ and } d = 1, \dots, D. \quad (2.2)$$

We will use $\mathbf{i}$ to denote the multi-index $\mathbf{i} = \{(i_1, i_2, \dots, i_D) \mid 1 \leq i_d \leq n_d, d = 1, \dots, D\}$. For example, a tensor T of order 3 with size $\mathbf{n} = (3, 3, 3)$ and elements $a_{\mathbf{i}}$ is displayed in Figure 2.1.

If a tensor $\bar{T}$ can be obtained from T by a permutation of its indices, we denote this by

$$A \simeq \bar{T}. \quad (2.3)$$

For the calculation with multi-dimensional tensors we need to define the following mathematical operations which include different products and contractions which we require later:

The **sum** of two tensors A and B from the same tensor space $\mathbb{W}_{\mathbf{n}}$ is a tensor of the same size and order which is computed by adding corresponding elements,

$$[A + B]_{\mathbf{i}} = a_{\mathbf{i}} + b_{\mathbf{i}}. \quad (2.4)$$

The **scalar multiplication** of a tensor $A \in \mathbb{W}_{\mathbf{n}}$ with a scalar $\lambda \in \mathbb{R}$ is performed elementwise, equivalent to the scalar multiplication with a matrix,

$$[\lambda A]_{\mathbf{i}} = \lambda a_{\mathbf{i}}. \quad (2.5)$$

The **inner product** $A \cdot B$ of two tensors with same size $A, B \in \mathbb{W}_{\mathbf{n}}$ of order D is the sum over the product of their entries,

$$\langle A, B \rangle = \sum_{i_1=1}^{n_1} \cdots \sum_{i_D=1}^{n_D} a_{i_1, \dots, i_D} b_{i_1, \dots, i_D}. \quad (2.6)$$

This inner product also defines the **tensor norm** of a tensor as $\|A\|^2 = \langle A, A \rangle$, which corresponds to the Frobenius norm of a matrix.

The multiplication of two tensors also works similar to a matrix-matrix multiplication. The **Contracted Product** $A \cdot B$ of a tensor $A \in \mathbb{W}_{(\mathbf{n}, \mathbf{m})}$ and $B \in \mathbb{W}_{(\mathbf{n}, \mathbf{p})}$ with $\mathbf{n} = (n_1, \dots, n_{D_1})$, $\mathbf{m} = (m_1, \dots, m_{D_2})$, and $\mathbf{p} = (p_1, \dots, p_{D_3})$ is defined as

$$[A \cdot B]_{\mathbf{j}, \mathbf{k}} = \sum_{i_1=1}^{n_1} \cdots \sum_{i_{D_1}=1}^{n_{D_1}} a_{i_1, \dots, i_{D_1}, \mathbf{j}} \cdot b_{i_1, \dots, i_{D_1}, \mathbf{k}}, \quad (2.7)$$

with multi-indices $\mathbf{j}$ and $\mathbf{k}$ as in (2.2). The resulting tensor is $(A \cdot B) \in \mathbb{W}_{(\mathbf{m}, \mathbf{p})}$.

If $\mathbf{n} = (n)$ only has dimension one, we call this product the **Single contracted product** $A \cdot B$ with $A \in \mathbb{W}_{(n, \mathbf{m})}$, $B \in \mathbb{W}_{(n, \mathbf{p})}$. Here, we only sum over one dimension

$$[A \cdot B]_{\mathbf{j}, \mathbf{k}} = \sum_{i=1}^n a_{i, \mathbf{j}} \cdot b_{i, \mathbf{k}}. \quad (2.8)$$

Remark 2.1. *The contracted product is also defined if the matching dimensions are located at different positions in the tensor, e.g. $A \in \mathbb{W}_{(\mathbf{m}, \mathbf{n})}$ and $B \in \mathbb{W}_{(\mathbf{p}, \mathbf{n})}$ or any other index pair. We will explicitly specify which contracted product we apply if the choice is not unique.*

The **Tensor Product** $A \otimes B$ of two tensors $A \in \mathbb{W}_{\mathbf{n}}$ of order D_1 and $B \in \mathbb{W}_{\mathbf{m}}$ of order D_2 results in a tensor of size $(\mathbf{n}, \mathbf{m}) = (n_1, \dots, n_{D_1}, m_1, \dots, m_{D_2})$ and order

$D_1 + D_2$ formed by the products of all element pairs with entries

$$[A \otimes B]_{\mathbf{i}\mathbf{j}} = a_{\mathbf{i}} b_{\mathbf{j}}, \quad (2.9)$$

for all multi-indices $\mathbf{i}, \mathbf{j}$. The tensor product is associative but not commutative.

The **Frobenius Product** $A : B$ of a tensor $A \in \mathbb{W}_{(\mathbf{m}, \mathbf{n})}$ of order $D_1 + D_2$ and a tensor $B \in \mathbb{W}_{\mathbf{n}}$ of order D_2 results in a D_1 -dimensional tensor of size $\mathbf{m}$ formed by the summation over the shared dimensions,

$$[A : B]_{(j_1, \dots, j_{D_1})} = \sum_{i_1=1}^{n_1} \cdots \sum_{i_{D_2}=1}^{n_{D_2}} a_{j_1, \dots, j_{D_1}, i_1, \dots, i_{D_2}} b_{i_1, \dots, i_{D_2}}. \quad (2.10)$$

The **q-Hadamard product** $A \odot_q B$ of two tensors $A \in \mathbb{W}_{(\mathbf{n}, \mathbf{m}_1)}$ with dimension D_1 and $B \in \mathbb{W}_{(\mathbf{n}, \mathbf{m}_2)}$ with dimension D_2 , where the number of elements in the first dimensions $\mathbf{n} = (n_1, \dots, n_q)$ must be equal, results in a tensor of order $D_1 + D_2 - q$ and size $(\mathbf{n}, \mathbf{m}_1, \mathbf{m}_2)$ with elements

$$[A \odot_q B]_{i_1, \dots, i_{D_1}, j_{q+1}, \dots, j_{D_2}} = a_{i_1, \dots, i_{D_1}} b_{i_1, \dots, i_q, j_{q+1}, \dots, j_{D_2}}. \quad (2.11)$$

Remark 2.2. The 2-Hadamard product $A \odot_2 B$ of two matrices $A, B \in \mathbb{R}^{n_1 \times n_2}$ is a matrix of the same dimension. Its elements are the result of element-wise multiplication of the elements in A and B . Their 1-Hadamard product would be a tensor of order 3 with elements

$$[A \odot_1 B]_{i,j,k} = a_{i,j} b_{i,k}. \quad (2.12)$$

To denote the tensor product or Hadamard product of multiple tensors, we will use the following notation,

$$A_1 \otimes \dots \otimes A_n = \bigotimes_{i=1}^n A_i, \quad A_1 \odot_q \dots \odot_q A_n = \bigodot_q^n A_i. \quad (2.13)$$

Here, for the tensor product A_i are of arbitrary size and for the q -Hadamard product we need $A_i \in \mathbb{W}_{\mathbf{n}, \mathbf{m}_i}$ where $\mathbf{n} = (n_1, \dots, n_q)$ and $\mathbf{m}_i$ are arbitrary for all $i = 1, \dots, n$.

We can reorder a tensor $T \in \mathbb{W}_{\mathbf{n}}$ into a vector v or a matrix. The **vectorization** results in a vector of length $n_1 \cdots n_D$,

$$v = \text{vec } T \quad \text{with } v(\bar{\mathbf{i}}) = [T]_{\mathbf{i}}. \quad (2.14)$$

Here $\bar{\mathbf{i}}$ denotes an index combined from $\mathbf{i}$ following the *big-endian* convention

$$\bar{\mathbf{i}} = \overline{i_1, \dots, i_D} = i_D + (i_{D-1} - 1)n_D + \cdots + (i_1 - 1)n_2 \cdots n_D. \quad (2.15)$$

For the **matricization** of $T \in \mathbb{W}_{\mathbf{n}}$ we can separate $\mathbf{n}$ into two subsets, $n_1, \dots, n_d$ and $n_{d+1}, \dots, n_D$ for any $d = 1, \dots, D$ to apply a matricization with respect to d . We denote this d -matricization with $\text{mat}_d(T) \in \mathbb{R}^{n_1 \cdots n_d \times n_{d+1} \cdots n_D}$ and it has the following entries,

$$[\text{mat}_d(T)]_{\overline{i_1, \dots, i_d} \overline{i_{d+1}, \dots, i_D}} = T_{\mathbf{i}}. \quad (2.16)$$

Here, indices $\overline{i_1, \dots, i_d}$ and $\overline{i_{d+1}, \dots, i_D}$ are each constructed according to Equation (2.15).

Remark 2.3. For two matrices $A \in \mathbb{R}^{n_1 \times n_2}$ and $B^{m_1 \times m_2}$ the matricization of $A \otimes B$ is called the **Kronecker Product**,

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \cdots & a_{1,n_2}B \\ \vdots & & \vdots \\ a_{n_1,1}B & \cdots & a_{n_1,n_2}B \end{bmatrix} \in \mathbb{R}^{n_1 m_1 \times n_2 m_2}. \quad (2.17)$$

Some useful properties of the tensor product and Hadamard product [61, 58]:

- Scalar multiplication with a scalar $s \in \mathbb{R}$ can be written with a factorization as

$$s \bigotimes_{i=1}^n A_i = \left(\prod_{i=1}^n s_i \right) \bigotimes_{i=1}^n A_i = \bigotimes_{i=1}^n s_i A_i, \quad (2.18)$$

where $s = \prod_{i=1}^n s_i$.

- Reordering factors of a tensor product leads to a permutation of the tensor,

e.g.,

$$(A \otimes B) \otimes (C \otimes D) \simeq (A \otimes C) \otimes (B \otimes D). \quad (2.19)$$

- For the 1-Hadamard product and tensor product we have the identities,

$$\bigcirc_{i=1}^n A_i \otimes \bigotimes_{i=1}^n B_i = \bigcirc_{i=1}^n (A_i \otimes B_i), \quad (2.20)$$

$$\bigotimes_{i=1}^n A_i \otimes \bigotimes_{i=1}^n B_i \simeq \bigotimes_{i=1}^n (A_i \otimes B_i), \quad (2.21)$$

$$\left(\bigcirc_{i=1}^n A_i \otimes B \right) \otimes \left(\bigcirc_{i=1}^n A_i \otimes B \right) \simeq \bigcirc_{i=1}^n (A_i \otimes B \otimes A_i \otimes B). \quad (2.22)$$

- The 2-Hadamard product of matrices $A_i \in \mathbb{R}^{n_1 \times n_2}$ and tensors $B_i \in \mathbb{W}_{(n_1, n_2, \mathbf{m})}$ with $i = 1, \dots, n$ satisfies the identity

$$\left(\bigcirc_{i=1}^n A_i \right) \odot_2 \left(\bigcirc_{i=1}^n B_i \right) = \bigcirc_{i=1}^n (A_i \odot_2 B_i). \quad (2.23)$$

2.2 Tensor decompositions

A (non-sparse) tensor of order D requires an estimate of $\mathcal{O}(n^D)$ of storage with n being the maximum of $n_1, \dots, n_D$. Thus, storage requirements for a tensor grow exponentially with their order. Storing a full tensor can quickly become infeasible which is known as the *curse of dimensionality*. Therefore, storage reducing representations and approximations are of high importance.

To find a low-rank representation (if existent) or approximation of a matrix $A \in \mathbb{R}^{n_1 \times n_2}$ it suffices to compute its singular value decomposition (SVD) [37],

$$A = U \Sigma V^T, \quad (2.24)$$

with $U \in \mathbb{R}^{n_1 \times n_1}$, $V \in \mathbb{R}^{n_2 \times n_2}$ and $\Sigma \in \mathbb{R}^{n_1 \times n_2}$. The matrices U and V have orthonormal columns and the matrix Σ holds the ordered singular values σ_i , $i = 1, \dots, \min(n_1, n_2)$ on its main diagonal. A low-rank approximation is then derived by

truncating all but the largest R singular values and corresponding rows and columns in U and V ,

$$A = U\Sigma V^T \approx U_R \Sigma_R V_R^T = \sum_{r=1}^R u_r \sigma_r v_r^T = \sum_{r=1}^R (u_r \sqrt{\sigma_r}) \otimes (v_r \sqrt{\sigma_r}). \quad (2.25)$$

We can efficiently compute a rank R approximation without computing the full SVD using Arnoldi or Lanczos iterations [37] and the best rank R approximation is always given by the leading R factors of the SVD [28]. If the rest of the singular values are zero, $\sigma_{R+1} = \dots = \sigma_{\min(n_1, n_2)} = 0$, we call R the true rank of the matrix A . For higher order tensors we can define a similar decomposition but finding it proves much more difficult.

Definition 2.2 (Rank r canonical representation). Every tensor $A \in \mathbb{W}_{(n_1, \dots, n_D)}$ can be represented as a sum of tensor products of tensors with order 1 (vectors),

$$A = \sum_{r=1}^R \bigotimes_{d=1}^D v^{(d,r)}, \quad v^{(d,r)} \in \mathbb{R}^{n_d}. \quad (2.26)$$

This is called the canonical representation of A with rank R . The smallest possible value of R is called the canonical rank of A .

When the rank R is sufficiently small this gives a memory efficient storage method as the rank R decomposition only requires $\mathcal{O}(RDn)$ of storage, breaking the curse of dimensionality. Unfortunately, computing such a canonical decomposition is not as easily done as a matrix SVD. Determining the underlying rank of a tensor is even NP-hard [45] and finding a good low-rank decomposition often proves difficult [25, 67]. As opposed to the SVD, finding the best rank $R + 1$ approximation is not equivalent to finding the best rank R approximation and adding an additional summand [57] - the whole approximation must be found at once and be recomputed to vary the rank. There exist a multitude of algorithms to compute a canonical decomposition (cf. [97, 58, 44] for extensive overviews) but they mostly lack numerical stability and efficiency. Most methods are based on the Alternating Least Squares (ALS) method [20] which solves a multi-dimensional minimization problem by considering only lower-dimensional local problems and subsequently cycling over its dimensions.

In order to find a low-rank decomposition, ALS minimizes the squared Frobenius norm

$$f(V^{(1)}, \dots, V^{(D)}) = \|A - \sum_{r=1}^R \bigotimes_{d=1}^D v^{(d,r)}\|_F^2 \quad (2.27)$$

for a given tensor $A \in \mathbb{W}_{\mathbf{n}}$ that we want to approximate and - in its general form - a fixed rank R . We collect all vectors of one dimension in the matrix $V^{(d)} \in \mathbb{R}^{n_d \times R}$ with columns $v^{(d,r)}$ for $d = 1, \dots, D$. Then, from an initial guess $V_0^{(d)}$, $d = 1, \dots, D$, in each iteration k of the ALS algorithm we successively solve D least squares problems, each minimizing one $V^{(d)}$ while fixing the rest as outlined in Algorithm 1.

Algorithm 1 Standard ALS iteration

Input: Tensor A , initial guess $V_0^{(1)}, \dots, V_0^{(D)}$, threshold ε , max. number of iterations I .

Output: $V^{(d)} = V_k^{(d)}$, $d = 1, \dots, D$

```

1: for  $k = 1, \dots, I$  do
2:   for  $d = 1, \dots, D$  do
3:     Set new initial guess  $U = V_{k-1}^{(d)} \in \mathbb{R}^{n_d \times R}$ 
4:     Update dimension  $d$  by solving

$$V_k^{(d)} = \min_U f(V_k^{(1)}, \dots, V_k^{(d-1)}, U, V_{k-1}^{(d+1)}, \dots, V_{k-1}^{(D)})$$

5:   end for
6:   if  $f(V_k^{(1)}, \dots, V_k^{(D)}) < \varepsilon$  then
7:     return
8:   end if
9: end for
```

This is repeated either until a tolerance threshold is reached or until a local saddle point is reached where there is no further improvement by continuing [40]. There are many improvements to the plain ALS algorithm, such as an orthogonalized version [90], enhanced line searches [80], or regularized regression [81] to speed up the generally slow convergence as well as some methods to make the algorithm rank adaptive such that R does not need to be chosen beforehand [39]. However, even with these improvements the ALS method for canonical decompositions often shows only slow or even no convergence [49].

A different, more stable decomposition is therefore very desirable. One reliable option is the Tucker format [58], which is especially popular in quantum chemistry [21, 64]. However, the Tucker format preserves the underlying dimension of the tensor in a *core tensor*.

Definition 2.3 (Tucker format [100]). For a tensor $A \in \mathbb{W}_{\mathbf{n}}$ the Tucker format reads

$$A = C \cdot \bigotimes_{d=1}^D A^{(d)} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \left(a_{r_1}^{(1)} \otimes \cdots \otimes a_{r_D}^{(D)} \right) \cdot [C]_{(r_1, \dots, r_D)}, \quad (2.28)$$

where the core C is a tensor of the same order as A with $C \in \mathbb{W}_{R_1, \dots, R_D}$ and $A^{(d)}$ are matrices $A^{(d)} \in \mathbb{R}^{n_d \times R_d}$ with R_d columns $a_{r_d}^{(d)}$, for $d = 1, \dots, D$.

Due to the core C it does not completely break the curse of dimensionality as the canonical composition does. As this decomposition is not unique many transformations have been suggested to *simplify* the core tensor (cf. [58]). We can compute approximations in this format in a numerically stable way via, e.g., the *higher-order SVD* [24]. There are many extensions to tensor decompositions, many combining different features of the CP and Tucker decompositions, such as the Hierarchical Tucker format [38, 43] or the Tensor Train (TT) format [67]. For the TT decomposition, stable rank adaptive strategies exist to decompose the tensor and to solve linear tensor-based equation systems, which makes it a favorable choice in a variety of applications, e.g., [9, 30, 56]. The TT decomposition was first presented in [67] and provides a computationally stable method to decompose tensors and solve tensor product based linear systems in an efficient and low-rank way [49, 68].

Definition 2.4 (Tensor Train format). The TT format for a tensor $A \in \mathbb{W}_{n_1, n_2, \dots, n_D}$ of order D is represented by its TT-cores $A^{(d)} \in \mathbb{R}^{R_{d-1} \times n_d \times R_d}$ for $d = 1, \dots, D$ with TT-ranks R_d . By convention we set $R_0 = R_D = 1$. Each TT-core can be interpreted as a parameter-dependent matrix

$$A^{(d)}(j_d) \in \mathbb{R}^{R_{d-1} \times R_d}, \quad (2.29)$$

with $j_d = 1, \dots, n_d$ and every element of A is then represented as the product

$$[A]_{j_1, \dots, j_D} = A^{(1)}(j_1) \cdots A^{(D)}(j_D). \quad (2.30)$$

We denote the TT format of a tensor A with TT cores $T^{(1)}, \dots, T^{(D)}$ as

$$T = tt\left(T^{(1)}, \dots, T^{(D)}\right). \quad (2.31)$$

The whole tensor can be written as a sum of Kronecker products,

$$A = \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^D A_{r_{d-1}, r_d}^{(d)} =: tt(A^{(1)}, \dots, A^{(D)}), \quad (2.32)$$

where the subscripts r_{d-1}, r_d are the row and column indices of the TT-core $A^{(d)}$, hence $A_{r_{d-1}, r_d}^{(d)} \in \mathbb{R}^{n_d}$. This corresponds to a canonical format from Definition 2.2 with (suboptimal) rank $R = R_1 \cdots R_D$. But opposed to finding a rank R for the canonical decomposition, finding $R_1, \dots, R_D$ for the TT decomposition can be achieved in a stable manner while only having slightly higher storage requirements [49]. There always exists a best TT approximation with bounded ranks [33, 49] and we can compute these approximations by sequences of SVDs [69, 67]. The procedure to compute a TT decomposition of a full tensor is outlined in Algorithm 2. Here, we

Algorithm 2 TT decomposition

Input: Full tensor $T \in \mathbb{W}_{\mathbf{n}}$ of order D and truncation threshold ε .

Output: Approximation U in TT format with cores $U^{(1)}, \dots, U^{(D)}$ and ranks $R_0, \dots, R_D$.

- 1: Set first and last rank, $R_0 = 1$ and $R_D = 1$
 - 2: Form first matricization $A = \text{mat}_1(T) \in \mathbb{R}^{n_1 \times n_2 \cdots n_D}$
 - 3: **for** $d = 1, \dots, D - 1$ **do**
 - 4: Compute SVD of A , i.e., $A = U \Sigma V^T$
 - 5: Truncate SVD with threshold ε to rank R_d as $A \approx U_{R_d} \Sigma_{R_d} V_{R_d}^T$
 - 6: Reshape $U \in \mathbb{R}^{R_{d-1} n_d \times R_d}$ to $U^{(d)} \in \mathbb{R}^{R_{d-1} \times n_d \times R_d}$
 - 7: Set TT-core $U^{(d)}$ and remainder $T = \Sigma_{R_d} V_{R_d}^T \in \mathbb{R}^{R_d \times n_{d+1} \cdots n_D}$
 - 8: Reshape T to *next* matricization $A = \text{mat}_2(T) \in \mathbb{R}^{R_d n_{d+1} \times n_{d+2} \cdots n_D}$
 - 9: **end for**
 - 10: Set D -th core to $U^{(D)} = T$.
-

subsequently form matricizations of which we compute a truncated SVD to get the TT cores. The ranks R_d are chosen to satisfy $\sqrt{\sigma_{R_{d+1}}^2 + \cdots \sigma_{\text{end}}^2} \leq \varepsilon$. With this we

get an accuracy of

$$\|T - U\| \leq \varepsilon \sqrt{D-1} \quad (2.33)$$

for the TT decomposition U of T [67].

2.3 The Alternating Minimal Energy solver

Let us now consider a linear equation system in tensor format. For an easy relation to a standard linear equation system $Ax = b$, with matrix $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, we will denote the solution tensor as $X \in \mathbb{W}_{\mathbf{n}}$, the right hand side as $B \in \mathbb{W}_{\mathbf{m}}$, with order D for both, and the linear system in tensor format as $A \in \mathbb{W}_{(\mathbf{m}, \mathbf{n})}$. The associated equation system reads

$$A \cdot X = B, \quad (2.34)$$

with a contracted product over $\mathbf{n}$. A convenient property of the TT format is the way we can integrate a linear operator such as A .

Definition 2.5 (TT matrix). A linear tensor operator $A \in \mathbb{W}_{(\mathbf{m}, \mathbf{n})}$ is said to be in the TT format (TT matrix) if it is represented by TT cores $A^{(d)} \in \mathbb{R}^{R_{d-1} \times n_d \times m_d \times R_d}$ with TT-ranks R_d for $d = 0, \dots, D$ and $R_0 = R_D = 1$. Each TT core can be interpreted as a matrix depending on two parameters,

$$A^{(d)}(i_d, j_d) \in \mathbb{R}^{R_{d-1} \times R_d}, \quad (2.35)$$

with $i_d = 1, \dots, n_d$, $j_d = 1, \dots, m_d$. Every element can be written as

$$[A]_{i_1, \dots, i_D, j_1, \dots, j_D} = A^{(1)}(i_1, j_1) \cdots A^{(D)}(i_D, j_D). \quad (2.36)$$

The whole TT matrix is the sum of tensor products of D matrices as

$$A = \sum_{r_0=1}^{R_0} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^D A_{r_{d-1}, r_d}^{(d)}. \quad (2.37)$$

Here, the subscripts are indices for the first and last dimension of $A^{(d)}$ and hence we

have matrices $A_{r_{d-1}, r_d}^{(d)} \in \mathbb{R}^{n_d, m_d}$.

To solve an equation system $AX = B$ in TT format, we can employ a TT variation of the ALS method in Algorithm 1 [49]. Other methods include density matrix normalizations [53, 88] or the alternating minimal energy (AMEn) scheme [27]. The AMEn method is especially efficient and allows the preservation of underlying block structures [26]. It is based on the minimization of the energy function cycling over the TT-cores [27]. This approach constructs low-dimensional systems of linear equations for each core which can then be solved with standard numerical methods. Here, we derive the method for symmetric A . Solving $AX = B$ then corresponds to the minimization of the energy function,

$$\min_X J(X) = \|X_* - X\|_A^2 = \langle X, AX \rangle - 2\langle X, B \rangle + \text{const}, \quad (2.38)$$

with the exact solution $X_* = A^{-1}B$. To find a solution for (2.38) we make an initial guess X_0 and cycle over its TT-cores where we solve a local problem to improve the current guess. For this, in iteration k all cores but the k -th are *frozen* and we minimize over

$$X_{\text{new}}^{(k)} = \min_{X_{\text{new}}^{(k)}} J \left(tt \left(X^{(1)}, \dots, X^{(k-1)}, X_{\text{new}}^{(k)}, X^{(k+1)}, \dots, X^{(D)} \right) \right). \quad (2.39)$$

The energy function $J(X)$ does not grow during updates and the solution will subsequently converge to a local minimum. The tensor-train format is linear in its cores, meaning

$$X = tt \left(X^{(1)}, \dots, X^{(k)}, \dots, X^{(D)} \right) = X_{\neq k} X_k, \quad (2.40)$$

where $X_{\neq k}$ is the Tensor-Train where the k -th core $X^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is replaced by an identity operator of the same size and X_k is a vectorization of $X^{(k)}$. With this, the energy function for the local problem becomes

$$J(X) = \langle AX_{\neq k} X_k, X_{\neq k} X_k \rangle - 2\langle B, X_{\neq k} X_k \rangle = \langle X_{\neq k}^* A X_{\neq k} X_k, X_k \rangle - 2\langle X_{\neq k}^* B, X_k \rangle. \quad (2.41)$$

The gradient of (2.41) with respect to X_k is zero when

$$(X_{\neq k}^* A X_{\neq k}) X_k = X_{\neq k}^* B. \quad (2.42)$$

Therefore, the solution to the local minimization is equal to a solution in a reduced basis. This problem is small and can be solved by standard numerical methods. For this general approach the resulting TT-ranks - and therefore the maximum accuracy - are fixed by the ranks set in the initial guess. But with some extension we can adapt the TT-ranks of the solution dynamically resulting in the AMEn method [27].

Algorithm 3 Low-rank AMEn

- 1: Input: Initial guess X with TT-ranks $r = (r_1, \dots, r_d)$, tolerance ϵ_0 and/or rank bounds $r_{\max}$, system in TT-format A, B
 - 2: **while** $\|AX - B\| > \epsilon_0$ and $k < \text{maxIter}$ **do**
 - 3: Form $A_1 = X_{\neq 1}^* A X_{\neq 1}$, $b_1 = X_{\neq 1}^* B$
 - 4: Solve $A_1 u^{(1)} = b_1$
 - 5: Optional: Do an SVD-based compression of $u^{(1)}$
 - 6: Let $\mathbf{u} = \text{tt}(u^{(1)}, X^{(2)}, \dots, X^{(D)})$ and $\mathbf{r} = A\mathbf{u} - B$
 - 7: **if** $\|\mathbf{r}\| < \epsilon_0$ **then**
 - 8: **return** $X = \mathbf{u}$
 - 9: **end if**
 - 10: Find smallest-rank residual approximation $\bar{\mathbf{r}}$ with $\|\bar{\mathbf{r}} - \mathbf{r}\| \leq \epsilon_0 \|\mathbf{r}\|$ and/or $\text{rank}(\bar{\mathbf{r}}^{(1)}) \leq r_{\max}$
 - 11: Expand basis by first TT-core $\bar{r}^{(1)}$ of $\bar{\mathbf{r}}$, $X^{(1)} = \begin{bmatrix} u^{(1)} & \bar{r}^{(1)} \end{bmatrix}$, and accordingly $X^{(2)} = \begin{bmatrix} X^{(2)} \\ 0 \end{bmatrix}$
 - 12: Form $X_1 = X^{(1)} \otimes I_{n_2, \dots, n_d}$ of size $(n_1, \dots, n_d) \times (r_1 n_2, \dots, n_d)$
 - 13: Form the $(d-1)$ -dimensional system $A_{\geq 2} X_{\geq 2} = b_{\geq 2}$ with $A_{\geq 2} = X_1^* A X_1$, $b_{\geq 2} = X_1^* b$
 - 14: **if** $d = 2$ **then**
 - 15: Solve $A_{\geq 2} X_{\geq 2} = b_{\geq 2}$ directly
 - 16: **else**
 - 17: Call AMEn recursively with $X \leftarrow \text{tt}(X^{(2)}, \dots, X^{(D)})$, $A \leftarrow A_{\geq 2}$, $B \leftarrow b_{\geq 2}$.
 - 18: **end if**
 - 19: **end while**
 - 20: Output: low-rank solution $X = \text{tt}(X^{(1)}, \dots, X^{(D)})$
-

The algorithm is outlined in Algorithm 3. This method first solves a local system as in Equation (2.42). Then we use the TT decomposition in Algorithm 2 to compute a low-rank approximation to the residual. Next, we use the first core of the low

rank residual $r^{(1)}$ to expand the components of the solution subsequently by local gradient information as in line 11 of the algorithm. Afterwards, the system dimension is reduced by one and the previous computation is repeated in a recursive way until a 2D matrix equation system is reached. Solving this equation system finalizes the outer iteration after which all steps are repeated until reaching the convergence tolerance. The method proved to be robust and has a fast convergence rate. For a detailed analysis and more information we refer the reader to [27].

Using the TT notation and the AMEn method, we can compute matrix-vector products with A or solve linear equation systems efficiently without forming the full tensor and get resulting solution vectors in a low-rank format.

3 Isogeometric Analysis

In Isogeometric Analysis, a domain is constructed using a set of B-splines or NURBS (Non-uniform rational B-splines). These discretizations often directly stem from CAD (Computer Aided Design) models [51]. The crucial step in Isogeometric Analysis lies in using the same basis functions as for the geometric design to build a solution space for a PDE problem [51].

The most common method to analyze a PDE on a geometric model is constructing an approximation to the shape with small *elements*, e.g., tetrahedra or cubes. The solution is then discretized on each element via some polynomial function. Joining them with appropriate conditions at their junctions gives rise to the Finite Element Method (FEM) most commonly used in both engineering and science applications [29]. The original models, however, are usually constructed using rational spline functions in CAD. Unfortunately, this difference in geometric representation makes the transferring task tedious and time-consuming, in particular when a design process needs multiple iterations of design and analysis. Only in 2005 a method was proposed to work directly with geometric models from CAD omitting the transferring step for FEM, namely Isogeometric Analysis (IGA) [22]. Its name stems from the fact that it works directly on the CAD geometric representation using the already existing underlying spline functions as a basis for the analysis. This results in an exact geometric representation with no accuracy loss, even for conic or circular shapes.

These high benefits come at a cost. The distinct elements in FEM ultimately lead to a very sparse equation system. However, the IGA basis functions are overlapping each other leading to a denser system and their quadrature of higher order splines is much more expensive [61]. Additionally, global basis functions with overlapping supports make the method suffer from the curse of dimensionality as memory requirements drastically increase already for three or four dimensional systems.

We will now first discuss the basics for IGA before proposing a method to overcome the problems in high dimensional system assembly with low-rank approximations for their construction as multi-dimensional tensors.

3.1 B-splines, NURBS, and geometric representations

The basis functions used for CAD design and Isogeometric Analysis are B-splines or their rationalizations NURBS (Non-uniform rational B-splines). The term *B-spline* is short for basis spline and denotes a certain type of recursively defined spline. Splines are piecewise polynomial functions of degree p which are continuously joined at so called *knots*. 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 [23].

A set of B-splines is uniquely defined by its knot vector and degree. Repetitions in its knot vector reduce the smoothness of the corresponding B-splines. Choosing a degree p , we define a vector ξ , called the open knot vector as $\xi = \{\hat{x}_1, \dots, \hat{x}_{n+p+1}\}$ with

$$0 = \hat{x}_1 = \dots = \hat{x}_{p+1} < \hat{x}_{p+2} \leq \dots \leq \hat{x}_n < \hat{x}_{n+1} = \dots = \hat{x}_{n+p+1} = 1. \quad (3.1)$$

To construct a spline space of degree p the end knots at 0 and 1 have to be repeated $p + 1$ times. For all other knots, duplicates are allowed up to multiplicity p . The parameter n determines the number of resulting B-splines $\beta_{i,p}$ with $i = 1, \dots, n$.

Definition 3.1 (B-splines). For each knot vector ξ as in (3.1), the according B-splines $\beta_{i,p}$ of degree p , with $i = 1, \dots, n$, are uniquely defined by the recursion

$$\beta_{i,0}(\hat{x}) = \begin{cases} 1 & \text{if } \hat{x}_i \leq \hat{x} < \hat{x}_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \quad (3.2)$$

$$\beta_{i,j}(\hat{x}) = \frac{\hat{x} - \hat{x}_i}{\hat{x}_{i+j} - \hat{x}_i} \beta_{i,j-1}(\hat{x}) + \frac{\hat{x}_{i+j+1} - \hat{x}}{\hat{x}_{i+j+1} - \hat{x}_{i+1}} \beta_{i+1,j-1}(\hat{x}), \quad (3.3)$$

where $j = 1, 2, \dots, p$ and $i = 1, \dots, n$.

Figure 3.1: Spline spaces with $p = 2$. In Figure 3.1b an additional knot repetition is inserted, rendering the splines non-differentiable at $\hat{x} = 1/3$.

Each resulting B-spline $\beta_{i,p}$ has the local support $[\hat{x}_i, \hat{x}_{i+p+1}]$ and in each internal knot with multiplicity k the B-splines have $p - k$ continuous derivatives. Thus, the smoothness of a B-spline can be adjusted by inserting knot repetitions as displayed in Figure 3.1.

Proposition 3.1 (B-spline derivative [79]). *The derivative of B-splines with knot vector ξ and degree p as in Definition 3.1 is given by*

$$\frac{\partial}{\partial \hat{x}} \beta_{i,p}(\hat{x}) = \frac{p}{\hat{x}_{i+p} - \hat{x}_i} \beta_{i,p-1}(\hat{x}) - \frac{p}{\hat{x}_{i+p+1} - \hat{x}_{i+1}} \beta_{i+1,p-1}(\hat{x}). \quad (3.4)$$

It has $p - k - 1$ continuous derivatives in a knot with multiplicity k .

Definition 3.2 (B-spline Space $\mathbb{S}_\xi^p$). Given a knot vector ξ and degree p , we define the spline space holding all resulting B-splines from Definition 3.1 as

$$\mathbb{S}_\xi^p := \{\beta_{1,p}, \beta_{2,p}, \dots, \beta_{n,p}\} \quad (3.5)$$

and the vector-valued function $B : [0, 1] \rightarrow \mathbb{R}^n$ holding all B-splines as

$$B(\hat{x}) := [\beta_{1,p}(\hat{x}), \beta_{2,p}(\hat{x}), \dots, \beta_{n,p}(\hat{x})]^T. \quad (3.6)$$

Due to the limited support of the basis functions $B(\hat{x})$ has a maximum of $p + 1$ non-zero entries for any $\hat{x} \in [0, 1]$. Note that until now we only looked at the general case of one-dimensional spline functions. When extending the definitions to

Figure 3.2: Two dimensional B-spline curve with basis functions from Figure 3.1b.

multi-dimensional splines, we will add a suffix to the one-dimensional variables, e.g. $B^{(d)}(\hat{x}^{(d)})$ to denote the one-dimensional splines corresponding to a certain dimension d .

To construct a B-spline curve in a D -dimensional space, the B-splines of $\mathbb{S}_{\xi}^p$ are combined with given values, called the *control points*, $C_1, \dots, C_n \in \mathbb{R}^D$.

Definition 3.3 (B-spline curve). Given a B-spline space $\mathbb{S}_{\xi}^p$ and n control points $C_i \in \mathbb{R}^D$, the curve $F : [0, 1] \rightarrow \mathbb{R}^D$ defined by

$$F(\hat{x}) := \sum_{i=1}^n C_i \beta_{i,p}(\hat{x}) \quad (3.7)$$

is called a B-spline curve of degree p .

An example for a B-spline curve with B-spline space $\mathbb{S}_{\xi_2}^2$ from Figure 3.1b with control points

$$C = \begin{bmatrix} 0 & -0.75 & 0.5 & 2 & 3 & 2 \\ 0 & -0.25 & 2 & 2 & 1 & 0.5 \end{bmatrix}^T \quad (3.8)$$

is displayed in Figure 3.2. Going on, for better readability we will denote the used spline functions only by $\beta_i(\hat{x})$ omitting their degree p when it does not matter.

By using a B-spline curve as defined in Equation (3.7), conic geometries can not be represented exactly [76]. Conic shapes can only be represented by rational functions.

Therefore, a generalization of the B-splines was developed, the so called NURBS (Non-uniform rational B-splines) [77]. The term *non-uniform* refers to the fact that NURBS usually are defined by a knot vector with non-uniformly sized knot spans.

To rationalize a B-spline into a NURBS, an additional dimension is introduced. We extend each D -dimensional Control Point $C_i = [c_{i,1}, \dots, c_{i,D}]^T$ with a new dimension with value 1 into $\hat{C}_i = [c_{i,1}, \dots, c_{i,D}, 1]^T$. This $\hat{C}$ can be interpreted as the projection of C_i into the $D + 1$ dimensional space with homogeneous coordinates. In homogeneous coordinates all points

$$\hat{C}^w := w\hat{C} = [wc_1, \dots, wc_D, w]^T, \text{ with } w \in \mathbb{R} \quad (3.9)$$

represent the same vector $C_i = [c_{i,1}, \dots, c_{i,D}]$ in $\mathbb{R}^D$. Thus, we can add a weight w_i to each control point C_i to get the function

$$\hat{F}(\hat{x}) = \sum_{i=1}^n w_i \hat{C}_i \beta_i(\hat{x}) = \begin{bmatrix} \sum_{i=1}^n w_i c_{i,1} \beta_i(\hat{x}) \\ \vdots \\ \sum_{i=1}^n w_i c_{i,D} \beta_i(\hat{x}) \\ \sum_{i=1}^n w_i \beta_i(\hat{x}) \end{bmatrix} \in \mathbb{R}^{D+1}. \quad (3.10)$$

By weighting this function with $(\sum_{i=1}^n w_i \beta_i(\hat{x}))^{-1}$ and projecting it back into the D -dimensional space, we get the motivation for the rationalization of B-splines.

Definition 3.4 (NURBS). Given a B-spline curve $F(\hat{x})$, we define a NURBS curve $N(\hat{x})$ by adding weights w_j , $j = 1, \dots, n$ to the B-spline functions and rationalizing the curve with the n weighted B-splines in the following manner,

$$N(\hat{x}) := \sum_{i=1}^n C_i \frac{\beta_i(\hat{x}) w_i}{\sum_{j=1}^n \beta_j(\hat{x}) w_j}, \quad (3.11)$$

leading to a set of rational basis functions $\frac{\beta_i(\hat{x}) w_i}{\sum_{j=1}^n \beta_j(\hat{x}) w_j}$, $i = 1, \dots, n$.

NURBS are used in a wide spectrum of computational applications, especially in CAD or CGI environments, where it became the standard tool to model any kind of required shape [34]. If all weights are set to one, the NURBS again corresponds to a simple B-spline. We will continue to use the notation for B-splines in the following

expressions and calculations but note that they could be replaced by NURBS basis functions as well.

To represent arbitrary D -dimensional geometries with B-splines or NURBS as necessary in isogeometric analysis, univariate spline spaces are combined to multivariate spaces via a tensor product, see (2.9) in Section 2.

Definition 3.5 (Multi-dimensional B-splines). Consider D different univariate spline spaces, denoted by $\mathbb{S}_{\xi_d}^{p_d}$ and univariate variables $\hat{x}^{(d)} \in \mathbb{R}$, with $d = 1, \dots, D$. We obtain a D -dimensional B-spline function as the product of D univariate basis functions $\beta_{i_d}^{(d)} \in \mathbb{S}_{\xi_d}^{p_d}$ as

$$\beta_{\mathbf{i}}(\hat{x}) := \prod_{d=1}^D \beta_{i_d}^{(d)}(\hat{x}^{(d)}), \quad (3.12)$$

with $\hat{x} \in [0, 1]^D$ and multi-index $\mathbf{i} \in I = \{(i_1, \dots, i_D) \mid i_d = 1, \dots, n_d, d = 1, \dots, D\}$.

The space holding all D -dimensional basis functions can be considered as a tensor-product spline space that we denote with $\mathbb{S}^D$,

$$\mathbb{S}^D = \mathbb{S}_{\xi_1}^{p_1} \otimes \dots \otimes \mathbb{S}_{\xi_D}^{p_D}, \quad (3.13)$$

and we define the tensor valued function $B : \mathbb{R}^D \rightarrow \mathbb{W}_{\mathbf{n}}$ with $\mathbf{n} = (n_1, \dots, n_D)$, holding all basis functions as

$$B(\hat{x}) = \bigotimes_{d=1}^D B^{(d)}(\hat{x}^{(d)}), \quad (3.14)$$

with variables $\hat{x} = (\hat{x}^{(1)}, \dots, \hat{x}^{(D)}) \in [0, 1]^D$ and univariate vector-valued functions $B^{(d)}(\hat{x}^{(d)})$ as defined in Definition 3.2. The gradient of this tensor-valued function is defined as

$$\nabla \otimes B(\hat{x}) = \begin{bmatrix} \frac{\partial}{\partial \hat{x}^{(1)}} \\ \vdots \\ \frac{\partial}{\partial \hat{x}^{(D)}} \end{bmatrix} \otimes B(\hat{x}) \in \mathbb{W}_{D, \mathbf{n}}, \quad (3.15)$$

resulting in a tensor of order $D + 1$.

Given such a basis $\mathbb{S}^D$ with $\mathbf{n} = (n_1, \dots, n_D)$ basis functions we can parametrize geometric shapes with control points similar to a curve as in Definition 3.3.

Definition 3.6 (Geometry mapping). We define a B-spline (or NURBS) geometry mapping $G : \hat{\Omega} \rightarrow \Omega$ from the D -dimensional unit cube $\hat{\Omega} := [0, 1]^D$ onto an arbitrary geometric shape $\Omega = G(\hat{\Omega}) \subset \mathbb{R}^{\tilde{D}}$ as

$$G(\hat{x}) := \sum_{\mathbf{i} \in I} C_{\mathbf{i}} \beta_{\mathbf{i}}(\hat{x}) = C : B(\hat{x}), \quad (3.16)$$

with control points $C_{\mathbf{i}} \in \mathbb{R}^D$ and multi-index $\mathbf{i} \in I = \{(i_1, \dots, i_D) \mid i_d = 1, \dots, n_d, d = 1, \dots, D\}$. Here $C \in \mathbb{W}_{(\tilde{D}, n_1, \dots, n_D)}$ denotes the tensor of order $D+1$ holding all control points and $C : B(\hat{x})$ denotes the Frobenius product of C with $B(\hat{x})$. We assume that the control points C are always chosen in a way that the geometry mapping is injective and therefore invertible.

Using this mapping, the dimensions may be chosen to fit the desired purpose, with $D \leq \tilde{D}$. We can parametrize geometric shapes of arbitrary dimensions, e.g., choosing $D = 2$ and $\tilde{D} = 3$ results in the parametrization of a 2-dimensional surface in the 3-dimensional space or choosing $D = \tilde{D} = 3$ parametrizes a 3D volume. For the scope of this thesis we will only consider $\tilde{D} = D$.

3.2 Isogeometric Galerkin method

Now that we specified the fundamentals for B-splines and their geometry mapping we now look at how to use them in a Galerkin method to discretize a PDE on the domain Ω . Assume a given spline space $B(\hat{x})$ and a geometry mapping $G : \hat{\Omega} \rightarrow \Omega$ as in Definition 3.6 with domains $\Omega \subset \mathbb{R}^D$ and $\hat{\Omega} = [0, 1]^D$.

We use the common analysis notation for PDEs and the Galerkin procedure, especially regarding function spaces, as found in, e.g., [32, 29].

Consider an elliptic boundary value problem on Ω with zero Dirichlet boundary conditions as: Find $u \in C^2(\Omega) \cap C(\bar{\Omega})$ such that u satisfies

$$[Lu](x) = f(x) \quad \text{in } \Omega, \quad (3.17)$$

$$u(x) = 0 \quad \text{on } \partial\Omega, \quad (3.18)$$

where $f \in C^0(\Omega)$ is a given source term and L is a linear elliptic partial differential operator, i.e.,

$$[Lu](x) = \sum_{i,j=1}^D \frac{\partial}{\partial x_i} \left(a_{i,j}(x) \frac{\partial}{\partial x_j} u(x) \right) + \sum_{i=1}^D b_i(x) \frac{\partial}{\partial x_i} u(x) + c(x)u(x), \quad (3.19)$$

with appropriate coefficient functions $a_{i,j}$, b_j and c . Here, $\partial\Omega$ denotes the boundary of Ω and $\bar{\Omega} = \Omega \cup \partial\Omega$ denotes its closure.

We can solve Equation (3.17) via its weak formulation that uses the function spaces $H^1(\Omega)$, which is the Hilbert space of functions in $L^2(\Omega)$ whose first derivatives also are functions in $L^2(\Omega)$, and $H_0^1(\Omega) := \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}$. H_0^1 is a dense subspace of $L^2(\Omega)$.

Definition 3.7 (Weak formulation). The weak formulation of the elliptic boundary value problem (3.17) is to find a function $u \in H_0^1(\Omega)$ that satisfies

$$a(u, v) = l(v) \quad \forall v \in H_0^1(\Omega), \quad (3.20)$$

with the bilinear form $a : H_0^1(\Omega) \times H_0^1(\Omega) \rightarrow \mathbb{R}$,

$$a(u, v) = \int_{\Omega} \sum_{i,j=1}^D \left(a_{i,j}(x) \frac{\partial}{\partial x_j} u(x) \frac{\partial}{\partial x_i} v(x) + b_i(x) \left(\frac{\partial}{\partial x_j} u(x) \right) v(x) \right) + c(x)u(x)v(x) \, dx \quad (3.21)$$

and the linear form $l : H_0^1(\Omega) \rightarrow \mathbb{R}$ with $l(v) = \int_{\Omega} f(x)v(x) \, dx$.

We call this the weak formulation as it only requires $u \in H^1(\Omega)$ instead of $u \in C^2(\Omega)$ as before. The existence and uniqueness of the solution of problem (3.20) results from Lax-Milgram's Lemma under some assumptions on the coefficients $a_{i,j}$, b_i , and c [32].

In this work we consider two special cases of the bilinear form (3.21). At first the so called **mass term** resulting from the identity linear operator $Lu := u$ as

$$a_m(u, v) = \int_{\Omega} uv \, dx. \quad (3.22)$$

The second important bilinear form results from $Lu := -\nabla \cdot (\alpha \nabla u)$ with some material parameter $\alpha \in \mathbb{R}$ as

$$\begin{aligned} a_k(u, v) &= - \int_{\Omega} \nabla \cdot (\alpha \nabla u) v \, dx, \\ &= \int_{\Omega} (\alpha \nabla u) \cdot \nabla v \, dx, \end{aligned} \quad (3.23)$$

where the second equation results from Green's identity and noticing that $v|_{\partial\Omega} = 0$. The bilinear form (3.23) is referred to as **stiffness term**.

We want to produce an approximation to the solution $u \in H_0^1(\Omega)$ with a discrete function $u_h \in V_h \subset H_0^1(\Omega)$ constructed with B-splines from the geometry mapping in Definition 3.6, $G : \hat{\Omega} = [0, 1]^D \rightarrow \Omega$. As the B-spline functions are defined on the domain $\hat{\Omega} = [0, 1]^D$, we need the inverse of the geometry mapping to form our discretization space.

The splines from the geometry mapping are now used to parametrize the discrete solution space V_h ,

$$V_h = \text{span}\{\beta_{\mathbf{i}} \circ G^{-1} : \mathbf{i} \in I_0\}, \quad (3.24)$$

$\beta_{\mathbf{i}}$ being the elements of the space B and index set I_0 as in (3.12) but omitting all multi-indices including either the first or last index in any dimension, i.e., $I_0 = \{(i_1, \dots, i_D) \mid i_d = 2, \dots, n_{d-1}, d = 1, \dots, D\}$. All remaining splines are zero on the boundary, see Definition 3.1.

The functions $u_h \in V_h$ are linear combinations of the basis functions with coefficients $u_{\mathbf{i}}$,

$$u_h = \sum_{\mathbf{i} \in I_0} u_{\mathbf{i}} (\beta_{\mathbf{i}} \circ G^{-1}). \quad (3.25)$$

The space V_h is now used for the Galerkin discretization [95] of the weak formulation (3.20) and the discrete weak formulation follows as the following task:

Find $u_h \in V_h$ such that

$$a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h. \quad (3.26)$$

Remark 3.1. *For simplicity, we only consider a problem with zero Dirichlet boundary conditions here but other boundary conditions are easily introduced with standard methods in a weak formulation [92, 5, 91]. Non-zero Dirichlet boundary conditions can also be applied in a direct way on the boundary splines we omitted in Equation (3.25) [22].*

As we do not consider any other boundary conditions, we will from now on use I instead of I_0 for notational simplicity.

Now a coordinate transformation leads to an easier formulation of the problem.

Proposition 3.2. *Let $\Omega, \hat{\Omega} \subset \mathbb{R}^D$ and $g : \hat{\Omega} \rightarrow \Omega$ and $f : \Omega \rightarrow \mathbb{R}$, then a coordinate transformation from Ω to $\hat{\Omega}$ leads to*

$$\int_{\Omega} f(x) \, dx = \int_{\hat{\Omega}} f(g(y)) |\det \nabla g(y)| \, dy. \quad (3.27)$$

This coordinate transformation leads to the discrete mass term

$$a_{m,h}(u_h, v_h) = \int_{\Omega} u_h(x) v_h(x) \, dx = \int_{\hat{\Omega}} \sum_{\mathbf{i} \in I} u_{\mathbf{i}} \beta_{\mathbf{i}}(\hat{x}) \sum_{\mathbf{j} \in I} v_{\mathbf{j}} \beta_{\mathbf{j}}(\hat{x}) \omega(\hat{x}) \, d\hat{x}, \quad (3.28)$$

with $\omega(\hat{x}) = |\det \nabla G(\hat{x})|$ and the discrete stiffness term

$$a_{k,h}(u_h, v_h) = \int_{\Omega} (\alpha \nabla u_h(x)) \cdot \nabla v_h(x) \, dx = \int_{\hat{\Omega}} (Q(\hat{x}) \sum_{\mathbf{i} \in I} u_{\mathbf{i}} \nabla \beta_{\mathbf{i}}(\hat{x})) \cdot \sum_{\mathbf{j} \in I} v_{\mathbf{j}} \nabla \beta_{\mathbf{j}}(\hat{x}) \, d\hat{x}, \quad (3.29)$$

with $Q(\hat{x}) = (\nabla G(\hat{x}))^{-1} \alpha (\nabla G(\hat{x}))^{-T} |\det \nabla G(\hat{x})|$ and the right hand side

$$l(v_h) = \int_{\Omega} f v_h \, dx = \int_{\hat{\Omega}} f(G(\hat{x})) \sum_{\mathbf{j} \in I} v_{\mathbf{j}} \beta_{\mathbf{j}}(\hat{x}) |\det \nabla G(\hat{x})| \, d\hat{x}. \quad (3.30)$$

With these discretizations, (3.26) has to hold for all $v_h \in V_h$, hence for all combinations of $v_{\mathbf{i}} \beta_{\mathbf{i}}$. However, as the basis functions $\beta_{\mathbf{i}}$ are linearly independent, it is sufficient if the equation holds for each $\beta_{\mathbf{j}}$ separately. Thus, we can rewrite the problem (3.26) as a linear algebraic equation system

$$Au = b, \quad (3.31)$$

where A takes the role of a mass matrix M or stiffness matrix K of size $A \in \mathbb{R}^{n \times n}$ where n is the number of basis functions, $n = \text{prod}(\mathbf{n}) = n_1 \cdots n_D$. The mass matrix has the elements

$$M_{i,j} = \int_{\hat{\Omega}} \beta_i(\hat{x}) \beta_j(\hat{x}) \omega(\hat{x}) d\hat{x}, \quad \text{for } i, j = 1, \dots, n \quad (3.32)$$

where i, j are some consolidation of the multi-indices $\mathbf{i}, \mathbf{j}$. The stiffness matrix K has the elements

$$K_{i,j} = \int_{\hat{\Omega}} (Q(\hat{x}) \nabla \beta_i(\hat{x})) \cdot \nabla \beta_j(\hat{x}) d\hat{x} \quad \text{for } i, j = 1, \dots, n \quad (3.33)$$

with the same ordering as for M and the right hand side has components

$$b_j = \int_{\hat{\Omega}} f(G(\hat{x})) \beta_j(\hat{x}) \omega(\hat{x}) d\hat{x}. \quad (3.34)$$

Solving this linear equation system, the vector u then holds a vectorization of the coefficients $u_{\mathbf{i}}$, $\mathbf{i} \in I$ for the basis functions of the solution $u_h(x) = \sum_{\mathbf{i} \in I} u_{\mathbf{i}} \beta_{\mathbf{i}}(G^{-1}(x))$.

To enrich the solution space, we can simply enlarge the spline space by knot insertion. A refinement step for the knot vector ξ would result in adding an additional knot in the center of each knot interval. A second popular technique is enriching the spline space by elevating the spline degrees, known as k-refinement [6, 22]. This is done by adding additional multiplicities to the end knots of ξ which leads to an even higher overlap in the basis functions.

Computing the entries for M and K needs an expensive quadrature method as the basis functions are overlapping and a multi-dimensional quadrature has to be applied. Ways to overcome or reduce this cost include reduced quadrature rules and hierarchical splines [47, 36, 86], or using Isogeometric collocation methods instead of a Galerkin method [31, 87, 3].

In this work, we want to focus on the underlying tensor structure of the matrices to reduce computational costs. The matrix-valued function $Q(\hat{x}) = (\nabla G(t)^{-1}) \alpha (\nabla G(\hat{x}))^{-T} \omega(\hat{x})$ and the scalar-valued function $\omega(\hat{x}) = |\det \nabla G(\hat{x})|$ are determined by the geometry mapping. If they can be represented or approximated as a product of univariate functions the integral computation can be done as a product of univariate integrals

which would reduce the assembly cost drastically. This notion leads the way for the low-rank tensor approximation methods for IGA [61, 89, 15].

3.3 Tensor Isogeometric Analysis

During the derivation of the mass and stiffness matrices, we did not pay attention to the tensor structure of $B(\hat{x})$. With the tensor notation from Section 2, a compact tensor representation of the mass and stiffness matrices in a multi-dimensional setting is possible. Exploiting this tensor structure leads to a compact Kronecker format of low rank univariate integrals for the mass and stiffness matrices [61].

We can integrate a tensor valued function $A(\hat{x})$ with elements $a_{\mathbf{i}}(\hat{x})$ by applying the integration to each of its entries. So

$$P = \int_{\Omega} A(\hat{x}) \, d\hat{x} \quad (3.35)$$

is a tensor with elements

$$p_{\mathbf{i}} = \int_{\Omega} a_{\mathbf{i}}(\hat{x}) \, d\hat{x}, \text{ with } \mathbf{i} \in I. \quad (3.36)$$

As the tensor integration is applied element-wise, integration and tensor operations such as a tensor or Hadamard product are interchangeable,

$$\int_{\Omega} A(\hat{x}) \otimes B(\hat{x}) \, d\hat{x} = \int_{\Omega} A(\hat{x}) \, d\hat{x} \otimes \int_{\Omega} B(\hat{x}) \, d\hat{x}, \quad (3.37)$$

$$\int_{\Omega} A(\hat{x}) \odot B(\hat{x}) \, d\hat{x} = \int_{\Omega} A(\hat{x}) \, d\hat{x} \odot \int_{\Omega} B(\hat{x}) \, d\hat{x}, \quad (3.38)$$

Consider the elements of the mass matrix (3.32). As seen in Equation (3.14), B is a tensor holding every $\beta_{\mathbf{i}}$ as elements. Thus, we can represent all combinations of elements $\beta_{\mathbf{i}}\beta_{\mathbf{j}}$ which make up the integrands of M as the tensor $B \otimes B$.

With this consideration we can write the mass term as a tensor M

$$M = \int_{\Omega} \omega B \otimes B \, d\hat{x} \in \mathbb{W}_{(\mathbf{n}, \mathbf{n})}, \quad (3.39)$$

having Equation (3.32) as elements. The stiffness term can be treated similarly. With the tensor gradient (3.15) we can write the stiffness term as a tensor K

$$K = \int_{\hat{\Omega}} [Q \cdot (\nabla \otimes B)] \cdot (\nabla \otimes B) d\hat{x} \in \mathbb{W}_{(\mathbf{n}, \mathbf{n})}. \quad (3.40)$$

As the tensor gradient is of size $(D, \mathbf{n})$, the single contracted product with $Q \in \mathbb{R}^{D \times D}$ over the first dimension leads to a tensor which is again of size $(D, \mathbf{n})$. We perform a second single contracted product with the tensor gradient to arrive at a tensor of size $(\mathbf{n}, \mathbf{n})$ whose elements are of the form (3.33).

The associated mass and stiffness matrices are obtained by matricization of the indices as the elements of the mass and stiffness tensors match the elements of the matrices.

For a more convenient representation of (3.40) with respect to ultimately splitting the integration into univariate ones, we use the identity

$$[Q \cdot (A \otimes B)] \cdot (A \otimes B) = Q \odot_2 [A \otimes B \otimes A \otimes B] : \mathbf{1}, \quad (3.41)$$

with the auxiliary matrix $\mathbf{1} \in \mathbb{R}^{D \times D}$ with all elements being equal to one [61]. With this we can rewrite (3.40) as

$$K = P : \mathbf{1}, \quad (3.42)$$

with the *pre-stiffness tensor*

$$P = \int_{\hat{\Omega}} Q \odot_2 [(\nabla \otimes B) \otimes (\nabla \otimes B)]^{\sim} d\hat{x} \in \mathbb{W}_{(D, D, \mathbf{n}, \mathbf{n})}. \quad (3.43)$$

To form the 2-Hadamard product with $Q \in \mathbb{R}^{D \times D}$ we need to permute the tensor to $[(\nabla \otimes B) \otimes (\nabla \otimes B)]^{\sim} \in \mathbb{W}_{D, D, \mathbf{n}, \mathbf{n}}$. Therefore, as described in Equation (2.3) $\sim$ denotes a permutation of the tensor product as $\nabla \otimes B \in \mathbb{W}_{D, \mathbf{n}}$ and subsequently $(\nabla \otimes B) \otimes (\nabla \otimes B) \in \mathbb{W}_{D, \mathbf{n}, D, \mathbf{n}}$.

The Frobenius product in (3.42) then is applied over the first two dimensions. The usage of the Hadamard and Frobenius products instead of the single contracted product will be more convenient for splitting the integration into univariate ones, but except for a permutation this representation yields the same result as (3.40).

In the following section, we will consider the low rank representation for the pre-stiffness tensor (3.43), which can be simply extended to the whole stiffness tensor as $K = P : \mathbf{1}$.

3.4 Low rank method for system matrix assembly

The ingredients for the mass and stiffness matrices are all univariately defined, except for the weight functions $\omega(\hat{x})$ in the mass and $Q(\hat{x})$ in the stiffness matrix. In Section 3.5 we will outline the approximation of the weight functions by univariate auxiliary functions.

For now, assume the scalar factor ω and each entry $q_{i,j}$, $i, j = 1, \dots, D$ of the matrix valued function Q are given as representations

$$\omega(\hat{x}) = \sum_{r=1}^R \prod_{d=1}^D \omega_r^{(d)}(\hat{x}^{(d)}), \quad q_{i,j}(\hat{x}) = \sum_{r=1}^R \prod_{d=1}^D (q_{i,j})_r^{(d)}(\hat{x}^{(d)}), \quad (3.44)$$

with $\omega_r^{(d)}, (q_{i,j})_r^{(d)} : [0, 1] \rightarrow \mathbb{R}$. To represent the full matrix valued function Q , we collect $(q_{i,j})_r^{(d)}$ in $R \cdot D$ matrix-valued functions $Q_r^{(d)} : [0, 1] \rightarrow \mathbb{R}^{D \times D}$. The 2-Hadamard product of two matrices corresponds to their element-wise product. Hence, we can write Q as

$$Q(\hat{x}) = \sum_{r=1}^R \bigotimes_{d=1}^D Q_r^{(d)}(\hat{x}^{(d)}). \quad (3.45)$$

These representations allow us to represent the mass and stiffness tensor using only univariate integrals. Let us first consider the mass tensor (3.39).

Theorem 3.1. *Given a decomposition of $\omega(\hat{x})$ as in (3.44), we can represent M as*

$$M = \sum_{r=1}^R \bigotimes_{d=1}^D M^{(d)}(\omega_r^{(d)}), \quad (3.46)$$

with the univariate integrals

$$M^{(d)}(\omega) = \int_0^1 \omega B^{(d)} \otimes B^{(d)} d\hat{x}^{(d)} \in \mathbb{R}^{n_d \times n_d}, \quad (3.47)$$

where n_d is the number of basis functions in $B^{(d)}$ for $d = 1, \dots, D$.

Proof. Inserting the representation of ω into the mass tensor (3.39), we get

$$\begin{aligned}
M &= \int_{\hat{\Omega}} \left(\sum_{r=1}^R \prod_{d=1}^D \omega_r^{(d)} \right) B \otimes B \, d\hat{x} \\
&= \int_{\hat{\Omega}} \left(\sum_{r=1}^R \prod_{d=1}^D \omega_r^{(d)} \right) \left(\bigotimes_{d=1}^D B^{(d)} \right) \otimes \left(\bigotimes_{d=1}^D B^{(d)} \right) d\hat{x} \\
&= \sum_{r=1}^R \int_{\hat{\Omega}} \left(\bigotimes_{d=1}^D w_r^{(d)} B^{(d)} \right) \otimes \left(\bigotimes_{d=1}^D B^{(d)} \right) d\hat{x}. \tag{3.48}
\end{aligned}$$

The last step is according to the product with a factorized scalar as in (2.18). Now we can reorder the tensor products which leaves us with a permutation of M as

$$M \simeq \sum_{r=1}^R \int_{\hat{\Omega}} \bigotimes_{d=1}^D \left(w_r^{(d)} B^{(d)} \otimes B^{(d)} \right) d\hat{x}. \tag{3.49}$$

Abusing notation between $\simeq$ and $=$ and interchanging the integration and D -factor tensor product gives us

$$M = \sum_{r=1}^R \bigotimes_{d=1}^D \int_{\hat{\Omega}} w_r^{(d)} B^{(d)} \otimes B^{(d)} d\hat{x}, \tag{3.50}$$

where each integrand is only dependent on one dimension d . $\square$

For the stiffness tensor we can proceed in a similar manner.

Theorem 3.2. *Given a decomposition of $Q(\hat{x})$ as in (3.45) we can write the stiffness tensor as a sum of tensor products,*

$$K = \sum_{r=1}^R \sum_{l,m=1}^D \bigotimes_{d=1}^D P_{l,m}^{(d)}(Q_r^{(d)}), \tag{3.51}$$

where $P_{l,m}^{(d)}(Q_r^{(d)}) \in \mathbb{R}^{n_d \times n_d}$ are the D^2 matrix slices of the univariate version of the

pre-stiffness tensor (3.43),

$$P^{(d)}(Q) = \int_0^1 Q \odot_2 [(\vartheta^{(d)} \otimes B^{(d)}) \otimes (\vartheta^{(d)} \otimes B^{(d)})]^\sim d\hat{x}^{(d)} \in \mathbb{W}_{(D,D,n_d,n_d)}, \quad (3.52)$$

where ϑ denotes the operator

$$\vartheta^{(d)} = [\vartheta_i^{(d)}]_{i=1\dots D}, \quad \text{where} \quad \vartheta_i^{(d)} = \begin{cases} \frac{\partial}{\partial \hat{x}^{(d)}} & \text{if } i = d, \\ 1 & \text{otherwise.} \end{cases} \quad (3.53)$$

Proof. First, we consider the pre-stiffness tensor (3.43) with Q as in (3.45),

$$P = \int_{\hat{\Omega}} \left(\sum_{r=1}^R \bigcirc_2^D Q_r^{(d)} \right) \odot_2 ((\nabla \otimes B) \otimes (\nabla \otimes B)) d\hat{x}. \quad (3.54)$$

The operator ϑ in Equation (3.53) can be used to rewrite the gradient operator with the 1-Hadamard product as

$$\nabla = \bigcirc_{d=1}^D \vartheta^{(d)} \in \mathbb{R}^D. \quad (3.55)$$

This gives us the pre-stiffness tensor as

$$P = \int_{\hat{\Omega}} \left(\sum_{r=1}^R \bigcirc_2^D Q_r^{(d)} \right) \odot_2 \left(\left(\bigcirc_{d=1}^D \vartheta^{(d)} \otimes \bigotimes_{d=1}^D B^{(d)} \right) \otimes \left(\bigcirc_{d=1}^D \vartheta^{(d)} \otimes \bigotimes_{d=1}^D B^{(d)} \right) \right) d\hat{x}. \quad (3.56)$$

We use the identities (2.20) and (2.22) to rewrite the gradients as

$$\left(\bigcirc_{d=1}^D \vartheta^{(d)} \otimes \bigotimes_{d=1}^D B^{(d)} \right) \otimes \left(\bigcirc_{d=1}^D \vartheta^{(d)} \otimes \bigotimes_{d=1}^D B^{(d)} \right) = \bigcirc_{d=1}^D (\vartheta^{(d)} \otimes B^{(d)} \otimes \vartheta^{(d)} \otimes B^{(d)})^\sim, \quad (3.57)$$

where the 2-Hadamard product is applied to the two dimensions gained by $\vartheta^{(d)}$. With this and the identity (2.23) the pre-stiffness tensor can be written as

$$P = \int_{\hat{\Omega}} \sum_{r=1}^R \bigcirc_2^D Q_r^{(d)} \odot_2 (\vartheta^{(d)} \otimes B^{(d)} \otimes \vartheta^{(d)} \otimes B^{(d)}) d\hat{x}. \quad (3.58)$$

We can exploit the interchangeability of sum, Hadamard product and integration to

arrive at a representation with univariate integrals as

$$P = \sum_{r=1}^R \bigcirc_{d=1}^D P^{(d)}(Q_r^{(d)}) \in \mathbb{W}_{D,D,\mathbf{n},\mathbf{n}}, \quad (3.59)$$

with the univariate pre-stiffness tensor

$$P^{(d)}(Q) = \int_0^1 Q \odot_2 [(\vartheta^{(d)} \otimes B^{(d)}) \otimes (\vartheta^{(d)} \otimes B^{(d)})]^\sim d\hat{x}^{(d)} \in \mathbb{W}_{(D,D,n_d,n_d)}. \quad (3.60)$$

To obtain the full stiffness tensor K we have to apply the Frobenius product with $\mathbf{1}$ as in (3.42). The Frobenius product collapses the first two dimensions, giving us an additional sum over $\sum_{l,m}^D$. $\square$

The mass and stiffness tensor in (3.46) and (3.51) now correspond to the TT matrix format from Definition 2.5 and we can use, e.e., the AMEn method [27] to solve linear systems featuring M and K in a compact low-rank way. By applying a matricization to $M^{(d)}$ and $P_{l,m}^{(d)}$ in the tensor product formats and applying a Kronecker product instead of the tensor product, we get the mass and stiffness matrices in Kronecker format. To obtain this Kronecker product format only univariate integration is necessary and only smaller matrices have to be stored. This is of high interest with regard to both computation time and memory requirements. What remains is the computation of the separable approximations to $\omega(\hat{x})$ and $Q(\hat{x})$.

3.5 Approximation of Weight Functions

The crucial step for arriving at the Kronecker format representation is the approximation of the weight functions $\omega(\hat{x})$ and $Q(\hat{x})$ through univariate functions. The functions $\omega(\hat{x})$ and $Q(\hat{x})$ only depend on the geometry mapping $G(\hat{x})$ but are not separable into one-dimensional factors as desired. Thus, we want to approximate them in a way that allows for the separation. We interpolate the functions with B-splines or NURBS and then compute a low-rank approximation of the coefficient tensor.

We denote the D -dimensional interpolation spline-space as in (3.13) with $\tilde{\mathbb{S}}^D$ composed from knot vectors $\tilde{\xi}_d$ and higher degrees $\tilde{p}_d > p_d$ than the basis functions that construct the geometry mapping (3.16), for $d = 1, \dots, D$. With this spline space we can interpolate the function $\omega(\hat{x})$ as

$$\omega(\hat{x}) \approx \sum_{\mathbf{j} \in J} W_{\mathbf{j}} \tilde{\beta}_{\mathbf{j}}(\hat{x}) = W : \tilde{B}(\hat{x}), \quad (3.61)$$

with multi-index set J as in (3.12) for the basis functions $\tilde{\beta}_{\mathbf{j}}(\hat{x})$ of $\tilde{\mathbb{S}}^D$ and the tensor $\tilde{B}(\hat{x})$ holding all $\tilde{\beta}_{\mathbf{j}}$. The weight tensor W is of the same size $\tilde{\mathbf{n}}$ as $\tilde{B}(\hat{x})$ and to obtain the interpolation entries we have to evaluate $\omega(\hat{x})$ at $\tilde{n} = \tilde{n}_1 \cdots \tilde{n}_D$ points. We can interpolate Q in the same manner by treating each of its D^2 entries as independent scalar valued functions.

To get a unique interpolation, we need to pay some attention to the interpolation points. One set of points always produces a unique interpolation, the Greville points [87]. Thus, they are the obvious choice for interpolation.

Definition 3.8 (Greville points, [3]). Let $\xi = (\hat{x}_1, \dots, \hat{x}_{n+p+1})$ be the knot vector and p the degree of an associated B-spline constructed from n B-spline basis functions. Then the Greville points to this B-spline are defined as

$$g_i = \frac{1}{p}(\hat{x}_{i+1} + \dots + \hat{x}_{i+p}) \text{ with } i = 1 \dots n. \quad (3.62)$$

With the n_d Greville points in each dimension of the B-spline space $\tilde{B}(\hat{x})$, we can form a grid of points for the interpolation,

$$\mathcal{G} = \left\{ [g_{i_1}^{(d)}, \dots, g_{i_{\tilde{n}_d}}^{(d)}] \mid \text{for } d = 1 \dots D \right\}, \quad (3.63)$$

holding $\tilde{n}$ different D dimensional points, where $g_1^{(d)} \dots g_{\tilde{n}_d}^{(d)}$ are the associated Greville points to the variable $\hat{x}^{(d)}$. The interpolation step can be written as

$$\text{vec}(\omega(\mathcal{G})) = \left(\tilde{B}^{(1)}(g^{(1)}) \otimes \dots \otimes \tilde{B}^{(D)}(g^{(D)}) \right) \text{vec}(W), \quad (3.64)$$

which we can solve by applying the AMEn method for tensor train from Section 2.3

by writing the tensor of interpolation values as

$$\omega(\mathcal{G}) = \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^D \text{vec}(\omega_d(r_{d-1}, :, r_d)) \quad (3.65)$$

to obtain a low-rank approximation to W in the TT-format

$$W \approx \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \bigotimes_{d=1}^D (\tilde{B}^{(d)}(g^{(d)}))^{-1} \text{vec}(w_d(r_{d-1}, :, r_d)) \quad (3.66)$$

The same procedure is applied to the entries of Q to arrive at the desired low-rank approximations of $\omega(\hat{x})$ and $Q(\hat{x})$ as in (3.44).

The accuracy of this low rank approximation depends on the spline spaces for the interpolation and the chosen ranks.

Proposition 3.3. *For a B-spline basis B with degrees $p_1, \dots, p_D$ the function $\det \nabla G$ is a D -dimensional B-spline with degrees $q_d = Dp_d - 1$ for $d = 1, \dots, D$. It can be interpolated exactly by a B-spline basis $\tilde{B}$ of degrees q_d with continuity in each knot reduced by one compared to B .*

Proof. The gradient $\nabla G(\hat{x})$ is a matrix-valued function $\nabla G : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times D}$ with elements

$$(\nabla G(\hat{x}))_{j,k} = \frac{\partial (G(\hat{x}))_k}{\partial \hat{x}_j} = \sum_{\mathbf{i} \in I} (C_{\mathbf{i}})_k \beta_{i_1}^{(1)}(\hat{x}^{(1)}) \cdots \frac{\partial \beta_{i_j}^{(j)}(\hat{x}^{(j)})}{\partial \hat{x}_j} \cdots \beta_{i_D}^{(D)}(\hat{x}^{(D)}), \quad (3.67)$$

for $j, k = 1, \dots, D$. Here $(G(\hat{x}))_k$ and $(C_{\mathbf{i}})_k$ denote the k -th entry of the geometry mapping $G(\hat{x})$ and the control point $C_{\mathbf{i}} \in \mathbb{R}^D$. The derivative of a B-spline with degree p has degree $p - 1$ and its continuity is reduced by one in every knot (see Proposition 3.1). Thus, the entry $(\nabla G(\hat{x}))_{j,k}$ has degree $p_j - 1$ for dimension j and degree p_l for $l \neq j$ for all $k = 1, \dots, D$. Following the Leibniz formula [50] the determinant of a matrix can be expressed as a sum of products, where each product has exactly one factor from each column. Thus, each summand is the product of D splines where one has degree $p_d - 1$ and the other $D - 1$ splines have degree p_d with respect to dimension d , resulting in a spline of degree $(D - 1)p_d + p_d - 1 = Dp_d - 1$ with continuity matching the continuity of the derivative. $\square$

With this $\det(\nabla G)$ can be interpolated exactly by a B-spline basis $\tilde{B}$ with degrees $\tilde{p}_i = Dp_i - 1$ where we have to adjust the continuity of $\tilde{B}$ accordingly. Let m be the multiplicity of any interior knot, then the B-splines of degree p_d are $p_d - m$ times continuously differentiable on this knot. To get the new knot vector with differentiability reduced by one, the new multiplicity m_2 has to match $q_d - m_2 = Dp_d - 1 - m_2 = p_d - m - 1$, resulting in $m_2 = (D - 1)p_d + m$. We construct the new knot vector $\tilde{\xi}$ by increasing the multiplicity of every interior knot by $(D - 1)p_d$ and the first and last knot to $q_d + 1 = Dp_d$. Note that we don't have to further decrease the differentiability on a knot if the old B-spline is already non-differentiable there. If this is the case the multiplicity is only increased by $(D - 1)p_d - 1$. With this, the spline space $\tilde{B}$ provides an exact interpolation of $\det \nabla G$. The only minor accuracy error during interpolation of $\omega = \|\det \nabla G\|$ might arise from the absolute value function.

As for the stiffness matrix, Q results in a rational function, thus an interpolation error will occur. To reduce this error to machine precision or some error tolerance, [61] suggests starting with the same knot vectors from the weight function ω and increasing the spline degree until the desired accuracy is reached.

Proposition 3.4 (Consistency error [61]). *The consistency error introduced during the matrix generation process depends on the error ε_P from the spline projection (here denoted by P) and the error ε_{T_R} from the low-rank tensor approximation (denoted by T_R) and is bounded as*

$$\|f - T_R P f\|_{inf} \leq \varepsilon_P + \varepsilon_{T_R}, \quad (3.68)$$

with f the function to be approximated.

Both errors should be controlled and monitored matching the overall accuracy of the method.

(a) Control points and two dimensional *slice* of 3D domain

(b) Stacked slices with control points in 3D domain

Figure 3.3: The model in Figure 1.1a is composed of 2D slices of different size with curved surfaces making up a volumetric domain.

3.6 Example: matrix assembly with low-rank Isogeometric Analysis

We want to illustrate the performance of the proposed method by an example. We will compare the assembly of the full system matrices with our low-rank assembly method for the mass and stiffness matrices. The domain Ω is derived from the CAD model from Figure 1.1a. This model was constructed with NURBS basis functions in the commercial CAD software *Rhino 3D* [63] with a total of 162 control points. The 3D volume was constructed stacking 2D slices of different sizes in a 3D space as seen in Figure 3.3.

For the comparison we assemble the full matrices using the GeoPDEs toolbox¹ for MATLAB . This toolbox exploits the tensor product structure of the basis functions as far as possible for a full formulation and uses a Gauss-Legendre quadrature [29] with adjustable point number for the evaluation of the weak formulations. We will use the same quadrature for both methods to avoid an impact of the potential quadrature error in our comparison. We conducted all experiments on a machine with an INTEL Core i7-4770 Quad-core processor running at 4×3400 MHz with 32 GB of RAM.

¹Version 3.2.2, open source available at <http://rafavzqz.github.io/geopdes> , documentation in [102]

Example 3.1 (Mass matrix assembly). First, let us look at the error and the time and memory savings for assembling the mass matrix M with a low-rank approximation for the weight function $\omega(\hat{x})$. We measure the relative difference δ in the Frobenius norm as

$$\delta = \frac{\|M - M_{LR}\|_2}{\|M\|_2}, \quad (3.69)$$

with the full mass matrix assembly M from geoPDEs and M_{LR} , the version assembled using our low-rank scheme.

Results with different discretization sizes and approximation thresholds ε are displayed in Table 3.1. Here the underlying B-splines have degree 2 and n denotes the total number of degrees of freedom for M . We see that we save a large amount of storage and time by using the low-rank assembly. As the weight function $\omega(\hat{x})$ is itself a B-spline, it can be interpolated exactly (up to machine precision). The errors we get stem only from the low-rank tensor approximation. The resulting ranks are adaptively chosen by AMEn (see Section 2.3) to fit the desired accuracy threshold ε . They vary slightly between discretizations but this is not surprising as we do not use the exact same spline spaces for interpolation in the different discretization levels but rather use the current discretization spline space enriched to degree $2p + 1$.

Remark 3.2. *As IGA constructs exact geometric representations, the geometry mapping G does not change for different levels of refinement. Hence, for an efficient computational scheme we can pre-compute a sufficiently good interpolation for the original model size and subsequently enrich only the solution space without having to recompute the weight approximation which can stay as small as required for the original model size.*

Example 3.2 (Stiffness matrix assembly). We will now examine the same procedure for the assembly of the stiffness matrix. Here, the weight function Q is not a B-spline itself but a rational function. Hence, an interpolation error will occur. A good size for the interpolation spline space highly depends on the geometry. We will show an example for assembling a stiffness matrix of size 756×756 with different interpolation subspaces. If a sufficiently good interpolation for Q is found we can reuse the subspace to quickly assemble a matrix with a higher number of degrees of freedom.

The subspace $\tilde{B}$ can be enriched either by enlarging the spline degrees and/or insert-

n	ε	residual	time (s)	memory (MB)	TT ranks
756	full assembly	–	1.47	0.78	
	1e-06	2.40e-09	0.30	0.02	[4, 2]
	1e-09	4.79e-12	0.33	0.02	[5, 2]
	1e-12	6.42e-15	0.35	0.02	[5, 2]
	1e-15	1.96e-15	0.82	0.03	[5, 3]
3500	full assembly	–	6.04	4.86	
	1e-06	9.62e-09	0.41	0.02	[4, 1]
	1e-09	4.39e-12	0.52	0.04	[5, 2]
	1e-12	8.46e-15	0.49	0.03	[4, 2]
	1e-15	3.43e-15	1.02	0.06	[5, 3]
20412	full assembly	–	38.27	33.71	
	1e-06	2.58e-08	1.02	0.03	[4, 1]
	1e-09	1.04e-12	1.10	0.03	[4, 1]
	1e-12	9.02e-15	1.27	0.07	[5, 2]
	1e-15	3.77e-15	3.65	0.17	[7, 4]
137564	full assembly	–	394.71	249.76	
	1e-06	1.54e-08	10.43	0.06	[4, 1]
	1e-09	9.17e-12	7.65	0.13	[5, 2]
	1e-12	9.30e-15	7.41	0.13	[5, 2]
	1e-15	5.95e-15	12.69	0.48	[7, 6]

Table 3.1: Residual to full assembly, assembly time, storage requirements, and resulting ranks for different discretization sizes and truncation thresholds.

$\tilde{p}$	τ	residual	TT ranks					
5	0	2.50e-04	[15, 3]	[15, 3]	[6, 2]	[15, 3]	[3, 3]	[5, 2]
	2	8.69e-06	[16, 2]	[16, 3]	[6, 2]	[17, 3]	[4, 3]	[5, 2]
	4	1.42e-08	[19, 3]	[20, 3]	[5, 2]	[19, 3]	[3, 2]	[4, 2]
7	0	2.29e-05	[18, 6]	[18, 5]	[6, 2]	[17, 5]	[4, 2]	[5, 2]
	2	2.59e-07	[19, 4]	[19, 3]	[6, 2]	[18, 4]	[3, 2]	[5, 2]
	4	8.06e-10	[19, 3]	[19, 3]	[5, 2]	[19, 3]	[4, 2]	[4, 2]

Table 3.2: Different subspaces for stiffness matrix approximation. We compare the resulting residual and TT ranks for Q for different spline degrees $\tilde{p}$ and numbers of additional knots τ .

ing additional knots. An examination of both strategies is displayed in Table 3.2. The original splines have degree $p = 2$, we will start with their knot vector and the degree used for the mass matrix $\tilde{p} = 2p + 1$ and enrich both. Here, τ denotes the number of additional knots inserted in each knot interval. The truncation tolerance was set to $1e-09$. We see that elevating the spline degree and inserting additional knots to enrich the spline space enhance the quality of the approximation. The column titled *residual* shows the resulting matrix residual for the stiffness matrix assembly corresponding to (3.69).

Additional examples can be found in [15], which is presented in Section 6.

4 PDE-constrained optimization

Let us now discuss the problem of PDE-constrained optimization as a model problem to apply our low-rank tensor method introduced in Section 3. Finding an optimal setup for a PDE system is of high interest in many areas of engineering and science [99]. The question of finding an optimal control to reach a desired result or to find the matching control setup for some measured data lead to the formulation of a PDE-constrained optimization problem.

In an optimal control scenario, we want to drive the state y of the PDE close to a desired state $\hat{y}$ by manipulating some control value u . The state y and control u are coupled through the underlying PDE. The model problem we consider is minimizing the following objective function on a space-time cylinder $Q = \Omega \times (0, T)$,

$$J(y, u) := \frac{1}{2} \|y - \hat{y}\|_{L^2(Q)}^2 + \frac{\alpha}{2} \|u\|_{L^2(Q)}^2 \quad (4.1)$$

$$= \frac{1}{2} \int_0^T \int_{\Omega} (y(x, t) - \hat{y}(x, t))^2 dx dt + \frac{\alpha}{2} \int_0^T \int_{\Omega} u(x, t)^2 dx dt. \quad (4.2)$$

The left term formulates the desire to have y and $\hat{y}$ as close as possible while the right term regularizes the problem. This *cost term* with the regularization parameter $\alpha > 0$ has to be introduced because the problem would be ill-posed if no cost term was associated to u [82]. Both terms are linked via a PDE, e.g.,

$$y_t(x, t) - \Delta y(x, t) = u(x, t) \quad \text{in } \Omega \times (0, T), \quad (4.3)$$

$$y(x, t) = 0 \quad \text{on } \partial\Omega \times (0, T), \quad (4.4)$$

$$y(x, 0) = y_0(x) \quad \text{in } \Omega, \quad (4.5)$$

as introduced in (1.1)–(1.3). This PDE corresponds to the heat equation with Dirichlet boundary conditions. PDE models of higher complexity are often of interest, especially in practical application. However, for our discussion we restrict ourselves to

this model problem which we will mostly discuss in the course of this work. But note that it can be extended to other PDE-constraints.

According to [99], there exists an optimal control u^* which is unique for $\alpha > 0$ if Ω is a Lipschitz domain and $y, \hat{y} \in \mathcal{L}^2(Q)$. In order to find this optimal control two distinct families of approaches exist, one can either *optimize-then-discretize* or *discretize-then-optimize*. When first optimizing, one constructs a Lagrangian function for the infinite dimensional problem and discretizes the resulting optimality conditions. This leads to an equation system with PDE, adjoint PDE and a gradient equation [99]. To reduce the system's size, one can avoid discretizing the control variable u altogether which is referred to as the variational discretization concept [48].

On the other hand, by first discretizing the optimal control problem, we can consider a discrete Lagrangian and solve its first order optimality conditions. For our model this leads to a symmetric equation system which is favorable for the choice of iterative solvers [93].

We discretize the constrained optimization problem in space using the IGA space discretization introduced in Section 3 and employ finite difference method for the temporal discretization of the objective function. We use a backwards Euler scheme [32] to discretize the PDE in time with time step size τ and number of time steps $N_t = T/\tau$ as

$$\frac{y_k - y_{k-1}}{\tau} - \Delta y_k = u_k \quad \text{for } k = 1, \dots, N_t. \quad (4.6)$$

Here, $y_k \approx y(\cdot, t_k)$ is the *time snapshot* of function $y(x, t)$ at the fixed time instance $t_k = k\tau$, respectively for u_k .

Now we can use the space discretization from Section 3 to get the fully discrete optimization system

$$\min_{\mathbf{y}, \mathbf{u}} \sum_{k=1}^{N_t} \frac{\tau}{2} \left((y_k - \hat{y}_k)^T M (y_k - \hat{y}_k) + \alpha u_k^T M u_k \right) \quad (4.7)$$

$$\text{subject to } \frac{M y_k - M y_{k-1}}{\tau} + K y_k = M u_k \quad k = 1, \dots, N_t \quad (4.8)$$

$$y_0 = y_0. \quad (4.9)$$

With abuse of notation, here $y_k, \hat{y}_k, u_k \in \mathbb{R}^n$ denote the vectors of coefficients for the discretized function in time step k , with n the number of basis functions as in (3.32). We denote $\mathbf{y} = [y_1^T, \dots, y_{N_t}^T]^T$ and accordingly $\mathbf{u}$ and $\hat{\mathbf{y}}$. Using the Kronecker product introduced in (2.17), we can write the objective function in (4.7) as

$$\frac{\tau}{2}(\mathbf{y} - \hat{\mathbf{y}})^T \mathcal{M}(\mathbf{y} - \hat{\mathbf{y}}) + \frac{\alpha\tau}{2} \mathbf{u}^T \mathcal{M} \mathbf{u} \quad (4.10)$$

with the $N_t \times N_t$ identity $\mathcal{I}_{N_t}$ and

$$\mathcal{M} = \mathcal{I}_{N_t} \otimes M = \begin{bmatrix} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \\ & & & & M \end{bmatrix}. \quad (4.11)$$

In a similar fashion we can compactly write the discrete PDE (4.8) as

$$\mathcal{K} \mathbf{y} = \tau \mathcal{M} \mathbf{u} + d \quad (4.12)$$

with

$$\mathcal{K} = \tau \mathcal{I}_{N_t} \otimes K + \mathcal{C} \otimes M = \begin{bmatrix} M + \tau K & & & & \\ -M & M + \tau K & & & \\ & -M & M + \tau K & & \\ & & \ddots & \ddots & \\ & & & -M & M + \tau K \end{bmatrix}, \quad (4.13)$$

where the auxiliary matrix $\mathcal{C}$ is derived from the backwards Euler scheme,

$$\mathcal{C} = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \quad (4.14)$$

and the vector d holding the initial conditions with y_0 as

$$d = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \otimes My_0 = \begin{bmatrix} My_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (4.15)$$

Note that other temporal discretization schemes are possible as our method only relies on the general structure of the system and the auxiliary matrices can be constructed differently to fit the desired discretization scheme.

Definition 4.1 (Discrete Lagrangian.). The function $\mathcal{L} : \mathbb{R}^{3nN_t} \rightarrow \mathbb{R}$,

$$\mathcal{L}(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{y}, \mathbf{u}) + \boldsymbol{\lambda}^T(-\mathcal{K}\mathbf{y} + \tau\mathcal{M}\mathbf{u} + d), \quad (4.16)$$

with Lagrangian multipliers $\lambda_i \in \mathbb{R}^n$, $i = 1, \dots, N_t$, collected in the vector $\boldsymbol{\lambda} = [\lambda_1^T, \dots, \lambda_{N_t}^T]^T \in \mathbb{R}^{N_t n}$, is called the Lagrangian of the optimization problem (4.7).

The optimal solution of this discrete optimization problem has to satisfy the necessary optimality conditions, so-called Karush-Kuhn-Tucker (KKT) conditions [13]. The KKT conditions state that for the optimal solution $(\mathbf{y}^*, \mathbf{u}^*)$ the Lagrangian has a saddle point,

$$\nabla \mathcal{L}(\mathbf{y}^*, \mathbf{u}^*, \boldsymbol{\lambda}^*) = 0. \quad (4.17)$$

Thus, the first order optimality conditions lead to the solution of the equation system of the gradient of $\mathcal{L}(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda})$,

$$\begin{bmatrix} \tau\mathcal{M} & 0 & -\mathcal{K}^T \\ 0 & \alpha\tau\mathcal{M} & \tau\mathcal{M} \\ -\mathcal{K} & \tau\mathcal{M} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \tau\mathcal{M}\hat{\mathbf{y}} \\ 0 \\ d \end{bmatrix}. \quad (4.18)$$

As it stems from the first order optimality conditions, we call it the KKT system. This equation system is a saddle point system which is characterized as a symmetric matrix A with block structure as

$$A = \begin{bmatrix} B & C^T \\ C & 0 \end{bmatrix}, \quad (4.19)$$

with a square block B and rectangular block C . Here we have

$$B = \begin{bmatrix} \tau\mathcal{M} & 0 \\ 0 & \alpha\tau\mathcal{M} \end{bmatrix}, \quad C = \begin{bmatrix} -\mathcal{K} & \tau\mathcal{M} \end{bmatrix} \quad (4.20)$$

Proposition 4.1. *For symmetric B and $C \neq 0$, the KKT system (4.18) is indefinite.*

Proof. Consider a vector $z = [x, y] \in \mathbb{R}^{3n}$ with $x \in \mathbb{R}^{2n}$, $y \in \mathbb{R}^n$. Then $z^T Az = x^T Bx + 2x^T Cy$, which is a linear function in y . So for any given x there exist both y for which $z^T Az < 0$ and y for which $z^T Az > 0$. $\square$

Methods to efficiently solve a large-scale problem of this type include multigrid methods [11, 12] and iterative Krylov subspace solvers, which are an efficient choice for large-scale sparse systems [93]. In our setup and many physical applications the mass matrix is symmetric. Thus, one obvious solver choice would be MINRES [70], which proves very effective especially with block preconditioning methods [9, 73, 74, 4].

However, these systems quickly get very large. Therefore, the development of low-rank techniques exploiting the system structure is receiving much deserved attention for many applications.

4.1 Low rank PDE-constrained optimization

As seen in Equations (4.11) and (4.13), the blocks in the saddle point system (4.18) are Kronecker products between the spatial and time discretization. This structure has been exploited for the construction of efficient solvers, such as a low-rank MINRES method [94] and more recently a Krylov subspace method reformulating the equation system into a matrix equation, presented in Section 7 representing our work in [16]. Additional low-rank tensor methods have already been applied, e.g., for optimization of parametric PDEs as the underlying basis in an interior point algorithm [35] or for optimal control of stochastic FEM problems [7].

For the discretization with low-rank IGA not only do we have a Kronecker product with the time discretization but the mass and stiffness matrix, M and K , are sums

of Kronecker (or tensor) products as well. Thus, we arrive at a system where each block is a sum of $(D + 1)$ -dimensional tensor products as

$$\mathcal{M} = \tau \mathcal{I}_{N_t} \otimes \sum_{r=1}^R \bigotimes_{d=1}^D M_r^{(d)}, \quad (4.21)$$

$$\mathcal{K} = \tau \mathcal{I}_{N_t} \otimes \sum_{r=1}^R \sum_{l,m=1}^D \bigotimes_{d=1}^D P_{l,m}^{(d)}(Q_r^{(d)}) + \mathcal{C} \otimes \sum_{r=1}^R \bigotimes_{d=1}^D M_r^{(d)}. \quad (4.22)$$

With this, each block corresponds to the TT matrix format from Definition 2.5 with order 4 and potentially high ranks. We can use this format to solve the KKT system for low-rank TT approximations of $\mathbf{u}$, $\mathbf{y}$, and $\boldsymbol{\lambda}$ using the AMEn method outlined in Section 2.3. This reduces the problem to smaller linear systems for individual TT blocks. To preserve numerical stability, we use a version of AMEn which preserves the underlying block structure of the KKT system and the three distinctive variables, called block TT format [26]. Recall from Equation (2.40) that the TT format is linear in its cores as for a tensor X in TT format we have

$$X = X_{\neq d} \cdot \text{vec} (X^{(d)}). \quad (4.23)$$

Now, we can apply the AMEn method from Algorithm 3 to a block structured equation system by considering a reduced block system as the local system (2.42). This reduced KKT system has the following structure,

$$\begin{bmatrix} \tau X_{\neq d}^T \mathcal{M} X_{\neq d} & 0 & X_{\neq d}^T \mathcal{K}^T X_{\neq d} \\ 0 & \tau \alpha X_{\neq d}^T \mathcal{M} X_{\neq d} & -\tau X_{\neq d}^T \mathcal{M} X_{\neq d} \\ X_{\neq d}^T \mathcal{K} X_{\neq d} & -\tau X_{\neq d}^T \mathcal{M} X_{\neq d} & 0 \end{bmatrix} \begin{bmatrix} \text{vec} (y^{(d)}) \\ \text{vec} (u^{(d)}) \\ \text{vec} (\lambda^{(d)}) \end{bmatrix} = \begin{bmatrix} \tau X_{\neq d}^T \mathcal{M} \hat{y} X_{\neq d} \\ 0 \\ 0 \end{bmatrix}. \quad (4.24)$$

This local system is now of a much smaller size and inherited the underlying structure of the original KKT system. Thus, we can apply standard methods for the solution of saddlepoint systems here to solve it.

Systems of this format arise in a variety of applications, such as a number of Bayesian inverse problems as they often can be associated with optimal control as we will outline in the following section.

5 Bayesian inverse problems

In Bayesian inverse problems, all model parameters are typically assumed to be random variables. Estimating an unobserved or uncertain variable often leads us to the computation of the maximum a posteriori (MAP) point which results in an optimal control problem [14]. This is the case for, e.g., reconstructing unknown initial values or estimating unobserved system parameters from noisy sensor data [2].

Recall the discrete PDE model in (4.12). There we have an input $\mathbf{u}$ and system state $\mathbf{y}$. Let us assume the state is only measured in $n_z \leq N_t n$ output points $\mathbf{z} = [z_1, \dots, z_{n_z}]$, coupled to $\mathbf{y}$ as

$$\mathbf{z} = C\mathbf{y} + \mathbf{e}. \quad (5.1)$$

These output points could be sensor positions or certain time instances. The variable $\mathbf{e} \in \mathbb{R}^{n_z}$ is additive noise appearing in the measurements. Let us collect the uncertain model parameters such as the initial state y_0 or input $\mathbf{u}$ in a variable p . Random variables are typically denoted by capital letters. We assume all variables to be random, so let Z , P , and E be the random variables associated with $\mathbf{z}$, p , and $\mathbf{e}$. Then we can write the parameter-to-observable map,

$$Z = f(P, E), \quad (5.2)$$

i.e. for the PDE example we would get,

$$\begin{aligned} Z = f(P, E) &= CK^{-1}(\tau MU + M_0 Y_0) + E \\ &= FP + E \end{aligned} \quad (5.3)$$

with the model parameters $p = [\mathbf{u}, y_0] \in \mathbb{R}^{n_p}$ and their random variables $P = [U, Y_0]$.

To quantify the probability of the random variables ranges we define the following probability density functions (pdf's):

- $\pi_{\text{noise}} : \mathbb{R}^{n_z} \rightarrow \mathbb{R}$, modeling the error and observation noise E ,
- $\pi_{\text{prior}} : \mathbb{R}^{n_p} \rightarrow \mathbb{R}$, describing additional information about the model parameters P ,
- and the likelihood function $\pi(z_{\text{obs}}|p)$, describing the relation between the (realization of) model parameters p and realization of the observables $\mathbf{z}$, i.e. the measurements, denoted as z_{obs} .

To link the probability of an outcome, which we denote with π_{post} , to our prior knowledge we can use a popular identity that plays a key role in Bayesian inference, the Bayes' theorem:

Theorem 5.1 (Bayes' Theorem [60]). *Given probability densities $\pi(p)$ and $\pi(z_{\text{obs}})$ and their conditional probability $\pi(z_{\text{obs}}|p)$ we can write their inverse conditional probability, or likelihood, as*

$$\pi(p|z) = \frac{\pi(p)\pi(z|p)}{\pi(z)}. \quad (5.4)$$

With Bayes' theorem we can combine the prior probability density and the likelihood function to form the posterior probability density $\pi_{\text{post}} : \mathbb{R}^{n_p} \rightarrow \mathbb{R}$ as

$$\pi_{\text{post}}(p) = \pi(p|z_{\text{obs}}) = \frac{\pi_{\text{prior}}(p)\pi(z_{\text{obs}}|p)}{\pi(z_{\text{obs}})} \propto \pi_{\text{prior}}(p)\pi(z_{\text{obs}}|p). \quad (5.5)$$

Here, $\propto$ denotes proportionality. In our model problem the realized observation z_{obs} is fixed and we want to consider the impact of its observation. Therefore, $\pi(z_{\text{obs}})$ is fixed and the posterior probability density is proportional to the numerator [60], that is the prior probability times the likelihood.

Following [55], as we only assume additive noise in our model (5.3),

$$Z = f(P) + E, \quad (5.6)$$

we can assume that E and P are mutually independent, that is

$$\pi(e|p) = \pi(e) = \pi_{\text{noise}}. \quad (5.7)$$

Then Y conditioned on P is distributed like E and we can write the likelihood function as

$$\pi(z_{\text{obs}}|p) = \pi_{\text{noise}}(z_{\text{obs}} - f(p)). \quad (5.8)$$

With this we get the final version of the posterior density (5.5) for the PDE setup as

$$\pi_{\text{post}}(p) \propto \pi_{\text{prior}}(p) \pi_{\text{noise}}(z_{\text{obs}} - f(p)). \quad (5.9)$$

This posterior density expresses the probability of the unknown parameters under the observed outcome z_{obs} under the assumptions of our prior belief. In a Bayesian framework, the posterior density is the solution of the *inverse problem* [19]. From this, we can follow two distinct questions; either computing the posterior density itself or extracting useful information from it such as computing the most likely parameters for p under given measurements z_{obs} .

We will now follow a setup described in [10] to show the relation to an optimal control problem. We assume that both the prior and noise probability densities are Gaussian,

$$\pi_{\text{prior}}(p) \propto \exp \left(-\frac{1}{2} (p - \bar{p}_{\text{prior}})^T \Gamma_{\text{prior}}^{-1} (p - \bar{p}_{\text{prior}}) \right) \quad (5.10)$$

$$\pi_{\text{noise}}(e) \propto \exp \left(-\frac{1}{2} e^T \Gamma_{\text{noise}}^{-1} e \right). \quad (5.11)$$

Here, $\bar{p}_{\text{prior}} \in \mathbb{R}^{n_p}$ is the mean of the model parameter prior pdf, the noise pdf is assumed to have zero mean, $\Gamma_{\text{prior}} \in \mathbb{R}^{n_p \times n_p}$ is the covariance matrix of the prior pdf, and $\Gamma_{\text{noise}} \in \mathbb{R}^{n_z \times n_z}$ is the covariance matrix of the noise pdf. We assume all of the above to be known or estimated. With these Gaussian distributed densities we can write the posterior probability density (5.9) as

$$\pi_{\text{post}}(p) \propto \exp \left(-\frac{1}{2} (p - \bar{p}_{\text{prior}})^T \Gamma_{\text{prior}}^{-1} (p - \bar{p}_{\text{prior}}) - \frac{1}{2} e^T \Gamma_{\text{noise}}^{-1} e \right) \quad (5.12)$$

$$= \exp \left(-\frac{1}{2} \|p - \bar{p}_{\text{prior}}\|_{\Gamma_{\text{prior}}^{-1}}^2 - \frac{1}{2} \|e\|_{\Gamma_{\text{noise}}^{-1}}^2 \right). \quad (5.13)$$

With this the posterior probability density $\pi_{\text{post}}(p)$ is not necessarily Gaussian as the function $f(p)$ in general might be non-linear. But as we have a linear function $f(P) = FP$ in (5.3) we get a Gaussian $\pi_{\text{post}}(p)$ [60] as

$$\pi_{\text{post}}(p) \propto \exp \left(-\frac{1}{2} \|p - \bar{p}\|_{\Gamma_{\text{prior}}^{-1}}^2 - \frac{1}{2} \|z_{\text{obs}} - Fp\|_{\Gamma_{\text{noise}}^{-1}}^2 \right). \quad (5.14)$$

An important quantity to examine from here is the *maximum a posteriori* (MAP) point [19] which gives us the mean $\bar{p}_{\text{post}}$ as

$$\bar{p}_{\text{post}} = \arg \max_p \pi_{\text{post}}(p) \quad (5.15)$$

$$= \arg \min_p \left(\frac{1}{2} \|p - \bar{p}\|_{\Gamma_{\text{prior}}^{-1}}^2 + \frac{1}{2} \|z_{\text{obs}} - Fp\|_{\Gamma_{\text{noise}}^{-1}}^2 \right). \quad (5.16)$$

One common application where we need to compute the MAP point is the reconstruction of an initial state or other model parameters under certain assumption, which are incorporated into Γ_{prior} and Γ_{noise} [18]. This minimization resembles the structure from PDE-constrained optimization problems such as (4.1). Therefore, it can be treated as a constrained optimization problem by substituting $z - Fp = u$ and setting this equation as a new constraint. For a PDE governed system the parameter-to-observable map contains the PDE solution. By separating it from the objective function we avoid the need to fully compute it which might be infeasible due to its size. Here, low-rank methods for PDE-constrained optimization could be applied as done in [94, 16, 8, 9, 35]. The only apparent difference lies in the matrices Γ_{noise} and Γ_{prior} . However, for the classical PDE constrained problems they are mass matrices or involve mass matrices yielding a very similar structure to classical PDE-constrained optimization [10]. We show the application of our low-rank scheme for IGA to a similar problem of reconstructing a sparse initial state for a PDE using sparsity inducing priors in Section 8.

A more challenging question lies in the estimation of variances and covariances associated with π_{post} , which we will outline in the following chapter and illustrate by an example.

5.1 Low-rank computation for the posterior covariance

Assume we are not only interested in inferring e.g. an initial state y_0 from observations of the output z_{obs} but we seek to quantify the uncertainty in this estimate. Then instead of computing the MAP we need to estimate its covariance Γ_{post} . This quantity is given by the inverse of the Hessian of $\pi_{\text{post}}(p)$ differentiated with respect to p , i.e.,

$$\Gamma_{\text{post}} = (F^T \Gamma_{\text{noise}}^{-1} F + \Gamma_{\text{prior}}^{-1})^{-1}. \quad (5.17)$$

This quantity gives us a measure for the posterior covariance in terms of the noise and prior covariances and depending on the linear part of the parameter-to-observable map, which in turn depends on the observed quantities we choose.

The posterior covariance (5.17) itself is usually infeasible to work with, as, in our setting, it is a large and dense matrix of size $\Gamma_{\text{post}} \in \mathbb{R}^{n_p \times n_p}$. Of particular interest to use are the posterior variances, i.e., the diagonal entries of Γ_{post} , which are given by

$$\text{var}_i = e_i^T \Gamma_{\text{post}} e_i. \quad (5.18)$$

To compute these quantities, a storage reduction in the computation of Γ_{post} is crucial. Fortunately, a low-rank approximation method has emerged, which delivers reliable results while significantly reducing the computational requirements [10, 14], which we want to present now.

The posterior covariance (5.17) can be approximated efficiently as explained in the following. For convenience, we reformulate it using the symmetric positive matrix square root $\Gamma_{\text{prior}}^{1/2}$ as

$$\Gamma_{\text{post}} = \Gamma_{\text{prior}}^{1/2} \underbrace{(\Gamma_{\text{prior}}^{1/2} F^T \Gamma_{\text{noise}}^{-1} F \Gamma_{\text{prior}}^{1/2} + I)}_{\mathcal{H}_{\text{misfit}}}^{-1} \Gamma_{\text{prior}}^{1/2}. \quad (5.19)$$

We follow the suggestion in [14] and calculate a low-rank approximation of the transformed data misfit Hessian,

$$\mathcal{H}_{\text{misfit}} = \Gamma_{\text{prior}}^{1/2} F^T \Gamma_{\text{noise}}^{-1} F \Gamma_{\text{prior}}^{1/2} \approx V_R \Lambda_R V_R^T. \quad (5.20)$$

Here R is the approximation rank, and V_R is the matrix with orthogonal columns of the leading eigenvectors, with the corresponding spectral values λ_i , $i = 1, \dots, R$ collected in the diagonal matrix Λ_R .

If we have such a decomposition, we can calculate an approximate inverse of $\mathcal{H}_{\text{misfit}} + I$ using the Sherman-Morrison-Woodbury formula, which yields

$$(\mathcal{H}_{\text{misfit}} + I)^{-1} \approx I - V_R \tilde{\Lambda}_R V_R^T, \quad (5.21)$$

with $\tilde{\Lambda}_R = \text{diag}(\frac{\lambda_1}{\lambda_1+1}, \dots, \frac{\lambda_R}{\lambda_R+1})$. We can compute an approximate spectral decomposition (5.20) of $\mathcal{H}_{\text{misfit}}$, e.g., via the Arnoldi method [98]. The rank R can be adjusted depending on the available computing time and memory resources. Finally, the posterior covariance matrix Γ_{post} and its diagonal entries can be evaluated

$$e_i^T \Gamma_{\text{post}} e_i \approx e_i^T \Gamma_{\text{prior}}^{1/2} (I - V_R \tilde{\Lambda}_R V_R^T) \Gamma_{\text{prior}}^{1/2} e_i = (\Gamma_{\text{prior}})_{ii} - e_i^T \Gamma_{\text{prior}}^{1/2} V_R \tilde{\Lambda}_R V_R^T \Gamma_{\text{prior}}^{1/2} e_i. \quad (5.22)$$

The requirement to form the square root of the prior covariance may appear prohibitive for high-dimensional problems. However, in practice we can work with distributions for which the square root is explicitly available.

5.2 Example: Low-rank variances for a machine tool

We want to illustrate the presented method by a numerical example. For the thermal-energetic layout of a machine tool within the DFG Collaborative Research Center *SFB/Transregio 96* a method for the efficient computation of posterior covariances for the propagation of uncertain initial conditions was of high interest. For a large machine tool such as displayed in Figure 1.1b the full computation of the posterior covariance matrix is infeasible. Therefore, we apply the approach from the previous Section 5.1 using low-rank TT calculations with AMEn.

Recall the parameter-to-observable map of the discrete PDE (5.3). Let us assume an uncertain initial state y_0 and no input u . Then we have

$$F = C\mathcal{K}^{-1}(M_0 y_0). \quad (5.23)$$

Inserting this into the equation for the data misfit Hessian (5.20), we obtain

$$\mathcal{H}_{\text{misfit}} = \Gamma_{\text{prior}}^{1/2} (M_0 \mathcal{K}^{-T} C^T) \Gamma_{\text{noise}}^{-1} (C \mathcal{K}^{-1} M_0) \Gamma_{\text{prior}}^{1/2}. \quad (5.24)$$

For the measurement noise covariance we use a Gaussian with distribution $\mathcal{N}(\bar{x}, \sigma_{\text{noise}}^2 I)$ and standard deviation σ_{noise} . For the prior, we employ a Laplacian-like covariance structure, which is in accordance with the infinite-dimensional nature of the initial state (cf. [14]). This prior covariance matrix reads as follows

$$\Gamma_{\text{prior}} = (M^{-1}(aK + bM))^{-2}, \quad (5.25)$$

where a and b are scalar parameters chosen to fit a previously determined correlation. This gives us the final equation for the data misfit Hessian,

$$\mathcal{H} = \frac{1}{\sigma_{\text{noise}}^2} (aK + bM)^{-1} M (M_0 \mathcal{K}^{-T} C^T) (C \mathcal{K}^{-1} M_0) (aK + bM)^{-1} M, \quad (5.26)$$

with $\Gamma_{\text{noise}} = \sigma_{\text{noise}}^2 I$ and Γ_{prior} as in (5.25) which gives us $\Gamma_{\text{prior}}^{1/2} = (aK + bM)^{-1} M$.

As we want to compute a spectral decomposition of $\mathcal{H}_{\text{misfit}} \approx V_R \Lambda_R V_R^T$ with $V = [v_1, \dots, v_R]$, we can use an approximation method that preserves the spectral features by projecting $\mathcal{H}_{\text{misfit}}$ onto a smaller suitable subspace. For this we use an Arnoldi algorithm tailored to work with tensor product structured data, previously proposed in [10].

Let us first review the standard Arnoldi algorithm [37]. For the matrix $\mathcal{H}_{\text{misfit}}$, the Arnoldi algorithm subsequently computes a matrix $Q \in \mathbb{R}^{p \times p}$ whose columns form an orthonormal basis for the p -dimensional Krylov subspace

$$K_p(\mathcal{H}_{\text{misfit}}, q) = \text{span}\{q, \mathcal{H}_{\text{misfit}} q, \mathcal{H}_{\text{misfit}}^2 q, \dots, \mathcal{H}_{\text{misfit}}^{p-1} q\} \quad (5.27)$$

based on an initial vector q . Q is then used to build an upper Hessenberg matrix as $H = Q^* \mathcal{H}_{\text{misfit}} Q \in \mathbb{R}^{p \times p}$. This can be interpreted as a projection of $\mathcal{H}_{\text{misfit}}$ onto the subspace $K_p(\mathcal{H}_{\text{misfit}}, q)$ and we can expect Q to be a good approximation to the p largest eigenvectors of $\mathcal{H}_{\text{misfit}}$. The iteration to compute this approximation is outlined in Algorithm 4.

The resulting Hessenberg matrix H possesses similar spectral features to $\mathcal{H}_{\text{misfit}}$ and we get the approximation

$$\mathcal{H}_{\text{misfit}} \approx QHQ^* \approx QV_H\Lambda V_H^T Q^*, \quad (5.28)$$

with spectral decomposition of H as $H \approx V_H\Lambda V_H^T$ [37].

Algorithm 4 Arnoldi iteration for $\mathcal{H}_{\text{mis}}$

Input: Matrix $\mathcal{H}_{\text{misfit}}$ and starting vector q_0

```

1: for  $k = 1, \dots, p$  do
2:    $q_k = \mathcal{H}_{\text{misfit}} q_{k-1}$ 

3:   for  $j = 1, \dots, k$  do
4:      $h_{j,k} = q_j^* q_k$ 
5:      $q_k = q_k - h_{j,k} q_j$ 
6:   end for
7:    $h_{k+1,k} = \|q_k\|$ 
8:   if  $h_{k+1,k} > \epsilon$  then
9:      $q_k = \frac{q_k}{h_{k+1,k}}$ 
10:  else
11:    return
12:  end if
13: end for
14: Output: Arnoldi vectors  $q_k$  and Hessenberg matrix  $H$ 
```

As our discrete PDE (4.8) is in tensor product format we can additionally reduce the computational requirements by using low-rank TT computations as outlined in Section 2.3. The model for the machine tool is discretized using FEM. Thus, the low-rank computation only affects two dimensions. We perform the matrix-vector product in line 2 of Algorithm 4 using the TT format. Here, we apply $\mathcal{K}^{-1}$ using the AMEn solver in Algorithm 3, getting a low-rank solution for q_j as in Equation (5.29). With this, we can assume each arising vector q_j to be well approximated as

$$q_j = \text{vec}(W_{j,1}, W_{j,2}^T), \quad (5.29)$$

with $W_{j,1} \in \mathbb{R}^{n,r_j}$, $W_{j,2} \in \mathbb{R}^{N_t,r_j}$ and $r \ll \min(n, N_t)$. Now, we can compute the

Figure 5.1: Prior variance computed from (5.25) with $a = 0.0593$ and $b = 1.8541$.

orthogonalization in line 4–5 of Algorithm 4 in a more efficient way using low-rank tensor train calculations as described in Section 2. Thus, the computational complexity stays far below that of the full Arnoldi algorithm. Depending on the required number of Arnoldi vectors one can also implement a restarted version [55] in low-rank format to further reduce the computational cost for computing a high number of Arnoldi vectors.

Experimental setup. We model a machine tool discretized with a total of $n = 74516$ spatial nodes and $n_z = 17$ outputs over a total of $N_t = 120$ time steps and end time $T = 120$. The parameters for the noise and prior are set to $\sigma_{\text{noise}} = 0.1$, $a = 0.0593$, and $b = 1.8541$. This gives us a prior variance as displayed in Figure 5.1.

In this example the number of outputs n_z is much smaller than the number of total states. In general, if we wanted to pre-compute F an adjoint approach would be much more efficient here [46] computing the matrix-valued final value problem,

$$\mathcal{M}^T \frac{\partial}{\partial t} S(t) = \mathcal{K} S(t), \quad (5.30)$$

$$\mathcal{M}^T S(T) = -C^T, \quad (5.31)$$

with sensitivity factors $S \in \mathbb{R}^{n \times n_z}$. Solving this ODE gives us the parameter-to-observable map F as

$$F = - \begin{bmatrix} S(T - t_1)^T \mathcal{M} \\ \vdots \\ S(T - t_{N_t})^T \mathcal{M} \end{bmatrix} \in \mathbb{R}^{N_t n_z \times n}. \quad (5.32)$$

In the course of the project the sensitivities were computed in this manner using an implicit Euler method and the parameters stated above. Even for this small number of outputs, $nn_z = 17$, the sensitivities arising in this approach require 1.2GB of disc space. We computed the first 50 Eigenvalues of $\mathcal{H}_{\text{mis}}$ using the full formulation of F and subsequently computed the posterior variances of our model as described in Equation (5.22). We adduce these results for a comparison with the presented low-rank approach. For the low-rank computations, we set a truncation tolerance of $1e-08$ and a stopping tolerance for AMEn of $1e-06$. The overall relative error between both computed posterior variances is $9.51e - 04$. We display some information about

Figure 5.2: The computed spectrum for the Hessenberg matrix with 50 Arnoldi vectors

Figure 5.3: Computed posterior variance for the machine too using low-rank computations.

storage and time consumption in Table 5.1. Here, p is the number of eigenvalues computed, respectively the number of Arnoldi steps for the low-rank method. Concerning computation time, both methods perform comparably when computing the

	p	total time (s)	storage of V_R
full sensitivity approach	50	8124	54.9 MB
low-rank approach	50	8362	28.8 MB
	20	5606	11.9 MB

Table 5.1: Comparison of both approaches for the posterior variance computation.

first 50 singular values. Taking a look at the resulting spectrum for the low-rank approach, displayed in Figure 5.2, we see that the eigenvalues drop sharply after index 16. Thus, the smaller number of 20 Arnoldi iterations would have sufficed. Concerning the storage requirements, not only needs the full approach a tremendous amount of storage for the sensitivities but also storing the eigenvectors for $\mathcal{H}_{\text{mis}}$ requires larger resources as they are stored in a low-rank tensor train format for the low-rank method.

6 A low-rank Tensor Method for PDE-constrained Optimization with Isogeometric Analysis

This paper is published as *A. Büniger, S. Dolgov, and M. Stoll, A Low-rank tensor method for PDE-constrained optimization with isogeometric analysis* [15]. First Published in SIAM Journal of Scientific Computing in 42 (2020), pp. A140 – A161, published by the Society for Industrial and Applied Mathematics (SIAM).

Novelty of this work. We developed a complete algorithmic scheme from setting up the low-rank discretization of IGA systems to solving the optimization problem in a low-rank Tensor Train format. The resulting system is decoupled in time and space. The use of Tensor Train approximations for the system setup was a new idea which has not been explored before. Using the resulting low-rank structure for further computations without assembly of the full system was a novel approach and the combination with a tailored Alternating Minimal Energy (AMEn) solver provides an efficient scheme to reduce memory requirements and computation time to a minimum.

Results from the paper. We were able to reduce the computation time and storage requirements for the system matrix assembly by multiple orders of magnitude while maintaining a high level of accuracy for the PDE solution which stays around the order of magnitude of the respective truncation tolerances. The large-scale PDE-constrained optimization problems examined in this work can be efficiently solved with the new scheme. Even for domains with high ranks the method provided reliable results with high memory savings.

Contribution of the author. The author of this dissertation provided about 80% of the work within the paper. This includes the theoretical discussion, development

of the algorithmic scheme, its implementation, and the experimental section. The AMEn block solver and the local solver were developed by the co-authors.

Competing works. Since the publication of this work a lot of focus has been laying on the development of low-rank methods and matrix free discretization for Isogeometric Analysis. Isogeometric collocation provide an alternative to standard Galerkin approaches and can significantly reduce assembly time [3, 31]. In addition matrix free methods have been proposed for high degree spline spaces [85]. In 2020 a matrix-free method was developed [65] which utilizes the same underlying idea of separating the discretization in its dimensions. Additionally, some works focused on different tensor methods for the assembly of the system matrices [61, 71], notably without taking into consideration later exploitation of the gained structures or by separating the tensor only partially [62].

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

$$\begin{aligned}
 (1.1) \quad & \min_{y,u} \quad \frac{1}{2} \int_0^T \int_{\Omega} (y - \hat{y})^2 \, dx \, dt + \frac{\alpha}{2} \int_0^T \int_{\Omega} u^2 \, dx \, dt \\
 (1.2) \quad & \text{such that (s.t.)} \quad y_t - \Delta y = u \quad \text{in } (0, T) \times \Omega, \\
 (1.3) \quad & y = 0 \quad \text{on } (0, T) \times \partial\Omega, \\
 (1.4) \quad & y = y_0 \quad \text{on } \Omega \text{ for } t = 0,
 \end{aligned}$$

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

$$(2.1) \quad 0 = \xi_1 = \dots = \xi_{p+1} < \xi_{p+2} \leq \dots \leq \xi_n < \xi_{n+1} = \dots = \xi_{n+p+1} = 1,$$

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

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

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

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

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

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

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

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 $d = 1, \dots, D$. We use $\hat{x}^{(d)} \in [0, 1]$ to denote the one-dimensional variables for each spline space and $\beta_1^{(d)}, \dots, \beta_{n_d}^{(d)}$ to denote its basis functions.

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

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

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

$$(2.7) \quad \mathbf{I} = \{(i_1, \dots, i_D) \mid i_d \in \{1, \dots, n_d\} \text{ for all } d = 1, \dots, D\}.$$

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

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

with control points $C_{\mathbf{i}} \in \mathbb{R}^D$. Here $C \in \mathbb{R}^{D \times n_1 \times \dots \times n_D}$ and $B(\hat{x}) \in \mathbb{R}^{n_1 \times \dots \times n_D}$ 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,

$$(2.9) \quad a_m(u, v) = \langle u, v \rangle_2 = \int_{\Omega} uv \, dx,$$

$$(2.10) \quad a_s(u, v) = - \int_{\Omega} (\Delta u) v \, dx = \int_{\Omega} \nabla u \cdot \nabla v \, dx,$$

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 ,

$$(2.11) \quad V_h = \text{span}\{\beta_{\mathbf{i}} \circ G^{-1} : \mathbf{i} \in \mathbf{I}\},$$

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 $u_{\mathbf{i}} \in \mathbb{R}$,

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

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

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

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

with the additional terms stemming from the domain transformation,

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

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

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

$$(2.17) \quad M_{\mathbf{i}, \mathbf{j}} = \int_{\hat{\Omega}} \beta_{\mathbf{i}} \beta_{\mathbf{j}} \omega \, d\hat{x},$$

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

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

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

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

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

$$(2.20) \quad K = \sum_{k, l=1}^D \int_{\hat{\Omega}} q_{k, l} \frac{\partial}{\partial \hat{x}_l} B \otimes \frac{\partial}{\partial \hat{x}_k} B \, d\hat{x}.$$

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 $Q(\hat{x}) = (\nabla G(\hat{x})^{-1})(\nabla G(\hat{x}))^{-T} \omega(\hat{x}) \in \mathbb{R}^{D \times D}$ 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 $d = 1, \dots, D$. The weight function $\omega(\hat{x})$ of (2.17) is interpolated as

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

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 $(\tilde{n}_1, \dots, \tilde{n}_D)$, 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,

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

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

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

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,

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

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

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

where $B^{(d)} \in \mathbb{R}^{n_d}$ is the vector holding all n_d univariate B-splines of $\mathbb{S}_{\xi_d}^{p_d}$. 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 $\omega^{(d)} = w_r^{(d)} \cdot \tilde{\beta}^{(d)}$,

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

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

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

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

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

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

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

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

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

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

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

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

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

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 pressing 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

$$(3.13) \quad W(i_1, \dots, i_D) = W_1(i_1) \cdots W_D(i_D),$$

where $W_d(i_d)$ is an $R_{d-1} \times R_d$ matrix for each fixed i_d , $1 \leq i_d \leq n_d$ 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),

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

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

$$(3.15) \quad \text{vec}(\omega(\hat{X})) = \underbrace{\left(B^{(1)}(\hat{X}^{(1)}) \otimes \cdots \otimes B^{(D)}(\hat{X}^{(D)}) \right)}_A \text{vec}(W).$$

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

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

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

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

which requires solving d linear systems of sizes $\tilde{n}_1, \dots, \tilde{n}_D$, 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,

$$(4.1) \quad \min_{y,u} \quad \frac{1}{2} \int_0^T \int_{\Omega} (y - \hat{y})^2 \, dx \, dt + \frac{\alpha}{2} \int_0^T \int_{\Omega} u^2 \, dx \, dt$$

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

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

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

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

$$(4.5) \quad \min_{y,u} \quad \sum_{k=1}^{N_t} \frac{\tau}{2} \left(\int_{\Omega} (y_k - \hat{y}_k)^2 \, dx + \alpha \int_{\Omega} u_k^2 \, dx \right)$$

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

$$(4.7) \quad y_k = 0 \quad \text{on } \partial\Omega, \text{ for } k = 1, \dots, N_t,$$

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

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

$$(4.9) \quad \text{s.t.} \quad \frac{M y_k - M y_{k-1}}{\tau} + K y_k = M u_k \quad \text{for } k = 1 : N_t,$$

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

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

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

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

For our discrete optimization problem this results in the conditions

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

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

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

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

We can rewrite these equations into an equation system

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

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

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

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),

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

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

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

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

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

$$(4.20) \quad \text{s.t.} \quad \frac{M y_k - M y_{k-1}}{\tau} + K y_k = M_2 u_k \quad \text{for } k = 1 : N_t,$$

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

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

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

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

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

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

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

for some $1 \leq d \leq D$. 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 d th TT factor instead of $\hat{F}_d$ and multiply $\Sigma V^\top$ with the $(d+1)$ th factor,

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

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

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 $n^D \times R_{d-1}n_d R_d$ *frame* matrix:

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

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

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

This linearity allows us to project the original problem into a subspace spanned by the columns of $F_{\neq d}$. Iterating over all $d = 1, \dots, D$, 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 $d = 1, \dots, D$, we compute the elements of $\hat{F}_d$ from the following reduced KKT system:

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

This system is small (each submatrix is now of size $R_{d-1}n_dR_d$), 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 $d \rightarrow d+1$) 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 $\bar{p}$	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,

$$(6.1) \quad \text{diff} = \frac{\|S - \tilde{S}\|_F}{\|S\|_F}.$$

We used two different spline spaces for the interpolation, one with splines of degree $\bar{p} = 5$ and one with degree $\bar{p} = 7$ 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 comparison 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

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

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

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 comparison 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. CALABRÒ, 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. KAZEYEV, 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. SORENSSEN, 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.

7 A low-rank matrix equation method for solving PDE-constrained optimization problems

This paper is published as *A. Bünger, V. Simoncini, and M. Stoll, A low-rank matrix equation method for solving PDE-constrained optimization problems* [16]. First Published in SIAM Journal of Scientific Computing in 43(5) (2021), pp. S637–S654, published by the Society for Industrial and Applied Mathematics (SIAM).

Novelty of this work. Looking at PDE-constrained optimization problems, if the space discretization is not given as in a low-rank format from Isogeometric Analysis, we can still decouple the resulting equation system in space and time. We can reformulate the KKT system into a Sylvester-like matrix equation. Approaching PDE-constrained optimization from this viewpoint lead us to a new low-rank solution method for this type of problem. By using an iterative rational Krylov method we project the matrix equation onto a small subspace. Then, imposing a Galerkin condition on its residual leads to a small reduced system. Subsequently enriching the subspace gives rise to a low-rank method.

Results from the paper. We can apply the matrix equation framework to a large number of example problems and get low-rank solutions within a fast time. Our method proved competitive against an existing low-rank method [94] by providing faster results with greater memory efficiency for most setups.

Contribution of the author. The author of this thesis provided about 60% of the work for this manuscript. This includes the derivation of the resulting matrix equation, the theoretical discussion, the derivation of the residual computation scheme, and the implementation and examination of the various examples within the paper as well as the comparison with a competing low-rank method.

Competing works. There already exists a low-rank method for PDE constrained optimization decoupling space and time [94]. This method uses a low-rank modification of a preconditioned MINRES [70] algorithm. We compare our new method to this existing low-rank approach, presenting faster and more memory efficient results for most setups.

A LOW-RANK MATRIX EQUATION METHOD FOR SOLVING
PDE-CONSTRAINED OPTIMIZATION PROBLEMS*ALEXANDRA BÜNGER[†], VALERIA SIMONCINI[‡], AND MARTIN STOLL[†]

Abstract. PDE-constrained optimization problems arise in a broad number of applications such as hyperthermia cancer treatment and blood flow simulation. Discretization of the optimization problem and using a Lagrangian approach result in a large-scale saddle-point system, which is challenging to solve, and acquiring a full space-time solution is often infeasible. We present a new framework to efficiently compute a low-rank approximation to the solution by reformulating the KKT system into a Sylvester-like matrix equation. This matrix equation is subsequently projected onto a small subspace via an iterative rational Krylov method, and we obtain a reduced problem by imposing a Galerkin condition on its residual. In our work we discuss implementation details and dependence on the various problem parameters. Numerical experiments illustrate the performance of the new strategy also when compared to other low-rank approaches.

Key words. PDE-constrained optimization, matrix equation, rational Krylov subspace

AMS subject classifications. 65F10, 49M41, 65F45, 65F55

DOI. 10.1137/20M1341210

1. Introduction. A vast number of today's technological advancements are due to the ever advancing improvement in modeling and solving of partial differential equation (PDE) problems. One class of especially challenging problems is PDE-constrained optimization, which has a host of applications, reaching from engineering design and control problems to medical applications like cancer treatment [21, 4, 9].

We consider a typical PDE-constrained optimization problem on a space-time cylinder $[0, T] \times \Omega$ given by

$$(1.1) \quad \min_{y,u} J(y, u)$$

$$(1.2) \quad \text{subject to } \dot{y} = \mathcal{L}(y) + u,$$

where we are interested in the minimization of the functional $J(y, u)$ constrained by a differential operator $\mathcal{L}(y)$ with space and time dependent state y and control u . An extensive analysis of this type of problem can be found, e.g., in [27] or [11]. In this work, we will follow the popular approach of first discretizing the problem and then formulating the discrete first-order optimality conditions as described in [12]. Finding an efficient numerical solution to the large system of equations resulting from these conditions is of high interest, and many approaches exist to solve the resulting saddle-point system such as using a block preconditioner to subsequently solve the preconditioned system with iterative solvers like MINRES (cf. [20, 17, 19, 22]).

*Received by the editors May 28, 2020; accepted for publication (in revised form) June 4, 2021; published electronically August 19, 2021.

<https://doi.org/10.1137/20M1341210>

Funding: The work of the first and third authors was supported by the German Science Foundation (DFG) through grant 1742243256 - TRR 9. The second author is a member of Indam-GNCS. Its support is gratefully acknowledged. This work was also supported by the DAAD-MIUR Mobility Program 2018 "Optimization and low-rank solvers for isogeometric analysis (34876)" between TU-Chemnitz and the Università di Bologna.

[†]Fakultät für Mathematik, Technische Universität Chemnitz, Chemnitz, 09126, Germany (alexandra.buenger@mathematik.tu-chemnitz.de, martin.stoll@mathematik.tu-chemnitz.de).

[‡]Alma Mater Studiorum - Università di Bologna, Bologna, 40126, Italy, and IMATI-CNR, Pavia, 27100, Italy (valeria.simoncini@unibo.it).

However, today's PDE models often result in millions of variables, and solving optimization problems governed by such huge models is still a challenging task due to their size and complexity. They often even prove to be impossible to solve on a normal computer due to the memory required to store the usually large and dense solution computed by standard algorithms. Therefore, memory-efficient solvers, which compute reduced approximations to the solution, are a desirable approach to solving these challenging problems. We here propose a widely applicable low-rank solver, which computes a projection of the solution onto a small subspace.

The method relies on the reformulation of the problem into a matrix equation, which will be illustrated in section 2. The numerical solution of this PDE-constrained optimization problem poses a significant challenge regarding the complexity of both storage and computational resources. We show that it is possible to separate spatial and temporal data via a novel low-rank matrix equation solver in section 3. We introduce a method to reduce the system of equations into a single low-rank matrix equation. In section 4 we discuss the choice of the approximation space, while in section 5 we analyze stopping criteria and a fast update scheme of the residual computation. And finally, in section 6, we examine the performance and applicability of our method on various numerical examples. We show the method's robustness with respect to the various problem parameters as well as assess the performance of our method compared with a previously proposed low-rank approach on a distributed heat equation model as well as a boundary control problem. Further, we show the applicability of our method to more challenging problems such as a partial state observation and a nonsymmetric PDE constraint.

2. Problem formulation. We consider a PDE-constrained optimal control problem on a space-time cylinder with time interval $[0, T]$ and a spatial domain Ω as in (1.1)–(1.2). The functional we want to minimize reads

$$(2.1) \quad J(y, u) = \frac{1}{2} \int_0^T \int_{\Omega_1} (y - \hat{y})^2 \, dx \, dt + \frac{\beta}{2} \int_0^T \int_{\Omega_u} u^2 \, dx \, dt.$$

Here y is the state, $\hat{y}$ is the desired state given on a subset Ω_1 of Ω , and u is the control on a subset Ω_u of Ω , which is regularized by the control cost parameter β .

For the PDE subject to which we want to minimize the functional $J(y, u)$, let us exemplarily consider the heat equation with $\mathcal{L}(y) = \Delta y$ and Dirichlet boundary condition,

$$(2.2) \quad \dot{y} - \Delta y = u \quad \text{in} \quad \Omega_u,$$

$$(2.3) \quad \dot{y} - \Delta y = 0 \quad \text{in} \quad \Omega/\Omega_u,$$

$$(2.4) \quad y = 0 \quad \text{on} \quad \partial\Omega.$$

There are two distinct options in proceeding with this problem. Either we formulate the optimality conditions and discretize them subsequently, or we first discretize the problem and then formulate the discrete first-order optimality conditions, known as the Karush–Kuhn–Tucker (KKT) conditions [12]. In this work we follow the second approach.

We discretize space and time in an all-at-once approach. The spatial discretization is done with finite elements, and to discretize in time, we split the time interval into n_T intervals of length $\tau = \frac{T}{n_T}$. Using a rectangle rule, the discretization of (2.1)

becomes

$$(2.5) \quad \sum_{t=1}^{n_T} \frac{\tau}{2} (y_t - \hat{y}_t)^T M_1 (y_t - \hat{y}_t) + \frac{\tau\beta}{2} u_t^T M_u u_t,$$

where $y_t, \hat{y}_t$, and u_t are spatial discretizations of $y, \hat{y}$ of size n and u of size n_u for each time step $t = 1, \dots, n_T$. Using an implicit Euler scheme [14] the discrete formulation of the PDE (2.2) reads

$$(2.6) \quad \frac{M(y_t - y_{t-1})}{\tau} + K y_t = N u_t \quad \text{for } t = 1, \dots, n_T,$$

$$(2.7) \quad y_0 = 0.$$

Here, $M, M_1 \in \mathbb{R}^{n \times n}$, $M_u \in \mathbb{R}^{n_u \times n_u}$, and $N \in \mathbb{R}^{n \times n_u}$ are the mass matrices of the spatial discretization, and $K \in \mathbb{R}^{n \times n}$ denotes the so-called stiffness matrix resulting from the discretization of the differential operator $\mathcal{L}(y)$. The boundary constraints are incorporated into the stiffness matrix. For further details about the discretization of PDE operators we refer the reader to [23].

In the course of this paper we cover three distinct setups for this problem:

- (i) All domains are the same, $\Omega = \Omega_1 = \Omega_u$, and thus we have a full domain observation and control, leading to $n = n_u$ and $M = M_1 = M_u = N$.
- (ii) We have a full observation $\Omega_1 = \Omega$ but only a partial control, e.g., a boundary control $\Omega_u = \partial\Omega$ leading to $n_u < n$ and $M = M_1 \neq M_u$, and N being rectangular.
- (iii) We have a full domain control $\Omega_u = \Omega$ but only a partial observation on a smaller domain Ω_1 , leading to $M_1 \neq M$ with M_1 being singular due to entries set to zero on the nonobserved parts of the state.

Additionally, we will address the case of $K \neq K^T$ later on.

We collect the discretizations of the variables in matrices $Y = [y_1, \dots, y_{n_T}] \in \mathbb{R}^{n \times n_T}$ and denote their vectorizations by $\underline{Y} = \text{vec}(Y)$, respectively for $\hat{y}$ and u . With this we write the optimization problem in compact form as

$$(2.8) \quad \min_{\underline{Y}, \underline{U}} \frac{\tau}{2} (\underline{Y} - \hat{\underline{Y}})^T \mathcal{M}_1 (\underline{Y} - \hat{\underline{Y}}) + \frac{\tau\beta}{2} \underline{U}^T \mathcal{M}_u \underline{U},$$

$$(2.9) \quad \text{s.t. } \mathcal{K} \underline{Y} - \tau \mathcal{N} \underline{U} = 0,$$

with the matrices

$$(2.10) \quad \mathcal{M} = \begin{bmatrix} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \end{bmatrix}, \quad \mathcal{K} = \begin{bmatrix} M + \tau K & & & \\ -M & \ddots & & \\ & \ddots & \ddots & \\ & & -M & M + \tau K \end{bmatrix},$$

where mass matrices $M \in \mathbb{R}^{n \times n}$ and stiffness matrix $K \in \mathbb{R}^{n \times n}$ repeat n_T times each. The matrices $\mathcal{M}_u$ and $\mathcal{N}$ are block diagonal matrices like $\mathcal{M}$ but with the different mass matrices M_u and N , respectively, on the diagonal.

To solve the discrete problem (2.8)–(2.9), we have to solve the system of equations resulting from the first-order optimality conditions [3]. They state that an optimal solution must be a saddle point of the Lagrangian of the problem

$$(2.11) \quad \nabla \mathcal{L}(\underline{Y}^*, \underline{U}^*, \underline{\Lambda}^*) = 0.$$

The Lagrangian of this problem reads

$$(2.12) \quad \mathcal{L}(\underline{Y}, \underline{U}, \underline{\Lambda}) = \frac{\tau}{2}(\underline{Y} - \hat{\underline{Y}})^T \mathcal{M}_1(\underline{Y} - \hat{\underline{Y}}) + \frac{\tau\beta}{2}\underline{U}^T \mathcal{M}_u \underline{U} + \underline{\Lambda}^T (\mathcal{K}\underline{Y} - \tau\mathcal{N}\underline{U}),$$

with Lagrange multipliers $\underline{\Lambda}$. Thus, the optimal solution solves the following set of linear equations:

$$(2.13) \quad 0 = \nabla_Y \mathcal{L}(\underline{Y}, \underline{U}, \underline{\Lambda}) = \tau \mathcal{M}_1(\underline{Y} - \hat{\underline{Y}}) + \mathcal{K}^T \underline{\Lambda},$$

$$(2.14) \quad 0 = \nabla_U \mathcal{L}(\underline{Y}, \underline{U}, \underline{\Lambda}) = \tau\beta \mathcal{M}_u \underline{U} - \tau \mathcal{N}^T \underline{\Lambda},$$

$$(2.15) \quad 0 = \nabla_{\Lambda} \mathcal{L}(\underline{Y}, \underline{U}, \underline{\Lambda}) = \mathcal{K}\underline{Y} - \tau \mathcal{N}\underline{U}.$$

Memorywise, an effective approach to solving this large-scale problem is to use a low-rank approach, which finds a cheap representation of the solution matrix within a low-rank subspace $\text{range}(V)$ and a reduced solution Z , whose number of columns is related to the number of employed time steps,

$$(2.16) \quad Y \approx VZ,$$

and similarly for the other matrix variables U and Λ .

In this work we show that we can compute such a low-rank solution based on a matrix equation rather than using the saddle-point formulation used in [26]. This is achieved by rearranging the large system of equations (2.13)–(2.15) into a generalized Sylvester matrix equation of the form

$$(2.17) \quad A_1 X + X C + A_2 X I_0 - A_3 X D - F_1 F_2 = 0.$$

The resulting matrix equation can be efficiently solved by using a tailored low-rank Krylov subspace method. This new approach greatly reduces storage and time requirements while being robust to parameter changes. This matrix equation oriented methodology allows one to clearly identify space and time dimensions and aims at reducing the problem dimension in the space variables; the time variable is handled using an all-at-once procedure. Similar strategies have been used, for instance, in [5, 16].

One assumption we make is that we have a low-rank approximation or representation of the desired state as

$$(2.18) \quad \hat{Y} \approx Y_1 Y_2,$$

with $Y_1 \in \mathbb{R}^{n \times r}$, $Y_2 \in \mathbb{R}^{r \times n_T}$, and $r < n_T$. First, we introduce the auxiliary matrices

$$(2.19) \quad \mathcal{I} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \text{ and } \mathcal{C} = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}$$

to rewrite the system matrices as Kronecker products $\mathcal{M} = \mathcal{I} \otimes M$, $\mathcal{M}_1 = \mathcal{I} \otimes M_1$, $\mathcal{N} = \mathcal{I} \otimes N$, and $\mathcal{K} = \mathcal{I} \otimes \tau K + \mathcal{C} \otimes M$. Note that here $\mathcal{C}$ results from the implicit Euler scheme. This scheme can be replaced by a different time stepping method like a Crank–Nicolson scheme [13], in which case $\mathcal{C}$ and the identities $\mathcal{I}$ in $\mathcal{N}$ and $\mathcal{K}$ will become different block matrices. This would maintain the structure of the

following steps but would naturally make solving the resulting matrix equation more challenging.

With this our system of KKT conditions (2.13)–(2.15) becomes

$$(2.20) \quad \tau(\mathcal{I} \otimes M_1)\underline{Y} + (\mathcal{I} \otimes \tau K^T + \mathcal{C}^T \otimes M^T)\underline{\Lambda} - \tau(\mathcal{I} \otimes M_1)\hat{Y} = 0,$$

$$(2.21) \quad \tau\beta(\mathcal{I} \otimes M_u)\underline{U} - \tau(\mathcal{I} \otimes N^T)\underline{\Lambda} = 0,$$

$$(2.22) \quad (\mathcal{I} \otimes \tau K + \mathcal{C} \otimes M)\underline{Y} - \tau(\mathcal{I} \otimes N)\underline{U} = 0.$$

We exploit the relation

$$(2.23) \quad (W^T \otimes V)\text{vec}(Y) = \text{vec}(VYW)$$

to rewrite the equations (2.20)–(2.22) as

$$(2.24) \quad \tau M_1 Y + \tau K^T \Lambda + M^T \Lambda C - \tau M_1 \hat{Y} = 0,$$

$$(2.25) \quad \tau\beta M_u U - \tau N^T \Lambda = 0,$$

$$(2.26) \quad \tau KY + MYC^T - \tau NU = 0.$$

The mass matrix M arising from a standard Galerkin method is symmetric and can be considered lumped, i.e., diagonal. Therefore, we can eliminate (2.25) by setting $U = \frac{1}{\beta} M_u^{-1} N^T \Lambda$ in the remaining two equations,

$$(2.27) \quad \tau M_1 Y + \tau K^T \Lambda + M^T \Lambda C - \tau M_1 \hat{Y} = 0,$$

$$(2.28) \quad \tau KY + MYC^T - \frac{\tau}{\beta} N M_u^{-1} N^T \Lambda = 0,$$

which are equivalent to

$$(2.29) \quad M^{-1} M_1 Y + M^{-1} K^T \Lambda + \Lambda \tilde{C} - M^{-1} M_1 \hat{Y} = 0,$$

$$(2.30) \quad M^{-1} KY + Y \tilde{C}^T - \frac{1}{\beta} M^{-1} N M_u^{-1} N^T \Lambda = 0,$$

where $\tilde{C} = \frac{1}{\tau} C$. For now, let us assume that $K = K^T$, but our approach can be easily generalized for nonsymmetric K as we will demonstrate later on. Now this representation corresponds to

$$(2.31) \quad \begin{aligned} & M^{-1} K \begin{bmatrix} Y & \Lambda \end{bmatrix} + \begin{bmatrix} Y & \Lambda \end{bmatrix} \underbrace{\begin{bmatrix} \tilde{C}^T & 0 \\ 0 & \tilde{C} \end{bmatrix}}_{C_1} + M^{-1} M_1 \begin{bmatrix} Y & \Lambda \end{bmatrix} \underbrace{\begin{bmatrix} 0 & \mathcal{I} \\ 0 & 0 \end{bmatrix}}_{I_0} \\ & + \frac{1}{\beta} M^{-1} N M_u^{-1} N^T \begin{bmatrix} Y & \Lambda \end{bmatrix} \underbrace{\begin{bmatrix} 0 & 0 \\ -\mathcal{I} & 0 \end{bmatrix}}_D - \underbrace{\begin{bmatrix} 0 & M^{-1} M_1 \hat{Y} \end{bmatrix}}_{F_1 F_2} = \begin{bmatrix} 0 & 0 \end{bmatrix}. \end{aligned}$$

We denote $X = \begin{bmatrix} Y & \Lambda \end{bmatrix}$, $A_1 = M^{-1} K$, $A_2 = M^{-1} M_1$, $A_3 = \frac{1}{\beta} M^{-1} N M_u^{-1} N^T$, $F_1 = M^{-1} M_1 Y_1$, and $F_2 = \begin{bmatrix} 0_{r \times n_T} & Y_2 \end{bmatrix}$, where $\hat{Y} = Y_1 Y_2$ with Y_1, Y_2 of low column and row rank, respectively. Then we get the desired format from (2.17) as

$$(2.32) \quad A_1 X + X C_1 + A_2 X I_0 + A_3 X D - F_1 F_2 = 0,$$

where the left-hand coefficient matrices A_1 , A_2 , and A_3 have size $n \times n$ while the right-hand ones C_1 , I_0 , and D have size $2n_T \times 2n_T$, so that $X \in \mathbb{R}^{n \times 2n_T}$.

3. Low-rank solution. The generalized Sylvester equation in (2.32) replaces the large Kronecker product based system of equations (2.20)–(2.22). Since the solution matrix $X \in \mathbb{R}^{n \times 2n_T}$ will be dense and potentially very large, to exploit the new setting it is desirable to find an appropriate approximation space and a low-rank reduced matrix approximation $Z \in \mathbb{R}^{p \times 2n_T}$ such that

$$(3.1) \quad X \approx V_p Z,$$

where the orthonormal columns of $V_p \in \mathbb{R}^{n \times p}$ generate the approximation space. With this setting we can construct a reduced version of the matrix equation (2.32). Let us assume we compute an approximation as in (3.1). Then the residual matrix associated with (2.32) reads

$$(3.2) \quad R = A_1 V_p Z + V_p Z C_1 + A_2 V_p Z I_0 + A_3 V_p Z D - F_1 F_2.$$

We impose the Galerkin orthogonality of the residual matrix with respect to the approximation space, which in the matrix inner product is equivalent to writing $V_p^T R = 0$, so that our equation becomes

$$(3.3) \quad V_p^T A_1 V_p Z + V_p^T V_p Z C_1 + V_p^T A_2 V_p Z I_0 + V_p^T A_3 V_p Z D - V_p^T F_1 F_2 = 0.$$

Let us denote the reduced $p \times p$ coefficient matrices as $A_{1,r} := V_p^T A_1 V_p$, $A_{2,r} := V_p^T A_2 V_p$, $A_{3,r} := V_p^T A_3 V_p$ and set $F_{1,r} = V_p^T F_1 \in \mathbb{R}^{p \times 1}$. The resulting reduced equation

$$(3.4) \quad A_{1,r} Z + \mathcal{I}_p Z C_1 + A_{2,r} Z I_0 + A_{3,r} Z D - F_{1,r} F_2 = 0$$

has the same structure as the original matrix equation (2.32), but its size is reduced to $p \times 2n_T$. By exploiting once again the relation in (2.23), we get the small linear system of equations

$$(3.5) \quad ((\mathcal{I}_{2n_T} \otimes A_{1,r}) + (C_1^T \otimes \mathcal{I}_p) + (I_0^T \otimes A_{2,r}) + (D^T \otimes A_{3,r})) \underline{Z} = \text{vec}(F_{1,r} F_2),$$

with $\underline{Z} = \text{vec}(Z)$. For a small subspace size $p \ll n$ this system of equations is significantly easier to solve, and we can use either a direct or an iterative method to do so. If the obtained approximate solution $V_p Z$ is not sufficiently good, then the approximation space can be expanded and a new approximation constructed, giving rise to an iterative method. The use of low-rank methods within optimization of large-scale systems has been successfully documented in several articles, and we refer the reader to [26, 7, 6, 2] for recent accounts.

4. Subspace computation. To construct the projection (3.1) we need an iterative subspace method, which constructs a relevant subspace for our problem. For this we make use of rational Krylov subspaces

$$(4.1) \quad \mathcal{K}_p(A, v, \mathbf{s}) = \text{span} \left\{ v, (A + s_1 I)^{-1} v, \dots, \prod_{j=1}^{p-1} (A + s_j I)^{-1} v \right\}$$

with shifts $\mathbf{s} = [s_1, \dots, s_j]$ as described in [25]. Rational Krylov subspaces have become an important tool in many dimension reduction problems such as eigenvalue problems, dynamical systems, approximation of matrix functions, and multiterm matrix equations. The main advantage is that they are able to capture crucial spectral

information on the matrix A in (4.1) already with small values of p , compared with standard (polynomial) Krylov subspaces. The reduced problem obtained by projection onto a rational Krylov subspace is thus able to preserve the leading spectral and functional properties of the original problem, but in a dramatically smaller dimensional space. As an initial vector (or block of vectors, in the case of $\text{rank}(F_1) > 1$) we take the right-hand side, $v_0 = F_1$, and to construct the subspace (4.1) we employ a tailored strategy to adapt to the different settings.

The idea is to generate an approximation space that contains spectral information of all matrices A_i , $i = 1, \dots, 3$, so as to be effective already for a small dimension. Without further information, this goal would be achieved by generating a subspace that is the sum of $\mathcal{K}_p(A_i, v_0, \mathbf{s})$, of dimension at most three times p . However, by accurately selecting the matrices involved in the generation of the Krylov subspaces, smaller dimensional spaces can be determined. We recall the three previously described setups and make distinct subspace choices for each of them:

- (i) $M_1 = M = N = M_u$, *all full rank*. In this case $A_2 = I$ and A_3 is a diagonal nonsingular matrix. We construct a rational Krylov subspace from $A = A_1$.
- (ii) $M_1 \neq M$, $M = M_u = N$, and M_1 of *lower rank*. We construct a mixed subspace where we add the following two new vectors in step k :

$$(4.2) \quad \{(A_1 + s_k^{(1)}I)^{-1}v_k, (A_2 + s_k^{(2)}I)^{-1}v_k\},$$

so that the space dimension grows by at most two per iteration, instead of one. This adds the influence of M_1 to the constructed subspace.

- (iii) $M_1 = M$, N *tall*, $n_u < n$. Here, $A_2 = I$ while $A_3 = M^{-1}NM_u^{-1}N^T$ is not invertible. We thus define $A_3(\alpha) = A_3 + \alpha A_1$; this selection is justified below. In this case we use the following two new vectors in step k :

$$(4.3) \quad \{(A_1 + s_k^{(1)}A_3(\alpha))^{-1}v_k, (A_3(\alpha) + s_k^{(2)}A_1)^{-1}v_k\},$$

and once again the space dimension grows by at most two per iteration, instead of one.

- (iv) $M = M_1 = M_u = N$, *all full rank*, $K \neq K^T$. For a number of PDE constraints, like convection-diffusion problems, the stiffness matrix is nonsymmetric. In this case, we have to slightly modify the first component A_1X in (2.32) to $A_1X \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} + M^{-1}K^T X \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$. Here again we construct a mixed subspace where we add two new vectors in step k :

$$(4.4) \quad \{(A_1 + s_k^{(1)}I)^{-1}v_k, (M^{-1}K^T + s_k^{(2)}I)^{-1}v_k\}.$$

A strategy similar to (ii)–(iv) was successfully adopted in [18] for a multiterm linear matrix equation in a different context. The effectiveness of the mixed approaches in (ii)–(iii) relies on the fact that the generated space is rich in eigencomponents of both matrices. In (ii), where A_2 is diagonal and singular with a bounded and tight nonzero spectral interval, a good spectral homogeneity in the space construction can be reached by shifting the matrix A_2 by some α so that the spectra of A_1 and $A_2 + \alpha_2 I_n$ are contained in the same interval. Hence, we introduce a shift parameter α_2 in (2.32) to get

$$(4.5) \quad A_1X + X(C_1 - \alpha_2 I_0) + (A_2 + \alpha_2 I_n)XI_0 - A_3XD - F_1F_2 = 0.$$

With these premises, an appropriate choice for α_2 is a rough approximation to the largest eigenvalue of A_1 . Due to the good properties of the transformed spectrum, the

shifts were computed by only using the projection of the matrix A_1 onto the generated space; more details on the shift selection can be found in [8]. In cases (i) and (iv), A_2 is not singular, but still the spectra of A_1 and A_2 differ greatly. Applying the same strategy to shift A_2 as in (4.5) also proved to be beneficial in these cases.

In (iii), the structure and singularity of A_3 was significantly more challenging, because $\|A_3\|$ inversely depends on β ; hence the spectrum of a shifted version of A_3 may not be enclosed in that of A_1 . We propose considering the equivalent problem

$$(4.6) \quad A_1 X(I - \alpha_3 D) + X C_1 + A_2 X I_0 - (A_3 + \alpha_3 A_1) X D - F_1 F_2 = 0,$$

where $\alpha_3 = \frac{1}{\sqrt{\beta}} \|A_3\|_F / \|A_1\|_F$. With this choice the Frobenius norms of $\alpha_3 A_1$ and A_3 have similar magnitude, and the influence of β is reflected realistically in the equation. With this formulation, we found particularly successful the use of the projection of the pair $(A_1, A_3 + \alpha_3 A_1)$ onto the current approximation space to determine the next shifts during the iteration.

Further subspace reduction. By computing $X \approx V_p Z$ we want to reduce storage requirements and computation time. This goal is only achieved if the approximation space dimension remains considerably smaller than n_T . By enriching the subspace as outlined above, however, the space dimension may in principle grow up to n in case the solution is not sufficiently accurate. To keep the subspace as small as possible when it is not optimal we include a truncation scheme which bounds the subspace dimension to less than n_T . Thus, in the worst case the solution will have maximum rank. To reduce the dimension we compute a singular value decomposition of $Z = U \Sigma V^T$, with $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p)$, where $\sigma_1 \geq \dots \geq \sigma_p$ are the singular values of Z . In case some of the singular values are too small, say $\sigma_i < \varepsilon$ for some i , it is clear that some of the subspace information is not needed for computing a good approximation to the solution. If this occurs, we truncate the subspace with $\tilde{p} = i - 1$ by setting

$$(4.7) \quad V_{\tilde{p}} = V_p U_{\tilde{p}},$$

with $U_{\tilde{p}}$ denoting the first $\tilde{p}$ columns of U . Note that the next subspace additions (4.2) are still conducted using v_k from the latest generated vectors in the original subspace V_p , and orthogonalized with respect to the reduced subspace. We refer the reader to the discussion of Table 6.4 for an illustration of the achievable memory savings by adopting this strategy.

5. Residuals and stopping criteria. To determine a computable stopping criterion we monitor the residuals of the two equations (2.27) and (2.28),

$$(5.1) \quad R_1 = \tau M_1 Y + \tau K^T \Lambda + M^T \Lambda C - \tau M_1 \hat{Y},$$

$$(5.2) \quad R_2 = \tau K Y + M Y C^T - \frac{\tau}{\beta} N M^{-1} N^T \Lambda,$$

which are closely related to the original system.

Computing the residuals poses a bottleneck in this scheme as straightforward computation is time consuming and would require forming the full solution $[Y \Lambda] = V_p Z$. To avoid this and substantially speed up the computation time, we rely on a low-rank representation of the residual and calculate its norm, making use of an

updated QR method. The matrix residuals can be rewritten as

$$(5.3) \quad R_1 = \tau M_1 V_p Z_Y + \tau K^T V_p Z_\Lambda + M V_p Z_\Lambda C - \tau M_1 \hat{Y}$$

$$(5.4) \quad = [\tau M_1 V_p \quad \tau K^T V_p \quad M V_p \quad -\tau M_1 Y_1] \cdot \begin{bmatrix} Z_Y \\ Z_\Lambda \\ Z_\Lambda C \\ Y_2 \end{bmatrix},$$

$$(5.5) \quad R_2 = \tau K V_p Z_Y + M V_p Z_Y C^T - \frac{\tau}{\beta} N M^{-1} N^T V_p Z_\Lambda$$

$$(5.6) \quad = [\tau K V_p \quad M V_p \quad -\frac{\tau}{\beta} N M^{-1} N^T V_p] \cdot \begin{bmatrix} Z_Y \\ Z_Y C^T \\ Z_\Lambda \end{bmatrix},$$

where Z_Y denotes the Y -component of Z and Z_Λ denotes the component associated with Λ , respectively. Let either of the two quantities be denoted by

$$(5.7) \quad R = R_L R_R.$$

Here the matrices have dimensions $R_L \in \mathbb{R}^{n \times (3p+1)}$ and $R_R \in \mathbb{R}^{(3p+1) \times n_T}$. We consider a reduced QR decomposition of the tall and skinny matrix R_L ,

$$(5.8) \quad R_L = Q_1 R_{1,1}, \quad \text{with } Q_1 \in \mathbb{R}^{n \times (4p+1)}, \quad R_{1,1} \in \mathbb{R}^{(4p+1) \times (4p+1)}.$$

At each iteration new columns are added to the matrices R_L and new rows to R_R . To further reduce the computational cost, we update the QR decomposition so as to avoid a new factorization from scratch at each iteration. Assume that we have a current subspace of size p so that $V_p = [v_1, \dots, v_p]$ and we add another vector v_{p+1} to the space. Thus, four new columns are added to R_L , which we add to the end of R_L , while keeping the same reordering for R_R . Adding a new column v_{p+1} gives

$$(5.9) \quad [R_L \quad v_{p+1}] = [Q_1 R_{1,1} \quad v_{p+1}] = [Q_1 \quad q_2] \begin{bmatrix} R_{1,1} & R_{1,2} \\ 0 & R_{2,2} \end{bmatrix},$$

where only two column vectors q_2 and $R_{1,2}$ and a scalar value $R_{2,2}$ have to be computed. We have

$$(5.10) \quad Q_1 R_{1,2} + q_2 R_{2,2} = v_{p+1}.$$

Setting $R_{1,2} = Q_1^T v_{p+1}$ and constructing the vector

$$(5.11) \quad \hat{q}_2 = v - Q_1(Q_1^T v_{p+1}) = v_{p+1} - Q_1 R_{1,2},$$

we can set $q_2 = \frac{\hat{q}_2}{\|\hat{q}_2\|}$ and thus $R_{2,2} = \|\hat{q}_2\|$. With this, (5.10) holds, and the new column q_2 is orthogonal to Q_1 ,

$$(5.12) \quad Q_1^T q_2 = Q_1^T \frac{\hat{q}_2}{R_{2,2}} = \frac{Q_1^T v_{p+1} - Q_1^T Q_1 Q_1^T v_{p+1}}{R_{2,2}} = 0.$$

Now we can completely avoid forming the full residuals, as with $R = R_L R_R = Q_1 R_{1,1} R_R$ the desired Frobenius norm of the residual becomes

$$(5.13) \quad \|R\|_F = \sqrt{\text{trace}(R^T R)} = \sqrt{\text{trace}(R_R^T R_{1,1}^T Q_1^T Q_1 R_{1,1} R_R)} = \sqrt{\text{trace}(R_R^T R_{1,1}^T R_{1,1} R_R)}.$$

To compute the residual norms in this form only a very small matrix of size $n_T \times n_T$ has to be computed. With this, the norm of both residuals in (5.1) and (5.2), $\|R_1\|_F$ and $\|R_2\|_F$, can be computed cheaply without forming the approximations Y and Λ .

Following the guidelines in [10] for linear matrix equations, we also monitor a scaled backward error norm of the two equations, that is,

$$(5.14) \quad \rho_3 = \frac{\|R_1\|_F}{\tau(\|M_1\|_F\|Z_Y\|_F + \|K\|_F\|Z_\Lambda\|_F + \|M\|_F\|\hat{Y}\|_F) + \|M\|_F\|Z_Y\|_F\|C\|_F} + \frac{\|R_2\|_F}{\tau(\|K\|_F\|Z_Y\|_F + \frac{1}{\beta}\|M\|_F\|Z_\Lambda\|_F + \|M\|_F\|Z_Y\|_F\|C\|_F)},$$

which takes into account the data order of magnitude. Summarizing, we stop our iteration whenever

$$(5.15) \quad \max\{\|R_1\|_F, \|R_2\|_F, \rho_3\} \leq \text{tol},$$

where tol is a prescribed tolerance. Note that this criterion requires that both the relative and absolute quantities satisfy the condition for the iteration to stop. We also remark that the use of the formulation in (5.13) may lead to small inaccuracies adding up to be significantly larger than machine precision. However, these inaccuracies always stayed far below our convergence tolerance, which was usually set even larger, say $\text{tol} = 10^{-6}$.

6. Numerical results. We now present the performance and flexibility of our method on multiple examples for different PDE-constrained optimization problems. The spatial finite element discretizations of the PDE operator were conducted using the deal.II framework [1] with Q1 finite elements, and the method described in the previous section was implemented in MATLAB R2018b. All experiments were run on a desktop computer with an INTEL Core i7-4770 Quad-core processor running at 4 × 3400 MHz with 32 GB of RAM.

We will show robustness with respect to the discretization sizes n and n_T as well as the control parameter β . Furthermore, we demonstrate the applicability of our method to multiple different setups such as partial observation of the desired state, boundary control, and a different nonsymmetric PDE operator.

Other low-rank solvers. To emphasize the competitiveness of our new approach we compare it with another low-rank method aimed at the same problem class. The idea of exploiting the system structure to avoid forming a large system of equations and finding a low-rank way to compactly represent the solution is not an entirely new approach. Earlier, in [26] the authors developed a technique involving rewriting the problem in a low-rank context and solving it with a preconditioned Krylov subspace solver, namely MINRES, introduced in [15]. We denote this solver as LRMINRES (low-rank MINRES).¹

Here, the variables are represented in a low-rank matrix format rather than vectors, as $Y = W_Y V_Y^T$, $U = W_U V_U^T$, and $\Lambda = W_\Lambda V_\Lambda^T$, and the system of equations (2.24)–(2.26) is split into a matrix product as

$$(6.1) \quad \begin{bmatrix} \tau M_1 W_Y & \tau K^T W_\Lambda & M \Lambda \end{bmatrix} \begin{bmatrix} V_Y^T \\ V_\Lambda^T \\ V_\Lambda^T C \end{bmatrix} = \tau M_1 \hat{Y}$$

¹The LRMINRES MATLAB code used for our comparison is available under <https://www.tu-chemnitz.de/mathematik/wire/pubs/PubCodes.zip>.

for (2.24) and respectively for (2.25) and (2.26). From here, the system of equations is solved with a tailored version of MINRES. As opposed to the original version of MINRES, the residual and all other upcoming variables are formulated as low-rank matrices rather than vectors and subsequently truncated to keep memory requirements low. Combined with exploiting the structure of the low-rank equation system in the upcoming matrix vector products and preconditioning, this method provides a memory-efficient and fast alternative to established approaches for solving PDE-constrained optimization problems.

A bottleneck in this approach, however, is the setup of a preconditioner. This becomes increasingly difficult for decreasing β . To construct the necessary low-rank preconditioner, a large-scale matrix equation needs to be solved, which significantly affects the overall performance. As opposed to this, our approach directly starts from a matrix equation and gains convergence speed from carefully selecting the subspace, which makes additional preconditioning obsolete. This appears to be more stable with respect to changing β . Therefore, our approach proves to be superior in many setups. Theoretical justification for the effectiveness of rational spaces can be found in the literature; see related references in [25].

6.1. Setup (i): Fully observed heat equation. As a first example PDE we use the heat equation with a full observation; thus $\Omega_1 = \Omega$ and $M_1 = M$. This leads to a simplified version of (2.17) as

$$(6.2) \quad A_1 X + X C_1 + X I_0 + A_3 X D - Y_1 F_2^T = 0.$$

First, we will use this simple setup to investigate the convergence behavior and our choice of stopping criteria by comparing the results to a solution obtained by a direct solver accurate up to machine precision. The constant-in-time desired state $\hat{Y}$ is displayed in Figure 6.1. This constant desired state leads to a low-rank representation of the right-hand side of rank 1. The reference solution was obtained by solving the linear equation system of a small setup with the MATLAB *backslash* operator. We used a discretization of $n = 1089$ and $n_T = 100$.

FIG. 6.1. Constant-in-time desired state for the experiments in section 6.1 with Q_1 —finite elements on a square domain. Here, the desired state is realized with $n = 1089$ spatial nodes and Dirichlet boundary conditions set to zero.

The monitored quantities (5.1), (5.2), (5.14) and the actual relative errors to the reference solution are displayed in Figure 6.2. We see that the monitored quantities are a good estimation for the magnitude of the errors as R_1 stays close above the actual errors. Inserting the direct solution into (2.32) gives a residual of $5.11 \cdot 10^{-9}$. Thus, the different scaling of the matrix equation prohibits further accuracy gains

TABLE 6.1

Example 6.1. Subspace size p and required time for different discretizations.

n	1089		4225		16641		66049		263169	
n_T	p	time (s)	p	time (s)	p	time (s)	p	time (s)	p	time (s)
20	8	0.02	11	0.07	11	0.23	11	0.95	12	5.37
100	7	0.03	11	0.09	11	0.25	11	1.02	12	5.37
500	6	0.04	11	0.19	11	0.36	11	1.11	13	6.06
2500	6	0.18	11	0.89	11	1.04	10	1.47	15	8.85

once a high accuracy is reached. This effect is reflected in Figure 6.2 as the last iterations do not provide further accuracy gains.

FIG. 6.2. Observed stopping criteria and actual error progression.

Example 6.1. We continue with the above simple example, i.e., the heat equation with full observation, $\Omega_1 = \Omega$ and $M_1 = M$, and a constant-in-time desired state. We now investigate the performance of our method with respect to time and space discretization. We vary the number of time steps from 20 to 2500 and the number of spatial discretization nodes from $n = 1089$ to $n = 263169$, which roughly resembles up to a total of 78 million degrees of freedom. Here again we fix the control parameter to $\beta = 10^{-4}$. As seen in Table 6.1, increasing the discretization size barely impacts the resulting subspace sizes p . Additionally, the time needed to solve the optimization problems increases considerably slowly.

TABLE 6.2

Example 6.2. Comparison of the resulting subspace size p , the solution rank r , and required CPU time between the new SYS2MATEQ and another low-rank scheme, LRMINRES, for $\tilde{Y}$ with rank one.

n		1089			4225			16641		
β	method	p	r	time (s)	p	r	time (s)	p	r	time (s)
10^{-1}	SYS2MATEQ	6	6	0.09	5	5	0.05	5	5	0.14
	LRMINRES	20	17	4.00	22	21	7.49	22	21	31.59
10^{-3}	SYS2MATEQ	7	7	0.07	7	7	0.09	7	7	0.22
	LRMINRES	11	7	1.80	14	11	5.46	11	10	24.75
10^{-5}	SYS2MATEQ	14	13	0.19	14	13	0.23	14	13	0.53
	LRMINRES	7	6	1.08	7	6	4.75	-	-	-

Example 6.2. With the same data as in Example 6.1 we report on performance comparisons with the low-rank preconditioned MINRES (LRMINRES) proposed in [26] and introduced above. We fixed the number of time steps to $n_T = 100$ and computed solutions for different discretization sizes and control parameters β . The maximum discretization size here is $n = 16641$. The results in Table 6.2 reveal that our method's performance is superior with regard to both required memory and CPU time. Here, our method is labeled as SYS2MATEQ. This acronym stands for *system to matrix equation*. The column denoted by p states the subspace size for SYS2MATEQ, while for LRMINRES it denotes the maximum rank per vector needed during the iterations, of which up to 15 are required. The column r states the rank of the final solution in both methods. Note that even though sometimes the ranks achieved by LRMINRES are smaller, the required memory to store the solution is still greater. This is because the LRMINRES scheme needs to store a low-rank representation $U_1 U_2^T$ for each of the three variables, as opposed to SYS2MATEQ, where the same subspace V_p is used for all variables. Additionally, during the iterations our method needs to store only one subspace of size p , whereas LRMINRES is required to store multiple vectors of size p . Also, our scheme does not rely on the often time-consuming computation of a preconditioner, which is needed in LRMINRES. The convergence speed depends on the quality of the preconditioner, and for LRMINRES this requires the solution of two matrix equations as part of the Schur-complement approximation of the saddle-point system. In order to achieve this, LRMINRES relies on the KPIK method (see [24]) introduced in [5] for a similar situation. This gets harder for large spatial discretizations combined with decreasing β . As a result, the last entry of Table 6.2 does not show convergence for LRMINRES.

With the same settings as before, we now raise the rank of the desired state $\hat{Y}$ to 6—leading to a larger rank matrix $F_1 F_2^T$ —and compare both methods once again. Table 6.3 displays the results as before. Again, our method successfully converges with a small subspace size within a very short amount of time, whereas LRMINRES is considerably slower, especially for larger problem sizes, and does not converge for the largest discretization with β being very small.

TABLE 6.3

Example 6.2. Comparison of SYS2MATEQ and LRMINRES for $\text{rank}(\hat{Y}) = 6$. The resulting subspace size p , the solution rank r , and CPU time are displayed.

n β	method	1089			4225			16641		
		p	r	time (s)	p	r	time (s)	p	r	time (s)
10^{-1}	SYS2MATEQ	21	21	0.38	21	21	0.47	19	19	0.72
	LRMINRES	29	22	20.77	27	25	13.38	29	29	23.63
10^{-3}	SYS2MATEQ	31	31	1.08	30	30	1.09	30	29	1.77
	LRMINRES	15	11	2.22	17	11	11.19	19	15	31.05
10^{-5}	SYS2MATEQ	49	43	3.75	49	44	3.97	51	41	5.88
	LRMINRES	11	8	1.49	11	9	6.14	-	-	-

We are also interested in the memory savings our method provides. Therefore, we monitor the memory consumption of SYS2MATEQ, LRMINRES, and a full rank MINRES method for the same setups displayed in Table 6.3. For SYS2MATEQ we monitor the memory needed to construct the subspace, the reduced solution, and the setup of the reduced equation system (3.5). For LRMINRES we monitor the memory for all vectors that are used. For comparison we additionally report the memory consumed by a standard MINRES method only to store the required vectors, not taking into account

the system matrix and preconditioner. The results are shown in Table 6.4. The quantities refer to the overall memory consumption in megabytes during the process of solving the equation system with the respective method. We see that even with the construction of the reduced equation (3.5), the memory requirements stay well below those of the competing approaches.

TABLE 6.4
Example 6.2. Memory requirements of all considered methods.

n	β	Memory (MB)		
		SYS2MATEQ	LRMINRES	MINRES
1089	1e-1	4.46	8.11	15.39
	1e-3	8.06	6.14	15.39
	1e-5	17.58	4.54	15.39
4225	1e-1	8.58	19.21	60.46
	1e-3	12.57	19.35	60.46
	1e-5	24.36	16.15	60.46
16641	1e-1	23.50	71.74	241.84
	1e-3	31.86	80.56	241.84
	1e-5	51.91	205.85	241.84

6.2. Setup (ii): Partial observation on heat equation. Until now we only investigated the case where $M = M_1$ and therefore the matrix A_2 is the identity. One highly interesting and challenging problem is the optimization under partial observation, meaning the desired state is only of interest in certain measurement areas on a partial domain Ω_1 . This leads to a singular matrix M_1 , which further increases the difficulty of the PDE-constrained optimization problem. When we set up the rational Krylov subspace with only A_1 for this setting, we do not reach convergence. Therefore, we add up to two new vectors in each iteration as outlined in section 4.

Example 6.3. We solved the problem with control parameter $\beta = 10^{-4}$ and 100 time steps on a grid of 1089 spatial degrees of freedom. In Figure 6.3a we see the desired state for this problem, and the white area is an example of 100 nonobserved nodes, roughly 10 %. Hence, the corresponding entries in M_1 are set to 0. Figures 6.3b and 6.3c show the resulting state and control, respectively, for a fixed time step $t = 25$. Table 6.5 shows the results when increasing the number of unobserved nodes from $n_0 = 0$ to $n_0 = 900$, which is about 90% of the nodes. Here, the constructed subspaces are far from optimal as the resulting solution rank r is much smaller than the achieved subspace size p in the first column of Table 6.5. Thus, we make use of the truncation modification outlined in section 4. We see that the time and iterations needed to solve this more challenging problem are higher than those for the fully observed case in the table's first row. For the nontruncated case with full V_p , the columns denoted by p for the subspace size and r for the solution rank show a discrepancy which disappears with stricter truncation. When truncating with a tolerance of 10^{-10} both values coincide for all cases. In some cases applying this truncation greatly increases the number of iterations needed to find a solution, but the end result is a higher memory reduction. Thus, this option can be toggled to better reflect the desired behavior.

Unfortunately, the LRMINRES scheme from [26] is not adapted to the case of $M \neq M_1$. Hence, we have no other low-rank method to compare the performance of our method with.

FIG. 6.3. Example 6.3. Solution at a fixed time instant for a partial observation.

TABLE 6.5

Example 6.3. Results for different levels of partial observation with subspace truncation. We compare the number of iterations, the resulting subspace size p , and the solution rank r .

n_0	time (s)	Full V_p			Truncated V_p , 10^{-12}				Truncated V_p , 10^{-10}			
		iters	p	r	time (s)	iters	p	r	time (s)	iters	p	r
0	0.06	7	7	7	0.05	7	7	7	0.06	7	7	7
100	0.67	19	31	29	0.67	19	30	29	0.69	19	23	23
300	1.46	26	42	34	1.41	26	35	34	1.20	28	26	26
500	2.55	32	51	37	2.52	34	38	36	24.41	142	29	29
700	1.50	26	42	37	1.59	27	38	36	2.46	42	29	29
900	0.82	21	34	34	0.84	21	33	33	0.75	21	25	25

6.3. Setup (iii): Boundary control. Another setup of high interest is having a nondistributed control. Here, we take a boundary control problem as an example. This leads to the control u not being distributed on Ω but only on a smaller part of the domain Ω_b . Therefore, we get a smaller mass matrix $M_u \in \mathbb{R}^{n_b \times n_b}$ associated with U present on $n_b < n$ spatial nodes, and $N \in \mathbb{R}^{n \times n_b}$ is now rectangular (tall). Therefore, we have a modification of (2.32) as

$$(6.3) \quad A_1 X + X C_1 + X I_0 + A_3 X D - F_1 F_2^T = 0$$

with $A_1 = M^{-1}K$ and $A_3 = M^{-1}N M_u^{-1} N^T$. Here, the subspace we compute is derived from A_1 and A_3 as in (4.2).

Example 6.4. As before, we use $n_T = 100$ time steps and a convergence threshold of 10^{-4} for this setup. Results with a constant-in-time desired state for different levels of spatial discretization and different values for β are displayed in Table 6.6. SYS2MATEQ produces robust results with respect to the discretization size as we can only see a small increase in the ranks throughout discretization. Additionally, the results are acquired within a short amount of time even for small β . For larger discretization sizes the required time grows only slightly, as opposed to LRMINRES, where time requirements increase substantially for larger discretization sizes.

TABLE 6.6

Example 6.4. Comparisons between the new SYS2MATEQ and LRMINRES for a boundary control problem with regard to the subspace size p , the solution rank r , and required CPU time.

n β	method	1089			4225			16641		
		p	r	time (s)	p	r	time (s)	p	r	time (s)
10^{-1}	SYS2MATEQ	72	47	3.92	80	47	7.33	80	47	16.35
	LRMINRES	29	15	29.47	28	16	113.99	28	16	496.70
10^{-3}	SYS2MATEQ	64	46	3.68	68	46	5.08	74	46	14.35
	LRMINRES	25	12	12.70	24	14	35.53	24	14	158.82
10^{-5}	SYS2MATEQ	104	45	12.36	116	44	18.95	107	44	28.48
	LRMINRES	21	16	34.20	21	16	142.83	20	16	694.93

6.4. Setup (iv): Nonsymmetric convection-diffusion PDE. Until now we only investigated the performance of our method under the constraint of a heat equation. But for more general PDEs our method works just as well, e.g., the convection-diffusion equation,

$$(6.4) \quad \dot{y} = \varepsilon \nabla^2 y - v \cdot \nabla y + u,$$

with diffusion coefficient ε and velocity field v . When $\varepsilon \ll 1$, the equation is convection-dominated, and solving it is a challenging task. As the stiffness matrix K of the resulting equation system is nonsymmetric, we have to adjust the matrix equation accordingly. Thus, we get another modification of (2.32) for this setup as

$$(6.5) \quad A_1 X \begin{bmatrix} \mathcal{I} & 0 \\ 0 & \mathcal{I} \end{bmatrix} + M^{-1} K^T X \begin{bmatrix} 0 & 0 \\ 0 & \mathcal{I} \end{bmatrix} + X C_1 + A_2 X I_0 + A_3 X D - F_1 F_2^T = 0.$$

Example 6.5. We consider Example 3.1.4 from [23] with a recirculating wind $v(x, y) = (2y(1 - x^2), -2x(1 - y^2))$ as the underlying model. The velocity field v is displayed in Figure 6.4a. In Figures 6.4b–6.4c the desired state and a snapshot of a control solution for $\beta = 10^{-5}$ are displayed for $n = 4225$ and $n_T = 100$. The SYS2MATEQ method produces reliable results throughout a range of different values for ε and β as reported in Table 6.7. Even very small values in ε do not pose a problem to the solver.

FIG. 6.4. Example 6.5. Solution at a fixed time instant for convection-diffusion problem.

TABLE 6.7

Example 6.5. Results for different diffusion coefficients ε for the convection-diffusion problem with different parameters β and different spatial discretizations with regard to required subspace size p , solution rank r , and CPU time.

n	β	ε	4225			16641			66049		
			p	r	time	p	r	time	p	r	time
10^{-1}	10^{-0}	10^{-0}	2	2	0.18	2	2	0.37	2	2	4.44
		10^{-1}	10	10	0.27	8	8	0.79	2	2	3.07
		10^{-2}	22	22	0.72	18	18	2.31	10	10	7.78
		10^{-3}	24	24	0.91	18	18	3.15	10	10	11.04
10^{-3}	10^{-0}	10^{-0}	12	12	0.44	14	14	1.67	10	10	9.92
		10^{-1}	18	18	0.53	16	16	1.71	18	18	14.63
		10^{-2}	6	6	0.14	4	4	0.49	4	4	3.10
		10^{-3}	56	43	5.82	6	6	1.49	4	4	6.00
10^{-5}	10^{-0}	10^{-0}	52	23	4.47	62	22	13.45	46	20	42.52
		10^{-1}	52	22	4.67	46	20	7.92	38	19	32.55
		10^{-2}	6	6	0.12	4	4	0.51	4	4	3.03
		10^{-3}	6	6	0.19	4	4	1.15	4	4	5.90

7. Conclusion and outlook. We proposed a new scheme to solve a class of large-scale PDE-constrained optimization problems in a low-rank format. This method relied on the reformulation of the KKT saddle-point system into a matrix equation, which was subsequently projected onto a low dimensional space generated with rational Krylov-type iterations. We showed that the method's convergence is robust with respect to different discretizations and parameters. Furthermore, we demonstrated higher memory and time savings compared to an established low-rank scheme. Additionally, SYS2MATEQ is very flexible with respect to different constraints such as nonsymmetric PDE operators and partial state observation.

In the future, we plan on further investigating the subspace choice and the performance of our scheme for other challenging setups. Further improvements are expected from realizing a truncation or restarting mechanism in cases where the subspaces become unexpectedly large.

REFERENCES

- [1] W. BANGERTH, R. HARTMANN, AND G. KANSCHAT, *deal.II—a general-purpose object-oriented finite element library*, ACM Trans. Math. Software, 33 (2007), 24, <https://doi.org/10.1145/1268776.1268779>.
- [2] P. BENNER, A. ONWUNTA, AND M. STOLL, *Block-diagonal preconditioning for optimal control problems constrained by PDEs with uncertain inputs*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 491–518, <https://doi.org/10.1137/15M1018502>.
- [3] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137, <https://doi.org/10.1017/S0962492904000212>.
- [4] L. T. BIEGLER, O. GHATTAS, M. HEINKENSCHLOSS, AND B. VAN BLOEMEN WAANDERS, *Large-scale PDE-constrained optimization: An introduction*, in Large-Scale PDE-Constrained Optimization, Springer, New York, 2003, pp. 3–13, <https://doi.org/10.1007/978-3-642-55508-4>.
- [5] T. BREITEN, V. SIMONCINI, AND M. STOLL, *Fast iterative solvers for fractional differential equations*, Electron. Trans. Numer. Anal., 45 (2016), pp. 107–132.
- [6] A. CICHOCKI, A.-H. PHAN, Q. ZHAO, N. LEE, I. V. OSELEDTS, M. SUGIYAMA, AND D. MANDIC, *Tensor networks for dimensionality reduction and large-scale optimizations. Part 2. Applications and future perspectives*, Found. Trends Mach. Learn., 9 (2017), pp. 431–673, <https://doi.org/10.1561/22000000067>.
- [7] S. DOLGOV, J. W. PEARSON, D. V. SAVOSTYANOV, AND M. STOLL, *Fast tensor product solvers for optimization problems with fractional differential equations as constraints*, Appl. Math.

- Comput., 273 (2016), pp. 604–623, <https://doi.org/10.1016/j.amc.2015.09.042>.
- [8] V. DRUSKIN AND V. SIMONCINI, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems Control Lett., 60 (2011), pp. 546–560, <https://doi.org/10.1016/j.sysconle.2011.04.013>.
- [9] M. FISHER, J. NOCEDAL, Y. TRÉMOLET, AND S. J. WRIGHT, *Data assimilation in weather forecasting: A case study in PDE-constrained optimization*, Optim. Engrg., 10 (2009), pp. 409–426, <https://doi.org/10.1007/s11081-008-9051-5>.
- [10] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002, <https://doi.org/10.1137/1.9780898718027>.
- [11] M. HINZE, R. PINNAU, M. ULBRICH, AND S. ULBRICH, *Optimization with PDE Constraints*, Math. Model. Theory Appl. 23, Springer-Verlag, New York, 2009, <https://doi.org/10.1007/978-1-4020-8839-1>.
- [12] K. ITO AND K. KUNISCH, *Lagrange Multiplier Approach to Variational Problems and Applications*, Adv. Des. Control 15, SIAM, Philadelphia, 2008, <https://doi.org/10.1137/1.9780898718614>.
- [13] M. L. JUNCOSA AND D. YOUNG, *On the Crank-Nicolson procedure for solving parabolic partial differential equations*, Proc. Cambridge Philos. Soc., 53 (1957), p. 448–461, <https://doi.org/10.1017/S0305004100032436>.
- [14] H. MINGYOU AND V. THOMÉE, *On the backward Euler method for parabolic equations with rough initial data*, SIAM J. Numer. Anal., 19 (1982), pp. 599–603, <https://doi.org/10.1137/0719040>.
- [15] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, <https://doi.org/10.1137/0712047>.
- [16] D. PALITTA, *Matrix equation techniques for certain evolutionary partial differential equations*, J. Sci. Comput., 87 (2021), 99, <https://doi.org/10.1007/s10915-021-01515-x>.
- [17] 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>.
- [18] C. E. POWELL, D. SILVESTER, AND V. SIMONCINI, *An efficient reduced basis solver for stochastic Galerkin matrix equations*, SIAM J. Sci. Comput., 39 (2017), pp. A141–A163, <https://doi.org/10.1137/15M1032399>.
- [19] T. REES, H. S. DOLLAR, AND A. J. WATHEN, *Optimal solvers for PDE-constrained optimization*, SIAM J. Sci. Comput., 32 (2010), pp. 271–298, <https://doi.org/10.1137/080727154>.
- [20] T. REES AND M. STOLL, *Block-triangular preconditioners for PDE-constrained optimization*, Numer. Linear Algebra Appl., 17 (2010), pp. 977–996, <https://doi.org/10.1002/nla.693>.
- [21] O. SCHENK, M. MANGUOLU, A. SAMEH, M. CHRISTEN, AND M. SATHE, *Parallel scalable PDE-constrained optimization: Antenna identification in hyperthermia cancer treatment planning*, Comput. Sci. Res. Develop., 23 (2009), pp. 177–183, <https://doi.org/10.1007/s00450-009-0080-x>.
- [22] J. SCHÖBERL AND W. ZULEHNER, *Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 752–773, <https://doi.org/10.1137/060660977>.
- [23] D. SILVESTER, H. ELMAN, AND A. WATHEN, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, UK, 2005, <https://doi.org/10.1093/acprof:oso/9780199678792.001.0001>.
- [24] V. SIMONCINI, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288, <https://doi.org/10.1137/06066120X>.
- [25] V. SIMONCINI, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441, <https://doi.org/10.1137/130912839>.
- [26] 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>.
- [27] F. TRÖLTZSCH, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, Grad. Stud. Math. 112, AMS, Providence, RI, 2010, <https://doi.org/10.1090/gsm/112>.

8 A low-rank tensor method to reconstruct sparse initial states for PDEs with Isogeometric Analysis

This manuscript was submitted in September 2021 and a preprint is available at arXiv:2109.03119 [17].

Novelty of this work. We examine a recently proposed method for solving inverse problems with a sparsity inducing Bayesian *hyperprior* model [18] for the use of reconstructing sparse initial states of a PDE from noisy measurements. For a PDE context, the method requires a large number of PDE solves and the computation of potentially large solution vectors. Instead of solving a quadratic minimization with e.g. CGLS in every iterative step of the method, we propose reformulating the system into a PDE-constrained optimization problem and applying the methods proposed in or previous work [15].

Results from the paper. For the context of PDE governed inverse problems, the parameter-to-observable map can usually not be pre-computed. Here, our method provides an efficient alternative to apply the sparsity promoting hyperprior model. We get a PDE-constrained optimization problem with variable regularization parameters which are iteratively updated. Solving the upcoming KKT systems in the Tensor Train format reliably provides the desired sparse initial states even for large-scale systems where the application of traditional methods would prove difficult.

Contribution of the author. The author of this thesis has provided around 90% of the work for this manuscript.

Related works. Other methods to promote sparsity in inverse problems mostly rely on ℓ_p -Tikhonov regularization with $1 \leq p \leq 2$ [41, 54], or include regularization

with non-convex constraints, which promote sparsity more strongly [103]. For PDE problems, most authors combine these regularization methods with classical PDE-constrained optimization methods, e.g. [96, 84, 66].

A low-rank tensor method to reconstruct sparse initial states for PDEs with Isogeometric Analysis

Alexandra Bünger* and Martin Stoll†

Abstract. When working with PDEs the reconstruction of a previous state often proves difficult. Good prior knowledge and fast computational methods are crucial to build a working reconstruction. We want to identify the heat sources on a three dimensional domain from later measurements under the assumption of small, distinct sources, such as hot chippings from a milling tool. This leads us to the need for a Prior reflecting this a priori information. Sparsity-inducing hyperpriors have proven useful for similar problems with sparse signal or image reconstruction. We combine the method of using a hierarchical Bayesian model with gamma hyperpriors to promote sparsity with low-rank computations for PDE systems in tensor train format.

Key words. Isogeometric Analysis, low rank decompositions, tensor train format, Bayesian inverse problem, sparse reconstruction

AMS subject classifications. 65F10, 65F50, 15A69, 35R30

1. Motivation. The reconstruction of sparse sources from noisy observation of a PDE model plays a significant role in many applications, varying from identifying heat sources [20] to finding the origin of a tumor growth [31]. Incorporating prior information about the sparsity of an unknown source into its recovery from noisy data arose as long as 40 years ago with first successful applications in geophysics, astrophysics, and ultrasonic imaging [9].

The general idea of using ℓ_p Tikhonov regularization techniques has since become highly popular to promote sparsity and has been extensively researched (cf. [13, 18] and references therein), mainly focusing on $1 \leq p \leq 2$ due to their global convexity properties. It has been shown that an ℓ_1 penalty can provide accurate sparse reconstructions [6] or even the sparsest possible solution under some additional assumptions [10]. Stronger sparsity promotion with $p < 1$ comes at the expense of global convexity and is therefore still an active research topic [32].

In the Bayesian framework, sparsity promotion can be efficiently modeled by considering a component-wise Gaussian prior model where the variances themselves are modeled as random variables underlying a sparsity promoting *hyperprior* model [5]. In this paper we will examine the application of a hybrid method using such hypermodels with gamma hyperpriors presented in [4]. The algorithm strongly promotes sparsity while retaining global convergence by switching between two models where the first is globally convex to drive the solution close to a unique minimizer and the other is more greedy towards stronger sparsity while suffering from local minima.

Let us assume a PDE model with Dirichlet boundary conditions on a 2D or 3D domain

*Technische Universität Chemnitz, Department of Mathematics, Chair of Scientific Computing, 09107 Chemnitz, Germany, alexandra.buenger@mathematik.tu-chemnitz.de

†Technische Universität Chemnitz, Department of Mathematics, Chair of Scientific Computing, 09107 Chemnitz, Germany, martin.stoll@mathematik.tu-chemnitz.de

Ω as

$$(1.1) \quad \frac{\partial}{\partial t} y(x, t) = \mathcal{L}y(x, t) \quad \text{on } \Omega \times (0, T],$$

$$(1.2) \quad y(x, t = 0) = u_0(x) \quad \text{on } \Omega \times 0,$$

$$(1.3) \quad y(x, t) = y_0(x) \quad \text{on } \partial\Omega \times (0, T],$$

with a PDE operator $\mathcal{L}$ such as $\mathcal{L}y(x, t) = \Delta y(x, t)$ for heat conduction. Here, $y(x, t)$ is the state at time t and $u_0(x, t)$ the respective initial state at time $t = 0$. Both are time-dependent functions on the domain Ω and time-frame $(0, T]$ with end time T .

We consider being given a noisy observation at the end time T stemming from the evolution via the PDE model (1.1)–(1.3),

$$(1.4) \quad z(x) = y(x, T) + e.$$

Our goal is to recover the initial state under the presence of additive noise e , e.g., measurement errors, and the prior assumption of u_0 being sparse.

In this paper, we will study a Bayesian approach to the problem of reconstructing u_0 under the assumption of Gaussian additive noise and a sparsity inducing prior formulation. We make use of the methods developed in [4] and tailor them to our model in section 2. The original algorithm requires a large number of matrix-vector multiplications with the parameter-to-observable map, which in our case includes a complete solution of the governing PDE and thus can be infeasibly large and dense.

The PDE model can be discretized in space and time into a linear equation system using some time-stepping method and a Galerkin type discretization, which we will introduce in section 3. We introduce the space discretization using an Isogeometric Analysis (IGA) scheme in subsection 3.1 and the full space-time discretization in subsection 3.2. As the resulting discretization would be large and dense, we propose a low-rank approximation method for the IGA discretization in section 4 and a suitable solver tailored to working with low-rank tensor structured data in section 5 to perform matrix-vector multiplications or solving the arising KKT system without computing the matrices directly. The resulting numerical scheme profits from small storage requirements and robustness allowing to compute even very large-scale problems, which would prove infeasible with traditional methods. In section 6 we provide numerical examples with an exemplary 2D problem and 3D models to show the effectiveness and robustness of the resulting method comparing different hyperprior setups.

2. Sparsity inducing hyper prior. We will first briefly review the method for sparsity inducing hyperpriors as the local hybrid IAS (iterative alternating sequential) algorithm introduced in [4] and later extended in [3] from a Bayesian point of view.

We consider the linear observation model with additive Gaussian noise resulting from the PDE problem with uncertain initial values u_0 ,

$$(2.1) \quad z = Au_0 + e, \quad e \sim \mathcal{N}(0, \Sigma),$$

where $A \in \mathbb{R}^{n \times m}$ is the parameter-to-observable map, which transfers the uncertain inputs u_0 to the observable outputs y , and $\Sigma \in \mathbb{R}^{m \times m}$ is a symmetric positive semidefinite covariance

matrix. Without loss of generality, we assume the noise to be white, i.e. $\Sigma = I$. This gives the likelihood probability density function (pdf)

$$(2.2) \quad \pi_{z|u_0}(z|u_0) \propto \exp\left(-\frac{1}{2}\|Au_0 - z\|^2\right).$$

We believe the unknown u_0 to be sparse. To incorporate this into the prior model, we consider a component-wise Gaussian prior model,

$$(2.3) \quad u_0 \sim \mathcal{N}(0, D_\theta), \quad D_\theta = \text{diag}(\theta_1, \dots, \theta_n) \in \mathbb{R}^{n \times n},$$

where the variances θ are not known. Thus, θ itself are random variables where smaller variances promote values closer to zero. The a priori belief about θ is thus incorporated into a *hyperprior* pdf $\pi_\theta(\theta)$ leading to a hierarchical prior model. The conditional prior density of u_0 with given θ now has the form

$$(2.4) \quad \pi_{u_0|\theta}(u_0|\theta) \propto \frac{1}{\prod_{j=1}^n \sqrt{\theta_j}} \exp\left(-\frac{1}{2}\|D_\theta^{-1/2}u_0\|^2\right) = \exp\left(-\frac{1}{2}\|D_\theta^{-1/2}u_0\|^2 - \frac{1}{2}\sum_{j=1}^n \log \theta_j\right).$$

Now we need to estimate θ as well as u_0 and chose an appropriate prior density for its representation. One class of hyperpriors π_θ to promote the sparsity of our solution is that of generalized gamma distributions,

$$(2.5) \quad \pi_\theta(\theta) = \pi_\theta(\theta|r, \beta, \vartheta) = \frac{|r|^n}{\Gamma(\beta)^n} \prod_{j=1}^n \frac{1}{\vartheta_j} \left(\frac{\theta_j}{\vartheta_j}\right)^{r\beta-1} \exp\left(-\left(\frac{\theta_j}{\vartheta_j}\right)^r\right),$$

with $r \in \mathbb{R} \setminus \{0\}, \beta > 0, \vartheta > 0$. This family of priors allows large outliers in θ while overall favoring small values [4].

With this we not only estimate x_0 but θ as well, based on their joint posterior distribution,

$$(2.6) \quad \pi_{u_0, \theta}(u_0, \theta|y) \propto \pi_{u_0|\theta}(u_0|\theta) \pi_\theta(\theta) \pi_{z|u_0}(z|u_0).$$

To find a suitable sparse reconstruction for u_0 , we need to compute the maximum a posteriori (MAP) estimate, which is the minimizer of the negative logarithm of the posterior pdf (2.6),

$$(2.7) \quad (u_0^*, \theta^*) = \arg \min_{u_0, \theta} \mathcal{F}(u_0, \theta|r, \beta, \vartheta)$$

$$(2.8) \quad = \arg \min_{u_0, \theta} \frac{1}{2}\|z - Au_0\|^2 + \frac{1}{2}\sum_{j=1}^n \frac{u_{0,j}^2}{\theta_j} - \left(r\beta - \frac{3}{2}\right) \sum_{j=1}^n \log \frac{\theta_j}{\vartheta_j} + \sum_{j=1}^n \left(\frac{\theta_j}{\vartheta_j}\right)^r.$$

Note that the objective function $\mathcal{F}(u_0, \theta)$ consists of four terms of which one depends on u_0 only, two depend only on θ and one is dependent on both variables. This gives rise to an hybrid algorithm alternating between updates of u_0 and θ .

The authors of [4] suggest using an Iterative Alternating Sequential algorithm (IAS) for this computation. Here, each iteration consists of two updates,

$$(2.9) \quad u^i, \theta^i \rightarrow u^{i+1} \rightarrow \theta_{i+1},$$

where we first fix θ to update u^{i+1} and afterwards fix u^{i+1} to update θ ,

$$(2.10) \quad u^{i+1} = \arg \min_u \{\mathcal{F}(u, \theta^i)\}, \quad \theta^{i+1} = \arg \min_\theta \{\mathcal{F}(u^{i+1}, \theta)\}.$$

Both minimizations have an exact condition for the minimizer and are relatively simple to compute. The update of u results in the solution of a quadratic minimization problem, i.e.,

$$(2.11) \quad u^{i+1} = \arg \min_u \|Au - z\|^2 + \|D_\theta^{-1/2} u\|^2.$$

To approximate the solution, we can solve (2.11) as it is or solve a reduced least squares problem. For this we introduce a change of variables as $w = D_\theta^{-1/2} u$ and find the least squares solution of

$$(2.12) \quad AD_\theta^{1/2} w = z.$$

This can be easily solved, e.g. by a CGLS (Conjugate Gradient Least Squares) algorithm, as suggested in [3]. We will later discuss the structure of A for our PDE model and propose a low rank method to efficiently solve this minimization problem.

To then update θ we can compute its components θ_j independently from each other. For each component the first order optimality conditions applied to (2.8) read

$$(2.13) \quad 0 = \frac{\partial \mathcal{F}}{\partial \theta_j} = -\frac{1}{2} \frac{u_j^2}{\theta_j^2} - \left(r\beta - \frac{3}{2}\right) \frac{1}{\theta_j} + r \frac{\theta_j^{r-1}}{\vartheta_j^r}.$$

To numerically solve this equation, [4] suggests the change of variables $\theta_j = \vartheta_j \xi_j$, $u_j = \sqrt{\vartheta_j} z_j$. Then, we can write $\xi_j = \varphi(|z_j|)$ and we get the initial value problem

$$(2.14) \quad \varphi'(z) = \frac{2z\varphi(z)}{2r^2\varphi(z)^{r+1} + z^2}, \quad \varphi(0) = \left(\frac{\eta}{r}\right)^{1/r}.$$

From this the update for θ_j can be computed with some numerical time integrator.

Alternating between updating u and updating θ gives rise to the standard IAS algorithm.

2.1. Hyperprior parameters and hierarchical IAS algorithm. The effectiveness of the alternating scheme depends highly on the choice of parameters for the hyperprior, r , β and ϑ . A detailed analysis of these parameters is available in [4], which we want to summarize now. We additionally analyze the choice of parameters in our numerical experiments in Section 6.

The choice of r and β affects the sparsity of the solution and determines the convexity of the objective function while ϑ can be seen as a sensitivity scaling where $\vartheta_j = \frac{C}{\|Ae_j\|^2}$ with some constant $C > 0$ (cf. [3]). The constant C should depend on the signal-to-noise ratio (SNR).

Calvetti et al. propose two modifications to the general IAS algorithm to update the hypermodel component-wise and ensure convexity. When $r > 1$ and β follows $r\beta > 3/2$ the objective function is globally convex. Additionally, the objective function is convex for $0 < r < 1$, $r\beta > 3/2$ or $r < 0$ and $\beta > 0$ when $\theta_j < \vartheta_j(\frac{\eta}{r|r-1|})$. This can be exploited by

choosing two hypermodels M_1 and M_2 , where M_1 satisfies the global convexity conditions and M_2 is the desired sparsity promoting model.

Global hybrid IAS. The first method proposed starts the iteration with model M_1 and carries out a switch to model M_2 for all parameters at once after a certain number of iterations. The first iterations drive the objective function towards the global minimum and afterwards we trade global convexity for stronger sparsity promotion. The algorithm is outlined in [Algorithm 2.1](#).

Local hybrid IAS. The second proposition starts with model M_2 as well but updates the models locally by switching individual components of θ_j to the other model M_2 when θ_j is below the threshold for convexity. To ensure preservation of convexity an additional bound constraint is applied to components, which were switched to model M_2 . [Algorithm 2.2](#) illustrates the iteration for this method.

We will consider both methods in our experiments for a gamma ($r = 0.5$) and inverse gamma ($r = -1$) hyperprior model.

Algorithm 2.1 Global hybrid IAS

- 1: Input: noisy z , parameter-to-observable map A , noise covariance matrix Σ , hypermodel M_1 with $(r_1, \beta_1, \vartheta_1)$, hypermodel M_2 with $(r_2, \beta_2, \vartheta_2)$, switch point $i_s > 0$,
 - 2: Set $\theta_0 = \vartheta_1$
 - 3: **for** $i = 1, \dots, \text{maxIter}$ or until convergence **do**
 - 4: Solve $\min_{u_i} \|Au_i - z\|^2 + \|D_\theta^{-1/2} u_i\|^2$
 - 5: **if** $i < i_s$ **then**
 - 6: Update θ with u_i and parameters from M_1
 - 7: **else**
 - 8: Update θ with u_i and parameters from M_2
 - 9: **end if**
 - 10: **end for**
 - 11: Output: estimated initial values $u_0 = u_i$ and variance θ
-

3. Space-time discretization of A . To use the method described in [section 2](#) we need a representation of the parameter-to-observable map A . For our PDE model [\(1.1\)–\(1.3\)](#), we need an appropriate discretization to build this mapping. We will now introduce Isogeometric Analysis for the space discretization in [subsection 3.1](#) and the time-discretization with an implicit Euler scheme in [subsection 3.2](#).

For the space discretization a Finite Element Method (FEM) [\[11\]](#) is the most prominently used choice and has seen a wide range of applications in both engineering and science [\[27\]](#). But recently a new approach called Isogeometric Analysis (IGA) [\[7\]](#) has gained popularity. This approach arose from the desire to integrate numerical analysis directly into CAD (computer aided design) tools [\[7\]](#). It abolishes the need for time consuming meshing steps and preprocessing needed in classical Finite Element Analysis [\[1\]](#) in addition to working on exact geometric representations instead of an approximation.

Furthermore, this method admits a tensor-product based structure which gives rise to a very effective low-rank method developed in [\[22\]](#) and refined in [\[21, 28\]](#). This has been shown

Algorithm 2.2 Local hybrid IAS

```

1: Input: noisy  $z$ , parameter-to-observable map  $A$ , noise covariance matrix  $\Sigma$ , hypermodel
    $M_1$  with  $(r_1, \beta_1, \vartheta_1)$ , hypermodel  $M_2$  with  $(r_2, \beta_2, \vartheta_2)$ 
2: Set  $\theta_0 = \vartheta_1$ ,  $I = \emptyset$ 
3: for  $i = 1, \dots, \text{maxIter}$  or until convergence do
4:   Solve  $\min_{u_i} \|Au_i - z\|^2 + \|D_\theta^{-1/2}u_i\|^2$ 
5:   Apply bound constraint to  $u_{i,j}$  with  $j \in I$ 
6:   for  $j = 1, \dots, n$  do
7:     if  $\theta_j \geq \vartheta_j(\frac{\eta}{r[r-1]})$  then
8:       update  $\theta_j$  with model  $M_1$ 
9:     else
10:      update  $\theta_j$  with model  $M_2$ 
11:       $i = I \cup \{j\}$ 
12:    end if
13:  end for
14: end for
15: Output: estimated initial values  $u_0 = u_i$  and variance  $\theta$ 

```

to be very efficient in combination with tensor train calculations [24] in our recent work on PDE-constrained optimal control problems [2].

3.1. Isogeometric Analysis discretization. Isogeometric Analysis benefits from the use of high order spline functions, such as B-splines and NURBS, as their basis. A geometric domain is represented exactly using a set of such splines, which are subsequently used to build a solution space for solving a PDE problem [17]. In this paper we will focus on the use of B-splines but note that a generalization to other popular spline spaces, such as NURBS (Non-uniform rational B-splines) [26, 12], is possible. The notation and derivation in this section as well as section 4 follow our work in [2].

A set of n B-splines is uniquely defined by choosing a vector $\xi = \{\hat{x}_1, \dots, \hat{x}_{n+p+1}\}$, called the open knot vector, with

$$(3.1) \quad 0 = \hat{x}_1 = \dots = \hat{x}_{p+1} < \hat{x}_{p+2} \leq \dots \leq \hat{x}_n < \hat{x}_{n+1} = \dots = \hat{x}_{n+p+1} = 1,$$

and a degree p , where the first and last knot are repeated $p+1$ times and for all other knots duplicate knots are allowed up to multiplicity p . The parameter n determines the number of resulting B-splines $\beta_{i,p}$ with $i = 1, \dots, n$.

For each knot vector ξ as in (3.1), the according B-splines $\beta_{i,p}$ of degree p with $i = 1, \dots, n$ are uniquely defined by the recursion

$$(3.2) \quad \beta_{i,0}(\hat{x}) = \begin{cases} 1 & \text{if } \hat{x}_i \leq \hat{x} < \hat{x}_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$

$$(3.3) \quad \beta_{i,j}(\hat{x}) = \frac{\hat{x} - \hat{x}_i}{\hat{x}_{i+j} - \hat{x}_i} \beta_{i,j-1}(\hat{x}) + \frac{\hat{x}_{i+j+1} - \hat{x}}{\hat{x}_{i+j+1} - \hat{x}_{i+1}} \beta_{i+1,j-1}(\hat{x}),$$

where $j = 1, 2, \dots, p$ and $i = 1, \dots, n$. Each resulting B-spline $\beta_{i,p}$ has the local support $[\hat{x}_i, \hat{x}_{i+p+1}]$, see Figure 1 for an example. We use $\mathbb{S}_\xi^p$ to denote the spline space spanned by the B-splines with degree p and knot vector ξ ,

$$(3.4) \quad \mathbb{S}_\xi^p = \text{span}\{\beta_{1,p}, \dots, \beta_{n,p}\}.$$

Figure 1: B-spline space with degree $p = 2$ and knot vector $\xi = [0, 0, 1/3, 1/2, 2/3, 1, 1]$.

To construct a space of D -dimensional basis functions $\mathbb{S}^D$, for $d = 1, \dots, D$ we define one-dimensional spline spaces $\mathbb{S}_{\xi_d}^{p_d}$ and vector valued functions $B^{(d)}(\hat{x}^{(d)})$, as the vectors holding all basis functions of dimension d ,

$$(3.5) \quad B^{(d)}(\hat{x}^{(d)}) = [\beta_1^{(d)}(\hat{x}^{(d)}), \dots, \beta_{n_d}^{(d)}(\hat{x}^{(d)})],$$

with one-dimensional variables $\hat{x}^{(d)}$.

We combine these spline spaces via tensor product to get a D -dimensional spline space $\mathbb{S}^D = \mathbb{S}_{\xi_1}^{p_1} \otimes \dots \otimes \mathbb{S}_{\xi_D}^{p_D}$ with an order D tensor of size $(n_1, \dots, n_D)$ collecting all basis functions as

$$(3.6) \quad B(\hat{x}) = \bigotimes_{d=1}^D B^{(d)}(\hat{x}^{(d)}).$$

The tensor in (3.6) then is a function $B(\hat{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^{n_1 \times \dots \times n_D}$ with variables $\hat{x} = (\hat{x}^{(1)}, \dots, \hat{x}^{(D)})^T$ and its elements are the D -dimensional basis functions

$$(3.7) \quad \beta_{\mathbf{i}}(\hat{x}) = \prod_{d=1}^D \beta_{i_d}^{(d)}(\hat{x}^{(d)}),$$

with multi-index $\mathbf{i} \in I = \{(i_1, \dots, i_D) \mid i_d = 1, \dots, n_d, d = 1, \dots, D\}$.

Given such a basis $\mathbb{S}^D$, we define a B-spline geometry mapping $G : \hat{\Omega} \rightarrow \Omega$ from the D -dimensional unit cube $\hat{\Omega} := [0, 1]^D$ onto an arbitrary geometric shape $\Omega \subset \mathbb{R}^D$ as

$$(3.8) \quad G(\hat{x}) = \sum_{\mathbf{i} \in I} C_{\mathbf{i}} \beta_{\mathbf{i}}(\hat{x}) = C : B(\hat{x}),$$

with control points $C_{\mathbf{i}} \in \mathbb{R}^D$. Here $C : B(\hat{x})$ denotes the Frobenius product of the two tensors, $C \in \mathbb{R}^{D \times n_1 \times \dots \times n_D}$ holding all the control points and B from (3.6).

Now that we have a spline representation of the geometry Ω , we use the same splines to construct discrete functions $u_h \in V_h \subset H^1(\Omega)$ as approximations to the solutions $u \in H^1(\Omega)$ of the PDE problem in (1.1)–(1.3).

The isogeometric solution space V_h is built as

$$(3.9) \quad V_h = \text{span}\{\beta_{\mathbf{i}} \circ G^{-1} : \mathbf{i} \in I\},$$

with an index set I such that $\beta_{\mathbf{i}}$ are the elements of $\mathbb{S}^D$. The functions in V_h are linear combinations of the basis functions with coefficients $u_{\mathbf{i}}$,

$$(3.10) \quad u_h = \sum_{\mathbf{i} \in I} u_{\mathbf{i}} (\beta_{\mathbf{i}} \circ G^{-1}).$$

We use this space for the spatial Galerkin discretization of the PDE system (1.1)–(1.3). Exemplary, for the boundary value problem $u(x) = f(x)$ in Ω , the weak formulation results in the bilinear form

$$(3.11) \quad a_m(u, v) = \langle u, v \rangle_2 = \int_{\Omega} uv \, dx,$$

called the mass term. Discretization with (3.9) results in the discrete mass term

$$(3.12) \quad a_{m,h}(u_h, v_h) = \int_{\Omega} u_h(x) v_h(x) \, dx = \int_{\hat{\Omega}} \sum_{\mathbf{i} \in I} u_{\mathbf{i}} \beta_{\mathbf{i}}(\hat{x}) \sum_{\mathbf{j} \in I} v_{\mathbf{j}} \beta_{\mathbf{j}}(\hat{x}) \omega(\hat{x}) \, d\hat{x},$$

with $\omega(\hat{x}) = |\det \nabla G(\hat{x})|$. The basis functions $\beta_{\mathbf{j}}$ are linearly independent. Thus, similarly to FEM, we can rewrite the bilinear forms as a matrix-vector product Au with vectorization of the coefficient set $u_{\mathbf{i}}$, $\mathbf{i} \in I$, where A is realized as a mass matrix M with elements

$$(3.13) \quad M_{\mathbf{i}, \mathbf{j}} = \int_{\Omega} \beta_{\mathbf{i}} \beta_{\mathbf{j}} \omega \, d\hat{x}.$$

The same strategy is applied to any PDE operator $\mathcal{L}u(x)$, e.g. $\mathcal{L}u(x) = -\Delta u(x)$ results in the stiffness term

$$(3.14) \quad a_s(u, v) = - \int_{\Omega} (\Delta u) v \, dx = \int_{\Omega} \nabla u \cdot \nabla v \, dx,$$

$$(3.15) \quad a_{s,h}(u_h, v_h) = \int_{\Omega} (\nabla u_h(x)) \cdot \nabla v_h(x) \, dx = \int_{\hat{\Omega}} (Q(\hat{x}) \sum_{\mathbf{i} \in I} u_{\mathbf{i}} \nabla \beta_{\mathbf{i}}(\hat{x})) \cdot \sum_{\mathbf{j} \in I} v_{\mathbf{j}} \nabla \beta_{\mathbf{j}}(\hat{x}) \, d\hat{x},$$

with $Q(\hat{x}) = (\nabla G(\hat{x})^T \nabla G(\hat{x}))^{-1} |\det \nabla G(\hat{x})|$ (cf. [22]) resulting in the stiffness matrix K with elements

$$(3.16) \quad K_{\mathbf{i},\mathbf{j}} = \int_{\hat{\Omega}} (Q \nabla \beta_{\mathbf{i}}) \cdot \nabla \beta_{\mathbf{j}} \, d\hat{x} = \sum_{k,l=1}^D \int_{\hat{\Omega}} q_{k,l} \frac{\partial}{\partial \hat{x}_l} \beta_{\mathbf{i}} \frac{\partial}{\partial \hat{x}_k} \beta_{\mathbf{j}} \, d\hat{x}.$$

With these matrices we can now discretize the parameter-to-observable map A in a general way before introducing additional low-rank approximations for M and K exploiting the underlying tensor product structure of the basis functions to reduce the computational effort and storage requirements of our problem. This low-rank approach has been successfully applied before to an optimal control problem in [2].

3.2. Discrete parameter-to-observable map. Considering the PDE problem given in (1.1) – (1.3), we now have a space discretization, which we need to combine with a suitable method for time discretization. For this time-stepping method we choose an implicit Euler scheme [23] and N_t equidistant time steps of length τ to discretize the time frame $t \in (0, T]$ into steps $t_i = i\tau$ for $i = 1, \dots, N_t$. With this we get the fully discrete system

$$(3.17) \quad \frac{My_i - My_{i-1}}{\tau} = -Ky_i, \quad \text{for } i = 2, \dots, N_t$$

$$(3.18) \quad My_1 = -\tau Ky_1 + Mu_0$$

$$(3.19) \quad z = Cy_{N_t} + e.$$

Here, y_i is the vector of B-spline coefficients for the discretization of the function $y(t_i)$ and C denotes the observation operator (e.g. an identity for a full observation at N_t). We will denote the vectors collecting all time steps as e.g. $\mathbf{y} = [y_1, \dots, y_{N_t}]^T$.

We can write this system of equations as one large equation system,

$$(3.20) \quad \underbrace{\begin{bmatrix} \tau K + M & & & \\ -M & \tau K + M & & \\ & \ddots & \ddots & \\ & & -M & \tau K + M \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_{N_t} \end{bmatrix}}_{\mathbf{M}_0} = \underbrace{\begin{bmatrix} M \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\mathbf{M}_0} u_0,$$

$$(3.21) \quad z = \underbrace{\begin{bmatrix} 0 & \dots & 0 & C \end{bmatrix}}_{\mathbf{C}} \begin{bmatrix} y_1 \\ \vdots \\ y_{N_t} \end{bmatrix} + e.$$

Using this formulation we can denote the equation from u_0 to z , the parameter-to-observable map A , as

$$(3.22) \quad z = Au_0 := \mathbf{C}\mathbf{K}^{-1}\mathbf{M}_0 u_0.$$

This matrix A typically is dense and large, and even storing the full representation of $\mathbf{K}$ may pose difficulties. Thus, computing A explicitly is usually infeasible. We will present a scheme to efficiently compute matrix-vector products with A without forming any full matrices. For

this, let us observe the structure of the arising matrices. We can also express them with the following Kronecker products,

$$(3.23) \quad \mathbf{C} = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \otimes C,$$

$$(3.24) \quad \mathbf{K} = \tau I_{N_t} \otimes K + II \otimes M,$$

$$(3.25) \quad \mathbf{M}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \otimes M,$$

where I_{N_t} is the $N_t \times N_t$ identity, II is a bidiagonal matrix with 1 on the diagonal and -1 on its first subdiagonal¹.

Note that here II results from the implicit Euler scheme. This scheme can be replaced by a different time stepping method like a Crank-Nicolson scheme [19] or others in which case II and the identity I_{N_t} will become different block matrices. Using a different time-stepping method would thus still maintain the general Kronecker product structure of the equations and be applicable for the following steps.

Now that we know the structure of our parameter-to-observable map we can take a closer look at solving the minimization problem in (2.11). In [4], the authors suggest using a CGLS method [14]. This method is most suited if the underlying matrix A is sparse and can be precomputed (cf. [29]) as it requires a large number of matrix-vector products with A and A^T .

Unfortunately, in the case of the PDE solution (3.22) the resulting matrix will most likely be dense and potentially very large. Thus, constructing A in full may not be feasible. In the following chapter we propose a low rank method that can be used to either calculate the matrix-vector products needed for CGLS in a compact and efficient way without assembly of A , or to solve the optimization problem (2.11) by rearranging it into a constrained minimization problem.

Substituting Au_0 in (2.11) by $\mathbf{C}y$ and using the PDE as a constraint leads to

$$(3.26) \quad \min_{u_0} \|\mathbf{C}y - z\|^2 + \|D_\theta^{-1/2} u_0\|^2$$

$$(3.27) \quad \text{s.t. } \mathbf{K}y = \mathbf{M}_0 u_0.$$

Formulating its first order optimality conditions gives us the large scale saddle point system,

$$(3.28) \quad \begin{bmatrix} \mathbf{C}^T M \mathbf{C} & 0 & -\mathbf{K}^T \\ 0 & D_\theta^{-1/2} M D_\theta^{-1/2} & \mathbf{M}_0^T \\ -\mathbf{K} & \mathbf{M}_0 & 0 \end{bmatrix} \begin{bmatrix} y \\ u_0 \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M}_0 \\ 0 \\ 0 \end{bmatrix}.$$

This type of system has been extensively researched and can generally be solved with e.g. preconditioned iterative methods (cf. [25]) or variational formulations [15]. But due to its potential size we will focus on a low rank method exploiting the underlying structure of the system. Here, each block has a Kronecker product structure as described in (3.23)–(3.23), which can be exploited efficiently by a low rank in time method presented in [30]. To further reduce the computational complexity we will now take a look back at the structure of the matrices M and K and outline how we can solve the resulting Kronecker product based system with a low rank tensor train method.

¹Syntax for MATLAB: $II = \text{spdiags}([-ones(N_t, 1), ones(N_t, 1)], -1 : 0, N_t, N_t);$

4. Low Rank IGA. We see that the time discretization in (3.23)–(3.25) leads to a tensor product structure in the parameter-to-observable map. Using the method presented in [2] we can additionally approximate M and K as similar tensor products and exploit the resulting structure with low rank tensor train calculations. This exploit can either be utilized for the matrix-vector products in the CGLS scheme or to solve the saddle point system (3.28).

During the derivation of the mass and stiffness matrices, we did not pay attention to the tensor product structure of $\mathbb{S}^D$. We can either arrange M and K as matrices or as tensors of size $(\mathbf{n}, \mathbf{n}) = (n_1, \dots, n_D, n_1, \dots, n_D)$. With this tensor notation the mass and stiffness matrices in a multi-dimensional setting are represented in a compact way. We can write the mass term as a tensor M

$$(4.1) \quad M = \int_{\hat{\Omega}} \omega B \otimes B \, d\hat{x} \in \mathbb{R}^{\mathbf{n} \times \mathbf{n}},$$

with elements coming from (3.13). The stiffness term can be treated similarly. With the tensor gradient we can write it as a tensor K

$$(4.2) \quad K = \int_{\hat{\Omega}} [Q \cdot (\nabla \otimes B)] \cdot (\nabla \otimes B) \, d\hat{x} \in \mathbb{R}^{\mathbf{n} \times \mathbf{n}}.$$

$$(4.3) \quad = \sum_{k,l=1}^D \int_{\hat{\Omega}} q_{k,l} \frac{\partial}{\partial \hat{x}_l} B \otimes \frac{\partial}{\partial \hat{x}_k} B \, d\hat{x}$$

whose elements are of the form in (3.16). The associated mass and stiffness matrices are obtained by reordering the indices since the elements of the mass and stiffness tensors match the elements of the matrices.

Except for the D-variate weight functions ω and Q , the entries of the mass and stiffness tensors (4.1) and (4.2) are the product of univariate B-splines. 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 [22], we can approximate these weight functions via interpolation by some combination of univariate functions,

$$(4.4) \quad \omega(\hat{x}) \approx \omega_1(\hat{x}^{(1)}) \cdots \omega_D(\hat{x}^{(D)}).$$

The integrands then are separable into products of univariate integrals. 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 [21, 22].

To do so, we interpolate the weight functions by a combination of univariate B-splines of higher order, denoted by the spline space $\tilde{\mathbb{S}}^D$ with suitable knot vectors $\tilde{\xi}_d$ and degrees $\tilde{p}_d$ with $d = 1, \dots, D$. The weight function $\omega(\hat{x})$ of the mass matrix is interpolated as

$$(4.5) \quad \omega(\hat{x}) \approx \sum_{\mathbf{j} \in \mathcal{J}} W_{\mathbf{j}} \tilde{\beta}_{\mathbf{j}}(\hat{x}) = W : \tilde{B}(\hat{x}),$$

where $\tilde{\beta}_{\mathbf{j}}(\hat{x})$ are the elements of the spline space $\tilde{\mathbb{S}}^D$ and $\tilde{B}(\hat{x})$ is the tensor holding all $\beta_{\mathbf{j}}$ ordered according to the index set $\mathcal{J}$. The weight tensor W has the same dimension as the

spline space $\tilde{\mathbb{S}}^D$, being $(\tilde{n}_1, \dots, \tilde{n}_D)$, and we get its entries by interpolating the weight function in a sufficient number of points, namely $\tilde{n} = \tilde{n}_1 \cdots \tilde{n}_D$.

We construct canonical low rank representations of the weight tensor,

$$(4.6) \quad W \approx \sum_{r=1}^R \bigotimes_{d=1}^D w_r^{(d)} =: W_R,$$

with $w_r^{(d)} \in \mathbb{R}^{n_d}$ to get a low rank representation of the weight function,

$$(4.7) \quad \omega(\hat{x}) \approx W_R : \tilde{B}(\hat{x}) = \sum_{r=1}^R \prod_{d=1}^D w_r^{(d)} \cdot \tilde{B}^{(d)}(\hat{x}^{(d)}).$$

Here $\tilde{B}^{(d)}(\hat{x}^{(d)}) \in \mathbb{R}^{n_d}$ denotes the vector holding all univariate basis functions evaluated in $\hat{x}^{(d)}$ as in (3.5), and “ $\cdot$ ” is the scalar product. The entries of the mass matrix can be approximated using this low rank representation and we can calculate each entry as the sum of products of univariate integrals,

$$(4.8) \quad M_{i,j} = \sum_{r=1}^R \prod_{d=1}^D \int_0^1 \beta_{i_d}^{(d)} \beta_{j_d}^{(d)} w_r^{(d)} \cdot \tilde{\beta}^{(d)} d\hat{x}^{(d)}.$$

With these univariate integrals we define a univariate mass matrix, which depends on some weight function ω , as

$$(4.9) \quad M^{(d)}(\omega) = \int_0^1 B^{(d)} \otimes B^{(d)} \omega d\hat{x}^{(d)},$$

where $B^{(d)} \in \mathbb{R}^{n_d}$ is the vector holding all n_d univariate B-splines of $\mathbb{S}_{\xi_d}^{p_d}$. According to the tensor representation in Equation (4.1), we can finally write the mass matrix as a sum of Kronecker products of small univariate mass matrices (4.9) with $\omega = w_r^{(d)} \cdot \tilde{\beta}^{(d)}$,

$$(4.10) \quad M = \sum_{r=1}^R \bigotimes_{d=1}^D M^{(d)}(w_r^{(d)} \cdot \tilde{\beta}^{(d)}).$$

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 (4.7), for each entry of Q we get the canonical low rank representation

$$(4.11) \quad q_{k,l}(\hat{x}) \approx V_{k,l,R} : \tilde{B}(\hat{x}) = \sum_{r=1}^R \prod_{d=1}^D v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)}(\hat{x}^{(d)}), \quad \text{for all } k, l = 1, \dots, D,$$

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

$$(4.12) \quad K_{i,j} = \sum_{k,l=1}^D \int_{\hat{\Omega}} \left(\prod_{d=1}^D \delta(l,d) \beta_{i_d}^{(d)} \delta(k,d) \beta_{j_d}^{(d)} \right) \sum_{r=1}^R \prod_{d=1}^D v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)} d\hat{x},$$

$$(4.13) \quad = \sum_{k,l=1}^D \sum_{r=1}^R \prod_{d=1}^D \int_0^1 \delta(l,d) \beta_{i_d}^{(d)} \delta(k,d) \beta_{j_d}^{(d)} v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)} d\hat{x}^{(d)},$$

where $\mathbf{j} = (j_1, \dots, j_D)$, and $\delta(k, d)$ denotes the operator acting on f as

$$(4.14) \quad \delta(k, d)f = \begin{cases} \frac{\partial f}{\partial \hat{x}_d} & \text{if } k = d, \\ f & \text{otherwise.} \end{cases}$$

To get a representation for the stiffness matrix corresponding to the mass matrix representation in (4.10), we define the D^2 univariate stiffness matrices dependent on some weight function $q^{(d)}(\hat{x}^{(d)})$ as

$$(4.15) \quad K_{k,l}^{(d)}(q^{(d)}) = \int_0^1 (\delta(l, d)B) \otimes (\delta(k, d)B) q^{(d)} d\hat{x}^{(d)}, \quad \text{for } k, l = 1, \dots, D.$$

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

$$(4.16) \quad K = \sum_{k,l=1}^D \sum_{r=1}^R \bigotimes_{d=1}^D K_{k,l}^{(d)}(v_{k,l,r}^{(d)} \cdot \tilde{\beta}^{(d)}).$$

Strategies to efficiently compute the low rank representation for W have been discussed in [21] for two dimensional settings and for three dimensional settings with partial decompositions in [28] and using tensor decompositions in [28, 2].

We follow the strategy presented in [2] using low rank Tensor Train methods [24, 8] for the decompositions and further computations as it allows us to exploit the resulting low-rank structure.

5. The Alternating Minimal Energy solver for Tensor Train. The IAS algorithm requires solving the minimization problem (2.11) multiple times in line 4 of Algorithms 2.1 and 2.2. To solve the KKT system (3.28) we use a Block-structured Alternating Minimal Energy solver (AMEn) as proposed in [2]. The saddle point system is large and each block has a Kronecker product based structure. This structure can be interpreted as a low-rank Tensor-Train (TT) [24] representation. Alternatively, using a CGLS method with the reduced formulation (2.12) requires a large number of matrix-vector products with the matrix A . This matrix is dense and quickly gets unfeasibly large. Therefore, we want to avoid its complete computation and instead solve the matrix-vector product without explicitly forming A . We propose using the TT AMEn method to apply this computation as well.

The TT format is especially convenient for our purpose as the underlying Kronecker product structure of the parameter-to-observable map (3.23) – (3.25) can be interpreted as a low-rank Tensor-Train representation. Using the TT format we can easily compute the desired result in a low-rank format without exceeding the memory limitations of a standard computer even for large models, which usually are infeasible.

The Tensor-Train format represents a d -dimensional tensor $\mathbf{T} \in \mathbb{W}_{n_1, n_2, \dots, n_d}$ of order $(n_1, n_2, \dots, n_d)$ with so-called TT-cores $T^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ with TT-ranks r_k for $k = 1, \dots, d$. By convention we set $r_0 = r_d = 1$. Each TT-core can be interpreted as a parameter-dependent matrix $T^{(i)}(j_k) \in \mathbb{R}^{r_{k-1} \times r_k}$, $j_k = 1, \dots, n_i$ and every element of $\mathbf{T}$ is represented as the product

$$(5.1) \quad \mathbf{T}(j_1, \dots, j_d) = T^{(1)}(j_1) \cdots T^{(d)}(j_d).$$

The whole tensor can be written as a sum of Kronecker products,

$$(5.2) \quad \mathbf{T} = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} \bigotimes_{k=1}^d \mathbf{T}_{\alpha_{k-1}, \alpha_k}^{(k)} =: tt(T^{(1)}, \dots, T^{(d)}),$$

where the subscripts α_{k-1}, α_k are row and column indices of the TT-core $T^{(k)}$. This format directly corresponds to the matrices in (3.23) – (3.25).

We want to solve an equation system $\mathbf{Ax} = b$ in this format, where $\mathbf{A}$ is a tensor of size $N \times N$ with $N = (n_1, \dots, n_d)$ and $\mathbf{x}$ and b are tensors of size N . This can be done efficiently with an energy function minimization cycling over the TT-cores, the *alternating linear scheme* (ALS) [16]. This approach constructs low-dimensional systems of linear equations for each core, which can then be solved with standard numerical methods. Here, we derive the method for symmetric $\mathbf{A}$. Solving $\mathbf{Ax} = b$ then corresponds to the minimization of the energy function,

$$(5.3) \quad \min_{\mathbf{x}} J(\mathbf{x}) = \|\mathbf{x}_* - \mathbf{x}\|_{\mathbf{A}}^2 = (\mathbf{x}, \mathbf{Ax}) - 2(\mathbf{x}, \mathbf{b}) + \text{const},$$

with the exact solution $\mathbf{x}_* = \mathbf{A}^{-1}\mathbf{b}$. To find a solution for (5.3) we make an initial guess $\mathbf{x}_0$ and cycle over its TT-cores where we solve a local problem to improve the current guess. For this, in iteration k all cores but the k -th are *frozen* and we minimize over

$$(5.4) \quad \mathbf{x}_{\text{new}} = tt(x^{(1)}, \dots, x^{(k-1)}, x_{\text{new}}^{(k)}, x^{(k+1)}, \dots, x^{(d)})$$

$$(5.5) \quad x_{\text{new}}^{(k)} = \min_{x^{(k)}} J(\mathbf{x}_{\text{new}}).$$

The energy function $J(\mathbf{x})$ does not grow during updates and the solution will subsequently converge to a local minimum. The tensor-train format is linear in its cores, as

$$(5.6) \quad \mathbf{x} = tt(x^{(1)}, \dots, x^{(k)}, \dots, x^{(d)}) = \mathbf{x}_{\neq k} x_k,$$

where $\mathbf{x}_{\neq k}$ is the Tensor-Train where the k -th core $\mathbf{x}^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is replaced by an identity operator of the same size and x_k is a vectorization of $x^{(k)}$. With this, the energy function for the local problem becomes

$$(5.7) \quad J(\mathbf{x}) = (\mathbf{Ax}_{\neq k} x_k, \mathbf{x}_{\neq k} x_k) - 2(\mathbf{b}, \mathbf{x}_{\neq k} x_k) = (\mathbf{x}_{\neq k}^* \mathbf{Ax}_{\neq k} x_k, x_k) - 2(\mathbf{x}_{\neq k}^* \mathbf{b}, x_k).$$

The gradient of (5.7) with respect to x_k is zero when

$$(5.8) \quad (\mathbf{x}_{\neq k}^* \mathbf{Ax}_{\neq k}) x_k = \mathbf{x}_{\neq k}^* \mathbf{b}.$$

Therefore, the solution to the local minimization is equal to a solution in a reduced basis. This problem is small and can be solved by standard numerical methods. For the general ALS approach the resulting TT-ranks - and therefore the maximum accuracy - are fixed by the ranks set in the initial guess. But with some extension we can adapt the TT-ranks of the solution dynamically. For the purpose of calculating a low-rank approximation, we chose the so-called *Alternating Minimal Energy* (AMEn) method [8].

Algorithm 5.1 Low-rank AMEn

```

1: Input: Initial guess  $\mathbf{x}$  with TT-ranks  $r = (r_1, \dots, r_d)$ , tolerance  $\epsilon_0$  and/or rank bounds
    $r_{\max}$ , system in TT-format  $\mathbf{A}, \mathbf{b}$ 
2: while  $\|\mathbf{Ax} - \mathbf{b}\| > \epsilon_0$  and  $k < \text{maxIter}$  do
3:   Form  $A_1 = \mathbf{x}_{\neq 1}^* \mathbf{A} x_{\neq 1}, b_1 = \mathbf{x}_{\neq 1}^* \mathbf{b}$ 
4:   Solve  $A_1 u^{(1)} = b_1$ 
5:   Optional: Do an SVD-based compression of  $u^{(1)}$ 
6:   Let  $\mathbf{u} = \text{tt}(u^{(1)}, x^{(2)}, \dots, x^{(d)})$  and  $\mathbf{r} = \mathbf{Au} - \mathbf{b}$ 
7:   if  $\|\mathbf{r}\| < \epsilon_0$  then
8:     return  $\mathbf{x} = \mathbf{u}$ 
9:   end if
10:  Find smallest-rank residual approximation  $\bar{\mathbf{r}}$  with  $\|\bar{\mathbf{r}} - \mathbf{r}\| \leq \epsilon_0 \|\mathbf{r}\|$  and/or  $\text{rank}(\bar{\mathbf{r}}^{(1)}) \leq$ 
    $r_{\max}$ 
11:  Expand basis by first TT-core  $\bar{r}^{(1)}$  of  $\bar{\mathbf{r}}, x^{(1)} = \begin{bmatrix} u^{(1)} & \bar{r}^{(1)} \end{bmatrix}$ , and accordingly  $x^{(2)} =$ 
    $\begin{bmatrix} x^{(2)} \\ 0 \end{bmatrix}$ 
12:  Form  $X_1 = x^{(1)} \otimes I_{n_2, \dots, n_d}$  of size  $(n_1, \dots, n_d) \times (r_1 n_2, \dots, n_d)$ 
13:  Form the  $(d-1)$ -dimensional system  $A_{\geq 2} x_{\geq 2} = b_{\geq 2}$  with  $A_{\geq 2} = X_1^* \mathbf{A} X_1, b_{\geq 2} = X_1^* \mathbf{b}$ .
14:  if  $d = 2$  then
15:    Solve  $A_{\geq 2} x_{\geq 2} = b_{\geq 2}$  directly
16:  else
17:    Call AMEn recursively with  $\mathbf{x} \leftarrow \text{tt}(x^{(2)}, \dots, x^{(d)}), \mathbf{A} \leftarrow A_{\geq 2}, \mathbf{b} \leftarrow b_{\geq 2}$ .
18:  end if
19: end while
20: Output: low-rank solution  $\mathbf{x} = \text{tt}(x^{(1)}, \dots, x^{(d)})$ 

```

The algorithm is outlined in [Algorithm 5.1](#). This method first solves a local system as in (5.8). Then it expands the components of the solution subsequently by local gradient information as in line 11 of the algorithm. Afterwards, the system dimension is reduced by one and the previous computation repeated in a recursive way until a 2D matrix equation system is reached. Solving this equation system finalizes the outer iteration after which all steps are repeated until reaching the convergence tolerance. The method proved to be robust and has a fast convergence rate. For a detailed analysis and more information we refer the reader to [8].

Using the TT notation and the AMEn method, we can compute matrix-vector products with A efficiently without forming the full matrix and get the resulting solution vectors in a low-rank format. For solving the block system (3.28) we use a block solver based on AMEn *AMEn Block Solve*, which exploits the block structure of (3.28) as well as the low rank tensor product structure of each component.

6. Numerical experiments. We want to illustrate the performance of our method with some numerical examples and analyze the choice of parameters for PDE problems. We first

r	η	β	ϑ
1	10^{-5}	$3/2 + \eta$	3.3
0.5	10^{-3}	$(3/2 + \eta)/r$	8.3
-1	$\beta + 3/2$	3	1.5×10^{-4}

Table 1: Parameters for the three hyper models

present results for different parameter settings for a two dimensional domain and later show results for a 3D model. The parameters for the different hypermodel setups are listed in Table 1.

Example 1: 2D domain. First, let us consider a two dimensional domain with sparse heat sources, as shown in Figure 2a. This domain is discretized using IGA with 32 B-splines per dimension resulting in 1024 spatial degrees of freedom. The total rank for the mass matrix is 8 and for the stiffness matrix we have a combined number of 12 low-rank components. For the time discretization we use $N_t = 50$.

This setup is governed by a heat distribution

$$(6.1) \quad \frac{\partial x}{\partial t}(t) = 0.1 \Delta x(t) \quad \text{in } \Omega \times (0, 1]$$

$$(6.2) \quad x(0) = x_0 \quad \text{in } \Omega \times (t = 0)$$

$$(6.3) \quad x(t) = 0 \quad \text{on } \partial\Omega \times [0, 1]$$

and we measure the state only once after 50 time steps at $T = 1$. The resulting state is shown in Figure 2b and is measured under noise $e \sim \mathcal{N}(0, 0.1)$. We run a non-modified IAS algorithm with hyperprior parameters as in the first row of Table 1 with $r = 1$, as suggested in [4]. We do 50 IAS steps and 30 CGLS steps per iteration. For the low-rank matrix vector products we set a tolerance of 10^{-6} . The solver tolerance for AMEn was set to 10^{-4} and we do a maximum of 20 AMEn steps per iteration. An exemplary result is shown in Figure 2c. We see that the solution indeed fits the very sparse initial state perfectly. The algorithm successfully reconstructs the 4 sources. For comparison the result for a least squares reconstruction is shown in Figure 2d.

Example 2: Hyperprior parameters. Let us now take a closer look at the parameters of the generalized gamma distribution used for the hyperprior in Equation (2.5) and the performance of the local and global hybrid IAS algorithm. We use the same setup as before. We set up a globally convex hyperprior model with $r = 1$ two different sparsity inducing models with $r = 0.5$ and $r = -1$ to compare the performance of the different algorithm variations. The rest of the parameters correspond to Table 1.

We will compare three different methods: CGLS with a full computation of A denoted by full CGLS that will not be feasible for large scale problems; CGLS with low-rank matrix-vector products in TT format denoted by TT CGLS; and solving the KKT system of the constrained optimization problem with the block AMEn method denoted by AMEn.

First, let us review the global hybrid IAS switching models after 10 iterations. In Figure 3 we see results for the two different hypermodel setups with $r = -1$ and $r = 0.5$. We compare

(a) Two dimensional domain with sparse heat sources.

(b) Two dimensional domain after 100 time steps at end time $T = 1$ with noise.

(c) Resulting sparse reconstruction.

(d) For comparison: Resulting reconstruction using only least squares without the hyperprior pdf.

Figure 2: Setup for the 2D example problem. The PDE is discretized using 1024 basis functions.

the performance of the low-rank CGLS and the constrained optimization with AMEn. Both setups start out with a globally convex hypermodel with $r = 1$. After 10 iterations the first algorithm switches to a greedy hypermodel with $r = -1$, indicated by a red vertical line. We can see the number of non-zero entries reducing shortly after the tenth iteration in Figure 3a for $r = -1$. In Figure 3b, we see that the CGLS method starts with the maximum number of iterations and the low-rank version requires more IAS steps until the number of local iterations starts reducing compared to the full CGLS method. This can be attributed to the inexact matrix-vector products leading to inexact descent directions. Nonetheless, the low-rank CGLS converges with only a small number of extra iterations required.

(a) Number of indices for which u_0 is non-zero per iteration for $r = -1$. The switch to the model with stronger sparsity promotion is indicated by the red dotted line.

(b) CGLS steps and AMEn sweeps per iteration for the global hybrid IAS with $r = -1$.

(c) Number of indices for which u_0 is non-zero per iteration for $r = 0.5$. The switch to the model with stronger sparsity promotion is indicated by the red dotted line.

(d) CGLS steps and AMEn sweeps per iteration for the global hybrid IAS with $r = 0.5$.

Figure 3: Development of non-zero indices and CGLS steps for the global hybrid IAS. We compare using the full matrix A with CGLS (full CGLS), CGLS with low-rank matrix-vector products (TT CGLS) and a constrained minimization with AMEn.

The constrained optimization method does not require the full number of 20 steps from the beginning. This is due to $D_\theta^{-1/2}$ starting out quite large. This could be interpreted as a large regularization parameter for u_0 in the optimization, which is generally easier to solve. As θ changes, the minimization gets more challenging before the iteration numbers

start decreasing again. All three algorithms converge with the local solvers needing 4 steps for their last iterations, which corresponds to the number of non-zero values in u_0 .

In Figures 3c and 3d we see very similar results for a second model with $r = 0.5$. Note, that both algorithms are equal until iteration 10 where we switch to the different second models. For $r = 0.5$ convergence is not reached as fast as for $r = -1$, which was to be expected as $r = -1$ promotes sparsity more strongly. Again we see a quick reduction in non-zero entries shortly after the switch to the second hyper model in Figure 3c. And as before the low-rank methods need some more iterations to converge, as seen in Figure 3d.

Figure 4: We compare using the full matrix A with CGLS (full CGLS), CGLS with low-rank matrix-vector products (TT CGLS) and a constrained minimization with AMEn.

		global		local	
		iterations	residual	iterations	residual
full CGLS	$r = -1$	15	2.3e-7	22	9.8e-3
	$r = 0.5$	16	5.9e-6	28	7.2e-3
TT CGLS	$r = -1$	18	9.2e-7	41	4.8e-2
	$r = 0.5$	26	2.6e-6	> 50	0.4
AMEn	$r = -1$	18	3.6e-4	32	8.3e-2
	$r = 0.5$	24	2.4e-4	38	8.2e-2

Table 2: Number of total iterations and residual error to the original initial state as $\|u_0 - u\|/\|u_0\|$ with true initial state u_0 and reconstruction u .

Next, we look at the local hybrid IAS. Here, the switch to the second models is done individually for each entry of θ as described in [subsection 2.1](#). The results are shown in [Figure 4](#). The first row shows the performance with $r = -1$ for the second model. In [Figure 4a](#) we see the number of entries in θ , which remain in the first hypermodel with $r = 1$. All others are switched to $r = -1$ as soon as their convexity condition is fulfilled. We see that the constrained minimization with AMEn performs quite well. The number of parameters remaining in the model with $r = 1$ is reducing quickly. The low-rank CGLS requires some more iterations but ultimately converges as well. In [Figure 4b](#), we see that again the AMEn method ends up converging with only 4 local iterations towards the end of the algorithm, similar to the full CGLS method. The low-rank TT CGLS however requires a large number of steps until the very end when it converges.

Additionally, in [Figures 4c](#) and [4d](#) we see the results for a model with $r = 0.5$. As expected the method needs more iterations to converge and the number of indices in θ remaining in the first model decreases more slowly for all three algorithms in [Figure 4c](#). Again, the full CGLS and AMEn converge with 4 local iterations towards the last iterations of the algorithms. However, the TT CGLS does not converge within 50 outer iterations. Here, a smaller residual tolerance for the matrix-vector products might result in more successful steps.

The number of iterations and the resulting residual with respect to the actual initial state are listed in [Table 2](#). We see that overall the results for the global method are very good. The larger residuals for AMEn correspond to the solver tolerance, which was set at 10^{-4} . For the local method however all methods required a larger number of iterations and the resulting reconstruction was not as close to the original as for the global method.

Example 2: 3D domain. We now consider a three dimensional domain with a more complex geometric model. The low-rank structure has a much higher rank than the previous example, namely 10 factors for the mass and 180 factors for the stiffness matrix. The initial state is displayed in [Figure 5a](#) and the final state we use for the reconstruction with noise $\sigma = 0.1$ in [Figure 5b](#). An exemplary reconstruction with global hybrid IAS and $r_2 = -1$ is displayed in [Figure 6](#).

We modify the problem size by changing the space discretization size and number of time steps to observe the scaling behavior of the method. For the TT CGLS method we set the

(a) Three dimensional domain with sparse heat sources.

(b) Three dimensional domain after 100 time steps at end time $T = 1$ with noise.

Figure 5: Setup for the 3D example problem

tolerance to $1e-6$. The result for two spatial discretizations and three different time resolutions is displayed in Table 3a. The setups with equal spatial discretization behave very similarly. For a total of 756 degrees of freedom all time discretizations result in less than 10 iterations. For the discretization with 6048 spatial nodes some more iterations are required but again the method displays similar behavior for all time discretizations. For the constrained minimization with AMEn the behavior is quite different. Here, we display three spatial discretizations and two time discretizations in Table 3. For the different spatial resolutions the number of IAS iterations is somewhat stable. But with a larger number of time steps the required iterations increase drastically. In future research, we aim to mitigate this behavior by further tailoring the AMEn solver regarding the enrichment process and the design of suitable preconditioners.

7. Outlook. We demonstrated that combining low rank tensor methods with sparsity inducing hyperprior models gives us a powerful method for the reconstruction of sparse initial states for large scale PDE systems. We tested different parameter setups for the local and global version of the hybrid IAS algorithm and compared the performance of different local solvers. The IAS algorithm requires a number of subsequent solutions of a large scale minimization problem. We can solve the arising problems with standard methods for quadratic

Figure 6: Resulting sparse reconstruction for the 3D example problem

$N_t \backslash n$	756	6048	48384
50	16	18	29
100	47	50	49

Table 3: Number of IAS steps using a constrained minimization formulation with AMEn for the global hybrid IAS with $r = -1$ and the 3D model problem for different levels of discretization.(a) Number of IAS steps for the global hybrid IAS with $r = -1$ and the 3D model problem for different discretization sizes. Here we used TT CGLS

minimization like CLGS or formulate a constrained optimization problem. In the context of IGA discretization, both representations can be solved efficiently using tensor train calculations and the Alternating Minimal Energy solver (AMEn) designed to work with low rank Kronecker product based problems.

In combination with an Isogeometric space discretization, tensor train calculations can be very efficient. In future research, we want to enhance the methods performance by equipping it with suitable preconditioning and a more fitting local solver. This will allow us to solve sparse reconstruction problems for large-scale three dimensional discretizations.

Acknowledgements. The authors would like to thank Daniela Calvetti for her helpful insights and sharing of implementations.

The work of the authors was supported by the German Science Foundation (DFG) through grant 1742243256 - TRR 9.

REFERENCES

- [1] Z. BOFANG, *The Finite Element Method*, John Wiley & Sons Singapore Pte. Ltd, Apr. 2018.
- [2] A. BÜNGER, S. DOLGOV, AND M. STOLL, *A low-rank tensor method for PDE-constrained optimization with isogeometric analysis*, SIAM J. Sci. Comput., 42 (2020), pp. A140–A161.
- [3] D. CALVETTI, M. PRAGLIOLA, AND E. SOMERSALO, *Sparsity promoting hybrid solvers for hierarchical Bayesian inverse problems*, SIAM J. Sci. Comput., 42 (2020), pp. A3761–A3784.
- [4] D. CALVETTI, M. PRAGLIOLA, E. SOMERSALO, AND A. STRANG, *Sparse reconstructions from few noisy data: Analysis of hierarchical Bayesian models with generalized gamma hyperpriors*, Inverse Probl., 36 (2020), p. 025010.
- [5] D. CALVETTI, E. SOMERSALO, AND A. STRANG, *Hierarchical Bayesian models and sparsity: ℓ_2 -magic*, Inverse Probl., 35 (2019), p. 035003.
- [6] E. J. CANDÉS, J. K. ROMBERG, AND T. TAO, *Stable signal recovery from incomplete and inaccurate measurements*, Comm. Pure Appl. Math., 59 (2006), pp. 1207–1223.
- [7] J. A. COTTRELL, T. J. HUGHES, AND Y. BAZILEVS, *Isogeometric Analysis*, John Wiley & Sons, Ltd, Aug. 2009.
- [8] 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.
- [9] D. L. DONOHO, *Superresolution via sparsity constraints*, SIAM J. Math. Anal., 23 (1992), pp. 1309–1331.
- [10] ———, *For most large underdetermined systems of linear equations the minimal ℓ_1 -norm solution is also the sparsest solution*, Comm. Pure Appl. Math., 59 (2006), pp. 797–829.
- [11] H. ELMAN, D. SILVESTER, AND A. WATHEN, *Finite Elements and Fast Iterative Solvers*, Numerical Mathematics and Scientific Computation, Oxford University Press, second ed., June 2014.
- [12] G. FARIN, *NURBS for Curve & Surface Design*, A K Peters/CRC Press, Mar. 1999.
- [13] M. GRASMAIR, M. HALTMEIER, AND O. SCHERZER, *Sparse regularization with ℓ_q penalty term*, Inverse Probl., 24 (2008), p. 055020.
- [14] M. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stan., 49 (1952), p. 409.
- [15] M. HINZE, *A variational discretization concept in control constrained optimization: The linear-quadratic case*, Comput Optim Applic, 30 (2005), pp. 45–61.
- [16] 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.
- [17] T. HUGHES, J. COTTRELL, AND Y. BAZILEVS, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Method. Appl. M., 194 (2005), pp. 4135–4195.
- [18] B. JIN, P. MAASS, AND O. SCHERZER, *Sparsity regularization in inverse problems*, Inverse Probl., 33 (2017), p. 060301.
- [19] M. JUNCOSA AND D. YOUNG, *On the Crank-Nicolson procedure for solving parabolic partial differential equations*, Math. Proc. Camb. Phil. Soc., 53 (1957), pp. 448–461.
- [20] B. KNAPIK, A. VAN DER VAART, AND J. VAN ZANTEN, *Bayesian recovery of the initial condition for the heat equation*, Communications in Statistics - Theory and Methods, 42 (2013), pp. 1294–1313.
- [21] A. MANTZAFARIS, B. JÜTTLER, B. 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 International Publishing, 2015, pp. 321–340.
- [22] A. MANTZAFARIS, B. JÜTTLER, B. N. KHOROMSKIJ, AND U. LANGER, *Low rank tensor methods in Galerkin-based isogeometric analysis*, Comput. Method. Appl. M., 316 (2017), pp. 1062–1085.
- [23] H. MINGYOU AND V. THOMÉE, *On the backward Euler method for parabolic equations with rough initial data*, SIAM J. Numer. Anal., 19 (1982), pp. 599–603.
- [24] I. OSELEDETS, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
- [25] 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.
- [26] L. PIEGL, *On NURBS: A survey*, IEEE Comput. Grap. Appl., 11 (1991), pp. 55–71.
 - [27] S. RAO, *The Finite Element Method in Engineering*, Elsevier, sixth edition ed., 2018.
 - [28] F. SCHOLZ, A. MANTZAFLARIS, AND B. JÄJTTLER, *Partial tensor decomposition for decoupling isogeometric Galerkin discretizations*, Comput. Method. Appl. M., 336 (2018), pp. 485–506.
 - [29] J. R. SHEWCHUK, *An introduction to the conjugate gradient method without the agonizing pain*, tech. rep., Carnegie Mellon University, USA, 1994.
 - [30] M. STOLL AND T. BREITEN, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci. Comput., 37 (2015), pp. B1–B29.
 - [31] S. SUBRAMANIAN, K. SCHEUFELE, M. MEHL, AND G. BIROS, *Where did the tumor start? an inverse solver with sparse localization for tumor growth models*, Inverse Probl., 36 (2020), p. 045006.
 - [32] C. A. ZARZER, *On Tikhonov regularization with non-convex sparsity constraints*, Inverse Probl., 25 (2009), p. 025006.

9 Theses & Summary

1. In many applications large scale PDE problems arise. Their discretization and solution often proves difficult. Low-rank methods to reduce storage requirements and computation time have become a powerful approach to treat many large scale problems. This thesis provides contributions to the low-rank approximation of large-scale IGA discretized PDE problems and their application to different scenarios.
2. A problem which consists of the sum of Kronecker products can be interpreted as a low-rank tensor product format and treated efficiently as such using the AMEn scheme. This solver computes efficient low-rank approximations in tensor train format.
3. In IGA basis functions are globally defined and multi-dimensional spline spaces are tensor products of one dimensional spline spaces. This thesis provides a scheme to exploit this property to approximate the system matrices arising from a PDE discretization as sums of Kronecker products, giving rise to an efficient low-rank tensor method.
4. Coupled with a time-discretization such as an implicit Euler scheme, we can interpret a PDE as the sum of tensor products in each dimension, breaking the curse of dimensionality associated with higher dimensional problems.
5. For IGA, three dimensional domains can be discretized efficiently by approximating their weight function via interpolation and computing low-rank tensor approximations of their coefficients.
6. In this low-rank method only univariate integration is necessary to assemble the system matrices, which greatly reduces storage requirements as well as computation time (see Section 3.6 and 6)

7. Instead of assembling the full matrices from their one-dimensional decompositions we can use the decomposed factors in low-rank computations for various applications.
8. We used low-rank IGA to solve large scale optimization problems. Their KKT systems are block structured matrices for which each block possesses a Kronecker product based structure. Inserting the IGA decompositions we can use low-rank tensor train computations in a block-based computation scheme to compute low-rank solutions (Section 6).
9. We can use the same strategy to solve large-scale Bayesian inverse problems (5) and applied it to a scheme for the reconstruction of sparse initial states from noisy PDE solution. Here, using an inexact AMEn method proved efficient for the numerical scheme (Section 8).
10. We additionally developed a method to solve PDE-constrained optimization problems with full system matrices such as ones derived from FEM. The developed method reformulates the problem into a matrix equation and solves it with a rational Krylov subspace solver (Section 7).
11. The low-rank solver AMEn can also be used efficiently in a FEM discretized problem, separating only the spatial dimension as shown for the computation of a low-rank approximation to the posterior variance on a machine tool in Section 5.2.

Bibliography

- [1] M. ADES, P. CAINES, AND R. MALHAME, *Stochastic optimal control under Poisson-distributed observations*, IEEE Transactions on Automatic Control, 45 (2000), pp. 3–13.
- [2] A. ALEXANDERIAN, *Optimal experimental design for infinite-dimensional Bayesian inverse problems governed by PDEs: A review*, Inverse Probl., 37 (2021), p. 043001.
- [3] F. AURICCHIO, L. B. DA VEIGA, T. HUGHES, A. REALI, AND G. SANGALLI, *Isogeometric collocation methods*, Math. Models Methods Appl. Sci., 20 (2010), pp. 2075–2107.
- [4] Z.-Z. BAI, *Block preconditioners for elliptic PDE-constrained optimization problems*, Computing, 91 (2011), pp. 379–395.
- [5] Y. BAZILEVS AND T. J. R. HUGHES, *Weak imposition of Dirichlet boundary conditions in fluid mechanics*, Computers & Fluids, 36 (2007), pp. 12–26.
- [6] L. BEIRÃO DA VEIGA, A. BUFFA, J. RIVAS, AND G. SANGALLI, *Some estimates for h - p - k -refinement in Isogeometric Analysis*, Numer. Math., 118 (2011), pp. 271–305.
- [7] P. BENNER, S. DOLGOV, A. ONWUNTA, AND M. STOLL, *Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data*, Comput. Method. Appl. M., 304 (2016), pp. 26–54.
- [8] P. BENNER, S. DOLGOV, A. ONWUNTA, AND M. STOLL, *Solving optimal control problems governed by random Navier-Stokes equations using low-rank methods*, ArXiv e-prints, (2017).

- [9] P. BENNER, A. ONWUNTA, AND M. STOLL, *Block-diagonal preconditioning for optimal control problems constrained by PDEs with uncertain inputs*, SIAM J. Matrix Anal. & Appl., 37 (2016), pp. 491–518.
- [10] P. BENNER, Y. QIU, AND M. STOLL, *Low-rank eigenvector compression of posterior covariance matrices for linear Gaussian inverse problems*, SIAM J. Uncertainty Quantification, 6 (2018), pp. 965–989.
- [11] A. BORZÍ, *Multigrid methods for parabolic distributed optimal control problems*, J. Comput. Appl. Math., 157 (2003), pp. 365–382.
- [12] A. BORZÍ AND V. SCHULZ, *Multigrid methods for PDE optimization*, SIAM Rev., 51 (2009), pp. 361–395.
- [13] J. BRINKHUIS AND V. TIKHOMIROV, *Optimization: Insights and Applications*, Princeton Series in Applied Mathematics, Princeton University Press, Dec. 2005.
- [14] T. BUI-THANH, O. GHATTAS, J. MARTIN, AND G. STADLER, *A computational framework for infinite-dimensional Bayesian inverse problems part I: The linearized case, with application to global seismic inversion*, SIAM J. Sci. Comput., 35 (2013), pp. A2494–A2523.
- [15] A. BÜNGER, S. DOLGOV, AND M. STOLL, *A low-rank tensor method for PDE-constrained optimization with isogeometric analysis*, SIAM J. Sci. Comput., 42 (2020), pp. A140–A161.
- [16] A. BÜNGER, V. SIMONCINI, AND M. STOLL, *A low-rank matrix equation method for solving PDE-constrained optimization problems*, SIAM J. Sci. Comput., 43 (2021).
- [17] A. BÜNGER AND M. STOLL, *A low-rank tensor method to reconstruct sparse initial states for PDEs with Isogeometric Analysis*, arXiv e-prints, (2021), p. arXiv:2109.03119.
- [18] D. CALVETTI, M. PRAGLIOLA, E. SOMERSALO, AND A. STRANG, *Sparse reconstructions from few noisy data: Analysis of hierarchical Bayesian models with generalized gamma hyperpriors*, Inverse Probl., 36 (2020), p. 025010.

- [19] D. CALVETTI AND E. SOMERSALO, *Introduction to Bayesian Scientific Computing*, vol. 2 of Surveys and Tutorials in the Applied Mathematical Sciences, Springer, 2007.
- [20] J. D. CARROLL AND J.-J. CHANG, *Analysis of individual differences in multidimensional scaling via an n -way generalization of “Eckart-Young” decomposition*, Psychometrika, 35 (1970), pp. 283–319.
- [21] D. CONTE AND C. LUBICH, *An error analysis of the multi-configurational time-dependent hartree method of quantum dynamics*, M2AN Math. Model. Numer. Anal., 44 (2010), p. 759.
- [22] J. A. COTTRELL, T. J. HUGHES, AND Y. BAZILEVS, *Isogeometric Analysis*, John Wiley & Sons, Ltd, Aug. 2009.
- [23] C. DE BOOR, *A Practical Guide to Splines*, Springer New York, 1978.
- [24] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. & Appl., 21 (2000), pp. 1253–1278.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] G. ECKART AND G. YOUNG, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

- [29] H. ELMAN, D. SILVESTER, AND A. WATHEN, *Finite Elements and Fast Iterative Solvers*, Numerical Mathematics and Scientific Computation, Oxford University Press, second ed., June 2014.
- [30] H. C. ELMAN AND T. SU, *A low-rank solver for the stochastic unsteady navier-stokes problem*, Computer Methods in Applied Mechanics and Engineering, 364 (2020), p. 112948.
- [31] J. EVANS, R. HIEMSTRA, T. HUGHES, AND A. REALI, *Explicit higher-order accurate isogeometric collocation methods for structural dynamics*, Comput. Method. Appl. M., 338 (2018), pp. 208–240.
- [32] L. EVANS, *Partial Differential Equations*, Grad. Stud. Math., 19 (1998).
- [33] W. FALCÓ, A. HACKBUSCH, *On minimal subspaces in tensor representations*, Foundations of Computational Mathematics, 12 (2012), pp. 765–803.
- [34] G. FARIN, *NURBS for Curve & Surface Design*, A K Peters/CRC Press, Mar. 1999.
- [35] 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.
- [36] C. GIANNELLI, B. JÜTTLER, S. K. KLEISS, A. MANTZAFLARIS, B. SIMEON, AND J. SPEH, *THB-splines: an effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis*, Comput. Methods Appl. Mech. Engrg., 299 (2016), pp. 337–365.
- [37] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, 4 ed., 2013.
- [38] L. GRASEDYCK, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.
- [39] L. GRASEDYCK AND S. KRÄMER, *Stable ALS approximation in the TT-format for rank-adaptive tensor completion*, Numer. Math., 143 (2019), pp. 855–904.

- [40] L. GRASEDYCK, D. KRESSNER, AND C. TOBLER, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78.
- [41] M. GRASMAIR, M. HALTMEIER, AND O. SCHERZER, *Sparse regularization with l - q penalty term*, Inverse Probl., 24 (2008), p. 055020.
- [42] W. HACKBUSCH, *Tensor Spaces and Numerical Tensor Calculus*, vol. 56 of Springer Series in Computational Mathematics, Springer International Publishing, 2019.
- [43] W. HACKBUSCH AND S. KÜHN, *A new scheme for the tensor representation*, The journal on Fourier analysis and applications, 15 (2009), pp. 706–722.
- [44] E. HALE AND A. PRATER-BENNETTE, *Comparison of CP and Tucker tensor decomposition algorithms*, in Big Data III: Learning, Analytics, and Applications, F. Ahmad, P. P. Markopoulos, and B. Ouyang, eds., vol. 11730, International Society for Optics and Photonics, SPIE, 2021, pp. 68 – 85.
- [45] J. HÅSTAD, *Tensor rank is NP-complete*, J. Algorithm., 11 (1990), pp. 644–654.
- [46] R. HERZOG, I. RIEDEL, AND D. UCINSKI, *Optimal sensor placement for joint parameter and state estimation problems in large-scale dynamical systems with applications to thermo-mechanics*, Optimization and Engineering, 19 (2018).
- [47] 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. Method. Appl. M., 316 (2017), pp. 966–1004. Special Issue on Isogeometric Analysis: Progress and Challenges.
- [48] M. HINZE, *A variational discretization concept in control constrained optimization: The linear-quadratic case*, Comput Optim Applic, 30 (2005), pp. 45–61.
- [49] 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.
- [50] R. A. HORN AND C. R. JOHNSON, *Matrix Analysis*, Cambridge University Press, 2nd ed., 1990.

- [51] T. HUGHES, J. COTTRELL, AND Y. BAZILEVS, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Method. Appl. M., 194 (2005), pp. 4135–4195.
- [52] K. ITO AND K. KUNISCH, *Lagrange Multiplier Approach to Variational Problems and Applications*, vol. 15 of Advances in Design and Control, Society for Industrial and Applied Mathematics, Jan. 2008.
- [53] E. JECKELMANN, *Dynamical density-matrix renormalization-group method*, Phys. Rev. B, 66 (2002), p. 045114.
- [54] B. JIN, P. MAASS, AND O. SCHERZER, *Sparsity regularization in inverse problems*, Inverse Probl., 33 (2017), p. 060301.
- [55] J. KAIPIO AND E. SOMERSALO, *Statistical and Computational Inverse Problems*, vol. 160 of App. Math. Sci., Springer-Verlag New York, 2005.
- [56] V. KAZEEV, M. KHAMMASH, M. NIP, AND C. SCHWAB, *Direct solution of the chemical master equation using quantized tensor trains*, PLoS Comput. Biol., 10 (2014), p. e1003359.
- [57] T. G. KOLDA, *A counterexample to the possibility of an extension of the eckart-young low-rank approximation theorem for the orthogonal rank tensor decomposition*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 762–767.
- [58] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
- [59] K. LEE AND H. C. ELMAN, *A preconditioned low-rank projection method with a rank-reduced scheme for stochastic partial differential equations*, SIAM J. Sci. Comp., 39 (2017), pp. S828–S850.
- [60] P. M. LEE, *Bayesian Statistics: an Introduction*, John Wiley & Sons, 4 ed., 2012.
- [61] A. MANTZAFLARIS, B. JÜTTLER, B. N. KHOROMSKIJ, AND U. LANGER, *Low rank tensor methods in Galerkin-based isogeometric analysis*, Comput. Method. Appl. M., 316 (2017), pp. 1062–1085.

- [62] A. MANTZAFLARIS, F. SCHOLZ, AND I. TOULOPOULOS, *Low-rank space-time decoupled isogeometric analysis for parabolic problems with varying coefficients*, Comput. Methods Appl. Math., 19 (2018), pp. 123–136.
- [63] R. MCNEEL ET AL., *Rhinoceros 3D, version 6.0*, Robert McNeel & Associates, Seattle, WA, (2010).
- [64] D. MEYER, F. GATTI, AND G. A. WORTH, *The multi-configurational time-dependent hartree (MCTDH) method*, Phys. Rep., 324 (2000), p. 1.
- [65] M. L. MIKA, T. J. HUGHES, D. SCHILLINGER, P. WRIGGERS, AND R. R. HIEMSTRA, *A matrix-free isogeometric Galerkin method for Karhunen–Loève approximation of random fields using tensor product splines, tensor contraction and interpolation based quadrature*, Comput. Method. Appl. M., 379 (2021), p. 113730.
- [66] J. MONTOYA-MARTÍNEZ, A. ARTÉS-RODRÍGUEZ, M. PONTIL, AND L. K. HANSEN, *A regularized matrix factorization approach to induce structured sparse-low-rank solutions in the eeg inverse problem*, EURASIP J. Adv. Signal Process., 97 (2014).
- [67] I. OSELEDETS, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
- [68] I. OSELEDETS AND S. DOLGOV, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739.
- [69] I. V. OSELEDETS AND E. E. TYRTYSHNIKOV, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 35 (2009), pp. 3744–3759.
- [70] C. PAIGE AND M. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
- [71] M. PAN AND F. CHEN, *Low-rank parameterization of volumetric domains for isogeometric analysis*, Comput.-Aided Des., 114 (2019), pp. 82–90.

- [72] J. W. PEARSON, M. PORCELLI, AND M. STOLL, *Interior-point methods and preconditioning for PDE-constrained optimization problems involving sparsity terms*, Numer Linear Algebra Appl., 27 (2020), p. e2276.
- [73] 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.
- [74] J. W. PEARSON AND A. J. WATHEN, *A new approach of the Schur complement in preconditioners for PDE-constrained optimization*, Numer Linear Algebra Appl., 19 (2012), pp. 816–829.
- [75] ———, *Fast iterative solvers for convection-diffusion control problems*, Electron. Trans. Numer. Anal., 40 (2013), pp. 294–310.
- [76] L. PIEGL, *On NURBS: A survey*, IEEE Comput. Grap. Appl., 11 (1991), pp. 55–71.
- [77] L. PIEGL AND W. TILLER, *The NURBS Book*, Monographs in Visual Communication, Springer Berlin Heidelberg, 1995.
- [78] M. PORCELLI, V. SIMONCINI, AND M. STOLL, *Preconditioning PDE-constrained optimization with l_1 -sparsity and control constraints*, Computers & Mathematics with Applications, 74 (2017), pp. 1059–1075. SI: SDS2016 – Methods for PDEs.
- [79] H. PRAUTZSCH, W. BOEHM, AND M. PALUSZNY, *Bézier and B-Spline Techniques*, Mathematics and Visualization, Springer Berlin Heidelberg, 2002.
- [80] M. RAJIH, P. COMON, AND R. A. HARSHMAN, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM J. Matrix Anal. & Appl., 30 (2008), pp. 1128–1147.
- [81] W. S. RAYENS AND B. C. MITCHELL, *Two-factor degeneracies and a stabilization of PARAFAC*, Chemometr. Intell. Lab., 38 (1997), pp. 173–181.
- [82] T. REES, H. S. DOLLAR, AND A. J. WATHEN, *Optimal solvers for PDE-constrained optimization*, SIAM J. Sci. Comput., 32 (2010), pp. 271–298.

- [83] T. REES AND A. J. WATHEN, *Preconditioning iterative methods for the optimal control of the Stokes equations*, SIAM J. Sci. Comput., 33 (2011), pp. 2903–2926.
- [84] S. ROY AND B. A., *A new optimization approach to sparse reconstruction of log-conductivity in acusto-electric tomography*, SIAM J. Imaging Sci., 11 (2018), pp. 1759–1784.
- [85] G. SANGALLI AND M. TANI, *Matrix-free isogeometric analysis: The computationally efficient k-method*, Research Gate, (2017).
- [86] D. SCHILLINGER, L. DEDE, M. SCOTT, J. EVANS, M. BORDEN, E. RANK, AND T. HUGHES, *An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and t-spline CAD surfaces*, Comp. Meth. Appl. Mech. Engrg., 249–252 (2012), pp. 116–150.
- [87] D. SCHILLINGER, J. A. EVANS, A. REALI, M. A. SCOTT, AND T. J. HUGHES, *Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations*, Comput. Method. Appl. M., 267 (2013), pp. 170–232.
- [88] U. SCHOLLWÖCK, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys-new. York., 326 (2011), pp. 96–192.
- [89] F. SCHOLZ, A. MANTZAFLARIS, AND B. JÜTTLER, *Partial tensor decomposition for decoupling isogeometric Galerkin discretizations*, Comput. Method. Appl. M., 336 (2018), pp. 485–506.
- [90] V. SHARAN AND G. VALIANT, *Orthogonalized ALS: A theoretically principled tensor decomposition algorithm for practical use*, in Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML2017, JMLR, 2017, pp. 3095–3104.
- [91] S. SHOJAEI, E. IZADPANAH, AND S. NAZARI, *Imposition of essential boundary conditions in a transient heat conduction problem based on Isogeometric Analysis*, Scientia Iranica B, 21 (2014), pp. 1962–1972.

- [92] S. SHOJAEI, E. IZADPENAH, AND A. HAERI, *Imposition of essential boundary conditions in Isogeometric Analysis using the Lagrange multiplier method*, International Journal of Optimization in Civil Engineering, 2 (2012), pp. 247–271.
- [93] M. STOLL, *Fast Iterative Solvers for Time Dependent PDE-Constrained Optimization Problems*, habilitation, Otto-von-Guericke Universität Magdeburg, 2016.
- [94] M. STOLL AND T. BREITEN, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci. Comput., 37 (2015), pp. B1–B29.
- [95] G. STRANG AND G. FIX, *An Analysis of the Finite Element Method*, Wellesley-Cambridge Press, 2008.
- [96] S. SUBRAMANIAN, K. SCHEUFELE, M. MEHL, AND G. BIROS, *Where did the tumor start? an inverse solver with sparse localization for tumor growth models*, Inverse Probl., 36 (2020), p. 045006.
- [97] G. TOMASI AND R. BRO, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Statist. Data Anal., 50 (2006), pp. 1700–1734.
- [98] L. N. TREFETHEN AND D. BAU, *Numerical Linear Algebra*, SIAM, Jan. 1997.
- [99] F. TRÖLTZSCH, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, American Mathematical Society, 2010.
- [100] L. R. TUCKER, *Some mathematical notes on the three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
- [101] M. ULBRICH AND S. ULBRICH, *Primal-dual-interior-point methods for PDE-constrained optimization*, Math. Program., 117 (2009), pp. 435–485.
- [102] 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.
- [103] C. A. ZARZER, *On tikhonov regularization with non-convex sparsity constraints*, Inverse Probl., 25 (2009), p. 025006.

- [104] J. C. ZIEMS AND S. ULBRICH, *Adaptive multilevel inexact SQP methods for PDE-constrained optimization*, SIAM J. Optim., 21 (2011), pp. 1–40.