CONSTRAINED OPTIMIZATION WITH LOW-RANK TENSORS AND APPLICATIONS TO PARAMETRIC PROBLEMS WITH PDEs

SEBASTIAN GARREIS* AND MICHAEL ULBRICH†

Abstract. Low-rank tensor methods provide efficient representations and computations for high-dimensional problems and are able to break the curse of dimensionality when dealing with systems involving multiple parameters. We present algorithms for constrained nonlinear optimization problems that use low-rank tensors and apply them to optimal control of PDEs with uncertain parameters and to parametrized variational inequalities. These methods are tailored to the usage of low-rank tensor arithmetics and allow to solve huge scale optimization problems. In particular, we consider a semismooth Newton method for an optimal control problem with pointwise control constraints and an interior point algorithm for an obstacle problem, both with uncertainties in the coefficients.

Key words. Nonlinear optimization, low-rank tensors, PDEs with uncertainties, optimal control under uncertainty, parametric variational inequalities, uncertainty quantification, semismooth Newton methods, interior point methods

1. Introduction. In this paper we develop low-rank tensor methods for solving problems that arise from discretizations of inequality-constrained optimization problems that involve PDEs with uncertain parameters. Especially, we consider optimal control of a PDE under uncertainty, which results in robust optimal solutions. Further, we address the computation of the whole parametric solution of variational inequalities (VIs) of obstacle type involving uncertainties. This provides a basis for uncertainty quantification (UQ) of solutions to these VIs.

In many application areas, for example when dealing with systems with many parameters like in uncertainty quantification or with many free variables like in quantum mechanics, tensors can be used to describe the state of these systems. In the finite-dimensional case a tensor is a multidimensional array, which, e.g., can represent a multivariate function by its values on a structured grid. If there are d dimensions and n degrees of freedom per dimension, then the resulting tensor is of order d and has n^d elements. Storage and computational time thus scale like $\mathcal{O}(n^d)$. The curse of dimensionality caused by the exponential dependence on the order d of the tensor renders working with full tensors intractable already for moderate orders. Thus, more efficient tensor representations are required, and for instance in quantum physics such approaches are in use for quite some time already, see, e.g., [34, 24] for references. In the last years, significant progress has been made in the development of a mathematically rigorous theory and algorithms for such low-rank tensor formats; see the book [17] and the surveys [16, 27] for an overview. Classical tensor decompositions either lack important theoretical properties – such as the well-posedness of the low-rank approximation problem, which is violated for the canonical polyadic (CP) decomposition – or have complexity issues – such as, e.g., the Tucker decomposition. Current investigations therefore deal with the approximation of tensors in new low-rank formats that combine low complexity with good theoretical and computational properties. In particular, the tensor train (TT) format [39, 40] and the hierarchical Tucker decomposition (HTD) [19, 32] meet the requirements of scaling

*Technical University of Munich, Chair of Mathematical Optimization, Department of Mathematics, Boltzmannstr. 3, 85748 Garching b. München, Germany (garreis@ma.tum.de).

†Technical University of Munich, Chair of Mathematical Optimization, Department of Mathematics, Boltzmannstr. 3, 85748 Garching b. München, Germany (mulbrich@ma.tum.de).

well in the order and the dimensions of the tensor. They also are attractive from a theoretical and algorithmic perspective. Currently, TT and HTD approximations of tensors are intensively investigated, using different techniques such as generalizations of the singular value decomposition [6, 15], pointwise evaluations [37, 2], and minimization of least squares functionals. The latter requires optimization algorithms for unconstrained problems with tensors of fixed or adaptive low rank, for example block coordinate descent methods like (M)ALS [23], DMRG [38], and AMEn [9], or Riemannian optimization approaches that work on low-rank tensor manifolds [5, 29]. Another field of research deals with the solution of large linear systems by iterative methods such as preconditioned conjugate gradient (PCG) algorithms for tensors in HTD [31], which is also closely related to the minimization of quadratic functionals. All these algorithms benefit from the good complexity of low-rank tensors.

In contrast to the mentioned methods for unconstrained optimization, we propose algorithms for *inequality constrained* optimization in this paper and apply them to optimal control problems with PDEs under uncertainty and to variational inequalities of obstacle type with uncertain coefficients. Exploiting the good complexity of low-rank tensors, the proposed algorithms are able to solve systems with multiple parameters. Compared to other methods that were recently applied in uncertainty quantification or for dealing with uncertainties in general, they enjoy attractive complexity properties, which is especially important for problems in high dimensions. Sparse grids can also reduce the computational complexity drastically, but in general still scale exponentially in the number of parameters [14] whereas low-rank tensors feature linear scaling. Sparse grid techniques can be further improved by constructing adaptive algorithms as in [28]. Monte Carlo (MC) type methods are known for their quite slow, but dimension-independent approximation properties. Accelerations of MC are possible, e.g., by Quasi-Monte Carlo [35, 33] or multi-level approaches [3].

We consider two types of constrained optimization problems, both involving an elliptic operator with uncertain coefficients. We formulate them in an appropriate tensor Hilbert space and discretize them by finite elements (FE) for the space variable and polynomial chaos for the random parameters.

In the first problem class, the PDE under uncertainty arises as the state equation in an optimal control problem with control constraints. The discretized state is represented by a tensor and the state equation results in a linear operator equation in tensor space. We reduce the optimal control problem to the control and apply the adjoint approach to compute the gradient of the reduced objective function. The state and adjoint equations are solved using AMEn [9], a block iterative method that operates on the core tensors of the TT representation. Based on this, the reduced objective function and its gradient can be evaluated. A semismooth Newton method is applied to the optimality conditions of the reduced problem. For solving the semismooth Newton system, a block elimination and CG are used.

As a second problem class, we consider a variational inequality of obstacle type with uncertain coefficients. After discretization, this is a quadratic optimization problem in a finite-dimensional tensor space with element-wise constraints on the whole tensor, for which we develop a primal interior point method. The first and second derivative of the log-barrier term require computing element-wise reciprocals of tensors. For this we use a Newton-Schulz iteration or, alternatively, AMEn. The solution of the barrier-Newton system is done by either AMEn or by a low-rank tensor PCG method [31]. Feasible step lengths are determined by the computation of the maximum element of a tensor, for which we use global optimization on the represented

continuous function.

This paper is organized as follows: We start with an introduction to tensors and low-rank formats in the next section. Then, our setting will be motivated by two example applications (section 3): the mentioned optimal control problem with uncertain coefficients in the state equation and the multi-parametric obstacle problem and their discretization using tensors. We discuss our algorithms for the solution of the presented problems in section 4. The numerical tests in section 5 show their efficiency and accuracy. Conclusions and an outlook are given in section 6.

2. Tensors and low-rank formats.

2.1. Basics and notation. In this paper, we denote by a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ a d-dimensional array of real numbers (cf. [32, sec. 1], where $\mathbb{C}$ is used as underlying field). These numbers could be values of a function on a d-dimensional rectilinear grid. The array dimension d is called the *order* of the tensor and the *modes* are numbered by $1, \ldots, d$. We denote the dimension of the i-th mode by n_i and write $\mathbf{x}(k_1, \ldots, k_d)$ for the $(k_1, \ldots, k_d)$-component of $\mathbf{x}$, where $k_i \in [n_i]$, $i \in [d]$, writing $[m] \coloneqq \{1, \ldots, m\}$ here and throughout. The index notation in parentheses is better readable than subscripts like $\mathbf{x}_{k_1,\ldots,k_d}$. The tensors of all ones will be denoted by $\mathbb{1}$ if the dimensions are clear. Reshaping a tensor as a vector in a certain order (e.g., reverse lexicographically as in [32]) will be written as $\mathrm{vec}(\mathbf{x}) \in \mathbb{R}^{n_1 \cdots n_d}$. *Matricization*, i.e., reshaping as a matrix is denoted by $\mathbf{x}^{(t)} \in \mathbb{R}^{(\prod_{i \in t} n_i) \times (\prod_{j \notin t} n_j)}$, where $t \subset [d]$ denotes the set of modes that become the rows of the resulting matrix. Analogously we write $\mathrm{ten}(x)$ for reshaping a vector or a matrix x back into a tensor when the dimensions are clear. For the extraction of parts of a tensor we also allow indexing with sets and write "$\bullet$" for the full index range of a dimension: $\mathbf{x}(\bullet, 4, \{1, 5\}, \bullet, \ldots, \bullet) \in \mathbb{R}^{n_1 \times 2 \times n_4 \times \ldots \times n_d}$ is obtained from $\mathbf{x}$ by fixing the second index at 4 and taking the first and fifth components of the third mode and all components of the remaining modes. Note that modes that are indexed by a single index are cut out in the result, while modes indexed by a set remain.

As tensors form a vector space, all vector space operations are defined for tensors. Additionally, element-wise operations are useful: $\mathbf{x} \odot \mathbf{y}$ denotes multiplication, $\mathbf{x}./\mathbf{y}$ division and $\mathbf{x}.^{\lambda}$ exponentiation. The element-wise application of a function $f : \mathbb{R} \to \mathbb{R}$ is written as $f(\mathbf{x})$.

Definition 2.1 (i-mode matrix product). Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ be a tensor, $i \in [d]$ a mode, and $A : \mathbb{R}^{n_i} \to \mathbb{R}^m$ a linear operator (or a matrix). Then the i-mode matrix product $A \circ_i \mathbf{x} \in \mathbb{R}^{n_1 \times \ldots \times n_{i-1} \times m \times n_{i+1} \times \ldots \times n_d}$ is defined by $(A \circ_i \mathbf{x})(k_1, \ldots, k_d) \coloneqq (A\mathbf{x}(k_1, \ldots, k_{i-1}, \bullet, k_{i+1}, \ldots, k_d))(k_i)$ for all $(k_1, \ldots, k_d) \in [n_1] \times \ldots \times [n_{i-1}] \times [m] \times [n_{i+1}] \times \ldots \times [n_d]$. In short notation: $A \circ_i \mathbf{x} = \mathrm{ten}\left(A\mathbf{x}^{(\{i\})}\right)$ (cf. [32, sec. 4.1]).

This definition means that we take all vectors that result from fixing all indices of the tensor $\mathbf{x}$ except the i-th one, apply A, and use the resulting vectors to form $A \circ_i \mathbf{x}$. We view tensors of order 1 as column vectors and order-2 tensors as matrices where the first index is for the rows. The contraction of two tensors is a further important operation. It generalizes inner and outer products as well as matrix multiplication.

Definition 2.2 (tensor contraction). Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ and $\mathbf{z} \in \mathbb{R}^{\tilde{n}_1 \times \ldots \times \tilde{n}_{\tilde{d}}}$ be tensors and $s = (s_1, \ldots, s_p)$, $t = (t_1, \ldots, t_p)$ with $s_i \in [d]$, $t_i \in [\tilde{d}]$ be ordered lists of modes that shall be contracted. Further, let $\bar{s}_i$, $1 \le i \le d - p$, and $\bar{t}_i$, $1 \le i \le \tilde{d} - p$, be the remaining, untouched modes in ascending order. Then the *contraction* $\langle \mathbf{x}, \mathbf{z} \rangle_{s,t} \in$

$\mathbb{R}^{\bar{s}_1\times\ldots\times\bar{s}_{d-p}\times\bar{t}_1\times\ldots\times\bar{t}_{\tilde{d}-p}}$ of $\mathbf{x}$ and $\mathbf{z}$ along the modes s and t is defined as

$$\langle \mathbf{x}, \mathbf{z}\rangle_{s,t}(k_{\bar{s}_1},\ldots,k_{\bar{s}_{d-p}},\ell_{\bar{t}_1},\ldots,\ell_{\bar{t}_{\tilde{d}-p}}) = \sum_{k_{s_1},\ldots,k_{s_p}=1}^{n_{s_1},\ldots,n_{s_p}} \mathbf{x}(k_1,\ldots,k_d)\mathbf{z}(\ell_1,\ldots,\ell_{\tilde{d}})|_{\ell_{t_i}=k_{s_i}\,\forall i\in[p]}$$

element-wise for all indices $k_{\bar{s}_i} \in [n_{\bar{s}_i}]$ $(i \in [d-p])$ and $\ell_{\bar{t}_j} \in [\tilde{n}_{\bar{t}_j}]$ $(j \in [\tilde{d}-p])$. When t or s are sets rather than ordered lists, then they stand for the lists of elements in ascending order.

Here, similar to a matrix product, a component of the resulting tensor is obtained by fixing indices in the untouched modes and computing an inner product of the resulting tensors of the same size. Tensor contraction can be nicely visualized by tensor network diagrams [32, sec. 5.1]. We sometimes need the following special cases of tensor contraction:

DEFINITION 2.3 (special cases of tensor contraction). Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_1\times\ldots\times n_d}$ and $\mathbf{z} \in \mathbb{R}^{\tilde{n}_1\times\ldots\times\tilde{n}_{\tilde{d}}}$ be tensors.

- The *outer product*[1] $\mathbf{x}\otimes\mathbf{z} := \langle \mathbf{x}, \mathbf{z}\rangle_{\emptyset,\emptyset} \in \mathbb{R}^{n_1\times\ldots\times n_d\times\tilde{n}_1\times\ldots\times\tilde{n}_{\tilde{d}}}$ of the tensors $\mathbf{x}$ and $\mathbf{z}$ is obtained as

$$(\mathbf{x}\otimes\mathbf{z})(k_1,\ldots,k_d,\ell_1,\ldots,\ell_{\tilde{d}}) = \mathbf{x}(k_1,\ldots,k_d)\mathbf{z}(\ell_1,\ldots,\ell_{\tilde{d}}).$$

- An *elementary tensor* (or rank-1-tensor) $\bigotimes_{i=1}^d u^i \in \mathbb{R}^{n_1\times\ldots\times n_d}$ is an outer product of vectors:

$$\Big(\bigotimes_{i=1}^d u^i\Big)(k_1,\ldots,k_d) = \prod_{i=1}^d u^i_{k_i},$$

  where $u^i_{k_i}$ is the k_i-th component of the vector $u^i \in \mathbb{R}^{n_i}$.
- The *inner product* $\langle \mathbf{x}, \mathbf{y}\rangle := \langle \mathbf{x}, \mathbf{y}\rangle_{[d],[d]} \in \mathbb{R}$ of two tensors of the same size is

$$\langle \mathbf{x}, \mathbf{y}\rangle = \sum_{k_1=1}^{n_1} \ldots \sum_{k_d=1}^{n_d} \mathbf{x}(k_1,\ldots,k_d)\mathbf{y}(k_1,\ldots,k_d).$$

- The *Frobenius norm* of a tensor $\mathbf{x}$ is $\|\mathbf{x}\|_F := \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$.

The i-mode matrix product is a matrix-tensor contraction: $A \circ_i \mathbf{x} = \langle A, \mathbf{x}\rangle_{\{2\},\{i\}}$.

2.2. Low-rank tensor formats. As the amount of data of a full tensor is in general very high (of order $\mathcal{O}(n_{\max}^d)$ with $n_{\max} := \max_{i\in[d]} n_i$), it is important to find a representation or approximation of it that requires substantially less memory. For instance, a tensor can be approximated by a rank-R-tensor, which is the sum of R elementary tensors:

$$\mathbf{x} \approx \sum_{j=1}^R \bigotimes_{i=1}^d u^{i,j} \quad \text{with} \quad u^{i,j} \in \mathbb{R}^{n_i} \text{ for } i \in [d],\ j \in [R].$$

This is also known as *canonic polyadic (CP) decomposition* (cf. [17, ch. 7]) and requires storage of order $\mathcal{O}(Rn_{\max}d)$. We note that this seems to be linear in d instead of exponential in the case of storing the full tensor. But one should take into account that R usually depends on the approximation accuracy, which again can

[1] Note that the vectorization/matricization of the outer product of two tensors coincides with the standard Kronecker product (also denoted by "$\otimes$") of vectorizations/matricizations of them.

depend on d. Furthermore, this format has certain drawbacks such as the possible ill-posedness of the best approximation of a given tensor by a tensor of rank at most R [7].

A widely used approach for the matrix case $d = 2$ is a low-rank approximation by a truncated singular value decomposition (SVD). The quadratic error (squared Frobenius norm) made by this approximation is bounded by the sum of the truncated, squared singular values [17, Lemma 2.30] and can thus be estimated easily. The SVD approximation is optimal w.r.t. this error in the set of matrices of a fixed maximum rank. A generalization of this approximation technique to d-dimensional tensors is therefore desirable, but cannot be available in the CP format due to the ill-posedness of the approximation problem. Thus, a different approach works with subspaces of the $\mathbb{R}^{n_i}$, as done in the *Tucker format* (or *tensor subspace representation* [17, ch. 8]). Here, for representing a tensor $\mathbf{x} \in \bigotimes_{i=1}^{d} \mathbb{R}^{n_i}$, bases of r_i-dimensional subspaces of the $\mathbb{R}^{n_i}$ are selected, stored as matrices $U_i \in \mathbb{R}^{n_i \times r_i}$ and a basis $\bigotimes_{i=1}^{d} U_i$ of the tensor product of subspaces is formed. One can combine the basis elements linearly with coefficients stored in the so-called core tensor $\mathbf{c} \in \mathbb{R}^{r_1 \times \ldots \times r_d}$ to obtain an approximation of $\mathbf{x}$. For this format the *HOSVD* (higher order SVD, which uses the standard SVD in substeps) is available for truncating a tensor to lower Tucker rank giving a quasi-optimal result[2] and offering error control similar to the one in standard SVD [6]. Unfortunately, defining $r_{\max} := \max_{i \in [d]} r_i$, the required storage for the Tucker format is $\mathcal{O}(n_{\max} r_{\max} d + {r_{\max}}^d)$ and grows exponentially in d.

A generalization and improvement of this idea is the *hierarchical Tucker decomposition (HTD)* [19] used in the `htucker` MATLAB toolbox [32]. It uses a binary tree $\mathcal{T}$ and stores a coefficient vector at the root, coefficient matrices at interior nodes, and basis matrices at the leaves. The nodes of the tree are denoted by nonempty subsets of $[d]$; the root node is $[d]$ while the d leaf nodes are (from left to right) $\{1\}, \ldots, \{d\}$. For the remaining nodes there holds that, when the non-leaf node t has left and right children $\mathrm{l}, \mathrm{r} \subset [d]$, then $\mathrm{l} \neq \varnothing \neq \mathrm{r}$ and $\mathrm{t} = \mathrm{l} \cup \mathrm{r}$. As a convention, we require $i < j$ for all $i \in \mathrm{l}$ and all $j \in \mathrm{r}$. In HTD the tensor $\mathbf{x}$ is recursively decomposed. In a first step, it is written as

$$\mathrm{vec}(\mathbf{x}) = (U_\mathrm{a} \otimes U_\mathrm{b}) B_{[d]},$$

where a and b are the left and right children of the root $[d]$, with suitably chosen subspace bases $U_\mathrm{a} \in \mathbb{R}^{n_\mathrm{a} \times r_\mathrm{a}}$ and $U_\mathrm{b} \in \mathbb{R}^{n_\mathrm{b} \times r_\mathrm{b}}$, where $n_\mathrm{t} := \prod_{i \in \mathrm{t}} n_i$ is defined for all nodes t. The vector $B_{[d]} \in \mathbb{R}^{r_\mathrm{a} r_\mathrm{b}}$ stores the coefficents. This is just a low-rank decomposition of the matricization $\mathbf{x}^{(\mathrm{a})}$. The crucial point is that only $B_{[d]}$ is stored, but *not* the matrices U_a and U_b. The latter are again factored in a similar way as the root tensor:

Inductively, consider an interior node t with a subspace basis $U_\mathrm{t} \in \mathbb{R}^{n_\mathrm{t} \times r_\mathrm{t}}$ (such as the two root children a, b from above). Denote by l and r the children of t. Now U_t is decomposed as

$$U_\mathrm{t} = (U_\mathrm{l} \otimes U_\mathrm{r}) B_\mathrm{t}$$

with suitably chosen subspace bases $U_\mathrm{l} \in \mathbb{R}^{n_\mathrm{l} \times r_\mathrm{l}}$ and $U_\mathrm{r} \subset \mathbb{R}^{n_\mathrm{r} \times r_\mathrm{r}}$ and the coefficient matrix $B_\mathrm{t} \in \mathbb{R}^{r_\mathrm{l} r_\mathrm{r} \times n_\mathrm{t}}$. Again, only the matrix B_t is stored at node t, but *not* the basis U_t. This procedure is continued. At non-leaf nodes, basis matrices are decomposed

[2] Best approximation up to a factor that depends on the order d of the tensor ($\sqrt{d}$ for the Tucker format, see, e.g., [16]).

as described, while at every leaf node $\{i\}$ the basis matrix $U_i = U_{\{i\}} \in \mathbb{R}^{n_i \times r_i}$ is stored, where $n_i = n_{\{i\}}$ and $r_i = r_{\{i\}}$.

Thus, the tensor is represented by the vector $B_{[d]} \in \mathbb{R}^{r_{\mathrm{a}} r_{\mathrm{b}}}$, the matrices $B_{\mathrm{t}} \in \mathbb{R}^{r_{\mathrm{l}} r_{\mathrm{r}} \times r_{\mathrm{t}}}$ at the $d-2$ interior nodes (the B_{t} at non-leaf nodes are also called transfer matrices), and the basis matrices $U_i \in \mathbb{R}^{n_i \times r_i}$ at the d leaf nodes. The storage requirement is $\mathcal{O}(n_{\max} r_{\max} d + {r_{\max}}^3 d)$, hence linear in d, where $n_{\max}$ and $r_{\max}$ are upper bounds for the n_i and r_{t}. The vector of numbers r_{t}, $\mathrm{t} \in \mathcal{T}$, is called *hierarchical Tucker (HT) rank*. Given this representation, the full tensor can be recovered by constructing the basis matrices U_{t} in a leaves-to-root sweep, noting that $\mathrm{vec}(\mathbf{x}) = U_{[d]}$. One can show that any tensor satisfying $\mathrm{rank}\, \mathbf{x}^{(\mathrm{t})} \le r_{\mathrm{t}}$ for all $\mathrm{t} \in \mathcal{T}$ can be represented exactly in this format with at most this HT rank.

The `htucker` toolbox [32] implements tensors in this format and efficient versions of the most important operations, e.g., extraction of parts of the tensor, application of linear operators to the i-th mode, contraction, tensor orthogonalization and truncation to a lower rank by HOSVD for HTD [15] with error control and quasi-optimal approximation up to the factor $\sqrt{2d-3}$. This truncation is crucial because when using the standard implementations for summing or multiplying two tensors elementwise, the ranks sum or multiply, respectively. Therefore, special truncated versions of these operations are available. All these operations can be performed with a reasonable time complexity (linearly in $n_{\max}$ and d and polynomially in $r_{\max}$, at most $\mathcal{O}(n_{\max} d {r_{\max}}^2 + d {r_{\max}}^4)$, but often less).

Closely related is the *tensor train (TT) format* [39], also known as *matrix product states* (MPS) in the quantum physics community. It has a similarly nice structure as HTD and its complexity as well as the available operations are comparable to HTD. The `TT-Toolbox` [40] provides implementations in MATLAB and in PYTHON. The TT rank of a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ is a vector $r := (r_0, r_1, \dots, r_d)$ with $r_0 = r_d = 1$ and $r_i \in [n_i]$ for $i \in \{2, \dots, d-1\}$. In the TT format, $\mathbf{x} = \mathcal{TT}(\mathbf{u})$ is represented by a d-tuple $\mathbf{u} := (\mathbf{u}_1, \dots, \mathbf{u}_d) \in \bigtimes_{i=1}^{d} \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ of order-3-tensors as follows:

$$\mathbf{x}(k_1, \dots, k_d) = \mathbf{u}_1(\bullet, k_1, \bullet)\mathbf{u}_2(\bullet, k_2, \bullet) \cdots \mathbf{u}_n(\bullet, k_n, \bullet) \in \mathbb{R} \quad \forall\, k_i \in [n_i],\ i \in [d].$$

Note that this is a simple product of matrices, resulting in a scalar value because the first and the last matrix of the chain are a row and column vector, respectively. As the HTD is somewhat more general, we started working with that format. During our work it turned out that some methods, which are only available for TT tensors, yielded better results. Thus we finally worked with HT tensors with linear (instead of usually balanced) dimension trees. This allowed for quite simple conversions between the two formats where necessary (cf. [17, sec. 12.2.2]).

3. Applications: problem formulation and discretization. We will use the good complexity of low-rank tensors to solve optimal control problems with PDEs as well as variational inequalities, both with uncertain parameters. To describe the setting, let $D \subset \mathbb{R}^n$ be an open, bounded domain with Lipschitz boundary ∂D and consider the following elliptic PDE with a parameter-dependent coefficient κ_α which depends continuously on the parameter vector $\alpha \in \mathbb{R}^m$:

$$-\mathrm{div}\,(\kappa_\alpha \nabla y_\alpha) = b \quad (\text{in } D), \qquad y_\alpha = 0 \quad (\text{on } \partial D). \tag{1}$$

Here, for fixed α, the state y_α lives in $Y := H_0^1(D)$. In the optimal control problem considered in section 3.1, this parametric PDE will describe a state equation under uncertainty and the right hand side $b \in Y^*$ will depend on a control $u \in U_{\mathrm{ad}} \subset U :=$

$L^2(D)$. The application in section 3.2 will consider a variational inequality of obstacle type under uncertainty, where the underlying parametric differential operator is of the same form as in (1). This then additionally involves pointwise inequality constraints on y_α.

The parameters $\alpha \in \mathbb{R}^m$, on which κ_α depends, are considered to be uncertain; the random variables α_i are assumed to be independently distributed with values in $\Gamma_i \subset \mathbb{R}$ and with induced probability measures $\mathbb{P}_{\alpha_i}$ on Γ_i. We set $\Gamma = \bigtimes_{i=1}^m \Gamma_i \subset \mathbb{R}^m$ and denote by $\mathbb{P}_\alpha$ the product measure. For uniform well-posedness, we assume the coefficient to be uniformly positive and bounded: $0 < \kappa_{\min} \le \kappa_\alpha(x) \le \kappa_{\max} < \infty$ for almost all $x \in D$ and all $\alpha \in \Gamma$.

Given the right hand side $b \in Y^*$ and a realization $\alpha \in \mathbb{R}^m$ of the uncertain parameters, the corresponding state $y_\alpha \in Y = H_0^1(D)$ is defined as the weak solution of (1), which, in variational form, reads:

$$(\kappa_\alpha \nabla y_\alpha, \nabla v)_{L^2(D)^n} = \langle b, v \rangle_{H^{-1}(D), H_0^1(D)} \quad \forall\ v \in H_0^1(D). \tag{2}$$

In cases where the coefficient is a more general random field $\kappa(x, \omega)$ one can (under suitable conditions) use the Karhunen-Loève expansion to get an approximation κ_α, which depends on finitely many, uncorrelated random variables (see, e.g., [13]) and can be treated as done here if these random variables are in fact independent.

Introducing the operator

$$A(\alpha) \in \mathcal{L}(Y, Y^*), \quad \langle A(\alpha) y,\ v \rangle_{Y^*, Y} := (\kappa_\alpha \nabla y, \nabla v)_{L^2(D)^n} \quad \forall\ y, v \in Y, \tag{3}$$

we can write (2) as a parametric linear operator equation

$$A(\alpha) y_\alpha = b. \tag{4}$$

One could also take the right hand side b of the equation or the domain D to be parameter-dependent, but to focus the presentation we keep them fixed. In order to solve this parametrized equation one could insert any realization of α and solve (4) by a suitable method (e.g., standard finite element discretization). The more parameter values are inserted, the more equations have to be solved and the computational effort grows.

A different perspective is to express the dependence on α by writing $\boldsymbol{y}(x, \alpha) = y_\alpha(x)$ with $\boldsymbol{y} \in \boldsymbol{Y} := H_0^1(D) \otimes L^2(\Gamma_1, \mathbb{P}_{\alpha_1}) \otimes \ldots \otimes L^2(\Gamma_m, \mathbb{P}_{\alpha_m}) = H_0^1(D) \otimes L^2(\Gamma, \mathbb{P}_\alpha)$ (cf. [17, sec. 4.2.3]), meaning that $\boldsymbol{y}$ is a H_0^1-function w.r.t. the space variable x and has finite variance w.r.t. the random variables α. In the following, we identify the dual space of $L^2(\Gamma_i, \mathbb{P}_{\alpha_i})$ with $L^2(\Gamma_i, \mathbb{P}_{\alpha_i})$. The Hilbert space $\boldsymbol{Y}$ is the closure of the algebraic tensor product space $H_0^1(D) \otimes_a L^2(\Gamma_1, \mathbb{P}_{\alpha_1}) \otimes_a \ldots \otimes_a L^2(\Gamma_m, \mathbb{P}_{\alpha_m}) := \operatorname{span}(y \otimes v_1 \otimes \ldots \otimes v_m, y \in H_0^1(D), v_i \in L^2(\Gamma_i, \mathbb{P}_{\alpha_i}), i \in [m])$ (with the outer product of real valued functions $(y \otimes v_1 \otimes \ldots \otimes v_m)(x, \alpha_1, \ldots, \alpha_m) := y(x) v_1(\alpha_1) \cdots v_m(\alpha_m)$) with respect to an appropriate norm, here $\|\boldsymbol{y}\|_{\boldsymbol{Y}} := \left(\int_\Gamma \int_D \boldsymbol{y}(x, \alpha)^2 + \|\nabla_x \boldsymbol{y}(x, \alpha)\|_2^2 \, dx \, d\mathbb{P}_\alpha \right)^{1/2}$. It is isomorphic to the Bochner space $L^2_{\mathbb{P}_\alpha}\left(\Gamma; H_0^1(D)\right)$ of $H_0^1(D)$-valued mappings on Γ with finite variance w.r.t. $\mathbb{P}_\alpha$ and its dual space $\boldsymbol{Y}^*$ can be identified with $H^{-1}(D) \otimes L^2(\Gamma_1, \mathbb{P}_{\alpha_1}) \otimes \ldots \otimes L^2(\Gamma_m, \mathbb{P}_{\alpha_m})$. More detailed explanations can be found in [12] as well as in [17, ch. 3 and 4].

Now we write (4) as an equation in the tensor space, in a weak sense also w.r.t.

the parameters:

$$\begin{aligned} \boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{A} : \boldsymbol{Y} \to \boldsymbol{Y}^*, \ \langle \boldsymbol{A}\boldsymbol{y}, \boldsymbol{v} \rangle_{\boldsymbol{Y}^*, \boldsymbol{Y}} = \int_\Gamma (\kappa_\alpha \nabla_x \boldsymbol{y}(\cdot, \alpha), \nabla_x \boldsymbol{v}(\cdot, \alpha))_{L^2(D)^n} \, d\mathbb{P}_\alpha, \\ \boldsymbol{b} \in \boldsymbol{Y}^*, \ \langle \boldsymbol{b}, \boldsymbol{v} \rangle_{\boldsymbol{Y}^*, \boldsymbol{Y}} = \int_\Gamma \langle b, \boldsymbol{v}(\cdot, \alpha) \rangle_{Y^*, Y} \, d\mathbb{P}_\alpha \quad \forall \ \boldsymbol{v} \in \boldsymbol{Y}. \end{aligned} \tag{5}$$

Due to the Lax-Milgram theorem this equation has also a unique solution in $\boldsymbol{Y}$, where it would be sufficient to take $\kappa \in L^\infty(D \times \Gamma, \lambda \otimes \mathbb{P}_\alpha)$ (with the Lebesgue measure λ on D) and to be uniformly positive and bounded from above almost everywhere on $D \times \Gamma$ (cf. [13]). Note that above we assumed the boundedness of κ indeed for all $\alpha \in \Gamma$ and assumed continuous dependence to be able to insert any value of α into (4) and to apply quadrature formulae or sampling methods later. Furthermore, the unique solution of (5) can then be constructed by solving (4) for any value of α, which yields the equivalence of the two formulations.

EXAMPLE 3.1. To be more concrete, let us divide the domain D into open subsets $D_i \subset D$ ($i \in [m]$), which are pairwise disjoint ($D_i \cap D_j = \varnothing$ for $i \neq j$) and cover the whole domain in the sense that $\overline{D} = \overline{\bigcup_{i=1}^m D_i}$. On each subset the coefficient will be disturbed independently. We model this by an average coefficient function $\kappa_0(x)$ and by functions $\nu_i(x)$ ($i \in [m]$) that describe the influence of the parameter α_i over the domain D. We set $\kappa_\alpha(x) := \kappa_0(x)\,(1 + \sum_{i=1}^m \alpha_i \nu_i(x))$. To be more specific, let us take κ_0 to be constant on each subdomain ($\kappa_0|_{D_i} \equiv \sigma_i$, $\sigma \in \mathbb{R}^m_{>0}$) and the random variables α to be uniformly distributed on the open interval $(-1, 1)$. The functions ν_i are supported on D_i and have the form $\nu_i(x) := \vartheta_i \cdot 1_{D_i}(x)$ with $\vartheta \in [0, 1)^m$. Then the assumptions from above are fulfilled and we get an elliptic operator $\boldsymbol{A}$. We will use this example in our numerical tests later. In this case, the operator $\boldsymbol{A}$ has the following useful structure:

LEMMA 3.2. *The operator $\boldsymbol{A}$ given in (5) with $\kappa_\alpha(x) := \kappa_0(x)\,(1 + \sum_{i=1}^m \alpha_i \nu_i(x))$ as in Example 3.1 can be written as*

$$\boldsymbol{A} = A_0 \otimes \Big(\bigotimes_{j=1}^m \mathrm{Id}\Big) + \sum_{i=1}^m A_i \otimes \Big(\bigotimes_{j=1}^m S_{ij}\Big), \tag{6}$$

with $A_0, A_i \in \mathcal{L}(Y, Y^*)$ and $S_{ij} \in \mathcal{L}(L^2(\Gamma_j, \mathbb{P}_{\alpha_j}), L^2(\Gamma_j, \mathbb{P}_{\alpha_j}))$ defined by

$$\begin{aligned} &\langle A_0 y, v \rangle_{Y^*, Y} := (\kappa_0 \nabla y, \nabla v)_{L^2(D)^n}, \quad \langle A_i y, v \rangle_{Y^*, Y} := (\nu_i \kappa_0 \nabla y, \nabla v)_{L^2(D)^n}, \\ &S_{ij} = \mathrm{Id} \ \text{ for } i, j \in [m],\ i \neq j, \quad (S_{jj} v_j)(\alpha_j) := \alpha_j \cdot v_j(\alpha_j) \ \text{ for } j \in [m]. \end{aligned}$$

Here, the Kronecker product of linear operators is defined on the algebraic tensor space by their action on elementary tensors [17, sec. 3.3]. In the current case we have for example

$$(A_i \otimes S_{i1} \otimes \ldots \otimes S_{im})(y \otimes v_1 \otimes \ldots \otimes v_m) := (A_i y) \otimes (S_{i1} v_1) \otimes \ldots \otimes (S_{im} v_m).$$

Proof. Using the structure of $A(\alpha)$ and of κ_α, we obtain:

$$\begin{aligned} \langle A(\alpha) y, v \rangle_{Y^*, Y} &= (\kappa_0 \nabla y, \nabla v)_{L^2(D)^n} + \sum_{i=1}^m (\kappa_0 \alpha_i \nu_i \nabla y, \nabla v)_{L^2(D)^n} \\ &= \Big\langle \Big(A_0 + \sum_{i=1}^m \alpha_i A_i \Big) y, v \Big\rangle_{Y^*, Y}. \end{aligned}$$

We next consider elementary tensors $\boldsymbol{y}, \boldsymbol{v} \in \boldsymbol{Y}$, i.e., $\boldsymbol{y}(x,\alpha) = y(x) \cdot \prod_{j=1}^{m} \upsilon_j(\alpha_j)$ (or $\boldsymbol{y} = y \otimes \upsilon_1 \otimes \ldots \otimes \upsilon_m$), $\boldsymbol{v}(x,\alpha) = v(x) \cdot \prod_{j=1}^{m} \xi_j(\alpha_j)$ (or $\boldsymbol{v} = v \otimes \xi_1 \otimes \ldots \otimes \xi_m$), and compute with the identification $L^2(\Gamma_j, \mathbb{P}_{\alpha_j})^* = L^2(\Gamma_j, \mathbb{P}_{\alpha_j})$:

$$\begin{aligned}
\langle \boldsymbol{A}\boldsymbol{y}, \boldsymbol{v}\rangle_{\boldsymbol{Y}^*,\boldsymbol{Y}} &= \int_\Gamma \Big\langle \Big(A_0 + \sum_{i=1}^{m} \alpha_i A_i\Big) y, v \Big\rangle_{Y^*,Y} \Big(\prod_{j=1}^{m} \upsilon_j(\alpha_j)\xi_j(\alpha_j)\Big)\, d\mathbb{P}_\alpha \\
&= \langle A_0 y, v\rangle_{Y^*,Y} \cdot \prod_{j=1}^{m} \int_{\Gamma_j} \upsilon_j(\alpha_j)\xi_j(\alpha_j)\, d\mathbb{P}_{\alpha_j} \\
&\quad + \sum_{i=1}^{m} \langle A_i y, v\rangle_{Y^*,Y} \cdot \int_{\Gamma_i} \alpha_i \upsilon_i(\alpha_i)\xi_i(\alpha_i)\, d\mathbb{P}_{\alpha_i} \cdot \prod_{j\neq i} \int_{\Gamma_j} \upsilon_j(\alpha_j)\xi_j(\alpha_j)\, d\mathbb{P}_{\alpha_j} \\
&= \Big\langle \Big(A_0 \otimes \Big(\bigotimes_{j=1}^{m} \mathrm{Id}\Big) + \sum_{i=1}^{m} A_i \otimes \Big(\bigotimes_{j=1}^{m} S_{ij}\Big)\Big)\boldsymbol{y}, \boldsymbol{v}\Big\rangle_{\boldsymbol{Y}^*,\boldsymbol{Y}}.
\end{aligned} \tag{7}$$

The operator $\Big(A_0 \otimes \Big(\bigotimes_{j=1}^{m} \mathrm{Id}\Big) + \sum_{i=1}^{m} A_i \otimes \Big(\bigotimes_{j=1}^{m} S_{ij}\Big)\Big)$ can be written as $\boldsymbol{y} \mapsto A_0 \circ_1 \boldsymbol{y} + \sum_{i=1}^{m} A_i \circ_1 S_{ii} \circ_{i+1} \boldsymbol{y}$. It is bounded since the operators A_i and S_{ij} are bounded. This follows from $\kappa_0, \nu_i \in L^\infty(D)$ and the boundedness of the sets Γ_i. Since $\boldsymbol{A}$ is linear and the space spanned by all elementary tensors is dense in $\boldsymbol{Y}$, we can conclude that (6) holds. □

This structure of the operator $\boldsymbol{A}$ makes it easily applicable to tensors in low-rank formats that allow applications of a linear operator to a mode and summation of tensors.

The next example shows that also more complicated operators that incorporate additional domain parametrizations can be handled:

Example 3.3. Consider a domain D containing $\frac{m}{2}$ open subdomains $D_i(\alpha_i) \subset D$, $i \in \big\{1, \ldots, \frac{m}{2}\big\}$, with m even, the sizes and shapes of which depend on the parameters α_i, respectively. One can think of circular disks with fixed midpoint and varying radius. The subsets shall be disjoint for all values $\alpha_i \in \Gamma_i$. The average coefficient on every $D_i(\alpha_i)$ is constant and given by σ_i, where $\sigma \in \mathbb{R}_{>0}^{m/2}$, while the coefficient on the rest $\tilde{D}(\alpha) := D \setminus \big(\bigcup_{i=1}^{m/2} D_i(\alpha_i)\big)$ of the domain is $\sigma_0 > 0$. Besides the parameter-dependent subdomain sizes, the coefficient on the subdomains depend again on parameters, now denoted by α_i, $i \in \{\frac{m}{2}+1, \ldots, m\}$. Again, we assume them to be independent and uniformly distributed on $(-1,1)$. The coefficients on the subdomains $D_i(\alpha_i)$ are $(1+\alpha_{i+m/2}\vartheta_i)\sigma_i$, where $\vartheta \in [0,1)^{m/2}$ describes the allowed relative variation. This yields the parametric coefficient function

$$\begin{aligned}
\kappa_\alpha(x) &= \left\{ \begin{array}{ll} \sigma_0 & \text{if } x \in \tilde{D}(\alpha) \\ \big(1 + \alpha_{i+m/2}\vartheta_i\big)\sigma_i & \text{if } x \in D_i(\alpha_i) \text{ for a } i \in \big[\tfrac{m}{2}\big] \end{array} \right\} \\
&= \sigma_0 + \sum_{i=1}^{m/2} \big(\sigma_i - \sigma_0 + \alpha_{i+m/2}\vartheta_i\sigma_i\big) \cdot 1_{D_i(\alpha_i)}(x).
\end{aligned}$$

Defining $\kappa_0(x) := \sigma_0$, $\nu_{i,\alpha_i}(x) := 1_{D_i(\alpha_i)}(x)$, and $\tau_i\big(\alpha_{i+m/2}\big) := \sigma_i - \sigma_0 + \vartheta_i\sigma_i\alpha_{i+m/2}$, we get

$$\kappa_\alpha(x) = \kappa_0(x) + \sum_{i=1}^{m/2} \tau_i(\alpha_{i+m/2})\nu_{i,\alpha_i}(x)$$

and thus a quite similar, but a bit more complicated structure as in Example 3.1:

LEMMA 3.4. *Having* $\kappa_\alpha(x) \coloneqq \kappa_0(x) + \sum_{i=1}^{m/2} \tau_i(\alpha_{i+m/2})\nu_{i,\alpha_i}(x)$ *as in Example 3.3, the operator* $\boldsymbol{A}$ *given in* (5) *can be written as*

$$\begin{aligned}&(\boldsymbol{A}\boldsymbol{y})(\cdot, \alpha_1, \dots, \alpha_{m/2}, \cdot, \dots, \cdot)\\ &= \Big(A_0 \circ_1 \boldsymbol{y} + \sum_{i=1}^{m/2} A_i(\alpha_i) \circ_1 S_{ii} \circ_{i+m/2+1} \boldsymbol{y}\Big)(\cdot, \alpha_1, \dots, \alpha_{m/2}, \cdot, \dots, \cdot)\end{aligned}$$

with $A_0, A_i(\alpha_i) \in \mathcal{L}(Y, Y^*)$ *and* $S_{ii} \in \mathcal{L}(L^2(\Gamma_{i+m/2}, \mathbb{P}_{\alpha_{i+m/2}}), L^2(\Gamma_{i+m/2}, \mathbb{P}_{\alpha_{i+m/2}}))$ *defined by*

$$\langle A_0 y, v\rangle_{Y^*,Y} \coloneqq (\kappa_0 \nabla y, \nabla v)_{L^2(D)^n}, \quad \langle A_i(\alpha_i) y, v\rangle_{Y^*,Y} \coloneqq (\nu_{i,\alpha_i} \nabla y, \nabla v)_{L^2(D)^n},$$

$$(S_{ii} v_{i+m/2})(\alpha_{i+m/2}) = \tau_i(\alpha_{i+m/2}) \cdot v_{i+m/2}(\alpha_{i+m/2}).$$

Proof. In a similar manner as in the proof of Lemma 3.2 we get $A(\alpha) = A_0 + \sum_{i=1}^{m/2} \tau_i(\alpha_{i+m/2})A_i(\alpha_i)$. Exactly the same way as in equation (7) we can apply it to an elementary tensor and end up with the result from the lemma. Again, it holds for arbitrary tensors by linearity and continuity. Note that the maps τ_i belong to $L^\infty(\Gamma_i, \mathbb{P}_{\alpha_i})$. ∎

The operator in the previous lemma is a sum of operators that cannot be written by i-mode operator multiplications. Rather the modes of α_i and x are coupled through the operation $\boldsymbol{v}(x, \alpha_i) \mapsto (A_i(\alpha_i)\boldsymbol{v}(\cdot, \alpha_i))(x)$, which maps a tensor of order 2 to a tensor of order 2. We propose a possibility for the application to low-rank tensors in the discrete case in the next section (Example 3.6).

3.1. Optimal control under uncertainty. We now consider an optimal control problem where the parametric PDE (4) constitutes the state equation and the right hand side b depends on the control $u \in L^2(D)$. We can write (4) as

$$E_\alpha(y_\alpha, u) \coloneqq A(\alpha)y_\alpha - b(u) = 0 \tag{8}$$

with $E_\alpha : Y \times U \to Y^*$. For concreteness, let $\langle b(u), v\rangle_{Y^*,Y} \coloneqq \int_D u(x)v(x)\,dx$ and consider a cost function of tracking type:

$$J(y_\alpha, u) \coloneqq \tfrac{1}{2}\|y_\alpha - \hat{y}\|^2_{L^2(D)} + \tfrac{\gamma}{2}\|u\|^2_{L^2(D)}, \tag{9}$$

with $\gamma \in \mathbb{R}_{>0}$, and an admissible control set $U_{\mathrm{ad}} \coloneqq \{u \in U : u_{\mathrm{l}} \le u \le u_{\mathrm{u}} \text{ a.e. in } D\}$, where $u_{\mathrm{l}} \le u_{\mathrm{u}}$, $u_{\mathrm{l}}, u_{\mathrm{u}} \in U$. The control u and the admissible set U_{ad} do not depend on the parameter α.

If we fix a particular realization of α (e.g., the expected value), we obtain a deterministic, parametrized optimal control problem:

$$\min_{(y_\alpha,u)\in Y\times U} J(y_\alpha, u) \quad \text{s. t.} \quad E_\alpha(y_\alpha, u) = 0, \quad u \in U_{\mathrm{ad}}. \tag{10}$$

Solving this problem for a certain parameter selection α yields a control that is optimal for exactly this realization. But, in fact, we do not know an exact value for α but only its distribution. Often, rather than considering fixed realizations of α it is much more desirable to obtain robustified controls that take into account all possible realizations

of the parameter α. Hence, we need to consider the tensor $\boldsymbol{y}$ of solutions for all parameters as state and the state equation becomes (cf. equation (5))

$$\boldsymbol{E}(\boldsymbol{y}, u) := \boldsymbol{A}\boldsymbol{y} - \boldsymbol{B}u = 0 \tag{11}$$

with $\boldsymbol{E} : \boldsymbol{Y} \times U \to \boldsymbol{Y}^*$ and $\langle \boldsymbol{B}u, \boldsymbol{v}\rangle_{\boldsymbol{Y}^*,\boldsymbol{Y}} := \int_\Gamma (u, \boldsymbol{v}(\cdot,\alpha))_{L^2(D)}\, d\mathbb{P}_\alpha$. Now we have to incorporate a suitable risk measure into the objective function to treat the stochastic dependencies.[3] We choose to minimize the mean squared deviation between the actual and the desired state (cf. [28]) and take the objective function

$$\boldsymbol{J}(\boldsymbol{y}, u) := \tfrac{1}{2}\mathbb{E}_\alpha\left(\|\boldsymbol{y} - \hat{y}\otimes\mathbb{1}\|^2_{L^2(D)}\right) + \tfrac{\gamma}{2}\|u\|^2_{L^2(D)}. \tag{12}$$

Note that there holds $\mathbb{E}_\alpha(\|\boldsymbol{y} - \hat{y}\otimes\mathbb{1}\|^2_{L^2(D)}) = \|\boldsymbol{y} - \hat{y}\otimes\mathbb{1}\|^2_{L^2(D)\otimes L^2(\Gamma,\mathbb{P}_\alpha)}$. The optimal control problem under uncertainty is then given by:

$$\min_{(\boldsymbol{y},u)\in\boldsymbol{Y}\times U} \boldsymbol{J}(\boldsymbol{y}, u) \quad \text{s.t.} \quad \boldsymbol{E}(\boldsymbol{y}, u) = 0, \quad u \in U_{\text{ad}}. \tag{13}$$

The existence of a unique solution to this problem can be shown exactly as for the deterministic one (10) since the operator $\boldsymbol{A}$ has a bounded inverse by the Lax-Milgram lemma (cf. [22, Theorem 1.43]).

3.1.1. Space discretization: finite elements. In order to solve the optimal control problems (10) and (13) numerically, we have to discretize them. In the case of problem (10) we insert a fixed realization α and use a standard finite element spatial discretization. The domain D is approximated by a domain D^h that is partitioned into a finite element mesh. We choose continuous, piecewise polynomial ansatz spaces (piecewise linear in our numerical experiments, but this is not essential). Denote by $\{\psi_\ell\}_{\ell=1}^{\tilde{N}}$ the finite element basis for the discrete control space $U^h \subset L^2(D^h)$ and by $\{\phi_{k_0}\}_{k_0=1}^{N_0}$ the basis of the discrete state space $Y^h \subset H^1_0(D^h)$. For a given realization of α, the sparse system matrix $\mathrm{A}^h(\alpha) \in \mathbb{R}^{N_0\times N_0}$ is given by:

$$(\mathrm{A}^h(\alpha))_{k_0 l_0} := (\kappa_\alpha \nabla\phi_{l_0}, \nabla\phi_{k_0})_{L^2(D^h)^n}.$$

Example 3.5. Using the particular form of the operators given in Lemmas 3.2 and 3.4, we can derive additional structure of the system matrix $\mathrm{A}^h(\alpha)$. For Example 3.1 we get

$$\begin{aligned}\left(\mathrm{A}^h(\alpha)\right)_{k_0 l_0} &= (\kappa_0\nabla\phi_{l_0}, \nabla\phi_{k_0})_{L^2(D^h)^n} + \sum_{i=1}^m \alpha_i (\kappa_0\nu_i\nabla\phi_{l_0}, \nabla\phi_{k_0})_{L^2(D^h)^n} \\ &=: (\mathrm{A}_0^h)_{k_0 l_0} + \sum_{i=1}^m \alpha_i (\mathrm{A}_i^h)_{k_0 l_0}.\end{aligned} \tag{14}$$

The matrices $\mathrm{A}_0^h, \mathrm{A}_1^h, \ldots, \mathrm{A}_m^h \in \mathbb{R}^{N_0\times N_0}$ are again sparse, and if the functions ν_i are indicators for subsets of D as in our example, one can restrict the assembly of A_i^h to those elements where ν_i is not identically zero. In a similar way the operator from Example 3.3 can be written as

$$\mathrm{A}^h(\alpha) = \mathrm{A}_0^h + \sum_{i=1}^{m/2} \tau_i\left(\alpha_{i+m/2}\right)\mathrm{A}_i^h\left(\alpha_i\right)$$

[3] Another approach is to make an optimal selection for the worst case.

with slightly different definitions of the occurring matrices.

Defining the matrix $\mathrm{B}^h \in \mathbb{R}^{N_0 \times \tilde{N}}$ as

$$(\mathrm{B}^h)_{k_0 l} \coloneqq (\psi_l, \phi_{k_0})_{L^2(D^h)}$$

we can state a discrete version of equation (8) as

$$\mathrm{E}^h_\alpha(\mathrm{y}^h, \mathrm{u}^h) \coloneqq \mathrm{A}^h(\alpha)\mathrm{y}^h - \mathrm{B}^h\mathrm{u}^h = 0, \tag{15}$$

where $\mathrm{y}^h \in \mathbb{R}^{N_0}$ contains the coefficients of the discrete state w.r.t. the basis $\{\phi_{k_0}\}_{k_0=1}^{N_0}$ and $\mathrm{u}^h \in \mathbb{R}^{\tilde{N}}$ are the coefficients for the discrete control.

The objective function (9) in the discrete space reads

$$\mathrm{J}^h(\mathrm{y}^h, \mathrm{u}^h) \coloneqq \tfrac{1}{2}(\mathrm{y}^h - \hat{\mathrm{y}}^h)^T \mathrm{M}^h(\mathrm{y}^h - \hat{\mathrm{y}}^h) + \tfrac{\gamma}{2}{\mathrm{u}^h}^T \tilde{\mathrm{M}}^h \mathrm{u}^h,$$

where $\hat{\mathrm{y}}^h \in \mathbb{R}^{N_0}$ is the discrete desired state, $\mathrm{M}^h \in \mathbb{R}^{N_0 \times N_0}$ is the mass matrix w.r.t. the basis $\{\phi_{k_0}\}_{k_0=1}^{N_0}$ i.e., $\left(\mathrm{M}^h\right)_{kl} \coloneqq (\phi_l, \phi_k)_{L^2(D^h)}$. Further, $\tilde{\mathrm{M}}^h \in \mathbb{R}^{\tilde{N} \times \tilde{N}}$ is the limped (and thus diagonal) mass matrix w.r.t. the basis $\{\psi_\ell\}_{\ell=1}^{\tilde{N}}$.

If an appropriate basis is used, the control constraints can be discretized as $\mathrm{u}^h_{\mathrm{l}} \le \mathrm{u}^h \le \mathrm{u}^h_{\mathrm{u}}$ (element-wise) with discrete versions $\mathrm{u}^h_{\mathrm{l}}, \mathrm{u}^h_{\mathrm{u}} \in \mathbb{R}^{\tilde{N}}$ of the pointwise bounds. We obtain the following discretized version of problem (10)

$$\min_{\mathrm{y}^h \in \mathbb{R}^{N_0}, \mathrm{u}^h \in \mathbb{R}^{\tilde{N}}} \mathrm{J}^h(\mathrm{y}^h, \mathrm{u}^h) \quad \text{s.t.} \quad \mathrm{E}^h_\alpha(\mathrm{y}^h, \mathrm{u}^h) = 0, \quad \mathrm{u}^h_{\mathrm{l}} \le \mathrm{u}^h \le \mathrm{u}^h_{\mathrm{u}}. \tag{16}$$

3.1.2. Polynomial chaos. We proceed with the discretization of (13). Here we have additionally the dependence on the random variables α. The expectation w.r.t. these variables in the objective function will be approximated by a full, multi-dimensional quadrature formula. This formula is derived from the respective one-dimensional quadrature rules for each variable α_i. Let $N_i \in \mathbb{N}$ be the number of different quadrature nodes $\{a^i_{k_i}\}_{k_i=1}^{N_i} \subset \Gamma_i$ used for the parameter α_i. In order to obtain high exactness we use Gaussian quadrature. The nodes are then the roots of the polynomial with degree N_i from a set of orthogonal polynomials w.r.t. $\mathbb{P}_{\alpha_i}$ with increasing degree. We consider Lagrange polynomials $\{\theta^i_{k_i}\}_{k_i=1}^{N_i}$ of degree $N_i - 1$ w.r.t. the nodes, implicitly defined by $\theta^i_{k_i}(a^i_{l_i}) = \delta_{k_i l_i}$. The one-dimensional quadrature weights are then given by $w^i_{k_i} \coloneqq \int_{\Gamma_i} \theta^i_{k_i}(\alpha_i)\, d\mathbb{P}_{\alpha_i}$ and a one-dimensional integral of a continuous, integrable function $z_i : \Gamma_i \to \mathbb{R}$ can be approximated by $\int_{\Gamma_i} z_i(\alpha_i)\, d\mathbb{P}_{\alpha_i} \approx \sum_{k_i=1}^{N_i} w^i_{k_i} z_i\left(a^i_{k_i}\right)$.
To discretize the space $L^2(\Gamma_i, \mathbb{P}_{\alpha_i})$ we use polynomials of degree $N_i - 1$ and the weighted Lagrange polynomials $\{\beta^i_{k_i}\}_{k_i=1}^{N_i}$, $\beta^i_{k_i}(\alpha_i) = \omega^i_{k_i}\theta^i_{k_i}(\alpha_i)$ with a vector $\omega^i \in \mathbb{R}^{N_i}_{>0}$ of positive weights, as basis. Note that these polynomials are always orthogonal which follows from the fact that Gaussian quadrature is exact for polynomials up to degree $2N_i - 1$ in this case. If we choose $\omega^i_{k_i} = (w^i_{k_i})^{-1/2}$, they are even orthonormal (cf. [13]). We construct a subspace basis of $L^2(\Gamma, \mathbb{P}_\alpha) = \bigotimes_{i=1}^m L^2(\Gamma_i, \mathbb{P}_{\alpha_i})$ by forming the full tensor product $\left\{\bigotimes_{i=1}^m \beta^i_{k_i}, k_i \in [N_i], i \in [m]\right\} = \left\{\boldsymbol{\beta} : \boldsymbol{\beta}(\alpha) = \prod_{i=1}^m \beta^i_{k_i}(\alpha_i), k_i \in [N_i], i \in [m]\right\}$ of the one-dimensional bases. Every element $\boldsymbol{z}^h$ of its span is represented by a tensor of coefficients $\mathbf{z}^h \in \mathbb{R}^{N_1 \times \ldots \times N_m}$:

$$\boldsymbol{z}^h(\alpha_1, \ldots, \alpha_m) = \sum_{k_1=1}^{N_1} \ldots \sum_{k_m=1}^{N_m} \mathbf{z}^h(k_1, \ldots, k_m)\beta^1_{k_1}(\alpha_1) \ldots \beta^m_{k_m}(\alpha_m). \tag{17}$$

These coefficients are then given by suitably weighted values of the multivariate polynomial at the points of the multi-dimensional grid $\bigtimes_{i=1}^m \{a_{k_i}^i\}_{k_i=1}^{N_i}$: $\mathbf{z}^h(k_1,\dots,k_m) = \boldsymbol{z}^h(a_{k_1}^1,\dots,a_{k_m}^m)\cdot\prod_{i=1}^m {\omega_{k_i}^i}^{-1}$. The integral of a multivariate function $\boldsymbol{z}:\Gamma\to\mathbb{R}$ can then be approximated by

$$\mathbb{E}_\alpha(\boldsymbol{z}) = \int_\Gamma \boldsymbol{z}(\alpha)\,d\mathbb{P}_\alpha \approx \sum_{k_1=1}^{N_1}\dots\sum_{k_m=1}^{N_m} w_{k_1}^1\cdots w_{k_m}^m\,\boldsymbol{z}(a_{k_1}^1,\dots,a_{k_m}^m) =: \langle \mathbf{z}^h, \mathbf{w}^h\odot\boldsymbol{\omega}^h\rangle$$

with the elementary weight tensors $\mathbf{w}^h := \bigotimes_{i=1}^m w^i \in \mathbb{R}^{N_1\times\dots\times N_m}$ and $\boldsymbol{\omega}^h := \bigotimes_{i=1}^m \omega^i \in \mathbb{R}^{N_1\times\dots\times N_m}$. Applying this quadrature rule to an integrand thus can be written as the inner product of an elementary weight tensor $\mathbf{w}^h\odot\boldsymbol{\omega}^h$ and the tensor $\mathbf{z}^h$ of coefficients that represents the discrete interpolant of $\boldsymbol{z}$ on the grid. Functions $\boldsymbol{z}=\boldsymbol{z}^h$ of the form (17) are integrated exactly with the above quadrature rule. The $L^2(\Gamma,\mathbb{P})$-inner product of two functions $\boldsymbol{z}^h$, $\tilde{\boldsymbol{z}}^h$ represented by the coefficients $\mathbf{z}^h$, $\tilde{\mathbf{z}}^h$ is $\int_\Gamma \boldsymbol{z}^h\tilde{\boldsymbol{z}}^h\,d\mathbb{P}_\alpha = \langle \boldsymbol{z}^h\odot\mathbf{w}^h\odot\boldsymbol{\omega}^h.^2, \tilde{\boldsymbol{z}}^h\rangle =: \langle\tilde{\mathbf{z}}^h,\tilde{\mathbf{z}}^h\rangle_{\mathbf{w}^h\odot\boldsymbol{\omega}^h.^2}$ due to the exactness of Gaussian quadrature and orthogonality of the polynomials. The operator $\mathbf{z}^h\mapsto(\mathbf{w}^h\odot\boldsymbol{\omega}^h.^2)\odot\mathbf{z}^h$ induces the inner product.

For the discrete state space $\boldsymbol{Y}^h\subset H_0^1(D^h)\otimes L^2(\Gamma_1,\mathbb{P}_{\alpha_1})\otimes\dots\otimes L^2(\Gamma_m,\mathbb{P}_{\alpha_m})$ we choose the tensor product $\{\phi_{k_0}\otimes\beta_{k_1}^1\otimes\dots\otimes\beta_{k_m}^m, k_i\in[N_i], i\in\{0,\dots,m\}\}$ of the FE basis and the basis of multivariate polynomials as a basis[4]. The coefficients are then written as a tensor $\mathbf{y}^h\in\mathbb{R}^{N_0\times\dots\times N_m}$; if a nodal FE basis is used, they are weighted function values of the represented function $\boldsymbol{y}^h\in\boldsymbol{Y}^h$. As in (17), the value of a multivariate function $\boldsymbol{y}^h$ represented by the coefficients $\mathbf{y}^h$ is given by:

$$\boldsymbol{y}^h(x,\alpha) = \sum_{k_0=1}^{N_0}\sum_{k_1=1}^{N_1}\dots\sum_{k_m=1}^{N_m}\mathbf{y}^h(k_0,\dots,k_m)\phi_{k_0}(x)\beta_{k_1}^1(\alpha_1)\dots\beta_{k_m}^m(\alpha_m). \tag{18}$$

Now we can formulate a discrete analogue to equation (11) by inserting state functions $\boldsymbol{y}^h$ and test functions $\boldsymbol{v}^h$ represented by the discrete tensors $\mathbf{y}^h,\mathbf{v}^h\in\mathbb{R}^{N_0\times\dots\times N_m}$ as in (18), respectively. This is a stochastic Galerkin ansatz. We obtain with $u^h=\sum_{\ell=1}^{\tilde N}\mathrm{u}_\ell^h\psi_\ell$

$$\begin{aligned}
&\int_\Gamma (u^h,\boldsymbol{v}^h(\cdot,\alpha))_{L^2(D^h)}\,d\mathbb{P}_\alpha\\
&= \sum_{k_0=1}^{N_0}\sum_{k_1=1}^{N_1}\dots\sum_{k_m=1}^{N_m}\mathbf{v}^h(k_0,\dots,k_m)(u^h,\phi_{k_0})_{L^2(D^h)}\int_\Gamma\beta_{k_1}^1(\alpha_1)\dots\beta_{k_m}^m(\alpha_m)\,d\mathbb{P}_\alpha\\
&= \langle(\mathrm{B}^h\mathrm{u}^h)\otimes(\mathbf{w}^h\odot\boldsymbol{\omega}^h),\mathbf{v}^h\rangle =: \langle\mathbf{B}^h\mathrm{u}^h,\mathbf{v}^h\rangle_{\mathbb{1}\otimes(\mathbf{w}^h\odot\boldsymbol{\omega}^h.^2)}
\end{aligned}$$

with $\mathbf{B}^h\mathrm{u}^h := (\mathrm{B}^h\mathrm{u}^h)\otimes\boldsymbol{\omega}^h.^{-1}$ and

$$\begin{aligned}
&\int_\Gamma(\kappa_\alpha\nabla_x\boldsymbol{y}^h(\cdot,\alpha),\nabla_x\boldsymbol{v}^h(\cdot,\alpha))_{L^2(D^h)^n}\,d\mathbb{P}_\alpha\\
&= \sum_{k_0,\dots,k_m=1}^{N_0,\dots,N_m}\sum_{l_0,\dots,l_m=1}^{N_0,\dots,N_m}\mathbf{y}^h(l_0,\dots,l_m)\mathbf{v}^h(k_0,\dots,k_m)\int_\Gamma(\mathrm{A}^h(\alpha))_{k_0l_0}\prod_{i=1}^m\beta_{l_i}^i(\alpha_i)\beta_{k_i}^i(\alpha_i)\,d\mathbb{P}_\alpha\\
&= \langle\langle\mathbf{a}^h,\mathbf{y}^h\rangle_{\{m+2,\dots,2m+2\},[m+1]},\mathbf{v}^h\rangle_{\mathbb{1}\otimes(\mathbf{w}^h\odot\boldsymbol{\omega}^h.^2)} =: \langle\mathbf{A}^h\mathbf{y}^h,\mathbf{v}^h\rangle_{\mathbb{1}\otimes(\mathbf{w}^h\odot\boldsymbol{\omega}^h.^2)}
\end{aligned}$$

with the tensor $\mathbf{a}^h\in\mathbb{R}^{N_0\times N_1\times\dots\times N_m\times N_0\times N_1\times\dots\times N_m}$ defined by

$$\mathbf{a}^h(k,l) := \frac{1}{\prod_{i=1}^m w_{k_i}^i(\omega_{k_i}^i)^2}\left(\int_\Gamma(\mathrm{A}^h(\alpha))_{k_0l_0}\prod_{i=1}^m\beta_{l_i}^i(\alpha_i)\beta_{k_i}^i(\alpha_i)\,d\mathbb{P}_\alpha\right). \tag{19}$$

[4] This is a basis of the tensor product of subspaces by [17, Lemma 3.11].

Here and in the following, we abbreviate $k = (k_0, \dots, k_m)$, $l = (l_0, \dots, l_m)$, $(k,l) = (k_0, \dots, k_m, l_0, \dots, l_m)$, etc.

The discrete counterpart to the state equation (11) is

$$\mathbf{E}^h(\mathbf{y}^h, \mathrm{u}^h) := \mathbf{A}^h \mathbf{y}^h - \mathbf{B}^h \mathrm{u}^h = 0.$$

For tractable computations involving the operator $\mathbf{A}^h$ or the tensor $\mathbf{a}^h$, respectively, it is essential to work with low-rank representations of the involved tensors $\mathbf{y}^h$, etc., and that $\mathbf{A}^h$ can be efficiently applied to such low-rank tensors. The latter holds, e.g., if $\mathbf{a}^h$ can be represented in a low-rank format as well or if $\mathbf{A}^h$ is given as a sum of Kronecker products of matrices, as it is the case for the operators $\mathbf{A}^h$ from our previous examples.

Example 3.6. Motivated by equation (7) we use the additional structure in the operator $\boldsymbol{A}$ from Example 3.1 to state its discrete version $\mathbf{A}^h$. Recall that by Lemma 3.2 we had $\boldsymbol{A}\boldsymbol{y} = A_0 \circ_1 \boldsymbol{y} + \sum_{i=1}^m A_i \circ_1 S_{ii} \circ_{i+1} \boldsymbol{y}$ with $S_{ii} : y_i(\alpha_i) \mapsto \alpha_i \cdot y_i(\alpha_i)$. For brevity, we write $\beta^i_{l_i}\beta^i_{k_i}$ instead of $\beta^i_{l_i}(\alpha_i)\beta^i_{k_i}(\alpha_i)$. We use (14) and the exactness of Gaussian quadrature to obtain:

$$\begin{aligned}
&\int_\Gamma (\mathrm{A}^h(\alpha))_{k_0 l_0} \prod_{j=1}^m \beta^j_{l_j}(\alpha_j)\beta^j_{k_j}(\alpha_j)\, d\mathbb{P}_\alpha \\
&= (\mathrm{A}^h_0)_{k_0 l_0} \prod_{j=1}^m \int_{\Gamma_j} \beta^j_{l_j}\beta^j_{k_j}\, d\mathbb{P}_{\alpha_j} + \sum_{i=1}^m (\mathrm{A}^h_i)_{k_0 l_0} \int_{\Gamma_i} \alpha_i \beta^i_{l_i}\beta^i_{k_i}\, d\mathbb{P}_{\alpha_i} \prod_{j\neq i} \int_{\Gamma_j} \beta^j_{l_j}\beta^j_{k_j}\, d\mathbb{P}_{\alpha_j} \\
&= (\mathrm{A}^h_0)_{k_0 l_0} \prod_{j=1}^m w^j_{l_j}(\omega^j_{l_j})^2 \delta_{l_j k_j} + \sum_{i=1}^m (\mathrm{A}^h_i)_{k_0 l_0} a^i_{l_i} w^i_{l_i}(\omega^i_{l_i})^2 \delta_{l_i k_i} \prod_{j\neq i} w^j_{l_j}(\omega^j_{l_j})^2 \delta_{l_j k_j} \\
&= (\prod_{j=1}^m w^j_{l_j}(\omega^j_{l_j})^2)((\mathrm{A}^h_0)_{k_0 l_0} + \sum_{i=1}^m (\mathrm{A}^h_i)_{k_0 l_0} a^i_{l_i})(\prod_{j=1}^m \delta_{l_j k_j}).
\end{aligned} \tag{20}$$

The first factor cancels in the definition (19) of $\mathbf{a}^h$. A short calculation yields:

$$\mathbf{A}^h \mathbf{y}^h = \langle \mathbf{a}^h, \mathbf{y}^h \rangle_{\{m+2,\dots,2m+2\},[m+1]} = \mathrm{A}^h_0 \circ_1 \mathbf{y}^h + \sum_{i=1}^m \mathrm{A}^h_i \circ_1 \mathrm{S}^h_i \circ_{i+1} \mathbf{y}^h$$

with $\mathrm{S}^h_i = \mathrm{diag}(a^i)$, where $a^i \in \mathbb{R}^{N_i}$ is the vector of grid points $a^i_1, \dots, a^i_{N_i}$. The product of Kronecker deltas yields that we get in fact a completely decoupled system, one $N_0 \times N_0$ system of linear equations for each combination $(a^1_{k_1}, \dots, a^m_{k_m})$ of parameter realizations:

$$\begin{aligned}
(\mathbf{A}^h \mathbf{y}^h)(\bullet, k_1, \dots, k_m) &= (\mathrm{A}^h_0 + \sum_{i=1}^m a^i_{k_i} \mathrm{A}^h_i)\mathbf{y}^h(\bullet, k_1, \dots, k_m) \\
&= \mathrm{A}^h(a^i_{k_i})\mathbf{y}^h(\bullet, k_1, \dots, k_m) = (\prod_{i=1}^m (\omega^i_{k_i})^{-1})(\mathrm{B}^h \mathrm{u}^h).
\end{aligned}$$

This relates the approach to stochastic collocation (cf. [13]), where one would have $\omega^i = \mathbb{1}$ for all i and then get the same weight 1 for all equations obtained by inserting the collocation points into equation (15). Taking orthonormal polynomials as basis, these equations are weighted differently.

For the operator of Example 3.3 we additionally have to think about how to apply the parametrized matrices $\mathrm{A}^h_i(\alpha_i)$. Performing a computation similar to (20), the integral

$$\int_{\Gamma_i} (\mathrm{A}^h_i(\alpha_i))_{k_0 l_0} \beta^i_{l_i}(\alpha_i)\beta^i_{k_i}(\alpha_i)\, d\mathbb{P}_{\alpha_i} \approx (\mathrm{A}^h_i(a^i_{l_i}))_{k_0 l_0} w^i_{l_i}(\omega^i_{l_i})^2 \delta_{l_i k_i},$$

appears and is approximated by Gaussian quadrature. Therefore, we have to assemble the $\sum_{i=1}^{m/2} N_i$ matrices $\mathrm{A}_i^h\left(a_{k_i}^i\right)$ for all $i \in [m/2]$ and $k_i \in [N_i]$. Since the functions τ_i from example 3.3 are affine, the remaining integrals can again be computed exactly with Gaussian quadrature. With $k = (k_0, \dots, k_m)$ and $l = (l_0, \dots, l_m)$ we obtain:

$$
\begin{aligned}
\mathbf{a}^h(k,l) &= ((\mathrm{A}_0^h)_{k_0 l_0} + \sum_{i=1}^{m/2} (\mathrm{A}_i^h(a_{l_i}^i))_{k_0 l_0} \tau_i(a_{l_{i+m/2}}^{i+m/2})) \prod_{j=1}^{m} \delta_{l_j k_j}, \\
(\mathbf{A}^h \mathbf{y}^h)(k) &= \sum_{l_0,\dots,l_m=1}^{N_0,\dots,N_m} ((\mathrm{A}_0^h)_{k_0 l_0} + \sum_{i=1}^{m/2} (\mathrm{A}_i^h(a_{l_i}^i))_{k_0 l_0} \tau_i(a_{l_{i+m/2}}^{i+m/2}))(\prod_{j=1}^{m} \delta_{l_j k_j}) \mathbf{y}^h(l) \\
&= \sum_{l_0=1}^{N_0} (\mathrm{A}_0^h)_{k_0 l_0} \mathbf{y}^h(l_0, k) + \sum_{l_0=1}^{N_0} \sum_{i=1}^{m/2} (\mathrm{A}_i^h(a_{k_i}^i))_{k_0 l_0} \tau_i(a_{k_{i+m/2}}^{i+m/2}) \mathbf{y}^h(l_0, k) \\
&\quad \times (\mathrm{A}_0^h \circ_1 \mathbf{y}^h + \sum_{i=1}^{m/2} \sum_{k_i=1}^{N_i} \mathrm{A}_i^h\left(a_{k_i}^i\right) \circ_1 \operatorname{diag}\left(e_{k_i}^i\right) \circ_{i+1} \operatorname{diag}\left(\tau_i(a^{i+m/2})\right) \circ_{i+1+m/2} \mathbf{y}^h)(k).
\end{aligned}
$$

Here, $e_{k_i}^i \in \mathbb{R}^{N_i}$ is the k_i-th unit vector. As all operations (i-mode matrix multiplication and summing) are performable in low-rank tensor formats, we can also efficiently treat the operator with domain parametrization by low-rank tensor techniques. Each i-mode matrix product can be performed with a cost of $\mathcal{O}(N_{\max} r_{\max})$ since the applied matrices are sparse; the more costly part is the truncated sum of m tensors, which can be performed in $\mathcal{O}(dN_{\max} m^2 {r_{\max}}^2 + dm^2 {r_{\max}}^4 + dm^3 {r_{\max}}^3)$ for H-Tucker tensors, where the m^2-terms dominate typically [32, sec. 6.3]. The truncation is important to avoid infeasible rank growth, but causes on the other hand errors which make the application of $\mathbf{A}^h$ inexact. The objective function is approximated as follows: the squared L^2-difference w.r.t. the first mode of the tensor (the space dimension) is induced by the mass matrix and the expectation w.r.t. the parameters can be calculated by a weighted inner product of tensors as seen above. Note that the constant function $\mathbb{1}(\alpha) = 1$ is represented by the tensor $\boldsymbol{\omega}^{h}.^{-1}$ in the discrete space.

$$
\begin{aligned}
\mathbf{J}^h(\mathbf{y}^h, \mathrm{u}^h) &:= \tfrac{1}{2} \langle \mathbf{y}^h - \hat{\mathbf{y}}^h, \mathrm{M}^h \circ_1 (\mathbf{y}^h - \hat{\mathbf{y}}^h) \rangle_{\mathbb{1} \otimes (\mathbf{w}^h \odot \boldsymbol{\omega}^{h}.^2)} + \tfrac{\gamma}{2} {\mathrm{u}^h}^T \tilde{\mathrm{M}}^h \mathrm{u}^h = \\
&= \tfrac{1}{2} \langle \mathbf{y}^h - \hat{\mathbf{y}}^h, \mathbf{M}^h (\mathbf{y}^h - \hat{\mathbf{y}}^h) \rangle + \tfrac{\gamma}{2} {\mathrm{u}^h}^T \tilde{\mathrm{M}}^h \mathrm{u}^h.
\end{aligned}
\tag{21}
$$

where $\mathbf{M}^h \mathbf{z}^h := \left[\mathrm{M}^h, \operatorname{diag}\left(w^1 \odot (\omega^1).^2\right), \dots, \operatorname{diag}\left(w^m \odot (\omega^m).^2\right)\right] \circ_{1,\dots,m+1} \mathbf{z}^h$ induces the inner product discretizing that on $L^2(D^h) \otimes L^2(\Gamma, \mathbb{P}_\alpha)$ and $\hat{\mathbf{y}}^h := \hat{\mathrm{y}}^h \otimes \boldsymbol{\omega}^{h}.^{-1}$.

As already discussed, we have two natural choices for the selection of the weight vectors ω^i: $\omega^i = (w^i).^{-1/2}$ gives orthonormal polynomials, while $\omega^i = \mathbb{1}$ relates the approach to stochastic collocation and weights all deterministic equations equally. As we wanted to compare the solution tensor to solutions obtained from collocation, we chose the latter. This has the consequence that the coefficients in the state tensor $\mathbf{y}^h$ are all of the same order of magnitude and are equally affected by tensor truncation.

We now can state the discrete version of the optimal control problem (13):

$$
\min_{(\mathbf{y}^h, \mathrm{u}^h) \in \mathbb{R}^{N_0 \times \dots \times N_m} \times \mathbb{R}^{\tilde{N}}} \mathbf{J}^h(\mathbf{y}^h, \mathrm{u}^h) \quad \text{s. t.} \quad \mathbf{E}^h(\mathbf{y}^h, \mathrm{u}^h) = 0, \quad \mathrm{u}_{\mathrm{l}}^h \le \mathrm{u}^h \le \mathrm{u}_{\mathrm{u}}^h. \tag{22}
$$

The state $\mathbf{y}^h$ is a discrete tensor and we will use low-rank tensor techniques for solving the state equation efficiently, which will be addressed in section 4.

3.2. A parametrized obstacle problem. In the discussed optimal control problem the state was a tensor and had to fulfill the state equation but no further constraints were posed on it. Now, we consider an obstacle problem with uncertain parameters. This results in an optimization problem where the unknown is a tensor that is subject to pointwise (or, after discretization, element-wise) inequality constraints. Assuming a parameter dependence of the same form as in the PDE (2), we consider now an obstacle problem

$$\bar{y}_\alpha \in Y_{\mathrm{ad}}, \quad \langle A(\alpha)\bar{y}_\alpha - b, v - \bar{y}_\alpha\rangle_{Y^*,Y} \geq 0 \quad \forall\ v \in Y_{\mathrm{ad}}. \tag{23}$$

with state space $Y := H_0^1(D)$, differential operator $A(\alpha)$ as defined in (3), force density $b \in Y^*$, obstacle function[5] $g \in H^2(D)$ with $g \leq 0$ on ∂D (cf. [26]), and feasible set $Y_{\mathrm{ad}} := \{y \in H_0^1(D) : y \geq g \text{ a.e. in } D\}$.

The parametric VI (23) is a necessary and sufficient optimality condition of, and thus equivalent to, the following parametric quadratic optimization problem:

$$\min_{y\in Y} F_\alpha(y_\alpha) := \tfrac{1}{2}\langle A(\alpha)y_\alpha, y_\alpha\rangle_{Y^*,Y} - \langle b, y_\alpha\rangle_{Y^*,Y} \quad \text{s.t.} \quad y \in Y_{\mathrm{ad}}. \tag{24}$$

Since the feasible set Y_{ad} is nonempty, closed and convex and the functional F_α is uniformly convex, this problem has a unique solution $\bar{y}_\alpha \in Y_{\mathrm{ad}}$.

Typical *uncertainty quantification* (UQ) tasks for analyzing solutions of (23) require to evaluate statistical quantities that depend on the random vector α and the solution $\bar{y}_\alpha$, which is a random field. Given the distribution $\mathbb{P}_\alpha$, this requires to, e.g., compute moments, such as the expectation $\mathbb{E}_\alpha Q = \int_\Gamma Q(\alpha, \bar{y}_\alpha)\, d\mathbb{P}_\alpha$ of a quantity of interest $Q(\alpha, \bar{y}_\alpha)$. One possibility is Monte-Carlo sampling, but accurate results then require a possibly very large amount of solves to (23) for different realizations of α. As an alternative, we propose to use low-rank tensor methods for computing good approximations of the whole parametric solution $\bar{\boldsymbol{y}}(x, \alpha) = \bar{y}_\alpha(x)$ at once. In the same way as in (5), we can aggregate all instances of (24) to obtain

$$\min_{\boldsymbol{y}\in \boldsymbol{Y}} \boldsymbol{F}(\boldsymbol{y}) := \tfrac{1}{2}\langle \boldsymbol{A}\boldsymbol{y}, \boldsymbol{y}\rangle_{\boldsymbol{Y}^*,\boldsymbol{Y}} - \langle \boldsymbol{b}, \boldsymbol{y}\rangle_{\boldsymbol{Y}^*,\boldsymbol{Y}} \quad \text{s.t.} \quad \boldsymbol{y} \geq \boldsymbol{g} \text{ a.e. in } D\times\Gamma \tag{25}$$

with $\boldsymbol{g} = g \otimes \mathbb{1}$, $\mathbb{1} : \Gamma \to \mathbb{R}$, $\mathbb{1}(\alpha) = 1$, i.e., $\boldsymbol{g}(x, \alpha) = g(x)$ for all α, and $\boldsymbol{A}$ and $\boldsymbol{b}$ defined as in (5). The corresponding equivalent VI reads

$$\bar{\boldsymbol{y}} \in \boldsymbol{Y}, \quad \bar{\boldsymbol{y}} \geq \boldsymbol{g} \text{ a.e. in } D\times\Gamma, \quad \langle \boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b}, \boldsymbol{v} - \bar{\boldsymbol{y}}\rangle_{\boldsymbol{Y}^*,\boldsymbol{Y}} \geq 0 \quad \forall\ \boldsymbol{v} \in \boldsymbol{Y}, \quad \boldsymbol{v} \geq \boldsymbol{g} \text{ a.e. in } D\times\Gamma.$$

The discretization of the problem is done in the same way as for the state equation in section 3.1: The discrete deterministic state space $Y^h \subset H_0^1(D^h)$ is spanned by the FE basis $\{\phi_{k_0}\}_{k_0=1}^{N_0}$ and we write $\mathrm{y}^h \in \mathbb{R}^{N_0}$ for the coefficient vector of the state FE function $y^h \in Y^h$. The parametrized stiffness matrix $\mathrm{A}^h(\alpha)$ and mass matrix M^h are as before. The vector $\mathrm{b}^h \in \mathbb{R}^{N_0}$ discretizes the functional b in the sense that ${\mathrm{b}^h}^T \mathrm{v}^h$ is the discrete version of $\langle b, v^h\rangle_{Y^*,Y}$. Further, the obstacle g is discretized by the coefficient vector $\mathrm{g}^h \in \mathbb{R}^{N_0}$. The problem (24) then becomes

$$\min_{\mathrm{y}_\alpha^h \in \mathbb{R}^{N_0}} \mathrm{F}_\alpha^h(\mathrm{y}_\alpha^h) := \tfrac{1}{2}{\mathrm{y}_\alpha^h}^T \mathrm{A}^h(\alpha)\mathrm{y}_\alpha^h - {\mathrm{b}^h}^T \mathrm{y}^h \quad \text{s.t.} \quad \mathrm{y}_\alpha^h \geq \mathrm{g}^h. \tag{26}$$

This is a quadratic optimization problem with a uniformly convex objective function and pointwise bounds; it has a unique solution $\bar{\mathrm{y}}_\alpha^h$.

[5] Again, we also could take this obstacle to be parameter-dependent.

As before, we discretize problem (25) by a multivariate weighted Lagrangian basis in the parameters:

$$\begin{aligned} \min_{\mathbf{y}^h \in \mathbb{R}^{N_0 \times \ldots \times N_m}} \quad & \tfrac{1}{2}\langle \mathbf{A}^h \mathbf{y}^h, \mathbf{y}^h \rangle_{\mathbb{1} \otimes (\mathbf{w}^h \odot \boldsymbol{\omega}^h.^2)} - \langle \mathbf{b}^h, \mathbf{y}^h \rangle_{\mathbb{1} \otimes (\mathbf{w}^h \odot \boldsymbol{\omega}^h.^2)} \\ \text{s. t.} \quad & \mathbf{y}^h \geq \mathbf{g}^h = \mathrm{g}^h \otimes \boldsymbol{\omega}^h.^{-1} \end{aligned} \tag{27}$$

with $\mathbf{b}^h = \mathrm{b}^h \otimes \boldsymbol{\omega}^h.^{-1}$. From a collocation perspective, we can view the cost function in (27) as a sum of the cost functions $\mathrm{F}^h_\alpha(\mathrm{y}^h_\alpha)$ in (26) over all parameter grid points $\alpha = (a^1_{k_1}, \ldots, a^m_{k_m})^T$, weighted by the corresponding entries in $\mathbf{w}^h \odot \boldsymbol{\omega}^h.^2$. The constraints are all collected and weighted by the entries of $\boldsymbol{\omega}^h.^{-1}$. If the target is, e.g., to calculate the expectation $\mathbb{E}_\alpha Q$ of some quantity $Q(\alpha, y_\alpha)$, this weighting seems appropriate.

If all the solutions of (26) for all realization of α are equally important, it is more natural to use the same weight for all cost functions and for all states, respectively. This corresponds to choosing $\boldsymbol{\omega}^h = \mathbb{1}$ and replacing $\mathbf{w}^h$ in the weighted inner product by $\mathbb{1}$ (or a positive multiple of it). This yields:

$$\min_{\mathbf{y}^h \in \mathbb{R}^{N_0 \times \ldots \times N_m}} \mathbf{F}^h(\mathbf{y}^h) := \tfrac{1}{2}\left\langle \mathbf{y}^h, \mathbf{A}^h \mathbf{y}^h \right\rangle - \left\langle \mathrm{b}^h \otimes \mathbb{1}, \mathbf{y}^h \right\rangle \quad \text{s. t.} \quad \mathbf{y}^h \geq \mathbf{g}^h = \mathrm{g}^h \otimes \mathbb{1}. \tag{28}$$

The next section will discuss how problem (28) (or (27)) can be solved by low-rank tensor methods. The inequality constraints on the tensor pose a particular challenge.

4. Constrained optimization in tensor space. We now develop efficient methods for solving constrained optimization problems with tensors that use low-rank formats. In our approach, we operate on the full tensor space $\mathbb{R}^{n_1 \times \ldots \times n_d}$ and use low-rank tensors for making all operations efficient. To this end, we adapt nonlinear optimization techniques to low-rank tensor arithmetics and we interlace the required operations with truncations to suitably maintain moderate ranks. This leads to inexact optimization algorithms that will be presented in this section.

An alternative approach would be to work directly with the parametrization of the representing low-rank tensor format. In the case of tensors in HTD, the basis matrices at the leaves and the transfer matrices at the remaining nodes then would be the optimization variables, and for TT tensors these would be the representing tensors $\mathbf{u}_i$, $i \in [m]$, see section 2.2. A short discussion of possible methods for this latter approach is given at the end of this paper. Currently, we use methods that operate directly with the low rank format only as subproblem solvers, e.g., AMEn for linear systems.

4.1. Algorithms for solving the discrete optimal control problem. We next propose a method that can be used to solve the problem (22). To this end, we consider optimal control problems of the following form:

$$\min_{(\mathbf{y}^h, \mathrm{u}^h) \in \mathbb{R}^{N_0 \times \ldots \times N_m} \times \mathbb{R}^{\tilde{N}}} \mathbf{J}^h(\mathbf{y}^h, \mathrm{u}^h) \quad \text{s. t.} \quad \mathbf{A}^h \mathbf{y}^h = (\mathrm{B}^h \mathrm{u}^h) \otimes \boldsymbol{\omega}^h.^{-1}, \quad \mathrm{u}^h_{\mathrm{l}} \leq \mathrm{u}^h \leq \mathrm{u}^h_{\mathrm{u}} \tag{29}$$

with $\mathbf{J}^h(\mathbf{y}^h, \mathrm{u}^h) := \frac{1}{2}\langle \mathbf{y}^h - \hat{\mathbf{y}}^h, \mathbf{M}^h(\mathbf{y}^h - \hat{\mathbf{y}}^h)\rangle + \frac{\gamma}{2} {\mathrm{u}^h}^T \tilde{\mathrm{M}}^h \mathrm{u}^h$, where $\hat{\mathbf{y}}^h := \hat{\mathrm{y}}^h \otimes \boldsymbol{\omega}^h.^{-1}$.

We need some prerequisites for the applicability of our method:

Assumption 4.1.

- The linear operators $\mathbf{M}^h$ and $\mathbf{A}^h$ are symmetric, positive definite and easily (at least approximately) applicable to the used low-rank tensors.
- A suitable, symmetric and positive definite preconditioner $\mathbf{T}^h$ for the operator $\mathbf{A}^h$ is available. It is efficiently applicable to the used type of low-rank tensors.

EXAMPLE 4.2. The operators $\mathbf{M}^h$ (see equation (21)) and $\mathbf{A}^h$ from our examples above fulfill this assumption. The operators $\mathbf{A}^h$ can be applied approximately to tensors in HTD by i-mode matrix products and truncated sums. They can also be implemented in the $\{\mathrm{d},\mathrm{R}\}$-format for operators acting on TT tensors as introduced in the `tamen` package [8] since they are sums of Kronecker products of sparse matrices, which is exactly one possible form for $\{\mathrm{d},\mathrm{R}\}$. The inverse of the nominal operator, i.e., $\mathbf{T}^h\mathbf{y}^h := \mathrm{A}^h\left(\mathbb{E}_\alpha\alpha\right)^{-1}\circ_1\mathbf{y}^h$, that is the inverse of the system matrix for the mean value of the parameters α applied to the first mode of the tensor, can be used as preconditioner for $\mathbf{A}^h$. The application of it can be done efficiently using a precomputed sparse Cholesky factorization of the matrix $\mathrm{A}^h\left(\mathbb{E}_\alpha\alpha\right)$.

As the state equation of (29) has a unique solution $\mathbf{y}^h(\mathrm{u}^h) = {\mathbf{A}^h}^{-1}((\mathrm{B}^h\mathrm{u}^h)\otimes\boldsymbol{\omega}^{h}.^{-1})$, we can work with the reduced problem

$$\begin{aligned}&\min_{\mathrm{u}^h\in\mathbb{R}^{\tilde{N}}}\hat{\mathbf{J}}^h(\mathrm{u}^h) := \tfrac{1}{2}\langle\mathbf{y}^h(\mathrm{u}^h)-\hat{\mathbf{y}}^h,\mathbf{M}^h(\mathbf{y}^h(\mathrm{u}^h)-\hat{\mathbf{y}}^h)\rangle + \tfrac{\gamma}{2}{\mathrm{u}^h}^T\tilde{\mathrm{M}}^h\mathrm{u}^h\\ &\text{s.t.}\quad \mathrm{u}_\mathrm{l}^h\le\mathrm{u}^h\le\mathrm{u}_\mathrm{u}^h.\end{aligned}\tag{30}$$

Since already for moderate parameter dimension m it is intractable to represent the state by a full tensor when doing computations, we use an HTD approximation instead. Thus, we have to formulate the optimization algorithm in a way that all tensor operations stay within the set of HTD-representable tensors of manageable rank. For solving the state equation, mainly two approaches were considered: Application of a version of the preconditioned CG method that uses HTD tensors and involves truncation (i.e., rounding) as proposed in [31]. This requires the efficient application of the operator $\mathbf{A}^h$ to an HTD tensor, which is possible as already described. The remaining required operations are inner products, vector space operations, and truncation, which are all available in HTD. The inverse nominal operator preconditioner $\mathbf{T}^h$ is cheap to apply and works well in practice. This approach is viable, but in our experiments we found that the block iterative method AMEn [9] yields better computing times, lower tensor ranks and even slightly smaller errors than the HTD PCG method. AMEn can be applied when the right hand side, in our case $(\mathrm{B}^h\mathrm{u}^h)\otimes\boldsymbol{\omega}^{h}.^{-1}$, is given in the TT format and the operator $\mathbf{A}^h$ in the $\{\mathrm{d},\mathrm{R}\}$-format, in our case simply all operators as sparse matrices that act on the modes of the tensor. AMEn computes an approximate solution in TT format by a block iteration over the TT cores $\mathbf{u}$ and adapts the TT ranks dynamically. It can be viewed as an improved, rank adaptive version of ALS with smaller subproblems than in the MALS approach [23].

For computing the reduced gradient $\nabla\hat{\mathbf{J}}^h(\mathrm{u}^h)\in\mathbb{R}^{\tilde{N}}$ we use the adjoint approach. On $\mathbb{R}^{\tilde{N}}$ and $\mathbb{R}^{N_0\times\ldots\times N_m}$, respectively, we use the L^2-type inner products induced by the operators $\tilde{\mathrm{M}}_\mathrm{L}^h$ and $\mathbf{M}^h$, respectively, and represent gradients and dual quantities w.r.t. these inner products. The lumped mass matrix $\tilde{\mathrm{M}}_\mathrm{L}^h$, $(\tilde{\mathrm{M}}_\mathrm{L}^h)_{kk} = \sum_{l=1}^{\tilde{N}}\tilde{\mathrm{M}}_{kl}^h$, is used in the discrete control space to get equivalence to the element-wise projection in $L^2(D)$. All this makes the approach compatible with the function space setting. The discrete adjoint state $\mathbf{p}^h\left(\mathrm{u}^h\right)\in\mathbb{R}^{N_0\times\ldots\times N_m}$ is then defined as the solution of the following adjoint equation:

$$\mathbf{A}^h\mathbf{p}^h = -\mathbf{M}^h\left(\mathbf{y}^h(\mathrm{u}^h\right)-\hat{\mathbf{y}}^h).\tag{31}$$

It can be solved again by PCG or AMEn exactly as the state equation. The right hand side is already known from evaluating the objective function in (30). The discrete

gradient w.r.t. the L^2-inner product induced by $\tilde{\mathrm{M}}_{\mathrm{L}}^h$ is now given by

$$\nabla\hat{\mathbf{J}}^h(\mathrm{u}^h) = -(\tilde{\mathrm{M}}_{\mathrm{L}}^h)^{-1}{\mathrm{B}^h}^T\left\langle\boldsymbol{\omega}^h{}_{\cdot}{}^{-1}, \mathbf{p}^h\left(\mathrm{u}^h\right)\right\rangle_{[m],1+[m]} + \gamma\mathrm{u}^h, \tag{32}$$

and requires a tensor contraction and matrix-vector operations only. By representing the state and adjoint state as low-rank tensors and applying a low-rank tensor solver, we can therefore get good approximations of the function value and gradient. This allows to apply derivative-based optimization methods.

Due to their proven efficiency for solving control constrained optimal control problems with PDEs [21, 22, 45], we focus on semismooth Newton methods [21, 44, 41, 45] here. To this end, we write the necessary and by convexity also sufficient optimality conditions of (30) as a nonsmooth equation:

$$\mathrm{R}^h(\bar{\mathrm{u}}^h) \coloneqq \bar{\mathrm{u}}^h - \mathrm{P}^h(\bar{\mathrm{u}}^h - \tau\nabla\hat{\mathbf{J}}^h(\bar{\mathrm{u}}^h)) = 0. \tag{33}$$

where $\tau > 0$ is arbitrary. In the following, we will choose $\tau = 1/\gamma$, thus achieving that the argument of P^h then only depends on $\mathbf{p}^h$. Here, $\mathrm{P}^h : \mathbb{R}^{\tilde{N}} \to \mathbb{R}^{\tilde{N}}$ denotes the projection onto the set $\{\mathrm{u}^h \in \mathbb{R}^{\tilde{N}} : \mathrm{u}_{\mathrm{l}}^h \le \mathrm{u}^h \le \mathrm{u}_{\mathrm{u}}^h \text{ element-wise}\}$, where we use the norm induced by the lumped mass matrix $\tilde{\mathrm{M}}_{\mathrm{L}}^h$. We have

$$(\mathrm{P}^h(\mathrm{u}^h))_j = \min\{\max\{(\mathrm{u}_{\mathrm{l}}^h)_j, (\mathrm{u}^h)_j\}, (u_{\mathrm{u}})_j\}.$$

As an element of its generalized Jacobian we can choose

$$D\mathrm{P}^h(\mathrm{u}^h) \in \mathbb{R}^{\tilde{N}\times\tilde{N}}\text{diagonal}, \quad (D\mathrm{P}^h(\mathrm{u}^h))_{jj} = \begin{cases} 0 & \text{if } (\mathrm{u}^h)_j < (\mathrm{u}_{\mathrm{l}}^h)_j \\ 1, & \text{if } (\mathrm{u}_{\mathrm{l}}^h)_j \le (\mathrm{u}^h)_j \le (\mathrm{u}_{\mathrm{u}}^h)_j, \\ 0 & \text{if } (\mathrm{u}_{\mathrm{u}}^h)_j < (\mathrm{u}^h)_j. \end{cases}$$

Note that for active bounds we could choose any $(D\mathrm{P}^h(\mathrm{u}^h))_{jj} \in [0,1]$ instead of 1.

It is well known that the function R^h in (33) is semismooth. For $\tau = 1/\gamma$ this also holds true in the function space setting since there the projection onto U_{ad} is semismooth from $L^q(D)$ to $L^2(D)$ for any $q > 2$ and it is not difficult to see that the map inside of the projection, $\boldsymbol{p} \mapsto \boldsymbol{B}^*\boldsymbol{p}$ is bounded from $\boldsymbol{Y}$ to $Y = H_0^1(D)$ (not just to U). The discrete semismooth Newton system is given by

$$\mathrm{s}^h - D\mathrm{P}^h(\mathrm{u}^h - \tau\nabla\hat{\mathbf{J}}^h(\mathrm{u}^h))(\mathrm{s}^h - \tau\nabla^2\hat{\mathbf{J}}^h(\mathrm{u}^h)\mathrm{s}^h) = -\mathrm{u}^h + \mathrm{P}^h(\mathrm{u}^h - \tau\nabla\hat{\mathbf{J}}^h(\mathrm{u}^h)) \tag{34}$$

for the direction $\mathrm{s}^h \in \mathbb{R}^{\tilde{N}}$. There holds

$$\nabla^2\hat{\mathbf{J}}^h(\mathrm{u}^h)\mathrm{s}^h = (\tilde{\mathrm{M}}_{\mathrm{L}}^h)^{-1}{\mathrm{B}^h}^T\langle\boldsymbol{\omega}^h{}_{\cdot}{}^{-1}, {\mathbf{A}^h}^{-1}\mathbf{M}^h{\mathbf{A}^h}^{-1}((\mathrm{B}^h\mathrm{s}^h)\otimes\boldsymbol{\omega}^h{}_{\cdot}{}^{-1})\rangle_{[m],1+[m]} + \gamma\mathrm{s}^h.$$

Defining $D_\tau(\mathrm{u}^h) \coloneqq \tau D\mathrm{P}^h(\mathrm{u}^h - \tau\nabla\hat{\mathbf{J}}^h(\mathrm{u}^h))$ and choosing $\tau = 1/\gamma$ yields

$$D_{\gamma^{-1}}(\mathrm{u}^h) = \gamma^{-1}D\mathrm{P}^h(\gamma^{-1}(\tilde{\mathrm{M}}_{\mathrm{L}}^h)^{-1}{\mathrm{B}^h}^T\langle\boldsymbol{\omega}^h{}_{\cdot}{}^{-1}, \mathbf{p}^h(\mathrm{u}^h)\rangle_{[m],1+[m]}).$$

The semismooth Newton equation (34) becomes

$$\mathrm{s}^h + D_{\gamma^{-1}}(\mathrm{u}^h)(\tilde{\mathrm{M}}_{\mathrm{L}}^h)^{-1}{\mathrm{B}^h}^T\langle\boldsymbol{\omega}^h{}_{\cdot}{}^{-1}, {\mathbf{A}^h}^{-1}\mathbf{M}^h{\mathbf{A}^h}^{-1}((\mathrm{B}^h\mathrm{s}^h)\otimes\boldsymbol{\omega}^h{}_{\cdot}{}^{-1})\rangle_{[m],1+[m]} = -\mathrm{R}^h(\mathrm{u}^h).$$

To avoid relatively costly state and adjoint solves in Hessian-vector products we replace ${\mathbf{A}^h}^{-1}$ by $\mathbf{T}^h$. Further, we multiply by $\tilde{\mathrm{M}}_{\mathrm{L}}^h$ from the left and obtain the following Newton-type system:

$$\tilde{\mathrm{M}}_{\mathrm{L}}^h\mathrm{s}^h + D_{\gamma^{-1}}(\mathrm{u}^h){\mathrm{B}^h}^T\langle\boldsymbol{\omega}^h{}_{\cdot}{}^{-1}, \mathbf{T}^h\mathbf{M}^h\mathbf{T}^h((\mathrm{B}^h\mathrm{s}^h)\otimes\boldsymbol{\omega}^h{}_{\cdot}{}^{-1})\rangle_{[m],1+[m]} = -\tilde{\mathrm{M}}_{\mathrm{L}}^h\mathrm{R}^h(\mathrm{u}^h). \tag{35}$$

As the left hand side operator is not necessarily symmetric, we either can solve this system by a GMRES method or we can symmetrize the system by a suitable block elimination and then apply CG. This reduction is done by distinguishing the index sets $\mathcal{I}(\mathrm{u}^h) := \{j \in [\tilde{N}] : (D_{\gamma^{-1}}(\mathrm{u}^h))_{jj} = 0\}$ and $\mathcal{A}(\mathrm{u}^h) := [\tilde{N}] \setminus \mathcal{I}(\mathrm{u}^h)$. From (35) we then obtain $\mathrm{s}^h_{\mathcal{I}(\mathrm{u}^h)} = -\mathrm{R}^h(\mathrm{u}^h)_{\mathcal{I}(\mathrm{u}^h)}$, and inserting this into (35) yields the following smaller, symmetric positive definite system:

$$(\tilde{\mathrm{M}}^h_{\mathrm{L}} + \mathrm{W}^h)_{\mathcal{A}(\mathrm{u}^h),\mathcal{A}(\mathrm{u}^h)}\mathrm{s}^h_{\mathcal{A}(\mathrm{u}^h)} = (-\tilde{\mathrm{M}}^h_{\mathrm{L}}\mathrm{R}^h(\mathrm{u}^h))_{\mathcal{A}(\mathrm{u}^h)} + \mathrm{W}^h_{\mathcal{A}(\mathrm{u}^h)\mathcal{I}(\mathrm{u}^h)}\mathrm{R}^h(\mathrm{u}^h)_{\mathcal{I}(\mathrm{u}^h)} \quad (36)$$

with $\mathrm{W}^h\mathrm{s}^h := \gamma^{-1}{\mathrm{B}^h}^T\langle \boldsymbol{\omega}^{h,-1}, \mathbf{T}^h\mathbf{M}^h\mathbf{T}^h((\mathrm{B}^h\mathrm{s}^h) \otimes \boldsymbol{\omega}^{h,-1})\rangle_{[m],1+[m]}$. Note that the matrix W^h is not explicitly computed. Instead, we extend $\mathcal{A}(\mathrm{u}^h)$-subvectors to $\mathbb{R}^{\tilde{N}}$ by filling the $\mathcal{I}(\mathrm{u}^h)$-block with zeros, applying the full operator and then selecting the $\mathcal{A}(\mathrm{u}^h)$-subblock of the result. It is difficult to devise a preconditioner, but the system is reasonably well conditioned since the operator is the discrete approximation of a compact perturbation of the identity. The system is thus solved approximately by CG. The proposed semismooth Newton-type method is summarized in Algorithm 1.

Algorithm 1: Semismooth Newton method for the opt. control problem (29)

Input: Operators $\mathbf{A}^h$, $\mathbf{T}^h$, $\mathbf{M}^h$, matrices B^h, $\tilde{\mathrm{M}}^h$, vectors $\mathrm{u}^h_{\mathrm{l}}$, $\mathrm{u}^h_{\mathrm{u}}$, initial iterate u^h_0.
Parameters: Stopping tolerance $\varepsilon > 0$.
Output: Computed optimal control $\bar{\mathrm{u}}^h$.

for $k := 0, 1, 2, \ldots$ **do**
- Obtain approx. solution $\mathbf{y}^h(\mathrm{u}^h_k)$ of state eq. in (29) with $\mathrm{u}^h = \mathrm{u}^h_k$ by AMEn.
- Compute $\mathbf{p}^h(\mathrm{u}^h_k)$ by solving (31) approx. with $\mathbf{y}^h(\mathrm{u}^h) = \mathbf{y}^h(\mathrm{u}^h_k)$ by AMEn.
- Evaluate $\nabla\hat{\mathbf{J}}^h(\mathrm{u}^h_k)$ by (32) and $\mathrm{R}^h(\mathrm{u}^h_k)$ by (33).
- **if** $\mathrm{R}^h(\mathrm{u}^h_k)^T\tilde{\mathrm{M}}^h_{\mathrm{L}}\mathrm{R}^h(\mathrm{u}^h_k) < \varepsilon^2$, **then return** $\bar{\mathrm{u}}^h := \mathrm{u}^h_k$.
- Compute s^h_k approximately from (36) using a PCG method.
- Set $\mathrm{u}^h_{k+1} := \mathrm{u}^h_k + \mathrm{s}^h_k$.

As an initial point u^h_0, the solution $\bar{\mathrm{u}}^h_{\mathbb{E}_\alpha\alpha}$ of the average deterministic problem (16) can be used, as we expect the deterministic solution not to differ too much from the solution under uncertainty. It can be computed again by semismooth Newton, but with significantly less effort since the state equation then is a deterministic PDE.

As already discussed, instead of low-rank tensor methods, other approaches could be used to solve the underlying PDE with uncertainties. In particular, sparse grids could be used as this is done in, e.g., [28]. Except for sparse grids adjusted to the energy norm, the complexity of sparse grid methods is exponential in m [14], which makes our low-rank tensor approach attractive, which is polynomial in m unless the required ranks would be exponential in m, which is not the case in our experiments. As we will see later in our numerical results, the storage and operation complexities for low-rank tensors scale very well w.r.t. their order and dimensions.

4.2. Algorithms for solving the discrete parametrized obstacle problem. The parametric obstacle problem (28) from section 3.2 poses the numerical challenge that now the optimization variable $\mathbf{y}^h$ is a tensor which is subject to pointwise inequality constraints. For dealing with these constraints in a way that is viable with low-rank tensors, we propose a primal interior point algorithm.

Following the idea of barrier methods, we approximate problem (28) by an unconstrained problem that incorporates the element-wise constraints by a log-barrier function (cf. [36, sec. 19.6]) weighted with the barrier parameter $\mu > 0$ and the spatial lumped mass matrix $\mathrm{M}_\mathrm{L}^h \in \mathbb{R}^{N_0 \times N_0}$, i.e., $\left(\mathrm{M}_\mathrm{L}^h\right)_{kk} = \sum_{l=1}^{N_0} \mathrm{M}_{kl}^h$):

$$\min_{\mathbf{y}^h \in \mathbb{R}^{N_0 \times \ldots \times N_m}} \mathbf{F}^h(\mathbf{y}^h) - \mu \langle \mathbb{1}, \mathrm{M}_\mathrm{L}^h \circ_1 \ln(\mathbf{y}^h - \mathbf{g}^h) \rangle$$

with $\mathbf{F}^h(\mathbf{y}^h) = \frac{1}{2}\langle \mathbf{y}^h, \mathbf{A}^h \mathbf{y}^h \rangle - \langle \mathrm{b}^h \otimes \mathbb{1}, \mathbf{y}^h \rangle$. This will be solved by Newton's method for a decreasing sequence $(\mu_k)_k \subset \mathbb{R}_{>0}$ converging to 0. For a fixed parameter $\mu_k > 0$ and an iterate $\mathbf{y}_k^h$ the Newton equation for the direction $\mathbf{s}_k^h \in \mathbb{R}^{N_0 \times \ldots \times N_m}$ is given by

$$\mathbf{A}^h \mathbf{s}_k^h + \mu_k (\mathrm{M}_\mathrm{L}^h \circ_1 (\mathbf{y}_k^h - \mathbf{g}^h).^{-2}) \odot \mathbf{s}_k^h = -\mathbf{A}^h \mathbf{y}_k^h + \mathrm{b}^h \otimes \mathbb{1} + \mu_k \mathrm{M}_\mathrm{L}^h \circ_1 \left(\mathbf{y}_k^h - \mathbf{g}^h\right).^{-1}. \quad (37)$$

Since M_L^h is diagonal, the operator $\mathrm{M}_\mathrm{L}^h \circ_1$ can be written as an element-wise multiplication and thus the operator acting on $\mathbf{s}_k^h$ is symmetric and positive definite if $\mathbf{y}^h$ is strictly feasible. Hence, the Newton system can be solved by the PCG method with truncation. A good precondiitoner for the first summand is given by the "average inverse" $\mathbf{T}^h$ but as μ gets smaller the second summand becomes significant on a subset of tensor components related to the active set. Tailoring a preconditioner to this situation is more difficult in a low-rank tensor setting than for the standard vector case. Alternatively, and this is the approach we take, one can solve the systems by AMEn again. For this purpose we extended the AMEn code and the $\{\mathrm{d}, \mathrm{R}\}$-format of the `tamen` package [8] such that it can handle element-wise multiplication operators. In order to build the equation, one still has to compute the pointwise inverse $\mathbf{z}^h.^{-1} := \left(\mathbf{y}^h - \mathbf{g}^h\right).^{-1}$, which can not be done directly with low-rank tensors. Instead, one can apply a truncated Newton-Schulz method as proposed in [32] and already stated for structured matrices in [18, sec. 4.1]. The idea behind it is to apply Newton's method to an appropriate pointwise function with root $\mathbf{z}^h.^{-1}$, in this case $f(\mathbf{x}^h) := \mathbf{x}^h.^{-1} - \mathbf{z}^h$; the resulting iteration is $\mathbf{x}_{k+1}^h := \mathbf{x}_k^h \odot (2 \cdot \mathbb{1} - \mathbf{z}^h \odot \mathbf{x}_k^h)$. A suitable initial iterate is $\mathbf{x}_0^h = \frac{1}{\langle \mathbf{z}^h, \mathbf{z}^h \rangle} \mathbf{z}^h$ (for general $\mathbf{z}^h$) or $\mathbf{x}_0^h = \frac{1}{\|\mathbf{z}^h\|_F} \mathbb{1}$ (if $\mathbf{z}^h > 0$). It is important that all computations for the Newton step are performable within the tensor format. This can involve rank-increasing operations, thus making rank control by truncation necessary to keep computations efficient. Suitable bounds on the truncation error to ensure convergence of the iterative method can be found in [18, sec. 2.2]. The error can be measured by $\|\mathbb{1} - \mathbf{z}^h \odot \mathbf{x}_k^h\|_F$. In a similar way, other component-wise functions can be approximated, such as the inverse square root or the sign function. Based on this, further function, such as the absolute value, can be approximated. The latter operations are not required here, but can be used for other types of methods, such as penalty or projection-based algorithms. We found that the Newton-Schulz method works fast and reliable if the components of $\mathbf{z}^h$ are not too close to zero, but gives less accurate results when the interior point algorithm proceeds and the tensor $\mathbf{y}_k^h$ approaches the bounds $\mathbf{g}^h$. Hence, when the Newton-Schulz error becomes too large, we resort to computing the element-wise reciprocal by applying AMEn to the symmetric, positive definite system $\mathbf{z}^h \odot \mathbf{x}^h = \mathbb{1}$. This yields smaller errors while using tensors of the same ranks, but is more time-consuming.

The remaining required tensor operations consist of application of matrices to a mode, application of the linear operator $\mathbf{A}^h$, element-wise multiplication and summation of tensors. They all can be performed within the HT and TT format. Having computed the Newton direction, we select a step size $\sigma_k > 0$ and set $\mathbf{y}_{k+1}^h := \mathbf{y}_k^h + \sigma_k \mathbf{s}_k^h$. It is crucial that the iterates stay strictly feasible, i.e., $\mathbf{y}_k^h + \sigma_k \mathbf{s}_k^h > \mathbf{g}^h$ is required.

Since we always maintain $\mathbf{y}_k^h > \mathbf{g}^h$ (element-wise) this always holds true if $\mathbf{s}_k^h \geq 0$. If $\mathbf{s}_k^h$ has negative elements, then the value τ_k of the (signed) maximum element of the tensor $-\mathbf{s}_k^h \odot (\mathbf{y}_k^h - \mathbf{g}^h).^{-1}$ is positive and the above condition is equivalent to $\sigma_k < 1/\tau_k$.

We examined several ways to reliably approximate the maximal component of a tensor $\mathbf{z}^h$, including variants of a vector iteration for computing the maximum eigenvalue of $\mathbf{x}^h \mapsto \mathbf{z}^h \odot \mathbf{x}^h$ [11] and also by computing the p-norm for large p. Vector iteration starting with the tensor $\mathbb{1}$ as well as p-norm computation for $p = 2^k$ can both be done by repeated element-wise squaring combined with normalization, starting with $\mathbf{x}^h = \mathbf{z}^h$. It turns out that both approaches are not sufficiently robust since repeated element-wise squaring soon results in tensors that often cannot be represented sufficiently accurately by tensors of moderate ranks. We made the best experience by using global optimization to find the maximum tensor element. To this end, we use that in our case the tensor $\mathbf{y}^h$ contains the components of the multivariate function $\boldsymbol{y}^h(x,\alpha)$ for the tensor product basis $\{\phi_{j_0}(x)\beta_{j_1}^1(\alpha_1)\cdots\beta_{j_m}^m(\alpha_m)\}_{(j_0,\ldots,j_m)}$. Since $\hat{\boldsymbol{y}}^h(\alpha) := \max_{x\in D^h} \boldsymbol{y}^h(x,\alpha)$ can be evaluated efficiently, see below, it turns out to be faster to numerically maximize $\hat{\boldsymbol{y}}^h(\alpha)$ w.r.t. α rather than $\boldsymbol{y}^h(x,\alpha)$ w.r.t. (x,α). Function values can be computed using contraction: Since we work with a nodal FE-basis, the vector of nodal values of the FE-function $x \mapsto \boldsymbol{y}^h(x,\alpha)$, with α fixed, is given by $\langle \mathbf{y}^h, \bigotimes_{i=1}^m (\beta_1^i(\alpha_i),\ldots,\beta_{N_i}^i(\alpha_i))^T\rangle_{1+[m],[m]} \in \mathbb{R}^{N_0}$ and $\hat{\boldsymbol{y}}^h(\alpha)$ is the maximum over this vector's entries. We use the multilevel coordinate search method MCS [25], a (non-rigorous) global optimization method for box-constrained problems that only requires function values and is available as a MATLAB implementation.

Algorithm 2: Interior point algorithm for solving problems of type (28)

Input: Operator $\mathbf{A}^h$, preconditioner $\mathbf{T}^h$, vector b^h, tensor $\mathbf{g}^h$ in low-rank format, initial low-rank tensor $\mathbf{y}_0^h > \mathbf{g}^h$, initial parameter $\mu_0 > 0$

Parameters: $\mu_{\mathrm{fac}} \in (0,1)$, damping parameter $\eta \in (0,1)$, stopping criterion

Output: A solution $\bar{\mathbf{y}}^h$ in a low-rank format

for $k := 0,1,2,\ldots$ **do**
- Compute $\mathbf{z}_k^h := \mathbf{y}_k^h - \mathbf{g}^h$.
- Approximate the reciprocal $\mathbf{x}_k^h :\approx \left(\mathbf{z}_k^h\right).^{-1}$ by the Newton-Schulz method.
- **if** $\|\mathbf{x}_k^h \odot \mathbf{z}_k^h - \mathbb{1}\|_F$ is too large, **then**
  - Approximate $\mathbf{x}_k^h$ by solving $\mathbf{x}_k^h \odot \mathbf{z}^h = \mathbb{1}$ by AMEn.
- Compute the approximate gradient $\mathbf{r}_k^h := \mathbf{A}^h\mathbf{y}_k^h - \mathrm{b}^h \otimes \mathbb{1} - \mu_k \mathrm{M}_{\mathrm{L}}^h \circ_1 \mathbf{x}_k^h$.
- Approximately compute $\left(\mathbf{x}_k^h\right).^2$ by truncated element-wise multiplication.
- Compute $\mathbf{s}_k^h$ by AMEn (or by PCG with precond. $\mathbf{T}^h$) as approximate solution of (37) with right hand side $-\mathbf{r}_k^h$. In the operator, use truncated multiplication with $\left(\mathbf{x}_k^h\right).^2$ when evaluating $(\mathrm{M}_{\mathrm{L}}^h \circ_1 (\mathbf{y}_k^h - \mathbf{g}^h).^{-2}) \odot \mathbf{s}_k^h$.
- Compute $\tau_k :\approx \max(-\mathbf{s}_k^h \odot (\mathbf{y}_k^h - \mathbf{g}^h).^{-1})$ by a method that approximates the maximum tensor element, and set the stepsize $\sigma_k := \min(1, \eta\sigma_{k,\max})$ with $\sigma_{k,\max} = \frac{1}{\tau_k}$ if $\tau_k > 0$ and $\sigma_{k,\max} = \infty$ otherwise.
- **if** "stopping criterion" holds, **then return** $\bar{\mathbf{y}}^h := \mathbf{y}_k^h$.
- Set $\mathbf{y}_{k+1}^h := \mathbf{y}_k^h + \sigma_k \cdot \mathbf{s}_k^h$ and $\mu_{k+1} := \mu_{\mathrm{fac}} \cdot \mu_k$.

We now have developed all necessary ingredients for the interior point algorithm. It is summarized in Algorithm 2. Note that we always perform a Newton step, which

is damped for retaining strict feasibility by a step length σ_k depending on the maximum possible steplength and a damping factor $\eta \in (0,1)$ if the iterate $\mathbf{y}_k^h$ approaches the bound $\mathbf{g}^h$. In the formulation of Algorithm 2 we consider an abstract stopping criterion. Later, in our numerical tests, this will be based on the performance of the Newton direction computation and the computed stepsize and could also be placed at another position in the algorithm.

We note that, for several reasons we just explained, solving the problem class (28) is challenging. To tackle the huge problem size we use low-rank tensors as one of the most promising developments for breaking the curse of dimensionality. Computing element-wise reciprocals, stepsizes to the boundary of the feasible set, and solving linear systems is not possible by direct methods and thus needs careful consideration. These computations become increasingly challenging as the state approaches the obstacle. Due to the operator $\mathbf{s}_k^h \mapsto \mu_k(\mathrm{M}_\mathrm{L}^h \circ_1 (\mathbf{y}_k^h - \mathbf{g}^h).^{-2}) \odot \mathbf{s}_k^h$, the linear system (37) is harder to solve than, e.g., the state equation in (29). In the building blocks of the algorithm, truncations – and thus inexact computations – occur at many places. We are working on rigorous convergence theory and the corresponding error control mechanisms, but due to space limitations this is out of the scope of this paper.

5. Numerical results. Next, we test the proposed methods for the presented uncertain control problem and the parametrized obstacle problem numerically for various numbers of parameters and settings. All computations are done on a Linux machine with 128 GB RAM and two Intel Xeon 64bit processors with 2.40 GHz and 8 cores. Our implementations use MATLAB R2015a and the `htucker` toolbox 1.2 [32] and the `TT-Toolbox 2.2` [40] for all low-rank tensor computations. These toolboxes are serial implementations which do not explicitly use parallelism. In `htucker` we choose a linear, TT-like dimension tree and we wrote functions to convert between `htensor` (from `htucker`) and `tt_tensor` (from `TT-Toolbox`) objects.

Most of the linear systems are solved by AMEn from the `tamen` package presented in [8], which we modified to also allow for operators given as the element-wise multiplication by TT tensors. For global optimization we use a MATLAB implementation of MCS [25]. In all cases the space discretization is done by a conforming triangulation of the domain $D \subset \mathbb{R}^2$, using standard linear finite element nodal bases for $L^2(D^h)$ (control) and $H_0^1(D^h)$ (state). Hence, the FE space dimensions $\tilde{N}$ and N_0, respectively, coincide with the numbers of nodes and interior nodes of the triangulation, respectively. The parameters α are independently uniformly distributed on $(-1,1) =: \Gamma_i$ for all $i \in [m]$. Each parameter space $L^2(\Gamma_i, \mathbb{P}_{\alpha_i})$ is discretized by polynomials of the same order $\hat{N}-1$ with the Lagrange polynomials on $(-1,1)$ w.r.t. the $N_i := \hat{N}$ Gaussian quadrature points as the basis (i.e., $\boldsymbol{\omega}^h = \mathbb{1}$ always). As explained, the corresponding Gaussian quadrature weights are used to compute stochastic integrals.

5.1. Results for the optimal control problem with parametrized subdomains. The optimal control problem (13) is solved for the case of a parametrized state equation operator with uncertain radii and uncertain coefficients as in Example 3.3. In fact, we also tested the setting of Example 3.1 with very good results, but for brevity only results for the more challenging problem class of Example 3.3 are presented here. The spatial domain is $D = (-1,1) \times (-1,1) \subset \mathbb{R}^2$. To be able to easily change the number m of parameters, we choose the subdomains $D_i(\alpha_i)$ to be disks with their midpoints placed on a circle with radius $\zeta_{\mathrm{circ}} = 0.7$ around zero and with their radii varying such that they stay within D and do not intersect with each other

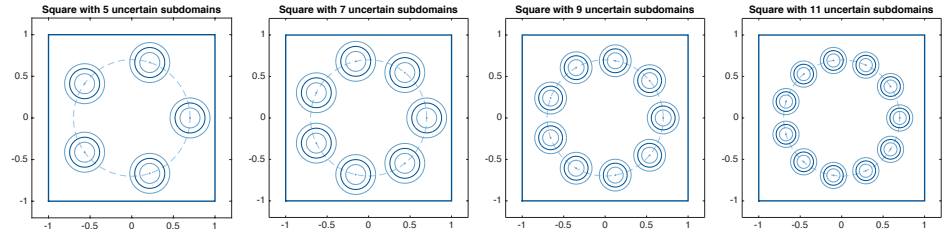

FIGURE 1. *The disks and their size ranges for the used numbers of parameters.*

for any selection of the parameters. Concretely, we choose

$$D_i(\alpha_i) := \{x \in \mathbb{R}^2 : (x_1 - \zeta_{\text{circ}} \cdot \cos(\tfrac{4\pi(i-1)}{m}))^2 + (x_1 - \zeta_{\text{circ}} \cdot \sin(\tfrac{4\pi(i-1)}{m}))^2 < \zeta(\alpha_i)^2\},$$

with $\zeta(\alpha_i) := ((1+\alpha_i)\zeta_{\text{maxfac}} + (1-\alpha_i)\zeta_{\text{minfac}})/2 \cdot \min(\zeta_{\text{circ}} \cdot \sin(2\pi/m), 1-\zeta_{\text{circ}})$ and radius factors $\zeta_{\text{minfac}} = 0.4 < 0.8 = \zeta_{\text{maxfac}}$. This means that the radii vary between $\zeta_{\text{minfac}} \in (0,1)$ and $\zeta_{\text{maxfac}} \in (\zeta_{\text{minfac}}, 1)$ times the maximum possible radius $\min(\zeta_{\text{circ}} \cdot \sin(2\pi/m), 1-\zeta_{\text{circ}})$. The subdomains for the average parameter selection and their minimum and maximum sizes can be seen for the used numbers of parameters in Figure 1. The coefficients on the disks are $\sigma_i = 5.0$ and on the rest of the domain $\sigma_0 = 1.5$, while the influence coefficients ϑ_i were all taken to be 25%. Radii ranging between 0.4 and 0.8 times a given maximum radius results in a deviation of more than 33%, which is quite a lot thinking about technical applications. We use the desired state $\hat{y}(x) := \frac{1}{10}\sin(\pi x_1) \cdot \sin(\pi x_2) \cdot \exp(x_1 + 1)$, the control bounds -2 and 2, respectively, and the regularization parameter $\gamma = 0.005$. The discretization uses $\hat{N} = 9$ nodes per parameter and $\tilde{N} = 10201$ finite element nodes on a uniform grid, $N_0 = 9801$ of them interior nodes. The assembly of the matrices $\mathrm{A}_i^h(a_{k_i}^i)$ is done by a smooth approximation of the indicator function, where the jump between 1 and 0 is smoothed by a polynomial of degree 9 within a range of half the diameter of the elements, and a quadrature rule of high order. But this procedure is not central here. The operator $\mathbf{A}^h$ is passed to AMEn in the $\{\mathrm{d},\mathrm{R}\}$-format with sparse matrices. Then we apply the semismooth Newton method (Algorithm 1) with the stopping tolerance $\varepsilon = 3.0 \cdot 10^{-5}$. As the KKT residual (33) is only computable up to a certain accuracy due to the error in the adjoint state, we also would stop the algorithm when observing its stagnation, but this does not occur in the runs we report here. In a preprocessing step we compute the optimal solution u_0^h for the deterministic problem (16) (inserting $\alpha = \mathbb{E}_\alpha \alpha$), also by a semismooth Newton method with $\varepsilon = 10^{-10}$, starting with initial control 0. We use it as initial point for the semismooth Newton method for the problem with uncertainties. The state and adjoint equations are solved by AMEn, starting with the "reference solution", i.e., the state or adjoint state of the deterministic problem for a mean parameter selection copied into the whole tensor, and with a maximum number of 20 sweeps in iteration $k = 0$ of Algorithm 1 and maximum obtained TT rank 105 for all problem sizes for comparability reasons. In all subsequent iterations $k \geq 1$ we choose the (adjoint) state from the previous iteration truncated to rank 80 as initial guess for the AMEn solution of the (adjoint) state equation and perform 5 sweeps. We have to mention that the rank here had to be chosen quite high compared to other low-rank tensor applications. But we believe that it is important to show that also quite hard problems can be handled by low-rank tensor methods. The small AMEn subproblems were solved directly, while the problems with 1000 unknowns or more

Number of parameters m:	10	14	18	22
Dimension $\prod_{i=0}^{m} N_i$ of the state:	$3.42 \cdot 10^{13}$	$2.24 \cdot 10^{17}$	$1.47 \cdot 10^{21}$	$9.65 \cdot 10^{24}$
Preprocess (deterministic problem):				
Number of iterations:	6	6	6	7
Obtained optimality measure:	$0.97 \cdot 10^{-11}$	$4.51 \cdot 10^{-11}$	$1.38 \cdot 10^{-11}$	$0.61 \cdot 10^{-11}$
Total Newton matrix-vector products:	25	21	24	26
Computing time (s):	2.18	1.85	1.97	2.27
Main process (problem with uncertainties):				
Number of iterations:	2	2	2	2
Optimality measure at initial point:	$1.67 \cdot 10^{-2}$	$1.85 \cdot 10^{-2}$	$1.23 \cdot 10^{-2}$	$1.07 \cdot 10^{-2}$
Obtained optimality measure:	$0.70 \cdot 10^{-5}$	$0.58 \cdot 10^{-5}$	$0.81 \cdot 10^{-5}$	$2.59 \cdot 10^{-5}$
Average reached rel. residual for states:	$0.63 \cdot 10^{-3}$	$2.57 \cdot 10^{-3}$	$3.23 \cdot 10^{-3}$	$3.87 \cdot 10^{-3}$
Total Newton matrix-vector products:	6	7	5	7
Average time for computing states (min):	13.8	24.1	34.7	47.9
Total computing time (h):	1.4	2.4	3.5	4.8
Relative L^2-difference $\frac{\|\mathrm{u}_0^h - \bar{\mathrm{u}}^h\|_{\tilde{\mathrm{M}}^h}}{\|\mathrm{u}_0^h\|_{\tilde{\mathrm{M}}^h}}$:	$0.94 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$0.67 \cdot 10^{-2}$	$0.57 \cdot 10^{-2}$

TABLE 1
Problem dimension and results of Algorithm 1 for the different numbers of parameters

were solved iteratively by CG. We selected the tolerances such that a good balance between accuracy and computing times was achieved (Table 1).

The uncertain problem was solved for 5, 7, 9 and 11 disks, i.e., $m \in \{10, 14, 18, 22\}$ parameters. The results are summarized in Table 1. Dividing the total number of Newton matrix times vector products by the number of Newton iterations, we see that in the semismooth Newton methods (deterministic and uncertain problems) the relative stopping tolerance 10^{-2} in the CG method for solving the semismooth Newton equation was reached in about 3 or 4 steps always. Comparable but increasing relative residuals in the discrete $H^{-1}(D) \otimes L^2(\Gamma; \mathbb{P}_\alpha)$-norm (row "Average reached rel. residual for states") are obtained by AMEn for all problem sizes while keeping the tensor rank bounded by 105 always. Table 1 confirms that a semismooth Newton method is a very good choice for the deterministic as well as the uncertain problem. The deterministic problem can be solved in 6 or 7 steps to high accuracy and provides a good initial iterate for solving the problem under uncertainty. The measure of optimality (discrete $L^2(D^h)$-norm of the residual $\mathrm{R}^h(\mathrm{u}_k^h)$, see (33)) is reduced by a factor of around 10^3 within 2 steps, enough to reach the desired tolerance of $3.0 \cdot 10^{-5}$ in all cases. The scaling of the method w.r.t. the number of parameters m can be seen in the computing times (average time for the solution of the state and adjoint equations and the total computing time). They increase approximately linearly with the number of parameters[6], which reflects the good complexity of low-rank tensor arithmetics. The main cost of the algorithms is the solution of the tensor equations (one state equation and one adjoint equation per iteration) while the rest of the computations is negligible: The overall computing time is about 6 times the average time for one tensor equation solve (2 solves in each of the 2 iterations and 2 more for the final residual that is accepted for termination).

Figure 2 shows two plots of the deterministic and the uncertain control for 10 parameters. One recognizes the disks where the coefficient is higher and especially that the uncertain control is smoother on the boundaries of the disks than the deterministic one due to the uncertain radii. The upper and lower bound (± 2) on the control are

[6] In theory they scale linearly with the number of parameters [9], but there are several effects like extra iterations for difficult subproblems, which influence the result in practice.

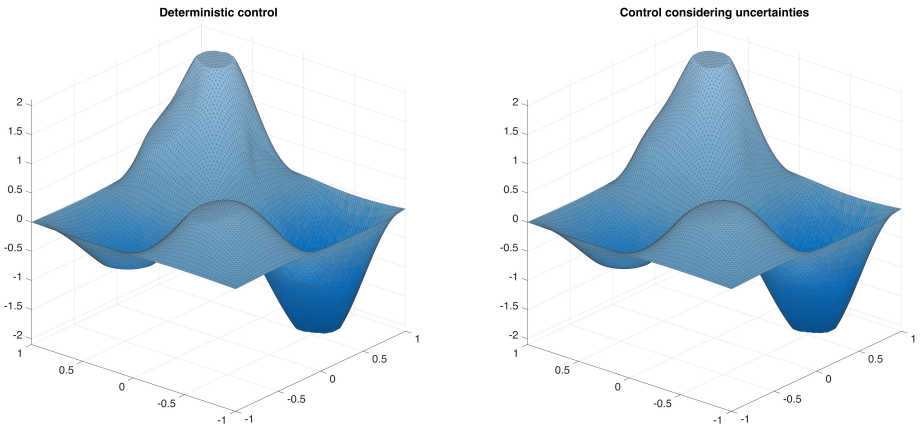

FIGURE 2. *The computed deterministic and uncertain controls for 10 parameters.*

Number of parameters m:	10	14	18	22
Errors in the state:				
Average relative $L^2(D^h)$-error:	$0.09 \cdot 10^{-3}$	$0.34 \cdot 10^{-3}$	$0.43 \cdot 10^{-3}$	$0.43 \cdot 10^{-3}$
Maximum relative $L^2(D^h)$-error:	$0.34 \cdot 10^{-3}$	$1.08 \cdot 10^{-3}$	$1.34 \cdot 10^{-3}$	$1.51 \cdot 10^{-3}$
Average relative $H^1(D^h)$-error:	$0.43 \cdot 10^{-3}$	$1.85 \cdot 10^{-3}$	$2.29 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$
Maximum relative $H^1(D^h)$-error:	$0.96 \cdot 10^{-3}$	$3.64 \cdot 10^{-3}$	$4.60 \cdot 10^{-3}$	$5.39 \cdot 10^{-3}$
Average relative $L^\infty(D^h)$-error:	$0.20 \cdot 10^{-3}$	$0.78 \cdot 10^{-3}$	$0.98 \cdot 10^{-3}$	$1.29 \cdot 10^{-3}$
Maximum relative $L^\infty(D^h)$-error:	$0.69 \cdot 10^{-3}$	$2.10 \cdot 10^{-3}$	$2.90 \cdot 10^{-3}$	$3.87 \cdot 10^{-3}$
Errors in the adjoint state:				
Average relative $L^2(D^h)$-error:	$0.09 \cdot 10^{-3}$	$0.27 \cdot 10^{-3}$	$0.31 \cdot 10^{-3}$	$0.46 \cdot 10^{-3}$
Maximum relative $L^2(D^h)$-error:	$0.72 \cdot 10^{-3}$	$2.84 \cdot 10^{-3}$	$1.89 \cdot 10^{-3}$	$1.96 \cdot 10^{-3}$
Average relative $H^1(D^h)$-error:	$0.26 \cdot 10^{-3}$	$1.30 \cdot 10^{-3}$	$1.55 \cdot 10^{-3}$	$2.60 \cdot 10^{-3}$
Maximum relative $H^1(D^h)$-error:	$1.98 \cdot 10^{-3}$	$5.68 \cdot 10^{-3}$	$5.84 \cdot 10^{-3}$	$8.48 \cdot 10^{-3}$
Average relative $L^\infty(D^h)$-error:	$0.24 \cdot 10^{-3}$	$0.92 \cdot 10^{-3}$	$1.08 \cdot 10^{-3}$	$1.57 \cdot 10^{-3}$
Maximum relative $L^\infty(D^h)$-error:	$1.11 \cdot 10^{-3}$	$4.44 \cdot 10^{-3}$	$4.52 \cdot 10^{-3}$	$5.73 \cdot 10^{-3}$

TABLE 2
Estimated errors in the computed state and adjoint state for the different numbers of parameters

both active on parts of the domain.

Clearly, the reliability of our results is of interest also. Therefore, to analyze the accuracy of the computed optimal state and the corresponding adjoint state, we proceed as follows: We randomly choose 5000 collocation points, i.e., realizations of the parameter vector α consisting of Gaussian nodes, and solve the deterministic state and adjoint equation for these parameter realizations to high accuracy (cf. [2, sec. 6]). Then we extract the corresponding solutions from the low-rank tensors and compare the results, as seen in Table 2. It is important to see that the errors are of comparable size for all numbers of parameters and only slightly increasing.

5.2. Results for the obstacle problem with fixed subdomains. As the obstacle problem (28) is in general more difficult to solve due to the pointwise constraints on the tensor, we consider it on fixed subdomains with uncertain coefficients on each subdomain. We follow Example 3.1 and use the unit disk $D := \{x \in \mathbb{R}^2 : \|x\|_2 < 1\}$ as the spatial domain with the $m \in \{5, 6, 7, 8\}$ subdomains chosen as shown in Figure 3. The average coefficients σ_i are all taken to be 8.0 while a deviation of $\vartheta_i = 15\%$ is allowed on each subdomain. The functional b is given by

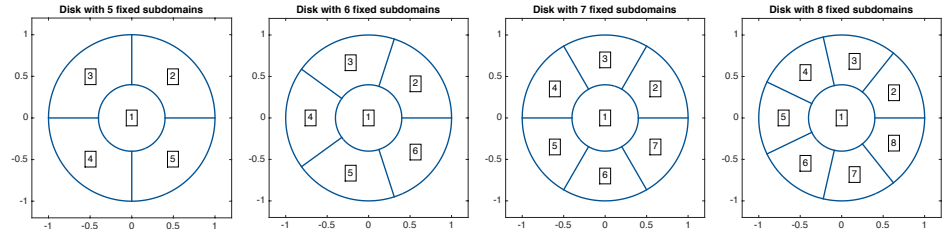

FIGURE 3. *Subdomains for the different numbers of parameters in the obstacle problem.*

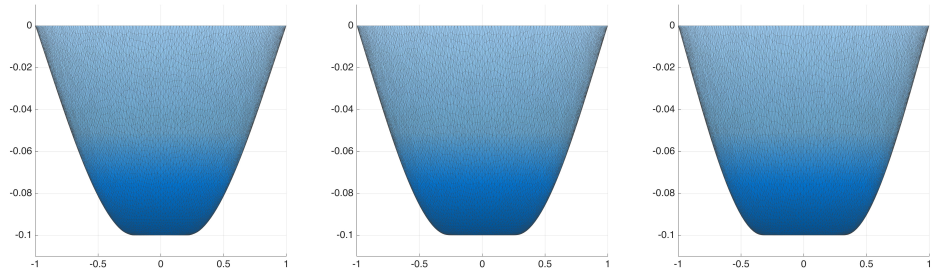

FIGURE 4. *Solutions of the obstacle problem with 5 parameters for maximum, mean and minimum coefficients (side view). These solutions were extracted from the resulting low-rank tensor.*

$\langle b, y\rangle_{Y^*,Y} = \int_D (4x_1^2 + 5x_2^2 - 6)y(x)\,dx$ and the obstacle is taken constant on the whole domain: $g \equiv -0.1$. We apply Algorithm 2 with parameters $\mu_0 = 0.1$, $\mu_{\text{fac}} = 0.65$ and $\eta = 0.8$. We observed that when the iterates get too close to the bound, the accuracy of the low-rank tensor approximation of the pointwise inverse is reduced, and, while the interior point algorithm proceeds in reducing μ, at some point this can lead to relatively poor results in the AMEn computation of the Newton step. Thus, we stop the whole algorithm when the obtained relative residual in the Frobenius norm (the AMEn default) is above 0.2. An alternative quality measure for the search direction is the obtained maximum step length. If it is below 0.01, our algorithm would also stop without performing this last step, but in all presented tests always the Newton system residual termination criterion became active.

Again we use linear finite elements for the space discretization with $N_0 = 8065$ interior nodes. In all configurations the discretization is the same for comparability reasons and chosen such that each element lies in exactly one of the approximate subdomains. For each parameter, $N_i = 17$ grid points are used. In a preprocessing step we obtain a solution $\bar{y}^h$ of the deterministic problem (26) for the parameters set to the average value by a semismooth Newton method and use $\mathbf{y}_0^h = 0.8 \cdot \bar{y}^h \otimes \mathbb{1}$ as the strictly interior initial tensor for the interior point method. The results are summarized in Table 3. In the algorithm, the iterates are truncated to rank 80 in each step. But we allow higher ranks ≤ 120 during the Newton-Schulz iteration, ≤ 100 for the element-wise reciprocal and ≤ 120 for the squared reciprocal as well as when applying AMEn. In AMEn we perform at most 15 sweeps first and solve the subproblems iteratively. Up to 5 more sweeps are performed with direct subproblem solver and maximum rank 82 if no satisfactory results were achieved before. We compute the maximum allowed stepsize to the bounds using the global optimization algorithm MCS [25]. Since only a quite low accuracy is required (a damping by $\eta = 0.8$ is done afterwards), the default

Number of parameters m:	5	6	7	8
Dimension $\prod_{i=0}^{m} N_i$ of discrete solution $\mathbf{y}^h$:	$1.15 \cdot 10^{10}$	$1.95 \cdot 10^{11}$	$3.31 \cdot 10^{12}$	$5.63 \cdot 10^{13}$
Results of the algorithm:				
Number of iterations:	14	13	13	14
Average comp. time for elem. reciprocal (s):	469.4	503.1	646.5	1113.9
Average comp. time for Newton direction (s):	134.5	158.1	184.3	355.3
Average computing time for step length (s):	136.0	95.8	189.5	226.9
Overall computing time (h):	3.1	3.0	4.0	7.2
Errors in the computed solution:				
Average relative $L^2(D^h)$-error:	$3.25 \cdot 10^{-3}$	$4.69 \cdot 10^{-3}$	$4.69 \cdot 10^{-3}$	$3.49 \cdot 10^{-3}$
Maximum relative $L^2(D^h)$-error:	$3.44 \cdot 10^{-3}$	$4.85 \cdot 10^{-3}$	$4.89 \cdot 10^{-3}$	$3.74 \cdot 10^{-3}$
Average relative $H^1(D^h)$-error:	$6.08 \cdot 10^{-3}$	$8.42 \cdot 10^{-3}$	$8.42 \cdot 10^{-3}$	$6.47 \cdot 10^{-3}$
Maximum relative $H^1(D^h)$-error:	$6.55 \cdot 10^{-3}$	$8.92 \cdot 10^{-3}$	$8.94 \cdot 10^{-3}$	$6.99 \cdot 10^{-3}$
Average relative $L^\infty(D^h)$-error:	$3.82 \cdot 10^{-3}$	$5.39 \cdot 10^{-3}$	$5.37 \cdot 10^{-3}$	$4.13 \cdot 10^{-3}$
Maximum relative $L^\infty(D^h)$-error:	$4.36 \cdot 10^{-3}$	$6.03 \cdot 10^{-3}$	$6.03 \cdot 10^{-3}$	$5.15 \cdot 10^{-3}$

Relative differences between the solutions for the highest and the lowest parameter selection:
L^2-difference: $10.36 \cdot 10^{-2}$ H^1-difference: $16.72 \cdot 10^{-2}$ L^∞-difference: $10.17 \cdot 10^{-2}$
Range of the number of active nodes: $[521, 1139]$

TABLE 3
Problem dimensions and results of the interior point method

settings of MCS, which target at high precision, were modified: We increased the stopping tolerance for the local search and decreased the number of steps for the local search, the number of levels and the number of sweeps. We observed good results of the method (when applying it for 5 parameters and evaluating the full tensor for comparison), but choose the relatively small damping parameter $\eta = 0.8$ as a safety margin. The steplength $\sigma_k = 1$ is accepted in the majority of cases. It only becomes smaller in the last one or two iterations due to the increasing errors in the computed search direction.

Table 3 shows that 13 to 14 interior point iterations are performed and that in all cases we achieve comparable accuracies. The computing times for the subproblems and hence the overall computing time grow with the number of iterations and the problem dimension. Only the computing time for the tensor maximum by MCS cannot be related to the problem dimension. Especially in the last iterations of the interior point algorithm AMEn is not able to reach the desired residual tolerance and has to perform the maximum number of sweeps and the additional 5 sweeps with the more expensive direct solves of linear equations. This influences the overall computing time, which can be kept below 8 hours in all cases. In the case of 8 parameters this results in an increased runtime: The stepsize computation in the final (14th) iteration and the computation of the reciprocal and Newton direction, which causes stopping the algorithm afterwards, take 2.4 hours in total. Performing the last iteration improves the maximum relative L^2-error of the solution from $4.90 \cdot 10^{-3}$ to $3.74 \cdot 10^{-3}$.

The errors are again evaluated on 5000 randomly selected collocation points. In all cases they are about 0.5% depending on the used norm. The maximum error in space and the average error do not differ much, which means that we get a uniform quality of the solutions for all parameter selections. Since tensor extraction can be performed quickly, this kind of error evaluation can, if desired, already be performed during the algorithm to get an online error prediction. It also enables quick access to solutions of problem (24) for any parameter selection α. Therefore, the low-rank tensor solution can be used for evaluating a wide range of UQ-relevant quantities efficiently. The graphs of some extracted solutions are displayed in Figure 4. The

lower bound is fulfilled and the weaker displacement for larger coefficients can be recognized, especially the varying size of the active sets. Below Table 3 the differences of the solutions for largest and smallest coefficients are listed. The range of the number of active nodes given in the last row also depicts the difference between the solutions.

6. Conclusions and outlook. Tensor representations and computations are well-suited for the treatment of problems with multiple parameters. Until now, mainly representation schemes, approximation algorithms, methods for solving linear systems, and algorithms for unconstrained optimization, especially linear least-squares problems, have been developed for low-rank tensors. In this paper, we considered *inequality constrained* optimization problems in tensor space and applied them to optimal control problems with uncertain coefficients and control constraints and to a parametrized obstacle problem. A suitable discretization was proposed that resulted in finite-dimensional problems, which respect the infinite-dimensional structure and involve tensors of high order. We developed inexact optimization methods for solving these huge-scale problems on the full tensor space that use efficient low-rank tensor arithmetics und truncation. Two classes of methods were discussed: Semismooth Newton methods for constrained optimal control under uncertainty and interior point methods for optimization problems with element-wise inequality constraints on the whole tensor. The methods work with the HTD and TT low-rank tensor formats and use AMEn for solving linear tensor equations and either the Newton-Schulz iteration or AMEn for computing element-wise reciprocals, which arise in the interior point method. Numerical tests showed the efficiency of the algorithms and especially their ability to make problems involving many parameters tractable.

We are currently developing a rigorous convergence theory for the discussed inexact algorithms. It builds on general convergence analyses of inexact optimization methods, see, e.g., [4, 10, 20, 28, 47] and combines it with error estimates for (tensor) discretizations, adaptive tensor rank selection, and truncation as well as error control for state equation and other system solves.

Another possible approach for optimization with low-rank tensors is to insert low-rank tensor formats directly into the problem formulation and to choose the representing coefficients as optimization variables. In the case of TT, the representation of a tensor then would be given by $\mathbf{y}^h = \mathcal{TT}(\mathbf{z}^h)$, where $\{\mathbf{z}_i^h\}$ are TT cores of given ranks. This approach is, on the one hand, appealing since it incorporates the low-rank structure directly into the problem and thus reduces the dimension. At the same time, however, it makes the problem highly nonlinear and nonconvex and usually these tensor representations are not unique in general. One can use orthonormalization and other techniques to obtain a unique representation via the quotient manifold [42, 46]. This can be used as a basis for Riemannian optimization techniques [1], which are currently investigated for unconstrained optimization problems with low-rank tensors (especially approximation problems) in, e.g., [5, 30, 43]. We plan to study extensions of these manifold optimization approaches to more general, inequality constrained tensor optimization problems, such as those considered in this paper, in the near future.

Acknowledgements. The first author was partially supported by "TopMath", which is a graduate program of the Elite Network of Bavaria, Germany, and a graduate center of TUM Graduate School, and the DFG/FWF International Research Training Group (IGDK 1754) "Optimization and Analysis for Partial Differential Equations with Nonsmooth Structures".

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, 2007.

[2] J. Ballani and L. Grasedyck, *Hierarchical tensor approximation of output quantities of parameter-dependent PDEs*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 852–872.

[3] A. Barth, C. Schwab, and N. Zollinger, *Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients*, Numer. Math., 119 (2011), pp. 123–161.

[4] S. Bellavia, *Inexact interior-point method*, J. Optim. Theory Appl., 96 (1998), pp. 109–121.

[5] C. Da Silva and F. Herrmann, *Optimization on the Hierarchical Tucker manifold – Applications to tensor completion*, Linear Algebra Appl., 481 (2015), pp. 131–173.

[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.

[8] S. V. Dolgov, *Alternating minimal energy approach to ODEs and conservation laws in tensor product formats*. arXiv preprint 1403.8085, 2014.

[9] 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.

[10] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 19 (1994), pp. 393–422.

[11] M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies, and E. Zander, *Efficient analysis of high dimensional data in tensor formats*, in Sparse Grids and Applications, J. Garcke and M. Griebel, eds., Springer-Verlag, 2013.

[12] A. Falcó and A. Nouy, *Proper generalized decomposition for nonlinear convex problems in tensor Banach spaces*, Numer. Math., 121 (2012), pp. 503–530.

[13] R. Forster and R. Kornhuber, *A polynomial chaos approach to stochastic variational inequalities*, J. Numer. Math., 18 (2010), pp. 235–255.

[14] J. Garcke, *Sparse grids in a nutshell*, in Sparse Grids and Applications, J. Garcke and M. Griebel, eds., Springer-Verlag, 2013.

[15] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.

[16] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78.

[17] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*, Springer-Verlag, 2012.

[18] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Approximate iterations for structured matrices*, Numer. Math., 109 (2008), pp. 365–383.

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

[20] M. Heinkenschloss and L. N. Vicente, *Analysis of inexact trust-region SQP algorithms*, SIAM J. Optim., 12 (2002), pp. 283–302.

[21] M. Hintermüller, K. Ito, and K. Kunisch, *The primal-dual active set strategy as a semismooth Newton method*, SIAM J. Optim., 13 (2002), pp. 865–888 (2003).

[22] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Springer, 2009.

[23] 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.

[24] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen, *Computations in quantum tensor networks*, Linear Algebra Appl., (2012).

[25] W. Huyer and A. Neumaier, *Global optimization by multilevel coordinate search*, J. Global Optim., 14 (1999), pp. 331–355.

[26] D. Kinderlehrer and G. Stampcchia, *An Introduction to Variational Inequalities and their Applications*, SIAM, 2000.

[27] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[28] D. P. Kouri, M. Heinkenschloss, D. Ridzal, and B. G. van Bloemen Waanders, *Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty*, SIAM J. Sci. Comput., 36 (2014), pp. A3011–A3029.

[29] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT, 54 (2014), pp. 447–468.

[30] ———, *Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure*, tech. report, EPF Lausanne, July 2015.

[31] D. Kressner and C. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316.
[32] ———, *htucker – a Matlab toolbox for tensors in hierarchical Tucker format*, ACM Trans. Math. Software, 40 (2014).
[33] F. Y. Kuo, C. Schwab, and I. H. Sloan, *Quasi-Monte Carlo finite elements methods for a class of elliptic partial differential equations with random coefficients*, SIAM J. Numer. Anal., 50 (2012), pp. 3351–5574.
[34] V. Murg, F. Verstraete, Ö. Legeza, and R. M. Noack, *Simulating strongly correlated quantum systems with tree tensor networks*, Physical Review B, 82 (2010).
[35] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, 1992.
[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, second ed., 2006.
[37] I. Oseledets and E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.
[38] I. V. Oseledets, *DMRG approach to fast linear algebra in the TT-format*, Comput. Methods Appl. Math., 11 (2011), pp. 382–393.
[39] ———, *Tensor-Train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.
[40] ———, *TT-toolbox 2.2: Fast multidimensional array operations in TT format*, 2012.
[41] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Math. Progamming, 58 (1993), pp. 353–367.
[42] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Numer. Anal., (2013).
[43] M. Steinlechner, *Riemannian optimization for high-dimensional tensor completion*, tech. report, EPF Lausanne, March 2015.
[44] M. Ulbrich, *Semismooth Newton methods for operator equations in function spaces*, SIAM J. Optim., 13 (2002), pp. 805–842 (2003).
[45] ———, *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*, vol. 11 of MOS-SIAM Series on Optimization, SIAM, Philadelphia, PA, 2011.
[46] A. Uschmajew and B. Vandereycken, *The geometry of algorithms using hiearchical tensors*, Linear Algebra Appl., 439 (2013), pp. 133–166.
[47] J. C. Ziems and S. Ulbrich, *Adaptive multilevel inexact SQP methods for PDE-constrained optimization*, SIAM J. Optim., 21 (2011), pp. 1–40.