Weierstraß-Institut
für Angewandte Analysis und Stochastik
Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Stochastic topology optimisation with hierarchical tensor reconstruction

Martin Eigel[1], Johannes Neumann[1], Reinhold Schneider[2], Sebastian Wolf[2]

submitted: December 22, 2016

[1] Weierstrass Institute
Mohrenstr. 39
10117 Berlin, Germany
E-Mail: martin.eigel@wias-berlin.de
johannes.neumann@wias-berlin.de

[2] Technische Universität Berlin – Sekretariat MA 5-3
Straße des 17. Juni 136
10623 Berlin, Germany
E-Mail: schneidr@math.tu-berlin.de
wolf@math.tu-berlin.de

No. 2362
Berlin 2016

2010 *Mathematics Subject Classification.* 35R60, 47B80, 60H35, 65C20, 65N12, 65N22, 65J10.

Key words and phrases. Partial differential equations with random coefficients, Tensor representation, Tensor train, Uncertainty quantification, Topology optimization, Phase field, Adaptive methods, Low-rank, Tensor reconstruction, Risk measures.

Research of J. Neumann was funded by the DFG MATHEON project SE13.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract. A novel approach for risk-averse structural topology optimization under uncertainties is presented which takes into account random material properties and random forces. For the distribution of material, a phase field approach is employed which allows for arbitrary topological changes during optimization. The state equation is assumed to be a high-dimensional PDE parametrized in a (finite) set of random variables. For the examined case, linearized elasticity with a parametric elasticity tensor is used. Instead of an optimization with respect to the expectation of the involved random fields, for practical purposes it is important to design structures which are also robust in case of events that are not the most frequent. As a common risk-aware measure, the Conditional Value at Risk (CVaR) is used in the cost functional during the minimization procedure. Since the treatment of such high-dimensional problems is a numerically challenging task, a representation in the modern hierarchical tensor train format is proposed. In order to obtain this highly efficient representation of the solution of the random state equation, a tensor completion algorithm is employed which only required the pointwise evaluation of solution realizations. The new method is illustrated with numerical examples and compared with a classical Monte Carlo sampling approach.

1. Introduction

This work is concerned with structural topology optimization subject to a high-dimensional parametric state equation. The general goal is to determine an optimal distribution of a limited amount of material in some design domain such that an objective functional is minimized. Known quantities for such a setting are usually the data of the underlying PDE, e.g. applied loads and material properties, boundary (support) conditions, the domain of the problem and possibly restrictions such as prescribed holes or solid areas. Such an optimization problem already has to be considered quite challenging since the shape and the connectivity (i.e. the topology) of the optimal structure is not known a priori and the required computations can become quite involved, in particular in 3D. For real-world engineering applications, however, the data of the problem usually is inherently uncertain, for instance due to deviations in a production process (material perturbations) or due to the random nature of some phenomenon such as wind forces acting on the structure. With these assumptions, carrying out the optimization for only the most likely events, i.e. especially the expectation of the random data, would lead to structures which are not robust to events occuring more rarely. However, it usually is these rare events which have to be considered in order to avoid unforseen damage or the brake down of structures, i.e. failure of the engineering system. Hence, in order to setup an optimization problem which is closer to realistic requirements, we extend the optimization functional by some risk measure, namely the conditional value-at-risk (CVaR), which is common in financial mathematics to implement risk-averse investment strategies. Based on the concept of the value-at-risk (VaR), which is defined as the β-quantile of a random variable for a specified probability level $0 < \beta < 1$, the CVaR is the expected value of the β-tail distribution. Hence, for large β, low probability events, which may become critical for the structure from an engineering point of view, are emphasized during the minimization procedure.

The uncertainty of the model data is specified by random fields, described by the common (truncated) Karhunen-Loève expansion with finitely many terms. By this, the state

equation depends on a finite set of parameters (or random variables) and the complexity for the computation of the stochastic solution grows significantly. This consequently also introduces a parameter dependence of the objective functional on the same set of (random) parameters as part of the CVaR formalism. A straightforward approach to obtain statistical information about the model problem (to be used subsequently in the minimization procedure) is to employ a sampling technique such as the classical Monte Carlo method. While this is simple to implement and test and hence is chosen frequently as a reference method as in our case, the slow convergenc of $O(N^{-1/2})$ with N samples has lead to the development of more advanced numerical methods which in many cases may converge at a (sometimes much) faster rate. A guiding idea is to exploit additional properties such as regularity (in the parameters) and sparsity of the stochastic solution manifold. With this, a functional representation of the stochastic solution can be computed which then allows for an efficient evaluation of statistics and arbitrary samples. We propose to exploit the low-rank structure of the parametric state equation by using hierarchical tensor representations. These formats have only recently attracted the attention in the numerical community, in particular for parametric PDEs, although they have been known for several decades e.g. in physics and chemistry. In principle, the representation is based on a singular value decomposition of a higher order tensor representing the coefficients of some tensor operator or vector. A truncation of this decomposition leads to a compression of the representation and the approach can also be understood as a hierarchical subspace approximation. We introduce the notion of this central tool in quite some depth and in particular describe our approach of the construction of such a tensor representation, in particular the required algebraic solver. While there are different ways to obtain a tensor of the stochastic solution, e.g. by discretizing and solving a tensor equation system of the parametric PDE, we suggest an approach which is based on basically the same information as the Monte Carlo method, namely a number of realizations of the solution at certain parameter points. This so called tensor reconstruction provides a functional approximation of the entire solution manifold with a comparatively small number of samples, thus providing much more information than what is obtained by Monte Carlo sampling.

For the complete functional representation, a basis of global orthonormal polynomials is used in the parameter space. The physical space is discretized with first order conforming finite elements. As a versatile representation of the topology, we employ a phase field approach which allows for topological changes and the nucleation of new holes in the structure. In order to speed up the computations, we describe a heuristic adaptive method which iteratively refines the computational mesh at the formed interfaces based on some simple error indicator.

The structure of this paper is as follows: In Section 2, the deterministic and the stochastic settings are introduced, including the minimization functional and the state equations. With the introduction of uncertainties in the linearized elasticity equation, the CVaR risk measure enters the optimization formulation. Moreover, the adaptive discretization with finite elements is described briefly. Section 3 scrutinizes hierarchical low rank tensor formats as an adequate representation for high-dimensional parametric problems. Since the use of these formats in numerical methods for PDEs is rather new and not widely

known, a detailed introduction is given. This eventuallt leads to the treatment of the actual problem at hand, namely the reconstruction of a compressed tensor for the solution manifold of a parametric PDE via the evaluation of pointwise solutions in parameter space. Here, also the functional representation of the parameter dependency is described. Moreover, the solution algorithm employed for the tensor reconstruction is presented. In Section 4, numerical examples illustrate the performance of the suggested approach. We show computations for the deterministic and the stochastic case. The latter is computed with Monte Carlo sampling as well as with the tensor reconstruction.

2. Setting

This section is concerned with the introduction of the state equation and the objective functional used in the minimization. First, we describe the deterministic setting, which is the basis for the extension to the parametrization subject to stochastic data. The model used throughout is the vector-valued PDE of linearized elasticity depending (amongst other data) on some material tensor. Second, for the stochastic model we introduce randomness to the material tensor as well as the loading. This setting is relevant for practical considerations, since the data is in fact usually inherently uncertain in reality. In order to not only optimize with respect to the expected value of the state equation, which would not be robust to less likely events, we introduce the Conditional Value at Risk (CVaR) as a risk measure in the objective functional. Third, the discretization of the state equation with finite elements (FE) and the optimization procedure is explained. For more efficient calculations, a heuristic adaptive algorithm is described for the FE discretization.

2.1. **Deterministic Formulation.**

We describe the distribution of material in the domain D by a phase field φ with $0 \leq \varphi(x) \leq 1$ for all $x \in D$. The parts of the domain with $\varphi \equiv 0$ are considered empty or void of material whereas $\varphi \equiv 1$ determines the parts which are solid. Since the phase field is a smooth approximation by construction, for practical reasons we allow an diffusive zone with $0 < \varphi < 1$ for the material.

The state equation is described by the standard linear elasticity model with the strain displacement $\mathcal{E}(u) = \operatorname{sym}(\nabla u)$ as follows,

$$\begin{aligned} -\operatorname{div}\left[\mathcal{C}(\varphi)\mathcal{E}(u)\right] &= 0 && \text{in } D, \\ u &= 0 && \text{on } \Gamma_D, \\ \left[\mathcal{C}(\varphi)\mathcal{E}(u)\right] &= g && \text{on } \Gamma_g, \\ u \cdot n &= 0 && \text{on } \Gamma_s, \\ \left[\mathcal{C}(\varphi)\mathcal{E}(u)\right] n &= 0 && \text{on } \Gamma_0 = \partial D \setminus (\Gamma_D \cup \Gamma_g \cup \Gamma_s). \end{aligned} \tag{2.1}$$

The material is fixed at the Γ_D domain boundary part and the load g is applied at Γ_g. On the boundary Γ_s the material is only bared from movement in the normal direction. This is the case for roller bearings or slip conditions. For the definition of the elasticity tensor we define the blend functional $w(\varphi) = (\varphi^3)_+ = \max\{\varphi^3, 0\}$ for a smooth transition between the phases.We then define the material tensor in the whole interval $[0, 1]$ by

$$\mathcal{C} = C_{\mathsf{mat}} w(\varphi) + C_{\mathsf{void}} w(1 - \varphi),$$

where we choose the isotropic material tensor for the solid phase to be $C_{\text{mat}} = 2\mu_{\text{mat}} + \lambda_{\text{mat}} \operatorname{tr} I$ with Lamé coefficients λ_{mat} and μ_{mat}. For the void we define the tensor as a small fraction of the material phase tensor to ensure solvability of the state equation in the whole domain. With the scaling factor $\varepsilon > 0$, we define $\mathcal{C}_{\text{void}} = \varepsilon^2 \mathcal{C}_{\text{mat}}$.

The Landau functional is given by

$$E^\varepsilon(\varphi) = \int_D \frac{\varepsilon}{2} |\nabla \varphi|^2 + \frac{1}{2} \psi_0(\varphi) \, \mathrm{d}x. \tag{2.2}$$

We choose the double well functional $\psi_0(\varphi) = (\varphi - \varphi^2)^2$ and aim to minimize the functional with the parameter $\gamma > 0$,

$$J^\varepsilon(\varphi) = \int_{\Gamma_g} g \cdot u(\varphi) \, \mathrm{d}s + \gamma E^\varepsilon(\varphi). \tag{2.3}$$

Without further restrictions the minimum of the above functional is trivial. Hence, we introduce the cost limiting volume constraint $\int_D \varphi = m|D|$. The weak formulation for the state equation (2.1) reads: Find $u \in H^1_D(D; \mathbb{R}^d)$ such that

$$\int_D E(u) : \mathcal{C}(\varphi) \mathcal{E}(v_u) \, \mathrm{d}x = \int_{\Gamma_g} g \cdot v_u \, \mathrm{d}s \qquad \text{for all } v_u \in H^1_0(D; \mathbb{R}^d). \tag{2.4}$$

After these definitions we can formulate the minimization problem at hand as follows,

$$\begin{gathered} \text{minimize } J^\varepsilon(\varphi) \text{ over } \varphi \in H^1_D(D) \\ \text{s.t. (2.4) holds, } 0 \le \varphi(x) \le 1 \text{ for all } x \in D, \text{ and } \int_D \varphi \, \mathrm{d}x = m|D|. \end{gathered} \tag{2.5}$$

Our goal is to apply the Allen–Cahn gradient flow for the steepest descent method. Therefore, the gradient $D_\varphi J^\varepsilon$ needs to be approximated which can become very expensive to compute directly as it involves the partial derivatives of the displacement u with respect to φ. This computation requires the solution of the state equation in every direction v_φ which is prohibitively expensive for sufficiently fine finite element discretizations of the phase field φ. In order to remedy this problem, we introduce the Lagrange parameter λ for the functional $M = \int_D (\varphi - m) \, \mathrm{d}x = 0$, the parameter $p \in H^1_D(D; \mathbb{R}^d)$ and the solution operator for the state equation

$$S = \int_D E(u) : \mathcal{C}(\varphi) \mathcal{E}(v_u) \, \mathrm{d}x - \int_{\Gamma_g} g \cdot v_u \, \mathrm{d}s = 0 \tag{2.6}$$

which gives the identity

$$J^\varepsilon = \Lambda \coloneqq J^\varepsilon + \lambda M + p^T S \tag{2.7}$$

and hence $D_\varphi J^\varepsilon = D_\varphi \Lambda$. We can thus reformulate the gradient as

$$\begin{aligned} D_\varphi \Lambda &= \frac{\partial J^\varepsilon}{\partial \varphi} + \frac{\partial J^\varepsilon}{\partial u} \frac{\partial u}{\partial \varphi} + \lambda \frac{\partial M}{\partial \varphi} + \frac{\partial p^T}{\partial \varphi} S + p^T \left(\frac{\partial S}{\partial u} \frac{\partial u}{\partial \varphi} + \frac{\partial S}{\partial \varphi} \right) \\ &= \frac{\partial J^\varepsilon}{\partial \varphi} + \left(\frac{\partial J^\varepsilon}{\partial u} + p^T \frac{\partial S}{\partial u} \right) \frac{\partial u}{\partial \varphi} + \lambda \frac{\partial M}{\partial \varphi} + \frac{\partial S}{\partial \varphi} p^T \\ &= \frac{\partial J^\varepsilon}{\partial \varphi} + \lambda \frac{\partial M}{\partial \varphi} + \frac{\partial S}{\partial \varphi} p^T \qquad \text{with} \qquad \frac{\partial J^\varepsilon}{\partial u} = -p^T \frac{\partial S}{\partial u}. \end{aligned} \tag{2.8}$$

The last condition for the parameter p is the adjoint equation and p is set as its unique solution, therefore called the adjoint (solution). For J^ε the adjoint problem is formulated as: Find $p \in H^1_D(D;\mathbb{R}^d)$ such that

$$\int_D \mathcal{E}(p) : \mathcal{C}(\varphi)\mathcal{E}(v_p)\ \mathrm{d}x = \int_{\Gamma_g} g \cdot v_p\ \mathrm{d}s \qquad \text{for all } v_p \in H^1_0(D;\mathbb{R}^d),$$

which is identical to the state equation. It thus holds the equivalence $p = u$. For linear state equations the adjoint can always be represented as a linear combination of the state solution. With the solution of the adjoint equation p we are able to compute the gradient step

$$\left(\varepsilon\tau^{-1}\left(\varphi - \varphi_n\right), 0\right)^T = -\left(D_\varphi \Lambda, D_\lambda \Lambda\right)^T, \tag{2.9}$$

according to (2.8) as the unique solution $(\varphi, \lambda) \in H^1_D(D) \times \mathbb{R}$ such that, for all $(v_\varphi, v_\lambda) \in H^1_0(D) \times \mathbb{R}$,

$$\left.\begin{aligned} &\frac{\varepsilon}{\tau}\int_D (\varphi - \varphi_n)\, v_\varphi\ \mathrm{d}x + \varepsilon\gamma \int_D \nabla\varphi \cdot \nabla v_\varphi\ \mathrm{d}x + \frac{\gamma}{\varepsilon}\int_D \frac{\partial \psi_0}{\partial \varphi}(\varphi_n) v_\varphi\ \mathrm{d}x \\ &\quad - \int_D \mathcal{E}(p) : \frac{\partial \mathcal{C}}{\partial \varphi} v_\varphi \mathcal{E}(u)\ \mathrm{d}x + \int_D (\varphi - m) v_\lambda\ \mathrm{d}x + \int_D v_\varphi \lambda\ \mathrm{d}x \end{aligned}\right\} = 0.$$

2.2. **Parametric Formulation.** In the stochastic setting, material properties and load scenarios are considered to be random. Consequently, the solution of the state equation as well as the solution of the adjoint equation become random variables. As a result, the gradient step itself is in fact a random distribution if one considers J^ε from (2.3). Another approach is to employ a risk measure of the functional instead, which then is minimized. A common choice is the conditional value at risk (CVaR) which gives the risk connected to the β-tail quantile for the given probability β. First, we define the value at risk (VaR). For some random variable X and the probability β it reads

$$\mathbf{VaR}_\beta[X] = \inf\left\{t \in \mathbb{R} \mid \mathbf{P}[X \le t] \ge \beta\right\}. \tag{2.10}$$

The infimum value t of (2.10) can be interpreted as the value of X at the β quantile. Now we can define the CVaR for X with the probability β as

$$\mathbf{CVaR}_\beta[X] = \mathbf{E}[X \mid X > \mathbf{VaR}_\beta[X]]. \tag{2.11}$$

The CVaR describes the expectation of the events considered as failure, as they are beyond the chosen admissible value t associated with the probability β. It is the expected value of the β tail quantile. These are all events which lie outside the β-quantile of the distribution. Practically speaking, it represents the expected loss in the unaccounted cases which lie outside of the safe (admissible) events with probability β. Minimizing this expected failure in the worst case scenarios leads to risk averse optimization. We define the risk aware functional for the probability β by

$$J^\varepsilon_\beta(\varphi) = \mathbf{CVaR}_\beta\left[\int_{\Gamma_g} g \cdot u(\varphi)\ \mathrm{d}s\right] + \gamma E^\varepsilon(\varphi). \tag{2.12}$$

Note that for $\beta = 0$ we have

$$J^\varepsilon_\beta = \mathbf{E}\left[\int_{\Gamma_g} g \cdot u\ \mathrm{d}s\right] + \gamma E^\varepsilon(\varphi). \tag{2.13}$$

We now introduce the optimization problem

$$\begin{aligned} &\text{minimize } J_\beta^\varepsilon(\varphi) \text{ over } \varphi \in H_D^1(D) \\ &\text{s.t. (2.4) holds a.s., } 0 \le \varphi(x) \le 1 \text{ for all } x \in D, \text{ and } \int_D \varphi \,\mathrm{d}x = m|D|. \end{aligned} \tag{2.14}$$

For the computation of the conditional value at risk we need to know the value at risk t associated with the probability β, which is not known in advance. However, the functional J_β^ε can be augmented in such a way that the minimum is sought over φ and t simultaneously as it holds

$$\min_\varphi J_\beta^\varepsilon = \min_{\varphi,t} \left(t + \frac{1}{1-\beta} \mathbf{E}\left[\left(\int_{\Gamma_g} g \cdot u \,\mathrm{d}s - t \right)_+ \right] + \gamma E^\varepsilon(\varphi) \right).$$

Similar to (2.8) we arrive at the adjoint problem to seek $p \in H_D^1(D;\mathbb{R}^d)$ which fulfills, for all $v_p \in H_0^1(D;\mathbb{R}^d)$,

$$\int_D \mathcal{E}(p) : \mathcal{C}(\varphi)\mathcal{E}(v_p) \,\mathrm{d}x = \begin{cases} 0 & \text{for } \int_{\Gamma_g} g \cdot u \,\mathrm{d}s - t \le 0, \\ \int_{\Gamma_g} (1-\beta)^{-1} g \cdot v_p \,\mathrm{d}s & \text{else.} \end{cases}$$

Hence, the unique solution of the adjoint equation is stochastic and can be represented by the random variable

$$p = \begin{cases} 0 & \text{for } \int_{\Gamma_g} g \cdot u \,\mathrm{d}s - t \le 0, \\ (1-\beta)^{-1} u & \text{else.} \end{cases}$$

Analogously, the gradient descent in φ and t is now given for step-widths τ_φ and τ_t by $\left(\varepsilon\tau_\varphi^{-1}(\varphi-\varphi_n), 0, \tau_t^{-1}(t-t_n)\right)^T = -(D_\varphi \Lambda, D_\lambda \Lambda, D_t \Lambda)^T$ which leads to the gradient equation. The update is the unique solution $(\varphi, \lambda, t) \in H_D^1(D) \times \mathbb{R} \times \mathbb{R}$ such that it holds for all $(v_\varphi, v_\lambda, v_t) \in H_0^1(D) \times \mathbb{R} \times \mathbb{R}$

$$0 = \begin{cases} \dfrac{\varepsilon}{\tau_\varphi} \displaystyle\int_D (\varphi - \varphi_n) v_\varphi \,\mathrm{d}x + \frac{\varepsilon}{\tau_t} \int_D (t - t_n) v_t \,\mathrm{d}x \\ + \varepsilon\gamma \displaystyle\int_D \nabla\varphi \cdot \nabla v_\varphi \,\mathrm{d}x + \frac{\gamma}{\varepsilon} \int_D \frac{\partial \psi_0}{\partial \varphi}(\varphi_n) v_\varphi \,\mathrm{d}x - \int_D \mathcal{E}(p) : \frac{\partial \mathcal{C}}{\partial \varphi} v_\varphi \mathcal{E}(u) \,\mathrm{d}x \\ + \displaystyle\int_D (\varphi - m) v_\ell \,\mathrm{d}x + \int_D v_\varphi \lambda \,\mathrm{d}x \\ + \begin{cases} \int_d v_t \,\mathrm{d}x & \text{for } \int_{\Gamma_g} g \cdot u \,\mathrm{d}s - t \le 0 \\ \int_D \left(1 - \frac{1}{1-\beta}\right) v_t \,\mathrm{d}x & \text{else.} \end{cases} \end{cases} \tag{2.15}$$

Note, that only for the initial value set to $t = 0$ we get the identical gradient descent as in the case for the expected value instead of the CVaR in (2.12).

2.3. **Discretization and Adaptivity.** This section is concerned with the introduction of the finite element (FE) discretization of the state, adjoint and gradient equations.

We assume that the design domain D is polygonal convex and is represented by a mesh $\mathcal{T}_\ell$ consisting of triangles T where ℓ is the iteration step of the optimization. The mesh depends on the iteration step ℓ since we use some mesh adaptivity to resolve the interfaces. For the current mesh, we denote the set of edges by $\mathcal{E}$, the restriction to

the edges of an element $T \in \mathcal{T}_\ell$ by $\mathcal{E}(T)$ and the element diameter by h_T. Moreover, the outer normal on some edge $E \in \mathcal{E}$ is denoted by n_E. We employ a conforming P1 FE discretization with the discrete space $V_h := C(\overline{D}) \cap P_1(\mathcal{T})$ with $P_1(\mathcal{T}) := \{v : v|_T \text{ is polynomial of degree 1 on every } T \in \mathcal{T}_\ell\}$.

In the deterministic setting, discretizations in space and time have to be carried out. Adaptive methods can be employed to reduce the computational cost. We discuss a straightforward heuristical approach which already makes a significant difference in the following.

The finite element discretization needs to be of sufficient quality such that the interface region in the phase field φ and the solution of the state equation u are well represented. This usually is achieved by suitable mesh refinement. We utilize the following heuristical error indicator

$$\eta_{\varphi,T}^2 := \sum_{E \in \mathcal{E}(T)} h_T \left\| \nabla \varphi \cdot n_E \right\|_{L^2(E)}^2, \qquad \eta_{u,T}^2 = \sum_{E \in \mathcal{E}(T)} h_T \left\| \nabla \, \mathbf{E}[u] \, \varphi \cdot n_E \right\|_{L^2(E)}.$$

With $\eta_\varphi^2 = \sum_{T \in \mathcal{T}} \eta_{\varphi,T}^2$ and $\eta_u^2 = \sum_{T \in \mathcal{T}} \eta_{u,T}^2$, we derive the combined heuristical error indicator

$$\eta_T = \eta_{\varphi,T}/\eta_\varphi + \eta_{u,T}/\eta_u. \tag{2.16}$$

In $\eta_{u,T}$ the weight φ is introduced because we are not interested in the displacement of the void domain but only want to focus on the interface and the solid regions. Because of the local constraints the gradient in $\eta_{\varphi,T}$ roughly represents the interface of φ, where the material is not in either of the pure phases with $\varphi = 0$ or $\varphi = 1$. In the regions where φ is in a pure phase, it is also constant. Hence, a coarse mesh suffices for the approximation. With the refinement fraction parameter $0 < \vartheta \leq 1$ we use the bulk criterion to chose the minimal set $\mathcal{M} \subseteq \mathcal{T}_\ell$ with

$$\sum_{T \in \mathcal{M}} \eta_T^2 \geq \vartheta \sum_{T \in \mathcal{T}} \eta_T^2 \tag{2.17}$$

as the elements for refinement to construct the next mesh level $\mathcal{T}_{\ell+1}$ with the well known Dörfler marking [10]. In practice the interface do move during the optimization. So instead of refining the last mesh, we start in every refinement with the initial mesh interpolate the current solution and displacement onto that mesh, refine according to the associated indicators and interpolate the current solution onto the finer mesh. We repeat this process until we have a mesh $\mathcal{T}_{\ell+1}$, that is suitably finer than $\mathcal{T}_\ell$.

For the adaptation of the pseudo-time step τ we apply an evolutionary proportional integral derivative controller (ePID) based on the relative changes $e_n = \|\varphi_{n+1} - \varphi_n\| \, \|\varphi_{n+1}\|^{-1}$ given by the update

$$\tau_{n+1} = \left(\frac{e_{n-1}}{e_n} \right)^{k_P} \left(\frac{\mathsf{TOL}}{e_n} \right)^{k_I} \left(\frac{e_{n-1}^2}{e_n e_{n-2}} \right)^{k_D} \tau_n. \tag{2.18}$$

The experience presented in [18] suggests $k_P = 0.075, k_I = 0.175$, and $k_D = 0.01$. Furthermore we limit the change rate in each step to $0.5 \leq \tau_{n+1} \tau_n^{-1} \leq 2$ and $\tau_{\min} \leq \tau_{n+1} \leq \tau_{\max}$ with $\tau_{\min} = 10^{-12}$ and $\tau_{\max} = 10^{-5}$ unless stated otherwise.

Finally we would like to avoid non physical or highly porous (i.e. non connected or oscillating) topologies. This is prevented by the interface functional in (2.3) weighted

Algorithm 1: Discrete optimization

Input: mesh $\mathcal{T}_0$, initial values φ_0, τ_0
for $\ell = 0, 1, \ldots$ *until converged* **do**
  for $n = 0, 1, \ldots$ *until converged* **do**
    solve state equation on mesh $\mathcal{T}_\ell \Rightarrow \quad u_n$
    solve adjoint equation on mesh $\mathcal{T}_\ell \Rightarrow \quad p_n$
    solve gradient equation on mesh $\mathcal{T}_\ell \Rightarrow \quad \varphi_{n+1}^*$
    project φ_{n+1}^* to $[0,1] \Rightarrow \quad \varphi_{n+1}$
    adapt τ according to (2.18) $\Rightarrow \quad \tau_{n+1}$
    adapt γ according to (2.19) $\Rightarrow \quad \gamma_{n+1}$
  adapt mesh according to (2.16) and (2.17) $\Rightarrow \quad \mathcal{T}_{\ell+1}$
  interpolate φ_n onto $\mathcal{T}_{\ell+1}$ as new φ_0

by γ. We choose γ such that the two contributions in the functional are balanced with respect to some factor

$$\gamma = c_J E^\varepsilon(\varphi)^{-1} \int_{\Gamma_g} g \cdot u \, \mathrm{d}s. \tag{2.19}$$

In the numerical computations we set $c_J = 0.1$ if not given otherwise.

The combination of the above heuristics leads to the discrete optimization method depicted in Algorithm 1. The initial mesh $\mathcal{T}_0$ has to be sufficiently fine to represent the final topology of the optimized shape adequately. We then solve the discrete optimization problem on that mesh, refine according to the described heuristics and iterate the progress. The initial phase field on the new mesh is set to be the interpolation of the minimum on the previous mesh. The procedure is formally depicted in Algorithm 1.

The stochastic setting requires the computation of the expected value of the gradient step in each iteration. We first apply Monte Carlo estimators for this step and latter the tensor reconstruction from Section 3. In each iteration we generate N samples of the random input data and solve the state equation, the adjoint equation as well as the gradient equation for each of them. Then we compute the mean of the next step as the updated form φ_{n+1}. The computation of the samples is parallelized.

3. Hierarchical tensor approximations

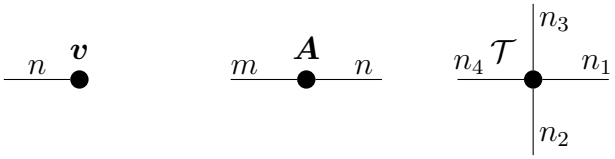
As an alternative to classical Monte Carlo sampling as described previously, it often can be beneficial to get a representaton of the entire parametric solution manifold of the state equation for a more accurate approximation of the expectation value of φ_{n+1} or of other functionals depending on the solution. Assuming sufficient sparsity of the function at hand, a promising approach is based on low rank tensor tensor representations for which we discuss modern reconstruction techniques. The basis for such an approach is the same as for Monte Carlo sampling, namely the pointwise (in parameter space) evaluation of the state equation. Analogously to the well established methods of matrix completion and matrix reconstruction, tensor reconstruction aims at reconstructing a higher order tensor from highly incomplete measurements using a low rank assumption.

Algorithm 2: Discrete stochastic optimization

Input: mesh $\mathcal{T}_0$, initial values φ_0, τ_0
for $\ell = 0, 1, \ldots$ *until converged* **do**
  for $n = 0, 1, \ldots$ *until converged* **do**
    for $i = 1, \ldots, N$ **do**
      sample random input data from $\omega_i \in \Omega$
      solve state equation on mesh $\mathcal{T}_\ell \Rightarrow \quad u_n(\omega_i)$
      solve adjoint equation on mesh $\mathcal{T}_\ell \Rightarrow \quad p_n(\omega_i)$
      solve gradient equation on mesh $\mathcal{T}_\ell \Rightarrow \quad \varphi^*_{n+1}(\omega_i)$
    compute the mean $\bar{\varphi}_{n+1} = \frac{1}{N} \sum_{i=1}^{N} \varphi^*_{n+1}(\omega_i)$
    project $\bar{\varphi}_{n+1}$ to $[0, 1] \Rightarrow \quad \varphi_{n+1}$
    adapt τ according to (2.18) $\Rightarrow \quad \tau_{n+1}$
    adapt γ according to (2.19) $\Rightarrow \quad \gamma_{n+1}$
  adapt mesh according to (2.16) and (2.17) $\Rightarrow \quad \mathcal{T}_{\ell+1}$
  interpolate φ_n onto $\mathcal{T}_{\ell+1}$ as new φ_0

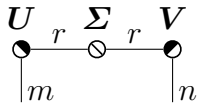
In contrast to matrices there are several possible definitions of the rank of a higher order tensor, e.g. the canonical rank, the Tucker rank and the Hierarchical Tucker (HT) rank and its important special case the Tensor-Train (TT) rank, see for example [13]. In this work we use the Hierarchical Tucker [14] and in particular the Tensor-Train (TT) format [21] as it provides several unique advantages. Most importantly the TT-format allows a linear scaling with respect to the tensors order for the storage requirements and common operations for tensors of fixed rank.

3.1. **Notations.** For the course of this work we regard higher order tensors simply as d-dimensional real arrays. To denote the contraction of two tensors along certain nodes we use the $\circ$ symbol indexed with by the nodes that are contracted, e.g. $\mathcal{A} \circ_{(i,j),(k,l)} \mathcal{B}$ means that the i-th and k-th mode of $\mathcal{A}$ and $\mathcal{B}$ are contracted, as well as the j-th and l-th mode. If no indices are given, a contraction of the last mode of the left operand and the first mode of the right operand is assumed. As writing this for larger tensor expressions quickly becomes cumbersome we also use a diagrammatic notation to visualize them. In this notation a tensor is depicted as a dot with edges corresponding to each of its modes. If appropriate the cardinality of the corresponding index set is given as well. From left to right the following shows this for an order 1 tensor (vector) $\boldsymbol{v} \in \mathbb{R}^n$, an order 2 tensor (matrix) $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and an order 4 tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$.

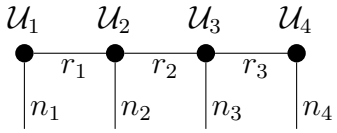


If a contraction is performed between two modes of two tensors the corresponding edges are joined. The following shows this exemplarily for the inner product of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ and a matrix-vector product with $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{v} \in \mathbb{R}^n$.

There are two special cases concerning orthogonal and diagonal matrices. If a specific matrification of a tensor yields an orthogonal or diagonal matrix, the tensor is depicted by a half filled circe (orthogonal) or a circle with a diagonal bar (diagonal). The half filling and the diagonal bar both divide the circle in two halves. The edges joined to either half, correspond to the mode sets of the matrification, which yields the orthogonal or diagonal matrix. As an example the diagrammatic notation can be used to depict the singular value decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with rank r, as shown in the following.



3.2. **The Tensor Train Format.** The idea of the TT-format is to separate the modes of a tensor into two order two tensors and $d-2$ order three tensors. This results in a tensor network, that is exemplarily shown for an order four tensor $\mathcal{T} = \mathcal{U}_1 \circ \mathcal{U}_2 \circ \mathcal{U}_3 \circ \mathcal{U}_4$ in the following diagram.



Formally the tensor train format can be defined as follows.

Definition 3.1 (Tensor-Train Format)**.** Let $\mathcal{T} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ be a tensor of order d. A factorization

$$\mathcal{T} = \mathcal{U}_1 \circ \mathcal{U}_2 \circ \ldots \circ \mathcal{U}_{d-1} \circ \mathcal{U}_d \ , \tag{3.1}$$

of $\mathcal{T}$, in component tensors $\mathcal{U}_1 \in \mathbb{R}^{n_1 \times r_1}$, $\mathcal{U}_i \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}, i = 2, \ldots, d-1$ and $\mathcal{U}_d \in \mathbb{R}^{r_{d-1}, n_d}$, is called a Tensor-Train (TT) representation of $\mathcal{T}$. The tuple of the dimensions $(r_1, \ldots, r_{d-1})$ of the component tensors is called the representation rank and is associated with the specific representation. In contrast the tensor train rank (TT-rank) is defined as the minimal rank tuple $\boldsymbol{r} = (r_1, \ldots, r_d)$ such that there exists a TT representation of $\mathcal{T}$ with representation rank equal to $\boldsymbol{r}$.

Proposition 3.2 ([21])**.** *Every tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ admits a TT-representation with minimal representation rank $\boldsymbol{r} = (r_1, \ldots, r_{d-1})$. Furthermore this minimal rank $\boldsymbol{r}$, which thereby is the TT-rank of $\mathcal{T}$, is related to the ranks of certain matrification of $\mathcal{T}$ via*

$$r_i = \textit{rank}\left(\hat{M}_{(1,2,\ldots,i)}\left(\mathcal{T}\right)\right) \ , \tag{3.2}$$

where $\hat{M}_{(1,2,\ldots,i)}\left(\mathcal{T}\right)$ denotes the matrification of $\mathcal{T}$ which combines the first i modes and the remaining $d-i$ modes into a single mode each.

Such a TT-representation with minimal representation rank can be obtained by successive singular value decompositions, called Higher order SVD or TT-SVD, as explained in detail in [21]. Storing a tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ with TT-rank $\boldsymbol{r} = (r_1, \ldots, r_{d-1})$ in a minimal TT representation requires $\Theta(dnr^2)$ memory for the component tensors, where $n = \max_i(n_i)$ and $r = \max_i(r_i)$. That is the storage requirement scales only linear in the order.

It can easily be seen that a the TT decomposition is not unique. One can "normalize" the decomposition by enforcing that all component tensors with smaller index than a given i, are left orthogonal, i.e. the matrification which combines all but the last mode is an orthogonal matrix, and all component tensors with larger index are right orthogonal, i.e. the matrification which combines all but the first mode is an orthogonal matrix. Note however that this is not a real normalization as the representation is still not unique. The component tensor which is not orthogonal is called the core – or more precisely is said to carry the core.

As a natural generalization of the low rank matrix manifold [2], it is shown by [16] that the set $\mathcal{M}_{\boldsymbol{r}}\left(\mathbb{R}^{n_1 \times \ldots \times n_d}\right)$ of tensors with rank $\boldsymbol{r}$ forms a manifold. In particular they prove the following theorem.

Theorem 3.3 (Tensor Train Manifold [16])**.** *For fixed dimensions $n_1, \ldots, n_d$ and a fixed rank $\boldsymbol{r} = (r_1, \ldots, r_{d-1})$ the set $\mathcal{M}_{\boldsymbol{r}}\left(\mathbb{R}^{n_1 \times \ldots \times n_d}\right)$ is a smooth manifold of dimension*

$$\dim\left(T_{\boldsymbol{r}}\right) = \sum_{i=1}^{d} r_{i-1} n_i r_i - \sum_{i=1}^{d-1} r_i^2 \ , \tag{3.3}$$

where as a convention $r_0 = r_d = 1$.

The dimension of the manifold also determines the degrees of freedom a tensor with fixed TT-rank possesses. These degrees of freedom are important for the reconstruction from incomplete measurements as they are a definite low bound on the number of measurements needed.

3.3. **Reconstruction of the UQ solution.** In the parametric setting we can regard the next iterate $\varphi_{n+1}(\boldsymbol{x}, \boldsymbol{\psi})$ as a function in the physical space and the stochastic parameters. Using for example a truncated basis of Legendre polynomials, the continuous parameters ψ_i can be mapped to the discrete parameter vectors $\boldsymbol{\xi}_i$. This also yields a discrete tensor approximation $\mathcal{X}$ of $\varphi_{n+1}(\boldsymbol{x}, \boldsymbol{\psi})$. Our pursued idea is to reconstruct the full solution tensor $\mathcal{X}$ from a rather small set of solutions of (deterministic) problem samples $(\boldsymbol{\xi}_1^{(k)}, \ldots, \boldsymbol{\xi}_D^{(k)})$ with $k = 1, \ldots, N$, i.e. the solution realizations used by the Monte Carlo simulation.

In an abstract setting we can formulate the tensor reconstruction problem analogously to the well established matrix reconstruction setting, i.e.

$$\begin{aligned} &\text{minimize } \text{TT-Rank}\left(\mathcal{X}\right) \\ &\text{subject to } \hat{\mathcal{A}}\left(\mathcal{X}\right) = \boldsymbol{b} \ . \end{aligned} \tag{3.4}$$

For tensors of degree two, i.e. matrices, this is exactly the matrix reconstruction problem. In our particular case $\hat{\mathcal{A}}$ and $\boldsymbol{b}$ are given as

$$\hat{\mathcal{A}}(\mathcal{X})_{k,\boldsymbol{x}} = (\boldsymbol{\xi}_1^{(k)} \otimes \ldots \otimes \boldsymbol{\xi}_D^{(k)}) \circ_{(1,\ldots,D),(1,\ldots,D)} \mathcal{X}(\boldsymbol{x}), \tag{3.5}$$

$$\boldsymbol{b}_{k,\boldsymbol{x}} = \varphi_{n+1}(\boldsymbol{x}, \boldsymbol{\xi}_1^{(k)}, \ldots, \boldsymbol{\xi}_D^{(k)}) \ . \tag{3.6}$$

For matrices it is established that under some assumptions on the measurement operator $\mathcal{A}$ there is a unique solution to this problem, which is attained by solving the relaxed trace norm minimization problem. The important result is that p measurements are sufficient, where p scales as $r(m+n)\log(mn)$, see e.g. [8] and [23]. Unfortunately equivalent results could not yet be obtained for higher order tensors. Nevertheless there is much numerical evidence that also for higher order tensors reconstruction from incomplete measurements is possible, see e.g. [17, 22]. One main problem in transferring the results from the matrix case to low rank tensors is to find a convex relaxation for the NP-hard rank minimization problem. In this work we make an educated guess on the optimal rank of the solution, see Section 3.5, and thereby bypass this problem. If we assume that the optimal rank $\boldsymbol{r}*$ is a priori known, (3.4) can be recast to an optimization problem on the low rank manifold $\mathcal{M}_{\boldsymbol{r}*}$

$$\text{argmin}_{\mathcal{X}\in\mathcal{M}_{\boldsymbol{r}*}} |\mathcal{A}(\mathcal{X}) - \boldsymbol{b}|. \tag{3.7}$$

There are several algorithms which solve such optimization problems posed on the low rank manifold, see e.g. [4] for a recent survey. Most popular is probably the alternating least squares (ALS) and its extension the DMRG algorithm [15]. A second more recent approach are Riemmanien optimization techniques which directly exploit the manifold structure of $\mathcal{M}$, see for example [17] for an application to tensor completion. For this work we use an alternating steepest decent algorithm which can be seen as being in between those two groups of algorithms.

3.4. **Alternating Steepest Decent.** The alternating steepest descent algorithm used in this work can be seen as an extension of the alternating directional fitting (ADF) algorithm developed in [12]. Our derivation however is quite different and may provide some additional insight.

As established in 3.3, $\mathcal{M}_{\boldsymbol{r}*}$ forms a smooth embedded manifold. It is shown in [16, Theorem 4.2] that the the tangent space at a point $\mathcal{X} = \mathcal{U}_1 \circ \ldots \circ \mathcal{U}_d$ is given by

$$\mathbb{T}\mathcal{M}_{\boldsymbol{r}}(\mathcal{X}) = V_1 \oplus \ldots \oplus V_d \tag{3.8}$$

where

$$\begin{aligned} V_j = \{&\mathcal{U}_1 \circ \ldots \circ \mathcal{U}_{j-1} \circ \Delta \circ \mathcal{U}_{j+1} \circ \ldots \circ \mathcal{U}_d \\ &\mid \Delta \in \mathbb{R}^{r_{j-1}\times n_i \times r_j}, \mathcal{U}_i \circ_{(1,2),(1,2)} \Delta = \mathbf{0}\Big\}, \\ V_d = \Big\{&\mathcal{U}_1 \circ \ldots \circ \mathcal{U}_{d-1} \circ \Delta \mid \Delta \in \mathbb{R}^{r_{d-1}\times n_d \times 1}\Big\}. \end{aligned}$$

Based on this orthogonal decomposition of the tangent space [20] derive a closed form for the projector $\hat{P}_{\mathbb{T}_{\mathcal{X}}\mathcal{M}_{\boldsymbol{r}}}$ onto the tangent space $\mathbb{T}\mathcal{M}_{\boldsymbol{r}}$, which is composed by the projectors onto the spaces V_i. Using this closed form of the projection operator, one could create an Riemmanien optimization algorithm similar to the ones used by [17]. A problem however

encountered independent of the choice of the retraction is that one has no direct control on the update carried out in each iteration, which makes for example a "good" step size hard to determine. The idea of the alternating steepest descent is instead to project the gradient onto *one* of the spaces V_j at a time and perform an update. The main advantage is that in this case no retraction is needed at all because one can easily show that such an update never leaves the manifold, independent of the step size. Therefor the optimal step size of a gradient descent with respect to the global optimization problem can directly be calculated as

$$\alpha_i = \frac{-\left\langle \hat{\mathcal{A}}(\mathcal{X}_i) - \boldsymbol{b},\ \hat{\mathcal{A}}(\hat{P}_j(\mathcal{Y}))\right\rangle}{\left\|\hat{\mathcal{A}}(\hat{P}_j(\mathcal{Y}))\right\|_2^2}\ . \tag{3.9}$$

Here $\hat{P}_i$ denotes the orthogonal projectors onto the spaces V_i and $\mathcal{Y}$ denotes the update direction, i.e. the negative gradient in this case. For the heuristics behind the algorithm it is not necessary that the spaces V_i are orthogonal. In fact numerical experience has shown that better results are obtained if

$$V_j = \left\{\mathcal{U}_1 \circ \ldots \circ \mathcal{U}_{j-1} \circ \Delta \circ \mathcal{U}_{j+1} \circ \ldots \circ \mathcal{U}_d \mid \Delta \in \mathbb{R}^{r_{j-1}\times n_i \times r_j}\right\}$$

is used instead for all j, i.e. for $j < d$ each space is enlarged by the part parallel to $\mathcal{U}_j$. The individual projectors are given by

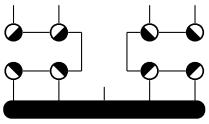
$$\begin{aligned}&\hat{P}_j(\mathcal{Y})\\ &= \mathcal{U}_{<j} \circ_{(j),(j)} \mathcal{U}_{<j} \circ_{(1,\ldots,j-1),(1,\ldots,j-1)} \mathcal{X} \circ_{(j+1,\ldots,d),(2,\ldots,d-j+1)} \mathcal{U}_{>j} \circ_{(1),(1)} \mathcal{U}_{>j},\end{aligned}$$

where

$$\mathcal{U}_{<j} = \mathcal{U}_1 \circ \mathcal{U}_2 \circ \ldots \circ \mathcal{U}_{j-1}, \tag{3.10}$$

$$\mathcal{U}_{>j} = \mathcal{U}_{j+1} \circ \mathcal{U}_{j+2} \circ \ldots \circ \mathcal{U}_d, \tag{3.11}$$

and it is assumed that the core position is at i, i.e. the core is not part of the projector. This can be expressed more intuitively by the graphical notation:

$\hat{P}_j(\mathcal{Y}) =$ 

A rough outline of the algorithm is given in listing 3. It uses a forward and a backward stack, which save intermediate results of the subsequent calculations and allow for a very efficient implementation. To be concrete, the stacks are defined as

$$\begin{aligned}F_{-1,k} = B_{d+1,k} &= \mathbf{0},\\ F_{j,k} &= F_{j-1,k} \circ \mathcal{U}_j \circ_{2,1} \boldsymbol{\xi}_j^{(k)},\\ B_{j,k} &= \boldsymbol{\xi}_j^{(k)} \circ_{1,2} \mathcal{U}_j \circ B_{j+1,k}.\end{aligned}$$

Creating the complete backward stack costs $\mathcal{O}(dnr^2N)$ and creating a single layer of the forward stack costs $\mathcal{O}(nr^2N)$. The calculation of the residual is carried out for any position j with

$$R_k = (F_{j-1,k} \circ \mathcal{U}_j \circ B_{j+1,k}) \circ \boldsymbol{\xi}_j^{(k)} - \boldsymbol{b}_k,$$

Algorithm 3: Alternating Steepest Descent for Tensor Reconstruction

Input: mesh $\mathcal{T}_0$, initial values φ_0, τ_0
for $i = 1, 2, \ldots$ *until converged* **do**
  Rebuild complete backward stack
  for $j = 0, 1, \ldots, d$ **do**
    Move core to position j
    Advance forward stack to position $j-1$
    Calculate the residual $\hat{A}(\mathcal{X}_i) - \boldsymbol{b}$
    Calculate the projected gradient $\hat{P}_j(\nabla F(\mathcal{X}_i))$
    Calculate the optimal step size according to (3.9)
    Update $\mathcal{X}_{i+1} = \mathcal{X}_i = \alpha \hat{P}_j(\nabla F(\mathcal{X}_i))$

which costs $\mathcal{O}(nr^2N)$. The projected gradient can be represented in the same TT-format as the current iterate with only the core being altered. Then, the core $\mathcal{C}$ of the projected gradient is calculated as

$$\mathcal{C} = \sum_{k=1}^{N} R_k \; F_{j-1,k} \otimes \boldsymbol{\xi}_j^{(k)} \otimes B_{j+1,k} \; .$$

This can again be done in $\mathcal{O}(nr^2N)$. At the same cost the optimal step size is calculated using

$$\alpha = \frac{\sum_{k=1}^{N} R_k (F_{j-1,k} \circ \mathcal{C} \circ B_{j+1,k}) \circ \boldsymbol{\xi}_j^{(k)})}{\sum_{k=1}^{N} ((F_{j-1,k} \circ \mathcal{C} \circ B_{j+1,k}) \circ \boldsymbol{\xi}_j^{(k)})^2} \; ,$$

which assumes that the measurements are orthonormal. While this is not true in the considered setting, this still provides a sufficiently good approximation. Finally moving the core by one position has negligible costs of $\mathcal{O}(nr^2)$. Putting all this together shows that the complete algorithm has numerical costs scaling as $\mathcal{O}(dnr^2N)$, i.e. it is linear in the degree of the tensor as well as the number of measurements.

The algorithm is stopped if after three sweeps over all positions the residuum has not significantly decreased.

3.5. **Initial Guess.** We use a perturbation based calculation to obtain an initial guess on the solution. This initial guess is used for two purposes: to obtain an estimate on the rank of the solution, as well as to get a starting point for the iteration.

Using Hermite polynomials as a basis, the solution up to first order is given by

$$\varphi(\boldsymbol{x}, \boldsymbol{\psi}) \approx \alpha(\boldsymbol{x}) + \sum_j \beta_j(\boldsymbol{x}) \, (2\psi_j) \, . \tag{3.12}$$

Let $\boldsymbol{\varepsilon}_j$ denote the vectors with the only non-zero entry ε at the j-th position. To obtain an initial guess of the solution, we use a set of fixed stochastic variables, i.e.

$$\varphi(\boldsymbol{x}, \mathbf{0}) = \alpha(\boldsymbol{x}), \tag{3.13}$$

$$\varphi(\boldsymbol{x}, \boldsymbol{\varepsilon}_j) = \varphi(\boldsymbol{x}, \mathbf{0}) + 2\varepsilon\beta_j(\boldsymbol{x}). \tag{3.14}$$

From this, the coefficients can be derived as

$$\alpha(\boldsymbol{x}) = \varphi(\boldsymbol{x}, \mathbf{0}), \tag{3.15}$$

$$\beta_j(\boldsymbol{x}) = (\varphi(\boldsymbol{x}, \boldsymbol{\varepsilon}_i) - \varphi(\boldsymbol{x}, \mathbf{0}))/2\varepsilon. \tag{3.16}$$

The coefficients correspond to single entries of the solution tensor $\mathcal{X}$, i.e

$$\mathcal{X}[0, \ldots, 0] = \alpha, \ \mathcal{X}[0, \ldots, 0, 1^{(\text{at } j)}, 0, \ldots, 0] = \beta_j. \tag{3.17}$$

Setting all other entries to zero, we obtain a sparse and thereby low-rank initial guess for the solution tensor.

3.6. **Postprocessing.** For the gradient iteration we are interested in the expectation value of φ. After the successful reconstruction of $\mathcal{X}$, this is directly available as $\mathcal{X}[0, \ldots, 0]$, i.e. without any additional evaluation costs.

4. Numerical Experiments

We illustrate the described method with some numerical examples. In particular, we

The numerical implementation is realized in Python using the very versatile `FEniCS` framework [11, 3, 19] which employs a C++ just-in-time compiler for the dynamic generation of high-performance code. For the solution of the linear systems we utilize `PETSc` [5, 6, 7] with the `UMFPACK` direct solver [9] for two dimensional problems and `gmres` for three-dimensional problems. For the parallel computation of the samples in the stochastic settings we use the `Joblib` python module [24]. The tensor algorithms are implemented in the efficient open source `C++` library `xerus` [1].

4.1. **Experiment 1.** The first experiment explores a load bearing framework. On the domain $D = [0, 2] \times [0, 1] \subset \mathbb{R}^2$ we define the boundaries $\Gamma_D = [-1, -0.9] \times \{0\}$ on which we set $u = 0$, $\Gamma_g = [-0.02, 0.02] \times \{0\}$ on which we apply the load $g = (0, -5000)^T$ and an additional slip boundary condition on $\Gamma_S = [0.9, 1.0] \times \{0\}$ on which we restrict the displacement in the normal direction of the boundary as $u \cdot n = 0$. The Lamé–coefficients λ and μ are set to 250 and the amount of material in the domain is set to $m = 0.4$. The interface parameter is set as $\varepsilon = 1/16$. Figure 1 shows the setting, the optimized sturcture, the stress, and the final mesh achieved after 258 iterations on meshes with increasing resolution with an initial distribution of $\varphi(x) = 0.5$ for all $x \in D$. The initial value does not need to fulfill the volume constraint as it is enforced in a relaxed manner with a Lagrange multiplier.

The resulting material distribution exhibits a clear structure consisting of an arc resting on the load bearing boundaries and six spokes connecting it to the load boundary. We emphasis that this structure is solely a result of the optimization process, as only a trivial uniform initial topology was given in the initial conditions. Despite the asymmetric boundary conditions the result is almost symmetrical as the load is perpendicular to the

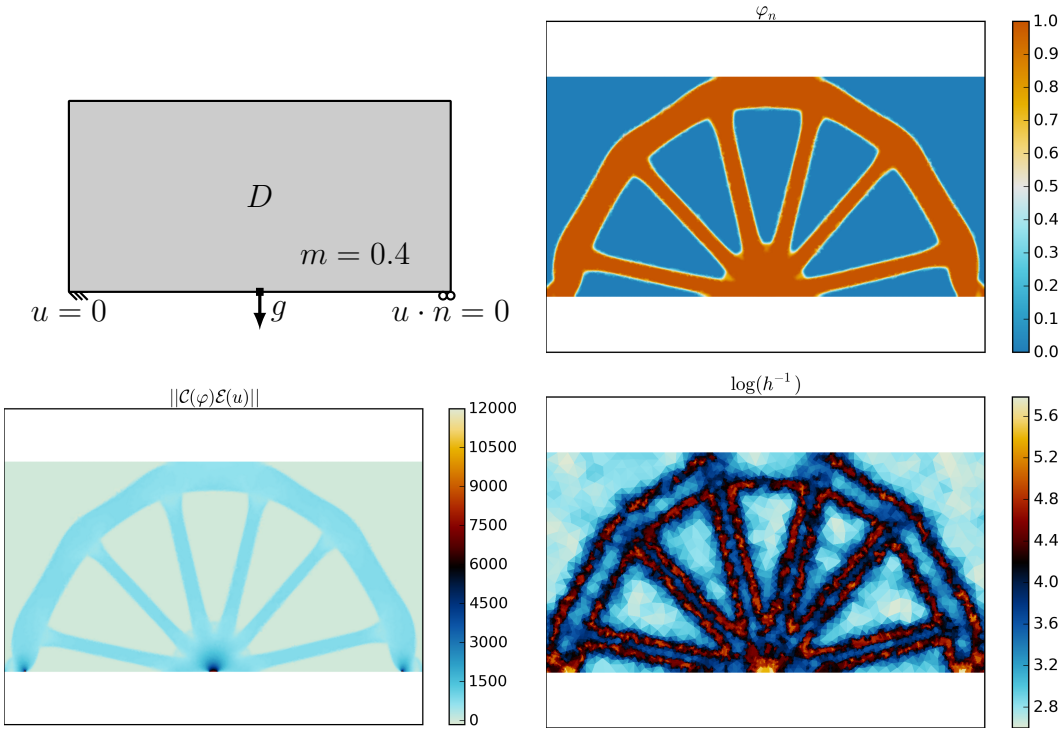


FIGURE 1. Setting, final structure, stress and mesh fineness in the deterministic setting.

load bearing boundaries. The peak stress is right at the load boundary and at the edges of the Dirichlet boundary. The rest of the material filled domain has an almost uniform stress distribution. Both the interface and the displacement are well resolved by the final mesh.

4.2. **Experiment 2.** The second experiment is defined on the domain $D = [0,5] \times [0,2.5] \times [0,3] \subset \mathbb{R}^3$ for which we set $m = 0.5$ as the target amount of material. The displacement is fixed on the boundary $\Gamma_D = \{0\} \times [0,2.5] \times [0,3]$ and a load $g = (0,0,-165)$ is applied at $\Gamma_g = [4.75,5] \times [0,2.5] \times \{0.0\}$ whereas we set $\lambda = \mu = 5000$. Additionally we alter the adaptive update of the time step τ with $\tau_{\max} = 10^{-3}$ and set a fixed $\gamma = 1.0$. A selection of the described setting and the computed structures is shown in Figure 2 for an initial $\varphi(x) = 0.5$ for all $x \in D$.

Once again, the resulting topology is not prescribed by the initial conditions yet it exhibits one hole and a thinner support beam. The peak stress is limited to the corners of the Dirichlet boundary whereas the rest of the domain has an almost uniform stress. The interfaces of φ and the displacement u are resolved well by the adaptive meshes.

4.3. **Experiment 3.** In the third experiment we modify the setting from the first experiment with stochastic input data. The Lamé–coefficients $\mu(\omega) = \lambda(\omega)$ are modeled as a truncated lognormal Karhunen–Loève expansion with 10 modes, a mean value of 150

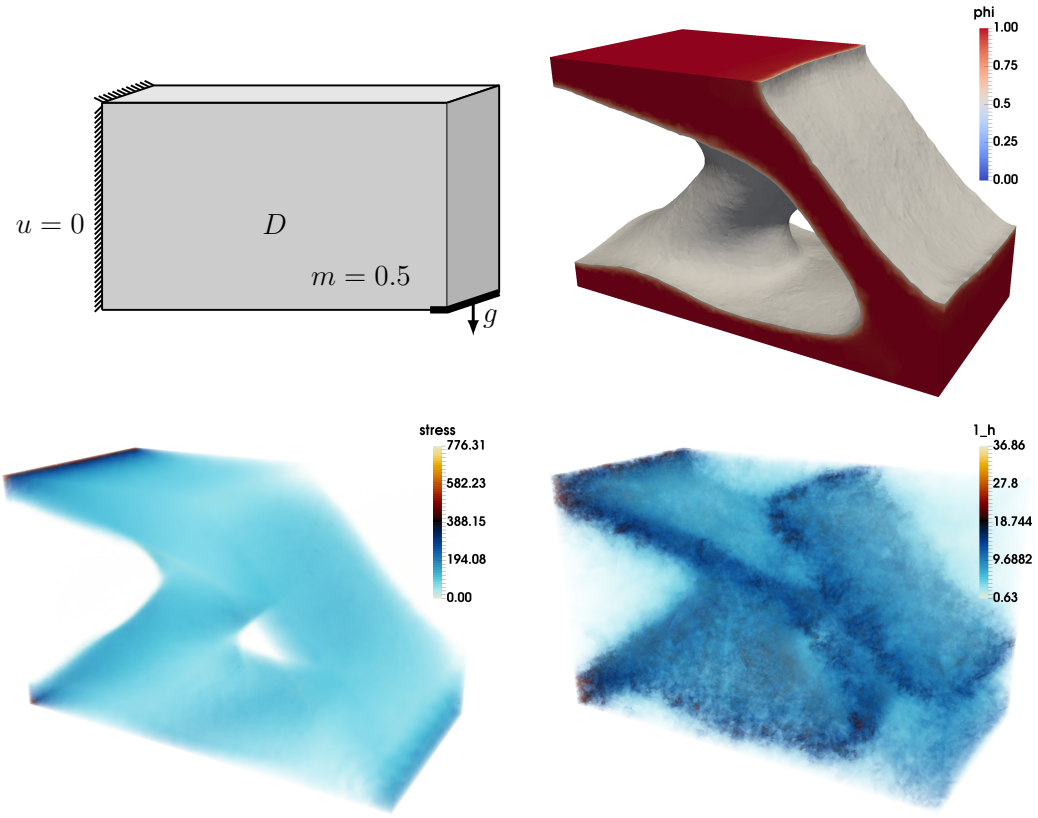


FIGURE 2. Setting and final structure cut at $\varphi \equiv 0.5$, stress and size of tetrahedrons in $1/h$ of the deterministic cantilever setting in $\mathbb{R}^3$.

and a covariance length of 0.1 which is scaled by a factor of 100. The load is assumed as a vector of length $(0, -5000)$ with a random rotation angle given through a truncated normal distribution with bounds $[-\pi/2, \pi/2]$, standard deviation 0.3 and mean 0. The setting with an example realization of the random field is depicted in Figure 3 alongside the optimized structures for different choices of β with the initial $t = 0$. In each iteration we use $N = 224$ samples which corresponds to a multiple of the numbers of processor cores.

With this choice for the initial t the optimization for $\beta = 0$ is identical with the unweighted expected value functional instead of **CVaR**. In this case we already observe the loss of the symmetry compared to the deterministic case since the load is almost always not perpendicular to the load bearing boundaries. The main difference is the side-wise strain on the Dirichlet boundary, which is introduce by the moved left most spoke. In contrast to this, the right-hand side closely resembles the deterministic case since the slip boundary cannot absorb energy in the tangential direction. Increasing the probability β will put more emphasis onto the introduced material weaknesses in the tensor $\mathcal{C}$ and onto the more extreme deflection of the load. As a result, fewer but thicker spokes form and especially the left-most spoke becomes thicker as this is the most efficient way to absorb the side-facing energies. A further increase in the risk avoiding probability leads

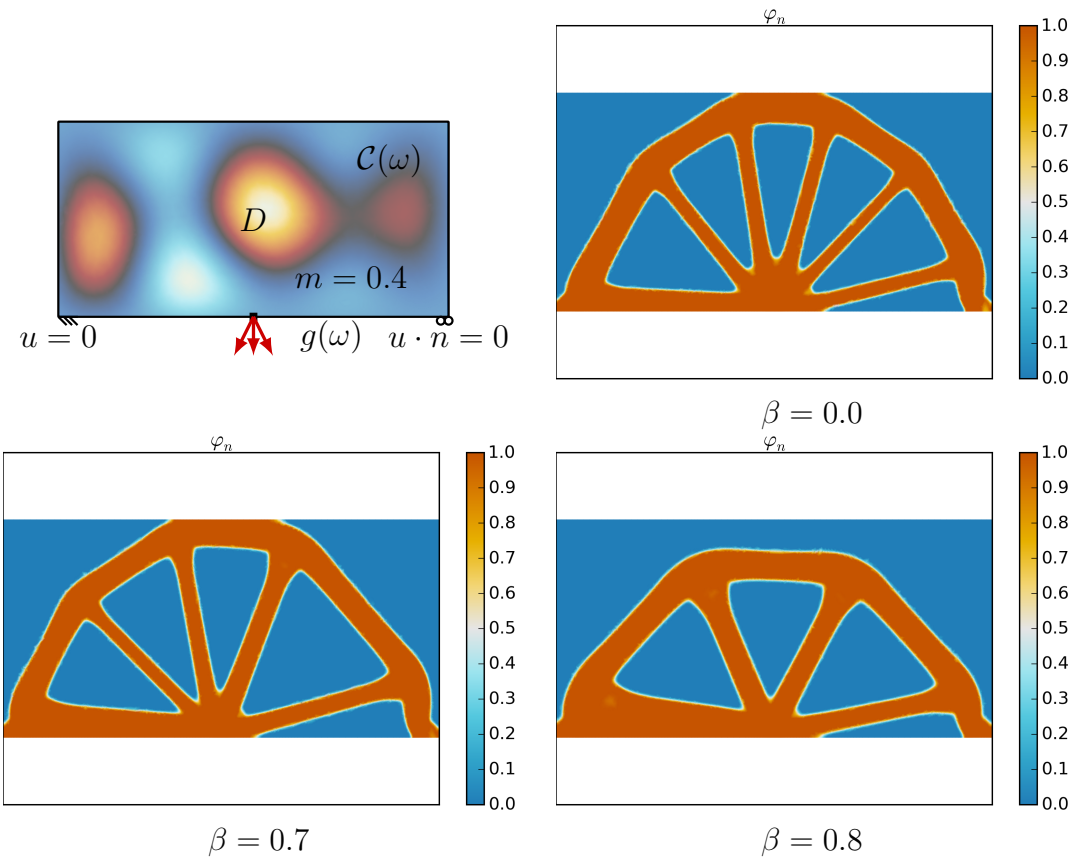


FIGURE 3. Stochastic setting and final form for various β.

to a trivial bar-like structure without any holes.

For this third experiment, in addition to Monte Carlo sampling, we also employ the tensor reconstruction algorithm described in 3. Figure 4 shows the final configuration found by the optimization for various values of β. For $\beta = 0$ and $\beta = 0.8$ the resulting general topology is the same as acquired by the Monte Carlo based optimization, although there are slight deviations in the thickness and position of the spokes. For $\beta = 0.7$ the final topology obtained using the tensor reconstruction approach resembles the one for $\beta = 0.8$, i.e. only four spokes, while the respective Monte Carlo based result exhibit five spokes. Unfortunately it is not trivial to give a comparable quantitative measure for the quality of the solutions of the two methods. Hence it is not clear which solution actually provides a more accurate result in this setting. However for both algorithm a transition from six to five to four spokes is observed with increasing β and the exact transition points are not deterministic since both algorithms rely on randomly chosen samples which most certainly do vary.

REFERENCES

[1] The xerus library for higher order tensor computations. `http://libxerus.org`, 2016.
[2] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

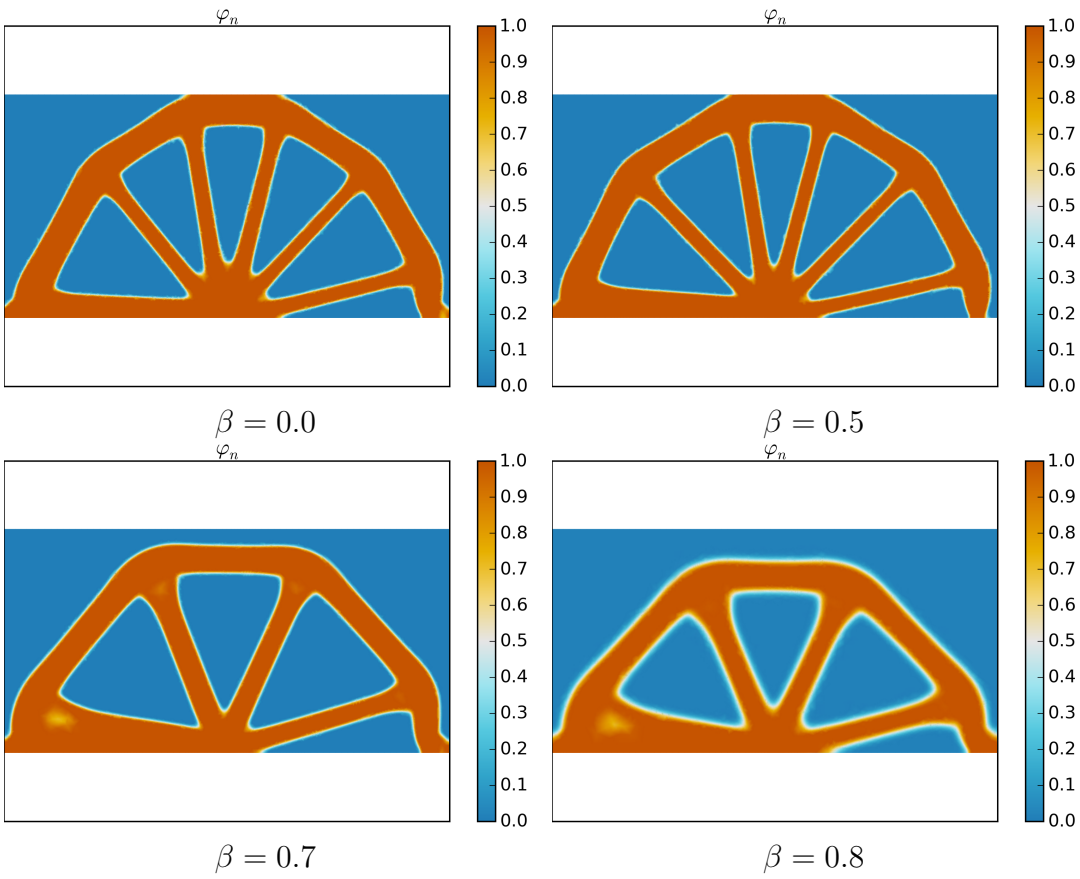


FIGURE 4. Stochastic setting and final form for various β using the low TT-rank reconstruction algorithm.

[3] Martin Sandve Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie Elisabeth Rognes, and Garth N. Wells. The FEniCS Project – Version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[4] Markus Bachmayr, Reinhold Schneider, and André Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, pages 1–50, 2016.

[5] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2016.

[6] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[7] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[8] Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[9] Timothy A. Davis. Algorithm 832: Umfpack v4.3 — an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):196–199, 2004.
[10] Willy Dörfler. A convergent adaptive algorithm for poisson's equation. *SIAM Journal on Numerical Analysis*, 33(3):1106–1124, 1996.
[11] FEniCS Project - Automated solution of Differential Equations by the Finite Element Method, `http://fenicsproject.org`.
[12] Lars Grasedyck, Melanie Kluge, and Sebastian Krämer. Alternating directions fitting (adf) of hierarchical low rank tensors. *Preprint*, 149, 2013.
[13] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.
[14] Wolfgang Hackbusch and Stefan Kühn. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009.
[15] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.
[16] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed tt-rank. *Numerische Mathematik*, 120(4):701–731, 2012.
[17] Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.
[18] Dmitri Kuzmin. Introduction to cfd. University Lecture TU Dortmund, 2007.
[19] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.
[20] Christian Lubich, Ivan V Oseledets, and Bart Vandereycken. Time integration of tensor trains. *SIAM Journal on Numerical Analysis*, 53(2):917–941, 2015.
[21] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
[22] Holger Rauhut, Reinhold Schneider, and Zeljka Stojanac. Low rank tensor recovery via iterative hard thresholding. *arXiv preprint arXiv:1602.05217*, 2016.
[23] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.
[24] Gael Varoquaux et al. Joblib: running python functions as pipeline jobs, 2010-2016. `https://pythonhosted.org/joblib`.