SIAM/ASA J. UNCERTAINTY QUANTIFICATION
Vol. 3, pp. 1109–1135

Polynomial Chaos Expansion of Random Coefficients and the Solution of Stochastic Partial Differential Equations in the Tensor Train Format[*]

Sergey Dolgov[†], Boris N. Khoromskij[‡], Alexander Litvinenko[§], and Hermann G. Matthies[¶]

Abstract. We apply the tensor train (TT) decomposition to construct the tensor product polynomial chaos expansion (PCE) of a random field, to solve the stochastic elliptic diffusion PDE with the stochastic Galerkin discretization, and to compute some quantities of interest (mean, variance, and exceedance probabilities). We assume that the random diffusion coefficient is given as a smooth transformation of a Gaussian random field. In this case, the PCE is delivered by a complicated formula, which lacks an analytic TT representation. To construct its TT approximation numerically, we develop the new block TT cross algorithm, a method that computes the whole TT decomposition from a few evaluations of the PCE formula. The new method is conceptually similar to the adaptive cross approximation in the TT format but is more efficient when several tensors must be stored in the same TT representation, which is the case for the PCE. In addition, we demonstrate how to assemble the stochastic Galerkin matrix and to compute the solution of the elliptic equation and its postprocessing, staying in the TT format. We compare our technique with the traditional sparse polynomial chaos and the Monte Carlo approaches. In the tensor product polynomial chaos, the polynomial degree is bounded for each random variable independently. This provides higher accuracy than the sparse polynomial set or the Monte Carlo method, but the cardinality of the tensor product set grows exponentially with the number of random variables. However, when the PCE coefficients are implicitly approximated in the TT format, the computations with the full tensor product polynomial set become possible. In the numerical experiments, we confirm that the new methodology is competitive in a wide range of parameters, especially where high accuracy and high polynomial degrees are required.

Key words. uncertainty quantification, polynomial chaos expansion, Karhunen–Loève expansion, stochastic Galerkin, tensor product methods, tensor train format, adaptive cross approximation, block cross

AMS subject classifications. 15A69, 65F10, 60H15, 60H35, 65C30

DOI. 10.1137/140972536

[*]Received by the editors June 11, 2014; accepted for publication (in revised form) August 31, 2015; published electronically November 3, 2015.
http://www.siam.org/journals/juq/3/97253.html

[†]Max-Planck-Institut für Dynamik komplexer technischer Systeme, Sandtorstr. 1, 39106 Magdeburg, Germany (dolgov@mpi-magdeburg.mpg.de). The work of this author was partially supported by RFBR grants 13-01-12061-ofi-m, 14-01-00804-A; RSCF grants 14-11-00806, 14-11-00659; and the Stipend of the President of Russia during his stay at the Institute of Numerical Mathematics of the Russian Academy of Sciences.

[‡]Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22, 04103 Leipzig, Germany (bokh@mis.mpg.de).

[§]King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia (alexander.litvinenko@kaust.edu.sa). The work of this author was supported by the SRI-UQ Center at King Abdullah University of Science and Technology. A part of this work was done during his stay at Technische Universität Braunschweig and was supported by the German DFG Project CODECS "Effective approaches and solution techniques for conditioning, robust design and control in the subsurface."

[¶]Institute for Scientific Computing, Technische Universität Braunschweig, Hans-Sommerstr. 65, Brunswick, Germany (wire@tu-bs.de).

1. Motivation.

Situations in which one is concerned with uncertainty quantification often come in the following guise: we are investigating physical models where inputs are not given precisely but instead are random quantities or random fields, or depend on a set of parameters. A classical example is the Darcy flow model with a random diffusion coefficient,

$$-\nabla\kappa(x,\omega)\nabla u(x,\omega) = f(x,\omega), \quad x \in D \subset \mathbb{R}^d, \tag{1.1}$$

where d is the spatial dimension, and $\kappa(x,\omega)$ is a random field dependent on a random parameter ω in a probability space $(\Omega, \mathcal{A}, \mathcal{P})$. The solution $u(x,\omega)$ belongs to $H^1(D)$ w.r.t. x and the same probability space w.r.t. ω. There is an established theory about the existence and uniqueness of the solution to (1.1) under various assumptions on κ and f; see, for example, [2, 21, 23, 44, 47].

In [21, 23] it is shown that under additional assumptions on the right-hand side f and special choices of the test space the problem (1.1) is well-posed. The case where the Lax–Milgram theorem is not applicable (e.g., upper and lower constants $\underline{\kappa}$, $\overline{\kappa}$ in $0 < \underline{\kappa} < \kappa < \overline{\kappa} < \infty$ do not exist) is also considered in [47]. In [21] the authors analyze assumptions on κ which were made in [2] to guarantee the uniqueness and the existence of the solution and to offer a new method with much weakened assumptions. If the expansion for κ is truncated, it is not guaranteed that the truncated κ will be strictly bounded from zero. As a result the existence of the approximate solution to (1.1) is questionable, unless precautions are taken as in [44]. The approach in [21] avoids this from the outset as the permeability coefficient κ is the exponential of smoothed white noise, and, as a result, the ellipticity condition is maintained.

To solve (1.1) we need to discretize the random field κ. This requires some knowledge of the probability space Ω and the probability measure. A widely used approach relies on two assumptions: κ is defined as an invertible smooth function of another random field with known distribution (e.g., normal), and the covariance function of either of those fields is given. After that $\kappa(x,\omega)$ can be *expanded* to a series of functions, depending on x and a set of new parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots)$. Typically used are polynomials of $\boldsymbol{\theta}$, which are hence called the *polynomial chaos expansion* (PCE) [67, 68] family of discretizations. Another approach is the collocation on some grid in $\boldsymbol{\theta}$ [1, 3]. Each of $\boldsymbol{\theta}$ is a random quantity depending on ω, but the domain of $\boldsymbol{\theta}$ is known deterministically. Introducing a discretization for $\boldsymbol{\theta}$, we turn the stochastic problem (1.1) into a high-dimensional deterministic one. However, its straightforward numerical solution suffers from the *curse of dimensionality*: even if $\boldsymbol{\theta}$ contains a finite number of variables (say, M), the number of discrete degrees of freedom grows exponentially with M.

To surmount this issue, a data-sparse approximation is needed. During recent years, low-rank tensor product techniques were successfully applied to the solution of high-dimensional stochastic and parametric PDEs. A recent literature survey of low-rank tensor approximation techniques can be found in [27]. The tensor product approach involves a *format*, in which the data are represented, and a set of *algorithms* to manipulate the data in the format. The algorithms can be roughly separated into three classes: methods performing basic approximate algebra (additions, products, and simple iterative methods) of data within the format; methods constructing the formatted representation from a few entries of the initial data; and methods aiming to solve equations, e.g., linear systems, ODEs, or eigenvalue problems, keeping all data in the format.

To some extent, these methods have already been applied to parametric problems. Nonintrusive (black box) tensor methods for multiparametric problems, i.e., "class 2," were developed in [4, 5, 15]. In particular, in [4] the authors follow the stochastic collocation approach and compute functionals of the solution of multiparametric PDEs. Since the stochastic collocation allows solving uncoupled deterministic problems for different collocation points, the functional of the solution (e.g., the average value) can be approximated straightforwardly via the black box hierarchical tensor interpolation algorithm. To compute the whole stochastic solution is a more difficult problem, especially in the stochastic Galerkin framework, where deterministic problems are coupled.

In [36, 37, 46, 61, 70] the authors develop iterative methods and preconditioners to solve numerically discretized multiparametric problems. Several manipulations of the PCE with a low-rank approximation have been considered. In [19] the authors assume that the solution has a low-rank *canonical polyadic* (CP) tensor format and develop methods for the CP-formatted computation of level sets. In [45, 18] the authors analyzed tensor ranks of the stochastic operator. The proper generalized decomposition was applied for solving high-dimensional stochastic problems in [50, 51]. In [33, 34, 35] the authors employed newer tensor formats, the tensor train (TT) and quantized TT (QTT), for the approximation of coefficients and the solution of stochastic elliptic PDEs. The theoretical study of the complexity of the stochastic equation was provided, for example, by means of the analytic regularity and (generalized) polynomial chaos (PC) approximation [68] for control problems constrained by linear parametric elliptic and parabolic PDEs [38].

Other classical techniques to cope with high-dimensional problems are sparse grids [28, 10, 49] and (quasi) Monte Carlo methods [26, 62, 39]. Nevertheless, tensor product methods are more flexible than sparse grids, as they avoid severe reductions of the model from the very beginning and adapt a suitable structure on the discrete level. Compared to Monte Carlo methods, tensor techniques work implicitly with the whole solution, and even a construction of a tensor format for entrywise given data in a black box manner uses less randomness than the Monte Carlo approach.

In this article we approximate the PCE of the input coefficient $\kappa(x, \omega)$ in the TT format. After that we compute the solution $u(x, \omega)$ and perform all postprocessing in the same TT format. The first stage, computation of the PCE of κ, involves a lengthy formula, defining each entry of the discretized coefficient. To perform this computation efficiently, we develop a new *block cross* approximation algorithm, which constructs the TT format for κ from a few evaluations of the entrywise formula. This formula delivers several tensors that are to be summed and approximated in a TT format. We show that the new algorithm is more efficient than several runs of a previously existing cross method [58] for each tensor separately. As soon as the coefficient is given in the TT format, it becomes very easy to construct the stiffness matrix, derived from the stochastic Galerkin discretization of (1.1). We apply the alternating iterative tensor algorithm to solve a large linear system arising from (1.1) and finally use the cross algorithm again to compute the exceedance probability from the solution.

In the next section, we outline the general Galerkin, polynomial chaos expansion (PCE), and Karhunen–Loève expansion (KLE) discretization schemes for a random field. An introduction to the TT methods and the new block cross interpolation algorithm are presented in section 3. Some details of how to apply the block cross algorithm to the PCE calculations are given in section 4.1. We start with the TT approximation of the multidimensional input coefficient κ. After that, in section 4.2, we construct the stochastic Galerkin matrix in the TT format. Section 4.4 is devoted to the efficient postprocessing (computation of the mean value, covariance, and probability of a particular event) in the TT format. Numerical results in section 5 demonstrate the practical performance of the TT approach in the outlined tasks.

2. Discretization and computation.

For brevity, we follow [43, 44], where more references may be found. See also the recent monograph [40] to study more about KLE, PCE, and multi-indices. In [17, 61, 64, 65] the authors discuss the stochastic Galerkin matrix, its sparsity, and preconditioners.

To discretize (1.1), we follow the Galerkin approach. The finite element method (for example, with piecewise linear functions) is a natural way to discretize the spatial part. We choose a finite dimensional subspace

$$\mathcal{V}_N = \operatorname{span}\{\varphi_1(x), \ldots, \varphi_N(x)\} \subset \mathcal{V}, \tag{2.1}$$

where $\mathcal{V} = H^1(D) \cap L_2^0(\bar{D})$ is the Hilbert space of functions on x. For simplicity, we impose the homogeneous Dirichlet boundary conditions.

Discretization in the probability space $(\Omega, \mathcal{A}, \mathcal{P})$ is less trivial. We use the KLE to determine a finite set of independent parameters, defining $\kappa(x,\omega)$. This set of independent parameters (random variables) and polynomials in them generate the finite dimensional Galerkin subspaces for the stochastic discretization. The full spaces for the formulation, which depend on the coefficient and right-hand side, are given in [21, 23]. In these spaces, the stochastic PDE is represented by a self-adjoint positive definite operator, and the problem is well posed.

2.1. Discretization of the input random field.

We assume that $\kappa(x,\omega)$ may be seen as a smooth transformation $\kappa = \phi(\gamma)$ of the Gaussian random field $\gamma(x,\omega)$. In this section, we explain how to compute the KLE of γ if the covariance of κ is given. For more details see [70, sections 3.4.1, 3.4.2] or [29].

A typical example is the log-normal field with $\phi(\gamma) = \exp(\gamma)$. Expanding ϕ in a series of the Hermite polynomials gives

$$\phi(\gamma) = \sum_{i=0}^{\infty} \phi_i h_i(\gamma) \approx \sum_{i=0}^{Q} \phi_i h_i(\gamma), \qquad \phi_i = \int_{-\infty}^{+\infty} \phi(z) \frac{1}{i!} h_i(z) \exp(-z^2/2) dz, \tag{2.2}$$

where $h_i(\cdot)$ is the ith Hermite polynomial, and Q is the number of terms after the truncation. Note that this expansion and further manipulations are performed at this point purely to find the relevant independent parameters. In the differential equation κ is unchanged.

The Gaussian field $\gamma(x,\omega)$ can be written as the KLE. First, given the covariance function of $\kappa(x,\omega)$, we may relate it with the covariance function of $\gamma(x,\omega)$ as (see details in [70, sections 3.4.1, 3.4.2])

$$\operatorname{cov}_\kappa(x,y) = \int_\Omega (\kappa(x,\omega) - \bar{\kappa}(x))(\kappa(y,\omega) - \bar{\kappa}(y))\, dP(\omega) \approx \sum_{i=0}^{Q} i! \phi_i^2 \operatorname{cov}_\gamma^i(x,y), \tag{2.3}$$

where $\bar{\kappa}(x)$ is the expectation of $\kappa(x, \omega)$. Solving this implicit Q-order equation, we derive $\mathrm{cov}_\gamma(x, y)$ [70]. Now, the KLE can be computed as follows:

$$\gamma(x, \omega) = \sum_{m=1}^{\infty} g_m(x)\theta_m(\omega), \quad \text{where} \quad \int_D \mathrm{cov}_\gamma(x, y) g_m(y) dy = \lambda_m g_m(x). \tag{2.4}$$

Here we assume that the eigenfunctions g_m absorb the square roots of the Karhunen–Loève eigenvalues. The stochastic variables θ_m are normalized (they are uncorrelated and jointly Gaussian).

The initial coefficient κ depends nonlinearly on θ_m. In the discrete setting, we truncate the PCE and write it for M random variables,

$$\kappa(x, \omega) \approx \sum_{\boldsymbol{\alpha} \in \mathcal{J}_M} \kappa(x, \boldsymbol{\alpha}) H_{\boldsymbol{\alpha}}(\boldsymbol{\theta}(\omega)), \quad \text{where} \quad H_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) := h_{\alpha_1}(\theta_1) \cdots h_{\alpha_M}(\theta_M) \tag{2.5}$$

is the multivariate Hermite polynomial, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$ is a multi-index (a tuple of multinomial orders), $h_{\alpha_m}(\theta_m)$ is the univariate Hermite polynomial, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ is a tuple of random variables, and $\mathcal{J}_M$ is a set of all multinomial orders (see definition below). The Galerkin coefficients $\kappa(x, \boldsymbol{\alpha})$ are evaluated as follows:

$$\kappa(x, \boldsymbol{\alpha}) = \frac{(\alpha_1 + \cdots + \alpha_M)!}{\alpha_1! \cdots \alpha_M!} \phi_{\alpha_1 + \cdots + \alpha_M} \prod_{m=1}^{M} g_m^{\alpha_m}(x), \tag{2.6}$$

where $\phi_{\alpha_1 + \cdots + \alpha_M}$ is the Galerkin coefficient of the transform function in (2.2) and $g_m^{\alpha_m}(x)$ means just the α_mth power of the KLE function value $g_m(x)$.

In practice, we restrict the polynomial orders in (2.5) to finite limits, which can be done in different ways.

Definition 2.1. *The* full *multi-index set is defined by restricting each component independently,*

$$\mathcal{J}_{M,\boldsymbol{p}} = \{\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M) : \quad \alpha_m = 0, \ldots, p_m, \quad m = 1, \ldots, M\},$$

where $\boldsymbol{p} = (p_1, \ldots, p_M)$ is a shortcut for the tuple of order limits.

The full set provides high flexibility for the resolution of stochastic variables [18, 19, 61]. However, its cardinality is equal to $\prod_{m=1}^{M}(p_m + 1) \le (p + 1)^M$ if $p_m \le p$. This may become an enormous number even if p and M are moderate ($p \sim 3$ and $M \sim 20$ are typical). In this paper, we do not store all $(p + 1)^M$ values explicitly but instead approximate them via a *tensor product* representation.

Another approach is to preselect the set of polynomials with the moderate cardinality [63, 44, 9, 11].

Definition 2.2. *The* sparse *multi-index set is defined by restricting the sum of components,*

$$\mathcal{J}_{M,p}^{sp} = \{\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M) : \quad \boldsymbol{\alpha} \ge 0, \quad \alpha_1 + \cdots + \alpha_M \le p\}.$$

The sparse set contains $\mathcal{O}(\frac{M!}{p!(M-p)!}) = \mathcal{O}(M^p)$ values if $M \gg p$, which is definitely less than $(p + 1)^M$. However, the negative side is that for a fixed p some variables are resolved worse than others, and the approximation accuracy may suffer. It may also be harmful to increase p since in the sparse set it contributes exponentially to the complexity.

The low-rank tensor approximation allows reduction of the storage of the coefficient with the full set $\mathcal{J}_{M,\boldsymbol{p}}$ from $\mathcal{O}(p^M)$ to $\mathcal{O}(Mpr^2)$, where r is a data-dependent *tensor rank*. Theoretical estimates of r are under development; numerical studies reflect that often r does not depend on p and depends linearly (or even more mildly) on M [33]. When M is moderate, and p is relatively large, the low-rank approach with the full index set (Definition 2.1) becomes preferable. Also, as soon as the coefficient κ is computed in a tensor product form, to assemble the Galerkin matrix of the stochastic PDE is a much easier task than with the sparse set.

Some complexity reduction in formula (2.6) can be achieved with the help of the KLE for the initial field $\kappa(x,\omega)$. Consider the expansion

$$\kappa(x,\omega) = \bar{\kappa}(x) + \sum_{\ell=1}^{\infty} \sqrt{\mu_\ell} v_\ell(x)\eta_\ell(\omega) \approx \bar{\kappa}(x) + \sum_{\ell=1}^{L} \sqrt{\mu_\ell} v_\ell(x)\eta_\ell(\omega), \tag{2.7}$$

where $v_\ell(x)$ are eigenfunctions of the integral operator with the covariance as the kernel (see, e.g., [70, 29, 44]). We know that the set $V(x) = \{v_\ell(x)\}_{\ell=1}^{L}$, where L is the number of KLE terms after the truncation, serves as an optimally reduced basis. It can hence be used for dimension reduction. Therefore, instead of using (2.6) directly, we project it onto $V(x)$:

$$\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha}) = \frac{(\alpha_1 + \cdots + \alpha_M)!}{\alpha_1! \cdots \alpha_M!} \phi_{\alpha_1+\cdots+\alpha_M} \int_D \prod_{m=1}^{M} g_m^{\alpha_m}(x) v_\ell(x) dx. \tag{2.8}$$

Note that the range $\ell = 1, \dots, L$ may be much smaller than the number of discretization points N in D. After that, we restore the approximate coefficients (2.6):

$$\kappa(x,\boldsymbol{\alpha}) \approx \bar{\kappa}(x) + \sum_{\ell=1}^{L} v_\ell(x) \tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha}). \tag{2.9}$$

This will be inserted into (2.5). But we stress that the expansions (2.4), (2.5), and (2.7) which give rise to the truncations (2.5) and (2.9) a priori only converge in $L_2(D \times \Omega)$. Equation (1.1), on the other hand, is only stable to perturbations in L_∞, and therefore on the continuous level we do not approximate κ by (2.5), and in (1.1) κ is unchanged. These approximations are computed here primarily to find the relevant random parameters for the discretization of the solution, and only then to obtain a representation of κ after the discretization has been performed.

2.2. Discretization of the stochastic operator equation. The same PCE ansatz of the coefficient (2.5) may be adopted to discretize the solution u and ultimately the bilinear form and operator associated with problem (1.1); see [18, 19, 43, 44]. For brevity, we illustrate the procedure in the case of a deterministic right-hand side $f = f(x)$.

The positive definite operator corresponding to (1.1) with coercivity constant, say, $C_c > 0$ is thereby represented on the finite dimensional discretization subspaces by its positive definite Bubnov–Galerkin projection, where the coercivity constant cannot decrease. Observe that as now this is an operator on a finite dimensional space, it is itself an element of a finite dimensional space, and on finite dimensional spaces all norms are equivalent—with possibly dimension dependent constants.

It is now that the approximation (2.9) of κ is used and inserted into the discretized operator. As noted before, the KLE (2.7) only converges in L_2. But now this is on a finite dimensional space. In view of the norm-equivalences mentioned before, this means that for any of the discretization spaces and for any $\epsilon > 0$ such that $C_c - \epsilon > 0$, one may find a truncation $L = L_\epsilon$ in the KLE (2.7), (2.9), such that the discretized and KLE truncated operator has coercivity constant larger than $C_c - \epsilon$ [44, Theorem 20]. In a way the KLE truncation is thus a "variational crime" similar to numerical integration in the normal finite element method, and the first Strang lemma applies for error estimates.

Given the KLE components (2.7) and the spatial discretization basis (2.1), we first assemble the spatial Galerkin matrices,

$$\boldsymbol{K}_0^{(x)}(i,j) = \int_D \bar{\kappa}(x)\nabla\varphi_i(x)\cdot\nabla\varphi_j(x)dx, \quad \boldsymbol{K}_\ell^{(x)}(i,j) = \int_D v_\ell(x)\nabla\varphi_i(x)\cdot\nabla\varphi_j(x)dx, \tag{2.10}$$

for $i,j = 1,\dots,N$, $\ell = 1,\dots,L$. Now we take into account the PCE (2.5), and, assuming that u is discretized the same way as (2.5) with the same $\mathcal{J}_{M,\boldsymbol{p}}$ or $\mathcal{J}^{sp}_{M,p}$, we have to compute the integral in (2.11) over stochastic coordinates $\boldsymbol{\theta}$ and compute the stochastic parts $\boldsymbol{K}_\ell^{(\omega)} \in \mathbb{R}^{\#\mathcal{J}_M \times \#\mathcal{J}_M}$ of the Galerkin matrix as follows (see also [29, 70, 43]):

$$\boldsymbol{K}_\ell^{(\omega)}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \int_{\mathbb{R}^M} H_{\boldsymbol{\alpha}}(\boldsymbol{\theta})H_{\boldsymbol{\beta}}(\boldsymbol{\theta}) \sum_{\boldsymbol{\nu}\in\mathcal{J}_M} \tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\nu})H_{\boldsymbol{\nu}}(\boldsymbol{\theta})\rho(\boldsymbol{\theta})d\boldsymbol{\theta} = \sum_{\boldsymbol{\nu}\in\mathcal{J}_M} \boldsymbol{\Delta}_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\nu}}\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\nu}), \tag{2.11}$$

where $\rho(\boldsymbol{\theta}) = \rho(\theta_1)\cdots\rho(\theta_M)$ is the probability density with $\rho(\theta_m) = \frac{1}{\sqrt{2\pi}}\exp(-\theta_m^2/2)$, and

$$\boldsymbol{\Delta}_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\nu}} = \Delta_{\alpha_1,\beta_1,\nu_1}\cdots\Delta_{\alpha_M,\beta_M,\nu_M}, \quad \Delta_{\alpha_m,\beta_m,\nu_m} = \int_{\mathbb{R}} h_{\alpha_m}(\theta)h_{\beta_m}(\theta)h_{\nu_m}(\theta)\rho(\theta)d\theta \tag{2.12}$$

is the triple product of the Hermite polynomials, and $\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\nu})$ is according to (2.8). Observe that the sum in (2.11) over the *infinite* set $\mathcal{J}_M$ is in fact only finite [44, Theorem 18], because, except for a finite number of multi-indices $\boldsymbol{\nu}$, the polynomial $H_{\boldsymbol{\nu}}$ is orthogonal to the product $H_{\boldsymbol{\alpha}}H_{\boldsymbol{\beta}}$, and the integral vanishes. Let us denote $\boldsymbol{\Delta}_0(\boldsymbol{\alpha},\boldsymbol{\beta}) = \Delta_{\alpha_1,\beta_1,0}\cdots\Delta_{\alpha_M,\beta_M,0}$, i.e., $\boldsymbol{\Delta}_0 \in \mathbb{R}^{\#J_M\times\#J_M}$. Putting together (2.9), (2.10), and (2.11), we obtain the whole discrete stochastic Galerkin matrix,

$$\mathbf{K} = \boldsymbol{K}_0^{(x)} \otimes \boldsymbol{\Delta}_0 + \sum_{\ell=1}^{L} \boldsymbol{K}_\ell^{(x)} \otimes \boldsymbol{K}_\ell^{(\omega)}, \tag{2.13}$$

which is a square matrix of size $N \cdot \#\mathcal{J}_M$, and $\otimes$ is the Kronecker product.

For the sparse index set, we need to compute $\mathcal{O}((\#\mathcal{J}^{sp}_{M,p})^3)$ entries of $\boldsymbol{\Delta}$ explicitly. For the full index set and $\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\nu})$ given in the tensor format, the direct product in $\boldsymbol{\Delta}$ in (2.12) allows us to exploit the same format for (2.13) and to simplify the procedure; see section 4.2.

The deterministic right-hand side is extended to the size of $\mathbf{K}$ easily,

$$\mathbf{f} = \boldsymbol{f}_0 \otimes \boldsymbol{e}_0, \qquad \boldsymbol{f}_0(i) = \int_D \varphi_i(x)f(x)dx, \quad i = 1,\dots,N, \tag{2.14}$$

and $\boldsymbol{e}_0$ is the first identity vector of size $\#\mathcal{J}_M$, $\boldsymbol{e}_0 = (1, 0, \ldots, 0)^\top$, which assigns the deterministic $f(x)$ to the zeroth-order Hermite polynomial in the parametric space.

2.3. Solution of the stochastic equation. Now the discrete equation writes as a linear system

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \tag{2.15}$$

where $\mathbf{u} \in \mathbb{R}^{N \cdot \#\mathcal{J}_M}$ is the vector of the Galerkin coefficients for the solution, with elements enumerated by $u(i, \boldsymbol{\alpha})$.

In [61] the authors propose two new strategies for constructing preconditioners for the stochastic Galerkin system to be used with Krylov subspace iterative solvers. The authors also research the block sparsity structure of the corresponding coefficient matrix as well as the stochastic Galerkin matrix. In [64, 65] the authors develop and analyze a Kronecker-product preconditioner.

In many cases it is sufficient to use the mean-field preconditioner $\boldsymbol{K}_0^{(x)} \otimes \boldsymbol{\Delta}_0$, which is easy to invert due to the Kronecker form. We follow this approach. To solve the system in a tensor product format, we employ alternating optimization methods [14].

3. Tensor product formats and low-rank data compression.

We see that the cardinality of the full polynomial set $\mathcal{J}_{M,\boldsymbol{p}}$ may rise to prohibitively large values $(p+1)^M$. In this paper, we study two ways to alleviate this problem. First, we can fix the basis set a priori. This is the case with the sparse set $\mathcal{J}_{M,p}^{sp}$. Due to particular properties of the stochastic elliptic equation, it is possible to derive a posteriori error indicators and build the sparse set adaptively [16, 11].

Another approach, which is applicable to a wider class of problems, is to use the full discretization set but approximate already discrete data, via a low-rank tensor product format. For stochastic PDEs, low-rank approximations were used in, e.g., [8, 15, 19, 33, 46, 51]. This approach requires specific computational algorithms, since the data are represented in a nonlinear fashion. In this section we suggest such an algorithm to construct a data-sparse format of the stochastic coefficient and quantities of interest.

3.1. Tensor train decomposition.

To show the techniques in the briefest way, we choose the so-called *matrix product states* (MPS) formalism [20], which introduces the following representation of a multivariate array, or *tensor*:

$$\mathbf{u}(\alpha_1, \ldots, \alpha_M) = \sum_{s_1=1}^{r_1} \sum_{s_2=1}^{r_2} \cdots \sum_{s_{M-1}=1}^{r_{M-1}} \boldsymbol{u}^{(1)}_{s_0,s_1}(\alpha_1) \boldsymbol{u}^{(2)}_{s_1,s_2}(\alpha_2) \cdots \boldsymbol{u}^{(M)}_{s_{M-1},s_M}(\alpha_M). \tag{3.1}$$

Surely, in the same form we may write $\boldsymbol{\kappa}(\boldsymbol{\alpha})$. In numerical linear algebra this format is known as *tensor train* (TT) representation [53, 55]. Each TT core (or *block*) is a three-dimensional array, $\boldsymbol{u}^{(m)} \in \mathbb{R}^{r_{m-1} \times (p_m+1) \times r_m}$, $m = 1, \ldots, M$, where p_m denotes the *mode size*, the polynomial order in the variable α_m, and $r_m = r_m(\mathbf{u})$ is the *TT rank*. The total number of entries scales as $\mathcal{O}(Mpr^2)$, which is tractable as long as $r = \max\{r_m\}$ is moderate.

We still have not specified the *border* rank indices s_0 and s_M. In the classical TT definition [53] and in (3.1), there is only one tensor $\mathbf{u}(\boldsymbol{\alpha})$ in the left-hand side; therefore $s_0 = s_M = 1$, and also $r_0 = r_M = 1$. However, the left-hand side might contain several tensors, such as

$\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha})$ in (2.8). Then we can associate $s_0 = \ell$ or $s_M = \ell$ and approximate all tensors for different ℓ in the same *shared* TT representation.

High-dimensional matrices (cf. $\mathbf{K}$ in (2.13)) can be also presented in the TT format,

$$\mathbf{K} = \sum_{s_0,\dots,s_{M-1}} \boldsymbol{K}^{(0)}_{s_0} \otimes \boldsymbol{K}^{(1)}_{s_0,s_1} \otimes \cdots \otimes \boldsymbol{K}^{(M)}_{s_{M-1}}.$$

The matrix by vector product $\mathbf{K}\mathbf{u}$ is then recast to TT blocks of $\mathbf{K}$ and $\mathbf{u}$. Similarly, using the multilinearity of the TT format, we can cast linear combinations of initial tensors to concatenations of TT blocks.

The principal benefit of the TT format in comparison with the canonical polyadic (CP) decomposition (which is also popular; see, e.g., [18, 19]) is a stable quasi-optimal rank reduction procedure [53] based on singular value decompositions (SVDs). The complexity scales as $\mathcal{O}(Mpr^3)$, i.e., it is free from the curse of dimensionality, while the full accuracy control takes place. This procedure can be used to reduce unnecessarily large ranks after the matrix by vector product or the linear combination in the TT format, and to compress a tensor if it is fully given as a multidimensional array and fits into the memory. However, in our situation this is not the case, and we need another approach to constructing a TT format.

3.2. Cross interpolation of matrices. If $M = 2$, the left-hand side of (3.1) can be seen as a matrix. For simplicity we consider this case first. The basic assumption is that any entry of a matrix (or tensor) can be evaluated. However, to construct a TT approximation, we do not want to compute all elements, but only few of them.

The principal ingredient for this is based on the efficiency of an incomplete Gaussian elimination in an approximation of a low-rank matrix, also known as the adaptive cross approximation (ACA) [6, 7]. Given is a matrix $\boldsymbol{U} = [\boldsymbol{U}(i,j)] \in \mathbb{R}^{p\times q}$; we select some "good" columns and rows to approximate the whole matrix,

$$\boldsymbol{U}(i,j) \approx \tilde{\boldsymbol{U}}(i,j) = \boldsymbol{U}(i,\mathbb{J}) \cdot \widehat{\boldsymbol{U}}^{-1} \cdot \boldsymbol{U}(\mathbb{I},j), \tag{3.2}$$

where $\widehat{\boldsymbol{U}} = \boldsymbol{U}(\mathbb{I},\mathbb{J})$, and $\mathbb{I} \subset \{1,\dots,p\}$, $\mathbb{J} \subset \{1,\dots,q\}$ are sets of indices of cardinality r, so, e.g., $\boldsymbol{U}(i,\mathbb{J}) \in \mathbb{R}^{1\times r}$. It is known that there exists a quasi-optimal set of interpolating indices $\mathbb{I},\mathbb{J}$.

Lemma 3.1 (maximum volume (*maxvol*) principle [25]). *If $\mathbb{I}$ and $\mathbb{J}$ are such that* $\det \boldsymbol{U}(\mathbb{I},\mathbb{J})$ *is maximal among all $r\times r$ submatrices of $\boldsymbol{U}$, then*

$$\|\boldsymbol{U} - \tilde{\boldsymbol{U}}\|_C \le (r+1) \min_{\operatorname{rank}(\boldsymbol{V})=r} \|\boldsymbol{U} - \boldsymbol{V}\|_2,$$

where $\|\cdot\|_C$ is the Chebyshev norm, $\|\boldsymbol{X}\|_C = \max_{i,j} |\boldsymbol{X}_{i,j}|$.

In practice, however, the computation of the true maximum volume submatrix is infeasible, since it is an NP-hard problem. Instead one performs a heuristic iteration in an alternating fashion [24]: we start with some (e.g., random) low-rank factor $\boldsymbol{U}^{(1)} \in \mathbb{R}^{p\times r}$, determine indices $\mathbb{I}$ yielding a quasi-maxvol $r\times r$ submatrix in $\boldsymbol{U}^{(1)}$, and compute $\boldsymbol{U}^{(2)}$ as r columns of $\boldsymbol{U}$ of the indices $\mathbb{I}$. Vice versa, in the next step, we find quasi-maxvol column indices in $\boldsymbol{U}^{(2)}$ and calculate corresponding pr elements, collecting them into the newer $\boldsymbol{U}^{(1)}$, which hopefully approximates the true low-rank factor better than the initial guess. This process continues until the convergence, which appears to be quite satisfactory in practice.

3.3. TT-cross interpolation of tensors. In higher dimensions we recurrently proceed in the same way, since the TT format constitutes a recurrent matrix low-rank factorization. Let us merge the first m and the last $M-m$ indices from $\boldsymbol{\alpha}$. The corresponding TT blocks will induce the following matrices.

Definition 3.2. *Given a TT format* (3.1) *and an index m, define the* left interface matrix $\boldsymbol{U}^{<m} \in \mathbb{R}^{(p_1+1)\cdots(p_{m-1}+1)\times r_{m-1}}$ *and the* right interface *matrix* $\boldsymbol{U}^{>m} \in \mathbb{R}^{r_m\times(p_{m+1}+1)\cdots(p_M+1)}$ *as follows:*

$$\boldsymbol{U}^{<m}_{s_{m-1}}(\alpha_1,\ldots,\alpha_{m-1}) = \sum_{s_1=1}^{r_1}\cdots\sum_{s_{m-2}=1}^{r_{m-2}} \boldsymbol{u}^{(1)}_{s_1}(\alpha_1)\cdots\boldsymbol{u}^{(m-1)}_{s_{m-2},s_{m-1}}(\alpha_{m-1}),$$

$$\boldsymbol{U}^{>m}_{s_m}(\alpha_{m+1},\ldots,\alpha_M) = \sum_{s_{m+1}=1}^{r_{m+1}}\cdots\sum_{s_{M-1}=1}^{r_{M-1}} \boldsymbol{u}^{(m+1)}_{s_m,s_{m+1}}(\alpha_{m+1})\cdots\boldsymbol{u}^{(M)}_{s_{M-1}}(\alpha_M).$$

Such matrices are convenient to relate high-dimensional TT expressions with their two-dimensional analogues. For example, the TT format (3.1) can be written in the following form:

$$\mathbf{u}(\boldsymbol{\alpha}) \approx \boldsymbol{U}^{<m}(\alpha_1,\ldots,\alpha_{m-1})\cdot\boldsymbol{u}^{(m)}(\alpha_m)\cdot\boldsymbol{U}^{>m}(\alpha_{m+1},\ldots,\alpha_M). \tag{3.3}$$

The *TT-cross* algorithm [56] assumes that the above expansion is valid on some $r_{m-1}(p_m+1)r_m$ indices and can thus be seen as a system of equations on elements of $u^{(m)}$. Let us be given r_{m-1} *left* indices $(\hat\alpha_1,\ldots,\hat\alpha_{m-1}) \in \mathbb{I}^{(m-1)}$ and r_m *right* indices $(\hat\alpha_{m+1},\ldots,\hat\alpha_M)\in\mathbb{J}^{(m+1)}$. For the single index α_m we allow its full range $\{\alpha_m\} = \{0,1,\ldots,p_m\}$. Requiring that (3.3) is valid on the combined indices

$$(\hat\alpha_1,\ldots,\hat\alpha_{m-1},\alpha_m,\hat\alpha_{m+1},\ldots,\hat\alpha_M) \equiv \left(\mathbb{I}^{(m-1)},\alpha_m,\mathbb{J}^{(m+1)}\right),$$

we obtain the following computational rule for $\boldsymbol{u}^{(m)}$:

$$\boldsymbol{u}^{(m)}(\alpha_m) = \widehat{\boldsymbol{U}}^{-1}_{<m}\cdot\mathbf{u}\left(\mathbb{I}^{(m-1)},\alpha_m,\mathbb{J}^{(m+1)}\right)\cdot\widehat{\boldsymbol{U}}^{-1}_{>m}, \tag{3.4}$$

where $\widehat{\boldsymbol{U}}_{<m} = \boldsymbol{U}^{<m}(\mathbb{I}^{(m-1)})$ and $\widehat{\boldsymbol{U}}_{>m} = \boldsymbol{U}^{>m}(\mathbb{J}^{(m+1)})$ are the submatrices of $\boldsymbol{U}^{<m}$ and $\boldsymbol{U}^{>m}$ at indices $\mathbb{I}^{(m-1)}$ and $\mathbb{J}^{(m+1)}$, respectively.

Of course, these submatrices must be nonsingular. In the previous subsection we saw a good strategy: if indices $\mathbb{I}^{(m-1)}$ and $\mathbb{J}^{(m+1)}$ are chosen in accordance with the maxvol principle for $\boldsymbol{U}^{<m}$ and $\boldsymbol{U}^{>m}$, the submatrices are not only nonsingular, but also provide a good approximation for (3.3). However, $\boldsymbol{U}^{<m}$ and $\boldsymbol{U}^{>m}$ are too large to be treated directly. Instead, we use the *alternating* iteration over the dimensions and build nested index sets.

The alternating iteration means that we loop over $m=1,2,\ldots,M$ (the so-called *forward* iteration) and $m=M,M-1,\ldots,1$ (*backward* iteration). Consider first the forward transition, $m-1\to m$. Given the set $\mathbb{I}^{(m-1)}$, the *nested* indices $\mathbb{I}^{(m)}$ are defined as follows. We concatenate $\mathbb{I}^{(m-1)}$ and α_m, i.e., consider $r_{m-1}(p_m+1)$ indices $(\hat\alpha_1,\ldots,\hat\alpha_{m-1},\alpha_m)\in\{\mathbb{I}^{(m-1)},\alpha_m\}$, and select r_m indices $\mathbb{I}^{(m)}$ only from $\{\mathbb{I}^{(m-1)},\alpha_m\}$, not from all $\mathcal{O}(p^m)$ possibilities. It can be seen that the next interface $\boldsymbol{U}^{<m+1}$, restricted to $\{\mathbb{I}^{(m-1)},\alpha_m\}$, can be computed as a product of

the previous submatrix $\widehat{\boldsymbol{U}}_{<m}$ and the current TT block $\boldsymbol{u}^{(m)}$. To formalize this, we need the following definition.

Definition 3.3. *Given a three-dimensional tensor $\boldsymbol{u}^{(m)} \in \mathbb{R}^{r_{m-1}\times(p_m+1)\times r_m}$, introduce the following reshapes, both pointing to the same data stored in $\boldsymbol{u}^{(m)}$:*

- Left folding: $\boldsymbol{u}^{|m\rangle}(s_{m-1},\alpha_m;\ s_m) = \boldsymbol{u}^{(m)}_{s_{m-1},s_m}(\alpha_m), \quad \boldsymbol{u}^{|m\rangle} \in \mathbb{R}^{r_{m-1}(p_m+1)\times r_m}$.
- Right folding: $\boldsymbol{u}^{\langle m|}(s_{m-1};\ \alpha_m, s_m) = \boldsymbol{u}^{(m)}_{s_{m-1},s_m}(\alpha_m), \quad \boldsymbol{u}^{\langle m|} \in \mathbb{R}^{r_{m-1}\times(p_m+1)r_m}$.

Then the restriction of $\boldsymbol{U}^{<m+1}$ writes as follows:

$$\boldsymbol{V}^{\langle m|} = \widehat{\boldsymbol{U}}_{<m}\boldsymbol{u}^{\langle m|} \quad \text{and} \quad \boldsymbol{U}^{<m+1}\left(\mathbb{I}^{(m-1)},\alpha_m\right) = \boldsymbol{V}^{|m\rangle} \in \mathbb{R}^{r_{m-1}(p_m+1)\times r_m}.$$

Thus, it is enough to apply the maximum volume algorithm [24] to the matrix $\boldsymbol{V}^{|m\rangle}$, deriving *local* maxvol indices $\hat{i}_m \subset \{1,\dots,r_{m-1}(p_m+1)\}$, and obtain both $\mathbb{I}^{(m)}$ and $\widehat{\boldsymbol{U}}_{<m+1}$ by the restriction

$$\mathbb{I}^{(m)} = \{\mathbb{I}^{(m-1)},\alpha_m\}(\hat{i}_m), \quad \widehat{\boldsymbol{U}}_{<m+1} = \boldsymbol{V}^{|m\rangle}(\hat{i}_k) \in \mathbb{R}^{r_m\times r_m}.$$

The backward transition $m+1 \to m$ for $\mathbb{J}^{(m)}$ and $\widehat{\boldsymbol{U}}_{>m}$ can be written analogously. We show it directly in Algorithm 1 below. In total, we need only $\mathcal{O}(n_{it}Mpr^2)$ entries of $\mathbf{u}$ to be evaluated, where n_{it} is the number of alternating iterations, typically on the order of 10.

3.4. Rank-adaptive DMRG-cross algorithm. A drawback of the TT-cross method is that the TT ranks are fixed; they have to be guessed a priori, which is also a problem of exponential complexity in M. A remedy is to consider larger portions of data in each step. The Density Matrix Renormalization Group (DMRG) algorithm was developed in the quantum physics community (see [66]; see also the review [59] and the references therein) to solve high-dimensional eigenvalue problems coming from the stationary spin Schrödinger equation. It is written in an alternating fashion similar to that of the TT-cross procedure described above. The crucial difference is that instead of one TT block as in (3.4), we calculate two neighboring TT blocks at once.

In the *DMRG-cross* [58] interpolation algorithm, this is performed as follows. Given $\mathbb{I}^{(m-1)}$, $\mathbb{J}^{(m+2)}$, we compute

$$\boldsymbol{u}^{(m,m+1)}(\alpha_m,\alpha_{m+1}) = \widehat{\boldsymbol{U}}_{<m}^{-1}\cdot \mathbf{u}\left(\mathbb{I}^{(m-1)},\alpha_m,\alpha_{m+1},\mathbb{J}^{(m+2)}\right)\cdot \widehat{\boldsymbol{U}}_{>m+1}^{-1}.$$

Then we need to separate indices α_m and α_{m+1} to recover the initial TT structure. This can be done via the SVD. The four-dimensional array $\boldsymbol{u}^{(m,m+1)}$ is reshaped to a matrix $\boldsymbol{U}^{(m,m+1)} \in \mathbb{R}^{r_{m-1}(p_m+1)\times(p_{m+1}+1)r_{m+1}}$, and the truncated SVD is computed,

$$\boldsymbol{U}^{(m,m+1)} \approx \boldsymbol{V\Sigma W}^\top, \quad \text{s.t.} \quad \left\|\boldsymbol{U}^{(m,m+1)} - \boldsymbol{V\Sigma W}^\top\right\|_F \le \varepsilon\left\|\boldsymbol{U}^{(m,m+1)}\right\|_F,$$

where $\boldsymbol{V} \in \mathbb{R}^{r_{m-1}(p_m+1)\times\hat{r}_m}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{\hat{r}_m\times\hat{r}_m}$, $\boldsymbol{W} \in \mathbb{R}^{(p_{m+1}+1)r_{m+1}\times\hat{r}_m}$, and $\|\cdot\|_F$ is the Frobenius norm. The new TT rank $\hat{r}_m$ is likely to differ from the old one r_m. After that, we rewrite $\boldsymbol{u}^{(m)}$ and $\boldsymbol{u}^{(m+1)}$ with $\boldsymbol{V}$ and $\boldsymbol{W}$, respectively, replace $r_m = \hat{r}_m$, and continue the iteration.

To ensure that the perturbations introduced to $\boldsymbol{u}^{(m,m+1)}$ and the whole tensor $\mathbf{u}$ coincide, we need to ensure that the interfaces $\boldsymbol{U}^{<m}$ and $\boldsymbol{U}^{>m+1}$ have orthonormal columns and rows, respectively. Fortunately, this requires only small-sized QR decompositions of matrices $\boldsymbol{u}^{|m\rangle}$

and $\boldsymbol{u}^{\langle m|}$ [53, 59] in the first iteration. Later, the SVD will provide orthogonal factors in $\boldsymbol{u}^{(m)}$ and $\boldsymbol{u}^{(m+1)}$ automatically.

This algorithm allows a fast adaptation of TT ranks toward the requested accuracy threshold. The price is, however, a larger degree of p in the complexity, since we perform a full search in both α_m and α_{m+1} in each step. The greedy-cross method [57] avoids this problem by maximizing the error over only $\mathcal{O}(rp)$ random entries among all $\mathcal{O}(r^2p^2)$ elements in $\boldsymbol{u}^{(m,m+1)}$. Here, instead of the neighboring block $\boldsymbol{u}^{(m+1)}$, it is the error that provides additional information and improves the approximation. However, for the KLE-PCE coefficient (2.8), a rank-adaptive procedure that does not involve the error or doubled blocks is possible.

3.5. Block TT-cross interpolation algorithm. Note that each call of (2.8) throws L values, corresponding to different $\ell = 1, \dots, L$. We may account for this in various ways. Since ℓ has the meaning of the reduced spatial variable, we may feature it as an additional dimension. But when we restrict the indices $\left\{\mathbb{I}^{(m-1)}, \alpha_m\right\}(\hat{i}_m) = \mathbb{I}^{(m)}$, we will remove some values of ℓ from consideration. Therefore, a vast majority of information cannot be used: we evaluate L values, but only a few of them will be used to improve the approximation. Another way is to run L independent cross (e.g., DMRG-cross) algorithms to approximate each $\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha})$ in its own TT format. Yet, this is also not very desirable. The TT ranks for the whole $\boldsymbol{\kappa}$ are usually comparable to the ranks of individual TT formats. Therefore, L cross algorithms consume almost L times more time than the single run.

A better approach is to store all $\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha})$ in the same TT representation, employing the idea of the *block* TT format [13]. The resulting method has a threefold advantage: all data in each call of (2.8) are assimilated, the algorithm adjusts the TT ranks automatically according to the given accuracy, and the output is returned as a single optimally compressed TT format convenient for further processing.

Assume that we have a procedure that, given an index $\alpha_1, \dots, \alpha_M$, throws L values $\mathbf{u}_\ell(\boldsymbol{\alpha})$, $\ell = 1, \dots, L$. When the tensor entries $\mathbf{u}_\ell(\mathbb{I}^{(m-1)}, \alpha_m, \mathbb{J}^{(m+1)})$ are evaluated, we modify (3.4) as follows:

$$\boldsymbol{y}^{(m)}(\alpha_m, \ell) = \widehat{\boldsymbol{U}}_{<m}^{-1} \cdot \mathbf{u}_\ell\left(\mathbb{I}^{(m-1)}, \alpha_m, \mathbb{J}^{(m+1)}\right) \cdot \widehat{\boldsymbol{U}}_{>m}^{-1}. \tag{3.5}$$

Now $\boldsymbol{y}^{(m)}$ is a four-dimensional tensor in the same way as in the DMRG-cross. We need to find a basis in $\boldsymbol{\alpha}$ that is best suited for all $\mathbf{u}_\ell$. Hence, we reshape $\boldsymbol{y}^{(m)}$ to the matrix $\boldsymbol{Y}^{(m)} \in \mathbb{R}^{r_{m-1}(p_m+1)\times L r_m}$ and compute the truncated SVD

$$\boldsymbol{Y}^{(m)} \approx \boldsymbol{u}^{|m\rangle}\boldsymbol{\Sigma}\boldsymbol{W}^\top, \qquad \boldsymbol{u}^{|m\rangle} \in \mathbb{R}^{r_{m-1}(p_m+1)\times \hat{r}_m}, \quad \boldsymbol{\Sigma} \in \mathbb{R}^{\hat{r}_m \times \hat{r}_m}, \quad \boldsymbol{W} \in \mathbb{R}^{L r_m \times \hat{r}_m}. \tag{3.6}$$

Again, the new rank $\hat{r}_m$ satisfies the Frobenius-norm error criterion and replaces r_m for the next iteration. In the backward iteration, we reshape $\boldsymbol{y}^{(m)}$ to $\boldsymbol{Y}^{(m)} \in \mathbb{R}^{L r_{m-1} \times (p_m+1) r_m}$ and take the right singular vectors to the new TT block,

$$\boldsymbol{Y}^{(m)} \approx \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{u}^{\langle m|}, \qquad \boldsymbol{W} \in \mathbb{R}^{L r_{m-1} \times \hat{r}_m}, \quad \boldsymbol{\Sigma} \in \mathbb{R}^{\hat{r}_m \times \hat{r}_m}, \quad \boldsymbol{u}^{\langle m|} \in \mathbb{R}^{\hat{r}_m \times (p_m+1) r_m}. \tag{3.7}$$

The whole procedure is summarized in the *Block TT-Cross algorithm*, Algorithm 1.

Algorithm 1. Block cross approximation of tensors in the TT format.

```
Require: A function to evaluate $\mathbf{u}_\ell(\alpha_1,\dots,\alpha_M)$, initial TT guess $\boldsymbol{u}^{(1)}(\alpha_1)\cdots\boldsymbol{u}^{(M)}(\alpha_M)$, relative accuracy threshold $\varepsilon$.
Ensure: Improved TT approximation $\boldsymbol{u}_\ell^{(1)}(\alpha_1)\boldsymbol{u}^{(2)}(\alpha_2)\cdots\boldsymbol{u}^{(M)}(\alpha_M)$.
 1: Initialize $\mathbb{I}^{(0)}=[],\quad \widehat{\boldsymbol{U}}_{<1}=1,\quad \mathbb{J}^{(M+1)}=[],\quad \widehat{\boldsymbol{U}}_{>M}=1$.
 2: for iteration $=1,2,\dots,n_{it}$ or until convergence do
 3:   for $m=1,2,\dots,M-1$ do {Forward sweep}
 4:     if iteration $>1$ then {All indices are available; assimilate the information}
 5:       Evaluate the tensors at cross indices and compute the common block by (3.5).
 6:       Compute the truncated SVD (3.6) with accuracy $\varepsilon$.
 7:     else {Warmup sweep: the indices are yet to be built}
 8:       Find QR decomposition $\boldsymbol{u}^{|m\rangle}=\boldsymbol{q}^{|m\rangle}\boldsymbol{R}$, $\left(\boldsymbol{q}^{|m\rangle}\right)^*\boldsymbol{q}^{|m\rangle}=\boldsymbol{I}$.
 9:       Replace $\boldsymbol{u}^{\langle m+1|}=\boldsymbol{R}\boldsymbol{u}^{\langle m+1|},\quad \boldsymbol{u}^{|m\rangle}=\boldsymbol{q}^{|m\rangle}$.
10:     end if
11:     Compute the prerestricted interface $\boldsymbol{V}^{\langle m|}=\widehat{\boldsymbol{U}}_{<m}\boldsymbol{u}^{\langle m|}$.
12:     Find local maxvol indices $\hat{i}_m=\mathtt{maxvol}\left(\boldsymbol{V}^{|m\rangle}\right)$.
13:     New indices $\mathbb{I}^{(m)}=\left\{\mathbb{I}^{(m-1)},\alpha_m\right\}(\hat{i}_m)$, interface $\widehat{\boldsymbol{U}}_{<m+1}=\boldsymbol{V}^{|m\rangle}(\hat{i}_m)\in\mathbb{R}^{r_m\times r_m}$.
14:   end for
15:   for $m=M,M-1,\dots,2$ do {Backward sweep}
16:     Evaluate the tensors at cross indices and compute the common block by (3.5).
17:     Compute the truncated SVD (3.7) with accuracy $\varepsilon$.
18:     Compute the prerestricted interface $\boldsymbol{V}^{|m\rangle}=\boldsymbol{u}^{|m\rangle}\widehat{\boldsymbol{U}}_{>m}$.
19:     Find local maxvol indices $\hat{j}_m=\mathtt{maxvol}((\boldsymbol{V}^{\langle m|})^\top)$.
20:     Restrict $\mathbb{J}^{(m)}=\left\{\alpha_m,\mathbb{J}^{(m+1)}\right\}(\hat{j}_m),\quad \widehat{\boldsymbol{U}}_{>m-1}=\boldsymbol{V}^{\langle m|}(\hat{j}_m)\in\mathbb{R}^{r_{m-1}\times r_{m-1}}$.
21:   end for
22:   Evaluate the first TT block $\boldsymbol{u}_\ell^{(1)}(\alpha_1)=\mathbf{u}_\ell\left(\alpha_1,\mathbb{J}^{(2)}\right)\widehat{\boldsymbol{U}}_{>1}^{-1}$.
23: end for
```

4. TT-structured calculations with PCE.

4.1. Computation of the PCE in the TT format via the cross interpolation.

Equipped with Algorithm 1, we may apply it to the PCE approximation, passing formula (2.8) as a function $u_\ell(\boldsymbol{\alpha})$ that evaluates tensor values on demand. The initial guess may even be a rank-1 TT tensor with all blocks populated by random numbers, since the cross iterations will adapt both the representation and TT ranks.

The complexity estimate can be written straightforwardly.

Statement 4.1. *The cost to compute $\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha})$ via the block TT-cross algorithm, Algorithm 1, is*

$$\mathcal{O}(r^2p(MN+NL)+r^3pL+r^3pL\cdot\min\{p,L\}).$$

The first two terms come from formula (2.8), and the last one is the complexity of SVDs (3.6) and (3.7).

Remark 4.2. It is unclear in general which term will dominate. For large N, we are typically expecting that it is the evaluation (3.5). However, if N is moderate (below 1000), but the rank is large (~ 100), the SVD consumes most of the time. For the whole algorithm, assuming also $L \sim M$, we can thus expect the $\mathcal{O}(n_{it} M^2 N p r^3)$ complexity, which is lower than $\mathcal{O}(MpL^3)$, which we could receive if we run independent cross algorithms for each ℓ.

As soon as the reduced PCE coefficients $\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha})$ are computed, the initial expansion (2.9) comes easily. Indeed, after the backward cross iteration, the ℓ index belongs to the first TT block, and we may let it play the role of the "zeroth" TT rank index,

$$\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\alpha}) = \sum_{s_1,\ldots,s_{M-1}} \boldsymbol{\kappa}^{(1)}_{\ell,s_1}(\alpha_1)\boldsymbol{\kappa}^{(2)}_{s_1,s_2}(\alpha_2)\cdots\boldsymbol{\kappa}^{(M)}_{s_{M-1}}(\alpha_M). \tag{4.1}$$

For $\ell = 0$ we extend this formula such that $\tilde{\boldsymbol{\kappa}}_0(\boldsymbol{\alpha})$ is the first identity vector $\boldsymbol{e}_0$; cf. (2.14). Now we collect the spatial components from (2.7) into the "zeroth" TT block,

$$\kappa^{(0)}(x) = \left[\kappa^{(0)}_\ell(x)\right]_{\ell=0}^{L} = \begin{bmatrix}\bar{\kappa}(x) & v_1(x) & \cdots & v_L(x)\end{bmatrix}; \tag{4.2}$$

then the PCE (2.5) writes as the following TT format:

$$\kappa(x,\boldsymbol{\alpha}) = \sum_{\ell,s_1,\ldots,s_{M-1}} \kappa^{(0)}_\ell(x)\boldsymbol{\kappa}^{(1)}_{\ell,s_1}(\alpha_1)\cdots\boldsymbol{\kappa}^{(M)}_{s_{M-1}}(\alpha_M). \tag{4.3}$$

4.2. Stiffness Galerkin operator in the TT format. With the full set $\mathcal{J}_{M,\boldsymbol{p}}$, we may benefit from the rank-1 separability of $\boldsymbol{\Delta}$, since each index α_m, β_m, ν_m varies independently of the others.

Lemma 4.3. *Given the PCE TT format* (4.3) *for the coefficient κ with the TT ranks $r(\kappa)$, the Galerkin operator* (2.13) *can be constructed in the TT format with the same ranks.*

Proof. Given the PCE (4.3) in the TT format, we split the whole sum over $\boldsymbol{\nu}$ in (2.11) into the individual variables,

$$\begin{aligned}\sum_{\nu\in\mathcal{J}_{M,\boldsymbol{p}}} \boldsymbol{\Delta}_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\nu}}\tilde{\boldsymbol{\kappa}}_\ell(\boldsymbol{\nu}) &= \sum_{s_1,\ldots,s_{M-1}} \boldsymbol{K}^{(1)}_{\ell,s_1}(\alpha_1,\beta_1)\boldsymbol{K}^{(2)}_{s_1,s_2}(\alpha_2,\beta_2)\cdots\boldsymbol{K}^{(M)}_{s_{M-1}}(\alpha_M,\beta_M),\\ \boldsymbol{K}^{(m)}(\alpha_m,\beta_m) &= \sum_{\nu_m=0}^{p_m} \Delta_{\alpha_m,\beta_m,\nu_m}\boldsymbol{\kappa}^{(m)}(\nu_m), \quad m=1,\ldots,M.\end{aligned} \tag{4.4}$$

A similar reduction of a large summation to one-dimensional operations arises in quantum chemistry [30]. Agglomerate $\boldsymbol{K}^{(x)}_\ell(i,j)$ from (2.10) to the "zeroth" TT block for the operator; then the TT representation for the operator writes with the same TT ranks as in $\tilde{\boldsymbol{\kappa}}$,

$$\mathbf{K} = \sum_{\ell,s_1,\ldots,s_{M-1}} \boldsymbol{K}^{(x)}_\ell \otimes \boldsymbol{K}^{(1)}_{\ell,s_1} \otimes \cdots \otimes \boldsymbol{K}^{(M)}_{s_{M-1}} \in \mathbb{R}^{(N\cdot\#\mathcal{J}_{M,\boldsymbol{p}})\times(N\cdot\#\mathcal{J}_{M,\boldsymbol{p}})}. \tag{4.5}$$ ■

One interesting property of the Hermite triples is that $\Delta_{\alpha,\beta,\nu} = 0$ if, e.g., $\nu > \alpha+\beta$. That is, if we set the same p for α, β, and ν, in the assembly of (2.13) we may miss some components, corresponding to $\alpha > p/2$, $\beta > p/2$. To compute the Galerkin operator exactly, it is reasonable to vary ν_m in the range $\{0,\ldots,2p\}$ and hence assemble $\tilde{\kappa}$ in the set $\mathcal{J}_{M,2\boldsymbol{p}}$. While in the sparse set it would inflate the storage of $\boldsymbol{\Delta}$ and $\mathbf{K}$ significantly, in the TT format it is feasible: the TT ranks do not depend on p, and the storage grows only linearly with p.

4.3. Computation of the solution function. Having solved the system (2.15), we obtain the PCE coefficients of the solution in the TT format,

$$u(x,\boldsymbol{\alpha}) = \sum_{s_0,\ldots,s_{M-1}} u^{(0)}_{s_0}(x)\boldsymbol{u}^{(1)}_{s_0,s_1}(\alpha_1)\cdots\boldsymbol{u}^{(M)}_{s_{M-1}}(\alpha_M). \tag{4.6}$$

Some statistics are computable directly from $u(x,\boldsymbol{\alpha})$, but generally we need a function in the initial random variables, $u(x,\boldsymbol{\theta})$. Since $\mathcal{J}_{M,\boldsymbol{p}}$ is a tensor product set, $u(x,\boldsymbol{\alpha})$ can be turned into $u(x,\boldsymbol{\theta})$ without changing the TT ranks, similarly to the construction of the Galerkin matrix in the previous subsection:

$$u(x,\boldsymbol{\theta}) = \sum_{s_0,\ldots,s_{M-1}} u^{(0)}_{s_0}(x)\left(\sum_{\alpha_1=0}^{p} h_{\alpha_1}(\theta_1)\boldsymbol{u}^{(1)}_{s_0,s_1}(\alpha_1)\right)\cdots\left(\sum_{\alpha_M=0}^{p} h_{\alpha_M}(\theta_M)\boldsymbol{u}^{(M)}_{s_{M-1}}(\alpha_M)\right). \tag{4.7}$$

4.4. Computation of statistics. In this section we discuss how to calculate some statistical outputs from the solution in the TT format, such as the mean, the (co)variance, and the probability of an event.

The *mean* value of u, in the same way as in the case of κ, can be derived as the PCE coefficient at $\boldsymbol{\alpha} = (0,\ldots,0)$, $\bar{u}(x) = u(x,0,\ldots,0)$. It requires no additional calculations.

The *covariance* is more complicated and requires both multiplication (squaring) and summation over $\boldsymbol{\alpha}$. By definition, the covariance reads

$$\begin{aligned}\operatorname{cov}_u(x,y) &= \int_{\mathbb{R}^M} (u(x,\boldsymbol{\theta}) - \bar{u}(x))\,(u(y,\boldsymbol{\theta}) - \bar{u}(y))\,\rho(\boldsymbol{\theta})d\boldsymbol{\theta} \\ &= \sum_{\substack{\boldsymbol{\alpha},\boldsymbol{\beta}\neq(0,\ldots,0),\\ \boldsymbol{\alpha},\boldsymbol{\beta}\in\mathcal{J}_{M,\boldsymbol{p}}}} u(x,\boldsymbol{\alpha})u(y,\boldsymbol{\beta})\int_{\mathbb{R}^M} H_{\boldsymbol{\alpha}}(\boldsymbol{\theta})H_{\boldsymbol{\beta}}(\boldsymbol{\theta})\rho(\boldsymbol{\theta})d\boldsymbol{\theta}.\end{aligned}$$

Knowing that $\int H_{\boldsymbol{\alpha}}(\boldsymbol{\theta})H_{\boldsymbol{\beta}}(\boldsymbol{\theta})\rho(\boldsymbol{\theta})d\boldsymbol{\theta} = \boldsymbol{\alpha}!\delta_{\boldsymbol{\alpha},\boldsymbol{\beta}}$, we take $\boldsymbol{u}_{s_0}(\boldsymbol{\alpha}) = \boldsymbol{u}^{(1)}_{s_0}(\alpha_1)\cdots\boldsymbol{u}^{(M)}(\alpha_M)$ from (4.6) and multiply it with the Hermite mass matrix (TT ranks do not change),

$$\boldsymbol{w}_{s_0}(\boldsymbol{\alpha}) := \boldsymbol{u}_{s_0}(\boldsymbol{\alpha})\sqrt{\boldsymbol{\alpha}!} = \sum_{s_1,\ldots,s_{M-1}} \left(\boldsymbol{u}^{(1)}_{s_0,s_1}(\alpha_1)\sqrt{\alpha_1!}\right)\cdots\left(\boldsymbol{u}^{(M)}_{s_{M-1}}(\alpha_M)\sqrt{\alpha_M!}\right), \tag{4.8}$$

and then take the scalar product $\boldsymbol{C} = [\boldsymbol{C}_{s_0,s_0'}]$, where $\boldsymbol{C}_{s_0,s_0'} = \left\langle \boldsymbol{w}_{s_0}, \boldsymbol{w}_{s_0'}\right\rangle$ with $\boldsymbol{w}$ defined in (4.8). Given the TT rank bound r for $u(x,\boldsymbol{\alpha})$, we deduce the $\mathcal{O}(Mpr^3)$ complexity of this step. After that, the covariance is given by the product of $\boldsymbol{C}$ with the spatial TT blocks,

$$\operatorname{cov}_u(x,y) = \sum_{s_0,s_0'=0}^{r_0} u^{(0)}_{s_0}(x)\boldsymbol{C}_{s_0,s_0'}u^{(0)}_{s_0'}(y), \tag{4.9}$$

where $u^{(0)}_{s_0}$ is the “zeroth” (spatial) TT block of the decomposition (4.6). Given N degrees of freedom for x, the complexity of this step is $\mathcal{O}(N^2r_0^2)$. Note that a low-rank tensor approximation of a large covariance matrix is very important, for example, in Kriging [52]. The *variance* is nothing else than the diagonal of the covariance, $\operatorname{var}_u(x) = \operatorname{cov}_u(x,x)$.

Other important outputs are the characteristic, level set functions, and the probability of a particular event [19].

Definition 4.4. *Let $\mathbb{S} \subset \mathbb{R}$ be a subset of real numbers.*

- *The* characteristic *function of u at $\mathbb{S}$ is defined pointwise for all $\boldsymbol{\theta} \in \mathbb{R}^M$ as follows:*

$$\chi_{\mathbb{S}}(x, \boldsymbol{\theta}) := \begin{cases} 1, & u(x, \boldsymbol{\theta}) \in \mathbb{S}, \\ 0, & u(x, \boldsymbol{\theta}) \notin \mathbb{S}. \end{cases} \tag{4.10}$$

- *The* level set *function reads $\mathcal{L}_{\mathbb{S}}(x, \boldsymbol{\theta}) := u(x, \boldsymbol{\theta})\chi_{\mathbb{S}}(x, \boldsymbol{\theta})$.*
- *The* probability *of $\mathbb{S}$ reads $\mathbb{P}_{\mathbb{S}}(x) = \int_{\mathbb{R}^M} \chi_{\mathbb{S}}(x, \boldsymbol{\theta})\rho(\boldsymbol{\theta})d\boldsymbol{\theta}$.*

The characteristic function can be computed using either the cross algorithm, Algorithm 1 (see also [5, 4]), which takes formula (4.10) as the function that evaluates a high-dimensional array χ at the indices in $x, \boldsymbol{\theta}$, or the Newton method for the sign function [19]. In both cases we may face a rapid growth of TT ranks during the cross or Newton iterations: the characteristic function is likely to have a discontinuity that is not aligned to the coordinate axes, and some of the singular values in the TT decomposition will decay very slowly. We face the same problem with increasing ranks during computing the level set functions and exceedance probabilities.

However, the probability is easier to compute, especially if it is relatively small. Using the same cross algorithm, we can directly compute the product $\hat{\chi}_{\mathbb{S}}(x, \boldsymbol{\theta}) = \chi_{\mathbb{S}}(x, \boldsymbol{\theta})\rho(\boldsymbol{\theta})$. The probability (at a fixed x) is then computed as a scalar product with the all-ones vector in the TT format, $\mathbb{P}_{\mathbb{S}}(x) = \langle \hat{\chi}, \mathbb{1} \rangle$. But if $\mathbb{P}$ is small, it means that most of the entries in $\hat{\chi}$ are small and do not inflate TT ranks, which might be the case for χ. Typically, the computation of small probabilities is used to predict the failure risk of a technical system. The event set has the form $\mathbb{S} = \{z \in \mathbb{R} : \ z > \tau\}$, and the probability is called the *exceedance* probability. This will be studied in numerical experiments.

5. Numerical experiments.

We verify the approach on the elliptic stochastic equation (1.1) in a two-dimensional L-shape domain, $x = (x_1, x_2) \in D = [-1, 1]^2 \backslash [0, 1]^2$. We pose zero Dirichlet boundary conditions and use the deterministic right-hand side $f = f(x) = 1$. The stochasticity is introduced in the diffusion coefficient $\kappa(x, \omega)$; we investigate log-normal and beta distributions for κ.

To generate the spatial mesh, we use the standard PDE Toolbox in MATLAB. We consider from 1 to 3 refinement levels of the spatial grid, denoted by R. The first refinement $R = 1$ yields 557 degrees of freedom, $R = 2$ gives 2145 points, and $R = 3$ corresponds to 8417 points. Since we have to store the stiffness matrices in a dense form, we cannot refine the grid further.

The KLE of both κ and γ is truncated to the same number of terms $L = M$.

All utilities related to the Hermite PCE were taken from the *sglib* [69], including discretization and solution routines in the sparse polynomial set $\mathcal{J}^{sp}_{M,p}$. However, to work with the TT format (for full $\mathcal{J}_{M,\boldsymbol{p}}$), we employ the *TT-Toolbox* [54]. The same polynomial orders are chosen in all variables, $\boldsymbol{p} = (p, \dots, p)$. We use the modules of *sglib* for low-dimensional stages and replace the parts corresponding to high-dimensional calculations by the TT algorithms. The block TT-cross Algorithm 1 is implemented in the procedure `amen_cross.m` from the TT-Toolbox, and the linear system (2.15) was solved via the Alternating Minimal Energy (AMEn) method [14], the procedure `amen_solve.m` from the companion package tAMEn [12].

Computations were conducted in MATLAB R2012a on a single core of the Intel Xeon X5650 CPU at 2.67GHz, provided by the Max Planck Institute in Magdeburg.

The accuracy of the coefficient and the solution was estimated using the Monte Carlo method with N_{mc} simulations. We approximate the average L_2-error as follows:

$$E_\kappa = \frac{1}{N_{mc}} \sum_{z=1}^{N_{mc}} \frac{\sqrt{\sum_{i=1}^{N} \left(\kappa(x_i, \boldsymbol{\theta}_z) - \kappa_*(x_i, \boldsymbol{\theta}_z)\right)^2}}{\sqrt{\sum_{i=1}^{N} \kappa_*^2(x_i, \boldsymbol{\theta}_z)}} \approx \int_{\mathbb{R}^M} \frac{\|\kappa(x, \boldsymbol{\theta}) - \kappa_*(x, \boldsymbol{\theta})\|_{L_2(D)}}{\|\kappa_*(x, \boldsymbol{\theta})\|_{L_2(D)}} \rho(\boldsymbol{\theta}) d\boldsymbol{\theta},$$

where $\{\boldsymbol{\theta}_z\}_{z=1}^{N_{mc}}$ are normally distributed random samples, $\{x_i\}_{i=1}^{N}$ are the spatial grid points, and $\kappa_*(x_i, \boldsymbol{\theta}_z) = \phi\left(\gamma(x_i, \boldsymbol{\theta}_z)\right)$ is the reference coefficient computed without using the PCE for ϕ. The same definition is used for the solution u, with $u_*(x, \boldsymbol{\theta}_z)$ being the solution of the deterministic PDE with the coefficient $\kappa_*(x, \boldsymbol{\theta}_z)$.

In addition we compare the statistics obtained with our approaches and the Monte Carlo method. For the mean and variance we use the same discrete approximation to the L_2-norm,

$$E_{\bar{u}} = \frac{\|\bar{u} - \bar{u}_*\|_{L_2(D)}}{\|\bar{u}_*\|_{L_2(D)}}, \qquad E_{\mathrm{var}_u} = \frac{\|\mathrm{var}_u - \mathrm{var}_{u*}\|_{L_2(D)}}{\|\mathrm{var}_{u*}\|_{L_2(D)}}.$$

To examine the computation of probabilities, we compute the exceedance probability. This task can be simplified by taking into account the maximum principle: the solution is convex w.r.t. x. We compute the maximizer of the mean solution, $x_{\max}: \ \bar{u}(x_{\max}) \geq \bar{u}(x) \ \forall x \in D$. Fix x to $x_{\max}$, and consider only the stochastic part $\mathbf{u}_{\max}(\boldsymbol{\theta}) = u(x_{\max}, \boldsymbol{\theta})$, and $\hat{u} = \bar{u}(x_{\max})$. Now, taking some $\tau > 1$, we compute

$$\mathbb{P} = \mathbb{P}\left(\mathbf{u}_{\max}(\boldsymbol{\theta}) > \tau \hat{u}\right) = \int_{\mathbb{R}^M} \chi_{\mathbf{u}_{\max}(\boldsymbol{\theta}) > \tau \hat{u}}(\boldsymbol{\theta}) \rho(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{5.1}$$

By $\mathbb{P}_*$ we will also denote the probability computed from the Monte Carlo method, and we estimate the error as $E_{\mathbb{P}} = \left|\mathbb{P} - \mathbb{P}_*\right| / \mathbb{P}_*$.

5.1. Log-normal distribution. Let

$$\kappa(x, \omega) = \exp\left(1 + \sigma \frac{\gamma(x, \omega)}{2}\right) + 10,$$

where $\gamma \sim \mathcal{N}(0, 1)$ is the standard normally distributed random field. The covariance function is taken Gaussian, $\mathrm{cov}_\kappa(x, y) = \exp(-\frac{\|x-y\|^2}{2 l_c^2})$, where l_c is the (isotropic) correlation length.

The default parameters are the following: number of KLE terms $M = 20$, polynomial order $p = 3$, correlation length $l_c = 1$, dispersion $\sigma = 0.5$, refinement level of the spatial grid $R = 1$, tensor approximation accuracy $\varepsilon = 10^{-4}$, and the number of Monte Carlo samples $N_{mc} = 10000$. Below we will vary each of these parameters, keeping the others fixed. For the computation of the probability (5.1) we use $\tau = 1.2$.

5.1.1. Verification of the block cross algorithm. Formula (2.8) can be evaluated for each KLE index ℓ independently, using existing cross approximation algorithms. We compare with the so-called DMRG cross method [58], which is conceptually the closest approach to our

Table 1
Performance of the block cross Algorithm 1 versus the DMRG cross [58].

M	Block cross CPU time, sec.	r_κ	DMRG crosses CPU time, sec.	r_κ	$\frac{\|\kappa_{DMRG} - \kappa_{Block}\|_F}{\|\kappa_{Block}\|_F}$
10	4.908	20	31.29	20	2.77e-5
15	10.36	27	114.9	27	2.24e-5
20	19.11	32	286.2	33	1.83e-4
30	49.89	39	1372.0	50	2.52e-4

Table 2
Detailed CPU times (sec.) versus p, log-normal distribution.

	TT (full index set $\mathcal{J}_{M,\boldsymbol{p}}$)			Sparse (index set $\mathcal{J}^{sp}_{M,p}$)		
p	T_κ	T_{op}	T_u	T_κ	T_{op}	T_u
1	9.6166	0.1875	1.7381	0.4525	0.2830	0.6485
2	14.6635	0.1945	2.9584	0.4954	3.2475	1.4046
3	19.1182	0.1944	3.4162	0.6887	1027.7	18.1263
4	24.4345	0.1953	4.2228	2.1597	—	—
5	30.9220	0.3155	5.3426	9.8382	—	—

Table 3
Performance versus p, log-normal distribution.

p	CPU time, sec.			r_κ	r_u	$r_{\hat{\chi}}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat{\chi}$				TT	Sparse	TT	Sparse	TT
1	11.54	1.38	0.23	32	42	1	3.75e-3	1.69e-1	9.58e-3	1.37e-1	0
2	17.81	5.14	0.32	32	49	1	1.35e-4	1.10e-1	4.94e-4	4.81e-2	0
3	22.72	1046	83.12	32	49	462	6.21e-5	2.00e-3	2.99e-4	5.29e-4	2.75e-4
4	28.85	—	69.74	32	50	416	6.24e-5	—	9.85e-5	—	1.21e-4
5	36.58	—	102.5	32	49	410	6.27e-5	—	9.36e-5	—	6.20e-4

Algorithm 1. In Table 1 we show the performance of the single run of Algorithm 1 (which gives the coefficient for all ℓ simultaneously) and of L DMRG crosses, followed by the summation of individual terms to the common representation. The last column in Table 1 presents the relative difference in Frobenius norm between the final coefficients of the form (4.3) (with x discretized), computed by the DMRG and block cross algorithms. We see that even if the TT ranks of the output are exactly the same, the times differ dramatically. This is because the ranks of individual components and of the whole coefficient are comparatively the same, and the DMRG approach requires roughly L times more operations. However, both approaches deliver the same data up to the approximation tolerance.

5.1.2. Experiment with the polynomial order p. First, we provide a detailed study of the computational time of each of the stages in the TT and sparse methods: construction of the coefficient (T_κ), construction of the operator (T_{op}), and the solution of the system (T_u). The results are shown in Table 2, and times are measured in seconds. The complexity of the cross algorithm, employed for the computation of κ in the TT format, grows linearly with p, since the TT ranks are stable w.r.t. p (see Table 3). However, it is much slower than the direct evaluation of the coefficients in the sparse set. This is mainly due to the SVDs, involving matrices whose sizes are on the order of hundreds.

Table 4
Performance versus M, log-normal distribution.

M	CPU time, sec.			r_κ	r_u	$r_{\hat\chi}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat\chi$				TT	Sparse	TT	Sparse	TT
10	6.401	6.143	1.297	20	39	70	2.00e-4	1.71e-1	3.26e-4	1.45e-1	2.86e-4
15	12.15	92.38	22.99	27	42	381	7.56e-5	1.97e-3	3.09e-4	5.41e-4	2.99e-4
20	21.82	1005	67.34	32	50	422	6.25e-5	1.99e-3	2.96e-4	5.33e-4	2.96e-4
30	52.92	48961	136.5	39	50	452	6.13e-5	9.26e-2	3.06e-4	5.51e-2	2.78e-4

Nevertheless, for the computation of the Galerkin matrix the situation is the opposite. In the TT format, the computations via formula (4.4) are very efficient, since they involve M products of $p^2 \times 2p$ matrices. In the sparse representation, we have to perform all $(\#\mathcal{J}^{sp}_{M,p})^3$ operations, which is very time-consuming. Since $\#\mathcal{J}^{sp}_{M,p}$ grows exponentially with p, we had to skip the cases with large p.

The solution stage is more simple, since the mean value preconditioner is quite efficient, both for the standard CG method with the sparse set and the AMEn method for the TT format. Again, the complexity of the TT solver grows linearly with p. The sparse solver works reasonably quickly as well, but it cannot be run before the matrix elements are computed;[1] hence it is also skipped for $p = 4, 5$.

In Table 3 we present the total CPU times required in both methods to find the solution u, the time for computing $\hat\chi$, maximal TT ranks of the coefficient (r_κ), the solution (r_u), and the weighted characteristic function ($r_{\hat\chi}$), as well as statistical errors in the coefficient (E_κ) and the solution (E_u). The probability $\mathbb{P}$ is presented only for the TT calculation. Since $\mathbb{P} \sim 10^{-4}$, 10000 simulations may be not enough to compute $\mathbb{P}$ with the Monte Carlo method. Below we present a dedicated test of the Monte Carlo approach.

5.1.3. Experiment with the KLE dimension M. The length of the truncated KLE is another crucial parameter of the stochastic PDE. The results are shown in Table 4.

We see that the accuracy of the TT approach is stable w.r.t. M, and the complexity grows mildly. Note, however, that the correlation length $l_c = 1$ is quite large and yields a fast decay of the KLE, such that $M = 20$ is actually enough for the accuracy 10^{-4}. The TT approach demonstrates stability w.r.t. the overapproximation at $M = 30$. This is not the case for the sparse approach: at high M the accuracy is lost. This is because $p = 3$ is not enough to transform the covariance (2.3) accurately. Since eigenvalues of the covariance decay rapidly, higher eigenpairs become strongly perturbed (the eigenvalues can even become negative), and a large error propagates to the PCE. In the full set, the maximal polynomial order is equal to pM, and the error of the covariance transform is negligible.

5.1.4. Experiment with the correlation length l_c. The Gaussian covariance function yields an exponential decay of the KLE coefficients [60, 42], but the actual rate is highly dependent on the correlation length [32, 41]. In this experiment, we study the range of lengths from 0.1 to 1.5. In order to have a sufficient accuracy for all values of l_c, we fix the

[1]It is sometimes advocated to avoid a construction of the matrix and to compute its elements only when they are needed in the matrix-vector product. It saves memory, but the computational time will be comparatively the same, since it is proportional to the number of operations.

Table 5
Performance versus l_c, log-normal distribution.

l_c	CPU time, sec.			r_κ	r_u	$r_{\hat{\chi}}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat{\chi}$				TT	Sparse	TT	Sparse	TT
0.1	216	55826	0.91	70	50	1	1.98e-2	1.98e-2	1.84e-2	1.84e-2	0
0.3	317	52361	41.8	87	74	297	3.08e-3	3.51e-3	2.64e-3	2.62e-3	8.59e-31
0.5	195	51678	58.1	67	74	375	1.49e-4	2.00e-3	2.58e-4	3.10e-4	6.50e-33
1.0	57.3	55178	97.3	39	50	417	6.12e-5	9.37e-2	3.18e-4	5.59e-2	2.95e-04
1.5	32.4	49790	121	31	34	424	3.24e-5	2.05e-1	4.99e-4	1.73e-1	7.50e-04

Table 6
Performance versus σ, log-normal distribution.

σ	CPU time, sec.			r_κ	r_u	$r_{\hat{\chi}}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat{\chi}$				TT	Sparse	TT	Sparse	TT
0.2	15.93	1008	0.348	21	31	1	5.69e-5	4.76e-5	4.19e-5	1.30e-5	0
0.4	18.72	968.3	0.341	29	42	1	6.88e-5	8.04e-4	1.40e-4	2.14e-4	0
0.5	21.23	970.1	79.96	32	49	456	6.19e-5	2.02e-3	3.05e-4	5.45e-4	2.95e-4
0.6	24.08	961.5	24.72	34	57	272	9.12e-5	4.42e-3	6.14e-4	1.16e-3	2.30e-3
0.8	31.69	969.1	67.93	39	66	411	4.40e-4	8.33e-2	2.02e-3	2.90e-2	8.02e-2
1.0	50.67	1071	48.44	44	82	363	1.73e-3	4.10e-1	4.96e-3	3.08e-1	9.17e-2

dimension $M = 30$. The results are presented in the same layout as before in Table 5.

In the TT approach, we see a clear decrease of the computational complexity and the error with growing covariance length. This is because the SVD approximation in the TT format automatically reduces the storage w.r.t. the latter (less important) variables if the KLE decay is fast enough. The TT errors reflect the amount of information discarded in the truncated KLE tail, which is large for small l_c and small otherwise.

The errors in the sparse approach behave similarly until $l_c = 0.5$, but for $l_c = 1, 1.5$ the dimension $M = 30$ is too large, and the instability w.r.t. the overapproximation occurs.

With fixed M, the exceedance probability is very sensitive to the correlation length. Truncating the KLE, we reduce the total variance of the random field. For a (quasi)-Gaussian distribution, a small perturbation of the variance has a small effect on the integral over the peak region but may have a very large relative effect on the tail region, which corresponds to the small exceedance probability.

5.1.5. Experiment with the dispersion σ.

The variance of the normally distributed field $\gamma(x, \omega)$ is equal to σ^2. Since it enters κ under the exponential, it influences the variance of κ significantly. In Table 6 we vary σ from 0.2 to 1 and track the performance of the methods. As expected, the computational complexity grows with σ, as does the contrast in the coefficient. However, we were able to perform all computations for each value of σ.

5.1.6. Experiment with the spatial grid refinement R.

Since the efforts of dealing with the full spatial matrix grow significantly with the grid refinement, in this test we limit ourselves to $M = 10$. The principal observations from Table 7 are that the TT rank and the accuracy[2]

[2]Note that the errors are estimated via the Monte Carlo method on the same grids; thus they show the accuracy of the PCE approximation—not the spatial discretization.

Table 7

Performance versus R, log-normal distribution. The left column shows the number of spatial degrees of freedom (#DoF) for $R = 1, 2, 3$.

#DoF	CPU time, sec			r_κ	r_u	$r_{\hat\chi}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat\chi$				TT	Sparse	TT	Sparse	TT
557	6.40	6.09	1.29	20	39	71	2.00e-4	1.71e-1	3.26e-4	1.45e-1	2.86e-4
2145	8.98	13.7	1.17	20	39	76	1.74e-4	1.89e-3	3.33e-4	5.69e-4	2.90e-4
8417	357	171	0.84	20	40	69	1.65e-4	1.88e-3	3.24e-4	5.64e-4	2.93e-4

Table 8

Verification of the Monte Carlo method, log-normal distribution.

N_{mc}	T_{MC}, sec.	$E_{\bar u}$	$E\mathrm{var}_u$	$\mathbb{P}_*$	$E_{\mathbb{P}}$	TT results	
10^2	0.6398	9.23e-3	1.49e-1	0	∞	T_{solve}	96.89 sec.
10^3	6.1867	1.69e-3	5.97e-2	0	∞	$T_{\hat\chi}$	157.0 sec.
10^4	$6.1801\cdot 10^1$	5.81e-4	7.12e-3	4.00e-4	5.53e-1	r_κ	39
10^5	$6.2319\cdot 10^2$	2.91e-4	2.91e-3	4.10e-4	5.15e-1	r_u	50
10^6	$6.3071\cdot 10^3$	1.23e-4	9.76e-4	4.60e-4	3.51e-1	$r_{\hat\chi}$	432
						$\mathbb{P}$	6.214e-4

are stable w.r.t. the grid refinement. Therefore, we may expect that the TT approach will also be efficient for finer grids if we find an efficient way to deal with the spatial dimension. Research on *nonintrusive* stochastic Galerkin methods, addressing this issue, has recently begun [22], and we plan to adopt it in the TT framework in the future.

5.1.7. Comparison with the Monte Carlo method.

For the Monte Carlo test, we prepare the TT solution with parameters $p = 5$ and $M = 30$. The results are presented in Table 8. In the left part of the table we show the performance of the Monte Carlo method with different numbers of simulations: total CPU time (T_{MC}), errors in the mean and variance of u, and a small exceedance probability with its error. The right part contains the results of the TT approach: the aggregated CPU time of construction of the coefficient, operator and solution (T_{solve}), the time to compute the weighted characteristic function ($T_{\hat\chi}$), TT ranks of all data, and the probability calculated from $\hat\chi$.

We see that the cost of the TT method is comparable with the cost of 40000 Monte Carlo tests. Although the Monte Carlo method with 40000 realizations provides a sufficiently good approximation of the mean, the accuracy of the variance, and especially the exceedance probability, is much worse than in the TT approach. Therefore, the tensor product methods can be recommended as a competitive alternative to classical techniques for computing exceedance probabilities.

5.2. Beta distribution.

The Hermite expansion (2.2) of the exp function in the log-normal case yields the coefficients of the form $\phi_i = \frac{c}{i!}$. Therefore, the PCE coefficient formula (2.6) resolves to a direct product of univariate functions of $\alpha_1, \ldots, \alpha_M$, and the tensor format of the PCE can be constructed explicitly [18]. To demonstrate the generality of the cross algorithm,

Table 9

Performance versus p, beta distribution.

p	CPU time, sec.			r_κ	r_u	$r_{\hat\chi}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat\chi$				TT	Sparse	TT	Sparse	TT
1	21.40	1.382	0.059	64	49	1	2.24e-3	5.13e-2	1.14e-2	2.50e-2	0
2	39.87	5.301	0.100	65	50	1	1.92e-4	5.50e-3	7.67e-4	1.28e-3	0
3	57.16	1000	70.98	65	50	445	9.07e-5	1.76e-3	5.01e-4	1.06e-3	1.88e-4
4	76.22	—	21.18	65	50	416	8.81e-5	—	1.41e-4	—	9.84e-5
5	100.6	—	119.7	65	50	428	8.89e-5	—	1.10e-4	—	1.23e-4

Table 10

Performance versus M, beta distribution.

M	CPU time, sec.			r_κ	r_u	$r_{\hat\chi}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat\chi$				TT	Sparse	TT	Sparse	TT
10	9.777	5.796	0.942	34	40	39	1.70e-4	1.65e-3	5.18e-4	1.08e-3	1.95e-4
15	26.46	90.34	25.16	50	48	374	1.03e-4	1.73e-3	4.96e-4	1.08e-3	1.94e-4
20	56.92	986.2	59.57	65	50	413	9.15e-5	1.80e-3	5.08e-4	1.09e-3	1.88e-4
30	156.7	55859	147.9	92	50	452	7.75e-5	7.01e-2	5.12e-4	4.53e-2	1.85e-4

Table 11

Performance versus l_c, beta distribution.

l_c	CPU time, sec.			r_κ	r_u	$r_{\hat\chi}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat\chi$				TT	Sparse	TT	Sparse	TT
0.1	665.8	55923	0.91	90	48	1	8.7e-3	8.77e-3	7.9e-3	7.92e-3	0
0.3	2983	53783	1.49	177	74	1	1.5e-3	2.02e-3	1.2e-3	1.30e-3	0
0.5	1138	54297	100	132	74	403	1.5e-4	1.71e-3	2.9e-4	8.21e-4	2.47e-23
1.0	158.8	56545	153	92	50	463	7.8e-5	6.92e-2	5.1e-4	4.47e-2	1.96e-04
1.5	62.20	55848	89.5	75	42	409	6.9e-5	7.85e-2	8.3e-4	4.56e-2	2.20e-03

Table 12

Performance versus R, beta distribution.

#DoF	CPU time, sec			r_κ	r_u	$r_{\hat\chi}$	E_κ		E_u		$\mathbb{P}$
	TT	Sparse	$\hat\chi$				TT	Sparse	TT	Sparse	TT
557	9.73	5.94	0.94	34	40	39	1.70e-4	1.65e-3	5.21e-4	1.08e-3	1.95e-4
2145	36.2	12.7	0.77	34	41	41	1.56e-4	1.64e-3	5.19e-4	1.08e-3	1.97e-4
8417	378	162	1.12	34	40	43	1.53e-4	1.62e-3	5.07e-4	1.06e-3	1.96e-4

we also consider a more exotic beta-distributed coefficient,

$$\kappa(x,\omega) = \mathcal{B}^{-1}_{5,2}\left(\frac{1+\operatorname{erf}\left(\frac{\gamma(x,\omega)}{\sqrt{2}}\right)}{2}\right)+1, \qquad \text{where} \quad \mathcal{B}_{a,b}(z) = \frac{1}{B(a,b)}\int_0^z t^{a-1}(1-t)^{b-1}dt.$$

For the purpose of computing ϕ_i, the function $\mathcal{B}_{5,2}(z)$ is inverted by the Newton method. Again, the covariance function is $\operatorname{cov}_\kappa(x,y) = \exp(-\frac{\|x-y\|^2}{2l_c^2})$.

Since this distribution varies more strongly than the log-normal one, for the computation of the probability (5.1) we use larger $\tau = 1.4$. All other parameters are the same as in the experiments with the log-normal coefficient. The performance of both the TT and sparse

Table 13
Verification of the Monte Carlo method, beta distribution.

N_{mc}	T_{MC}, sec.	$E_{\bar{u}}$	$E\mathrm{var}_u$	$\mathbb{P}_*$	$E_{\mathbb{P}}$	TT results	
10^2	0.9428	9.12e-3	1.65e-1	0	∞	T_{solve}	278.4014 sec.
10^3	9.5612	1.04e-3	6.04e-2	0	∞	$T_{\hat{\chi}}$	179.4764 sec.
10^4	$8.849\cdot10^1$	4.38e-4	5.56e-3	0	∞	r_κ	92
10^5	$8.870\cdot10^2$	2.49e-4	3.06e-3	7.00e-5	6.80e-1	r_u	50
10^6	$8.883\cdot10^3$	8.16e-5	8.56e-4	1.07e-4	9.94e-2	$r_{\hat{\chi}}$	406
						$\mathbb{P}$	1.1765e-04

approaches in the case of the beta distribution is shown in Tables 9, 10, 11, 12, and 13 for p, M, l_c, the spatial grid level, and the Monte Carlo tests, respectively.

We see the same behavior as in the log-normal case. The only significant difference is the lower error of the sparse method in the case $M = 10$, $R_l = 1$, which is 1.08e-3 for the beta distribution and 1.45e-1 for the log-normal one.

6. Conclusion.

We have developed the new block TT cross algorithm to compute the TT approximation of the polynomial chaos expansion of a random field with the tensor product set of polynomials, where the polynomial degrees are bounded individually for each random variable. The random field can be given as a transformation of a Gaussian field by an arbitrary smooth function. The new algorithm builds the TT approximation of the PCE in a black box manner. Compared to the previously existing cross methods, the new approach assimilates all KLE terms simultaneously, which reduces the computational cost significantly.

The uncertain (diffusion) coefficient in the elliptic PDE is approximated via PCE. We show that the tensor product polynomial set allows a very efficient construction of the stochastic Galerkin matrix in the TT format, provided the coefficient is precomputed in the TT format. Interestingly, we can even compute the Galerkin matrix exactly by preparing the coefficient with two times larger polynomial orders than those employed for the solution. In the TT format, we can store the Galerkin matrix in the dense form, since the number of the TT elements $\mathcal{O}(Mp^2r^2)$ is feasible for $p \sim 10$. This also means that other polynomial families, such as the Chebyshev and Laguerre, may be used straightforwardly.

The Galerkin matrix defines a large linear system on the PCE coefficients of the solution of the stochastic equation. We solve this system in the TT format via the alternating minimal energy algorithm and calculate the postprocessing of the solution, such as the mean, variance, and exceedance probabilities.

We demonstrate that with the new TT approach we can go to a larger number of random variables (e.g., $M = 30$) used in the PCE (larger stochastic dimension) and take a larger order of the polynomial approximation in the stochastic space ($p = 5$) on a usual PC desktop computer. For all stages of numerical experiments (computation of the coefficient, operator, solution, and statistical functionals) we report the computational times and the storage costs (TT ranks) and show that they stay moderate in the investigated range of parameters.

In particular, the TT ranks do not grow with the polynomial degree p. This remains in sharp contrast to the traditional sparse polynomial approximation, where the total polynomial degree is bounded. The cardinality of this sparse polynomial set grows exponentially with p, but the tensor product decomposition is not possible anymore. This renders the total computational cost of the sparse PCE approach higher than the cost of the TT method. Besides, the tensor product PCE is more accurate than the expansion in the sparse set due to a larger total polynomial degree. Comparison with the classical Monte Carlo method shows that the TT methods can compute the exceedance probabilities more accurately, since the TT format approximates the whole stochastic solution implicitly.

Several directions of research can be pursued in the future. Currently, we store both the matrix and the inverted mean-field preconditioner in the dense form. This imposes rather severe restrictions on the spatial discretization. The spatial part of the Galerkin matrix must be dealt with in a more efficient way.

The characteristic function is difficult to compute in the TT format, since it requires rather large ranks to achieve a satisfactory accuracy. However, the TT approximation might be useful for the reduction of variance (e.g., as a *control variate*) in Monte Carlo methods. Such an approach has already been applied in the framework of sparse grids [48] and showed its computational advantages.

With the tensor product methods the stochastic collocation approach seems very attractive [33]. We may introduce quite large discretization grids in each random variable θ_m: additional data compression can be achieved with the QTT approximation [31]. It is important that the deterministic problems are decoupled in the stochastic collocation. The cross algorithms can become an efficient nonintrusive approach to stochastic equations.

Acknowledgment. We would like to thank Elmar Zander for his assistance in the usage of the stochastic Galerkin library *sglib*.

REFERENCES

[1] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM J. Numer. Anal., 45 (2007), pp. 1005–1034.
[2] I. Babuška, R. Tempone, and G. E. Zouraris, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM J. Numer. Anal., 42 (2004), pp. 800–825.
[3] J. Bäck, F. Nobile, L. Tamellini, and R. Tempone, *Stochastic spectral Galerkin and collocation methods for PDEs with random coefficients: A numerical comparison*, in Spectral and High Order Methods for Partial Differential Equations, Springer, New York, 2011, pp. 43–62.
[4] J. Ballani and L. Grasedyck, *Hierarchical Tensor Approximation of Output Quantities of Parameter-Dependent PDEs*, Tech. report 385, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Aachen, Germany, 2014.
[5] J. Ballani, L. Grasedyck, and M. Kluge, *Black box approximation of tensors in hierarchical Tucker format*, Linear Algebra Appl., 438 (2013), pp. 639–657.
[6] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.
[7] M. Bebendorf, *Adaptive cross approximation of multivariate functions*, Construct. Approx., 34 (2011), pp. 149–179.
[8] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.
[9] G. Blatman and B. Sudret, *An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis*, Probab. Engrg. Mech., 25 (2010), pp. 183–197.
[10] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.
[11] A. Chkifa, A. Cohen, and Ch. Schwab, *Breaking the curse of dimensionality in sparse polynomial approximation of parametric PDEs*, J. Math. Pures Appl., 103 (2015), pp. 400–428.
[12] S. Dolgov, *tAMEn*, https://github.com/dolgov/tamen.

[13] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, *Computation of extreme eigenvalues in higher dimensions using block tensor train format*, Comput. Phys. Comm., 185 (2014), pp. 1207–1216.

[14] 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.

[15] A. Doostan, A. Validi, and G. Iaccarino, *Non-intrusive low-rank separated approximation of high-dimensional stochastic models*, Comput. Methods Appl. Mech. Engrg., 263 (2013), pp. 42–55.

[16] M. Eigel, C. J. Gittelson, Ch. Schwab, and E. Zander, *Adaptive stochastic Galerkin FEM*, Comput. Methods Appl. Mech. Engrg., 270 (2014), pp. 247–269.

[17] O. G. Ernst and E. Ullmann, *Stochastic Galerkin matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1848–1872.

[18] M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies, and Ph. Wähnert, *Efficient low-rank approximation of the stochastic Galerkin matrix in tensor formats*, Comput. Math. Appl., 67 (2014), pp. 818–829.

[19] 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, Springer, New York, 2013, pp. 31–56.

[20] M. Fannes, B. Nachtergaele, and R. F. Werner, *Finitely correlated states on quantum spin chains*, Comm. Math. Phys., 144 (1992), pp. 443–490.

[21] J. Galvis and M. Sarkis, *Approximating infinity-dimensional stochastic Darcy's equations without uniform ellipticity*, SIAM J. Numer. Anal., 47 (2009), pp. 3624–3651.

[22] L. Giraldi, A. Litvinenko, D. Liu, H. G. Matthies, and A. Nouy, *To be or not to be intrusive? The solution of parametric and stochastic equations—the "plain vanilla" Galerkin case*, SIAM J. Sci. Comput., 36 (2014), pp. A2720–A2744.

[23] C. J. Gittelson, *Stochastic Galerkin discretization of the log-normal isotropic diffusion problem*, Math. Models Methods Appl. Sci., 20 (2010), pp. 237–263.

[24] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms, Applications, V. Olshevsky and E. Tyrtyshnikov, eds., World Scientific, Hackensack, NJ, 2010, pp. 247–256.

[25] S. A. Goreinov and E. E. Tyrtyshnikov, *The maximal-volume concept in approximation by low-rank matrices*, in Structured Matrices in Mathematics, Computer Science, and Engineering I (Boulder, CO, 1999), Contemp. Math. 280, AMS, Providence, RI, 2001, pp. 47–51.

[26] I. G. Graham, F. Y. Kuo, D. Nuyens, R. Scheichl, and I. H. Sloan, *Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications*, J. Comput. Phys., 230 (2011), pp. 3668–3694.

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

[28] M. Griebel, *Sparse grids and related approximation schemes for higher dimensional problems*, in Foundations of Computational Mathematics, Santander 2005, London Math. Soc. Lecture Note Ser. 331, Cambridge University Press, Cambridge, UK, 2006, pp. 106–161.

[29] A. Keese, *Numerical Solution of Systems with Stochastic Uncertainties. A General Purpose Framework for Stochastic Finite Elements*, Ph.D. thesis, TU Braunschweig, Brunswick, Germany, 2004.

[30] V. Khoromskaia and B. N. Khoromskij, *Grid-based lattice summation of electrostatic potentials by assembled rank-structured tensor approximation*, Comput. Phys. Commun., 185 (2014), pp. 3162–3174.

[31] B. N. Khoromskij, *$\mathcal{O}(d \log n)$–quantics approximation of N–d tensors in high-dimensional numerical modeling*, Construct. Approx., 34 (2011), pp. 257–280.

[32] B. N. Khoromskij, A. Litvinenko, and H. G. Matthies, *Application of hierarchical matrices for computing the Karhunen-Loève expansion*, Computing, 84 (2009), pp. 49–67.

[33] B. N. Khoromskij and I. V. Oseledets, *Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs*, Comput. Methods Appl. Math., 10 (2010), pp. 376–394.

[34] B. N. Khoromskij and I. V. Oseledets, *QTT approximation of elliptic solution operators in higher dimensions*, Russian J. Numer. Anal. Math. Model., 26 (2011), pp. 303–322.

[35] B. N. Khoromskij and Ch. Schwab, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comput., 33 (2011), pp. 364–385.

[36] D. Kressner and Ch. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316.

[37] D. Kressner and Ch. Tobler, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Comput. Methods Appl. Math., 11 (2011), pp. 363–381.

[38] A. Kunoth and C. Schwab, *Analytic regularity and GPC approximation for control problems constrained by linear parametric elliptic and parabolic PDEs*, SIAM J. Control Optim., 51 (2013), pp. 2442–2471.

[39] F. Y. Kuo, Ch. Schwab, and I. H. Sloan, *Multi-level quasi-Monte Carlo finite element methods for a class of elliptic PDEs with random coefficients*, Found. Comput. Math., 15 (2015), pp. 411–449.

[40] O. P. Le Maître and O. M. Knio, *Spectral Methods for Uncertainty Quantification*, Scientific Computation, Springer, New York, 2010.

[41] A. Litvinenko and H. G. Matthies, *Sparse data representation of random fields*, Proc. Appl. Math. Mech., 9 (2009), pp. 587–588.

[42] M. Marcel and Ch. Schwab, *Sparse high order FEM for elliptic sPDEs*, Comput. Methods Appl. Mech. Engrg., 198 (2009), pp. 1149–1170.

[43] H. G. Matthies, *Uncertainty quantification with stochastic finite elements*, in Encyclopedia of Computational Mechanics. Part 1. Fundamentals, John Wiley and Sons, Chichester, UK, 2007.

[44] H. G. Matthies and A. Keese, *Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 1295–1331.

[45] H. G. Matthies, A. Litvinenko, O. Pajonk, B. V. Rosić, and E. Zander, *Parametric and uncertainty computations with tensor product representations*, in Uncertainty Quantification in Scientific Computing, IFIP Advances in Information and Communication Technology 377, Springer, Berlin, Heidelberg, 2012, pp. 139–150.

[46] H. G. Matthies and E. Zander, *Solving stochastic systems with low-rank tensor compression*, Linear Algebra Appl., 436 (2012), pp. 3819–3838.

[47] A. Mugler and H.-J. Starkloff, *On elliptic partial differential equations with random coefficients*, Stud. Univ. Babes-Bolyai Math., 56 (2011), pp. 473–487.

[48] F. Nobile, L. Tamellini, F. Tesei, and R. Tempone, *An Adaptive Sparse Grid Algorithm for Elliptic PDEs with Log-Normal Diffusion Coefficient*, MATHICSE Technical Report 04, EPFL, Lausanne, Switzerland, 2015.

[49] F. Nobile, R. Tempone, and C. G. Webster, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM J. Numer. Anal., 46 (2008), pp. 2309–2345.

[50] A. Nouy, *A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 4521–4537.

[51] A. Nouy, *Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems*, Arch. Comput. Methods Engrg., 17 (2010), pp. 403–434.

[52] W. Nowak and A. Litvinenko, *Kriging and spatial design accelerated by orders of magnitude: Combining low-rank covariance approximations with FFT-techniques*, Math. Geosci., 45 (2013), pp. 411–435.

[53] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

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

[55] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759.

[56] I. V. Oseledets and E. E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.

[57] D. V. Savostyanov, *Quasioptimality of maximum–volume cross interpolation of tensors*, Linear Algebra Appl., 458 (2014), pp. 217–244.

[58] D. V. Savostyanov and I. V. Oseledets, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, in Proceedings of the 7th International Workshop on Multidimensional Systems (nDS), IEEE, Washington, DC, 2011, pp. 1–8.

[59] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys., 326 (2011), pp. 96–192.

[60] Ch. Schwab and R. A. Todor, *Karhunen-Loève approximation of random fields by generalized fast multipole methods*, J. Comput. Phys., 217 (2006), pp. 100–122.

[61] B. Sousedík and R. Ghanem, *Truncated hierarchical preconditioning for the stochastic Galerkin FEM*, Internat. J. Uncertainty Quant., 4 (2014), pp. 333–348.

[62] A. L. Teckentrup, R. Scheichl, M. B. Giles, and E. Ullmann, *Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients*, Numer. Math., 125 (2013), pp. 569–600.

[63] R. A. Todor and Ch. Schwab, *Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients*, IMA J. Numer. Anal., 27 (2007), pp. 232–261.

[64] E. Ullmann, *A Kronecker product preconditioner for stochastic Galerkin finite element discretizations*, SIAM J. Sci. Comput., 32 (2010), pp. 923–946.

[65] E. Ullmann, H. C. Elman, and O. G. Ernst, *Efficient iterative solvers for stochastic Galerkin discretizations of log-transformed random diffusion problems*, SIAM J. Sci. Comput., 34 (2012), pp. A659–A682.

[66] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356.

[67] N. Wiener, *The homogeneous chaos*, Amer. J. Math., 60 (1938), pp. 897–936.

[68] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.

[69] E. Zander, *Stochastic Galerkin Library*, http://github.com/ezander/sglib (2008).

[70] E. Zander, *Tensor Approximation Methods for Stochastic Problems*, Ph.D. thesis, Dissertation, Technische Universität Braunschweig, Brunswick, Germany, 2013.