SIAM/ASA J. UNCERTAINTY QUANTIFICATION
Vol. 3, pp. 622–649

Low-Rank Solution of Unsteady Diffusion Equations with Stochastic Coefficients*

Peter Benner†, Akwum Onwunta‡, and Martin Stoll†

Abstract. We study the solution of linear systems resulting from the discretization of unsteady diffusion equations with stochastic coefficients. In particular, we focus on those linear systems that are obtained using the so-called stochastic Galerkin finite element method (SGFEM). These linear systems are usually very large with Kronecker product structure, and thus solving them can be both time- and computer memory-consuming. Under certain assumptions, we show that the solution of such linear systems can be approximated with a vector of low tensor rank. We then solve the linear systems using low-rank preconditioned iterative solvers. Numerical experiments demonstrate that these low-rank preconditioned solvers are effective, especially when the fluctuations in the random data are not too large relative to their mean values.

Key words. stochastic Galerkin system, low-rank solution, preconditioning, iterative methods

AMS subject classifications. 35R60, 60H15, 60H35, 65N22, 65F10

DOI. 10.1137/130937251

1. Introduction.

Many problems in science and engineering are modeled using partial differential equations (PDEs). One such important model is the diffusion equation which arises in, for instance, fluid flow and transport of chemicals in heterogeneous porous media (see, e.g., [6], [22]), as well as in temperature prediction of biological bodies [29], etc. More often than not, the diffusion equation is modeled deterministically. However, in the transport models for groundwater flows, for example, it is only possible to measure the hydraulic conductivity at a limited number of spatial locations; this leads to uncertainty in the groundwater flow simulations [6]. Hence, it is reasonable to model the hydraulic conductivity as a random field. This, in turn, implies that the solution to the resulting stochastic model is necessarily also a random field. There is, therefore, the need to quantify the uncertainty in the solution of the model.

Generally, in order to solve PDEs with stochastic inputs, three competing methods are standard in the literature: the Monte Carlo method (MCM), the stochastic collocation method (SCM), and the spectral stochastic Galerkin finite element method (SGFEM); see, e.g., [6], [1], [2], [13], [11]. In contrast to the MCM and SCM (both of which are based on stochastic sampling), the SGFEM is a nonsampling approach which transforms a PDE with uncertain inputs into a large system of coupled deterministic PDEs. Despite the curse of dimensionality

*Received by the editors September 16, 2013; accepted for publication (in revised form) March 18, 2015; published electronically August 6, 2015.
http://www.siam.org/journals/juq/3/93725.html

†Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany (benner@mpi-magdeburg.mpg.de, stollm@mpi-magdeburg.mpg.de).

‡Corresponding author. Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany (onwunta@mpi-magdeburg.mpg.de). The research of this author was supported by a research grant from the International Max Planck Research School (IMPRS) for Advanced Methods in Process and System Engineering (Magdeburg).

problem associated with the SGFEM, the beauty of the approach lies in, among other things, the ease with which it lends itself to the computation of such quantities of interest as the moments and the density of the solution.

In the past two decades, research on the solution of diffusion equations with random inputs using the SGFEM has been focused mainly on developing solvers for steady-state problems; see, e.g., [1], [8], [27], [10], [22], etc. Time-dependent problems have not yet received adequate attention. A few attempts in this direction include [29], [19], [23], [30]. Unlike the steady-state problem, the time-dependent model problem presents the additional challenge of solving a large coupled linear system for each time step. As opposed to the literature above on unsteady diffusion problems, the main aim of this paper is to tackle this dimensionality problem using low-rank iterative solvers studied in [18] in the framework of parametrized linear systems. The rest of the paper is organized as follows. In section 2, we give some basic notions on which we shall rely in the rest of the paper. Next, we present our model problem and provide an overview of its discretization in section 3. Since our approach is based on low-rank approximation, we first show the existence of a low-rank approximation of the solution to the stochastic Galerkin system in section 4 before proceeding to discuss our preconditioned low-rank iterative solvers in section 5. In section 6, we present numerical results to illustrate that, provided the standard deviation of the random input is relatively low, these low-rank iterative solvers are effective especially with respect to the reduction of the computational time and memory requirements of large-scale simulations. Finally, we draw some conclusions in section 7 based on our findings in the paper.

2. Basic notions and definitions.

Let the triplet $(\Omega, \mathcal{F}, \mathbb{P})$ be a complete probability space, where Ω is a sample space of events. Here, $\mathcal{F}$ denotes a σ-algebra on Ω and is endowed with an appropriate probability measure $\mathbb{P}$. Moreover, let $\mathcal{D} \subset \mathbb{R}^d$ with $d \in \{1, 2, 3\}$ be a bounded open set with Lipschitz boundary $\partial\mathcal{D}$.

Definition 2.1. *A mapping $\kappa : \mathcal{D} \times \Omega \to \mathbb{R}$ is called a random field if for each fixed $\mathbf{x} \in \mathcal{D}$, $\kappa(\mathbf{x}, \cdot)$ is a random variable with respect to $(\Omega, \mathcal{F}, \mathbb{P})$.*

We denote the mean of κ at a point $\mathbf{x} \in \mathcal{D}$ by $\bar{\kappa}(\mathbf{x}) := \langle \kappa(\mathbf{x}, \cdot)\rangle$. The covariance of κ at $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ is given by

$$\mathrm{Cov}_\kappa(\mathbf{x}, \mathbf{y}) := \langle (\kappa(\mathbf{x}, \cdot) - \bar{\kappa}(\mathbf{x}))(\kappa(\mathbf{y}, \cdot) - \bar{\kappa}(\mathbf{y}))\rangle . \tag{2.1}$$

Note that the variance $\mathrm{Var}(\kappa) = \sigma_\kappa^2$ of κ at $\mathbf{x} \in \mathcal{D}$ is obtained if we set $\mathbf{x} = \mathbf{y}$ in (2.1) and the standard deviation of κ is $\sqrt{\mathrm{Var}(\kappa)}$. Let $L^2(\Omega, \mathcal{F}, \mathbb{P})$ denote the space of square-integrable random fields defined on $(\Omega, \mathcal{F}, \mathbb{P})$.

We shall also need the concepts of Kronecker products and vec$(\cdot)$ operators.

Definition 2.2. *Let $X = [x_1, \ldots, x_m] \in \mathbb{R}^{n\times m}$ and $Y \in \mathbb{R}^{p\times q}$. Then*

$$X \otimes Y = \begin{pmatrix} x_{11}Y & \ldots & x_{1m}Y \\ \vdots & & \vdots \\ x_{1n}Y & \ldots & x_{nm}Y \end{pmatrix} \in \mathbb{R}^{np\times mq}, \quad \mathrm{vec}(X) = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \in \mathbb{R}^{nm}. \tag{2.2}$$

It follows from (2.2) that the vec$(\cdot)$ operator essentially reshapes a matrix into a column vector. In MATLAB notation, for example, we have `vec(X)=reshape(X,n*m,1)`. More precisely, we consider the vec$(\cdot)$ operator as a vector space isomorphism vec : $\mathbb{R}^{n\times m} \to \mathbb{R}^{nm}$ and denote its inverse by $\mathrm{vec}^{-1}:\mathbb{R}^{nm}\to\mathbb{R}^{n\times m}$. Kronecker product and $\mathrm{vec}(\cdot)$ operators exhibit the following properties (see, e.g., [7]):

$$\mathrm{vec}(AXB)=(B^T\otimes A)\mathrm{vec}(X), \tag{2.3}$$

$$(A\otimes B)(C\otimes D)=AC\otimes BD. \tag{2.4}$$

Finally, we introduce the tensor rank of a vectorized matrix; see, e.g., [15].

Definition 2.3. *Let $X\in\mathbb{R}^{n\times n}$ and $\mathbf{x}=\mathrm{vec}(X)\in\mathbb{R}^{n^2}$. Then the tensor rank of $\mathbf{x}$ is the smallest $k\in\mathbb{Z}_+$ such that*

$$\mathbf{x}=\sum_{i=1}^{k}u_i\otimes v_i,$$

where $u_i,v_i\in\mathbb{R}^n$. In particular, the tensor rank of the vector $\mathbf{x}$ coincides with the rank of the matrix X.

3. A model problem with stochastic inputs.

In this section, we introduce and discretize our model problem. More precisely, we consider the stochastic initial-boundary value problem: find a random function $u:\mathcal{D}\times\Omega\times[0,T]\to\mathbb{R}$ such that, $\mathbb{P}$-almost surely in Ω, the following parabolic equation holds:

$$\begin{cases}\dfrac{\partial u(\mathbf{x},\omega,t)}{\partial t}=\nabla\cdot(\kappa(\mathbf{x},\omega)\nabla u(\mathbf{x},\omega,t))+f(\mathbf{x}) & \text{in } \mathcal{D}\times\Omega\times(0,T],\\ u(\mathbf{x},\omega,t)=0, \quad \mathbf{x}\in\partial\mathcal{D},\ \omega\in\Omega,\ t\in[0,T], & \\ u(\mathbf{x},\omega,0)=0, \quad \mathbf{x}\in\mathcal{D},\ \omega\in\Omega, & \end{cases} \tag{3.1}$$

where, for ease of exposition, we limit our discussion to a sufficiently smooth, time-independent deterministic source term, as well as Dirichlet boundary conditions. However, our discussion naturally generalizes to other stochastic boundary conditions and stochastic time-dependent source terms. In the model (3.1), we note here that $\kappa(\mathbf{x},\omega)$, and hence the solution $u(\mathbf{x},\omega,t)$ are random fields. We assume that the random input κ is $\mathbb{P}$-almost surely uniformly positive; that is,

$$\exists\,\alpha,\beta \ \text{ such that } \ 0<\alpha\le\beta<+\infty,$$

with

$$\alpha\le\kappa(\mathbf{x},\omega)\le\beta \quad \text{a.e. in } \mathcal{D}\times\Omega. \tag{3.2}$$

To ensure regularity of the solution u with respect to the spatial variable x, we additionally assume that κ is globally Lipschitz in $\mathcal{D}\times\Omega$. The well-posedness of the model (3.1) then follows from the classical Lax–Milgram lemma (see, e.g., [20]).

Next, to solve (3.1), one essentially seeks a weak solution in a finite-dimensional subspace of a Hilbert space consisting of the tensor products of deterministic functions defined on the spatial domain and random functions defined on the probability space. More precisely, the idea is to first put the model in variational form before it is restricted to a finite-dimensional subspace; see, e.g., [21], [22] for more details. Assuming the model is in its variational form, we next proceed to review the discretization of (3.1) using the SGFEM.

3.1. Overview of stochastic Galerkin finite element method. As we noted in section 1, the discretization of PDEs with random coefficients using the classical SGFEM is standard in the literature. Indeed, one usually follows a three-step procedure in this method; see, e.g., [29], [23], [22]. The randomness in the model is, first of all, represented with a finite number of random variables via Karhunen–Loève expansion (KLE) (which indeed decouples the random and spatial dependencies in the random field, κ). Next, we approximate the solution as a finite-term expansion using basis orthogonal polynomials—the so-called generalized polynomial chaos expansion (PCE). The final stage entails performing a Galerkin projection on the set of polynomial basis functions. The above procedure transforms the stochastic problem (3.1) to a system of (usually) large coupled deterministic diffusion equations, which can then be solved with the appropriate methods for deterministic PDEs. In what follows, we briefly review the three steps.

3.2. Karhunen–Lòeve representation of stochastic inputs. Let $\kappa : \mathcal{D} \times \Omega \to \mathbb{R}$ be a random field with continuous covariance function $C_\kappa(\mathbf{x}, \mathbf{y})$. Then κ admits a proper orthogonal decomposition (or KLE)

$$\kappa(\mathbf{x}, \omega) = \bar{\kappa}(\mathbf{x}) + \sigma_\kappa \sum_{i=1}^{\infty} \sqrt{\lambda_i} \varphi_i(\mathbf{x}) \xi_i(\omega), \tag{3.3}$$

where σ_κ is the standard deviation of κ. The random variables $\xi := \{\xi_1, \xi_2, \ldots\}$ are centered, normalized, and uncorrelated[1] (but not necessarily independent) with

$$\xi_i(\omega) = \frac{1}{\sigma_\kappa \sqrt{\lambda_i}} \int_{\mathcal{D}} (\kappa(\mathbf{x}, \omega) - \bar{\kappa}(\mathbf{x})) \varphi_i(\mathbf{x}) \, d\mathbf{x},$$

and $\{\lambda_i, \varphi_i\}$ is the set of eigenvalues and eigenfunctions corresponding to $C_\kappa(\mathbf{x}, \mathbf{y})$. In other words, the eigenpairs $\{\lambda_i, \varphi_i\}$ solve the integral equations

$$\int_{\mathcal{D}} C_\kappa(\mathbf{x}, \mathbf{y}) \varphi_i(\mathbf{y}) \, d\mathbf{y} = \lambda_i \varphi_i(\mathbf{x}).$$

The eigenfunctions $\{\varphi_i\}$ form a complete orthogonal basis in $L^2(\mathcal{D})$. The eigenvalues $\{\lambda_i\}$ form a sequence of nonnegative real numbers decreasing to zero. In practice, the series (3.3) is truncated after, say, N terms based on the speed of decay of the eigenvalues since the series converges in $L^2(\mathcal{D} \times \Omega)$ due to

$$\sum_{i=1}^{\infty} \lambda_i = \int_\Omega \int_{\mathcal{D}} (\kappa(\mathbf{x}, \omega) - \bar{\kappa}(\mathbf{x}))^2 \, d\mathbf{x} d\mathbb{P}(\omega).$$

However, one has to ensure that the truncated random field

[1]We also make the simplifying assumption that they are independent and that the density function is a product of univariate terms.

$$\kappa_N(\mathbf{x},\omega) = \bar{\kappa}(\mathbf{x}) + \sigma_\kappa \sum_{i=1}^{N} \sqrt{\lambda_i}\varphi_i(\mathbf{x})\xi_i(\omega) \tag{3.4}$$

satisfies the positivity condition (3.2) so that the model (3.1) is still well-posed. In particular, we assume throughout this paper that $\bar{\kappa}(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathcal{D}$. It should be noted, though, that the truncated KLE (3.4) is a finite representation of $\kappa(\mathbf{x},\omega)$ with the minimal mean-square error over all such finite representations.

For some random inputs, the covariance functions and eigenpairs can be computed explicitly. If they are not known a priori, then they can be approximated numerically; see, e.g., [11] for details regarding the computation and convergence of KLE. We admit, however, that it is not always easy to find the KLE of random fields; we restrict our discussion in this paper essentially to problems for which the KLE is available.

3.3. Generalized polynomial chaos expansion. Generalized PCE is a means of representing a random field $u \in L^2(\Omega,\mathcal{F},\mathbb{P})$ parametrically through a set of random variables. More precisely, we have

$$u(\mathbf{x},\omega,t) = \sum_{j=0}^{\infty} u_j(\mathbf{x},t)\psi_j(\xi(\omega)), \tag{3.5}$$

where u_j, the deterministic modes of the expansion, are given by

$$u_j(\mathbf{x},t) = \frac{\langle u(\mathbf{x},\omega,t)\psi_j(\xi)\rangle}{\left\langle \psi_j^2(\xi)\right\rangle},$$

ξ is a finite-dimensional random vector as in (3.4), and ψ_j are multivariate orthogonal polynomials satisfying

$$\langle\psi_0(\xi)\rangle = 1, \quad \langle\psi_j(\xi)\rangle = 0,\ j > 0, \quad \langle\psi_j(\xi)\psi_k(\xi)\rangle = \left\langle\psi_j^2(\xi)\right\rangle \delta_{jk},$$

with

$$\langle\psi_j(\xi)\rangle = \int_{\omega\in\Omega} \psi_j(\xi(\omega))\, d\mathbb{P}(\omega) \tag{3.6}$$

$$= \int_{\xi\in\Pi} \psi_j(\xi)\rho(\xi)\, d\xi, \tag{3.7}$$

where Π and ρ are, respectively, the support and probability density of ξ. The random variables are chosen such that their probability density coincides with the weight function of the orthogonal polynomials used in the expansion, e.g., Hermite polynomials and Gaussian random variables, Legendre polynomials and uniform random variables, Jacobi polynomials and beta random variables, etc. Note that n-dimensional orthogonal polynomials are constructed by taking n products of one-dimensional orthogonal polynomials.

By the Cameron–Martin theorem, the series (3.5) converges in the Hilbert space $L^2(\Omega,\mathcal{F},\mathbb{P})$; see, e.g., [9]. Thus, as in the case of KLE, we truncate (3.5) after, say, P terms to obtain

$$u(\mathbf{x},\omega,t) = \sum_{j=0}^{P-1} u_j(\mathbf{x},t)\psi_j(\xi(\omega)), \tag{3.8}$$

where P is determined by the expression

$$P = 1 + \sum_{k=1}^{Q} \frac{1}{k!} \prod_{j=0}^{k-1} (N+j) = \frac{(N+Q)!}{N!Q!}. \tag{3.9}$$

In (3.9), Q is the highest degree of the orthogonal polynomial used to represent u. A detailed discussion on how to choose Q (and hence P) can be found in, for instance, [22].

Observe from (3.3) and (3.5) that the expansions decouple the random fields into stochastic and deterministic dependencies. Besides, the KLE in (3.3) is a special case of the PCE in (3.5) with $Q = 1$.

3.4. Stochastic Galerkin approach. If we substitute the expressions (3.4) and (3.8) into the model (3.1), we get

$$\begin{aligned}\sum_{i=0}^{P-1} \frac{\partial u_i(\mathbf{x},t)}{\partial t}\psi_i = \sum_{i=0}^{P-1} \nabla \cdot \left(\left(\bar\kappa(\mathbf{x}) + \sigma_\kappa \sum_{k=1}^{N} \sqrt{\lambda_k}\varphi_k(\mathbf{x})\xi_k \right) \nabla u_i(\mathbf{x},t)\psi_i \right) \\ + \ f(\mathbf{x}).\end{aligned} \tag{3.10}$$

Next, we project (3.10) onto the space spanned by the P polynomial chaos basis functions to obtain, for $j = 0, 1, \ldots, P-1$,

$$\left\langle \psi_j^2 \right\rangle \frac{\partial u_j(\mathbf{x},t)}{\partial t} = \sum_{i=0}^{P-1} \nabla \cdot (a_{ij}(\mathbf{x}) \nabla u_i(\mathbf{x},t)) + \langle \psi_j \rangle f(\mathbf{x}), \tag{3.11}$$

where

$$\begin{aligned} a_{ij}(\mathbf{x}) &= \bar\kappa(\mathbf{x}) \langle \psi_i \psi_j \rangle + \sigma_\kappa \sum_{k=1}^{N} \sqrt{\lambda_k}\varphi_k(\mathbf{x}) \langle \xi_k \psi_i \psi_j \rangle \\ &= \bar\kappa(\mathbf{x}) \left\langle \psi_j^2 \right\rangle \delta_{ij} + \sigma_\kappa \sum_{k=1}^{N} \sqrt{\lambda_k}\varphi_k(\mathbf{x}) \langle \xi_k \psi_i \psi_j \rangle . \end{aligned} \tag{3.12}$$

It should be noted that the system of P deterministic diffusion equations in (3.11) are coupled. Designing a fast solver for such a large coupled system can be quite a challenge. This is the main purpose of the remainder of this paper.

In practice, the quantity of interest is not the solution u of the model (3.1) itself; rather, one is usually interested in some functional of u. Once the modes u_i, $i = 0, 1, \ldots, P-1$, have been computed, the intended quantities of interest, such as the moments and probability density of the solution, can easily be deduced. For instance, the mean and the variance of the solution are, respectively, given explicitly by

$$\begin{aligned}
\langle u(\mathbf{x},\xi,t)\rangle &= \left\langle \sum_{j=0}^{P-1} u_j(\mathbf{x},t)\psi_j(\xi) \right\rangle \\
&= \sum_{j=0}^{P-1} u_j(\mathbf{x},t)\,\langle \psi_j(\xi)\rangle \\
&= \sum_{j=0}^{P-1} u_j(\mathbf{x},t)\delta_{0,j} = u_0(\mathbf{x},t) \qquad (3.13)
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{Var}(u(\mathbf{x},\xi,t)) &= \langle u(\mathbf{x},\xi,t)^2\rangle - \langle u(\mathbf{x},\xi,t)\rangle^2 \\
&= \left\langle \sum_{i=0}^{P-1}\sum_{j=0}^{P-1} u_i(\mathbf{x},t)u_j(\mathbf{x},t)\psi_i(\xi)\psi_j(\xi) \right\rangle - u_0^2(\mathbf{x},t) \\
&= \sum_{i=0}^{P-1}\sum_{j=0}^{P-1} u_i(\mathbf{x},t)u_j(\mathbf{x},t)\,\langle \psi_i(\xi)\psi_j(\xi)\rangle - u_0^2(\mathbf{x},t) \\
&= \sum_{i=0}^{P-1}\sum_{j=0}^{P-1} u_i(\mathbf{x},t)u_j(\mathbf{x},t)\,\langle \psi_i^2(\xi)\rangle\,\delta_{i,j} - u_0^2(\mathbf{x},t) \\
&= \sum_{i=0}^{P-1} u_i^2(\mathbf{x},t)\,\langle \psi_i^2(\xi)\rangle - u_0^2(\mathbf{x},t) \\
&= \sum_{i=1}^{P-1} u_i^2(\mathbf{x},t)\,\langle \psi_i^2(\xi)\rangle . \qquad (3.14)
\end{aligned}$$

3.5. Spatial and time discretizations. In the spirit of [22], [23], we use classical finite elements to discretize the spatial domain. Furthermore, we assume that each of the deterministic coefficients u_i, $i = 0, 1, \dots, P-1$, in (3.11) is discretized on the same mesh and with an equal number of elements. More precisely, with J basis functions $s_j(\mathbf{x})$, each mode u_i is approximated as a linear combination of the form

$$u_i(\mathbf{x},t) \approx \sum_{j=1}^{J} u_{ij}(t)s_j(\mathbf{x}), \quad i = 0, \dots, P-1.$$

After spatial discretization and some algebraic manipulations (see, e.g., [22]), one gets the following system of ordinary differential equations:

$$(G_0 \otimes M)\,\frac{d\mathbf{u}(t)}{dt} + \left(\sum_{i=0}^{N} G_i \otimes K_i\right)\mathbf{u}(t) = \mathbf{g}_0 \otimes \mathbf{f}_0, \qquad (3.15)$$

where

$$\mathbf{u}(t) = \begin{pmatrix} u_0(t) \\ \vdots \\ u_{P-1}(t) \end{pmatrix}, \quad \text{with } \; u_i(t) \in \mathbb{R}^J, \; i = 0, 1, \ldots, P-1. \tag{3.16}$$

The stochastic matrices $G_i \in \mathbb{R}^{P\times P}$ are given by

$$G_0(j,k) = \langle \psi_j(\xi)\psi_k(\xi)\rangle, \quad G_i(j,k) = \langle \xi_i \psi_j(\xi)\psi_k(\xi)\rangle, \quad i = 1, \ldots, N, \tag{3.17}$$

and the vectors $\mathbf{g}_0$ and $\mathbf{f}_0$ are defined via

$$\mathbf{g}_0(j) = \langle \psi_j(\xi)\rangle, \quad \mathbf{f}_0(j) = \int_{\mathcal{D}} f(\mathbf{x}) s_j(\mathbf{x})\, d\mathbf{x}. \tag{3.18}$$

Now, suppose we denote (normalized) univariate orthogonal polynomials by $\{\phi_k\}$. Then, recalling that $\{\phi_k\}$ satisfy the three-term recurrence relation

$$\phi_{k+1}(x) = (x - \alpha_k)\phi_k(x) - \beta_k \phi_{k-1}(x), \quad x \in \mathbb{R},$$

with $\phi_0 = 1, \phi_{-1} = 0$, it turns out that

$$G_0(j,k) = \langle \psi_j, \psi_k\rangle = \prod_{i=1}^{N} \langle \phi_{j_i}, \phi_{k_i}\rangle = \prod_{i=1}^{N} \delta_{j_i k_i} = \delta_{jk}, \tag{3.19}$$

and for $k > 0$, we have

$$\begin{aligned} G_i(j,k) &= \langle \xi_i \psi_j, \psi_k\rangle \\ &= \langle \xi_i \phi_j, \phi_k\rangle \prod_{l=1, l\neq i}^{N} \langle \phi_{j_l}, \phi_{k_l}\rangle \\ &= \left(\langle \phi_{j_i+1}, \phi_{k_i}\rangle + \alpha_{j_i}\langle \phi_{j_i}, \phi_{k_i}\rangle + \beta_{j_i}\langle \phi_{j_i-1}, \phi_{k_i}\rangle\right) \prod_{l=1, l\neq i}^{N} \langle \phi_{j_l}, \phi_{k_l}\rangle. \end{aligned} \tag{3.20}$$

Hence, G_0 is a diagonal matrix, whereas for $k > 0$, the matrix G_k has at most three nonzero elements per row. Moreover, for symmetric density functions ρ, the coefficients α_j in the recurrence relation vanish so that the matrices G_k have a most two nonzeros per row; see, e.g., [22]. The mass matrix $M \in \mathbb{R}^{J\times J}$ and the stiffness matrices $K_i \in \mathbb{R}^{J\times J}$, $i = 0, 1, \ldots, N$, are given, respectively, by

$$M(j,k) = \int_{\mathcal{D}} s_j(\mathbf{x}) s_k(\mathbf{x})\, d\mathbf{x}, \tag{3.21}$$

$$K_0(j,k) = \int_{\mathcal{D}} \bar{\kappa}(\mathbf{x}) \nabla s_j(\mathbf{x}) \nabla s_k(\mathbf{x})\, d\mathbf{x}, \tag{3.22}$$

$$K_i(j,k) = \sigma_\kappa \sqrt{\lambda_i} \int_{\mathcal{D}} \varphi_i(\mathbf{x}) \nabla s_j(\mathbf{x}) \nabla s_k(\mathbf{x})\, d\mathbf{x}. \tag{3.23}$$

Observe, in particular, from (3.22) and (3.23) that the matrix K_0 contains the mean information of the random field κ, whereas the matrices K_i, $i > 0$, capture the fluctuations therein.

For time discretization, we use implicit Euler to avoid stability issues. To this end, we set $t_n = n\tau$, $n = 0, 1, \ldots, n_{max}$, with $\tau = T/n_{max}$. Moreover, we define the computed numerical approximation $\mathbf{u}(t_n) := \mathbf{u}^n$ so that (3.15) yields

$$G_0 \otimes M \left(\frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\tau} \right) + \left(\sum_{i=0}^{N} G_i \otimes K_i \right) \mathbf{u}^n = (\mathbf{g}_0 \otimes \mathbf{f}_0)^n , \tag{3.24}$$

or, equivalently,

$$\mathcal{A}\mathbf{u}^n = \mathbf{b}^n, \tag{3.25}$$

where

$$\mathbf{b}^n = (G_0 \otimes M)\, \mathbf{u}^{n-1} + \tau\, (\mathbf{g}_0 \otimes \mathbf{f}_0)^n , \tag{3.26}$$

and

$$\begin{aligned} \mathcal{A} &= G_0 \otimes M + \tau \sum_{i=0}^{N} G_i \otimes K_i \\ &= G_0 \otimes (M + \tau K_0) + \tau \sum_{i=1}^{N} G_i \otimes K_i \\ &= G_0 \otimes \tilde{K}_0 + \sum_{i=1}^{N} G_i \otimes \tilde{K}_i, \end{aligned} \tag{3.27}$$

with $\tilde{K}_0 := M + \tau K_0$, $\tilde{K}_i = \tau K_i$, $i = 1, \ldots, N$.

We note that the global stochastic Galerkin matrix $\mathcal{A}$ as defined in (3.27) is sparse in the block sense, symmetric, and positive definite. Indeed, in practical applications such as flow problems, the length N of the random vector ξ is usually large due to the presence of small correlation length in the covariance function of κ. This, in turn, increases the value of P in (3.9) (and hence the dimension of $\mathcal{A}$) quite fast; see, e.g., [11]. This is a major drawback of the SGFEM. In order to break the curse of dimensionality for this problem, we consider a low-rank approximation to the solution of the linear system (3.25). We want to emphasize that it is often impossible to compute the full solution to an SGFEM discretized problem, as the matrix dimensions quickly become prohibitively large with respect to the discretization parameters. The low-rank technique presented here needs to store only a small portion of the vectors in comparison to the full problem, and we want to theoretically justify this approach in the next section.

4. Existence of low-rank solution of the stochastic Galerkin system.

In what follows, we focus our attention on the solution of the system (3.25) using low-rank iterative solvers. First, however, following [4], we show, under certain conditions, that the solution of (3.27) can be approximated with a vector of low tensor rank. To this end, for arbitrary $\mathcal{A} \in \mathbb{R}^{m\times m}$ and $\mathbf{b} \in \mathbb{R}^m$, consider the following linear system:

$$\mathcal{A}\mathbf{x} = \mathbf{b}. \tag{4.1}$$

Define, for $k \in \mathbb{N}$, the following quadrature points and weights:

$$h_{st} = \pi^2/\sqrt{k}, \tag{4.2}$$

$$t_j = \log\left(\exp(jh_{st}) + \sqrt{1+\exp(2jh_{st})}\right), \tag{4.3}$$

$$w_j = h_{st}/\sqrt{1+\exp(-2jh_{st})}. \tag{4.4}$$

Our point of departure is the following lemma from [15].

Lemma 4.1. *Let the matrix $\mathcal{A} \in \mathbb{R}^{m\times m}$ in (4.1) be symmetric and positive definite. Suppose that the spectrum of $\mathcal{A}$ is contained in the strip $\Lambda := [\lambda_{min}, \lambda_{max}] \subset \mathbb{R}_+$, and let Γ be the interval $[1, 2\lambda_{min}/\lambda_{max}+1]$. Let $k \in \mathbb{N}$ and $j = -k, \ldots, k$. Then the solution $\mathbf{x} = \mathcal{A}^{-1}\mathbf{b}$ to the system (4.1) can be approximated by*

$$\tilde{\mathbf{x}} := -\sum_{j=-k}^{k} \frac{2w_j}{\lambda_{min}} \exp\left(-\frac{2t_j}{\lambda_{min}}\mathcal{A}\right)\mathbf{b}, \tag{4.5}$$

with the approximation error

$$||\mathbf{x} - \tilde{\mathbf{x}}||_2 \le \frac{C_{st}}{\pi\lambda_{min}} \exp\left(\frac{1}{\pi} - \pi\sqrt{k}\right)|\Gamma|||\mathbf{b}||_2, \tag{4.6}$$

where $|\Gamma|$ is the length of Γ and the quadrature weights t_j, w_j are given by (4.3) and (4.4).

A sharper bound can, in fact, be obtained in (4.6) if $\mathcal{A}$ possesses some special Kronecker product structure; see, e.g., [17]. Next, we recall the so-called Sherman–Morrison–Woodbury formula (see, e.g., [14]), on which, together with Lemma 4.1, we shall rely to prove our main result.

Lemma 4.2. *Let $X \in \mathbb{R}^{n\times n}$ be nonsingular, and let $Y, Z \in \mathbb{R}^{n\times m}$, with $m \le n$. Then $X + YZ^T$ is invertible if and only if $I_m + Z^TX^{-1}Y$ is invertible, with*

$$(X+YZ^T)^{-1} = X^{-1} - X^{-1}Y(I_m + Z^TX^{-1}Y)^{-1}Z^TX^{-1}. \tag{4.7}$$

We can now state our main result, which shows that the solution of the system (3.25) can indeed be approximated with a vector of low tensor rank. For this purpose, we split the matrix (3.27) as follows:

$$\mathcal{A} = \underbrace{G_0 \otimes \tilde{K}_0}_{=\mathcal{L}} + \sum_{i=1}^{N} G_i \otimes \tilde{K}_i. \tag{4.8}$$

Observe then from (3.17), (3.21), (3.22), and (3.27) that $\mathcal{L}$ in (4.8) is symmetric and positive definite. Furthermore, let the stochastic matrices $G_i,\ i=1,\dots,N$, be decomposed in low-rank format:

$$G_i := U_i V_i^T,\ \ U_i, V_i \in \mathbb{R}^{P\times r_i},\ i=1,\dots,N. \tag{4.9}$$

We illustrate the low-rank nature of these matrices in section 6. Since also the stiffness matrices $\tilde{K}_i$, $i=1,\dots,N$, are symmetric, then each of them admits the factorization

$$\tilde{K}_i := L_i D_i L_i^T = \tilde{L}_i L_i^T,\ \ \tilde{L}_i, L_i \in \mathbb{R}^{J\times J},\ i=1,\dots,N, \tag{4.10}$$

where $\tilde{L}_i := L_i D_i$, $i=1,\dots,N$, with D_i and L_i (and hence $\tilde{L}_i$) being, respectively, diagonal and lower triangular matrices. The following result holds.

Theorem 4.3. *Let $\mathcal{A}$ denote a matrix of Kronecker product structure as in (3.27). Assume that the spectrum of $\mathcal{L}$ in (4.8) is contained in the strip $\Lambda := [\lambda_{min}, \lambda_{max}] \subset \mathbb{R}_+$, and let Γ be the interval $[1, 2\lambda_{max}/\lambda_{min}+1]$. Let G_i, $i=1,\dots,N$, have the low-rank representation (4.9) with $r=\sum_{j=1}^N r_j$, and let $\tilde{K}_i$, $i=1,\dots,N$, be given by the decomposition (4.10). Suppose further that $U=[U_1\otimes\tilde{L}_1,\dots,U_N\otimes\tilde{L}_N]$ and $V=[V_1\otimes L_1,\dots,V_N\otimes L_N]$. For all time steps $n\geq 2$, let the tensor rank of $\mathbf{b}^n \leq \ell$, where $\ell \ll JP$. Then, for $k\in\mathbb{N}$, the solution $\mathbf{u}^n$ of (3.25) can be approximated by a vector $\tilde{\mathbf{u}}^n$ of the form*

$$\tilde{\mathbf{u}}^n = -\sum_{j=-k}^{k} \frac{2w_j}{\lambda_{min}} \left(\exp(G_0)\otimes \exp\left(-\frac{2t_j}{\lambda_{min}}\tilde{K}_0\right)\right)[\mathbf{b}^n - U\mathcal{Y}], \tag{4.11}$$

where the vector $\mathcal{Y}\in\mathbb{R}^{J\cdot r}$ is the solution of

$$(I_{J\cdot r} + V^T\mathcal{L}^{-1}U)\mathcal{Y} = V^T\mathcal{L}^{-1}\mathbf{b}^n, \tag{4.12}$$

and t_j, w_j are the quadrature weights and points as given by (4.3) and (4.4). The corresponding approximation error is given by

$$||\mathbf{u}^n - \tilde{\mathbf{u}}^n||_2 \leq \frac{C_{st}}{\pi\lambda_{min}} \exp\left(\frac{1}{\pi} - \pi\sqrt{k}\right) |\Gamma| ||\mathbf{b}^n - U\mathcal{Y}||_2. \tag{4.13}$$

Moreover, the tensor rank of $\tilde{\mathbf{u}}^n$ in (4.11) is at most

(i) $(2k+1)\cdot(r+1)$ *if* $n=1$ *and*

(ii) $(2k+1)\cdot(r+\ell)$ *if* $n\geq 2$.

Proof. Observe first from (2.4), (4.9), and (4.10) that we have the low-rank representation

$$\sum_{i=1}^N G_i\otimes\tilde{K}_i = \sum_{i=1}^N (U_iV_i^T)\otimes(\tilde{L}_iL_i^T) = \sum_{i=1}^N (U_i\otimes\tilde{L}_i)(V_i^T\otimes L_i^T) = UV^T. \tag{4.14}$$

Hence, from Lemma 4.2, (4.8), and (4.14), we note that

$$\mathcal{A}^{-1} = (\mathcal{L}+UV^T)^{-1} = \mathcal{L}^{-1} - \mathcal{L}^{-1}U(I_{J\cdot r}+V^T\mathcal{L}^{-1}U)^{-1}V^T\mathcal{L}^{-1},$$

so that

$$\mathbf{u}^n = \mathcal{A}^{-1}\mathbf{b}^n \Leftrightarrow \mathbf{u}^n = \mathcal{L}^{-1}\left[\mathbf{b}^n - U\underbrace{(I_{J\cdot r} + V^T\mathcal{L}^{-1}U)^{-1}V^T\mathcal{L}^{-1}\mathbf{b}^n}_{=\mathcal{Y}}\right]. \tag{4.15}$$

Now, by definition, the matrix $\mathcal{L} = G_0 \otimes \tilde{K}_0$ is symmetric and positive definite. Thus, using the fact that

$$\begin{aligned}\exp(-\beta\mathcal{L}) &= \exp(-\beta(G_0 \otimes \tilde{K}_0))\\ &= \exp(G_0 \otimes (-\beta\tilde{K}_0))\\ &= \exp(G_0) \otimes \exp(-\beta\tilde{K}_0),\end{aligned}$$

where $\beta := 2t_j/\lambda_{min}$, together with (4.15) and Lemma 4.1, immediately yields (4.11) and (4.13).

To show (i), it suffices to show that the tensor rank of $\mathbf{b}^1 - U\mathcal{Y}$ is at most $r+1$. Now, note that

$$\operatorname{rank}(\operatorname{vec}^{-1}(\mathbf{b}^1 - U\mathcal{Y})) \leq \operatorname{rank}(\operatorname{vec}^{-1}(\mathbf{b}^1)) + \operatorname{rank}(\operatorname{vec}^{-1}(-U\mathcal{Y})). \tag{4.16}$$

From (3.26), we see that $\mathbf{b}^1 = \tau\left(\mathbf{g}_0 \otimes \mathbf{f}_0\right)$, since $\tilde{\mathbf{u}}^0 = 0$ and the source term f is time-independent. But then, since the orthogonal polynomials $\{\psi_j\}$ satisfy

$$\mathbf{g}_0(j) = \langle\psi_j\rangle = \begin{cases}1, & j = 0,\\ 0 & \text{otherwise},\end{cases}$$

it follows from (3.18) that $\operatorname{vec}^{-1}(\mathbf{g}_0 \otimes \mathbf{f}_0) \in \mathbb{R}^{J\times P}$ is a matrix of rank 1. Hence, $\mathbf{b}^1$ is a vector of tensor rank 1. Next, following arguments similar to those in the proof of Theorem 1 in [4], we show that the tensor rank of $U\mathcal{Y}$ is r, which, together with (4.16), completes the proof of (i). Now, let $\mathcal{Y}_{r_i}$ denote $J \cdot r_i$ elements of $\mathcal{Y}$, and observe from (2.3) that

$$\begin{aligned}U\mathcal{Y} &= [U_1 \otimes \tilde{L}_1, \ldots, U_N \otimes \tilde{L}_N]\mathcal{Y}\\ &= [U_1 \otimes \tilde{L}_1, \ldots, U_N \otimes \tilde{L}_N]\operatorname{vec}\left(\operatorname{vec}^{-1}(\mathcal{Y})\right)\\ &= \sum_{i=1}^{N} \operatorname{vec}\left(\tilde{L}_i \operatorname{vec}^{-1}(\mathcal{Y}_{r_i})U_i^T\right)\\ &= \sum_{i=1}^{N}\sum_{j=1}^{r_i} \operatorname{vec}\left(\tilde{L}_{i,j}Y_{i,j}^T\right),\end{aligned} \tag{4.17}$$

where $Y_i^T := \operatorname{vec}^{-1}(\mathcal{Y}_{r_i})U_i^T$. Applying (2.3) again to (4.17), we obtain

$$U\mathcal{Y} = \sum_{i=1}^{N}\sum_{j=1}^{r_i}(Y_{i,j} \otimes \tilde{L}_{i,j})\operatorname{vec}(1) = \sum_{i=1}^{N}\sum_{j=1}^{r_i}(Y_{i,j} \otimes \tilde{L}_{i,j}). \tag{4.18}$$

But then, by assumption the r_i sum up to r. Hence, the tensor rank of $U\mathcal{Y}$ is r.

Finally, to prove the assertion (ii), suppose that, for $n \geq 2$, the tensor rank of $\mathbf{b}^n$ is at most $\ell \ll JP$. Since the tensor rank of $U\mathcal{Y}$ is r, it trivially follows from the previous argument and the definition of $\mathbf{b}^n$ in (3.26) that (ii) holds with $\ell \geq 1$. ■

Remark 1. Note that G_0 is just a $P \times P$ identity matrix if we work with orthonormal basis polynomials $\{\psi_i\}$. Hence, in this special case, (4.11) reduces to

$$\tilde{\mathbf{u}}^n = -\sum_{j=-k}^{k} \frac{2w_j}{\lambda_{min}} \left(I_P \otimes \exp\left(-\frac{2t_j}{\lambda_{min}} \tilde{K}_0 \right) \right) [\mathbf{b}^n - U\mathcal{Y}].$$

Remark 2. The assumption in Theorem 4.3 that, for all $n \geq 2$, the tensor rank of the right-hand side $\mathbf{b}^n$ is at most ℓ, where $1 \leq \ell \ll JP$, is justified by the fact that the tensor rank tends to grow as the time step n increases. In practical computations, the tensor rank of $\mathbf{u}^{n-1}$ is truncated with respect to its singular value decay to ensure that the tensor rank of $\mathbf{b}^n$ is kept under control. The singular value decay of the right-hand sides and final solution (reshaped as $J \times P$ matrices) are numerically illustrated in section 6.

Remark 3. We note here that Theorem 4.3 provides theoretical evidence for the existence of low-rank approximation to the solution of (3.25) as $JP \to \infty$.

5. Computing low-rank approximations.

Although the stochastic Galerkin matrix $\mathcal{A}$ in (3.27) is block sparse, symmetric, and positive definite, it is generally ill-conditioned with respect to stochastic and spatial discretization parameters,[2] e.g., the finite element mesh size, the length N of the random vector ξ, or the total degree, Q, of the multivariate stochastic basis polynomials $\{\psi_i\}$ [22]. Hence, a natural iterative solver for the system is a preconditioned conjugate gradient (CG) method [22, 27]. Nevertheless, the choice of an "appropriate" preconditioner is of utmost concern in this regard. In dealing with steady problems with relatively small σ_κ, many authors use the so-called mean-based preconditioner proposed originally in [12]. Ullmann in [27] points out that the mean-based preconditioner does not take into account all the information contained in $\mathcal{A}$ and thus proposes and analyzes an optimal preconditioner based on an approach discussed in [28]. In what follows, we call this the Ullmann preconditioner.

The relative efficiency and optimality of the two preconditioners above notwithstanding, a major issue in solving (3.25) is evident. More precisely, for each time step n, one has to solve an enormous elliptic system. Due to the coupled nature of the systems, this exercise can be both computer memory- and time-consuming. To mitigate this problem, we propose solving (3.25) with the two preconditioners mentioned in the previous paragraph, together with the low-rank CG method proposed in [18] in the framework of parameterized steady problems. First, however, we introduce the preconditioners.

5.1. Preconditioning.

5.1.1. Mean-based preconditioner.

The mean-based preconditioner is given by

[2]There are, however, no conditioning issues with respect to stochastic parameters from uniform distributions and Legendre polynomial chaos.

$$\mathcal{M}_0 := G_0 \otimes \tilde{K}_0. \tag{5.1}$$

Now, observe that this is just the matrix $\mathcal{L}$ in Theorem 4.3 and that G_0 is a diagonal matrix due to the orthogonality of the stochastic basis functions $\{\psi_i\}$. Hence, $\mathcal{M}_0$ is a block diagonal matrix. Moreover, by definition, $\tilde{K}_0 = M + \tau K_0$, so that $\tilde{K}_0$ is symmetric and positive definite since M and K_0 are both symmetric and positive definite from (3.21) and (3.22). So, $\mathcal{M}_0$ is positive definite and $\mathcal{M}_0^{-1} = G_0^{-1} \otimes \tilde{K}_0^{-1}$, where $G_0^{-1}(j,j) = 1/G_0(j,j) > 0$. The preconditioner then entails the approximate action of P uncoupled copies of $\tilde{K}_0^{-1}$.

5.1.2. Ullmann preconditioner. This preconditioner is of the form

$$\mathcal{M}_1 := \underbrace{\sum_{i=0}^{N} \frac{\operatorname{trace}(\tilde{K}_i^T \tilde{K}_0)}{\operatorname{trace}(\tilde{K}_0^T \tilde{K}_0)} G_i}_{:=G} \otimes \tilde{K}_0. \tag{5.2}$$

The Ullmann preconditioner (5.2) can be thought of as a "perturbed" version of $\mathcal{M}_0$ since

$$\mathcal{M}_1 = \underbrace{G_0 \otimes \tilde{K}_0}_{:=\mathcal{M}_0} + \sum_{i=1}^{N} \frac{\operatorname{trace}(\tilde{K}_i^T \tilde{K}_0)}{\operatorname{trace}(\tilde{K}_0^T \tilde{K}_0)} G_i \otimes \tilde{K}_0. \tag{5.3}$$

It is inspired by the first part of the following result obtained by Van Loan and Pitsianis.

Lemma 5.1 (see [28]). *Suppose $m = m_1 m_2$, $n = n_1 n_2$, and $X \in \mathbb{R}^{m\times n}$. If $R \in \mathbb{R}^{m_2 \times n_2}$ is fixed, then the matrix $L \in \mathbb{R}^{m_1\times n_1}$ defined by*

$$L_{i,j} := \frac{\operatorname{trace}(X_{i,j}^T R)}{\operatorname{trace}(R^T R)}, \quad i = 1,\ldots,m_1, \; j = 1,\ldots,n_1, \tag{5.4}$$

minimizes $||X - L \otimes R||_F$ where $X_{i,j}^T = X((i-1)m_2 + 1 : im_2, (j-1)n_2 + 1 : jn_2)$. Likewise, if $L \in \mathbb{R}^{m_1\times n_1}$ is fixed, then the matrix $R \in \mathbb{R}^{m_2\times n_2}$ defined by

$$R_{i,j} := \frac{\operatorname{trace}(\tilde{X}_{i,j}^T L)}{\operatorname{trace}(L^T L)}, \quad i = 1,\ldots,m_2, \; j = 1,\ldots,n_2, \tag{5.5}$$

minimizes $||X - L \otimes R||_F$ where $\tilde{X}_{i,j}^T = X(i : m_2 : m, j : n_2 : n)$.

Van Loan and Pitsianis further show that the matrices L defined in (5.4) and R defined in (5.5) are symmetric and positive definite, provided X and R or L, respectively, are symmetric and positive definite.

Now if we set $X = \mathcal{A}$ and $R = \tilde{K}_0$ in (3.27), it follows from (5.4) that the matrix G in (5.2) minimizes $||\mathcal{A} - G \otimes \tilde{K}_0||_F$. More interestingly, $\mathcal{M}_1$ inherits the sparsity pattern, symmetry, and positive definiteness of the Galerkin matrix $\mathcal{A}$. Besides, unlike $\mathcal{M}_0$, it makes use of all the information in $\mathcal{A}$. Unfortunately, by reason of its construction, $\mathcal{M}_1$ loses the block diagonal structure enjoyed by $\mathcal{M}_0$, which makes it more expensive to invert than the latter.

5.2. A preconditioned iterative solver. Having presented the preconditioners, we proceed in this section to discuss the low-rank preconditioned conjugate gradient (LRPCG) method [18]. The basic idea behind the LRPCG is that the iterates in the algorithm are truncated based on the decay of their singular values. Thus, at each iteration, the iterates are put in low-rank format (cf. (4.9)). The truncation, no doubt, introduces further error in the solution. However, the truncation tolerance can be so tightened that the error becomes negligible. More importantly, the computer memory required to store the matrices is reduced, thereby enabling large-scale computations.

First, we present LRPCG in Algorithm 1.

Algorithm 1 Low-rank preconditioned conjugate gradient method.

```
**Input:** Matrix functions $\mathcal{A}, \mathcal{M} : \mathbb{R}^{J\times P} \to \mathbb{R}^{J\times P}$, right-hand side $B^n \in \mathbb{R}^{J\times P}$ in low-rank format. Truncation operator $\mathcal{T}$ w.r.t. relative accuracy $\varepsilon_{rel}$.
**Output:** Matrix $\mathbf{u}^n \in \mathbb{R}^{J\times P}$ fulfilling $||\mathcal{A}(\mathbf{u}^n) - B^n||_F \leq \text{tol}$.
$\mathbf{u}_0^n = 0,\ R_0 = B^n,\ Z_0 = \mathcal{M}^{-1}(R_0),\ P_0 = Z_0,\ Q_0 = \mathcal{A}(P_0),$
$\vartheta_0 = \langle P_0, Q_0\rangle,\ k = 0.$
while $||R_k||_F > \text{tol}$ do
    $\omega_k = \langle R_k, P_k\rangle / \vartheta_k$
    $\mathbf{u}_{k+1}^n = \mathbf{u}_k^n + \omega_k P_k,$            $\mathbf{u}_{k+1}^n \leftarrow \mathcal{T}(\mathbf{u}_{k+1}^n)$
    $R_{k+1} = B^n - \mathcal{A}(\mathbf{u}_{k+1}^n),$        $Optionally : R_{k+1} \leftarrow \mathcal{T}(R_{k+1})$
    $Z_{k+1} = \mathcal{M}^{-1}(R_{k+1})$
    $\beta_{k+1} = -\langle Z_{k+1}, Q_k\rangle / \vartheta_k$
    $P_{k+1} = Z_{k+1} + \beta_k P_k,$              $P_{k+1} \leftarrow \mathcal{T}(P_{k+1})$
    $Q_{k+1} = \mathcal{A}(P_{k+1}),$              $Optionally : Q_{k+1} \leftarrow \mathcal{T}(Q_{k+1})$
    $\vartheta_{k+1} = \langle P_k, Q_k\rangle$
    $k = k + 1$
end while
$\mathbf{u}^n = \mathbf{u}_k^n$
```

We point out a few things regarding the implementation of LRPCG with respect to the solution of (3.25). Note that, in Algorithm 1, all vectors in $\mathbb{R}^{J\cdot P}$ (cf. (3.16)) are reshaped into $\mathbb{R}^{J\times P}$ matrices by the vec^{-1} operator. Now, recall that for each fixed time step $n = 1, 2, \ldots, n_{max}$, we need to solve an elliptic system using the LRPCG algorithm. In particular, for each solve, we need to evaluate $\mathcal{A}(X)$, where $X := \mathbf{u}_k^n$ or P_k. For this purpose, we set

$$\mathcal{A}\text{vec}(X) = \left(\sum_{i=0}^{N} G_i \otimes \tilde{K}_i\right) \text{vec}(X), \tag{5.6}$$

where $X \in \mathbb{R}^{J\times P}$ is of low rank, say, k:

$$X = UV^T,\ \ U \in \mathbb{R}^{J\times k},\ \ V \in \mathbb{R}^{P\times k},\ \ k \ll J, P,$$
$$U = [u_1, \ldots, u_k],\ \ V \in [v_1, \ldots, v_k],$$

so that, using (2.3), one gets

$$\operatorname{vec}(X) = \operatorname{vec}\left(\sum_{i=1}^{k} u_j v_j^T\right) = \sum_{j=1}^{k} \operatorname{vec}(u_j v_j^T) = \sum_{j=1}^{k} v_j \otimes u_j. \tag{5.7}$$

Hence, we have

$$\begin{aligned}
\mathcal{A}\operatorname{vec}(X) &= \left(\sum_{i=0}^{N} G_i \otimes \tilde{K}_i\right) \operatorname{vec}(X) \\
&= \left(\sum_{i=0}^{N} G_i \otimes \tilde{K}_i\right)\left(\sum_{j=1}^{k} v_j \otimes u_j\right) \\
&= \sum_{i=0}^{N}\sum_{j=1}^{k} (G_i v_j) \otimes (\tilde{K}_i u_j) \in \mathbb{R}^{J\cdot P\times 1}, \qquad (5.8)
\end{aligned}$$

and we then have to reshape (5.8) to have

$$\mathcal{A}(X) := \operatorname{vec}^{-1}(\mathcal{A}\operatorname{vec}(X)) \in \mathbb{R}^{J\times P}. \tag{5.9}$$

Moreover, in order to apply either of the two preconditioners to the residual matrices R_k, that is, $\mathcal{M}^{-1}(R_k)$, we have to ensure that R_k are in low-rank format as in (5.7), so we can obtain expressions similar to those in (5.8) and (5.9), since $\mathcal{M}^{-1} := \mathcal{M}_i^{-1}$, $i = 0, 1$, have the same size and Kronecker product structure as $\mathcal{A}$. The right-hand side of (3.25), that is, $\mathbf{b}^n = (G_0 \otimes M)\,\mathbf{u}^{n-1} + \tau\,(\mathbf{g}_0 \otimes \mathbf{f}_0)$, is also reshaped such that $B^n := \operatorname{vec}^{-1}(\mathbf{b}^n) \in \mathbb{R}^{J\times P}$. Finally, the iterates $\mathbf{u}_k^n$ are truncated in every iteration by the truncation operator $\mathcal{T}$ based on the decay of their singular values. In what follows, we describe how the truncation operation works, as well as how to exploit the low-rank format of the matrices to compute the inner products in Algorithm 1.

5.3. Truncation operator and matrix inner products.

We start this section by assuming that the matrix of interest X is represented by two low-rank factors U and V, i.e., $X = UV^T$. Our iterative procedure starts with a low-rank decomposition of the right-hand side, but the ranks of the low-rank factors increase either via the low-rank matrix vector products or vector recurrences. For this purpose, it is necessary to find new low-rank approximations $\tilde{U}$ and $\tilde{V}$ that approximate the old product $UV^T \approx \tilde{U}\tilde{V}^T$ using a small truncation tolerance.

Kressner and Tobler discuss in [17] that one can obtain the new low-rank representation by performing skinny QR factorizations of both matrices, i.e., $U = Q_u R_u$ and $V = Q_v R_v$. We then note that $X = Q_u R_u R_v^T Q_v^T$ and a singular value decomposition (SVD) [14] of $R_u R_v^T = B\Sigma C^T$ allows us to compute a representation of lower rank. Depending on the truncation tolerance, we can drop small singular values in Σ. The new low-rank factors are then obtained via

$$\tilde{V} = Q_v B(:, 1:k) \quad \text{and} \quad \tilde{U} = Q_v C(:, 1:k)\Sigma(1:k, 1:k).$$

Here, the truncation rank $k' \le k$ is chosen such that the singular values s_k satisfy

$$\sqrt{s_{k'+1}^2 + \cdots + s_k^2} \le \mathit{trunctol}\sqrt{s_1^2 + \cdots + s_k^2},$$

where *trunctol* is the truncation tolerance. This leads to $X \approx \tilde{U}\tilde{V}^T$, where we have used MATLAB notation. An alternative approach that we used, which due to some internal handling within MATLAB typically produces fast results, exploits the MATLAB function `svds` to directly compute a truncated SVD of $UV^T \approx B\Sigma C^T$ (see also [25]). Again, we drop small singular values in Σ to obtain $\tilde{V}$ and $\tilde{U}$. The computation of the truncated SVD is typically done via a procedure based on a Krylov subspace method where we require multiplication with the matrix UV^T. It is easy to see that we can perform this multiplication using the matrix factored form. This approach proved advantageous in terms of the time needed for the truncation. Alternative ways to compute the truncated SVD are possible and can be found in [16, 3, 24]. The cost of computing the truncation depends, for example in the truncated SVD approach, on the cost of multiplying with the matrix UV^T. Assuming that $U \in \mathbb{R}^{J\times k}$, then every iteration of an iterative procedure to compute the truncated SVD needs $O(Jk)$ flops to compute the multiplication with U and analogously $O(Pk)$ for the multiplication with V^T.

Additionally, we have to ensure that the inner products within the iterative solver are computed efficiently. Due to the properties of the trace operator,[3] we are in luck, as

$$\operatorname{trace}\left(\underbrace{\left(U_X V_X^T\right)^T}_{\text{Large}} \underbrace{\left(U_Y V_Y^T\right)}_{\text{Large}}\right) = \operatorname{trace}\left(\underbrace{V_Y^T V_X}_{\text{Small}} \underbrace{U_X^T U_Y}_{\text{Small}}\right)$$

allows us to compute the trace of small matrices rather than of those from the full model.

Having discussed the low-rank solver, we proceed to the next section to investigate its performance in conjunction with the preconditioners.

6. Numerical experiments.

To demonstrate the performance of the approach presented in this paper, we consider the two-dimensional (2D) version of our model problem (3.1), which was studied in, for instance, [22]. More precisely, we choose $f = 1$ and $\mathcal{D} = [-1,1]^2$. The random input κ is characterized by the covariance function

$$C_\kappa(\mathbf{x},\mathbf{y}) = \sigma^2 \exp\left(-\frac{|x_1 - y_1|}{\ell_1} - \frac{|x_2 - y_2|}{\ell_2}\right) \quad \forall(\mathbf{x},\mathbf{y}) \in \mathcal{D}. \tag{6.1}$$

The eigenpairs (λ_j, φ_j) of the KL expansion of κ are given explicitly in [13]. In the simulations, we set the correlation lengths $\ell_1 = \ell_2 = 1$ and the mean of the random field $\bar{\kappa} = 1$. Note that, as already reported in the literature, e.g, [22], decreasing the correlation lengths slows down the decay of the eigenvalues in the KLE of κ, and therefore more random variables are then required to sufficiently capture the randomness in the model. In other words, the resulting effect is an increase in the parameter N. The reverse is the case when the correlation lengths are increased.

Next, we investigate the behavior of the solvers for different values of the discretization parameters J, N, Q, σ_κ. Moreover, we choose $\xi = \{\xi_1, \ldots, \xi_N\}$ such that $\xi_j \sim \mathcal{U}[-1,1]$, and $\{\psi_j\}$ are N-dimensional Legendre polynomials with support in $[-1,1]^N$. We perform spatial discretization using $Q1$ finite elements. Moreover, all the numerical experiments are performed

[3] Recall that $\langle X, Y\rangle = \operatorname{vec}(X)^T \operatorname{vec}(Y) = \operatorname{trace}\left(X^T Y\right)$.

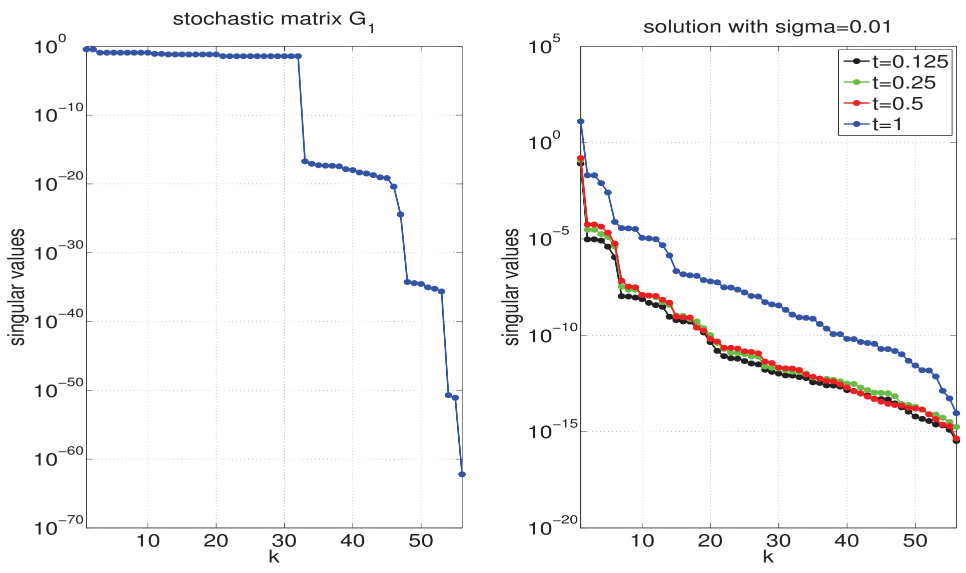

Figure 1. *Singular value decay of the stochastic matrix G_1 (left) and the right-hand sides at different time steps $T \in \{0.125, 0.25, 0.5\}$, as well as the final solution at $T = 1$ (right), with $J = 6080$, $N = 5, Q = 3$, $\sigma_\kappa = 0.01$, and tol $= 10^{-8}$.*

on an Ubuntu Linux machine with 2GB RAM using MATLAB 7.14 together with a MATLAB version of `HSL MI20` [5] based on the classical AMG method as described in [26]. We implement each of the two preconditioners $\mathcal{M}_0$ and $\mathcal{M}_1$ using one V-cycle of AMG with symmetric Gauss–Seidel (SGS) smoothing to approximately invert $\tilde{K}_0$. We remark here that we apply the method as a black box in each experiment and the setup of the approximation to $\tilde{K}_0$ only needs to be performed once. Also, no parallelism is exploited at any stage of all the simulations. In the considered examples, the linear systems are solved for time $T = 1$ and 16 time steps. All figures are obtained with the mean-based preconditioner $\mathcal{M}_0$. Unless otherwise stated, all iterations for all solvers herein are terminated when the relative residual error, measured in the Euclidean norm, is reduced to $tol = 10^{-4}$. We remark here that the stopping iteration tolerance tol should be chosen such that the truncation tolerance $trunctol \leq tol$; otherwise, one would be essentially iterating on the "noise" from the low-rank truncations, as it were.

First, in Figures 1, 2, and 3, we illustrate, for the 2D model problem, the singular value decay of the stochastic matrix G_1, as well as those of the right-hand sides at different time steps and the final solution at $T = 1$. In these figures, we see that the decay is slow. Nevertheless, the matrix G_1 is rank deficient, which justifies its low-rank representation in Theorem 4.3. We note here that the singular values of the stochastic matrices G_k are indeed the same since the matrices are permutations of one another and, hence, their ranks are equal. In particular, their rank is roughly $P/2$ for all $k > 0$. However, as already pointed out, G_0 is diagonal and of full rank P.

Next, as an illustration of the results of Theorem 4.3, observe first from Figure 1 that the rank of the matrices G_k (represented here by G_1) is 32, while $P = 56$. Now, recall from the theorem that the rank of the low-rank solution is determined mainly by those of the stochastic

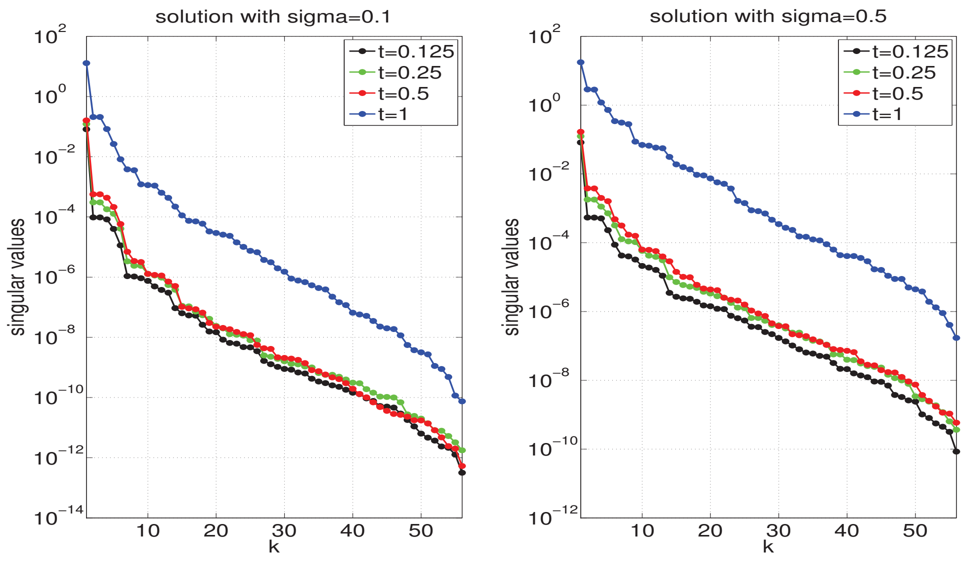

Figure 2. *Singular value decay of the right-hand sides at different time steps* $T \in \{0.125, 0.25, 0.5\}$*, as well as the final solution at* $T = 1$*, with* $\sigma_\kappa = 0.1$ *(left) and with* $\sigma_\kappa = 0.5$ *(right) using* $J = 6080$*,* $N = 5$*,* $Q = 3$*, and tol* $= 10^{-8}$.

matrices G_k regardless of the dimension of the stiffness matrices $\tilde{K}_k$. More precisely, we have from the figure and the theorem that $r = \sum_{j=1}^{N} r_j = 5 \times \text{rank}(G_1) = 5 \times 32 = 160$. Thus, with the truncation tolerance $trunctol = 10^{-10}$, for example, we see from Figure 1 that the tensor ranks of the right-hand sides $\mathbf{b}^n$ are at most ℓ, where $\ell = 20$. Hence, one can approximate the solution to the linear systems with a solution vector whose tensor rank at each time step equals $(160 + \ell)(2k + 1) < 180(2k + 1)$, where $k \in \mathbb{N}$. So, it turns out that the result of the theorem is particularly important if the size of the stiffness matrices $\tilde{K}_k$ increases; that is, $J \to \infty$, while P is kept constant. As for the right-hand sides, the decays at all the respective time steps (e.g., $T = 0.125, 0.25, 0.5$) are quite similar. Thus, we truncate the right-hand sides with the same truncation tolerance. Figures 1 and 2, in particular, illustrate that, keeping other parameters fixed, increasing the variance of κ slows down the decay of the singular values of both the right-hand sides and the final solution.

Tables 1, 2, 3, 4, and 5 report further the results of the simulations of the model. Here, the linear systems are solved using the LRPCG algorithm, as well as using the standard preconditioned CG method, which we have denoted as full model (FM), that is, without low-rank truncation. As benchmarks to compare the performance of the solution methods, we report the total iteration counts, the total CPU times, memory requirements (in kilobytes), the ranks of the truncated solutions, and the relative error from the LRPCG solution with respect to the FM solution, measured in the Euclidean norm. By the memory requirement of a low-rank solution $X = UV^T$ we mean the sum of the two separate computer memories occupied by its factors U and V^T, since X is computed and stored in this format, unlike the solution from FM.

In Tables 1, 2, and 5, we show results for varying P, J, and σ_κ, respectively, while keeping

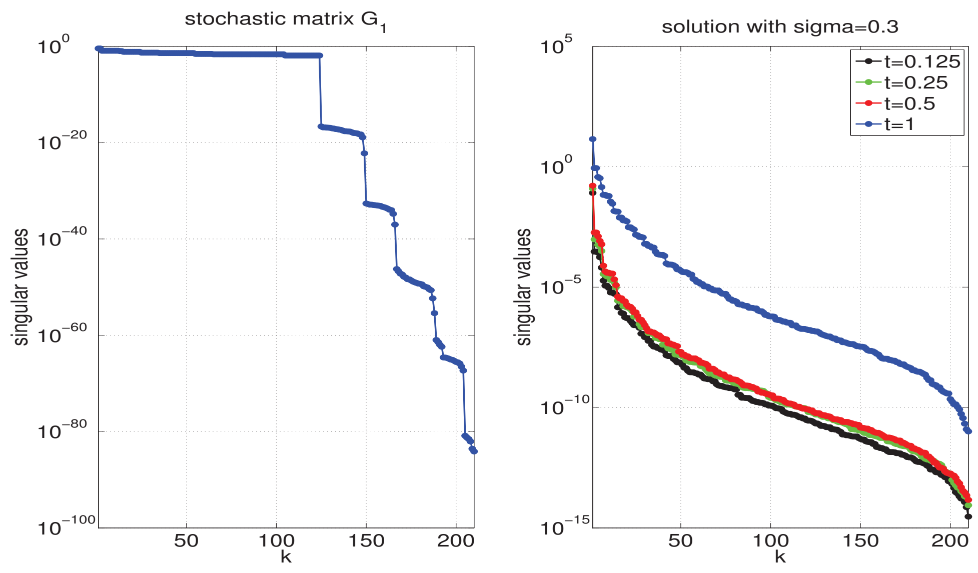

Figure 3. *Singular value decay of the stochastic matrix G_1 (left) and the right-hand sides at different time steps $t \in \{0.125, 0.25, 0.5\}$, as well as the final solution at $t = T$ (right), with $J = 6080$, $N = 6$, $Q = 4$, $\sigma_\kappa = 0.3$, and $tol = 10^{-8}$.*

other parameters constant. In all the tables reported in this paper, the second and third columns show the outputs from the LRPCG, while the last two are from FM (using just the MATLAB command `pcg`). Also in the second and third columns, the quantities in brackets are outputs computed with the corresponding truncation tolerance. Note that in Tables 1 and 4, we have specifically used the tuple of parameters (N, Q, P). Thus, $(5, 3, 56)$, for example, implies that $N = 5$, $Q = 3$, and $P = 56$ (cf. (3.9)). The results in all the tables are intended to give insights regarding the capabilities of the solvers.

A major general observation from Tables 1, 2, 3, and 5 is that for a relatively small variance (that is, $\sigma_\kappa \leq 0.2$) and independently of the preconditioner used, the low-rank approach LRPCG clearly performs better than the conventional method FM in terms of both CPU times and memory requirements, while maintaining fairly the same iteration counts as FM. From Tables 2 and 3, the efficiency as $J \to \infty$ of the LRPCG compared to FM with respect to CPU times and memory reduction is particularly noteworthy. For instance, if $J = 24448$ and $trunctol = 10^{-6}$, we see that the low-rank approach reduces the computational time by roughly 10 times and memory required to store the solution by 4 times, while maintaining the same iteration counts with FM. In fact, this observation further corroborates the theoretical implication of Theorem 4.3 that the low-rank approach is of particular interest if the size of the stiffness matrices $\tilde{K}_k$ gets arbitrarily large; FM deteriorates in this case, as it suddenly struggles to cope with the increased computational complexity. Note in particular from Table 3 that with FM, MATLAB indeed fails with $J = 392704$ and $P = 210$, as the size of the global stochastic Galerkin matrix $\mathcal{A}$ at each time step is now increased to more than 82 million degrees of freedom. Yet, the LRPCG handles this task in about 200 minutes with $trunctol = 10^{-6}$, that is, roughly 13 minutes per time step. In this case, however, we are not able to report the

Table 1

Simulation results showing relative errors, total CPU times (in seconds), ranks of truncated solutions, memory (in KB), and total number of iterations from preconditioned low-rank solvers (second and third columns) compared with those from standard preconditioned CG (last two columns) for $\sigma_\kappa = 0.01$, $J = 6080$, *and various tuples* (N, Q, P).

Time steps = 16 Truncation tol	$\mathcal{M}_0$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_1$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_0$ + FM	$\mathcal{M}_1$ + FM
Par= (3,3,20)				
Ranks	6 (8)	6 (10)		
Memory	381.3 (524.2)	285.9 (571.9)	950	950
#iter	32 (32)	32 (32)	32	32
Total CPU time	20.4 (21.9)	20.7 (21.1)	119.4	123.6
Rel error	8.0e-5 (8.9e-6)	2.4e-4 (1.1e-6)		
Par= (5,3,56)				
Ranks	9 (12)	9 (16)		
Memory	527.3 (814.9)	431.4 (910.8)	2660	2660
#iter	32 (32)	32 (32)	32	32
Total CPU time	52.4 (58.0)	54.7 (58.8)	197.0	195.1
Rel error	2.2e-4 (1.2e-5)	4.0e-4 (4.1e-6)		
Par= (4,4,70)				
Ranks	8 (10)	8 (13)		
Memory	480.5 (672.6)	384.4 (768.7)	3325	3325
#iter	33 (32)	32 (33)	32	32
Total CPU time	54.5 (52.7)	54.5 (57.3)	208.5	208.3
Rel error	8.0e-5 (1.3e-5)	3.3e-4 (3.8e-6)		
Par= (6,3,84)				
Ranks	9 (14)	10 (18)		
Memory	577.9 (866.8)	481.6 (1059.4)	3990	3990
#iter	32 (32)	32 (32)	32	32
Total CPU time	139.6 (133.1)	112.5 (156.1)	228.1	229.9
Rel error	3.0e-4 (1.3e-5)	4.4e-4 (4.3e-6)		

relative error, unlike in the other tables, because the solution from FM terminates with "out of memory," which we have denoted as "OoM." On the other hand, if J is relatively small and P is varied as in Table 4, then FM does better than LRPCG in terms of CPU time only. Although reported only for the case $\sigma_\kappa = 0.01$ in Table 2, we also observed a similar trend as σ_κ is varied and a small J is kept constant. But then, in practical applications one is usually more interested in large-scale simulations in which case (J and P are large and) LRPCG will naturally be a preferred option. Another key observation evident from all the tables is that decreasing the truncation tolerance generally reduces the relative error but, as expected, at the cost of comparatively more computational time and memory requirements.

Regarding the preconditioners, we note that, compared to the Ullmann preconditioner $\mathcal{M}_1$, the mean-based preconditioner $\mathcal{M}_0$ generally yields lower ranks of the low-rank solution, less CPU time, and fewer memory requirements for small truncation tolerance. However, both of them maintain relatively equal iteration counts either with LRPCG or FM.

Notwithstanding the advantages enjoyed by LRPCG as outlined above, its performance is adversely affected by increase in the standard deviation σ_κ of the input data. This observation is also true of FM, albeit to a lesser degree. It is indeed evident from Table 5 that both

Table 2

Simulation results showing relative errors, total CPU times (in seconds), ranks of truncated solutions, memory (in KB), and total number of iterations from preconditioned low-rank solvers (second and third columns) compared with those from standard preconditioned CG (last two columns) for $N = 6$, $Q = 3$ (i.e., $P = 84$), $\sigma_\kappa =$ 0.01, and various J.

Time steps = 16 Truncation tol	$\mathcal{M}_0$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_1$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_0$ + FM	$\mathcal{M}_1$ + FM
$J = 368$				
Ranks	10 (14)	10 (17)		
Memory	42.4 (68.1)	42.7 (77.7)	241.5	241.5
#iter	32 (32)	32 (32)	32	32
Total CPU time	35.0 (41.1)	45.9 (43.5)	10.2	14.2
Rel error	3.0e-4 (1.2e-5)	4.2e-4 (6.0e-6)		
$J = 1504$				
Ranks	9 (14)	10 (17)		
Memory	148.8 (223.3)	124.1 (260.5)	987	987
#iter	32 (32)	32 (33)	32	32
Total CPU time	64.8 (66.5)	69.7 (70.0)	21.8	27.2
Rel error	3.0e-4 (1.2e-5)	4.4e-4 (6.0e-6)		
$J = 6080$				
Ranks	9 (14)	10 (18)		
Memory	577.9 (866.8)	481.6 (1059.4)	3990	3990
#iter	32 (32)	32 (32)	32	32
Total CPU time	139.6 (133.1)	112.5 (156.1)	228.1	229.9
Rel error	3.0e-4 (1.3e-5)	4.4e-4 (4.3e-6)		
$J = 24448$				
Ranks	9 (14)	10 (18)		
Memory	2299.9 (3449.8)	1916.5 (4216.4)	16044	16044
#iter	32 (32)	32 (32)	32	32
Total CPU time	352.0 (426.7)	347.8 (419.4)	3769.4	3853.4
Rel error	3.0e-4 (1.3e-5)	4.5e-4 (4.3e-6)		

LRPCG and FM exhibit deteriorating performance as σ_κ increases, regardless of which of the two preconditioners (that is, $\mathcal{M}_0$ or $\mathcal{M}_1$) is used. Accordingly, the decay of singular values of the solution matrices becomes slower and slower, as demonstrated earlier by Figures 1, 2, and 3. Furthermore, as we can see from Table 5, even though relatively high variance limits the performance of both considered preconditioners, $\mathcal{M}_0$ tends to be more adversely affected by the increase than $\mathcal{M}_1$ in terms of both iteration counts and CPU time. This is perhaps explained by the fact that, unlike $\mathcal{M}_1$, the mean-based preconditioner $\mathcal{M}_0$ is block diagonal; thus, as σ_κ increases, we see from (3.22), (3.23), and (3.27) that the off-diagonal blocks of the global stochastic Galerkin matrix $\mathcal{A}$ become more significant and they are not represented in the preconditioner. Here, we note, in particular, that the deteriorating performance of $\mathcal{M}_0$ as variance increases confirms a similar observation made in an earlier study [22] by Powell and Elman in which CG was preconditioned with $\mathcal{M}_0$ but without low-rank truncations. Due to this drawback, we remark here that we have done most of our computations using relatively small values of variance. In particular, we used $\sigma_\kappa = 0.01$ to obtain the results in Tables 1, 2, 3, and 4. We also did further experiments with $\sigma_\kappa \in \{0.1, 0.2\}$ and *trunctol* $= 10^{-8}$ and made similar observations in the performance of LRPCG and FM.

Table 3

Simulation results showing total CPU times (in seconds), ranks of truncated solutions, memory (in KB), and total number of iterations using low-rank preconditioned CG for $J = 392704$, $N = 6$, $Q = 4$ (i.e., $P = 210$), and various σ_κ.

Time steps $=16$ Trunc tol	$\mathcal{M}_0$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_1$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_0$ or $\mathcal{M}_1$ + FM
$\sigma_\kappa = 0.001$			
Ranks	1 (10)	2 (12)	
Memory	3069.6 (49114.25)	6139.2 (49114.25)	
#iter	24 (32)	32 (32)	
Total CPU time	2680.7 (4775.5)	3335.4 (4944.9)	OoM
$\sigma_\kappa = 0.01$			
Ranks	9 (12)	10 (18)	
Memory	36835.7 (55253.5)	30696.4 (67532.1)	
#iter	32 (32)	32 (32)	
Total CPU time	4157.3 (5149.9)	4115.3 (5249.8)	OoM
$\sigma_\kappa = 0.1$			
Ranks	20 (47)	20 (52)	
Memory	89019.6 (174969.5)	82880.3 (171899.9)	
#iter	49 (49)	51 (48)	
Total CPU time	8354.5 (12419.0)	8069.3 (11801.0)	OoM

Table 4

Simulation results showing relative errors, total CPU times (in seconds), ranks of truncated solutions, memory (in KB), and total number of iterations from preconditioned low-rank solvers (second and third columns) compared with those from standard preconditioned CG (last two columns) for $\sigma_\kappa = 0.01$, $J = 1504$, and various tuples (N, Q, P).

Time steps $=16$ Truncation tol	$\mathcal{M}_0$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_1$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_0$ + FM	$\mathcal{M}_1$ + FM
Par= (3,3,20)				
Ranks	6 (8)	6 (10)		
Memory	95.25 (130.9)	71.4 (142.9)	235	235
#iter	32 (32)	32 (32)	32	32
Total CPU time	19.4 (17.0)	19.9 (18.1)	12.7	15.3
Rel error	8.0e-5 (8.7e-6)	2.4e-4 (1.0e-6)		
Par= (5,3,56)				
Ranks	9 (12)	9 (16)		
Memory	134.1 (207.2)	109.7 (243.8)	658	658
#iter	33 (32)	32 (33)	32	32
Total CPU time	63.8 (69.4)	69.2 (69.5)	20.0	23.6
Rel error	2.2e-4 (1.2e-5)	3.9e-4 (4.1e-6)		

Finally, to demonstrate the behavior of the first two moments (that is, the mean and the variance; cf. (3.13) and (3.14)) of the solution from the MATLAB solver `pcg` and the low-rank solver, we have included Figures 4, 5, 6, and 7. In Figures 4 and 5, we see that the low-rank truncations do not increase the moments. The same explanation holds for Figures 6 and 7. However, in both cases, it is observed that an increase in variance of the random input yields a similar effect on the solution variance.

Table 5

Simulation results showing relative errors, total CPU times (in seconds), ranks of truncated solutions, memory (in KB), and total number of iterations from preconditioned low-rank solvers (second and third columns) compared with those from standard preconditioned CG (last two columns) for $J = 6,080$, $N = 6$, $Q = 3$ (i.e., $P = 84$), and various σ_κ.

Time steps = 16 Truncation tol	$\mathcal{M}_0$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_1$ + LRPCG $10^{-4}(10^{-6})$	$\mathcal{M}_0$ + FM	$\mathcal{M}_1$ + FM
$\sigma_\kappa = 0.001$				
Ranks	1 (13)	2 (12)		
Memory	48.2 (770.5)	96.3 (770.5)	3990	3990
#iter	23 (32)	32 (32)	32	32
Total CPU time	120.2 (112.5)	136.6 (122.5)	222.4	224.9
Rel error	1.0e-3 (6.1e-7)	2.4e-4 (3.3e-6)		
$\sigma_\kappa = 0.01$				
Ranks	9 (14)	10 (18)		
Memory	577.9 (866.8)	481.6 (1059.4)	3990	3990
#iter	32 (32)	32 (32)	32	32
Total CPU time	139.6 (133.1)	112.5 (156.1)	228.1	229.9
Rel error	3.0e-4 (1.3e-5)	4.4e-4 (4.3e-6)		
$\sigma_\kappa = 0.1$				
Ranks	27 (54)	21 (55)		
Memory	2070.7 (2744.9)	1348.4 (2696.8)	3990	3990
#iter	49 (49)	48 (48)	49	48
Total CPU time	206.0 (275.7)	196.4 (283.7)	342.6	352.6
Rel error	8.7e-4 (1.1e-4)	9.0e-4 (2.1e-4)		
$\sigma_\kappa = 0.2$				
Ranks	46 (73)	49 (77)		
Memory	3033.8 (3804.3)	2985.7 (3804.3)	3990	3990
#iter	65 (72)	65 (64)	65	64
Total CPU time	200.6 (353.8)	218.6 (286.0)	508.9	524.9
Rel error	1.3e-4 (3.2e-4)	9.3e-4 (2.9e-6)		
$\sigma_\kappa = 0.3$				
Ranks	81 (84)	84 (84)		
Memory	(5393.5) (5700.6)	6308.5 (5778.8)	3990	3990
#iter	102 (242)	90 (108)	83	80
Total CPU time	911.6 (5478.7)	750.6 (1251.9)	590.1	835.1
Rel error	1.0e-3 (2.8e-4)	9.5e-4 (3.7e-4)		

7. Conclusions and outlook.

The use of classical spectral SGFEM in discretizing linear PDEs with uncertain inputs is standard in the literature. For it to compete favorably with other approaches like the MCM and SCM in solving time-dependent problems, efficient solvers with appropriate preconditioners have to be developed to solve the resulting large-dimensional coupled linear system. With a view to reducing the computational time and memory requirements of the solution of such arbitrarily large linear systems, we have provided a theoretical basis for a low-rank solver to achieve these goals. More precisely, we solved the linear systems (3.25) using a low-rank CG iterative solver, together with two different preconditioners. In general, the combination of each of the preconditioners and the low-rank iterative solver seems quite promising for large-scale simulation of models whose random input data have comparatively low variance, as it reduces the computer memory and computational time required to

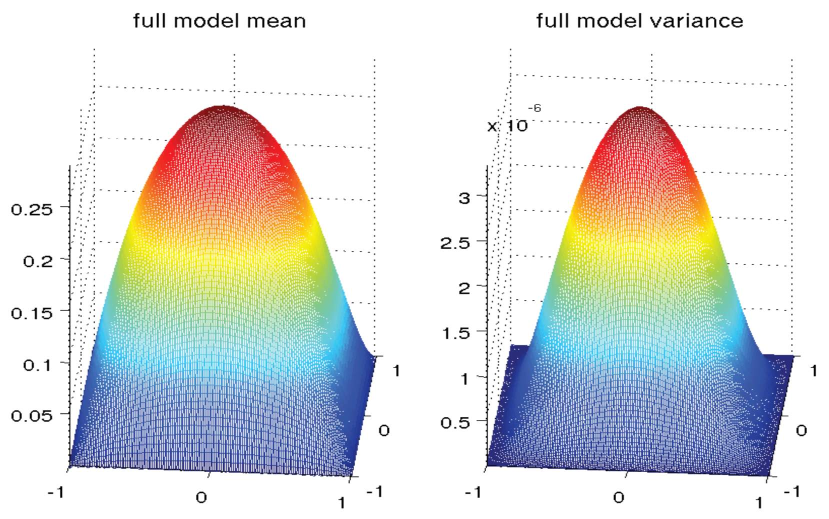

Figure 4. *The mean (left) and variance (right) of the solution from the FM with* $J = 6080$*,* $N = 5$*,* $Q = 3$*, and* $\sigma_\kappa = 0.01$*.*

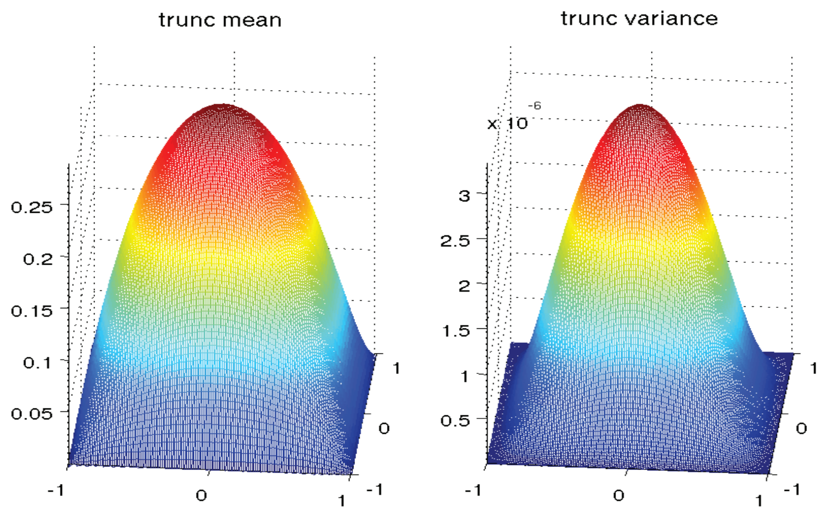

Figure 5. *The mean (left) and variance (right) of the low-rank (truncated) solution with* $J = 6080$*,* $N = 5$*,* $Q = 3$*,* $\sigma_\kappa = 0.01$*, and trunctol* $= 10^{-6}$*. The relative error of the truncated solution with respect to the FM solution is* 8.7×10^{-6}*.*

solve the stochastic Galerkin linear system compared to the conventional method. Although the low-rank approach introduces further error in the simulation due to the low-rank truncations, the relative tolerance of the truncation operator can be so tightened that the error will become negligible. Most importantly, even though the low-rank truncation does not come free of charge it enables the solution of unsteady UQ problems that would otherwise be intractable.

We would like to point out here that the low-rank approach discussed in this paper is not

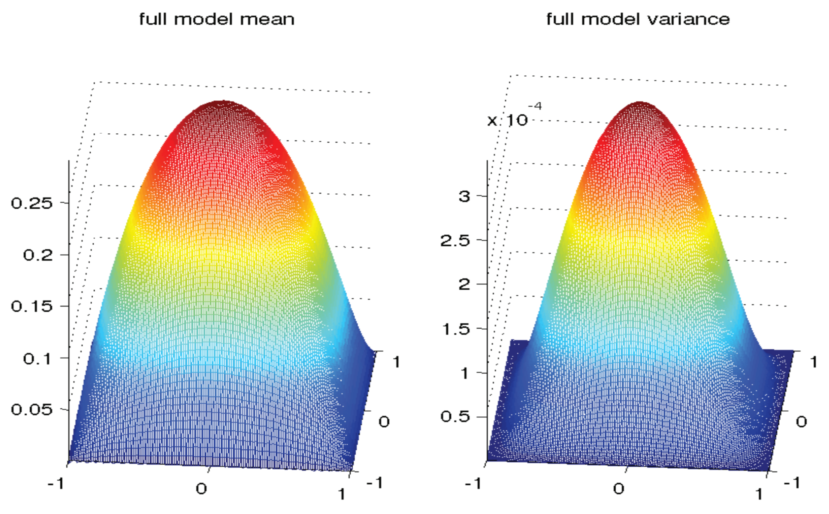

Figure 6. *The mean (left) and variance (right) of the solution from the FM with* $J = 6080$, $N = 5$, $Q = 3$, *and* $\sigma_\kappa = 0.1$.

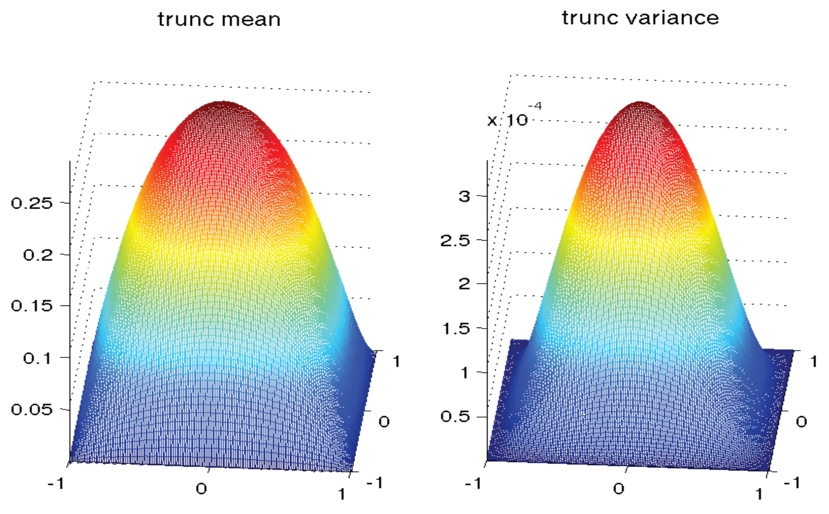

Figure 7. *The mean (left) and variance (right) of the low-rank (truncated) solution with* $J = 6080$, $N = 5$, $Q = 3$, $\sigma_\kappa = 0.1$, *and trunctol* $= 10^{-6}$. *The relative error of the truncated solution with respect to the FM solution is* 8.0×10^{-5}.

only applicable to time-dependent problems, but also one can apply it to stationary problems.

Acknowledgments. The authors would like to thank Tobias Breiten and Piotr Skrzypacz for fruitful discussions in the course of writing this paper. Our profound gratitude also goes to the two anonymous referees for their keen error-spotting eyes; their invaluable comments and constructive criticisms were undoubtedly quite helpful in improving the quality of this paper.

REFERENCES

[1] I. Babuška and P. Chatzipantelidis, *On solving elliptic stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 4093–4122.

[2] 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.

[3] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.

[4] P. Benner and T. Breiten, *Low-rank methods for a class of generalized Lyapunov equations and related issues*, Numer. Math., 124 (2013), pp. 441–470.

[5] J. Boyle, M. D. Mihajlovic, and J. A. Scott, *HSL MI20: An Efficient AMG Preconditioner*, Tech. Report RAL-TR-2007-021, Rutherford Appleton Laboratory (CCLRC), Oxfordshire, UK, 2007.

[6] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, *Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients*, Comput. Vis. Sci., 14 (2011), pp. 3–15.

[7] T. Damm, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations*, Numer. Linear Algebra Appl., 15 (2008), pp. 853–871.

[8] H. Elman and D. Furnival, *Solving steady-state diffusion problem using multigrid*, IMA J. Numer. Anal., 27 (2007), pp. 675–688.

[9] O. G. Ernst, A. Mugler, H.-J. Starkloff, and E. Ullmann, *On the convergence of generalized polynomial chaos expansions*, ESAIM Math. Model. Numer. Anal., 46 (2012), pp. 317–339.

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

[11] P. Frauenfelder, C. Schwab, and R. A. Todor, *Finite elements for elliptic problems with stochastic coefficients*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 205–228.

[12] R. G. Ghanem and R. M. Kruger, *Numerical solution of spectral stochastic finite element systems*, Comput. Methods Appl. Mech. Engrg., 129 (2005), pp. 289–303.

[13] R. G. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, 1996.

[14] G. H. Golub and C. H. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1996.

[15] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.

[16] M. E. Hochstenbach, *A Jacobi–Davidson type SVD method*, SIAM J. Sci. Comput., 23 (2001), pp. 606–628.

[17] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714.

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

[19] O. P. Le Maitre, O. M. Knio, B. J. Debusschere, H. N. Najm, and R. G. Ghanem, *A multigrid solver for two-dimensional stochastic diffusion equations*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 4723–4744.

[20] G. J. Lord, C. E. Powell, and T. Shardlow, *An Introduction to Computational Stochastic PDEs*, Cambridge University Press, New York, 2014.

[21] F. Nobile and R. Tempone, *Analysis and implementation issues for the numerical approximation of parabolic equations with random coefficients*, Internat. J. Numer. Methods Engrg., 80 (2009), pp. 979–1006.

[22] C. E. Powell and H. C. Elman, *Block-diagonal preconditioning for spectral stochastic finite-element systems*, IMA J. Numer. Anal., 29 (2009), pp. 350–375.

[23] E. Rosseel, T. Boonen, and S. Vandewalle, *Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients*, Numer. Linear Algebra Appl., 15 (2008), pp. 141–163.

[24] M. Stoll, *A Krylov-Schur approach to the truncated SVD*, Linear Algebra Appl., 436 (2012), pp. 2795–2806.

[25] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci.

Comput., 37 (2015), pp. B1–B29.

[26] K. Stüben, *An introduction to algebraic multigrid*, in Multigrid, A. Schuller, U. Trottenberg, and C. Oosterlee, eds., Academic Press, San Diego, CA, 2001, pp. 413–532.

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

[28] C. F. Van Loan and N. Pitsianis, *Approximation with Kronecker products*, in Linear Algebra for Large Scale and Real-Time Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992, pp. 293–314.

[29] D. Xiu and G. E. Karniadakis, *A new stochastic approach to transient heat conduction modeling with uncertainty*, Int. J. Heat Mass Transfer, 46 (2003), pp. 4681–4693.

[30] D. Xiu and J. Shen, *Efficient stochastic Galerkin methods for random diffusion*, J. Comput. Phys., 228 (2009), pp. 266–281.