AN EFFICIENT REDUCED BASIS SOLVER FOR STOCHASTIC GALERKIN MATRIX EQUATIONS.

C.E. POWELL *, D. SILVESTER†, AND V. SIMONCINI‡

Abstract. Stochastic Galerkin finite element approximation of PDEs with random inputs leads to linear systems of equations with coefficient matrices that have a characteristic Kronecker product structure. By reformulating the systems as multi-term linear matrix equations, we develop an efficient solution algorithm which generalizes ideas from rational Krylov subspace approximation. The new approach determines a low-rank approximation to the solution matrix by performing a projection onto a low-dimensional space and provides an efficient solution strategy whose convergence rate is independent of the spatial approximation. Moreover, it requires far less memory than the standard preconditioned conjugate gradient method applied to the Kronecker formulation of the linear systems.

Key words. generalized matrix equations, PDEs with random data, stochastic finite elements, iterative solvers, rational Krylov subspace methods.

AMS subject classifications. 35R60 , 60H35, 65N30, 65F10

1. Introduction. This paper is concerned with the design and implementation of efficient iterative solution algorithms for high-dimensional linear algebra systems that arise from Galerkin approximation of elliptic PDE problems with correlated random inputs. The tensor product structure of the Galerkin approximation space and the low-rank structure that is inherent in the discrete systems is exploited in our innovative solution strategy. This strategy builds on recent progress in rational Krylov subspace approximation (see, for example, Simoncini [21]) and generates an accurate reduced basis approximation of the spatial component of the Galerkin solution.

We focus on stochastic steady-state diffusion equations with homogeneous Dirichlet boundary conditions. To this end, let $D \subset \mathbb{R}^2$ be a sufficiently regular spatial domain and let Ω be a sample space associated with a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Our goal is to approximate $u : D \times \Omega \to \mathbb{R}$ such that $\mathbb{P}$-a.s.,

$$\begin{aligned} -\nabla \cdot (a(\vec{x},\omega)\nabla u(\vec{x},\omega)) = &\quad f(\vec{x}) \quad \text{in} \quad D \\ u(\vec{x},\omega) = &\quad 0 \qquad \text{on} \quad \partial D. \end{aligned} \tag{1.1}$$

We assume that f is deterministic and a is a random field that can be expressed as a linear function of a finite number of real-valued random variables $\xi_r : \Omega \to \Gamma_r \subset \mathbb{R}$ of the form

$$a(\vec{x},\omega) = a_0(\vec{x}) + \sum_{r=1}^{m} a_r(\vec{x})\,\xi_r(\omega). \tag{1.2}$$

A common choice is a truncated Karhunen–Loève (KL) expansion (see Lord et al. [13] for further discussion). In that case,

$$a(\vec{x},\omega) = \mu(\vec{x}) + \sigma \sum_{r=1}^{m} \sqrt{\lambda_r}\phi_r(\vec{x})\,\xi_r(\omega), \tag{1.3}$$

where μ is the expected value of the diffusion coefficient, σ is the standard deviation and $\{\lambda_r, \phi_r\}$ are eigenpairs of the integral operator $\mathcal{B}$ associated with $B(\vec{x}_1, \vec{x}_2\,) = (1/\sigma^2)\,C(\vec{x}_1, \vec{x}_2\,)$ where

*School of Mathematics, University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom (c.powell@manchester.ac.uk).

†School of Mathematics, University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom (d.silvester@manchester.ac.uk).

‡Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato 5, I-40127 Bologna, Italy, and IMATI-CNR, Pavia, Italy (valeria.simoncini@unibo.it).

$C : D \times D \to \mathbb{R}$ is the covariance function. We assume $\lambda_1 \geq \lambda_2 \geq \cdots$ so that the terms retained in (1.3) correspond to the m largest eigenvalues of the covariance.

Weak formulations of (1.1) have been well studied in the literature (e.g., see [13, Chapter 9]) and well-posedness follows straightforwardly from the Lax–Milgram lemma if realizations of a are positive and bounded. To ensure this, we restrict our attention to uniform random variables ξ_r on $\Gamma_r = [-1, 1]$ and assume that there exist constants $a_0^{\min}$ and $a_0^{\max}$ satisfying

$$0 < a_0^{\min} \leq a_0(\vec{x}) \leq a_0^{\max} < \infty, \qquad \text{a.e. in } D \tag{1.4}$$

and the coefficient functions a_r satisfy

$$\sum_{r=1}^{m} \|a_r\|_\infty < a_0^{\min}. \tag{1.5}$$

We will further assume that the random variables ξ_r are independent and m is chosen so that the approximation error associated with the truncation of the series (1.2) is sufficiently small. Stochastic Galerkin methods (introduced by Babuška and collaborators [1], [4]) seek approximations to the weak solution u of (1.1) in finite-dimensional approximation spaces of the form $Z_h \otimes S_p$ where $Z_h \subset H_0^1(D)$ and $S_p \subset L_\varrho^2(\Gamma)$. As usual, $H_0^1(D)$ is the Sobolev space associated with D, and $L_\varrho^2(\Gamma)$ denotes the set of functions v on $\Gamma := \Gamma_1 \times \cdots \times \Gamma_m = [-1, 1]^m$ satisfying

$$\langle v, v \rangle_\varrho := \int_\Gamma \varrho(\mathbf{y}) v(\mathbf{y})^2 \, d\mathbf{y} < \infty,$$

where $\varrho(\mathbf{y})$ is the joint density of $[\xi_1, \ldots, \xi_m]$ and $\mathbf{y} = [y_1, \ldots, y_m]$ with $y_r := \xi_r(\omega)$. In particular, we choose $Z_h = \operatorname{span}\{\varphi_1(\vec{x}), \ldots, \varphi_{n_x}(\vec{x})\}$ to be a finite element space associated with a spatial mesh of D with characteristic element size h and we choose $S_p = \operatorname{span}\{\psi_1(\mathbf{y}), \ldots, \psi_{n_\xi}(\mathbf{y})\}$ to be the set of multivariate polynomials of total degree p or less in $y_1, \ldots, y_m$ on Γ. This yields

$$n_\xi = \dim(S_p) = \frac{(m+p)!}{m!p!}. \tag{1.6}$$

The structure of the expansion (1.2) and the tensor product form of the approximation space $Z_h \otimes S_p$ leads to a structured linear algebraic system $\mathcal{A}\mathbf{x} = \mathbf{b}$ of $n_x \cdot n_\xi$ equations with

$$\mathcal{A} = G_0 \otimes K_0 + \sum_{r=1}^{m} G_r \otimes K_r, \qquad \mathbf{b} = \mathbf{g}_0 \otimes \mathbf{f}_0. \tag{1.7}$$

The vector $\mathbf{x}$ contains the coefficients in the expansion of the stochastic Galerkin approximation in the tensor product basis $\{\varphi_i \psi_j\}, i = 1, \ldots, n_x, j = 1, \ldots, n_\xi$. The matrices K_0 and K_r are finite element matrices of size $n_x \times n_x$ associated with the space Z_h and the matrices G_0 and G_r associated with the space S_p are of size $n_\xi \times n_\xi$. The vector $\mathbf{g}_0$ is the first column of G_0 and $\mathbf{f}_0$ is the vector associated with the finite element discretization of the deterministic analogue of (1.1). We give more details in section 3.

From now on, we will refer to the algebraic linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$ as the *Kronecker formulation* of the discrete problem. Alternatively, by introducing the solution matrix

$$X = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_\xi}\right] \in \mathbb{R}^{n_x \times n_\xi}, \qquad \mathbf{x} = \mathtt{vec}(X)$$

whose jth column is the subvector of $\mathbf{x}$ containing the coefficients associated with the jth basis function ψ_j for S_p, the linear system can also be rewritten as a linear multi-term matrix equation

$$K_0 X G_0^\top + \sum_{r=1}^{m} K_r X G_r^\top = F. \tag{1.8}$$

The $n_x \times n_\xi$ matrix F satisfies $\mathtt{vec}(F) = \mathbf{b}$ or, equivalently, $F = \mathbf{f}_0\,\mathbf{g}_0^\top$. We will refer to (1.8) as the *matrix equation formulation* of the discrete problem.

There is a substantial body of work on solving the Kronecker formulation of discrete problems associated with stochastic Galerkin approximation of elliptic PDEs. Since $\mathcal{A}$ is sparse, symmetric and positive definite, the standard conjugate gradient (CG) method can be applied. However, this must be done in a smart way, without assembling $\mathcal{A}$ and matrix-vector products should be done by exploiting the relation

$$\sum_{r=1}^{m} (G_r \otimes K_r)\,\mathbf{v} = \mathtt{vec}\left(\sum_{r=1}^{m} K_r V G_r^\top\right), \quad \mathbf{v} = \mathtt{vec}(V). \tag{1.9}$$

Preconditioners that exploit the structure of $\mathcal{A}$ are another essential ingredient (see, for example, [17], [23], [22], [18], [8]). In particular, the mean-based preconditioner $P = G_0 \otimes K_0$ analyzed by Powell & Elman in [17] requires n_ξ inexact decoupled solves with K_0 in each CG iteration. Mean-based preconditioning is often used in practice due to its non-intrusive nature, despite its lack of robustness when we increase the ratio σ/μ in the KL expansion (1.3).

We will focus on the matrix equation formulation (1.8) in this paper. In particular, we propose a new method that determines a low-rank approximation to the solution matrix X by performing a projection onto a low-dimensional subspace of $\mathbb{R}^{n_x}$ of typical size much smaller than n_ξ. Our strategy only attempts to achieve a reduction in the complexity associated with the high dimension of the deterministic part of the problem, and so our working assumption is that the dimensions of the stochastic Galerkin spaces S_p and Z_h satisfy $n_\xi \ll n_x$. Our philosophy is that long vectors of size $n_x \cdot n_\xi$ should never be constructed in generating a low-rank approximation. Previous work on low-rank approximation methods has focussed on finding the dominant eigenvectors of (KL like) expansions of the solution matrix (see, for example, Matthies & Keese [15]). There has also been significant recent research on the construction of low-rank tensor decompositions of the discrete solution; in particular, by Khoromskij & Schwab [12], Matthies & Zander [16] and Ballani & Grasedyck [2]. The novel aspect of our strategy is that a reduced approximation space is built up on-the-fly from rational Krylov subspaces. A projection is then used to obtain a reduced matrix equation whose solution yields the low-rank approximation.

An outline of the paper is as follows. The construction of reduced rational Krylov approximation spaces for multi-term linear matrix equations is discussed in general terms in the next section. In section 3, we discuss the model problem (1.1) and modify the associated stochastic Galerkin matrix equations so that the general algorithm from section 2 can be applied efficiently. Details of our implementation are given in section 4. Finally, the performance of the resulting solver is assessed for two test problems in section 5. Our experiments demonstrate that huge computational savings—compared to solving the Kronecker formulation of the matrix equations using standard preconditioned CG—can be achieved when the dimension $n_x \cdot n_\xi$ is large. That is, when one is looking to compute an accurate approximation to the solution of (1.1) on a fine spatial mesh when the coefficient (1.2) is a function of as many as $m = 20$ random variables.

2. Reduced rational Krylov approximation. To introduce the concept of reduced rational Krylov approximation, suppose that we are given a set of m sparse symmetric positive definite matrices $A_1, \ldots, A_m$ of size $n_x \times n_x$ whose spectra are all contained in a small interval $\mathcal{S} \subset \mathbb{R}^+$. The significance of $\mathcal{S}$ will become apparent later. Now suppose that we wish to solve a general multi-term linear matrix equation of the form

$$XB_0 + A_1 X B_1 + \cdots + A_m X B_m = \mathbf{f}_0\,\mathbf{g}_0^\top \tag{2.1}$$

for the unknown matrix X of size $n_x \times n_\xi$ (with $n_\xi \ll n_x$), where the matrices B_r, $r = 0, 1, \ldots, m$ are of size $n_\xi \times n_\xi$, and the vectors $\mathbf{f}_0$ and $\mathbf{g}_0$ are of length n_x and n_ξ respectively. In this

section, we simply view (2.1) as a model problem and temporarily leave aside the specific details of stochastic Galerkin matrix equations. Suppose that we can generate an orthonormal basis for an approximation space $\mathcal{K}_k \subset \mathbb{R}^{n_x}$ of dimension $n_k \ll n_x$. If we collect the n_k basis vectors into a matrix V_k of size $n_x \times n_k$ then an approximate solution to (2.1) can be sought in the form $X_k = V_k Y_k \approx X$, where the $n_k \times n_\xi$ matrix Y_k (the *reduced solution matrix*) can be defined by insisting that the residual

$$R_k := X_k B_0 + A_1 X_k B_1 + \cdots + A_m X_k B_m - \mathbf{f}_0 \mathbf{g}_0^\top$$

satisfies the *Galerkin* condition $V_k^\top R_k = 0$. Substituting $X_k = V_k Y_k$ into this expression we see that Y_k is the solution of the reduced (or *projected*) matrix equation

$$\underbrace{(V_k^\top V_k)}_{I} Y_k B_0 + \underbrace{(V_k^\top A_1 V_k)}_{\overline{A}_1} Y_k B_1 + \cdots + \underbrace{(V_k^\top A_m V_k)}_{\overline{A}_m} Y_k B_m = \underbrace{V_k^\top \mathbf{f}_0}_{\overline{\mathbf{f}_0}} \mathbf{g}_0^\top. \tag{2.2}$$

Comparing (2.2) with (2.1) we see that the left matrices have been reduced in size (from $n_x \times n_x$ to $n_k \times n_k$) but will be dense rather than sparse. The right matrices B_r are unchanged. Readers who are more familiar with the Kronecker formulation of matrix equations will note that (2.2) is equivalent to the algebraic system $\mathcal{A}_k \mathbf{y}_k = \mathbf{b}_k$ where $\mathbf{y}_k = \mathtt{vec}(Y_k)$ and

$$\mathcal{A}_k = B_0 \otimes I + \sum_{r=1}^{m} B_r \otimes \overline{A}_r, \quad \mathbf{b}_k = \mathbf{g}_0 \otimes \overline{\mathbf{f}_0}. \tag{2.3}$$

To effectively implement such a strategy we will need to answer two questions: (i) how do we construct the approximation space $\mathcal{K}_k$, keeping n_k as small as possible while still maintaining accuracy? and (ii) can we solve the reduced matrix equation (2.2) for Y_k efficiently? For the latter, we will consider the Kronecker formulation (2.3). We can compute $\mathbf{y}_k$ using a direct method if $n_k \cdot n_\xi$ is small, or else use a matrix-oriented iterative solver such as CG, taking care to ensure that matrix-vector products are performed efficiently using (1.9). For the former, we want to develop an iterative procedure for building a sequence of approximation spaces $\mathcal{K}_1 \subset \mathcal{K}_2 \subset \ldots \subset \mathcal{K}_j \subset \ldots$ that terminates after k iterations when $\mathcal{K}_k$ is judged to be rich enough. To achieve this objective, starting from the initial (scaled) vector[1] $\mathbf{v}_0 = \mathbf{f}_0/\|\mathbf{f}_0\| \in \mathbb{R}^{n_x}$ and given a vector of real positive parameters $\mathbf{s} = [s_1, s_2, \ldots]$, we will iteratively generate spaces $\mathcal{K}_j$, $j = 1, 2, \ldots$ of increasing dimension n_j that are built from a nested sequence of rational Krylov approximation spaces. We give a brief outline of the construction of these spaces next. All other implementation details are deferred to section 4.

At the first iteration $j = 1$, we start with $V_0 = \mathbf{v}_0$ and compute

$$W = [(A_1 + s_1 I)^{-1}\mathbf{v}_0, \ldots, (A_m + s_1 I)^{-1}\mathbf{v}_0] \in \mathbb{R}^{n_x \times m}.$$

Next, we identify the ℓ_1 most significant directions of this matrix. This is done by computing the singular values σ_i of W and retaining only the ℓ_1 left singular vectors $\mathbf{u}_i$, $i = 1, \ldots, \ell_1$ associated with the values that contribute a fixed proportion $\beta\%$ of the total (typically $\beta = 99$). That is, we choose ℓ_1 such that $\sum_{i=1}^{\ell_1} \sigma_i > \frac{\beta}{100} \sum_{i=1}^{m} \sigma_i$. The approximation space $\mathcal{K}_1$ will be of dimension $n_1 = \ell_1 + 1 \leq m + 1$ and is given by range(V_1) once the vectors have been modified to form an *orthonormal* basis

$$V_1 = \mathtt{orth}\big([\mathbf{v}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{\ell_1}]\big) = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{\ell_1}] \in \mathbb{R}^{n_x \times (\ell_1 + 1)}.$$

[1] The vector $\mathbf{f}_0$ is the initial residual associated with the deterministic algebraic system and a zero initial guess.

The second iteration is defined by computing a new matrix

$$W = [(A_1 + s_2 I)^{-1}\mathbf{v}_1, \ldots, (A_m + s_2 I)^{-1}\mathbf{v}_1],$$

and then computing its singular value decomposition in order to determine which ℓ_2 directions should be kept. The approximation space $\mathcal{K}_2$ has dimension $n_2 = \ell_2 + \ell_1 + 1 \leq 2m + 1$. At the jth iteration, the jth column $\mathbf{v}_{j-1}$ of the basis matrix V_{j-1} is used to expand the space. The procedure continues to enrich the space until, after k steps, an appropriate convergence criterion is satisfied. The orthogonality property $V_k^\top R_k = 0$ ensures finite termination in exact arithmetic, since the exact solution will be determined in an approximation space of dimension at most n_x. However, a good approximation is clearly sought in a space with a much smaller dimension.

The vector of positive parameters $\mathbf{s} = [s_1, s_2, \ldots]$ is selected before the iteration begins. Notice that since each A_r is symmetric and positive definite, then each of the shifted matrices $(A_r + s_j I)$, $r = 1, \ldots, m$ is invertible, regardless of the particular positive value of s_j chosen. We delay a discussion of specific choices for s_j until section 4.2. Here, we simply note that we apply the same parameter s_j when computing each of the m columns of W. In this setting, the user must supply only one parameter s_j at the jth iteration. Of course, the number k of required iterations will not be known in advance so we may cycle through a fixed number of chosen values, say $\mathbf{s} = [s_1, s_2, \ldots, s_{10}]$. For more general multi-term matrix equations, up to m different parameters could be employed in each iteration. However, we will not consider that case here. Depending on the problem, further simplifications may be possible. For instance, it might be appropriate to choose a single value of s_j in all iterations. We will discuss multiple-parameter and parameter-free strategies in section 4.2 and investigate them numerically in section 5. For now, we use the subscript j to allow for the possibility of dynamically updating s_j as the iteration proceeds.

To describe the Krylov approximation space generated, let

$$\mathbb{K}_\ell(A_r, \mathbf{v}_0, \mathbf{s}) := \operatorname{span}\left\{ \mathbf{v}_0, (A_r + s_{r_1} I)^{-1}\mathbf{v}_0, \ldots, \prod_{j=1}^{\ell} (A_r + s_{r_j} I)^{-1}\mathbf{v}_0 \right\}, \tag{2.4}$$

be the rational Krylov subspace associated with the matrix A_r, the initial vector $\mathbf{v}_0$, and the chosen vector of parameters $\mathbf{s}$, where $[s_{r_1}, s_{r_2}, \ldots, s_{r_\ell}]$ contains an appropriate subset of length ℓ of the components of $\mathbf{s}$ (the parameters appear in different orders and combinations for each $r = 1, \ldots, m$). After k iterations, the approximation space contains the rational Krylov subspaces $\mathbb{K}_\ell(A_r, \mathbf{v}_0, \mathbf{s})$, $r = 1, \ldots, m$ for some appropriate values of ℓ (which may be different for each r). However, it is actually a richer space, in the sense that it also contains combinations of rational functions in the coefficient matrices $\{A_1, \ldots, A_m\}$ that are not included in each of the individual Krylov subspaces. As an illustration, consider the specific case of $m = 3$. Suppose we perform three iterations and choose distinct parameters s_1, s_2 and s_3. The process generates the following space (where, for ease of exposition, we do not orthogonalize or deflate the computed vectors)

$$\begin{aligned}
&\operatorname{span}\{\mathbf{v}_0, \underbrace{(A_1 + s_1 I)^{-1}\mathbf{v}_0}_{=:\mathbf{v}_1}, \underbrace{(A_2 + s_1 I)^{-1}\mathbf{v}_0}_{=:\mathbf{v}_2}, \underbrace{(A_3 + s_1 I)^{-1}}_{=:\mathbf{v}_3}\mathbf{v}_0, \\
&\quad (A_1 + s_2 I)^{-1}\mathbf{v}_1, (A_2 + s_2 I)^{-1}\mathbf{v}_1, (A_3 + s_2 I)^{-1}\mathbf{v}_1, \\
&\quad (A_1 + s_3 I)^{-1}\mathbf{v}_2, (A_2 + s_3 I)^{-1}\mathbf{v}_2, (A_3 + s_3 I)^{-1}\mathbf{v}_2\} \\
&\quad = \operatorname{span}\{\mathbf{v}_0, (A_1 + s_1 I)^{-1}\mathbf{v}_0, (A_2 + s_1 I)^{-1}\mathbf{v}_0, (A_3 + s_1 I)^{-1}\mathbf{v}_0, \\
&\quad (A_1 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0, (A_2 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0, (A_3 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0, \\
&\quad (A_1 + s_3 I)^{-1}(A_2 + s_1 I)^{-1}\mathbf{v}_0, (A_2 + s_3 I)^{-1}(A_2 + s_1 I)^{-1}\mathbf{v}_0, (A_3 + s_3 I)^{-1}(A_2 + s_1 I)^{-1}\mathbf{v}_0\},
\end{aligned}$$

which consists of rational functions in the matrices $\{A_1, A_2, A_3\}$. We see that the space contains the subspaces

$$\mathbb{K}_2(A_1, \mathbf{v}_0, \mathbf{s}) = \text{span}\Big\{\mathbf{v}_0, (A_1 + s_1 I)^{-1}\mathbf{v}_0, (A_1 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0\Big\},$$
$$\mathbb{K}_2(A_2, \mathbf{v}_0, \mathbf{s}) = \text{span}\Big\{\mathbf{v}_0, (A_2 + s_1 I)^{-1}\mathbf{v}_0, (A_2 + s_3 I)^{-1}(A_2 + s_1 I)^{-1}\mathbf{v}_0\Big\},$$

associated with A_1 and A_2 and the subspace

$$\mathbb{K}_1(A_3, \mathbf{v}_0, \mathbf{s}) = \text{span}\Big\{\mathbf{v}_0, (A_3 + s_1 I)^{-1}\mathbf{v}_0\Big\},$$

associated with A_3. However, the four vectors

$$(A_2 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0, \quad (A_3 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0,$$
$$(A_1 + s_3 I)^{-1}(A_2 + s_1 I)^{-1}\mathbf{v}_0, \quad (A_3 + s_3 I)^{-1}(A_2 + s_1 I)^{-1}\mathbf{v}_0$$

are not contained in any of these individual subspaces.

REMARK 2.1. *The low-rank projection method described above may be viewed as a generalization of the standard Galerkin method applied to the Sylvester equation*

$$AX + XG = \mathbf{f}\mathbf{g}^\top,$$

where the dimension of G is much smaller than that of A. The analogy with our strategy is that a projection is only performed on the 'large' part of the equation. The smaller right-looking part (involving G) is not reduced but is solved with exactly. See Simoncini [21] for a complete survey.

3. Application to stochastic Galerkin matrix equations. Let us now return to the multi-term matrix equation (1.8) associated with stochastic Galerkin approximations of (1.1),

$$K_0 X G_0^\top + \sum_{r=1}^{m} K_r X G_r^\top = \mathbf{f}_0\, \mathbf{g}_0^\top, \tag{3.1}$$

where the left matrices K_r, $r = 0, 1, \ldots, m$ are symmetric "stiffness" matrices of size $n_x \times n_x$. If the diffusion coefficient has the form (1.2) then we have

$$[K_0]_{i,j} = \int_D a_0 \nabla\varphi_i \cdot \nabla\varphi_j \, d\vec{x}, \qquad [K_r]_{i,j} = \int_D a_r \nabla\varphi_i \cdot \nabla\varphi_j \, d\vec{x}, \qquad r = 1, \ldots, m,$$

where φ_i, φ_j, $i, j = 1, \ldots, n_x$ are finite element basis functions for the approximation space Z_h. Notice that each matrix corresponds to a different term in the expansion (1.2). In addition

$$[\mathbf{f}_0]_i = \int_D f\, \varphi_i \, d\vec{x}, \qquad i = 1, \ldots, n_x.$$

The right matrices G_r, $r = 0, 1, \ldots, m$ are of size $n_\xi \times n_\xi$ and are symmetric. They are given by

$$[G_0]_{s,t} = \langle \psi_s, \psi_t \rangle_\varrho, \qquad [G_r]_{s,t} = \langle y_r \psi_s, \psi_t \rangle_\varrho, \qquad r = 1, \ldots, m,$$

where ψ_s, ψ_t, $s, t = 1, \ldots, n_\xi$ are basis functions for S_p. If we choose these basis functions to be orthonormal with respect to $\langle \cdot, \cdot \rangle_\varrho$ (which is standard) then $G_0 = I$ and $G_1, \ldots, G_m$ all have at most two non-zero entries per row. Moreover, if we are working with uniform random variables on $\Gamma_r = [-1, 1]$, then it can also be shown that the eigenvalues of G_r are contained in $[-1, 1]$ for $r = 1, 2, \ldots, m$. That is, $G_1, \ldots, G_m$ are all indefinite and have spectral bounds that do not depend

on the polynomial degree p or the number of random variables m. See Powell & Elman [17] and Ernst & Ullmann [9] for further discussion. Next, since $\mathbf{g}_0$ is the first column of G_0, we have

$$F = \mathbf{f}_0\,\mathbf{g}_0^T = [\mathbf{f}_0, \mathbf{0}, \ldots, \mathbf{0}] \in \mathbb{R}^{n_x \times n_\xi},$$

so that only the first column of the matrix F on the right-hand side of (3.1) is non-zero.

The matrix equation (3.1) has the same structure as (2.1) but the left matrices are generally indefinite. This is an issue for two reasons. On the one hand, positive definiteness of the left matrices ensures that the rational Krylov subspaces generated in section 2 are well-defined (the coefficient matrices used in the construction of W are non-singular); on the other hand, it ensures that the projected (reduced) matrices $V_k^\top A_r V_k$ in (2.2) are non-singular. The matrix K_0 is always positive definite due to (1.4). However, $K_1, \ldots, K_m$ are indefinite whenever the functions $a_1, \ldots, a_m$ are not strictly positive. We encounter this situation when working with KL expansions (1.3) where $a_r := \sigma\sqrt{\lambda_r}\phi_r$ and ϕ_r is an eigenfunction of a covariance operator. Spectral bounds for $K_r, r = 0, 1, \ldots, m$ are given in [17] and these depend on the finite element mesh parameter h in an unfavourable way. To generate a matrix equation (2.1) that has positive definite left matrices whose spectral intervals do not grow as $h \to 0$, we will need to make two modifications to (3.1).

First, we formally divide the matrix system in (3.1) by K_0. This corresponds to applying the preconditioner $P = I \otimes K_0$ to (1.7). Since K_0 is symmetric and positive definite, we may exploit the Cholesky factorization $K_0 = LL^\top$. Multiplying on the left in (3.1) by L^{-1}, defining $\widehat{X} := L^T X$, and using $G_0 = I$ and $G_r^\top = G_r$ yields the modified matrix equation

$$\widehat{X} + \sum_{r=1}^{m} \widehat{K}_r \widehat{X} G_r = \widehat{\mathbf{f}}_0\,\mathbf{g}_0^\top, \tag{3.2}$$

where $\widehat{\mathbf{f}}_0 := L^{-1}\mathbf{f}_0$ and $\widehat{K}_r := L^{-1}K_r\,L^{-T}$ for $r = 1, \ldots, m$. When K_r is indefinite then $\widehat{K}_r$ is also indefinite. The eigenvalues of $\widehat{K}_r$ coincide with those of $K_0^{-1}K_r$ and bounds for these are established in [17, Lemma 3.4] in the case where the diffusion coefficient is given by (1.3). Generalising this result to (1.2) and assuming for simplicity now that $a_0 > 0$ is a constant, it is straightforward to show that the eigenvalues of $\widehat{K}_r$ lie in the bounded interval $[-\tau_r, \tau_r]$, where

$$\tau_r := a_0^{-1}\|a_r\|_{L^\infty(D)}. \tag{3.3}$$

If $a_r(\vec{x}) > \nu_r > 0$ for all $\vec{x} \in D$, so that K_r is positive definite, then a tighter bound for the eigenvalues of $\widehat{K}_r$ is given by $\left[a_0^{-1}\nu_r, \tau_r\right]$ which is a subset of $[-\tau_r, \tau_r]$. The key point is that these spectral bounds do not depend on the finite element mesh parameter h.

Next, we want to find positive shifts α_r so that $\widehat{K}_r + \alpha_r I$, for $r = 1, \ldots, m$ are positive definite. Using [17, Lemma 3.4], we know that the eigenvalues of $\widehat{K}_r + \alpha_r I$ lie in the interval

$$\mathcal{S}_r := [\alpha_r - \tau_r, \alpha_r + \tau_r], \tag{3.4}$$

(although this bound is not tight if a_r is strictly positive on D). If (1.5) holds then we have $\sum_{r=1}^{m} \tau_r < 1$ and thus $\tau_r < 1$ for each $r = 1, 2, \ldots, m$. Hence, in our problems, choosing $\alpha_r = 1$ for $r = 1, \ldots, m$ suffices.

To incorporate these shifts, we note that (3.2) is equivalent to

$$\widehat{X} + \sum_{r=1}^{m} \left(\widehat{K}_r \widehat{X} G_r \pm \alpha_r \widehat{X} G_r\right) = \widehat{\mathbf{f}}_0\,\mathbf{g}_0^\top,$$

which can be rearranged to give a matrix equation of the desired form (2.1),

$$\widehat{X}\Big(\underbrace{I - \sum_{r=1}^{m} \alpha_r G_r}_{=:B_0}\Big) + \sum_{r=1}^{m} \big(\underbrace{\widehat{K}_r + \alpha_r I}_{=:A_r}\big)\widehat{X}\,\underbrace{G_r}_{=:B_r} = \widehat{\mathbf{f}}_0\,\mathbf{g}_0^\top. \tag{3.5}$$

The equivalent Kronecker formulation of (3.5) is given by $\mathcal{A}\widehat{\mathbf{x}} = \mathbf{b}$ with $\widehat{\mathbf{x}} = \mathtt{vec}(\widehat{X})$ and

$$\mathcal{A} = B_0 \otimes I + \sum_{r=1}^{m} B_r \otimes A_r, \quad \mathbf{b} = \mathbf{g}_0 \otimes \widehat{\mathbf{f}}_0. \tag{3.6}$$

In section 5 the efficiency of the low-rank projection method applied to (3.5) will be compared to that of solving the Kronecker formulation (3.6) using CG.

4. Implementation details. The complete low-rank projection algorithm consists of a preprocessing or set-up phase followed by an iterative solution phase. In the preprocessing phase, the matrices K_r and G_r are transformed into A_r and B_r and the vector $\mathbf{f}_0$ is transformed into $\widehat{\mathbf{f}}_0$, as outlined in the previous section. More precisely, the matrices A_r and the vector $\widehat{\mathbf{f}}_0$ are not explicitly assembled. It is enough to provide the vector $\mathbf{f}_0$, the matrices K_r, the Cholesky factor L of K_0 and the shifts $\alpha_1, \ldots, \alpha_m$. The parameters $\mathbf{s} = [s_1, s_2, \ldots]$ are also selected. Strategies for choosing them will be discussed in section 4.2. We stress that the preprocessing does not entail any approximation, hence the transformed matrix equation (3.5) is mathematically equivalent to the original system (3.1). The solution phase is outlined in general terms in Algorithm **4.1**. A careful implementation is essential if it is to be efficient in practice and we address this next.

Algorithm 4.1: Multi-term Reduced Basis Solver (MultiRB)

INPUT: the matrices $\{A_r\}_{r=1,2,\ldots,m}$ and $\{B_r\}_{r=0,1,\ldots,m}$, the vectors $\widehat{\mathbf{f}}_0$ and $\mathbf{g}_0$, the parameters $\mathbf{s} = [s_1, s_2, \ldots]$, the truncation threshold β and the maximum number of iterations $j_{\max}$.

0. $\mathbf{v} = \widehat{\mathbf{f}}_0/\|\widehat{\mathbf{f}}_0\|$, $V_0 = \mathbf{v}$, $j = 1$,
1. While **not converged** and $j < j_{\max}$
  1.1 Compute $W = [(A_1 + s_j I)^{-1}\mathbf{v}, \ldots, (A_m + s_j I)^{-1}\mathbf{v}]$
  1.2 Truncate $\widehat{W} := \mathrm{trunc}(W) \in \mathbb{R}^{n_x \times \ell_j}$ and expand $V_j := \mathrm{orth}([V_{j-1}, \widehat{W}])$
  1.3 Update projected matrices $\overline{A}_r = V_j^\top A_r V_j$, $r = 1, \ldots, m$, $\overline{\mathbf{f}} = V_j^\top \widehat{\mathbf{f}}_0$
  1.4 Approximately solve

$$Y_j B_0 + \overline{A}_1 Y_j B_1 + \ldots + \overline{A}_m Y_j B_m = \overline{\mathbf{f}}\mathbf{g}_0^\top$$

  for Y_j and check convergence
  1.5 If stopping criterion is satisfied set $k = j$ and stop; else
  1.6 Set $\mathbf{v} = V_j \mathbf{e}_{j+1}$ (take the next column of V_j to expand the space)
  1.7 Set $j = j + 1$
2. Optional post-processing step: Compute rank-revealing factorization $Y_k \approx Y^{(1)}(Y^{(2)})^\top$.

OUTPUT: solution factors $X^{(1)} = V_k$ and $X^{(2)} = Y_k$ (or $X^{(1)} = V_k Y^{(1)}$ and $X^{(2)} = Y^{(2)}$ if step **2** is performed).

4.1. Solution phase. The most time-consuming parts of the solution phase are (i) step **1.1**, which entails m system solves with sparse matrices of dimension $n_x \times n_x$, (ii) step **1.2**, which performs the orthogonalization of the current reduced basis, and (iii) step **1.4**, where the solution of the projected problem is computed and its accuracy is assessed. In step **1.2**, the orthogonality of the approximation space basis is maintained by enforcing orthogonality of the newly added vectors by means of the Gram-Schmidt process. The new basis vectors are always taken to be the singular vectors corresponding to the dominant singular values of the latest computed matrix

W. Note that steps **1.1** and **1.2** must be performed at every iteration, whereas step **1.4** can be performed every few iterations, if appropriate. We now elaborate on steps **1.1** and **1.4**.

In step **1.1**, the shifted systems that must be solved share the same right-hand side vector $\mathbf{v}$. In principle, they could be solved by a sparse direct solver. Indeed, if s_j does not change with the iteration, and if memory is not a concern, then one could save sparse factorizations of each of the matrices $(A_r + s_j I)$ for $r = 1, \ldots, m$, and reuse them in each iteration. For large n_x and m, however, storing all the factorizations becomes prohibitive. Moreover, the direct solution of m large systems may well be expensive. An effective alternative is the use of iterative methods, and in particular, the simultaneous iterative solution of all m sparse systems using CG. The iterations for each system are performed at the same time: multiple `daxpys` are used so that the matrix-vector products are carried out simultaneously, making the access to memory allocations more efficient. It goes without saying that if iterative methods are used to solve the systems in step **1.1** then we do not need to solve them to machine precision. A loose tolerance will be employed. A crucial point is that the matrices $A_r = L^{-1} K_r L^{-\top} + \alpha_r I$, will never be explicitly assembled. If a system of the form $(A_r + s_j I)\mathbf{y} = \mathbf{v}$ is solved by a direct method, then this corresponds to solving

$$(K_r + (\alpha_r + s_j)K_0)\widetilde{\mathbf{y}} = L\mathbf{v}, \qquad \widetilde{\mathbf{y}} = L^{-\top}\mathbf{y}, \tag{4.1}$$

where K_r and K_0 have the same sparsity pattern. On the other hand, if the same system is solved by an iterative method, then matrix-vector products of the type $(A_r + s_j I)\mathbf{w}$ can be performed as $(A_r + s_j I)\mathbf{w} = L^{-1}((K_r + (\alpha_r + s_j)K_0)(L^{-\top}\mathbf{w}))$. In either case operations with the sum $K_r + (\alpha_r + s_j)K_0$ are performed so $\widehat{K}_r$ is not formed explicitly.

In step **1.4**, once the deterministic part of the problem has been reduced, the projected matrix equation needs to be solved and the quality of the current approximation assessed. We consider the Kronecker formulation and apply CG once more. As is typical in these circumstances, a matrix-oriented version of CG is employed to exploit the presence of `blas2` and `blas3` operations in all the required matrix operations. We also remark that to speed up convergence, the initial guess for this second CG iteration (step **1.4**) is taken to be the solution Y_{j-1} obtained at the previous outer iteration. To ensure the matrix has the correct dimension, we pad Y_{j-1} with as many zero rows as basis vectors are added to V_j at step j. Summarising, the algorithm contains two inner iterations: the first one expands the approximation space by solving m linear systems with the same right-hand side, but with different coefficient matrices (step **1.1**). The second one determines the approximate solution to the reduced problem (step **1.4**). While for the former a fixed loose stopping tolerance can be used, for the latter an increasingly more accurate solution is sought as the outer iteration converges.

Monitoring convergence of the outer iteration is not straightforward. Since we assume that n_x and n_ξ are so large that vectors of dimension $n_x \cdot n_\xi$ should not be constructed, the residual matrix associated with the current approximation cannot be explicitly generated. Instead, at the jth iteration we evaluate the relative difference

$$\frac{\|X_j - X_{j-1}\|_F}{\|X_j\|_F} = \frac{\|Y_j - [Y_{j-1}; 0]\|_F}{\|Y_j\|_F}, \tag{4.2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. For a matrix X, this norm is equivalent to computing the Euclidean norm of $\mathtt{vec}(X)$. The outer iteration is terminated once the relative difference in the approximation falls below a user-specified tolerance `tol_outer`. The stopping tolerance for the inner iteration in step **1.4** of Algorithm **4.1** can be tuned with respect to the outer tolerance. In section 5 we use a dynamically decreasing tolerance `tol_inner` for CG on the projected problems.

After k iterations, when the outer iteration is judged to have converged, the approximate solution X_k can be returned in factored form. That is, the matrices V_k and Y_k can be returned. However, the reduced solution matrix $Y_k \in \mathbb{R}^{n_k \times n_\xi}$ is not necessarily of full rank. Specifically,

many of the singular values of Y_k may fall below machine precision, or some other tolerance that is below the final tolerance requested for convergence. This fact can be exploited at convergence (if desired) in step **2**. Indeed, assuming that $Y_k = Y^{(1)}(Y^{(2)})^\top$ with $Y^{(1)}, Y^{(2)}$ of rank t, then the approximate solution to the original problem is also of rank t and is given by $X_k = X^{(1)}(X^{(2)})^\top$ with $X^{(1)} := V_k Y^{(1)}$ and $X^{(2)} := Y^{(2)}$. In practice, even if Y_k is full rank it is interesting to approximate Y_k by a low rank matrix $Y^{(1)}(Y^{(2)})^\top$ that retains the largest singular values of Y_k without affecting the overall approximation. We describe how to do this next. For ease of exposition we assume that $n_k \leq n_\xi$. Let $Y_k = U\Theta Q^\top$ be the economy-size singular value decomposition of Y_k, that is $Y_k = \sum_{r=1}^{n_k} \theta_r \mathbf{u}_r \mathbf{q}_r^\top$ where the values θ_r are sorted decreasingly. Suppose that $\theta_{\hat{r}+1}$ is the first singular value satisfying

$$\theta_{\hat{r}+1} \leq \theta_1 \frac{\texttt{tol_outer}}{\sqrt{n_\xi}}, \tag{4.3}$$

and let $U_{\hat{r}} = [\mathbf{u}_1, \ldots, \mathbf{u}_{\hat{r}}], \Theta_{\hat{r}} = \mathrm{diag}(\theta_1, \ldots, \theta_{\hat{r}})$ and $Q_{\hat{r}} = [\mathbf{q}_1, \ldots, \mathbf{q}_{\hat{r}}]$. Then Y_k can be approximated by $Y^{(1)}(Y^{(2)})^\top$ with $Y^{(1)} = U_{\hat{r}}\Theta_{\hat{r}}$ and $Y^{(2)} = Q_{\hat{r}}$, where the error is given by

$$\|Y_k - Y^{(1)}(Y^{(2)})^\top\|_2 = \theta_{\hat{r}+1}$$

(see Golub & van Loan [10]). Note that this does not affect the final approximation in a significant manner. Indeed the new approximation satisfies the same stopping criterion with a value of the tolerance equal to[2] $2\cdot$`tol_outer`. To show this, let us write $Y_k = Y^{(1)}(Y^{(2)})^\top + E$, with $\|E\|_2 = \theta_{\hat{r}+1}$ where $\theta_{\hat{r}+1}$ satisfies (4.3) and recall that $\|Y_k\|_2 = \|Y^{(1)}(Y^{(2)})^\top\|_2 = \theta_1$, $\|Y_k\|_F = (\theta_1^2 + \ldots + \theta_{n_k}^2)^{1/2} \geq \|Y^{(1)}(Y^{(2)})^\top\|_F$ and $\|E\|_F \leq \sqrt{n_\xi}\|E\|_2$. In this case we have that

$$\begin{aligned}
\frac{\|Y^{(1)}(Y^{(2)})^\top - [Y_{k-1}; 0]\|_F}{\|Y^{(1)}(Y^{(2)})^\top\|_F} &= \frac{\|(Y_k - E) - [Y_{k-1}; 0]\|_F}{\|Y^{(1)}(Y^{(2)})^\top\|_F} \\
&\leq \frac{\|Y_k - [Y_{k-1}; 0]\|_F}{\|Y^{(1)}(Y^{(2)})^\top\|_F} + \frac{\|E\|_F}{\|Y^{(1)}(Y^{(2)})^\top\|_F} \\
&\leq \frac{\|Y_k - [Y_{k-1}; 0]\|_F}{\|Y_k\|_F} + \frac{\sqrt{n_\xi}\|E\|_2}{\|Y^{(1)}(Y^{(2)})^\top\|_F} \\
&\leq \texttt{tol_outer} + \frac{\theta_1 \texttt{tol_outer}}{\|Y^{(1)}(Y^{(2)})^\top\|_2} = 2 \cdot \texttt{tol_outer}.
\end{aligned}$$

This computation may be expensive if n_k and n_ξ are both large so it may be performed as an optional post-processing step. It is not an essential part of the solution phase. However, the cost of performing step **2** is included in our experiments in section 5.

We also remark that the product $X^{(1)}(X^{(2)})^\top$ does not have to be performed explicitly. Single columns of the solution matrix can be computed as $X^{(1)}((X^{(2)})^\top \mathbf{e}_i)$, $i = 1, \ldots, n_\xi$, where $\mathbf{e}_i$ denotes the ith column of the identity matrix. In particular, if we assume that the first basis function for S_p is $\psi_1 = 1$, then the first column of the solution matrix provides the coefficients that determine the mean of the stochastic Galerkin approximation.

Finally, we note that the dimension of the approximation space constructed in Algorithm **4.1** is n_k with $n_k \leq 1 + m \cdot k$, since at most m new vectors are added to the space at each iteration. Hence, the more random variables we need to use in (1.2) to represent the diffusion coefficient, the more basis vectors may be added to the approximation space at each iteration. If memory limitations are likely to be exceeded then, instead of iterating until the norm of the relative difference in the solution falls below `tol_outer`, the user may wish to terminate the algorithm after a maximum number $j_{\max}$ of outer iterations—in this case the memory requirements are bounded by $(n_x + n_\xi) \cdot m \cdot j_{\max}$.

[2] If a final accuracy of `tol_outer` is required then the threshold in the stopping test could be set to `tol_outer`/2.

4.2. Selection of the parameters. It remains to choose the parameters s_j used in step **1.1** of Algorithm **4.1**. Recall that in section 2 we stipulated that $A_1, \ldots, A_m$ should all have eigenvalues contained in a small interval $\mathcal{S} \subset \mathbb{R}^+$. If this is true, then using a single parameter s_j at the jth iteration to construct all the columns of W is justified. Two strategies for choosing s_j are outlined below. To implement them, we need to estimate $\mathcal{S}$ cheaply and so we explain how to do this first.

For stochastic Galerkin matrix equations we have $A_r = \widehat{K}_r + \alpha_r I$, $r = 1, \ldots, m$ where the shift α_r is chosen to make A_r positive definite. In section 3 we explained that $\alpha_r = 1$ suffices and a bound for the spectral interval of A_r is then given by $\mathcal{S}_r = [1-\tau_r, 1+\tau_r]$ where $\tau_r < 1$ is defined in (3.3). Since the series (1.2) is assumed to converge as $m \to \infty$, we must also have $\tau_r \to 0$ as $r \to \infty$. Hence, the interval $\mathcal{S}_r$ contracts to $\alpha_* = 1$ as $r \to \infty$ and $\mathcal{S} := \mathcal{S}_1 = [1-\tau_1, 1+\tau_1]$ contains the eigenvalues of each of $A_1, \ldots, A_m$. The end points of this interval can be estimated once the diffusion coefficient (1.2) has been chosen. No eigenvalues have to be computed.

It is often the case, especially when working with KL expansions (1.3), that the first eigenfunction ϕ_1 is a strictly positive function, so that, as explained in section 3, the matrix $\widehat{K}_1$ is positive definite and $\mathcal{S}_1$ is not a sharp bound for the eigenvalues of $A_1 = \widehat{K}_1 + \alpha_1 I$. However, for $r \geq 2$, the matrices $\widehat{K}_r$ are indefinite and do have spectral intervals centred around the origin (because the associated eigenfunctions are symmetric). For those matrices, the bound $\mathcal{S}_r$ is sharper. In this case, the following alternative strategy for choosing the shifts may be effective. Let $\lambda_{1,\min}, \lambda_{1,\max}$ be the extreme eigenvalues of $\widehat{K}_1$ and choose

$$\alpha_1 = 1 - (\lambda_{1,\min} + \lambda_{1,\max})/2, \qquad \alpha_2 = \alpha_3 = \ldots = \alpha_m = 1. \tag{4.4}$$

This ensures that the eigenvalues of $A_r = \widehat{K}_r + \alpha_r I$ for each $r = 1, \ldots, m$ are centred around the unit value. The intervals $\mathcal{S}_r$ for $r \geq 2$ are now nested and collapse to the unit value as $r \to \infty$. Estimates for $\lambda_{1,\min}$ and $\lambda_{1,\max}$ can be obtained cheaply by computing the extreme eigenvalues of the matrix pencil (K_1, K_0), where K_1 and K_0 are assembled on a coarse finite element mesh (since the eigenvalues do not depend on h). After applying the shifts in (4.4), the interval $\mathcal{S} := \mathcal{S}_2$ or $\mathcal{S} = [1 - \lambda_{2,\min}, 1 + \lambda_{2,\max}]$, where $\lambda_{2,\min}$ and $\lambda_{2,\max}$ are estimates for the extreme eigenvalues of the matrix pencil (K_2, K_0), typically contains the eigenvalues of each of $A_1, \ldots, A_m$. With an estimate of $\mathcal{S}$ in hand, we can now address how to choose the parameters s_j.

4.2.1. Multiple parameter strategy. One idea is to choose s_j using standard parameter estimation ideas using rational approximation. For example, see [21]. Assume for simplicity that $m = 1$, so that the problem to be solved is the Sylvester equation

$$XB_0 + A_1 X B_1 = \mathbf{f}_0 \mathbf{g}_0^\top. \tag{4.5}$$

Assume that B_0 is nonsingular and that we can write the full eigendecomposition $B_1 B_0^{-1} Q = Q\Lambda$, where $\Lambda = \mathrm{diag}(\Lambda_1, 0)$ and Λ_1 is diagonal and nonsingular. Then (4.5) can be rewritten as

$$XQ + A_1 XQ\Lambda = \mathbf{f}_0 \mathbf{g}_0^\top B_0^{-1} Q.$$

Setting $Q = [Q_1, Q_2]$ with the same partitioning as for Λ, we see that the first block satisfies

$$XQ_1 + A_1 XQ_1\Lambda_1 = \mathbf{f}_0 \mathbf{g}_0^\top B_0^{-1} Q_1.$$

Moreover, the jth column $\mathbf{z}_j$ of XQ_1 is the solution to a shifted system,

$$\left(\frac{1}{\lambda_j} I + A_1\right) \mathbf{z}_j = \mathbf{d}_j,$$

where $\mathbf{d}_j$ is the jth column of the right-hand side matrix $\frac{1}{\lambda_j}\mathbf{f}_0\mathbf{g}_0^\top B_0^{-1}Q_1$. We now follow arguments for estimating $\mathbf{z}_j$ used in rational Galerkin methods. For example, see Druskin & Simoncini [6] and Druskin et al. [5]. An approximation of $\mathbf{z}_j$ is determined[3] as $\mathbf{z}_{j,k} \in \overline{\mathbb{K}}_k(A_1, \mathbf{v}_0, \mathbf{s})$ where

$$\overline{\mathbb{K}}_k(A_1, \mathbf{v}_0, \mathbf{s}) = \operatorname{span}\left\{ (A_1 + s_1 I)^{-1}\mathbf{v}_0, (A_1 + s_2 I)^{-1}(A_1 + s_1 I)^{-1}\mathbf{v}_0, \ldots, \prod_{j=1}^{k}(A_1 + s_j I)^{-1}\mathbf{v}_0 \right\},$$

and $\mathbf{v}_0$ is the normalized version of $\mathbf{d}_j$. The vector of parameters $\mathbf{s}$ is still to be selected. Let $\overline{A}_1 = V_k^\top A_1 V_k$, where the columns of V_k are an orthonormal basis of $\overline{\mathbb{K}}_k(A_1, \mathbf{v}_0, \mathbf{s})$. If a Galerkin condition is imposed to determine $\mathbf{z}_{j,k}$, then $\mathbf{z}_{j,k} = V_k(\frac{1}{\lambda_j}I + \overline{A}_1)^{-1}V_k^\top \mathbf{d}_j$ and it can be shown [5] that the residual satisfies

$$\mathbf{d}_j - \left(\frac{1}{\lambda_j}I + A_1\right)\mathbf{z}_{j,k} = \frac{r_k(A_1)\mathbf{d}_j}{r_k(-1/\lambda_j)}, \qquad r_k(x) = \prod_{j=1}^{k}\frac{x - \theta_j}{x + s_j},$$

where θ_j are the eigenvalues of $\overline{A}_1$. That is, the residual is characterised by the rational function r_k. Note that the values θ_j are contained in the spectral interval of $\overline{A}_1$ and hence in that of A_1. We now have

$$\min_{s_1,\ldots,s_k}\left\|\mathbf{d}_j - \left(\frac{1}{\lambda_j}I + A_1\right)\mathbf{z}_{j,k}\right\| = \min_{s_1,\ldots,s_k}\frac{1}{|r_k(-1/\lambda_j)|}\|r_k(A_1)\mathbf{d}_j\|,$$

where $\|\cdot\|$ denotes the Euclidean norm. We are not going to use the quantity $1/|r_k(-1/\lambda_j)|$. However, it is worth noticing that $|r_k(-1/\lambda_j)|$ should not be too small. That is, the eigenvalues of $\overline{A}_1$ (and of A_1) should not be too close to $-1/\lambda_j$, and this fact confirms the solvability condition of the initial Sylvester equation.

We thus seek parameters $s_1, \ldots, s_k$ that minimize $\|r_k(A_1)\mathbf{d}_j\|$. The procedure for this is well known in the context of solving Lyapunov equations using the ADI method. A closely related optimization problem is obtained by replacing θ_j with s_j in the numerator of r_k, giving the classical Zolotarev minimax problem

$$\min_{s_1,\ldots,s_k}\max_{x\in\operatorname{spec}(A_1)}\left|\prod_{j=1}^{k}\frac{x - s_j}{x + s_j}\right|.$$

If the Galerkin method is used with the values of s_j obtained by solving this problem, then the rational function r_k is the same as the function being minimized in the Zolotarev problem. That is, $\theta_j = s_j$ holds. This crucial property makes the Galerkin condition with the rational Krylov space and the ADI method equivalent; see Simoncini [21]. To make the problem tractable, however, the minimax problem above is usually replaced with

$$\min_{s_1,\ldots,s_k}\max_{x\in\mathcal{S}}\left|\prod_{j=1}^{k}\frac{x - s_j}{x + s_j}\right|,$$

where $\mathcal{S}$ is an interval containing $\operatorname{spec}(A_1)$. The set of parameters solving this problem can be computed using elliptic functions. Their derivation is discussed, e.g. by Lu & Wachspress [14], and their computation by a few lines of MATLAB code is described by Sabino [19, p.43]. Other

[3] To simplify the derivation, $\mathbf{v}_0$ is not included in the approximation space.

selections associated with quasi-optimal rational approximation of matrix functions could be used instead; see for example, Beckermann & Reichel [3] or Güttel [11].

In our setting (with $m > 1$), given an estimate for the interval $\mathcal{S}$ that contains the eigenvalues of each of the matrices $A_1, \ldots, A_m$, we may compute say 5 or 10 nodes $s_j \in \mathcal{S}$ using the method described above which we cycle through as the iteration proceeds. The theoretical arguments cannot be easily extended to the case $m > 1$ but nevertheless, the strategy seems to work well in practice. See section 5. A more thorough justification and a convergence analysis will be the subject of future work.

4.2.2. Parameter-free strategy. Looking at (4.1) we can see that both α_r and s_j act as shifts, although the rationale presented for choosing them is different. In the case of stochastic Galerkin matrix equations, the shifts α_r and the parameters s_j can often be combined, leading to a simpler version of Algorithm **4.1**. We now develop this idea.

Instead of selecting say 5 or 10 parameters $s_j \in \mathcal{S}$ as described above, a simpler strategy is to choose $s_j = s_*$ for each $j = 1, 2, \ldots$. That is, choose a single point in $\mathcal{S}$ and use the same parameter in each iteration. For Lyapunov equations, the choice $s_* = \sqrt{a \cdot b}$ was suggested in [14] where the estimated spectral interval is given by $\mathcal{S} = [a, b]$. In our setting, if we choose the shifts to be $\alpha_r = 1$ for $r = 1, \ldots, m$ then each of the intervals $\mathcal{S}_r$ will be centred about $\alpha_* = 1$, and a natural choice is $s_* = 1$ (the mid-point). $\mathcal{S}$ does not even need to be estimated. In (4.1), we then have $\alpha_r + s_j = 2$ for $r = 1, \ldots, m$, in each iteration. A mathematically equivalent version of Algorithm **4.1** is then obtained by setting the constants α_r to zero and choosing $s_* = (\alpha_r + s_j) = 2$. Note that with this combination, the preprocessing phase is essentially redundant. Only the Cholesky factor of K_0 needs to be precomputed. Once the truncation threshold β is fixed, the MultiRB solver is completely parameter-free!

Whether we use a multi-parameter or parameter-free strategy, it is worth noting that if the spectrum of each matrix A_r, $r = 1, \ldots, m$, is independent of the mesh parameter h, as in our setting, then the computed values of s_j do not depend on h. Since the spectrum of each matrix $A_r + s_j I$ is then independent of the mesh parameter, we anticipate that the dimension of the approximation space generated by our algorithm will be insensitive to h. This feature will be confirmed by numerical experiments next.

5. Numerical experiments. In this final section, we consider two test problems of the form (1.1) that are implemented in the MATLAB software package S-IFISS [20]. The iterative solver implemented in S-IFISS tackles the Kronecker formulation of the original linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$ with $\mathcal{A}$ in (1.7), not the matrix equation formulation considered here. Example 5.1 corresponds to test problem 5 in S-IFISS and has a diffusion coefficient of the form (1.2) taken from Eigel et al. [7]. Example 5.2 corresponds to S-IFISS test problem 2 and in this case, the diffusion coefficient is a truncated KL expansion (1.3) with relatively slowly decaying eigenvalues λ_r. In both examples, we choose the stochastic Galerkin approximation space Z_h to be the set of piecewise bilinear functions defined on uniform subdivisions of a square domain D. We perform experiments with two specific finite element grids with $n_x = 16,129$ and $n_x = 65,025$ degrees of freedom. The second of these (grid level 8) is a uniform refinement of the first (grid level 7). We choose the space S_p to be the set of Legendre polynomials of total degree p in $y_1, \ldots, y_m$ on $\Gamma = [-1, 1]^m$. We vary p and choose m appropriately for the problem at hand. Recall that n_ξ is given by (1.6). All experiments were performed in MATLAB 7.10.0 on a DELL Precision T7500 desktop computer with 12 cores and a total of 48 GB RAM.

To solve the test problems we apply (i) the new MultiRB solver (Algorithm **4.1**) to the matrix equation (3.5) and (ii) CG to the equivalent Kronecker formulation (3.6). Notice that both methods essentially use the mean-based preconditioner $P = I \otimes K_0$. We compare the efficiency of the two approaches and investigate how the dimension n_k of the approximation space constructed

by the MultiRB solver behaves when we increase the number of spatial degrees of freedom n_x, the polynomial degree p and the number of random variables m. In the implementation of MultiRB, we apply CG in step **1.1** until the relative residual error for all m systems is below 10^{-4}. The stopping tolerance for the outer iteration is chosen to be $\texttt{tol_outer} = 10^{-5}$. The projected problem in step **1.4** is solved iteratively with CG at each iteration and we use the dynamically decreasing inner tolerance $\texttt{tol_inner} = 10^{-3} \cdot \texttt{tol_outer}$. Unless otherwise stated, we fix the truncation threshold to be $\beta = 99$.

When we apply CG to the Kronecker formulation, we exploit the structure and use (1.9) when performing matrix-vector products. Variants of CG for linear systems in Kronecker form are written with matrix-matrix operations, and typically include truncation strategies that maintain iterates of low rank as the iteration proceeds, see for example [16]. We do not adopt such a truncation strategy in our comparison because doing so would affect the optimality properties of the CG solver and we want our reference method to have a reliable convergence rate. To make fair comparisons with the reduced basis solver, the stopping criterion for CG applied to (3.6) is also based on the relative difference in the approximate solution. The memory requirements of CG applied to (3.6) are 4 vectors of length $n_x \cdot n_\xi$ each: the current approximation, the direction, the residual, and an extra vector for storing the matrix-vector multiplication; the total memory requirements are thus $4 \cdot n_x \cdot n_\xi$. In contrast, the memory requirements for the reduced basis solver are of order $(n_x + n_\xi) \cdot n_k$. We stress that CG requires full length vectors of size $n_x \cdot n_\xi$ whereas the new solver does not.

EXAMPLE 5.1. Consider the square domain $D = [0,1] \times [0,1]$ and define the diffusion coefficient to be (1.2) with $a_0 = 1$ and

$$a_r(\vec{x}) = \gamma_r \cos(2\pi\beta_1(r)x_1) \cos(2\pi\beta_2(r)x_2)$$

where $\gamma_r := 0.832\, r^{-4}$ and $\beta_1(r) := r - s(r)\,(s(r+1))/2$, $\beta_2(r) := s(r) - \beta_1(r)$ with $s(r) := \lfloor -1/2 + \sqrt{1/4 + 2r} \rfloor$. We note that the coefficients γ_r, which determine the weightings of the terms a_r, decay rapidly as $r \to \infty$. If we choose $m = 5$ then we retain all terms in the expansion (1.2) with coefficients $\gamma_r \geq 10^{-3}$. However, if we choose $m = 9$ and $m = 16$ then we retain all terms $\gamma_r \geq 10^{-4}$ and 10^{-5}, respectively.

			Multi-parameter				Parameter-free			
m	p	n_ξ	k	n_k	rank	time	k	n_k	rank	time
5	2	21	23	84	19	5.96e0	16	66	19	3.67e0
	3	56	26	97	29	7.02e0	19	77	28	4.76e0
	4	126	28	104	37	7.80e0	19	77	36	4.63e0
	5	252	28	104	43	8.24e0	23	94	42	6.29e0
9	2	55	16	74	26	6.96e0	14	79	26	5.72e0
	3	220	16	74	34	7.21e0	16	94	34	6.96e0
	4	715	16	74	40	7.84e0	17	100	42	8.19e0
	5	2,002	22	105	47	1.67e1	18	102	47	1.27e1
16	2	153	15	87	32	1.19e1	12	82	32	9.94e0
	3	969	16	89	40	1.40e1	14	106	41	1.21e1
	4	4,845	16	89	45	2.53e1	15	117	46	2.82e1
	5	20,349	19	105	49	1.71e2	15	117	51	1.52e2

TABLE 5.1
Results obtained with the MultiRB solver for Example 5.1 and grid level 7: multi-parameter vs parameter-free.

First, we fix the grid level to be 7 and compare the multiple-parameter and parameter-free implementations of the reduced basis solver outlined in sections 4.2.1 and 4.2.2. For the latter,

we set $s_* = 2$. For the former, we choose the shifts as in (4.4) and then, on the estimated interval $\mathcal{S} = [\alpha_1 + \lambda_{1,\min}, \alpha_1 + \lambda_{1,\max}] \approx [0.1695, 1.8305]$, we compute 5 nodes $[s_1, s_2, s_3, s_4, s_5]$ using the MATLAB code from [19, p.43]. Note that α_1 is close to 1 in this example since $\lambda_{1,\min} \approx -\lambda_{1,\max}$. The matrix $\widehat{K}_1$ is indefinite. Results are presented in Table 5.1. We record the number of outer iterations k required by the MultiRB solver, the dimension n_k of the approximation space generated (the number of columns of V_k) and the rank of the final approximation X_k. We also record the CPU time in seconds for the solution phase. This includes the time taken to compute the truncated singular value decomposition of Y_k. For $m = 9$ and $m = 16$ we see that the approximation space generated by the multi-parameter version is smaller than that generated by the parameter-free version. However, timings are similar for both implementations.

m	p	n_ξ	k	inner	n_k	rank	time	CG time (its)
5	2	21	16	12.9	66	19	3.67e0	1.42e0 (10)
	3	56	19	15.6	77	28	4.76e0	3.85e0 (12)
	4	126	19	16.7	77	36	4.63e0	9.32e0 (14)
	5	252	23	18.9	94	42	6.29e0	1.77e1 (14)
9	2	55	14	13.3	79	26	5.72e0	3.68e0 (10)
	3	220	16	15.6	94	34	6.96e0	1.59e1 (12)
	4	715	17	17.1	100	42	8.19e0	6.20e1 (14)
	5	2,002	18	18.4	102	47	1.27e1	1.71e2 (14)
16	2	153	12	12.7	82	32	8.94e0	1.26e1 (10)
	3	969	14	15.1	106	41	1.21e1	1.64e1 (12)
	4	4,845	15	16.7	117	46	2.82e1	5.44e2 (14)
	5	20,349	15	17.5	117	51	1.52e2	3.31e3 (14)

TABLE 5.2
Results for Example 5.1 and grid level 7: MultiRB solver vs standard CG.

m	p	n_ξ	k	inner	n_k	rank	time	CG time (its)
5	2	21	16	12.9	66	19	2.14e1	8.13e0 (10)
	3	56	19	15.6	77	28	2.65e1	2.17e1 (12)
	4	126	19	16.7	77	36	2.52e1	5.31e1 (14)
	5	252	23	18.9	94	42	3.23e1	1.03e2 (14)
9	2	55	14	13.3	79	26	3.27e1	2.03e1 (10)
	3	220	16	15.6	94	34	3.86e1	9.11e1 (12)
	4	715	17	17.1	100	42	4.22e1	3.35e2 (14)
	5	2,002	18	18.3	102	47	4.91e1	9.35e2 (14)
16	2	153	12	12.7	82	32	5.04e1	6.76e1 (10)
	3	969	14	15.1	106	41	6.19e1	4.90e2 (12)
	4	4,845	15	16.7	117	46	8.47e1	2.81e3 (14)
	5	20,349	15	17.5	117	51	2.15e2	Out of Memory

TABLE 5.3
Results for Example 5.1 and grid level 8: MultiRB solver vs standard CG.

Adopting the parameter-free implementation for simplicity, we now compare the efficiency of using the MultiRB method to solve the matrix equation (3.5) and using CG to solve (3.6). Results obtained for grid levels 7 and 8 are presented in Tables 5.2 and 5.3 respectively. In column five, we also now record the average number of inner CG iterations required to solve the projected matrix equations. The CPU time taken to solve the Kronecker formulation using CG is reported in the

final column and the associated number of CG iterations is given in parentheses for reference.

Comparing the results in Tables 5.2 and 5.3 we see that the dimension n_k of the approximation space built by the MultiRB solver is independent of h, as predicted. The rank of the final approximation is also observed to be independent of h. When n_ξ is small we see that there is nothing to be gained by using the low-rank solver. Using CG on the Kronecker formulation is quicker. However, when the choice of m and p is such that the dimension n_ξ is large (typically $n_\xi > 100$) using the reduced basis solver on the matrix formulation is much quicker. Although we do not report them here, we note that the results obtained with the multi-parameter implementation of the low-rank solver exhibit completely analogous behaviour. Of course, timings are not the only consideration. In the largest experiment, where $n_\xi = 20,349$ and the grid level is 8, the Kronecker formulation consists of 1.3 billion equations! We run out of memory when we apply CG to (3.6). However, since the reduced basis solver never forms vectors of length $n_x \cdot n_\xi$, it is able to make more efficient use of the available memory. As mentioned in section 4.2, when n_ξ and n_x are large, computing the truncated singular value decomposition of Y_k is costly. In Table 5.3, in the case $m = 16$ and $p = 5$, we note that 35% of the total reported solution time for the reduced basis solver was spent on this post-processing task. Recall that this step is not part of the actual solution phase and can be turned off by the user.

We observe that both the dimension n_k of the approximation space generated by the reduced basis solver and the rank of the final solution matrix increase slightly with the number of random variables m and the polynomial degree p. Notice also that n_k is larger than the rank in all cases. This suggests that the reduced basis built by the new method is not optimal. One possible reason for this is that since the terms in (1.2) decay so rapidly in this example, the spectral intervals of the matrices A_r also decay very rapidly. The matrix A_1 has a noticeably larger spectral interval than $A_2, \ldots, A_m$. Choosing a parameter s_j to form $(A_1 + s_j I)$ that is distinct from the parameter s_j used to form $(A_r + s_j I)$, for $r \geq 2$ in the construction of W in step **1.1** of the MultiRB method may then be more effective. The current implementation is more general however and the savings in both time and memory usage compared to using standard CG are self-evident.

In the results above we used the default value $\beta = 99$ for the truncation threshold. The results presented in Table 5.4 illustrate the effect of relaxing the threshold to $\beta = 96$ so that fewer vectors are retained at each step. Comparing the results in Table 5.4 with those in Table 5.3 for the case $m = 9$ we see although the dimension n_k is now a little smaller, more outer iterations are required and the timings have increased.

m	p	n_ξ	k	inner	n_k	rank	time
9	2	55	23	13.1	82	26	4.94e1
	3	220	24	14.8	83	34	5.10e1
	4	715	27	16.8	93	42	5.97e1
	5	2,002	27	17.6	95	47	6.44e1

TABLE 5.4
Results for Example 5.1 and grid level 8: MultiRB solver with $\beta = 96$.

EXAMPLE 5.2. Consider the square domain $D = [-1, 1] \times [-1, 1]$ and let the diffusion coefficient take the form (1.3) with constant mean $\mu = 1$. Let $\xi_r \sim U(-\sqrt{3}, \sqrt{3})$ and consider the covariance function

$$C(\vec{x}_1, \vec{x}_2) = \sigma^2 \exp\left(-\frac{\|\vec{x}_1 - \vec{x}_2\|_1}{\ell}\right),$$

where ℓ denotes the correlation length. The eigenvalues and eigenfunctions of the associated covariance operator can be computed analytically (e.g., see [13, Chapter 7]). The eigenvalues

λ_r determine the weightings of the individual terms in the expansion (1.3). Note that these decay much more slowly than the coefficients γ_r in Example 5.1. If we set $m = \infty$ in (1.3) then $\int_D \mathrm{Var}(a(\vec{x},\omega))\,d\vec{x} = \sigma^2 \mathrm{Leb}(D) = 4\sigma^2$. If we truncate the series after m terms, then the integral of the variance of the approximated coefficient is $\sigma^2 \sum_{r=1}^{m} \lambda_r$. We can use this fact to determine how to choose m. In particular, setting $\ell = 2$ and then retaining $m = 8, 12$, and 20 terms means that we incorporate 87%, 89% and 93%, respectively, of the integral of the variance of a. We present results for $\ell = 2$ and two distinct values of the standard deviation: $\sigma = 0.1$ and $\sigma = 0.3$.

			Multi-parameter				Parameter-free			
m	p	n_ξ	k	n_k	rank	time	k	n_k	rank	time
8	2	45	8	58	40	2.27e0	8	59	40	2.18e0
	3	165	9	66	65	2.50e0	9	67	66	2.71e0
	4	495	9	66	66	2.65e0	9	67	66	2.75e0
	5	1,287	9	66	66	3.01e0	9	67	66	3.08e0
12	2	91	7	76	62	3.25e0	8	87	63	3.86e0
	3	455	8	87	87	3.98e0	8	87	87	3.99e0
	4	1,820	8	87	87	5.07e0	8	87	87	5.10e0
	5	6,188	8	87	87	1.23e1	8	87	87	1.22e1
20	2	231	10	172	125	1.07e1	10	171	125	1.21e1
	3	1,771	10	172	172	1.54e1	10	171	171	1.41e1
	4	10,626	10	172	172	5.38e1	10	171	171	5.12e1

TABLE 5.5

Results obtained with the MultiRB solver for Example 5.2 with $\sigma = 0.1$ and grid level 7: multi-parameter vs parameter-free implementations.

			Multi-parameter				Parameter-free			
m	p	n_ξ	k	n_k	rank	time	k	n_k	rank	time
8	2	45	16	124	45	5.37e0	17	128	45	5.57e0
	3	165	21	163	128	8.08e0	20	152	127	7.18e0
	4	495	24	187	182	1.05e1	24	183	178	1.04e1
	5	1,287	26	203	203	1.87e1	27	207	207	1.87e1
12	2	91	15	166	89	8.52e0	15	165	89	8.54e0
	3	455	18	202	197	1.22e0	18	201	197	1.20e1
	4	1,820	21	236	236	2.59e1	21	236	236	2.60e1
	5	6,188	25	282	282	1.17e2	25	282	282	1.14e2
20	2	231	16	283	203	2.08e1	16	281	206	2.00e1
	3	1,771	23	403	403	6.45e1	23	399	399	6.28e1
	4	10,626	26	459	459	4.38e2	26	454	454	4.11e2

TABLE 5.6

Results obtained with the MultiRB solver for Example 5.2 with $\sigma = 0.3$ and grid level 7: multi-parameter vs parameter-free implementations.

First, we fix the grid level to be 7 and compare the multiple-parameter and parameter-free implementations of the reduced basis solver. For the latter, we set $s_* = 2$ as before. For the former, we choose the shifts as in (4.4). The matrix $\widehat{K}_1$ is positive definite now. In the case $\sigma = 0.1$ (the low standard deviation case), we obtain $\alpha_1 = 0.8794$. The eigenvalues of A_1 are then contained in $[0.9737, 1.0263]$. This interval is not large enough to contain the eigenvalues of each of $A_2, \ldots, A_m$ so we use $\mathcal{S} = [1 + \lambda_{2,\min}, 1 + \lambda_{2,\max}] \approx [0.9219, 1.0784]$. Similarly, in the case $\sigma = 0.3$, we obtain

$\alpha_1 = 0.6381$ and use $\mathcal{S} = [1 + \lambda_{2,\min}, 1 + \lambda_{2,\max}] \approx [0.7657, 1.2344]$. Notice that $\mathcal{S}$ is larger when σ increases. For both values of the standard deviation, we compute 5 nodes $[s_1, s_2, s_3, s_4, s_5]$ on $\mathcal{S}$, using the MATLAB code from [19, p.43] as before. Results are presented in Tables 5.5 and 5.6.

We see that both implementations of the reduced basis solver work equally well. This is no surprise since the end points of the interval $\mathcal{S}$ used by the multi-parameter version are a much smaller perturbation from 1 here than in Example 5.1. The computed values of s_j are all close to 1 (the mid-point of $\mathcal{S}$), especially in the case $\sigma = 0.1$, which is effectively the choice made in the parameter-free version. It is also interesting to note that both versions of our new method generate approximation spaces whose dimensions match the rank of the final approximation (unless n_ξ is small). In this sense, the reduced basis is optimal. Notice also that the recorded ranks are much larger than in Example 5.1 and they increase when both the standard deviation σ and the number of random variables m increase. As σ and m increase, the variance of the truncated diffusion coefficient increases so it is intuitive that a richer approximation space would then be needed to capture the solution.

m	p	n_ξ	k	inner	n_k	rank	time	CG time (its)
8	2	45	17	9.8	128	45	5.57e0	2.47e0 (8)
	3	165	20	12.1	152	127	7.18e0	9.93e0 (10)
	4	495	24	14.5	183	178	1.04e1	3.58e1 (12)
	5	1,287	27	17.0	207	207	1.87e1	9.90e1 (13)
12	2	91	15	9.9	165	89	8.54e0	5.52e0 (8)
	3	455	18	12.2	201	197	1.20e1	3.31e1 (10)
	4	1,820	21	15.0	236	236	2.60e1	1.54e2 (12)
	5	6,188	25	18.6	282	282	1.14e2	5.72e2 (13)
20	2	231	16	9.4	281	206	2.01e1	1.72e1 (8)
	3	1,771	23	12.3	399	399	6.28e1	1.70e2 (10)
	4	10,626	26	15.4	454	454	4.11e2	1.18e3 (12)

TABLE 5.7
Results for Example 5.2 with $\sigma = 0.3$ and grid level 7: MultiRB solver vs standard CG.

m	p	n_ξ	k	inner	n_k	rank	time	CG time (its)
8	2	45	17	9.8	128	45	3.21e1	1.34e1 (8)
	3	165	21	12.2	160	129	4.14e1	5.66e1 (10)
	4	495	24	14.5	183	178	5.11e1	1.97e2 (12)
	5	1,287	27	16.9	207	207	6.40e1	5.53e2 (13)
12	2	91	15	9.9	165	89	4.78e1	3.00e1 (8)
	3	455	18	12.2	201	196	6.16e1	1.75e2 (10)
	4	1,820	21	15.0	236	236	8.64e1	8.21e2 (12)
	5	6,188	25	18.6	281	281	1.88e2	3.07e3 (13)
20	2	231	16	9.4	281	206	1.11e2	9.47e1 (8)
	3	1,771	23	12.3	399	399	1.97e2	8.45e2 (10)
	4	10,626	26	15.4	454	454	5.56e2	Out of Memory

TABLE 5.8
Results for Example 5.2 with $\sigma = 0.3$ and grid level 8: MultiRB solver vs standard CG.

Adopting the parameter-free implementation for simplicity, we now compare the efficiency of using the MultiRB method to solve the matrix equation (3.5) and using CG to solve (3.6). Results obtained for grid levels 7 and 8 are presented in Tables 5.7 and 5.8 respectively. Here, we consider

only the more challenging case $\sigma = 0.3$. Comparing the results in Tables 5.7 and 5.8 we see that the dimension n_k of the approximation space built by the MultiRB solver is independent of h, as predicted. The rank of the final approximation is also observed to be independent of h. Once again, when n_ξ is small there is nothing to be gained by using the low-rank solver. Using CG on the Kronecker formulation is quicker and uses less memory. However, when n_ξ is large (typically $n_\xi > 100$) using the reduced basis solver on the matrix formulation is much quicker and uses substantially less memory. Indeed, in the largest experiment, where $n_\xi = 10,626$ and the grid level is 8, the Kronecker formulation consists of 690 million equations. We run out of memory when we apply CG to (3.6) but our new method is able to solve the matrix formulation of the system easily.

6. Summary and conclusions. This paper describes the design and implementation of the MultiRB method, an innovative solver for linear systems arising from stochastic Galerkin finite element approximation of elliptic PDEs with random coefficients. We consider the multi-term matrix equation formulation of the systems and develop a new solver that builds up an approximation space on-the-fly that exploits the inherent low-rank structure. The approximation space generated is inspired by ideas from rational approximation. Our numerical results demonstrate that high-dimensional ($> 10^9$ unknowns) stochastic Galerkin linear systems can be solved in a few minutes on contemporary desktop computers. Moreover, the new solver requires significantly less memory than conventional mean-based preconditioned CG.

REFERENCES

[1] I. Babuška, R. Tempone, and G.E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM J. Numer. Anal.*, 42(2):800–825, 2004.

[2] J. Ballani and L. Grasedyck. A projection method to solve linear systems in tensor format. *Numer. Linear Algebra Appl.*, 20:27–43, 2013.

[3] B. Beckermann and L. Reichel. Error estimation and evaluation of matrix functions via the Faber transform. *SIAM J. Numer. Anal.*, 47(5):3849–3883, 2009.

[4] M.K. Deb, I.M. Babuška, and J.T. Oden. Solution of stochastic partial differential equations using Galerkin finite element techniques. *Comput. Methods Appl. Mech. Engrg.*, 190(48):6359–6372, 2001.

[5] V. Druskin, C. Lieberman, and M. Zaslavsky. On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. *SIAM J. Sci. Comput.*, 32(5):2485–2496, 2010.

[6] V. Druskin and V. Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems and Control Letters*, 60:546–560, 2011.

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

[8] H. Elman and D. Furnival. Solving the stochastic steady-state diffusion problem using multigrid. *IMA J. Numer. Anal.*, 27(4):675–688, 2007.

[9] O.G. Ernst and E. Ullmann. Stochastic Galerkin matrices. *SIAM J. Matrix Anal. Appl.*, 31(4):1848–1872, 2009/10.

[10] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, Maryland, 1996.

[11] S. Güttel. Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. *GAMM Mitteilungen*, 36(1):8–31, 2013.

[12] B. Khoromskij and C. Schwab. Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. *SIAM J. Sci. Comput.*, 33(1):364–385, 2011.

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

[14] A. Lu and E.L. Wachspress. Solution of Lyapunov equations by Alternating Direction Implicit iteration. *Computers and Mathematics with Applications*, 21(9):43–58, 1991.

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

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

[17] C.E. Powell and H.C. Elman. Block-diagonal preconditioning for spectral stochastic finite-element systems. *IMA J. Numer. Anal.*, 29(2):350–375, 2009.

[18] E. Rosseel and S. Vandewalle. Iterative solvers for the stochastic finite element method. *SIAM J. Sci. Comput.*, 32(1):372–397, 2010.
[19] J. Sabino. *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*. PhD thesis, Rice University, 2006.
[20] D.J. Silvester, A. Bespalov, and C.E. Powell. S-IFISS version 1.01, April 2015. available online at `http://www.manchester.ac.uk/ifiss/s-ifiss1.0.tar.gz`.
[21] V. Simoncini. Computational methods for linear matrix equations. Technical report, Università di Bologna, March 2013.
[22] B. Sousedik, R.G. Ghanem, and E.T. Phipps. Hierarchical Schur complement preconditioner for the stochastic Galerkin finite element methods. *Numer. Linear Algebra Appl.*, 21(1):136–151, 2014.
[23] E. Ullmann. A Kronecker product preconditioner for stochastic Galerkin finite element discretizations. *SIAM J. Sci. Comput.*, 32(2):923–946, 2010.