SIAM J. SCI. COMPUT.
Vol. 41, No. 4, pp. A2657–A2680

LOW-RANK SOLUTION METHODS FOR STOCHASTIC EIGENVALUE PROBLEMS*

HOWARD C. ELMAN† AND TENGFEI SU‡

Abstract. We study efficient solution methods for stochastic eigenvalue problems arising from discretization of self-adjoint PDEs with random data, where the underlying operators depend linearly on the random parameters. With the stochastic Galerkin approach, the solutions are represented as generalized polynomial chaos expansions. When these solutions can be approximated well by low-rank objects, we introduce a low-rank variant of the inverse subspace iteration algorithm for computing one or several minimal eigenvalues and corresponding eigenvectors of parameter-dependent matrices. In the algorithm, the iterates are approximated by low-rank matrices, which leads to significant cost savings. The algorithm is tested on two benchmark problems: a stochastic diffusion problem with some poorly separated eigenvalues and an operator derived from a discrete stochastic Stokes problem whose minimal eigenvalue is related to the inf-sup stability constant. Numerical experiments show that the low-rank algorithm produces accurate solutions compared to the Monte Carlo method, and it uses much less computational time than the original algorithm without low-rank approximation.

Key words. stochastic eigenvalue problem, inverse subspace iteration, low-rank approximation

AMS subject classifications. 35R60, 65F15, 65F18, 65N22

DOI. 10.1137/18M122100X

1. Introduction.

Approaches for solving stochastic eigenvalue problems can be broadly divided into nonintrusive methods, including Monte Carlo methods and stochastic collocation methods [1, 32], and intrusive stochastic Galerkin methods. The Galerkin approach gives parametrized descriptions of the eigenvalues and eigenvectors, represented as expansions with stochastic basis functions. A commonly used framework is the generalized polynomial chaos (gPC) expansion [44]. A direct projection onto the subspace spanned by the basis functions will result in large coupled nonlinear systems that can be solved by a Newton-type algorithm [5, 13]. Alternatives that do not use nonlinear solvers are stochastic versions of the (inverse) power methods and subspace iteration algorithms [16, 17, 28, 38, 42]. These methods have been shown to produce accurate solutions compared with the Monte Carlo or collocation methods. However, due to the extra dimensions introduced by randomness, solving the linear systems, as well as other computations, can be expensive. In this paper, we develop new efficient solution methods that use low-rank approximations for the stochastic eigenvalue problems within the stochastic Galerkin approach.

Low-rank methods have been explored for solution of stochastic/parametrized PDEs and high-dimensional PDEs. Discretization of such PDEs gives large, sparse, and in general structured linear systems. Iterative solvers construct approximate

*Submitted to the journal's Methods and Algorithms for Scientific Computing section October 31, 2018; accepted for publication (in revised form) May 21, 2019; published electronically August 22, 2019.

https://doi.org/10.1137/18M122100X

Funding: This work was supported by the U.S. Department of Energy Office of Advanced Scientific Computing Research, Applied Mathematics program, under award DE-SC0009301 and by the U.S. National Science Foundation under grant DMS1819115.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (elman@cs.umd.edu).

‡Applied Mathematics & Statistics, and Scientific Computation Program, University of Maryland, College Park, MD 20742 (tengfesu@math.umd.edu).

solutions of low-rank matrix or tensor structure so that the matrix-vector products can be computed cheaply. Combined with rank compression techniques, the iterates are forced to stay in low-rank format. This idea has been used with Krylov subspace methods [2, 4, 22, 25] (note that with low-rank compression, these methods become inexact methods) and multigrid methods [9, 15]. The low-rank solution methods solve the linear systems to a certain accuracy with much less computational effort and facilitate the treatment of larger problem scales. Low-rank iterative solvers were also used in [3] for optimal control problems constrained by stochastic PDEs.

In this study, we use the stochastic Galerkin approach to compute gPC expansions of one or more minimal eigenvalues and corresponding eigenvectors of parameter-dependent matrices, arising from discretization of stochastic self-adjoint PDEs. Our work builds on the results in [28, 38]. We devise a low-rank variant of the stochastic inverse subspace iteration algorithm, where the iterates and solutions are approximated by low-rank matrices. In each iteration, the linear system solves required by the inverse iteration algorithm are performed by low-rank iterative solvers. The orthonormalization and Rayleigh quotient computations in the algorithm are also computed with the low-rank representation. To test the efficiency of the proposed algorithm, we consider two benchmark problems: a stochastic diffusion problem and a Schur complement operator derived from a discrete stochastic Stokes problem. The diffusion problem has some poorly separated eigenvalues, and we show that a generalization of Rayleigh–Ritz refinement for the stochastic problem can be used to obtain good approximations. A low-rank geometric multigrid method is used for solving the linear systems. For the Stokes problem, the minimal eigenvalue of the Schur complement operator is the square of the parametrized inf-sup stability constant for the Stokes operator. Each step of the inverse iteration entails solving a Stokes system for which a low-rank variant of the MINRES method is used. We demonstrate the accuracy of the solutions and efficiency of the low-rank algorithms by comparison with the Monte Carlo method and the full subspace iteration algorithm without using low-rank approximation.

We note that a low-rank variant of locally optimal block preconditioned conjugate gradient method was studied in [23] for eigenvalue problems from discretization of high-dimensional elliptic PDEs. A low-rank Arnoldi method was proposed in [6] to approximate the posterior covariance matrix in stochastic inverse problems. Another dimension reduction technique is the reduced basis method. This idea was used in [11, 18, 27], where the eigenvectors are approximated from a linear space spanned by carefully selected sample "snapshot" solutions obtained via, for instance, a greedy algorithm that minimizes an a posteriori error estimator. Inf-sup stability problems were also studied in [19, 36] in which lower and upper bounds for the smallest eigenvalue of a stochastic Hermitian matrix are computed using successive constraint methods in the reduced basis context.

The rest of the paper is organized as follows. In section 2 we review the stochastic inverse subspace iteration algorithm for computing several minimal eigenvalues and corresponding eigenvectors of parameter-dependent matrices. In section 3 we introduce the idea of low-rank approximation in this setting and discuss how computations in the inverse subspace iteration algorithm are done efficiently with quantities in low-rank format. The stochastic diffusion problem and the stochastic Stokes problem are discussed in sections 4 and 5, respectively, with numerical results showing the effectiveness of the low-rank algorithms. Conclusions are drawn in the last section.

2. Stochastic inverse subspace iteration. Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability triplet where Ω is a sample space with σ-algebra $\mathcal{F}$ and probability measure $\mathcal{P}$. Define a random variable $\xi : \Omega \to \Gamma \subset \mathbb{R}^m$ with uncorrelated components, and let μ be the induced measure on Γ. Consider the following stochastic eigenvalue problem: Find n_e minimal eigenvalues $\lambda^s(\xi)$ and corresponding eigenvectors $u^s(\xi)$ such that

$$A(\xi)u^s(\xi) = \lambda^s(\xi)u^s(\xi), \quad s = 1, 2, \ldots, n_e, \tag{2.1}$$

almost surely, where $A(\xi)$ is a matrix-valued random variable. We will use a version of stochastic inverse subspace iteration studied in [28, 38] for the solution of (2.1). The approach derives from a stochastic Galerkin formulation of subspace iteration, which is based on projection onto a finite-dimensional subspace of $L^2(\Gamma)$ spanned by the gPC basis functions $\{\psi_k(\xi)\}_{k=1}^{n_\xi}$. These functions are orthonormal with

$$\langle \psi_i \psi_j \rangle = \mathbb{E}[\psi_i \psi_j] = \int_\Gamma \psi_i(\xi)\psi_j(\xi)\mathrm{d}\mu = \delta_{ij}, \tag{2.2}$$

where $\langle \cdot \rangle$ is the expected value and δ_{ij} is the Kronecker delta. The stochastic Galerkin solutions are expressed as expansions of the gPC basis functions,

$$\lambda^s(\xi) = \sum_{r=1}^{n_\xi} \lambda_r^s \psi_r(\xi), \quad u^s(\xi) = \sum_{j=1}^{n_\xi} u_j^s \psi_j(\xi). \tag{2.3}$$

We briefly review the stochastic subspace iteration method in the case where $A(\xi)$ admits an affine expansion with respect to components of the random variable ξ:

$$A(\xi) = A_0 + \sum_{l=1}^{m} A_l \xi_l, \tag{2.4}$$

where each A_l is an $n_x \times n_x$ deterministic matrix, obtained from, for instance, finite element discretization of a PDE operator. The matrix A_0 is the mean value of $A(\xi)$. Such a representation can be obtained from a Karhunen–Loève (KL) expansion [26] of the stochastic term in the problem (see (4.2)). Let $\{u^{s,(i)}(\xi)\}_{s=1}^{n_e}$ be a set of approximate eigenvectors obtained at the ith step of the inverse subspace iteration. Then at step $i+1$, one needs to solve

$$\langle A(\xi)v^{s,(i+1)}\psi_k \rangle = \langle u^{s,(i)}\psi_k \rangle, \quad k = 1, 2, \ldots, n_\xi, \tag{2.5}$$

for $\{v^{s,(i+1)}\}_{s=1}^{n_e}$ and compute $\{u^{s,(i+1)}\}_{s=1}^{n_e}$ via orthonormalization. If $n_e = 1$, for the latter requirement, $v^{s,(i+1)}$ is normalized so that $||u^{s,(i+1)}||_2 = 1$ almost surely. If $n_e > 1$, a stochastic version of the Gram–Schmidt process is applied, and the resulting vectors $\{u^{s,(i+1)}\}_{s=1}^{n_e}$ satisfy $\langle u^{s,(i+1)}, u^{t,(i+1)} \rangle_{\mathbb{R}^{n_x}} = \delta_{st}$ almost surely, where $\langle \cdot, \cdot \rangle_{\mathbb{R}^{n_x}}$ is the Euclidean inner product in $\mathbb{R}^{n_x}$. With the iterates expressed as gPC expansions, for instance, $u^{s,(i)}(\xi) = \sum_{j=1}^{n_\xi} u_j^{s,(i)} \psi_j(\xi)$, collecting the n_ξ equations in (2.5) for each s yields an $n_x n_\xi \times n_x n_\xi$ linear system

$$\sum_{l=0}^{m} (G_l \otimes A_l)\mathbf{v}^{s,(i+1)} = \mathbf{u}^{s,(i)}, \tag{2.6}$$

where $\otimes$ is the Kronecker product, each G_l is an $n_\xi \times n_\xi$ matrix with $[G_l]_{kj} = \langle \xi_l \psi_k \psi_j \rangle$

($\xi_0 \equiv 1$ and $G_0 = I$), and

$$\mathbf{u}^{s,(i)} = \begin{pmatrix} u_1^{s,(i)} \\ u_2^{s,(i)} \\ \vdots \\ u_{n_\xi}^{s,(i)} \end{pmatrix} \in \mathbb{R}^{n_x n_\xi}. \tag{2.7}$$

Note that the matrices $\{G_l\}$ are sparse due to orthogonality of the gPC basis functions [10, 30]. The initial iterate is given by solving the mean problem $A_0 \bar{u}^s = \bar{\lambda}^s \bar{u}^s$ and

$$\mathbf{u}^{s,(0)} = \begin{pmatrix} \bar{u}^s \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{2.8}$$

When the variance of the random parameters in (2.4) is small, the mean problem provides a good initial value for the algorithm. The complete algorithm is summarized as Algorithm 2.1. The details of the computations in steps 4 and 7 are given in sections 3.2 and 3.3.

Algorithm 2.1: Stochastic inverse subspace iteration.

1: **initialization**: initial iterate $\mathbf{u}^{s,(0)}$.
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Solve the stochastic Galerkin system (2.6) for $\mathbf{v}^{s,(i+1)}$, $s = 1, 2, \ldots, n_e$.
4: If $n_e = 1$, compute $\mathbf{u}^{s,(i+1)}$ by normalization. Otherwise, apply a stochastic Gram–Schmidt process for orthonormalization.
5: Check convergence.
6: **end**
7: Compute eigenvalues using a Rayleigh quotient.

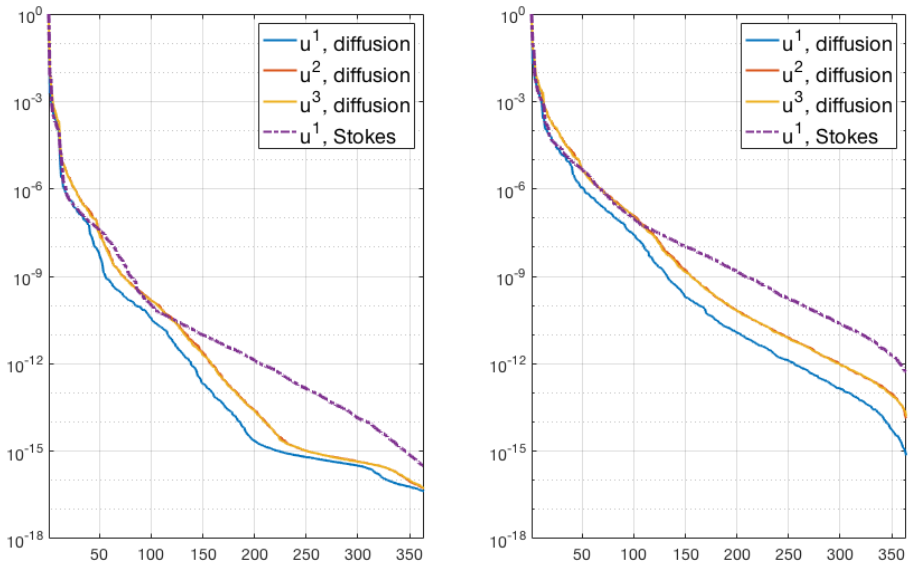
3. Low-rank approximation. In this section we discuss the idea of low-rank approximation and how this can be used to reduce the computational costs of Algorithm 2.1. The size of the Galerkin system (2.6) is in general large, and solving the system can be computationally expensive. We utilize low-rank iterative solvers where the iterates are approximated by low-rank matrices and the system is efficiently solved to a specified accuracy. In addition, low-rank forms can be used to reduce the costs of the orthonormalization and Rayleigh quotient computations in the algorithm.

3.1. System solution. For any random vector $x(\xi)$ with expansion $x(\xi) = \sum_{j=1}^{n_\xi} x_j \psi_j(\xi)$, where each x_j is a vector of length n_x, let

$$X = \mathrm{mat}(\mathbf{x}) = [x_1, x_2, \ldots, x_{n_\xi}] \in \mathbb{R}^{n_x \times n_\xi}. \tag{3.1}$$

Then the Galerkin system $\sum_{l=0}^{m} (G_l \otimes A_l)\mathbf{x} = \mathbf{f}$ is equivalent to the matrix form

$$\sum_{l=0}^{m} A_l X G_l^T = F = \mathrm{mat}(\mathbf{f}). \tag{3.2}$$



(a) $\sigma = 0.01$ (b) $\sigma = 0.1$

FIG. 3.1. *Singular values (relative to the largest one) of the matrix representations of the stochastic eigenvectors for the numerical examples in sections* 4 *and* 5 *with standard deviations* $\sigma = 0.01$ *and* $\sigma = 0.1$. $n_c = 5$, $b = 4.0$, $m = 11$, $n_\xi = 364$.

It was shown in [4] that for a symmetric and positive definite problem where the matrices $\{G_l\}$ and F have small ranks, the solution X can be approximated by a low-rank matrix. For the examples considered in this study, when the variance of the random parameters is small, the singular values of the solution matrix decay exponentially fast (see Figure 3.1), and a low-rank approximate solution can be obtained by dropping the terms corresponding to small singular values in a singular value decomposition (SVD).

To take advantage of the low rank of the solution matrix, we construct iterative solvers that produce a sequence of low-rank approximate iterates. Let $X^{(i)} = \mathrm{mat}(\mathbf{x}^{(i)})$ be the ith iterate computed by an iterative solver applied to (3.2), and suppose $X^{(i)}$ is represented as the product of two rank-κ matrices, i.e., $X^{(i)} = Y^{(i)} Z^{(i)T}$, where $Y^{(i)} \in \mathbb{R}^{n_x \times \kappa}$, $Z^{(i)} \in \mathbb{R}^{n_\xi \times \kappa}$. If this factored form is used throughout the iteration without explicitly forming $X^{(i)}$, then the matrix-vector product $(G_l \otimes A_l)\mathbf{x}$ will have the same structure,

$$A_l X^{(i)} G_l^T = (A_l Y^{(i)})(G_l Z^{(i)})^T, \tag{3.3}$$

and it is only necessary to compute $A_l Y^{(i)}$ and $G_l Z^{(i)}$. If $\kappa \ll \min(n_x, n_\xi)$, this means that the computational costs of the matrix operation are reduced from $O(n_x n_\xi)$ to $O((n_x + n_\xi)\kappa)$. (Note that the matrices $\{G_l\}$ and $\{A_l\}$ obtained from the stochastic Galerkin method are sparse.) On the other hand, summing terms with the factored form tends to increase the rank, and rank compression techniques must be used in each iteration to force the matrix rank κ to stay low. In particular, if $X_1^{(i)} = Y_1^{(i)} Z_1^{(i)T}$, $X_2^{(i)} = Y_2^{(i)} Z_2^{(i)T}$, where $Y_1^{(i)} \in \mathbb{R}^{n_x \times \kappa_1}$, $Z_1^{(i)} \in \mathbb{R}^{n_\xi \times \kappa_1}$, $Y_2^{(i)} \in \mathbb{R}^{n_x \times \kappa_2}$, $Y_2^{(i)} \in \mathbb{R}^{n_\xi \times \kappa_2}$,

then

$$X_1^{(i)} + X_2^{(i)} = [Y_1^{(i)}, Y_2^{(i)}][Z_1^{(i)}, Z_2^{(i)}]^T. \tag{3.4}$$

The addition gives a matrix of rank $\kappa_1 + \kappa_2$ in the worst case. Rank compression can be achieved by an SVD-based truncation operator $\tilde{X}^{(i)} = \mathcal{T}(X^{(i)})$ so the matrix $\tilde{X}^{(i)}$ has a much smaller rank than $X^{(i)}$ [22]. Specifically, we compute QR factorizations $Y^{(i)} = Q_Y R_Y$ and $Z^{(i)} = Q_Z R_Z$ and an SVD $R_Y R_Z^T = \hat{Y}\mathrm{diag}(\sigma_1, \ldots, \sigma_\kappa)\hat{Z}^T$, where $\sigma_1, \ldots, \sigma_\kappa$ are the singular values in decreasing order. We can truncate to a rank-$\tilde{\kappa}$ matrix by dropping the terms corresponding to small singular values with a relative criterion $\sqrt{\sigma_{\tilde{\kappa}+1}^2 + \cdots + \sigma_\kappa^2} \le \epsilon_{\mathrm{rel}}\sqrt{\sigma_1^2 + \cdots + \sigma_\kappa^2}$ or an absolute one $\tilde{\kappa} = \max\{\tilde{\kappa} \mid \sigma_{\tilde{\kappa}} \ge \epsilon_{\mathrm{abs}}\}$. In MATLAB notation, the truncated matrix is $\tilde{X}^{(i)} = \tilde{Y}^{(i)}\tilde{Z}^{(i)T}$ with

$$\tilde{Y}^{(i)} = Q_Y\hat{Y}(:, 1:\tilde{\kappa}), \quad \tilde{Z}^{(i)} = Q_Z\hat{Z}(:, 1:\tilde{\kappa})\mathrm{diag}(\sigma_1, \ldots, \sigma_{\tilde{\kappa}}). \tag{3.5}$$

Low-rank approximation and truncation have been used for Krylov subspace methods [4, 22, 25] and multigrid methods [9]. More details can be found in these references. We will use examples of such solvers for linear systems arising in eigenvalue computations, as discussed in sections 4 and 5.

3.2. Orthonormalization. In Algorithm 2.1, if $n_e = 1$, the solution $v^{s,(i+1)}(\xi)$ is normalized so that $||u^{s,(i+1)}(\xi)||_2 = 1$ almost surely. With the superscripts omitted, assume $u(\xi) = \sum_{j=1}^{n_\xi} u_j\psi_j(\xi)$ is the normalized random vector constructed from $v(\xi)$. This expansion can be computed using sparse grid quadrature $\{\xi^{(q)}, \eta^{(q)}\}_{q=1}^{n_q}$, where $\{\eta^{(q)}\}$ are the weights [12]:

$$u_j = \langle u(\xi)\psi_j(\xi)\rangle = \left\langle \frac{v(\xi)}{\|v(\xi)\|_2}\psi_j(\xi) \right\rangle \approx \sum_{q=1}^{n_q} \frac{v(\xi^{(q)})}{\|v(\xi^{(q)})\|_2}\psi_j(\xi^{(q)})\eta^{(q)}. \tag{3.6}$$

Suppose the "matricized" version of the expansion coefficients of $v(\xi)$ is represented in low-rank form

$$V = [v_1, v_2, \ldots, v_{n_\xi}] = Y_v Z_v^T, \tag{3.7}$$

where $Y_v \in \mathbb{R}^{n_x \times \kappa_v}$, $Z_v \in \mathbb{R}^{n_\xi \times \kappa_v}$. With $\Psi(\xi^{(q)}) = [\psi_1(\xi^{(q)}), \psi_2(\xi^{(q)}), \ldots, \psi_{n_\xi}(\xi^{(q)})]^T$, we have

$$v(\xi^{(q)}) = \sum_{j=1}^{n_\xi} v_j\psi_j(\xi^{(q)}) = V\Psi(\xi^{(q)}) = Y_v Z_v^T\Psi(\xi^{(q)}). \tag{3.8}$$

Let $U = [u_1, u_2, \ldots, u_{n_\xi}]$. Then (3.6) yields

$$[U]_{:,j} = u_j = \sum_{q=1}^{n_q} \frac{Y_v Z_v^T\Psi(\xi^{(q)})}{\|Y_v Z_v^T\Psi(\xi^{(q)})\|_2}\psi_j(\xi^{(q)})\eta^{(q)} \tag{3.9}$$

and

$$U = \sum_{q=1}^{n_q} \frac{Y_v Z_v^T\Psi(\xi^{(q)})}{\|Y_v Z_v^T\Psi(\xi^{(q)})\|_2}\Psi(\xi^{(q)})^T\eta^{(q)}. \tag{3.10}$$

Thus, the matrix U can be expressed as an outer product of two low-rank matrices $U = Y_u Z_u^T$ with

$$Y_u = Y_v \in \mathbb{R}^{n_x \times \kappa_v}, \quad Z_u = \sum_{q=1}^{n_q} \frac{\Psi(\xi^{(q)})(\Psi(\xi^{(q)})^T Z_v)}{\|Y_v(Z_v^T \Psi(\xi^{(q)}))\|_2} \eta^{(q)} \in \mathbb{R}^{n_\xi \times \kappa_v}. \tag{3.11}$$

This implies that the expansion coefficients of the normalized vector $u(\xi)$ can be written as a low-rank matrix with the same rank as the analogous matrix associated with $v(\xi)$. The cost of computing Z_u is $O((n_x + n_\xi)n_q\kappa_v)$. Since in general $n_q \gg \kappa_v$, it can be further reduced to $O((n_x + n_q)\kappa_v^2 + n_\xi n_q \kappa_v)$ by first computing a QR factorization of Y_v and factoring out the orthogonal matrix in the denominator.

In the general case where more than one eigenvector is computed ($n_e > 1$), a stochastic version of the Gram–Schmidt process is applied to compute an orthonormal set $\{u^{s,(i+1)}(\xi)\}_{s=1}^{n_e}$ [28, 38]. With the superscript $(i+1)$ omitted, the process is based on the calculation

$$u^s(\xi) = v^s(\xi) - \sum_{t=1}^{s-1} \chi^{ts}(\xi) = v^s(\xi) - \sum_{t=1}^{s-1} \frac{\langle v^s(\xi), u^t(\xi)\rangle_{\mathbb{R}^{n_x}}}{\langle u^t(\xi), u^t(\xi)\rangle_{\mathbb{R}^{n_x}}} u^t(\xi) \tag{3.12}$$

for $s = 2, \ldots, n_e$. If we write $\chi^{ts}(\xi) = \sum_{k=1}^{n_\xi} \chi_k^{ts} \psi_k(\xi)$ and assume $u^t(\xi)$ is already normalized in previous steps, then

$$\begin{aligned} \chi_k^{ts} &= \langle v^s(\xi)^T u^t(\xi) u^t(\xi) \psi_k(\xi)\rangle \\ &\approx \sum_{q=1}^{n_q} v^s(\xi^{(q)})^T u^t(\xi^{(q)}) u^t(\xi^{(q)}) \psi_k(\xi^{(q)}) \eta^{(q)} \\ &= \sum_{q=1}^{n_q} (\Psi(\xi^{(q)})^T Z_{v^s} Y_{v^s}^T)(Y_{u^t} Z_{u^t}^T \Psi(\xi^{(q)})) Y_{u^t} Z_{u^t}^T \Psi(\xi^{(q)}) \psi_k(\xi^{(q)}) \eta^{(q)}. \end{aligned} \tag{3.13}$$

The last line follows from (3.8). Let $\zeta^{ts}(\xi^{(q)}) = (\Psi(\xi^{(q)})^T Z_{v^s} Y_{v^s}^T)(Y_{u^t} Z_{u^t}^T \Psi(\xi^{(q)}))$; then the matrix $X^{ts} = [\chi_1^{ts}, \chi_2^{ts}, \ldots, \chi_{n_\xi}^{ts}]$ can be expressed in low-rank form $X^{ts} = Y_{\chi^{ts}} Z_{\chi^{ts}}^T$ with

$$Y_{\chi^{ts}} = Y_{u^t}, \quad Z_{\chi^{ts}} = \sum_{q=1}^{n_q} \Psi(\xi^{(q)})(\Psi(\xi^{(q)})^T Z_{u^t}) \zeta^{ts}(\xi^{(q)}) \eta^{(q)}. \tag{3.14}$$

With low-rank representation, the computational cost is $O((n_x+n_\xi)n_q \max(\kappa_{v^s}, \kappa_{u^t}))$. Note that in (3.12) the summation will increase the matrix rank, and thus a truncation operator is applied to compress the rank. In numerical experiments presented below (see sections 4 and 5), we use an absolute truncation criterion for this compression with $\epsilon_{\text{abs}} = 10^{-8}$.

3.3. Rayleigh quotient. The Rayleigh quotient in step 7 of Algorithm 2.1 is computed (only once) after convergence of the inverse subspace iteration to find the eigenvalues. Given a normalized eigenvector $u(\xi)$ of problem (2.1), the computation of the stochastic Rayleigh quotient

$$\lambda(\xi) = u(\xi)^T A(\xi) u(\xi) \tag{3.15}$$

involves two steps:

(1) Compute matrix-vector product $w(\xi) = A(\xi)u(\xi)$, where $w(\xi) = \sum_{k=1}^{n_\xi} w_k\psi_k(\xi)$ and $w_k = \langle Au\psi_k\rangle$. In Kronecker product form,

$$\mathbf{w} = \sum_{l=0}^{m}(G_l \otimes A_l)\mathbf{u}. \tag{3.16}$$

If $\mathbf{u}$ has low-rank representation $U = Y_u Z_u^T$, then

$$W = \sum_{l=0}^{m}(A_l Y_u)(G_l Z_u)^T. \tag{3.17}$$

This is followed by a truncation operation to compress the matrix rank. Again, in experiments discussed below, we use an absolute truncation operation with $\epsilon_{\text{abs}} = 10^{-8}$.

(2) Compute eigenvalue $\lambda(\xi) = u(\xi)^T w(\xi)$, where $\lambda(\xi) = \sum_{r=1}^{n_\xi} \lambda_r\psi_r(\xi)$ and $\lambda_r = \langle u^T w\psi_r\rangle$. Equivalently,

$$\lambda_r = \langle \tilde{G}_r, H\rangle_{\mathbb{R}^{n_\xi\times n_\xi}} = \sum_{j,k=1}^{n_\xi} [\tilde{G}_r]_{jk}H_{jk}, \tag{3.18}$$

where $H_{jk} = u_j^T w_k$ and thus $H = U^T W = Z_u(Y_u^T Y_w)Z_w^T$. The matrices $\{\tilde{G}_r\}_{r=1}^{n_\xi}$ are sparse with $[\tilde{G}_r]_{jk} = \langle\psi_r\psi_j\psi_k\rangle$. In fact, if the basis functions are written as products of univariate polynomials, i.e.,

$$\psi_r(\xi) = \psi_{r_1}(\xi_1)\psi_{r_2}(\xi_2)\cdots\psi_{r_m}(\xi_m), \tag{3.19}$$

then $[\tilde{G}_r]_{jk}$ is nonzero only if $|j_l - k_l| \le r_l \le j_l + k_l$ and $r_l + j_l + k_l$ is even for all $1 \le l \le m$ [10]. This observation greatly reduces the cost of assembling the matrices $\{\tilde{G}_r\}$. For example, if $m = 11$, the degree of the gPC basis functions is $p \le 3$, and $n_\xi = (m+p)!/(m!p!) = 364$, then with the above rule, a total of 31098 nonzero entries must be computed instead of the much larger number $n_\xi^3 = 48228544$ if the sparsity of $\{\tilde{G}_r\}$ is not used.

3.4. Convergence criterion. To check convergence, we can look at the magnitude of the expected value of the residual

$$r^s(\xi) = A(\xi)u^s(\xi) - \lambda^s(\xi)u^s(\xi), \quad s = 1, 2, \ldots, n_e. \tag{3.20}$$

Alternatively, without computing the Rayleigh quotient at each iteration, error assessment can be done using the relative difference of the gPC coefficients of two successive iterates, i.e.,

$$\epsilon_{\Delta u}^{s,(i)} = \frac{1}{n_\xi}\sum_{k=1}^{n_\xi}\frac{\|u_k^{s,(i)} - u_k^{s,(i-1)}\|_2}{\|u_k^{s,(i-1)}\|_2}. \tag{3.21}$$

However, in the case of clustered eigenvalues (that is, if two or more eigenvalues are close to each other), the convergence of the inverse subspace iteration for single eigenvectors will be slow. Instead, we look at the angle between the eigenspaces [7] in two consecutive iterations

$$\theta^{(i)}(\xi) = \angle(\text{span}(u^{1,(i)}(\xi), \ldots, u^{n_e,(i)}(\xi)), \text{span}(u^{1,(i-1)}(\xi), \ldots, u^{n_e,(i-1)}(\xi))). \tag{3.22}$$

The expected value $\mathbb{E}[\theta^{(i)}]$ is taken as error indicator and is also calculated using sparse grid quadrature

$$\epsilon_\theta^{(i)} = \mathbb{E}[\theta^{(i)}] \approx \sum_{q=1}^{n_q} \theta^{(i)}(\xi^{(q)})\eta^{(q)}. \tag{3.23}$$

At each quadrature point, $\theta^{(i)}(\xi^{(q)})$ is evaluated by MATLAB function `subspace` for the largest principal angle.

4. Stochastic diffusion equation. In this section we consider the following elliptic equation with Dirichlet boundary conditions:

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = \lambda(\omega)u(x,\omega) & \text{in } \mathcal{D}\times\Omega \\ u(x,\omega) = 0 & \text{on } \partial\mathcal{D}\times\Omega, \end{cases} \tag{4.1}$$

where $\mathcal{D}$ is a two-dimensional spatial domain and Ω is a sample space. The uncertainty in the problem is introduced by the stochastic diffusion coefficient $a(x,\omega)$. Assume that $a(x,\omega)$ is bounded and strictly positive and admits a truncated KL expansion

$$a(x,\omega) = a_0(x) + \sum_{l=1}^{m} \sqrt{\beta_l} a_l(x)\xi_l(\omega), \tag{4.2}$$

where $a_0(x)$ is the mean function, $(\beta_l, a_l(x))$ is the lth eigenpair of the covariance function, and $\{\xi_l\}$ are a collection of uncorrelated random variables. The weak form is to find $(u(x,\xi), \lambda(\xi))$ such that for any $v(x) \in H_0^1(\mathcal{D})$,

$$\int_{\mathcal{D}} a(x,\xi)\nabla u(x,\xi)\cdot\nabla v(x)\mathrm{d}x = \lambda(\xi)\int_{\mathcal{D}} u(x,\xi)v(x)\mathrm{d}x \tag{4.3}$$

almost surely.

Finite element discretization in the physical domain $\mathcal{D}$ with basis functions $\{\phi_i(x)\}$ gives

$$K(\xi)u(\xi) = \lambda(\xi)Mu(\xi), \tag{4.4}$$

where $K(\xi) = \sum_{l=0}^{m} K_l\xi_l$ and

$$\begin{aligned} [K_l]_{ij} &= \int_{\mathcal{D}} \sqrt{\beta_l} a_l(x)\nabla\phi_i(x)\cdot\nabla\phi_j(x)\mathrm{d}x, \\ [M]_{ij} &= \int_{\mathcal{D}} \phi_i(x)\phi_j(x)\mathrm{d}x, \quad i,j = 1,2,\ldots,n_x, \end{aligned} \tag{4.5}$$

with $\beta_0 = 1$ and $\xi_0 \equiv 1$. The result is a generalized eigenvalue problem where the matrix M on the right-hand side is deterministic. With the Cholesky factorization $M = LL^T$, (4.4) can be converted to standard form

$$A(\xi)w(\xi) = \lambda(\xi)w(\xi), \tag{4.6}$$

where $A(\xi) = L^{-1}K(\xi)L^{-T}$, $w(\xi) = L^T u(\xi)$.

We use stochastic inverse subspace iteration to find n_e minimal eigenvalues of (4.6). As discussed in section 2, the linear systems to be solved in each iteration are in the form

$$\sum_{l=0}^{m} (G_l \otimes (L^{-1}K_l L^{-T}))\mathbf{v}^{s,(i+1)} = \mathbf{u}^{s,(i)}, \quad s = 1,2,\ldots,n_e. \tag{4.7}$$

Let $\mathbf{v}^{s,(i)} = (I \otimes L^T)\hat{\mathbf{v}}^{s,(i)}$. Then (4.7) is equivalent to

$$\sum_{l=0}^{m} (G_l \otimes K_l)\hat{\mathbf{v}}^{s,(i+1)} = (I \otimes L)\mathbf{u}^{s,(i)}. \tag{4.8}$$

4.1. Low-rank multigrid. We developed a low-rank geometric multigrid method in [9] for solving linear systems with the same structure as (4.8). The complete algorithm for solving $\mathscr{A}(X) = F$ is given in Algorithm 4.1, where $\mathscr{A}$ is a generic matrix operator and for (4.8), $\mathscr{A}(X) = \sum_{l=0}^{m} K_l X G_l^T$. All the iterates are expressed in low-rank form, and truncation operations are used to compress the ranks of the iterates. $\mathcal{T}_{\text{rel}}$ and $\mathcal{T}_{\text{abs}}$ are truncation operators with a relative tolerance ϵ_{rel} and an absolute tolerance ϵ_{abs}, respectively. In each iteration, one V-cycle is applied to the residual equation. On the coarse grids, coarse versions of $\{K_l\}$ are assembled while the matrices $\{G_l\}$ stay the same. The prolongation operator is $\mathscr{P} = I \otimes P$, where P is the same prolongation matrix as in a standard geometric multigrid solver, and the restriction operator is $\mathscr{R} = I \otimes P^T$. The smoothing operator $\mathscr{S}$ is based on a stationary iteration and is also a Kronecker product of two matrices. The grid transfer and smoothing operations do not affect the rank. For instance, for any matrix iterate in low-rank form $X^{(i)} = Y^{(i)}Z^{(i)T}$,

$$\mathscr{P}(X^{(i)}) = (PY^{(i)})(IZ^{(i)})^T. \tag{4.9}$$

On the coarsest grid ($h = h_0$), the system is solved with direct methods.

4.2. Rayleigh–Ritz refinement. It is known that in the deterministic case with a constant diffusion coefficient, (4.4) typically has repeated eigenvalues [8], for example, $\lambda^2 = \lambda^3$. The parametrized versions of these eigenvalues in the stochastic problem will be close to each other. In the deterministic setting, Rayleigh–Ritz refinement is used to accelerate the convergence of subspace iteration when some eigenvalues have nearly equal modulus and the convergence to individual eigenvectors is slow [40, 41]. Assume that a $n_x \times n_x$ Hermitian matrix S has eigendecomposition

$$S = V\Lambda V^T = \begin{pmatrix} V_1 & V_2 \end{pmatrix} \begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = V_1\Lambda_1 V_1^T + V_2\Lambda_2 V_2^T, \tag{4.10}$$

where $\Lambda = \text{diag}(\lambda^1, \lambda^2, \ldots, \lambda^{n_x})$ with eigenvalues in increasing order and $V = [V_1, V_2]$ is orthogonal. Let the column space of Q be a good approximation to that of V_1. Such an approximation is obtained from the inverse subspace iteration. The Rayleigh–Ritz procedure computes

(1) Rayleigh quotient $T = Q^T S Q$;
(2) eigendecomposition $T = W\Sigma W^T$.

Then Σ and QW represent good approximations to Λ_1 and V_1.

The stochastic inverse subspace iteration algorithm produces solutions $\{u_{\text{SG}}^s(\xi)\}$ expressed as gPC expansions as in (2.3), and sample eigenvectors are easily computed. The sample eigenvalues are generated from the stochastic Rayleigh quotient (3.15). However, in the case of poorly separated eigenvalues, the sample solutions obtained this way are not accurate enough. Experimental results that demonstrate this are given in section 4.3; see Table 4.2. Instead, we use a version of the Rayleigh–Ritz procedure to generate sample eigenvalues and eigenvectors with more accuracy. Specifically, a parametrized Rayleigh quotient $T(\xi)$ is computed using the approach of section 3.3 with

$$[T]_{st}(\xi) = u_{\text{SG}}^s(\xi)^T A(\xi) u_{\text{SG}}^t(\xi), \quad s, t = 1, 2, \ldots, n_e. \tag{4.11}$$

Algorithm 4.1: Low-rank multigrid method.

initialization: $i = 0$, $R^{(0)} = F$ in low-rank format, $r_0 = \|F\|_F$
while $r > tol * r_0$ & $i \leq maxit$ **do**
  $C^{(i)} = \text{VCYCLE}(A, 0, R^{(i)})$
  $\tilde{X}^{(i+1)} = X^{(i)} + C^{(i)}$, $X^{(i+1)} = \mathcal{T}_{\text{abs}}(\tilde{X}^{(i+1)})$
  $\tilde{R}^{(i+1)} = F - \mathscr{A}(X^{(i+1)})$, $R^{(i+1)} = \mathcal{T}_{\text{abs}}(\tilde{R}^{(i+1)})$
  $r = \|R^{(i+1)}\|_F$, $i = i + 1$
end
function $X^h = \text{VCYCLE}(A^h, X_0^h, F^h)$
  if $h == h_0$ **then**
    solve $\mathscr{A}^h(X^h) = F^h$ directly
  else
    $X^h = \text{SMOOTH}(A^h, X_0^h, F^h)$
    $\tilde{R}^h = F^h - \mathscr{A}^h(X^h)$, $R^h = \mathcal{T}_{\text{rel}}(\tilde{R}^h)$
    $R^{2h} = \mathscr{R}(R^h)$
    $C^{2h} = \text{VCYCLE}(A^{2h}, 0, R^{2h})$
    $X^h = X^h + \mathscr{P}(C^{2h})$
    $X^h = \text{SMOOTH}(A^h, X^h, F^h)$
  end
end
function $X = \text{SMOOTH}(A, X, F)$
  for ν *steps* **do**
    $\tilde{X} = X + \mathscr{S}(F - \mathscr{A}(X))$, $X = \mathcal{T}_{\text{rel}}(\tilde{X})$
  end
end

Then one can sample the matrix T and for each realization $\xi^{(r)}$ solve a small $(n_e \times n_e)$ deterministic eigenvalue problem $T(\xi^{(r)}) = W(\xi^{(r)})\Sigma(\xi^{(r)})W(\xi^{(r)})^T$ to get better approximations for the minimal eigenvalues and corresponding eigenvectors:

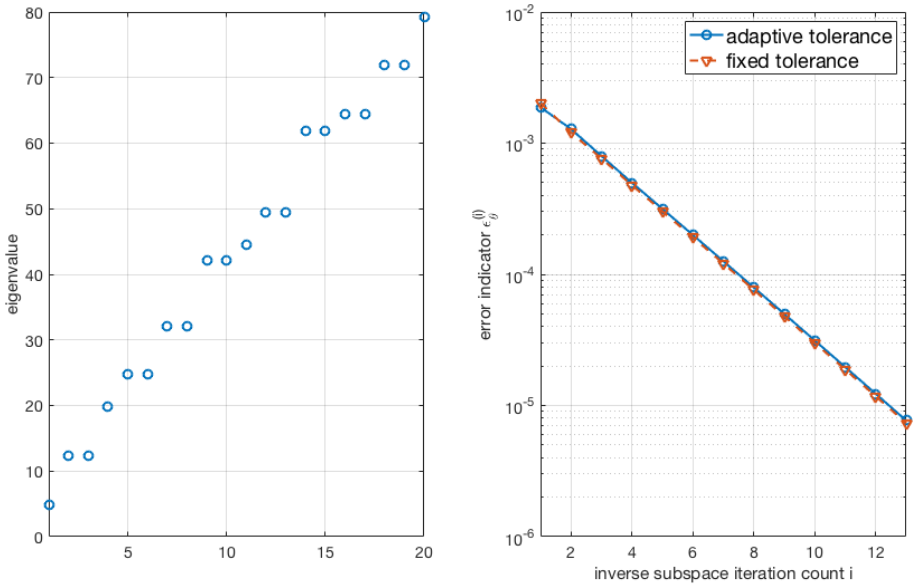
$$\begin{aligned} &\tilde{\lambda}^s_{\text{SG}}(\xi^{(r)}) = [\Sigma(\xi^{(r)})]_{ss}, \\ &\tilde{u}^s_{\text{SG}}(\xi^{(r)}) = [u^1_{\text{SG}}(\xi^{(r)}), u^2_{\text{SG}}(\xi^{(r)}), \ldots, u^{n_e}_{\text{SG}}(\xi^{(r)})][W(\xi^{(r)})]_{:,s}. \end{aligned} \tag{4.12}$$

The effectiveness of this procedure will also be demonstrated in section 4.3; see Table 4.3.

4.3. Numerical experiments. Consider a two-dimensional domain $\mathcal{D} = [-1, 1]^2$. Let the spatial discretization consist of piecewise bilinear basis functions on a uniform square mesh. The finite element matrices are assembled using the IFISS software package [34]. The number of spatial degrees of freedom is $n_x = (2/h - 1)^2$, where h is the mesh size. Define the grid level n_c such that $2/h = 2^{n_c}$. In the KL expansion (4.2), we use an exponential covariance function

$$r(x, y) = \sigma^2 \exp\left(-\frac{1}{b}\|x - y\|_1\right), \tag{4.13}$$

and $(\beta_l, a_l(x))$ is the lth eigenpair of $r(x, y)$. The correlation length b affects the decay of the eigenvalues $\{\beta_l\}$. The number of random variables m is chosen so that



(a) (b)

FIG. 4.1. (a): *Smallest* 20 *eigenvalues of the mean problem.* (b): *Reduction of the error indicator* $\epsilon_\theta^{(i)}$ *for an adaptive multigrid tolerance* (4.14) *and a fixed tolerance* $tol_{mg} = 10^{-6}$. $n_c = 6$, $b = 4.0$, $m = 11$.

$(\sum_{l=1}^{m} \beta_l)/(\sum_{l=1}^{\infty} \beta_l) \geq 95\%$. Take the standard deviation $\sigma = 0.01$, the mean function $a_0(x) \equiv 1.0$, and $\{\xi_l\}$ to be independent and uniformly distributed on $[-\sqrt{3}, \sqrt{3}]^m$. Legendre polynomials are used for gPC basis functions, whose total degree does not exceed $p = 3$. The number of gPC basis functions is $n_\xi = (m+p)!/(m!p!)$. For the quadrature rule in section 3.2, we use a Smolyak sparse grid with Clenshaw–Curtis quadrature points and grid level 3, computed from the SPINTERP toolbox [20]. For $m = 11$, the number of sparse grid points is 2069. All computations in this paper are done in MATLAB 9.4.0 (R2018a) on a MacBook with 4 GB SDRAM.

We apply low-rank stochastic inverse subspace iteration to compute three minimal eigenvalues ($n_e = 3$) and corresponding eigenvectors for (4.4). The smallest 20 eigenvalues for the mean problem $K_0 u = \lambda M u$ are plotted in Figure 4.1(a). For the stochastic problem, the three smallest eigenvalues consist of one isolated smallest eigenvalue $\lambda^1(\xi)$ and (as mentioned in the previous subsection) two eigenvalues $\lambda^2(\xi)$ and $\lambda^3(\xi)$ that have nearly equal modulus. For the inverse subspace iteration, we take $\epsilon_\theta^{(i)}$ in (3.23) as error indicator and use a stopping criterion $\epsilon_\theta^{(i)} \leq tol_{\text{isi}} = 10^{-5}$. The low-rank multigrid method of section 4.1 is used to solve the system (4.8), where damped Jacobi iteration is employed for the smoothing opeator $\mathscr{S} = \omega_s \text{diag}(\mathscr{A})^{-1} = \omega_s (I \otimes K_0^{-1})$ with weight $\omega_s = 2/3$. Two smoothing steps are applied ($\nu = 2$). We also use the idea of inexact inverse iteration methods [14, 24, 33] so that in the first few steps of subspace iteration, the systems (2.6) are solved with larger error tolerances than in later steps. Specifically, we set the multigrid tolerance as

$$tol_{\text{mg}}^{(i)} = \max\{\min\{10^{-2} * \epsilon_\theta^{(i-1)}, 10^{-3}\}, 10^{-6}\}, \tag{4.14}$$

and truncation tolerances $\epsilon_{\text{abs}}^{(i)} = 10^{-2} * tol_{\text{mg}}^{(i)}$, $\epsilon_{\text{rel}} = 10^{-2}$ [9]. This is shown to be

TABLE 4.1

Iterate ranks after the multigrid solve and numbers of multigrid steps required in the inverse subspace iteration algorithm. $n_c = 6$, $b = 4.0$, $m = 11$.

(i)		1	2	3	4	5	6	7	8	9	10	11	12	13
Rank	$\mathbf{u}^{1,(i)}$	11	22	26	32	40	44	44	46	49	49	49	49	49
	$\mathbf{u}^{2,(i)}$	17	23	25	33	41	41	41	41	41	41	41	41	41
	$\mathbf{u}^{3,(i)}$	17	25	28	37	39	40	40	40	40	40	40	40	40
it_{mg}		3	5	5	6	6	6	6	7	7	7	7	7	7

useful in reducing the computational costs while not affecting the convergence of the subspace iteration algorithm (see Figure 4.1(b)).

Table 4.1 shows the ranks of the multigrid solutions in each iteration. It indicates that all the systems solved have low-rank approximate solutions ($n_x = 3969$, $n_\xi = 364$). With the inexact solve, the solutions have much smaller ranks in the first few iterations. In the last row of Table 4.1 are the numbers of multigrid steps it_{mg} required to solve (4.8) for $s = 1$; similar numbers of multigrid steps are required for $s = 2, 3$. In addition, in Algorithm 2.1 an absolute truncation operator with $\epsilon_{\mathrm{abs}} = 10^{-8}$ is applied after the computations in (3.12) and (3.16) (both require addition of quantities represented as low-rank matrices in implementation) to compress the iterate ranks. Rayleigh–Ritz refinement discussed in section 4.2 is used to obtain good approximations to individual sample eigenpairs.

To show the accuracy of the low-rank stochastic Galerkin solutions, we compare them with reference solutions from Monte Carlo simulations. The stochastic Galerkin method produces a surrogate stochastic solution expressed with gPC basis functions that can be easily sampled. The Monte Carlo solutions are computed by the `eigs` function from MATLAB, which uses the implicitly restarted Arnoldi method to compute several minimal eigenvalues [37]. For both methods, we use the same sample values $\{\xi^{(r)}\}$ of the random variables to generate sample eigenvalues and eigenvectors. Define the relative errors

$$\epsilon_{\lambda^s} = \frac{1}{n_r}\sum_{r=1}^{n_r}\frac{|\lambda^s_{\mathrm{SG}}(\xi^{(r)}) - \lambda^s_{\mathrm{MC}}(\xi^{(r)})|}{|\lambda^s_{\mathrm{MC}}(\xi^{(r)})|},$$
$$\epsilon_{u^s} = \frac{1}{n_r}\sum_{r=1}^{n_r}\frac{\|u^s_{\mathrm{SG}}(\xi^{(r)}) - u^s_{\mathrm{MC}}(\xi^{(r)})\|_2}{\|u^s_{\mathrm{MC}}(\xi^{(r)})\|_2}, \tag{4.15}$$

where λ^s_{SG} and u^s_{SG} denote the stochastic Galerkin sample solutions (they are replaced by $\tilde{\lambda}^s_{\mathrm{SG}}$ and $\tilde{u}^s_{\mathrm{SG}}$ in (4.12) if Rayleigh–Ritz refinement is used), λ^s_{MC} and u^s_{MC} are the Monte Carlo solutions, n_r is the sample size, and $s = 1, 2, \ldots, n_e$. We use a sample size $n_r = 10000$.

We examine the accuracy for the three smallest eigenvalues obtained from inverse subspace iteration when they are computed both with and without Rayleigh–Ritz refinement. Table 4.2 shows the results (for one spatial mesh size) when Rayleigh–Ritz refinement is not used. It can be seen that (the poorly separated) eigenvalues λ^2 and λ^3 are significantly less accurate than λ^1 and that the eigenvectors u^2 and u^3 are highly inaccurate. In contrast, Table 4.3 (with results for three mesh sizes) demonstrates dramatically improved accuracy when refinement is done. In all cases, convergence takes 13 iterations.

There are several things to consider in order to assess the efficiency of the low-rank algorithm. First, note that the stochastic Galerkin method depends on two separate

TABLE 4.2
Relative differences between low-rank stochastic Galerkin solutions (without Rayleigh–Ritz refinement) and Monte Carlo solutions. $nc = 6$, $b = 4.0$, $m = 11$.

ϵ_{λ^1}	4.8752×10^{-10}	ϵ_{u^1}	2.2318×10^{-7}
ϵ_{λ^2}	5.1938×10^{-4}	ϵ_{u^2}	5.2216×10^{-1}
ϵ_{λ^3}	5.1872×10^{-4}	ϵ_{u^3}	5.2215×10^{-1}

TABLE 4.3
Relative differences between low-rank stochastic Galerkin solutions (with Rayleigh–Ritz refinement) and Monte Carlo solutions. $b = 4.0$, $m = 11$.

n_c	6	7	8
ϵ_{λ^1}	4.8753×10^{-10}	4.8789×10^{-10}	4.8777×10^{-10}
ϵ_{λ^2}	1.7339×10^{-9}	1.7996×10^{-9}	1.7856×10^{-9}
ϵ_{λ^3}	1.6481×10^{-9}	1.7122×10^{-9}	1.7189×10^{-9}
ϵ_{u^1}	1.1390×10^{-7}	1.8687×10^{-7}	3.8855×10^{-7}
ϵ_{u^2}	8.2047×10^{-6}	8.3449×10^{-6}	8.5969×10^{-6}
ϵ_{u^3}	8.2795×10^{-6}	8.4110×10^{-6}	8.6885×10^{-6}

computations: the inverse subspace iteration algorithm to compute the surrogate stochastic solution and the repeated evaluation of the surrogate solution, to be done in a simulation. (The associated costs are denoted as t_{solve} and t_{sample}, respectively.) In the parlance of reduced basis methods [43], the first part can be viewed as an offline computation and the second part as an online computation. One issue is how the costs of each of these steps for the low-rank algorithm compare with a more standard version of inverse subspace iteration that does not use low-rank constructions, which we refer to as the full-rank version. In contrast, each step of the Monte Carlo method requires the solution of a single eigenvalue problem. The cost of this computation will be much smaller than that of the offline computation required for the stochastic Galerkin method, but each step of a Monte Carlo simulation will be more costly than when a surrogate approximation is used.

Thus, the efficiency of the low-rank algorithm is demonstrated by comparison with (i) stochastic inverse subspace iteration with the full-rank stochastic Galerkin method, with the same tolerances tol_{isi} and tol_{mg}, and (ii) the Monte Carlo method. For the latter method, each deterministic eigenvalue problem is now solved by a locally optimal block preconditioned conjugate gradient (LOBPCG) method [21], preconditioned with one V-cycle of an algebraic multigrid method (AMG) of the mean matrix K_0, using a stopping tolerance 10^{-3} for the norm of the eigenvalue residual $\|K(\xi^{(r)})u_{\text{MC}}(\xi^{(r)}) - \lambda_{\text{MC}}(\xi^{(r)})Mu_{\text{MC}}(\xi^{(r)})\|_2$, chosen so that LOBPCG produces sample solutions of accuracy comparable to that obtained using the stochastic Galerkin approach.[1]

Computational costs are shown in Table 4.4. It can be seen that the low-rank approximation greatly reduces both t_{solve} and t_{sample} for the stochastic Galerkin approach, especially as the mesh size is refined. Moreover, the total time required by

[1] There are choices for the deterministic solver used for Monte Carlo. We also tried `eigs` with a mild stopping tolerance. For this (diffusion) problem, we found the costs of `eigs` and LOBPCG to be similar; however, LOBPCG is more efficient for the Stokes problem considered in section 5 below since it does not require solving linear systems associated with $BK(\xi^{(r)})^{-1}B^T$ for each sample $\xi^{(r)}$. We used LOBPCG for all cost assessments.

TABLE 4.4

Time comparison (in seconds) between stochastic Galerkin method and Monte Carlo simulation for various n_c. $b = 4.0$, $m = 11$, $n_\xi = 364$, $n_r = 10000$.

n_c		6	7	8
n_x		3969	16129	65025
low-rank SG	t_{solve}	265.63	792.06	2971.15
	t_{sample}	4.16	15.41	66.17
full-rank SG	t_{solve}	452.11	1898.85	19699.92
	t_{sample}	25.29	94.70	426.99
MC		385.39	1989.60	8897.27

TABLE 4.5

Computational times (in seconds) of low-rank Galerkin method with a CG solver for various n_c. Stopping tolerance $tol_{cg}^{(i)} = tol_{mg}^{(i)}$ and truncation tolerance $\epsilon_{rel}^{(i)} = 10^{-2} * tol_{cg}^{(i)}$ for all rank-compression computations required by CG. $b = 4.0$, $m = 11$, $n_\xi = 364$.

n_c		6	7	8
n_x		3969	16129	65025
low-rank SG	t_{solve}	153.84	604.29	3242.29

the low-rank stochastic Galerkin method is much less than that for the Monte Carlo method with a sample size $n_r = 10000$, whereas the full-rank counterpart can be more expensive than Monte Carlo. This will be discussed further in section 5 below (see Figure 5.2). Also, as a reference, we include in Table 4.5 the computational times t_{solve} of the low-rank method if instead of Algorithm 4.1, a low-rank conjugate gradient (CG) method [22] with a mean-based preconditioner $G_0 \otimes K_0 = I \otimes K_0$ is used for solving the linear systems (4.8). The results are similar to those in Table 4.4, but the timings for low-rank CG increase more rapidly than those for low-rank multigrid as the mesh is refined.

Table 4.6 shows the performance of the stochastic Galerkin approach for various n_ξ, the number of degrees of freedom in the stochastic part. As expected, the Monte Carlo method is basically unaffected by the number of random variables in the KL expansion, whereas the cost of the stochastic Galerkin method increases as the number of parameters m increases. In the cases where m is moderate, the low-rank approximation reduces the computational cost of the stochastic Galerkin approach so that the computing time becomes smaller than that for the Monte Carlo simulations. The low-rank algorithm is also effective for $m = 16$, where the full-rank stochastic Galerkin method becomes too expensive or requires too much memory.

More details on the computational costs of the low-rank stochastic Galerkin method are given in Table 4.7 for various n_x and n_ξ. The table shows the percentages of t_{solve} used for the low-rank multigrid solver (t_{mg}), the Gram–Schmidt process (t_{gs}), the convergence criterion (t_{err}), and the Rayleigh quotient (t_{rq}) in the stochastic inverse subspace iteration algorithm. It is clear that the dominant cost is that associated with solving the linear systems. As n_ξ increases, the percentages of time for the Gram–Schmidt process and the Rayleigh quotient both increase, although they are still much smaller than that for system solves.

We briefly comment on the storage costs. For an approximate solution of rank κ, the relative storage requirements of the low-rank and full-rank solutions are $(n_x + n_\xi)\kappa/(n_x n_\xi)$; for $\mathbf{u}^{1,(i)}$, as shown in Table 4.1, $\kappa \leq 49$, which is slightly less than 15%. However, some iterates within the multigrid solver, especially after the matrix-vector product computation in (4.8) (which involves a sum of $m+1$ terms), may have higher

TABLE 4.6
Time comparison (in seconds) between stochastic Galerkin method and Monte Carlo simulation for various m. $n_r = 10000$.

$m(b)$		8(5.0)	11(4.0)	16(3.0)
n_ξ		165	364	969
low-rank SG	t_{solve}	296.51	792.06	3198.15
	t_{sample}	11.56	15.41	22.56
full-rank SG	t_{solve}	642.16	1898.85	12229.23
	t_{sample}	45.77	94.70	260.40
MC		1963.53	1989.60	1809.25

(a) $n_c = 7, n_x = 16129$

$m(b)$		8(5.0)	11(4.0)	16(3.0)
n_ξ		165	364	969
low-rank SG	t_{solve}	1137.60	2971.15	10720.43
	t_{sample}	39.95	66.17	86.19
full-rank SG	t_{solve}	4673.44	19699.92	out of
	t_{sample}	194.66	426.99	memory
MC		7515.48	8897.27	8536.08

(b) $n_c = 8, n_x = 65025$

TABLE 4.7
Time consumption percentages for different parts of computations in the low-rank stochastic Galerkin method for various n_c and m.

$m(b)$		8(5.0)	11(4.0)	16(3.0)
n_ξ		165	364	969
$n_c = 7$ $n_x = 16129$	t_{mg}	79.84%	76.57%	72.46%
	t_{gs}	5.93%	8.11%	9.10%
	t_{err}	10.49%	11.68%	9.98%
	t_{rq}	1.62%	2.60%	8.02%
$n_c = 8$ $n_x = 65025$	t_{mg}	76.27%	74.54%	74.35%
	t_{gs}	6.50%	8.59%	8.84%
	t_{err}	11.30%	12.97%	12.05%
	t_{rq}	1.96%	2.05%	3.48%

ranks than n_ξ and thus require more storage than the full-rank method. We also note that the storage requirements for Monte Carlo simulation are much smaller ($O(n_x)$) since only deterministic subproblems are solved. These are also true for the Stokes problem discussed below.

5. Stochastic Stokes equation. The second example of a stochastic eigenvalue problem that we consider is used to estimate the inf-sup stability constant associated with a discrete stochastic Stokes problem. Consider the following stochastic incompressible Stokes equation in a two-dimensional domain,

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla \vec{u}(x,\omega)) + \nabla p(x,\omega) = \vec{0} & \text{in } \mathcal{D} \times \Omega \\ \nabla \cdot \vec{u}(x,\omega) = 0 & \text{in } \mathcal{D} \times \Omega, \end{cases} \tag{5.1}$$

with a Dirichlet inflow boundary condition $\vec{u}(x,\omega) = \vec{u}_D(x)$ on $\partial\mathcal{D}_D \times \Omega$ and a Neumann outflow boundary condition $a(x,\omega)\nabla\vec{u}(x,\omega)\cdot\vec{n} - p(x,\omega)\vec{n} = \vec{0}$ on $\partial\mathcal{D}_N \times \Omega$. Such

problems and more general stochastic Navier–Stokes equations have been studied in [31, 39]. As in the diffusion problem, we assume that the stochastic viscosity $a(x,\omega)$ is represented by a truncated KL expansion (4.2) with random variables $\{\xi_l\}_{l=1}^m$. The weak formulation of the problem is the following: Find $\vec{u}(x,\xi)$ and $p(x,\xi)$ satisfying

$$\begin{cases} \int_{\mathcal{D}} a(x,\xi)\nabla\vec{u}(x,\xi) : \nabla\vec{v}(x) - p(x,\xi)\nabla\cdot\vec{v}(x)\,\mathrm{d}x = 0 \\ \qquad\qquad \int_{\mathcal{D}} q(x)\nabla\cdot\vec{u}(x,\xi)\,\mathrm{d}x = 0 \end{cases} \tag{5.2}$$

almost surely for any $\vec{v}(x) \in H_0^1(\mathcal{D})^2$ (zero boundary conditions on $\partial\mathcal{D}_D$) and $q(x) \in L^2(\mathcal{D})$. Here $\nabla\vec{u} : \nabla\vec{v}$ is a componentwise scalar product ($\nabla u_{x_1}\cdot\nabla v_{x_1} + \nabla u_{x_2}\cdot\nabla v_{x_2}$ for two-dimensional (u_{x_1}, u_{x_2})). Finite element discretization with basis functions $\{\vec{\phi}_i(x)\}$ for the velocity field and $\{\varphi_k(x)\}$ for the pressure field results in a linear system in the form

$$\begin{pmatrix} K(\xi) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{u}(\xi) \\ p(\xi) \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{5.3}$$

where $K(\xi) = \sum_{l=0}^m K_l\xi_l$ and

$$\begin{aligned} [K_l]_{ij} &= \int_{\mathcal{D}} \sqrt{\beta_l}a_l(x)\nabla\vec{\phi}_i(x) : \nabla\vec{\phi}_j(x)\mathrm{d}x, \\ [B]_{kj} &= -\int_{\mathcal{D}} \varphi_k(x)\nabla\cdot\vec{\phi}_j(x)\mathrm{d}x, \end{aligned} \tag{5.4}$$

for $i,j = 1,2,\ldots,n_u$ and $k = 1,2,\ldots,n_p$. The Dirichlet boundary condition is incorporated in the right-hand side.

We are interested in the parametrized inf-sup stability constant $\gamma(\xi)$ for the discrete problem. Evaluation of the inf-sup constant for various parameter values plays an important role for a posteriori error estimation for reduced basis methods [29, 43]. For this, we exploit the fact that $\gamma(\xi)$ has an algebraic interpretation [8]

$$\gamma^2(\xi) = \min_{q(\xi)\neq 0} \frac{\langle BK(\xi)^{-1}B^T q(\xi), q(\xi)\rangle_{\mathbb{R}^{n_p}}}{\langle Mq(\xi), q(\xi)\rangle_{\mathbb{R}^{n_p}}}, \tag{5.5}$$

where M is the mass matrix with $[M]_{ij} = \int_{\mathcal{D}} \varphi_i(x)\varphi_j(x)$, $i,j = 1,2,\ldots,n_p$. Thus, finding $\gamma(\xi)$ is equivalent to finding the smallest eigenvalue of the generalized eigenvalue problem

$$BK(\xi)^{-1}B^T q(\xi) = \lambda(\xi)Mq(\xi) \tag{5.6}$$

associated with the stochastic pressure Schur complement $BK(\xi)^{-1}B^T$. This can be written in standard form as

$$L^{-1}BK(\xi)^{-1}B^T L^{-T} w(\xi) = \lambda(\xi)w(\xi), \tag{5.7}$$

where $M = LL^T$ is a Cholesky factorization and $w(\xi) = L^T q(\xi)$.

The eigenvalue problem (5.7) does not have exactly the same form as (2.1) since it involves the inverse of $K(\xi)$. If we use the stochastic inverse iteration algorithm to compute the minimal eigenvalue of (5.7), then each iteration requires solving

$$\langle L^{-1}BK^{-1}B^TL^{-T}v^{(i+1)}\psi_k\rangle = \langle u^{(i)}\psi_k\rangle, \quad k = 1, 2, \ldots, n_\xi, \tag{5.8}$$

for $v^{(i+1)}(\xi)$. We can reformulate (5.8) to take advantage of the Kronecker product structure and low-rank solvers. Let $s(\xi) = -K(\xi)^{-1}B^TL^{-T}v^{(i+1)}(\xi)$, and let $\hat{v}^{(i+1)}(\xi) = L^{-T}v^{(i+1)}(\xi)$. Then (5.8) is equivalent to the coupled system

$$\langle (Ks + B^T\hat{v}^{(i+1)})\psi_k\rangle = 0, \quad \langle Bs\psi_k\rangle = \langle -Lu^{(i)}\psi_k\rangle, \quad k = 1, 2, \ldots, n_\xi. \tag{5.9}$$

As discussed in section 2, the random vectors are expressed as gPC expansions. Thus, (5.9) can be written in Kronecker product form as a discrete Stokes system for coefficient vectors $\mathbf{s}$, $\hat{\mathbf{v}}^{(i+1)}$,

$$\begin{pmatrix} \sum_{l=0}^m (G_l \otimes K_l) & I \otimes B^T \\ I \otimes B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \hat{\mathbf{v}}^{(i+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -(I \otimes L)\mathbf{u}^{(i)} \end{pmatrix}, \tag{5.10}$$

and $\mathbf{v}^{(i+1)} = (I \otimes L^T)\hat{\mathbf{v}}^{(i+1)}$.

In addition, for the eigenvalue problem (5.7), computing the Rayleigh quotient (3.15) requires solving a linear system. In the first step of (3.15), for the matrix-vector product, one needs to compute $w(\xi) = K(\xi)^{-1}\hat{u}(\xi)$, where $\hat{u}(\xi) = B^TL^{-T}u(\xi)$. For the weak formulation, this corresponds to solving a linear system

$$\left(\sum_{l=0}^m G_l \otimes K_l\right)\mathbf{w} = \hat{\mathbf{u}}. \tag{5.11}$$

5.1. Low-rank MINRES. We discuss a low-rank iterative solver for (5.10). The system is symmetric but indefinite with a positive-definite $(1,1)$ block. A low-rank preconditioned MINRES method for solving $\mathscr{A}(X) = F$ is used and described in Algorithm 5.1. The preconditioner is block-diagonal,

$$\mathscr{M} = \begin{pmatrix} \mathscr{M}_{11} & 0 \\ 0 & \mathscr{M}_{22} \end{pmatrix}. \tag{5.12}$$

We use an approximate mean-based preconditioner [30] for the $(1,1)$ block: $\mathscr{M}_{11} = G_0 \otimes \hat{K}_0 = I \otimes \hat{K}_0$. Here, $\hat{K}_0^{-1}$ is defined by approximation of the action of K_0^{-1}, using one V-cycle of AMG. For the $(2,2)$ block, we take $\mathscr{M}_{22} = I \otimes \hat{M}$, where the action of M^{-1} is approximated by 10 steps of Chebyshev iteration [35]. As in the multigrid method, all the quantities are in low-rank format, and truncation operations are applied to compress matrix ranks. Algorithm 5.1 requires the computation of inner products of two low-rank matrices $\langle X_1, X_2\rangle_{\mathbb{R}^{n_x \times n_\xi}}$. Let $X_1 = Y_1Z_1^T$, $X_2 = Y_2Z_2^T$, with $Y_1 \in \mathbb{R}^{n_x \times \kappa_1}$, $Z_1 \in \mathbb{R}^{n_\xi \times \kappa_1}$, $Y_2 \in \mathbb{R}^{n_x \times \kappa_2}$, $Z_2 \in \mathbb{R}^{n_\xi \times \kappa_2}$. Then the inner product can be computed with a cost of $O((n_x + n_\xi + 1)\kappa_1\kappa_2)$ [22]:

$$\langle X_1, X_2\rangle = \mathrm{trace}(X_1^TX_2) = \mathrm{trace}(Z_1Y_1^TY_2Z_2^T) = \mathrm{trace}((Z_2^TZ_1)(Y_1^TY_2)). \tag{5.13}$$

We apply the low-rank MINRES method to the matricized version of (5.10) and represent the components of the solution vector, $\mathbf{s}$ and $\hat{\mathbf{v}}^{(i+1)}$, as two separate low-rank matrices S and $\hat{V}^{(i+1)}$. This representation is suitable for computing matrix-vector

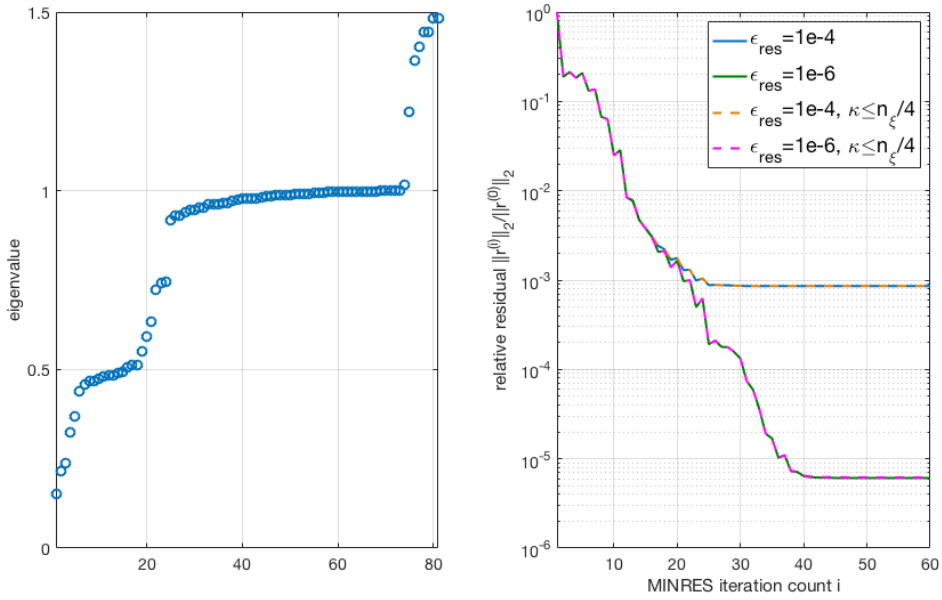
products. For instance, the first equation becomes $\sum_{l=1}^{m} K_l S G_l^T + B^T \hat{V}^{(i+1)} I^T = 0$. Other computations in Algorithm 5.1, including vector additions and truncations, are applied to each low-rank matrix component of the iterates.

Algorithm 5.1: Low-rank preconditioned MINRES method.

initialization: $V^{(0)} = 0$, $W^{(0)} = 0$, $W^{(1)} = 0$, $\gamma_0 = 0$. Choose $X^{(0)}$, compute $V^{(1)} = F - \mathscr{A}(X^{(0)})$. $P^{(1)} = \mathscr{M}^{-1}(V^{(1)})$, $\gamma_1 = \sqrt{\langle P^{(1)}, V^{(1)} \rangle}$. Set $\eta = \gamma_1$, $s_0 = s_1 = 0$, and $c_0 = c_1 = 1$.
for $j = 1, 2, \ldots$ **do**
  $P^{(j)} = P^{(j)} / \gamma_j$
  $\tilde{R}^{(j)} = \mathscr{A}(P^{(j)})$, $R^{(j)} = \mathcal{T}_{\text{rel}}(\tilde{R}^{(j)})$
  $\delta_j = \langle R^{(j)}, P^{(j)} \rangle$
  $\tilde{V}^{(j+1)} = R^{(j)} - (\delta_j / \gamma_j) V^{(j)} - (\gamma_j / \gamma_{j-1}) V^{(j-1)}$, $V^{(j+1)} = \mathcal{T}_{\text{rel}}(\tilde{V}^{(j+1)})$
  $P^{(j+1)} = \mathscr{M}^{-1}(V^{(j+1)})$
  $\gamma_{j+1} = \sqrt{\langle P^{(j+1)}, V^{(j+1)} \rangle}$
  $\alpha_0 = c_j \delta_j - c_{j-1} s_j \gamma_j$
  $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_{j+1}^2}$
  $\alpha_2 = s_j \delta_j + c_{j-1} c_j \gamma_j$
  $\alpha_3 = s_{j-1} \gamma_j$
  $c_{j+1} = \alpha_0 / \alpha_1$, $s_{j+1} = \gamma_{j+1} / \alpha_1$
  $\tilde{W}^{(j+1)} = (P^{(j)} - \alpha_3 W^{(j-1)} - \alpha_2 W^{(j)}) / \alpha_1$, $W^{(j+1)} = \mathcal{T}_{\text{rel}}(\tilde{W}^{(j+1)})$
  $\tilde{X}^{(j)} = X^{(j-1)} + c_{j+1} \eta W^{(j+1)}$, $X^{(j)} = \mathcal{T}_{\text{rel}}(\tilde{X}^{(j)})$
  $\eta = -s_{j+1} \eta$
  Check convergence
end

5.2. Numerical experiments. Consider a two-dimensional channel flow on domain $\mathcal{D} = [-1, 1]^2$ with uniform square meshes. Let $\partial\mathcal{D}_D = \{(x_1, x_2) \mid x_1 = -1, \text{ or } x_2 = 1, \text{ or } x_2 = -1\}$ and $\partial\mathcal{D}_N = \{(x_1, x_2) \mid x_1 = 1\}$. Define grid level n_c so that $2/h = 2^{n_c}$, where h is the mesh size. We use the Taylor–Hood method for finite element discretization with biquadratic basis functions $\{\vec{\phi}_i(x)\}$ for the velocity field and bilinear basis functions $\{\varphi_k(x)\}$ for the pressure field. For the velocity field the basis functions are in the form $\left\{ \begin{pmatrix} \phi_i(x) \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \phi_i(x) \end{pmatrix} \right\}$, where $\{\phi_i(x)\}$ are scalar-value biquadratic basis functions. The number of degrees of freedom in the spatial discretization is $n_x = n_u + n_p$, where $n_u = 2((2^{n_c+1} + 1)^2 - n_{\partial\mathcal{D}_D})$, $n_{\partial\mathcal{D}_D}$ is the number of Dirichlet boundary nodes, and $n_p = (2^{n_c} + 1)^2$. Assume the viscosity $a(x, \xi)$ has a KL expansion with the same specifications as in the diffusion problem. For the quadrature rule in section 3.2, we use a Smolyak sparse grid with Clenshaw–Curtis quadrature points and grid level 3.

We use the stochastic inverse iteration algorithm to find the minimal eigenvalue of (5.6). The eigenvalues of $BK_0^{-1}B^T q = \lambda M q$ are plotted in Figure 5.1(a) with $n_c = 3$. It shows that the minimal eigenvalue is isolated from the larger ones. For the inverse iteration, we take $\epsilon_\theta^{(i)}$ in (3.23) as error indicator and use a stopping criterion $\epsilon_\theta^{(i)} \leq tol_{\text{isi}} = 10^{-5}$. The error tolerance for the MINRES solver $tol_{\text{minres}}^{(i)}$ is set as in (4.14). Figure 5.1(b) shows the convergence of the low-rank MINRES method for different relative truncation tolerances ϵ_{rel}. It indicates the accuracy



(a) (b)

FIG. 5.1. (a): *Eigenvalues of* $BK_0^{-1}B^T q = \lambda Mq$. $n_c = 3$. (b): *Reduction of the relative residual for the low-rank MINRES method with various truncation criteria. Solid lines: relative tolerance* ϵ_{rel}*; dashed lines: relative tolerance* ϵ_{rel} *with rank* $\kappa \le n_\xi/4$. $n_c = 4$, $b = 4.0$, $m = 11$.

TABLE 5.1
Iterate ranks after the MINRES solve and numbers of MINRES steps required in the inverse iteration algorithm. $n_c = 4$, $b = 4.0$, $m = 11$.

(i)		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rank	$\mathbf{s}$	4	12	13	13	17	18	21	24	28	31	30	31	30	30	31	30
	$\hat{\mathbf{v}}^{(i)}$	7	12	13	16	19	25	31	38	44	49	49	49	50	49	50	50
it_{minres}		22	35	35	37	37	39	39	41	42	43	43	43	43	43	43	43

that MINRES can achieve is related to ϵ_{rel}. In the numerical experiments we use $\epsilon_{\text{rel}}^{(i)} = 10^{-1} * tol_{\text{minres}}^{(i)}$. In addition, we have observed that in many cases the truncations in lines 4, 6, and 14 of Algorithm 5.1 produce relatively high ranks, which increases the computational cost. To handle this, we impose a bound on the ranks κ of the outputs of these truncation operators such that $\kappa \le n_\xi/4$ (in general $n_x \ge n_\xi$). It is shown in Figure 5.1(b) that the convergence of low-rank MINRES is unaffected by this strategy.

Table 5.1 shows the ranks of the MINRES solutions $\mathbf{s}$ and $\hat{\mathbf{v}}^{(i)}$ in (5.10) and numbers of MINRES steps it_{minres} required in each iteration. The solution matrices S and $\hat{V}^{(i)}$ have sizes $n_u \times n_\xi$ and $n_p \times n_\xi$ (for $n_c = 4$ and $m = 11$, $n_u = 1984$, $n_p = 289$, $n_\xi = 364$), whereas their respective ranks are no larger than 31 and 50. In the Rayleigh quotient computation, the system (5.11) is solved by a low-rank conjugate gradient method [22] with a relative residual smaller than 10^{-8}.

As in the diffusion problem, we show the accuracy of the low-rank stochastic Galerkin approach by comparing the results with the reference solutions from Monte Carlo simulations using `eigs`. Let $m = 11$, $p = 3$, $n_\xi = 364$. We use a sample size $n_r = 1000$. Table 5.2 shows the accuracy of the stochastic Galerkin solutions where

TABLE 5.2

Relative difference between stochastic Galerkin solutions and Monte Carlo solutions. $b = 4.0$, $m = 11$, $n_\xi = 364$.

n_c	4	5	6
ϵ_{λ^1}	5.8903×10^{-9}	6.8722×10^{-9}	7.6883×10^{-9}
ϵ_{u^1}	4.4363×10^{-5}	5.1253×10^{-5}	5.3235×10^{-5}

TABLE 5.3

Time comparison (in seconds) between stochastic Galerkin method and Monte Carlo simulation for various n_c. $n_r = 1000$.

n_c		4	5	6
n_p		289	1089	4225
n_x		2273	9153	36737
low-rank SG	t_{solve}	269.84	1006.36	4382.16
	t_{sample}	0.11	0.13	0.25
full-rank SG	t_{solve}	324.74	1264.05	6272.26
	t_{sample}	0.11	0.22	0.97
MC		122.58	417.62	1594.47

(a) $b = 4.0$, $m = 11$, $n_\xi = 364$

n_c		4	5	6
n_p		289	1089	4225
n_x		2273	9153	36737
low-rank SG	t_{solve}	79.48	323.40	1557.33
	t_{sample}	0.06	0.07	0.09
full-rank SG	t_{solve}	132.63	538.44	2636.93
	t_{sample}	0.07	0.07	0.40
MC		128.16	411.34	1539.16

(b) $b = 5.0$, $m = 8$, $n_\xi = 165$

ϵ_{λ^1} and ϵ_{u^1} are defined in (4.15) (no Rayleigh–Ritz procedure is used here). In all cases, convergence of the inverse iteration takes 16–18 steps.

As we did for the diffusion problem, we assess the the efficiency of the low-rank stochastic Galerkin method by comparison with the full-rank method and Monte Carlo simulation. For the latter, we use an LOBPCG solver preconditioned with the pressure mass matrix M, and the action of M^{-1} is again approximated by 10 steps of Chebyshev iteration. In this case, a stopping tolerance of 10^{-6} is used for LOBPCG to produce solutions with accuracy comparable to those obtained using the stochastic Galerkin approach. Table 5.3 shows the comparative costs of these methods when 1000 samples are used in a simulation. It is clear that the low-rank stochastic Galerkin method is more efficient than its full-rank counterpart, and the simulations using the surrogate solution obtained from the stochastic Galerkin approach are very cheap compared with Monte Carlo simulation. If we take the total cost of the stochastic Galerkin method to be the sum of t_{solve} and t_{sample}, then the comparison depends on the number of samples used, and in this measure, for 1000 samples it is cheaper to perform Monte Carlo simulation. This issue is explored in more detail in Figure 5.2, which interpolates the costs from timings using 1000, 5000, and 10000 samples and shows "crossover" sample sizes for which the stochastic Galerkin methods will be more efficient than Monte Carlo methods; these are approximately 2500 for the low-rank version and 4000 for the full-rank one.

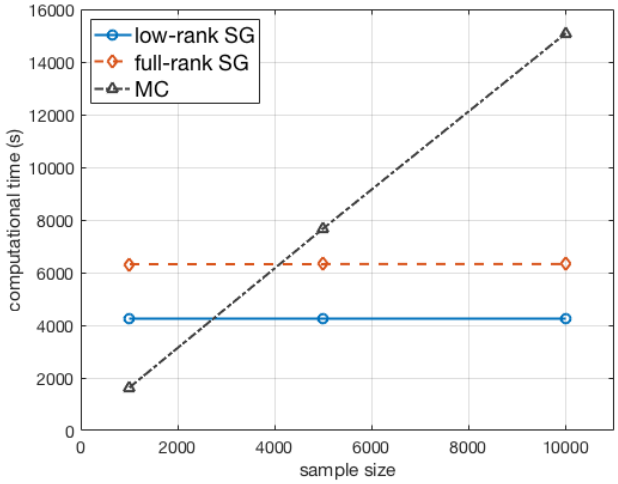


FIG. 5.2. *Computational time required by the low-rank stochastic Galerkin method, the full-rank stochastic Galerkin method, and the Monte Carlo method to generate large numbers of sample solutions.* $nc = 6$, $b = 4.0$, $m = 11$, $n_r = 1000, 5000, 10000$.

6. Summary. We studied low-rank solution methods for stochastic eigenvalue problems. The stochastic Galerkin approach was used to compute surrogate approximations to the minimal eigenvalues and corresponding eigenvectors, which are stochastic functions with gPC expansions. We introduced low-rank approximations to enhance efficiency of the stochastic inverse subspace iteration algorithm. Two detailed benchmark problems, the stochastic diffusion problem and an operator associated with a discrete stochastic Stokes equation, were considered for illustrating the effectiveness of the proposed low-rank algorithm. It was confirmed in the numerical experiments that the low-rank solution method produces accurate results with much less computing time, making the stochastic Galerkin method more competitive compared with the sample-based Monte Carlo approach.

Acknowledgment. The authors thank the anonymous referees for constructive comments.

REFERENCES

[1] R. ANDREEV AND C. SCHWAB, *Sparse tensor approximation of parametric eigenvalue problems*, in Numerical Analysis of Multiscale Problems, I. G. Graham, T. Y. Hou, O. Lakkis, and R. Scheichl, eds., Springer, Berlin, 2012, pp. 203–241.

[2] J. BALLANI AND L. GRASEDYCK, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43.

[3] P. BENNER, S. DOLGOV, A. ONWUNTA, AND M. STOLL, *Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data*, Comput. Methods Appl. Mech. Engrg., 304 (2016), pp. 26–54.

[4] P. BENNER, A. ONWUNTA, AND M. STOLL, *Low-rank solution of unsteady diffusion equations with stochastic coefficients*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 622–649.

[5] P. BENNER, A. ONWUNTA, AND M. STOLL, *A low-rank inexact Newton–Krylov method for stochastic eigenvalue problems*, Comput. Methods Appl. Math., 19 (2019), pp. 5–22.

[6] P. BENNER, Y. QIU, AND M. STOLL, *Low-rank eigenvector compression of posterior covariance matrices for linear Gaussian inverse problems*, SIAM/ASA J. Uncertain. Quantif., 6 (2018), pp. 965–989.

[7] Å. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Math. Comp., 27 (1973), pp. 579–594.
[8] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, 2nd ed., Oxford University Press, Oxford, 2014.
[9] H. C. Elman and T. Su, *A low-rank multigrid method for the stochastic steady-state diffusion problem*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 492–509.
[10] O. G. Ernst and E. Ullmann, *Stochastic Galerkin matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1848–1872.
[11] I. Fumagalli, A. Manzoni, N. Parolini, and M. Verani, *Reduced basis approximation and a posteriori error estimates for parametrized elliptic eigenvalue problems*, ESAIM Math. Model. Numer. Anal., 50 (2016), pp. 1857–1885.
[12] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numer. Algorithms, 18 (1998), pp. 209–232.
[13] R. Ghanem and D. Ghosh, *Efficient characterization of the random eigenvalue problem in a polynomial chaos decomposition*, Internat. J. Numer. Methods Engrg., 72 (2007), pp. 486–504.
[14] G. H. Golub and Q. Ye, *Inexact inverse iteration for generalized eigenvalue problems*, BIT, 40 (2000), pp. 671–684.
[15] W. Hackbusch, *Solution of linear systems in high spatial dimensions*, Comput. Vis. Sci., 17 (2015), pp. 111–118.
[16] H. Hakula, V. Kaarnioja, and M. Laaksonen, *Approximate methods for stochastic eigenvalue problems*, Appl. Math. Comput., 267 (2015), pp. 664–681.
[17] H. Hakula and M. Laaksonen, *Asymptotic convergence of spectral inverse iterations for stochastic eigenvalue problems*, Numer. Math., (2019), pp. 1–33.
[18] T. Horger, B. Wohlmuth, and T. Dickopf, *Simultaneous reduced basis approximation of parameterized elliptic eigenvalue problems*, ESAIM Math. Model. Numer. Anal., 51 (2017), pp. 443–465.
[19] D. B. P. Huynh, G. Rozza, S. Sen, and A. T. Patera, *A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants*, Comptes Rendus Math., 345 (2007), pp. 473–478.
[20] A. Klimke and B. Wohlmuth, *Algorithm* 847: *SPINTERP: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB*, ACM Trans. Math. Software, 31 (2005), pp. 561–579.
[21] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
[22] 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.
[23] D. Kressner and C. Tobler, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Comput. Methods Appl. Math., 11 (2011), pp. 363–381.
[24] Y.-L. Lai, K.-Y. Lin, and W.-W. Lin, *An inexact inverse iteration for large sparse eigenvalue problems*, Numer. Linear Algebra Appl., 4 (1997), pp. 425–437.
[25] K. Lee and H. C. Elman, *A preconditioned low-rank projection method with a rank-reduction scheme for stochastic partial differential equations*, SIAM J. Sci. Comput., 39 (2017), pp. 828–850.
[26] M. Loève, *Probability Theory*, Van Nostrand, New York, 1960.
[27] L. Machiels, Y. Maday, I. B. Oliveira, A. T. Patera, and D. V. Rovas, *Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems*, C. R. Math. Acad. Sci. Paris, 331 (2000), pp. 153–158.
[28] H. Meidani and R. Ghanem, *Spectral power iterations for the random eigenvalue problem*, AIAA J., 52 (2014), pp. 912–925.
[29] N. Ngoc Cuong, K. Veroy, and A. T. Patera, *Certified real-time solution of parametrized partial differential equations*, in Handbook of Materials Modeling, S. Yip, ed., Springer, Dordrecht, the Netherlands, 2005, pp. 1529–1564.
[30] 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.
[31] C. E. Powell and D. J. Silvester, *Preconditioning steady-state Navier–Stokes equations with random data*, SIAM J. Sci. Comput., 34 (2012), pp. A2482–A2506.
[32] H. J. Pradlwarter, G. I. Schuëller, and G. S. Szekely, *Random eigenvalue problems for large systems*, Comput. Struct., 80 (2002), pp. 2415–2424.
[33] M. Robbé, M. Sadkane, and A. Spence, *Inexact inverse subspace iteration with preconditioning applied to non-Hermitian eigenvalue problems*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 92–113.

[34] D. Silvester, H. Elman, and A. Ramage, *Incompressible Flow and Iterative Solver Software (IFISS)*, Version 3.4, 2015, http://www.manchester.ac.uk/ifiss.
[35] D. J. Silvester and V. Simoncini, *An optimal iterative solver for symmetric indefinite systems stemming from mixed approximation*, ACM Trans. Math. Software, 37 (2011), p. 42.
[36] P. Sirković and D. Kressner, *Subspace acceleration for large-scale parameter-dependent Hermitian eigenproblems*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 695–718.
[37] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[38] B. Sousedík and H. C. Elman, *Inverse subspace iteration for spectral stochastic finite element methods*, SIAM/ASA J. Uncertain. Quantif., 4 (2016), pp. 163–189.
[39] B. Sousedík and H. C. Elman, *Stochastic Galerkin methods for the steady-state Navier–Stokes equations*, J. Comput. Phys., 316 (2016), pp. 435–452.
[40] G. W. Stewart, *Accelerating the orthogonal iteration for the eigenvectors of a Hermitian matrix*, Numer. Math., 13 (1969), pp. 362–376.
[41] G. W. Stewart, *Matrix Algorithms: Volume* II: *Eigensystems*, SIAM, Philadelphia, 2001.
[42] C. V. Verhoosel, M. A. Gutiérrez, and S. J. Hulshoff, *Iterative solution of the random eigenvalue problem with application to spectral stochastic finite element systems*, Internat. J. Numer. Methods Engrg., 68 (2006), pp. 401–424.
[43] K. Veroy and A. T. Patera, *Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: Rigorous reduced-basis a posteriori error bounds*, Internat. J. Numer. Methods Fluids, 47 (2005), pp. 773–788.
[44] D. Xiu and G. E. Karniadakis, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, J. Comput. Phys., 187 (2003), pp. 137–167.