SIAM/ASA J. UNCERTAINTY QUANTIFICATION
Vol. 6, No. 4, pp. 1744–1776

Inexact Methods for Symmetric Stochastic Eigenvalue Problems*

Kookjin Lee† and Bedřich Sousedík‡

Abstract. We study two inexact methods for solutions of random eigenvalue problems in the context of spectral stochastic finite elements. In particular, given a parameter-dependent, symmetric matrix operator, the methods solve for eigenvalues and eigenvectors represented using polynomial chaos expansions. Both methods are based on the stochastic Galerkin formulation of the eigenvalue problem and they exploit its Kronecker-product structure. The first method is an inexact variant of the stochastic inverse subspace iteration [B. Sousedík and H. C. Elman, *SIAM/ASA J. Uncertain. Quantif.*, 4 (2016), pp. 163–189]. The second method is based on an inexact variant of the Newton iteration. In both cases, the problems are formulated so that the associated stochastic Galerkin matrices are symmetric, and the corresponding linear problems are solved using preconditioned Krylov subspace methods with several novel hierarchical preconditioners. The accuracy of the methods is compared with that of Monte Carlo and stochastic collocation, and the effectiveness of the methods is illustrated by numerical experiments.

Key words. eigenvalues, subspace iteration, inverse iteration, Newton iteration, stochastic spectral finite element method

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

DOI. 10.1137/18M1176026

1. Introduction.

Eigenvalue analysis is important in a number of applications, for example, in modeling of vibrations of mechanical structures, neutron transport criticality computations, or stability of dynamical systems, to name a few. The behavior of the underlying mathematical models depends on proper choice of parameters entering the model through coefficients, boundary conditions, or forces. However, in practice the exact values of these parameters are not known and they are treated as random processes. The uncertainty is translated by discretization into the matrix operators and subsequently into eigenvalues and

*Received by the editors March 19, 2018; accepted for publication (in revised form) October 11, 2018; published electronically December 18, 2018.

http://www.siam.org/journals/juq/6-4/M117602.html

Funding: This work is based upon work 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 DMS1521563. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the U.S. Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

†This work was performed while pursuing a Ph.D. in the Department of Computer Science at the University of Maryland, College Park, MD 20742. Current affiliation: Extreme-scale Data Science and Analytics Department, Sandia National Laboratories, Livermore, CA 94550 (koolee@sandia.gov).

‡Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21250 (sousedik@umbc.edu).

eigenvectors. The standard techniques to solve this problems include Monte Carlo methods [2, 25, 31], which are robust but relatively slow, and perturbation methods [15, 17, 32, 38], which are limited to models with low variability of uncertainty.

In this study, we use spectral stochastic finite element methods (SSFEM) [7, 18, 20, 43] for the solution of symmetric eigenvalue problems. The assumption in these methods is that the parametric uncertainty is described in terms of polynomials of random variables, and they compute solutions that are also polynomials in the same random variables in the so-called generalized polynomial chaos (gPC) framework [7, 44]. There are two main approaches: stochastic collocation and stochastic Galerkin methods. The first approach is based on sampling, so the problem is translated into a set of independent deterministic systems; the second one is based on stochastic Galerkin projection and the problem is translated into one large coupled deterministic system. While the SSFEM methods have become quite popular for solving stochastic partial differential equations, the literature addressing eigenvalue problems is relatively limited. The stochastic inverse iteration in the context of the stochastic Galerkin framework was proposed by Verhoosel, Gutiérrez, and Hulshoff [37]. Meidani and Ghanem [22, 23] formulated a stochastic subspace iteration using a stochastic version of the modified Gram–Schmidt algorithm. Sousedík and Elman [33] introduced the stochastic inverse subspace iteration (SISI) by combining the two techniques, they showed that deflation of the mean matrix can be used to compute expansions of the interior eigenvalues, and they also showed that the stochastic Rayleigh quotient alone provides a good approximation of an eigenvalue expansion; see also [3, 4, 27] for closely related methods. The authors of [23, 33] used a quadrature-based normalization of eigenvectors. Normalization based on a solution of a small nonlinear problem was proposed by Hakula, Kaarnioja, and Laaksonen [12], and Hakula and Laaksonen [13] also provided an asymptotic convergence theory for the stochastic iteration. In an alternative approach, Ghanem and Ghosh [6, 9] proposed two numerical schemes—one based on Newton iteration and another based on an optimization problem (see also [8, 10]). Most recently, Benner, Onwunta, and Stoll [1] formulated an inexact low-rank Newton–Krylov method, in which the stochastic Galerkin linear systems are solved using the BiCGStab method with a variant of mean-based preconditioner. In alternative approaches, Pascual and Adhikari [28] introduced several hybrid perturbation-polynomial chaos methods, and Williams [40, 41, 42] presented a method that avoids the nonlinear terms in the conventional method of stochastic eigenvalue calculation but introduces an additional independent variable.

We formulate two inexact methods for symmetric eigenvalue problems formulated in the SSFEM framework and based on the stochastic Galerkin formulation. The first method is an inexact variant of SISI from [33], in which the linear stochastic Galerkin systems are solved using the conjugate gradient method with the truncated hierarchical preconditioner [35] (see also [36]). The second method is an inexact variant of the Newton iteration from [6], in which the linear stochastic Galerkin systems are solved using preconditioned MINRES and GMRES. The methods are derived using the Kronecker-product formulation and we also comment on the so-called matricized format. The formulation of the Newton's method is closely related to that of [1]; however, we consider general parametrization of stochastic coefficients, the Jacobian matrices are symmetrized, and we propose a class of hierarchical preconditioners, which can be viewed as extensions of the hierarchical preconditioners used for the first method. We also note that we have recently successfully combined an inexact Newton–Krylov method

with the stochastic Galerkin framework in a different context [19, 34]. The performance of both methods is illustrated by numerical experiments, and the results are compared to those of Monte Carlo and stochastic collocation methods.

The paper is organized as follows. In section 2 we introduce the stochastic eigenvalue problem, in section 2.1 we recall the solution techniques using sampling methods (Monte Carlo and stochastic collocation), in section 2.2 we introduce the stochastic Galerkin formulation, in section 3 we formulate the inverse subspace iteration and in section 4 the Newton iteration, in section 5.1 we report the results of numerical experiments, and in section 6 we summarize our work. In Appendices A and B we describe algorithmic details, and in Appendix C we discuss the computational cost.

2. Stochastic eigenvalue problem. Let D be a bounded physical domain, and let $(\Omega, \mathcal{F}, \mathcal{P})$ be a complete probability space, that is, Ω is a sample space with a σ-algebra $\mathcal{F}$ and a probability measure $\mathcal{P}$. We assume that the randomness in the mathematical model is induced by a vector $\xi : \Omega \mapsto \Gamma \subset \mathbb{R}^{m_\xi}$ of independent, identically distributed random variables $\xi_1(\omega), \ldots, \xi_{m_\xi}(\omega)$, where $\omega \in \Omega$. Let $\mathcal{B}(\Gamma)$ denote the Borel σ-algebra on Γ induced by ξ and let μ denote the induced measure. The expected value of the product of measurable functions on Γ determines a Hilbert space $T_\Gamma \equiv L^2(\Gamma, \mathcal{B}(\Gamma), \mu)$ with inner product

$$\langle u, v \rangle = \mathbb{E}[uv] = \int_\Gamma u(\xi) v(\xi)\, \mu(\xi) d\xi, \tag{2.1}$$

where the symbol $\mathbb{E}$ denotes the mathematical expectation.

In computations we will use a finite-dimensional subspace $T_p \subset T_\Gamma$ spanned by a set of multivariate polynomials $\{\psi_\ell(\xi)\}$ that are orthonormal with respect to the density function μ, that is, $\mathbb{E}[\psi_k \psi_\ell] = \delta_{k\ell}$, where $\delta_{k\ell}$ is the Kronecker delta, and ψ_0 is constant. This will be referred to as the gPC basis [44]. The dimension of the space T_p depends on the polynomial degree. For polynomials of total degree p, the dimension is $n_\xi = \binom{m_\xi + p}{p}$. We suppose we are given a symmetric matrix-valued random variable $A(x, \xi)$ represented as

$$A(x, \xi) = \sum_{\ell=1}^{n_a} A_\ell(x) \psi_\ell(\xi), \tag{2.2}$$

where each A_ℓ is a deterministic matrix of size $n_x \times n_x$ with size determined by the discretization of the physical domain, and A_1 is the mean value matrix, that is, $A_1 = \mathbb{E}[A(x, \cdot)]$. The representation (2.2) is obtained from either the Karhunen–Loève expansion or, more generally, a stochastic expansion of an underlying random process.

We are interested in a solution of the following stochastic eigenvalue problem: find a set of stochastic eigenvalues λ^s and corresponding eigenvectors u^s, $s = 1, \ldots, n_s$, which almost surely satisfy the equation

$$A(x, \xi) u^s(x, \xi) = \lambda^s(\xi) u^s(x, \xi) \qquad \forall x \in D, \tag{2.3}$$

where $\lambda^s(\xi) \in \mathbb{R}$ and $u^s(x, \xi) \in \mathbb{R}^{n_x}$, along with a normalization condition

$$\langle u^s(x, \xi), u^s(x, \xi) \rangle_{\mathbb{R}} = 1, \tag{2.4}$$

where $\langle \cdot, \cdot \rangle_{\mathbb{R}}$ denotes the inner product of two vectors.

We will search for expansions of eigenpairs (λ^s, u^s), $s = 1, \ldots, n_s$, in the form

$$\lambda^s(\xi) = \sum_{k=1}^{n_\xi} \lambda_k^s \psi_k(\xi), \qquad u^s(x,\xi) = \sum_{k=1}^{n_\xi} u_k^s \psi_k(\xi), \tag{2.5}$$

where $\lambda_k^s \in \mathbb{R}$ and $u_k^s \in \mathbb{R}^{n_x}$ are the coefficients corresponding to the basis $\{\psi_k\}$. Equivalently to (2.5), using the symbol $\otimes$ for the Kronecker product, we write

$$\lambda^s(\xi) = \Psi(\xi)^T \bar{\lambda}^s, \qquad u^s(x,\xi) = (\Psi(\xi)^T \otimes I_{n_x})\bar{u}^s, \tag{2.6}$$

where $\Psi(\xi) = [\psi_1(\xi), \ldots, \psi_{n_\xi}(\xi)]^T$, $\bar{\lambda}^s = [\lambda_1^s, \ldots, \lambda_{n_\xi}^s]^T$, and $\bar{u}^s = [(u_1^s)^T, \ldots, (u_{n_\xi}^s)^T]^T$.

Remark 2.1. One can in general consider a different number of terms in the two expansions (2.5). However, since the numerical experiments in [33] and also in the present work indicate virtually no effect when the number of terms in eigenvalue expansion is larger than in the eigenvector expansion, we consider here the same number of terms in both expansions; see also Remark 3.1.

2.1. Sampling methods. Both Monte Carlo and stochastic collocation methods are based on sampling. The coefficients are defined by a discrete projection

$$\lambda_k^s = \langle \lambda^s, \psi_k \rangle, \quad k = 1, \ldots, n_\xi, \qquad u_k^s = \langle u^s, \psi_k \rangle, \quad k = 1, \ldots, n_\xi. \tag{2.7}$$

The evaluations of coefficients in (2.7) entail solving a set of independent deterministic eigenvalue problems at a set of sample points $\xi^{(q)}$, $q = 1, \ldots, n_{MC}$ or n_q,

$$A(\xi^{(q)})u^s(\xi^{(q)}) = \lambda^s\left(\xi^{(q)}\right) u^s\left(\xi^{(q)}\right), \qquad s = 1, \ldots, n_s.$$

In the Monte Carlo method, the sample points $\xi^{(q)}$, $q = 1, \ldots, n_{MC}$, are generated randomly following the distribution of the random variables ξ_i, $i = 1, \ldots, m_\xi$, and moments of solution are computed by ensemble averaging. In addition, the coefficients in (2.5) can be computed as[1]

$$\lambda_k^s = \frac{1}{n_{MC}} \sum_{q=1}^{n_{MC}} \lambda^s(\xi^{(q)})\psi_k\left(\xi^{(q)}\right), \qquad u_{mk}^s = \frac{1}{n_{MC}} \sum_{q=1}^{n_{MC}} u^s(x_m, \xi^{(q)})\psi_k(\xi^{(q)}),$$

where u_{mk}^s is the mth element of u_k^s. For stochastic collocation, which is used here in the form of the so-called nonintrusive stochastic Galerkin method, the sample points $\xi^{(q)}$, $q = 1, \ldots, n_q$, consist of a predetermined set of *collocation points*, and the coefficients λ_k^s and u_k^s in expansions (2.5) are determined by evaluating (2.7) in the sense of (2.1) using numerical quadrature

$$\lambda_k^s = \sum_{q=1}^{n_q} \lambda^s(\xi^{(q)})\psi_k(\xi^{(q)})w^{(q)}, \qquad u_{mk}^s = \sum_{q=1}^{n_q} u^s(x_m, \xi^{(q)})\psi_k(\xi^{(q)})w^{(q)}, \tag{2.8}$$

[1] In numerical experiments we avoid projections on the gPC and work with the sampled quantities.

where $\xi^{(q)}$ are the quadrature (collocation) points and $w^{(q)}$ are quadrature weights. We refer, e.g., to [18] for a discussion of quadrature rules. Details of the rules we used in numerical experiments are discussed in section 5.1.

2.2. Stochastic Galerkin formulation. The main contribution of this paper is the development of two inexact methods based on the stochastic Galerkin formulation of eigenvalue problem (2.3)–(2.4). The formulation entails a projection

$$\langle Au^s, \psi_k \rangle = \langle \lambda^s u^s, \psi_k \rangle, \qquad k = 1, \dots, n_\xi, \quad s = 1, \dots, n_s, \tag{2.9}$$

$$\left\langle u^{sT} u^s, \psi_k \right\rangle = \delta_{k1}, \qquad k = 1, \dots, n_\xi, \quad s = 1, \dots, n_s. \tag{2.10}$$

Let us introduce the notation

$$[H_\ell]_{kj} = h_{\ell,kj}, \quad h_{\ell,kj} \equiv \mathbb{E}\left[\psi_\ell \psi_k \psi_j\right], \qquad \ell = 1, \dots, n_a, \quad j, k = 1, \dots, n_\xi. \tag{2.11}$$

Substituting (2.2) and (2.5) into (2.9)–(2.10) yields a nonlinear system,

$$\left(\sum_{\ell=1}^{n_a} H_\ell \otimes A_\ell \right) \overline{u}^s = \left(\sum_{i=1}^{n_\xi} H_i \otimes \lambda_i^s I_{n_x} \right) \overline{u}^s, \quad s = 1, \dots, n_s, \tag{2.12}$$

$$\sum_{j=1}^{n_\xi} \sum_{i=1}^{n_\xi} \left[H_k \circ \left\langle u_i^s, u_j^s \right\rangle_{\mathbb{R}} \right]_{ij} = \delta_{k1}, \qquad k = 1, \dots, n_\xi, \quad s = 1, \dots, n_s, \tag{2.13}$$

where the symbol $\circ$ is the Hadamard product; see, e.g., [14, Chapter 5]. An equivalent formulation of (2.12)–(2.13) is obtained as follows. Substituting (2.6) into (2.3)–(2.4) and rearranging, we get

$$(\Psi^T \otimes A)\bar{u}^s = ((\bar{\lambda}^s)^T \Psi \Psi^T \otimes I_{n_x})\bar{u}^s,$$
$$\bar{u}^{sT}(\Psi(\xi)\Psi(\xi)^T \otimes I_{n_x})\bar{u}^s = 1,$$

and employing Galerkin projection (2.9)–(2.10) yields the equivalent formulation

$$\mathbb{E}[\Psi \Psi^T \otimes A]\bar{u}^s = \mathbb{E}[((\bar{\lambda}^s)^T \Psi)\Psi \Psi^T \otimes I_{n_x})]\bar{u}^s, \tag{2.14}$$

$$\mathbb{E}[\Psi \otimes (\bar{u}^{sT}(\Psi \Psi^T \otimes I_{n_x})\bar{u}^s)] = \mathbb{E}[\Psi \otimes 1]. \tag{2.15}$$

Finally, we note that the methods can be equivalently formulated in the so-called matricized format, which can also simplify the implementation. To this end, we make use of isomorphism between $\mathbb{R}^{n_x n_\xi}$ and $\mathbb{R}^{n_x \times n_\xi}$ determined by the operators vec and mat: $\bar{u}^s = \mathrm{vec}(\bar{U}^s)$, $\bar{U}^s = \mathrm{mat}(\bar{u}^s)$, where $\bar{u}^s \in \mathbb{R}^{n_x n_\xi}$, $\bar{U}^s \in \mathbb{R}^{n_x \times n_\xi}$ and the upper-/lowercase notation is assumed throughout the paper, so $\bar{R}^s = \mathrm{mat}(\bar{r}^s)$, etc. Specifically, we define the *matricized* coefficients of the eigenvector expansion

$$\bar{U}^s = \mathrm{mat}(\bar{u}^s) = \left[u_1^s, u_2^s, \dots, u_{n_\xi}^s \right] \in \mathbb{R}^{n_x \times n_\xi}, \tag{2.16}$$

where the column k contains the coefficients associated with the basis function ψ_k.

In the rest of the paper we explore two methods for solving the eigenvalue problem (2.12)–(2.13), resp., (2.14)–(2.15): the first is based on inverse subspace iteration (section 3), and the second is based on Newton iteration (section 4).

3. Inexact stochastic inverse subspace iteration. We formulate an inexact variant of the inverse subspace iteration from [33] for the solution of (2.12)–(2.13). Stochastic inverse iteration was formulated in [37] for the case when a stochastic expansion of a single eigenvalue is sought. It was suggested in [33] that the matrix A_1 can be deflated, rather than applying a shift, to find an expansion of an interior eigenvalue, and a stochastic version of a modified Gram–Schmidt process [23] can be applied if more eigenvalues are of interest. In this section, we formulate an inexact variant of the SISI [33, Algorithm 3.2], whereby the linear systems (3.3) are solved only approximately using the preconditioned conjugate gradient method (PCG). The method is formulated as Algorithm 1. We now describe its components in detail, and for simplicity we drop the superscript $^{(n)}$ in the description.

Algorithm 1 Inexact stochastic inverse subspace iteration.

1: Find the n_s smallest eigenpairs of

$$A_1 w^s = \mu^s w^s, \qquad s = 1, \dots, n_s. \tag{3.1}$$

2: **if** $\mu^1 = \min(\mu^s) > 0$, set $\rho = 0$, **else** shift $A_1 = A_1 + \rho I_{n_x}$, where $\rho > |\mu^1|$. **end if**

3: Initialize

$$u_1^{s,(0)} = w^s, \quad u_i^{s,(0)} = 0, \qquad s = 1, \dots, n_s, \quad i = 2, \dots, n_\xi. \tag{3.2}$$

4: **for** $n = 0, 1, 2, \dots$ **do**

5: Use conjugate gradients with preconditioner from Algorithm 2 or 3 to solve

$$\left(\sum_{\ell=1}^{n_a} H_\ell \otimes A_\ell \right) \overline{v}^{s,(n)} = \overline{u}^{s,(n)}, \quad s = 1, \dots, n_s. \tag{3.3}$$

6: **if** $n_s = 1$ **then** normalize using the quadrature rule (3.8): $\overline{u}^{1,(n+1)} \leftarrow \overline{v}^{1,(n)}$.

7: **else** orthogonalize using the stochastic modified Gram–Schmidt process: $\overline{u}^{s,(n+1)} \leftarrow \overline{v}^{s,(n)}, \quad s = 1, \dots, n_s$.

8: **end if**

9: Check convergence.

10: **end for**

11: Use the stochastic Rayleigh quotient (3.5) to compute the eigenvalue expansions.

12: **if** $\rho > 0$, shift $\lambda_1^s = \lambda_1^s - \rho$ for $s = 1, \dots, n_s$. **end if**

Matrix-vector product. The conjugate gradient method and computation of the stochastic Rayleigh quotient require a stochastic version of a matrix-vector product, which corresponds to evaluation of the projection

$$v_k^s = \langle v^s, \psi_k \rangle = \langle A u^s, \psi_k \rangle, \qquad k = 1, \dots, n_\xi.$$

Since $(V \otimes W)\,\mathrm{vec}(X) = \mathrm{vec}(WXV^T)$, the coefficients of the expansion are

$$\bar{v}^s = \mathbb{E}[\Psi \Psi^T \otimes A] \bar{u}^s = \sum_{\ell=1}^{n_a} (H_\ell \otimes A_\ell) \bar{u}^s \quad \Leftrightarrow \quad \bar{V}^s = \sum_{\ell=1}^{n_a} A_\ell \bar{U}^s H_\ell^T. \tag{3.4}$$

The use of this computation for the Rayleigh quotient is described below. We also note that Algorithm 1 can be modified to perform subspace iteration [23, Algorithm 4] for identifying

the largest eigenpairs. In this case, the solve (3.3) is simply replaced by a matrix-vector product (3.4).

Stochastic Rayleigh quotient. In the deterministic case, the Rayleigh quotient is used to compute the eigenvalue corresponding to a normalized eigenvector u as $\lambda = u^T v$, where $v = Au$. For the stochastic Galerkin method, the Rayleigh quotient defines the coefficients of a stochastic expansion of the eigenvalue defined via a projection

$$\lambda_k^s = \langle \lambda^s, \psi_k \rangle = \left\langle u^{sT} v^s, \psi_k \right\rangle, \qquad k = 1, \dots, n_\xi.$$

The coefficients of v^s are computed using (3.4) and the coefficients λ_k^s are

$$\lambda_k^s = \mathbb{E}\left[\left(\left(\Psi^T \otimes 1 \right) \bar{\lambda}^s \right) \psi_k \right] = \mathbb{E}\left[\left(\bar{u}^{sT} \left(\Psi \Psi^T \otimes I_{n_x} \right) \bar{v}^s \right) \psi_k \right], \qquad k = 1, \dots, n_\xi,$$

which is

$$\lambda_k^s = \sum_{j=1}^{n_\xi} \sum_{i=1}^{n_\xi} \left[H_k \circ \left\langle u_i^s, v_j^s \right\rangle_{\mathbb{R}} \right]_{ij} = \sum_{j=1}^{n_\xi} \sum_{i=1}^{n_\xi} \left[H_k \circ \left(\bar{U}^{sT} \bar{V}^s \right) \right]_{ij}, \qquad k = 1, \dots, n_\xi. \tag{3.5}$$

Remark 3.1. The Rayleigh quotient (3.5) finds n_ξ coefficients of the eigenvalue expansion, which is consistent with the Newton iteration formulated in section 4 and also with the literature [23, 37]. We note that it would be possible to compute the coefficients λ_k for $k > n_\xi$ as well, because the inner product $u^T v$ of two eigenvectors which are expanded using chaos polynomials up to degree p has nonzero chaos coefficients up to degree $2p$. An alternative is to use a *full* representation of the Rayleigh quotient based on the projection of $u^T Au$. However, from our experience in the present and previous work [33], the representation (3.5) is sufficient.

Normalization and the Gram–Schmidt process. Let $\|\cdot\|_2$ denote the vector norm, induced by the inner product $\langle \cdot, \cdot \rangle_{\mathbb{R}}$. That is, for a vector u evaluated at a point ξ,

$$\|u(\xi)\|_2 = \sqrt{\sum_{n=1}^{n_x} \left([u(\xi)]_n \right)^2}. \tag{3.6}$$

At each step of stochastic iteration the coefficients of a given set of vectors $\{v^s\}_{s=1}^{n_s}$ are transformed into an orthonormal set $\{u^s\}_{s=1}^{n_s}$ such that the condition

$$\left\langle u^s(\xi), u^t(\xi) \right\rangle_{\mathbb{R}} = \delta_{st} \qquad \text{a.s.}, \tag{3.7}$$

and in particular (2.13), is satisfied. We adopt the same strategy as in [23, 33], whereby the coefficients of the orthonormal eigenvectors are calculated using a discrete projection and a quadrature rule. An alternative approach to normalization, based on solution of a relatively small nonlinear system, was proposed by Hakula, Kaarnioja, and Laaksonen [12].

Let us first consider *normalization* of a vector, so $n_s = 1$. The coefficients in column k of $\bar{U}^1$ corresponding to coefficients of a normalized vector are computed as

$$u_k^1 = \sum_{q=1}^{n_q} \frac{v^1\left(\xi^{(q)}\right)}{\left\| v^1\left(\xi^{(q)}\right) \right\|_2} \psi_k\left(\xi^{(q)}\right) w^{(q)}. \tag{3.8}$$

When $n_s > 1$, the *orthonormalization* (3.7) is performed by a combination of stochastic

Galerkin projection and the modified Gram–Schmidt algorithm as proposed in [23],

$$\mathbb{E}\left[\Psi \otimes u^s\right] = \mathbb{E}\left[\Psi \otimes v^s\right] - \sum_{t=1}^{s-1} \mathbb{E}\left[\Psi \otimes \left(\frac{\langle v^s, u^t\rangle_{\mathbb{R}}}{\langle u^t, u^t\rangle_{\mathbb{R}}} u^t\right)\right], \quad s = 2, \ldots, n_s. \tag{3.9}$$

Using the expansion (2.6) and rearranging, the coefficients in column k of $\bar{U}^s$ are

$$u_k^s = v_k^s - \sum_{t=1}^{s-1} \chi_k^{ts}, \qquad k = 1, \ldots, n_\xi, \quad s = 2, \ldots, n_s,$$

where

$$\chi^{ts}(\xi) = \frac{\langle v^s(\xi), u^t(\xi)\rangle_{\mathbb{R}}}{\langle u^t(\xi), u^t(\xi)\rangle_{\mathbb{R}}} u^t(\xi),$$

and the coefficients χ_k^{ts} are computed using a discrete projection as in (2.8),

$$\chi_k^{ts} = \sum_{q=1}^{n_q} \chi^{ts}\left(\xi^{(q)}\right) \psi_k\left(\xi^{(q)}\right) w^{(q)}.$$

Stopping criteria. The inexact iteration entails in each step of Algorithm 1 a solution of the stochastic Galerkin problem (3.3) using the PCG method. We use the criteria proposed by Golub and Ye [11, equation (1)]; the criteria are satisfied when the relative residual of PCG gets smaller than a factor of the nonlinear residual from the previous step, that is,

$$\frac{\|\overline{u}^{s,(n)} - (\sum_{\ell=1}^{n_a} H_\ell \otimes A_\ell)\overline{v}^{s,(n)}\|_2}{\|\overline{u}^{s,(n)}\|_2} < \tau \left\| \left(\sum_{\ell=1}^{n_a} H_\ell \otimes A_\ell - \sum_{i=1}^{n_\xi} H_i \otimes \lambda_i^{s,(n-1)} I_{n_x} \right) \overline{u}^{s,(n-1)} \right\|_2, \tag{3.10}$$

where the factor $\tau = 10^{-2}$. It is important to note that Algorithm 1 provides only the coefficients of expansion of the projection of residual on the gPC basis, that is,

$$\widetilde{r}_k^s = \left\langle Au^s - \lambda^s u^s, \psi_k \right\rangle, \qquad k = 1, \ldots, n_\xi, \quad s = 1, \ldots, n_s. \tag{3.11}$$

One could assess accuracy using Monte Carlo sampling of this residual by computing

$$r^s\left(\xi^i\right) = A\left(\xi^i\right) u^s\left(\xi^i\right) - \lambda^s\left(\xi^i\right) u^s\left(\xi^i\right), \qquad i = 1, \ldots, N_{MC}, \quad s = 1, \ldots, n_s.$$

However, in the numerical experiments we use a much less expensive computation, which is based on using coefficients $\widetilde{r}_k^s$ directly as an error indicator. In particular, we monitor the norms of the terms of $\widetilde{r}_k^s$ corresponding to expected value and variance,

$$\varepsilon_1^{s,(it)} = \left\| \widetilde{r}_1^{s,(n)} \right\|_2, \qquad \varepsilon_{\sigma^2}^{s,(it)} = \left\| \sum_{k=2}^{n_\xi} \left(\widetilde{r}_k^{s,(n)} \right)^2 \right\|_2, \qquad s = 1, \ldots, n_s. \tag{3.12}$$

3.1. Preconditioners for the stochastic inverse iteration. We use two preconditioners for problem (3.3)—the mean-based preconditioner [29, 30] and the hierarchical Gauss–Seidel preconditioner [35]. Both preconditioners are formulated in the Kronecker-product format, and we also comment on the matricized formulation. We assume that a preconditioner M_1 for the mean matrix A_1 is available.

The mean-based preconditioner (MB) is given in Algorithm 2. Since $H_1 = I_{n_\xi}$, the preconditioner entails n_ξ block diagonal solves with M_1, and recalling that we can write $\bar{R}^s = \mathrm{mat}(\bar{r}^s)$, $\bar{V}^s = \mathrm{mat}(\bar{v}^s)$, its action can be equivalently obtained by solving

$$M_1 \bar{V}^s = \bar{R}^s. \tag{3.13}$$

The hierarchical Gauss–Seidel preconditioner (hGS) is given in Algorithm 3. We will denote by $v^s_{(i:n)}$ a subvector of $\bar{v}^s$ containing gPC coefficients $i, i+1, \ldots, n$, and, in particular,

Algorithm 2 [29, 30] Mean-based preconditioner (MB).

The preconditioner $M_{\mathrm{MB}} : \bar{r}^s \longmapsto \bar{v}^s$ for (3.3) is defined as

$$(H_1 \otimes M_1)\, \bar{v}^s = \bar{r}^s.$$

Algorithm 3 [35, Algorithm 3] Hierarchical Gauss–Seidel preconditioner (hGS).

The preconditioner $M_{\mathrm{hGS}} : \bar{r}^s \longmapsto \bar{v}^s$ for (3.3) is defined as follows.

1: Set the initial solution $\bar{v}^s$ to zero and update in the following steps:
2: Solve

$$M_1 v^s_1 = r^s_1 - \mathcal{F}_1 v^s_{(2:n_\xi)}, \qquad \text{where } \mathcal{F}_1 = \sum_{t \in \mathcal{I}_t} \left(\left[h_{t,(1)(2:n_\xi)} \right] \otimes A_t \right). \tag{3.14}$$

3: **for** $d = 1, \ldots, p-1$ **do**
4: Set $\ell = (n_\ell + 1 : n_u)$, where $n_\ell = \binom{m_\xi + d - 1}{d-1}$ and $n_u = \binom{m_\xi + d}{d}$.
5: Solve

$$(I_{n_u - n_\ell} \otimes M_1)\, v^s_{(\ell)} = r^s_{(\ell)} - \mathcal{E}_{d+1} v^s_{(1:n_\ell)} - \mathcal{F}_{d+1} v^s_{(n_u+1:n_\xi)}, \tag{3.15}$$

where

$$\mathcal{E}_{d+1} = \sum_{t \in \mathcal{I}_t} \left(\left[h_{t,(\ell)(1:n_\ell)} \right] \otimes A_t \right), \qquad \mathcal{F}_{d+1} = \sum_{t \in \mathcal{I}_t} \left(\left[h_{t,(\ell)(n_u+1:n_\xi)} \right] \otimes A_t \right).$$

6: **end for**
7: Set $\ell = (n_u + 1 : n_\xi)$.
8: Solve

$$\left(I_{n_\xi - n_u} \otimes M_1\right) v^s_{(\ell)} = r^s_{(\ell)} - \mathcal{E}_{p+1} v^s_{(1:n_u)}, \qquad \text{where } \mathcal{E}_{p+1} = \sum_{t \in \mathcal{I}_t} \left(\left[h_{t,(\ell)(1:n_u)} \right] \otimes A_t \right).$$

9: **for** $d = p-1, \ldots, 1$ **do**
10: Set $\ell = (n_\ell + 1 : n_u)$, where $n_\ell = \binom{m_\xi + d - 1}{d-1}$ and $n_u = \binom{m_\xi + d}{d}$.
11: Solve (3.15).
12: **end for**
13: Solve (3.14).

$\bar{v}^s = v^s_{(1:n_\xi)}$. There are two components of the preconditioner. The first component consists of block-diagonal solves with blocks of varying sizes but is computed just as in Algorithm 2, resp., in (3.13). The second component is used in the setup of the right-hand sides for the solves and consists of matrix-vector products by certain subblocks of the stochastic Galerkin matrix by vectors of corresponding sizes. To this end, we will write $\left[h_{t,(\ell)(k)}\right]$, with (ℓ) and (k) denoting a set of (consecutive) rows and columns of matrix H_t so that, in particular, $H_t = [h_{t,(1:n_\xi)(1:n_\xi)}]$. Then, the matrix-vector products can be written (cf. (3.4) and note the symmetry of H_t) as

$$v^s_{(\ell)} = \sum_{t\in\mathcal{I}_t} (\left[h_{t,(\ell)(k)}\right] \otimes A_t) u^s_{(k)} \quad \Leftrightarrow \quad V^s_{(\ell)} = \sum_{t\in\mathcal{I}_t} A_t U^s_{(k)} \left[h_{t,(k)(\ell)}\right], \tag{3.16}$$

where $\mathcal{I}_t$ is an index set $\mathcal{I}_t \subseteq \{1,\dots,n_\xi\}$ indicating that the matrix-vector products may be truncated. Possible strategies for truncation are discussed in [35]. In this study, we use $\mathcal{I}_t = \{1,\dots,n_t\}$ with $n_t = \binom{m_\xi+p_t}{p_t}$ for some $p_t \le p$ and, in particular, we set $t = \{0,1,2\}$. We also note that, since the initial guess is zero in Algorithm 3, the multiplications by $\mathcal{F}_1$ and $\mathcal{F}_{d+1}$ vanish from (3.14)–(3.15).

4. Newton iteration. Use of the Newton iteration to solve (2.9)–(2.10) was proposed in [6] and most recently studied in [1]. We use a similar strategy here and formulate a line-search Newton method as Algorithm 4. To begin, we consider the system of nonlinear equations (2.14)–(2.15) and rewrite it as

$$\begin{bmatrix} F(\bar{u}^s, \bar{\lambda}^s) \\ G(\bar{u}^s) \end{bmatrix} = 0, \qquad s = 1,\dots,n_s, \tag{4.1}$$

where

$$F(\bar{u}^s, \bar{\lambda}^s) \equiv \mathbb{E}[\Psi\Psi^T \otimes A]\bar{u}^s - \mathbb{E}[((\bar{\lambda}^s)^T\Psi)\Psi\Psi^T \otimes I_{n_x}]\bar{u}^s, \tag{4.2}$$

$$G(\bar{u}^s) \equiv \mathbb{E}[\Psi \otimes \left((\bar{u}^{sT}(\Psi\Psi^T \otimes I_{n_x})\bar{u}^s) - 1\right)]. \tag{4.3}$$

The Jacobian matrix of (4.1) is

$$\mathcal{J}(\bar{u}^s, \bar{\lambda}^s) = \begin{bmatrix} \frac{\partial F}{\partial \bar{u}^s} & \frac{\partial F}{\partial \bar{\lambda}^s} \\ \frac{\partial G}{\partial \bar{u}^s} & 0 \end{bmatrix}, \tag{4.4}$$

where

$$\frac{\partial F}{\partial \bar{u}^s}(\bar{\lambda}^s) = \mathbb{E}[\Psi\Psi^T \otimes A] - \mathbb{E}[((\bar{\lambda}^s)^T\Psi)\Psi\Psi^T \otimes I_{n_x}], \tag{4.5}$$

$$\frac{\partial F}{\partial \bar{\lambda}^s}(\bar{u}^s) = -\mathbb{E}[\Psi^T \otimes (\Psi\Psi^T \otimes I_{n_x})\bar{u}^s], \tag{4.6}$$

$$\frac{\partial G}{\partial \bar{u}^s}(\bar{u}^s) = 2\mathbb{E}[\Psi \otimes ((\bar{u}^s)^T(\Psi\Psi^T \otimes I_{n_x}))]. \tag{4.7}$$

Step n of Newton iteration entails solving a linear system

$$\begin{bmatrix} \frac{\partial F}{\partial \bar{u}^s}(\bar{\lambda}^{s,(n)}) & \frac{\partial F}{\partial \bar{\lambda}^s}(\bar{u}^{s,(n)}) \\ \frac{\partial G}{\partial \bar{u}^s}(\bar{u}^{s,(n)}) & 0 \end{bmatrix} \begin{bmatrix} \delta\bar{u}^s \\ \delta\bar{\lambda}^s \end{bmatrix} = - \begin{bmatrix} F(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}) \\ G(\bar{u}^{s,(n)}) \end{bmatrix}, \tag{4.8}$$

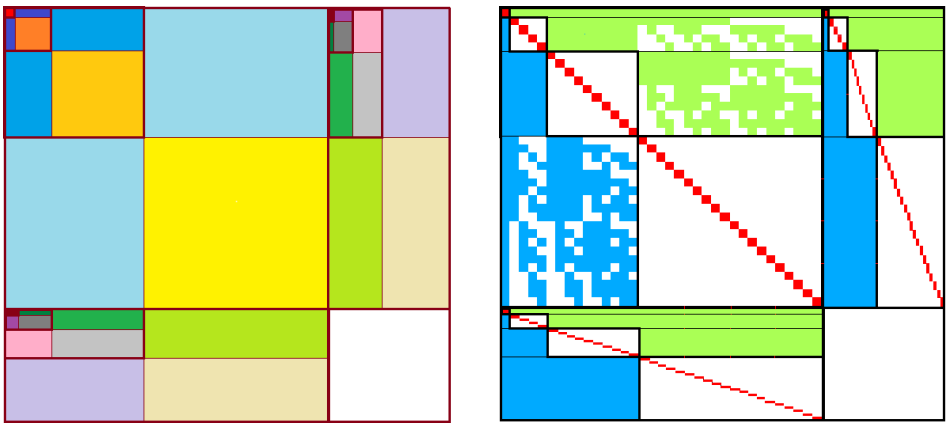

Figure 4.1. *Hierarchical structure of the symmetric Jacobian matrix from (4.10) (left) and splitting operator for the constraint hierarchical Gauss–Seidel preconditioner from Algorithm 7–8 (right).*

followed by an update of the solution

$$\begin{bmatrix} \bar{u}^{s,(n+1)} \\ \bar{\lambda}^{s,(n+1)} \end{bmatrix} = \begin{bmatrix} \bar{u}^{s,(n)} \\ \bar{\lambda}^{s,(n)} \end{bmatrix} + \begin{bmatrix} \delta\bar{u}^{s} \\ \delta\bar{\lambda}^{s} \end{bmatrix}. \tag{4.9}$$

The matrix $\mathcal{J}(\bar{u}^s, \bar{\lambda}^s)$ is nonsymmetric, but since $\frac{\partial F}{\partial \bar{\lambda}^s}(\bar{u}^{s,(n)}) = \left[-\frac{1}{2}\frac{\partial G}{\partial \bar{u}^s}(\bar{u}^{s,(n)})\right]^T$, we modify linear system (4.8) in our implementation as

$$\begin{bmatrix} \frac{\partial F}{\partial \bar{u}^s}(\bar{\lambda}^{s,(n)}) & \frac{\partial F}{\partial \bar{\lambda}^s}(\bar{u}^{s,(n)}) \\ \left[\frac{\partial F}{\partial \bar{\lambda}^s}(\bar{u}^{s,(n)})\right]^T & 0 \end{bmatrix} \begin{bmatrix} \delta\bar{u}^{s} \\ \delta\bar{\lambda}^{s} \end{bmatrix} = \begin{bmatrix} -F(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}) \\ \frac{1}{2}G(\bar{u}^{s,(n)}) \end{bmatrix}, \tag{4.10}$$

which restores symmetry of linear systems solved in each step of Newton iteration. The symmetric Jacobian matrix in (4.10) will be denoted by $J(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)})$. The hierarchical structure of the Jacobian matrix, which is due to the stochastic Galerkin projection, is illustrated by the left panel of Figure 4.1. The systems (4.10) are solved inexactly using a preconditioned Krylov subspace method, and the details of evaluation of the right-hand side and the matrix-vector product are given in Appendix A.

4.1. Inexact line-search Newton method. In order to improve global convergence behavior of the Newton iteration, we consider a line-search modification of the method following [26, Algorithm 11.4]. To begin, let us define the merit function as the sum of squares,

$$f(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}) = \frac{1}{2}\|r(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)})\|_2^2,$$

where r is the residual of (4.1), and denote

$$f_n = f(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}), \qquad r_n = r(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}), \qquad J_n = J(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}).$$

As the initial approximation of the solution, we use the eigenvectors and eigenvalues of the associated mean problem given by the matrix A_1 concatenated by zeros, that is, $\bar{u}^{s,(0)} = [(u_1^{s,(0)})^T, 0, \ldots]^T$ and $\bar{\lambda}^{s,(0)} = [\lambda_1^{s,(0)}, 0, \ldots]^T$, and the initial residual is

$$r_0 = \begin{bmatrix} F(\bar{u}^{s,(0)}, \bar{\lambda}^{s,(0)}) \\ G(\bar{u}^{s,(0)}) \end{bmatrix}.$$

The line-search Newton method is summarized in our setting as Algorithm 4, and the choice of parameters ρ and c in the numerical experiments is discussed in section 5.1.

Algorithm 4 [26, Algorithm 11.4] Line-search Newton method.

1: Given $\rho, c \in (0,1)$, set $\alpha^* = 1$.
2: Set $\bar{u}^{(0)}$ and $\bar{\lambda}^{(0)}$.
3: **for** $n = 0, 1, 2, \ldots$ **do**
4: $\quad J_n p_n = -r_n$ (Find the Newton update p_n.)
5: $\quad \begin{bmatrix} \delta\bar{u}^{(n)} \\ \delta\bar{\lambda}^{(n)} \end{bmatrix} = p_n$
6: $\quad \alpha_n = \alpha^*$
7: $\quad$ **while** $f(\bar{u}^{(n)} + \alpha_n \delta\bar{u}^{(n)}, \bar{\lambda}^{(n)} + \alpha_n \delta\bar{\lambda}^{(n)}) > f_n + c\,\alpha_n \nabla f_n^T p_n$ **do**
8: $\quad\quad \alpha_n \leftarrow \rho\,\alpha_n$
9: $\quad$ **end while**
10: $\quad \bar{u}^{(n+1)} \leftarrow \bar{u}^{(n)} + \alpha_n \delta\bar{u}^{(n)}$
11: $\quad \bar{\lambda}^{(n+1)} \leftarrow \bar{\lambda}^{(n)} + \alpha_n \delta\bar{\lambda}^{(n)}$
12: $\quad$ Check for convergence.
13: **end for**

The inexact iteration entails in each step a solution of the stochastic Galerkin linear system in line 4 of Algorithm 4 given by (4.10) using a Krylov subspace method. In our algorithm we use the adaptive stopping criteria for the method,

$$\frac{\|r_n + J_n p_n\|_2}{\|r_n\|_2} < \tau \|r_{n-1}\|_2, \tag{4.11}$$

where $\tau = 10^{-1}$. The for-loop is terminated when the convergence check in line 12 is satisfied; in our numerical experiments we check if $\|r_n\|_2 < 10^{-10}$.

4.2. Preconditioners for the Newton iteration. The Jacobian matrices in (4.10) are symmetric, indefinite, and so the linear systems can be ideally solved using the MINRES iterative method. It is well known that a preconditioner for MINRES must be symmetric and positive definite; cf., e.g., [39]. A popular choice is a block diagonal preconditioner (cf. [24]),

$$\begin{bmatrix} \widetilde{A} & 0 \\ 0 & \widetilde{S} \end{bmatrix},$$

where $\widetilde{A} \approx A$ and the Schur complement $\widetilde{S} \approx BA^{-1}B^T$ are obtained as approximations of the blocks in (A.4). Such a preconditioner, based on truncation of the series in (A.1) and (A.2)

to the very first term, was used in [1]. In such a setup, we get

$$\begin{aligned}\widetilde{A} &= I_{n_\xi} \otimes A_1 - (\lambda_1^s I_{n_\xi} \otimes I_{n_x}) = I_{n_\xi} \otimes (A_1 - \lambda_1^s I_{n_x}) \\ &\approx I_{n_\xi} \otimes (1 - \lambda_1^s)(A_1 - I_{n_x}) \\ &\approx I_{n_\xi} \otimes M_1^s,\end{aligned}$$

where the second line was used in [1]. In this study, we use the third line with

$$M_1^s = A_1 - \epsilon_M \mu^s I_{n_x}, \tag{4.12}$$

where μ^s is the eigenvalue of the mean problem; cf. (3.1). We note that it might be desirable to set the parameter $\epsilon_M \approx 1$, but $\epsilon_M \neq 1$ in order to guarantee nonsingular M_1^s, however; more details for setup and use of (4.12) are given in numerical experiments. Considering the first column of (A.2) (cf. (4.6) and (A.3)), we get

$$\widetilde{B}^T = -(I_{n_\xi} \otimes u_1^s),$$

and the approximation $\widetilde{S}$ is

$$\begin{aligned}\widetilde{S} &= (I_{n_\xi} \otimes u_1^{sT}) \left[I_{n_\xi} \otimes (A_1 - \lambda_1^s I_{n_x})\right]^{-1} (I_{n_\xi} \otimes u_1^s) \\ &\approx I_{n_\xi} \otimes \left[u_1^{sT} (1 - \lambda_1^s)^{-1} (A_1 - I_{n_x})^{-1} u_1^s\right] \\ &\approx I_{n_\xi} \otimes \left[u_1^{sT} (M_1^s)^{-1} u_1^s\right],\end{aligned}$$

where the second line was used in [1]. In this study, we use the third line with $(M_1^s)^{-1} u_1^s$ denoting an application of M_1^s to u_1^s. The ideal choice of u_1^s are the coefficients of the mean of eigenvector s, and we consider two approximations here: (a) u_1^s is set as the corresponding eigenvector of the mean matrix A_1, or (b) u_1^s is the approximation of the gPC coefficients of the corresponding eigenvector updated after each step of Newton iteration (Algorithm 4). The preconditioners are thus either (a) *fixed* during Newton iteration or (b) *updated* after each step. These two variants and our version of the mean-based preconditioner (NMB) for problem (4.10) are summarized in Algorithm 5. Clearly, if M_1^s is symmetric, positive definite, so is the preconditioner M_{NMB}, but the preconditioner loses positive definiteness if the eigenvalue of interest is not the smallest one (cf. (4.12)), and therefore, along with MINRES, we also use GMRES and develop several preconditioners for this method.

Next, we propose a variant of the so-called constraint preconditioner (cf. [16]),

$$\begin{bmatrix} \widetilde{A} & \widetilde{B}^T \\ \widetilde{B} & 0 \end{bmatrix}.$$

Similarly as above, both $\widetilde{A}$ and $\widetilde{B}$ are approximations of the blocks in (A.4). The preconditioner is clearly indefinite (which also precludes use of MINRES). Our variant of the constraint mean-based preconditioner (cMB) is listed as Algorithm 6.

Algorithm 5 Mean-based preconditioner for the Newton iteration (NMB).

The preconditioner $M_{\mathrm{NMB}} : \left(\bar{r}^{(u),s}, \bar{r}^{(\lambda),s}\right) \longmapsto \left(\bar{v}^{(u),s}, \bar{v}^{(\lambda),s}\right)$ is defined as

$$\begin{bmatrix} I_{n_\xi} \otimes M_1^s & 0 \\ 0 & I_{n_\xi} \otimes \left[w^{s,(n)T} \left(M_1^s\right)^{-1} w^{s,(n)} \right] \end{bmatrix} \begin{bmatrix} \bar{v}^{(u),s} \\ \bar{v}^{(\lambda),s} \end{bmatrix} = \begin{bmatrix} \bar{r}^{(u),s} \\ \bar{r}^{(\lambda),s} \end{bmatrix}, \tag{4.13}$$

where $w^{s,(n)}$ is (a) eigenvector w^s of A_1 corresponding to eigenvalue μ^s (cf. (3.1)) or (b) the first (mean) gPC coefficients $u_1^{s,(n)}$ of eigenvector s at step n of Algorithm 4.

Algorithm 6 Constraint mean-based preconditioner (cMB).

The preconditioner $M_{\mathrm{cMB}} : \left(\bar{r}^{(u),s}, \bar{r}^{(\lambda),s}\right) \longmapsto \left(\bar{v}^{(u),s}, \bar{v}^{(\lambda),s}\right)$ is defined as

$$\begin{bmatrix} I_{n_\xi} \otimes M_1^s & -I_{n_\xi} \otimes w^{s,(n)} \\ -I_{n_\xi} \otimes w^{s,(n)T} & 0 \end{bmatrix} \begin{bmatrix} \bar{v}^{(u),s} \\ \bar{v}^{(\lambda),s} \end{bmatrix} = \begin{bmatrix} \bar{r}^{(u),s} \\ \bar{r}^{(\lambda),s} \end{bmatrix}, \tag{4.14}$$

where $w^{s,(n)}$ is set as in Algorithm 5.

In an analogy to Algorithm 2 and (3.13), the action of the preconditioners from Algorithms 5 and 6 can be equivalently obtained by solving

$$\mathcal{M}_1 \begin{bmatrix} \bar{V}^{(u),s} \\ \bar{V}^{(\lambda),s} \end{bmatrix} = \begin{bmatrix} \bar{R}^{(u),s} \\ \bar{R}^{(\lambda),s} \end{bmatrix}, \tag{4.15}$$

where $\mathcal{M}_1$ is the deterministic part the preconditioners from (4.13) or (4.14), that is,

$$\mathcal{M}_1 = \begin{bmatrix} M_1^s & 0 \\ 0 & w^{s,(n)T} \left(M_1^s\right)^{-1} w^{s(n)} \end{bmatrix} \quad \text{or} \quad \mathcal{M}_1 = \begin{bmatrix} M_1^s & -w^{s(n)} \\ -w^{s,(n)T} & 0 \end{bmatrix}.$$

We also formulate a constraint version of the preconditioner from Algorithm 3, which is called a constraint hierarchical Gauss–Seidel preconditioner (chGS). It is formulated as Algorithm 7–8, and a scheme of the splitting operator is illustrated by the right panel of Figure 4.1. There are two components of the preconditioner. The first component consists of block-diagonal solves with blocks of varying sizes computed just as in Algorithm 6, resp., (4.15). The second component is used in the setup of the right-hand sides for the solves and consists of matrix-vector products by certain subblocks of the stochastic Jacobian matrices by vectors of corresponding sizes. An example of matrix-vector product with a subblock of the stochastic Jacobian matrix is given in Appendix B. We also note that, since the initial guess is zero, the multiplications by $\mathcal{F}_1$ and $\mathcal{F}_{d+1}$ vanish from (4.16)–(4.17).

5. Numerical experiments.

We implemented the methods in MATLAB, and in this section we present the results of numerical experiments in which the proposed inexact solvers are applied to two benchmark problems: a diffusion problem with stochastic coefficient and stiffness of Mindlin plate with stochastic Young's modulus.

Algorithm 7 Constraint hierarchical Gauss–Seidel preconditioner (chGS).

The preconditioner $M_{\text{chGS}} : \left(\bar{r}^{(u),s}, \bar{r}^{(\lambda),s}\right) \longmapsto \left(\bar{v}^{(u),s}, \bar{v}^{(\lambda),s}\right)$ is defined as follows.

1: Set the initial solution $\left(\bar{v}^{(u),s}, \bar{v}^{(\lambda),s}\right)$ to zero and update in the following steps:

2: Solve

$$\mathcal{M}_1 \begin{bmatrix} v_1^{(u),s} \\ v_1^{(\lambda),s} \end{bmatrix} = \begin{bmatrix} r_1^{(u),s} \\ r_1^{(\lambda),s} \end{bmatrix} - \mathcal{F}_1 \begin{bmatrix} v_{(2:n_\xi)}^{(u),s} \\ v_{(2:n_\xi)}^{(\lambda),s} \end{bmatrix}, \tag{4.16}$$

where

$$\mathcal{M}_1 = \begin{bmatrix} M_1^s & -w^{s,(n)} \\ -w^{s,(n)T} & 0 \end{bmatrix}, \text{where } w^{s,(n)} \text{ is set as in Algorithm 5,}$$

$$\mathcal{F}_1 = \begin{bmatrix} \sum_{t\in\mathcal{I}_t} \left[h_{t,(1)(2:n_\xi)}\right] \otimes A_t - \sum_{t\in\mathcal{I}_t} \left[h_{t,(1)(2:n_\xi)}\right] \otimes \lambda_t^{s,(n)} I_{n_x} & \mathcal{G}_1 \\ \mathcal{H}_1 & 0 \end{bmatrix},$$

$$\mathcal{G}_1 = \sum_{t\in\mathcal{I}_t} \left[h_{t,(1)(2:n_\xi)}\right] \otimes w_t^{s,(n)},$$

$$\mathcal{H}_1 = \sum_{t\in\mathcal{I}_t} \left[h_{t,(1)(2:n_\xi)}\right] \otimes (w_t^{s,(n)})^T,$$

where $w_t^{s,(n)}$ is the eigenvector s at step n of Algorithm 4.

3: **for** $d = 1, \ldots, p-1$ **do**

4: Set $\ell = (n_\ell + 1 : n_u)$, where $n_\ell = \binom{n_\xi+d-1}{d-1}$ and $n_u = \binom{n_\xi+d}{d}$.

5: Solve

$$\mathcal{M}_{d+1} \begin{bmatrix} v_{(\ell)}^{(u),s} \\ v_{(\ell)}^{(\lambda),s} \end{bmatrix} = \begin{bmatrix} r_{(\ell)}^{(u),s} \\ r_{(\ell)}^{(\lambda),s} \end{bmatrix} - \mathcal{E}_{d+1} \begin{bmatrix} v_{(1:n_\ell)}^{(u),s} \\ v_{(1:n_\ell)}^{(\lambda),s} \end{bmatrix} - \mathcal{F}_{d+1} \begin{bmatrix} v_{(n_u+1:n_\xi)}^{(u),s} \\ v_{(n_u+1:n_\xi)}^{(\lambda),s} \end{bmatrix}, \tag{4.17}$$

where

$$\mathcal{M}_{d+1} = \left(I_{n_u-n_\ell} \otimes \begin{bmatrix} M_1^s & -w^{s,(n)} \\ -w^{s,(n)T} & 0 \end{bmatrix}\right), \text{where } w^{s,(n)} \text{ is set as in Algorithm 5,}$$

$$\mathcal{E}_{d+1} = \begin{bmatrix} \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_\ell)}\right] \otimes A_t - \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_\ell)}\right] \otimes \lambda_t^{s,(n)} I_{n_x} & \mathcal{G}_{d+1}^{\mathcal{E}} \\ \mathcal{H}_{d+1}^{\mathcal{E}} & 0 \end{bmatrix},$$

$$\mathcal{G}_{d+1}^{\mathcal{E}} = \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_\ell)}\right] \otimes w_t^{s,(n)},$$

$$\mathcal{H}_{d+1}^{\mathcal{E}} = \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_\ell)}\right] \otimes (w_t^{s,(n)})^T,$$

$$\mathcal{F}_{d+1} = \begin{bmatrix} \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(n_u+1:n_\xi)}\right] \otimes A_t - \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(n_u+1:n_\xi)}\right] \otimes \lambda_t^{s,(n)} I_{n_x} & \mathcal{G}_{d+1}^{\mathcal{F}} \\ \mathcal{H}_{d+1}^{\mathcal{F}} & 0 \end{bmatrix},$$

$$\mathcal{G}_{d+1}^{\mathcal{F}} = \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(n_u+1:n_\xi)}\right] \otimes w_t^{s,(n)},$$

$$\mathcal{H}_{d+1}^{\mathcal{F}} = \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(n_u+1:n_\xi)}\right] \otimes (w_t^{s,(n)})^T.$$

6: **end for**

Algorithm 8 Constraint hierarchical Gauss–Seidel preconditioner (chGS), continued.

7: Set $\ell = (n_u + 1 : n_\xi)$.

8: Solve*

$$\mathcal{M}_{p+1} \begin{bmatrix} v_{(\ell)}^{(u),s} \\ v_{(\ell)}^{(\lambda),s} \end{bmatrix} = \begin{bmatrix} r_{(\ell)}^{(u),s} \\ r_{(\ell)}^{(\lambda),s} \end{bmatrix} - \mathcal{E}_{p+1} \begin{bmatrix} v_{(1:n_u)}^{(u),s} \\ v_{(1:n_u)}^{(\lambda),s} \end{bmatrix},$$

where

$$\mathcal{M}_{p+1} = \left(I_{n_\xi - n_u} \otimes \begin{bmatrix} M_1^s & -w^{s,(n)} \\ -w^{s,(n)T} & 0 \end{bmatrix} \right), \text{ where } w^{s,(n)} \text{ is set as in Algorithm 5,}$$

$$\mathcal{E}_{p+1} = \begin{bmatrix} \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_u)}\right] \otimes A_t - \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_u)}\right] \otimes \lambda_t^{s,(n)} I_{n_x} & \mathcal{G}_{p+1}^{\mathcal{E}} \\ \mathcal{H}_{p+1}^{\mathcal{E}} & 0 \end{bmatrix},$$

$$\mathcal{G}_{p+1}^{\mathcal{E}} = \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_u)}\right] \otimes w_t^{s,(n)},$$

$$\mathcal{H}_{p+1}^{\mathcal{E}} = \sum_{t\in\mathcal{I}_t} \left[h_{t,(\ell)(1:n_u)}\right] \otimes (w_t^{s,(n)})^T.$$

where $w_t^{s,(n)}$ is the eigenvector s at step n of Algorithm 4.

9: **for** $d = p-1, \ldots, 1$ **do**

10: Set $\ell = (n_\ell + 1 : n_u)$, where $n_\ell = \binom{n_\xi+d-1}{d-1}$ and $n_u = \binom{n_\xi+d}{d}$.

11: Solve (4.17).

12: **end for**

13: Solve (4.16).

5.1. Stochastic diffusion problem with lognormal coefficient. For the first benchmark problem we consider the elliptic equation with stochastic coefficient and deterministic Dirichlet boundary condition

$$\begin{aligned} -\nabla \cdot (a(x,\xi)\nabla u(x,\xi)) &= \lambda(\xi) u(x,\xi) && \text{in } D \times \Gamma, \\ u(x,\xi) &= 0 && \text{on } \partial D \times \Gamma, \end{aligned}$$

where D is a two-dimensional physical domain. The uncertainty in the model is introduced by the stochastic expansion of the diffusion coefficient, considered as

$$a(x,\xi) = \sum_{\ell=1}^{n_a} a_\ell(x)\psi_\ell(\xi), \tag{5.1}$$

to be a truncated lognormal process transformed from the underlying Gaussian process [5]. That it, $\psi_\ell(\xi)$, $\ell = 1, \ldots, n_a$, is a set of Hermite polynomials and, denoting the coefficients of the Karhunen–Loève expansion of the Gaussian process by $g_j(x)$ and $\eta_j = \xi_j - g_j$, $j = 1, \ldots, m_\xi$, the coefficients in expansion (5.1) are computed as

$$a_\ell(x) = \frac{\mathbb{E}\left[\psi_\ell(\eta)\right]}{\mathbb{E}\left[\psi_\ell^2(\eta)\right]} \exp\left[g_0 + \frac{1}{2} \sum_{j=1}^{m_\xi} (g_j(x))^2 \right].$$

The covariance function of the Gaussian field, for points $X_1 = (x_1, y_1)$ and $X_2 = (x_2, y_2)$ in D, was chosen to be

$$C(X_1, X_2) = \sigma_g^2 \exp\left(-\frac{|x_2 - x_1|}{L_x} - \frac{|y_2 - y_1|}{L_y}\right), \tag{5.2}$$

where L_x and L_y are the correlation lengths of the random variables ξ_i, $i = 1, \ldots, m_\xi$, in the x and y directions, respectively, and σ_g is the standard deviation of the Gaussian random field. According to [21], in order to guarantee a complete representation of the lognormal process by (5.1), the degree of polynomial expansion of $a(x, \xi)$ should be twice the degree of the expansion of the solution. We follow the same strategy here. Therefore, the values of n_ξ and n_a are (cf., e.g., [7, p. 87] or [43, section 5.2]) $n_\xi = \frac{(m_\xi+p)!}{m_\xi!p!}$, $n_a = \frac{(m_\xi+2p)!}{m_\xi!(2p)!}$. In the numerical experiments, the lognormal diffusion coefficient (5.1) is parameterized using $m_\xi = 3$ random variables. The correlation length is $L_{\text{corr}} = 2$, and the coefficient of variation (CoV) of the lognormal process is set to either 0.1 (10%) or 0.25 (25%), where $CoV = \sigma/a_1$, the ratio of the standard deviation σ, and the mean of the diffusion coefficient a_1. For the gPC expansion of eigenvalues/eigenvectors (2.5), the maximal degree of gPC expansion is $p = 3$, so then $n_\xi = 20$ and $n_a = 84$.

Finite element discretization leads to a generalized eigenvalue problem

$$K(\xi)u = \lambda M u, \tag{5.3}$$

where $K(\xi) = \sum_{\ell=1}^{n_a} K_\ell \psi_\ell(\xi)$ is the stochastic expansion of the stiffness matrix, and the mass matrix M is deterministic. Using Cholesky factorization $M = LL^T$, the generalized eigenvalue problem (5.3) can be transformed into the standard form

$$A(\xi)w = \lambda w, \tag{5.4}$$

where $u = L^{-T}w$ and the expansion of A corresponding to (2.2) is

$$A = \sum_{\ell=1}^{n_a} A_\ell \psi_\ell(\xi) = \sum_{\ell=1}^{n_a} \left[L^{-1} K_\ell L^{-T}\right] \psi_\ell(\xi). \tag{5.5}$$

We consider the physical domain $D = [-1, 1]^2$, discretized using a structured grid using 256 bilinear finite elements, that is, with 225 nodes interior to D, which determines the size of matrices A_ℓ in (5.5). The 25 smallest eigenvalues of the mean matrix A_1 are displayed in Figure 5.1. For the quadrature rule, in section 2.1, we use Smolyak sparse grid with Gauss–Hermite quadrature and grid level 4 and 10^4 samples for the Monte Carlo method. With these settings, the size of $h_{\ell,kj}$ in (2.11) was $84 \times 20 \times 20$ with 806 nonzeros, and there were 69 points on the sparse grid.

Inexact stochastic inverse subspace iteration. First, we examine the performance of the inexact SISI from Algorithm 1 for computing the five smallest eigenvalues and corresponding eigenvectors of problem (5.4). Linear systems (3.3) are solved using the PCG method with the mean-based preconditioner (Algorithm 2) and the hierarchical Gauss–Seidel preconditioner (Algorithm 3). We ran the SISI algorithm with a fixed number of steps set to 20. Figure 5.2 illustrates convergence history in terms of the two error indicators ϵ_1 and ϵ_{σ^2} from (3.12) with

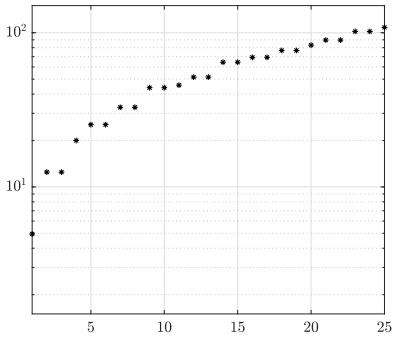

Figure 5.1. *The smallest* 25 *eigenvalues of the mean matrix* A_1.

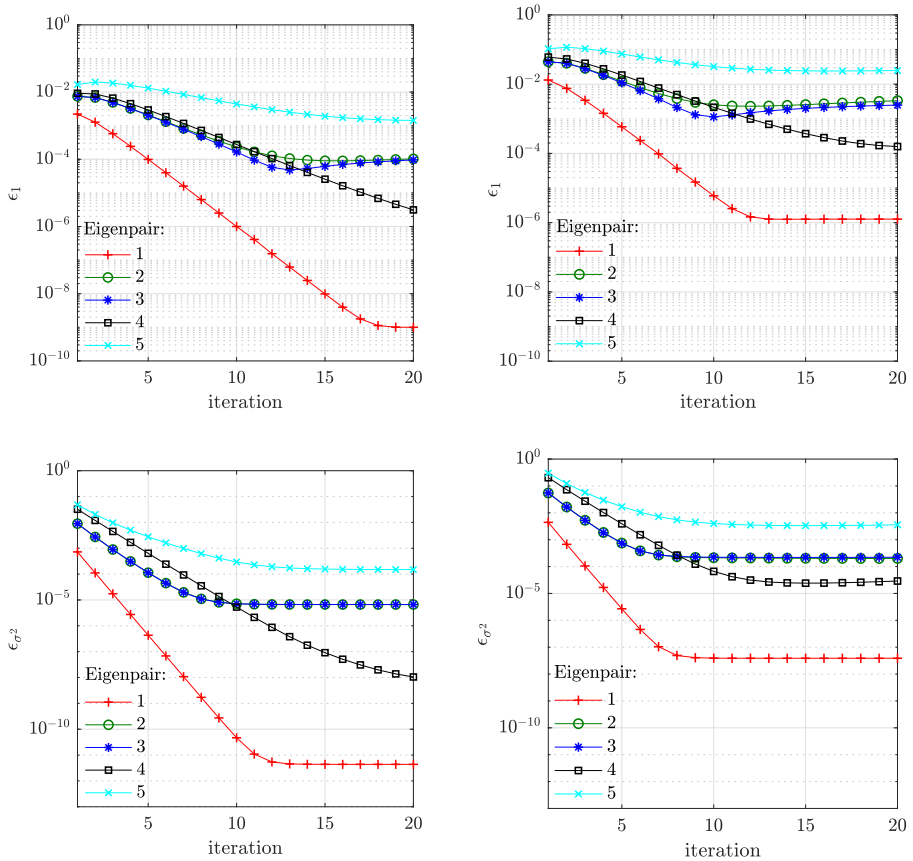

Figure 5.2. *Convergence history of the inexact SISI in terms of indicators* ϵ_1 *(top) and* ϵ_{σ^2} *(bottom) defined by* (3.12) *with* $CoV = 10\%$ *(left) and* 25% *(right).*

$CoV = 10\%$ (left panels) and 25% (right panels). The plots were generated using the hGS preconditioner with $p_t = 2$ ($\mathcal{I}_t = \{1, \ldots, 10\}$), but convergence with other preconditioners was virtually identical.

Next, we examine performance of PCG with the two preconditioners used to solve linear systems (3.3) with zero initial guess and stopping criterion (3.10). We computed the five small-

Table 5.1

Average number of PCG iterations for computing the five smallest eigenvalues and corresponding eigenvectors of the diffusion problem with $CoV = 10\%$ (left) and 25% (right) using the inexact SISI (Algorithm 1).

	$CoV = 10\%$					$CoV = 25\%$				
Preconditioner	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
MB	6.45	3.90	3.90	4.60	3.75	8.60	5.55	5.55	6.05	4.75
hGS ($p_t = 1$)	3.10	1.95	1.95	2.25	1.95	3.65	2.75	2.75	2.65	2.00
hGS ($p_t = 2$)	2.35	1.70	1.70	1.65	1.00	2.60	1.90	1.90	1.85	1.75
hGS (no trunc.)	2.15	1.00	1.00	1.45	1.00	2.60	1.80	1.80	1.75	1.65

est eigenvalues using 20 steps of the inexact SISI method. Table 5.1 shows the number of the PCG iterations required by the inexact solves, averaged over the 20 steps of the inexact SISI method for the model eigenvalue problem with $CoV = 10\%$ and 25%. Specifically, we compare the mean-based preconditioner from Algorithm 2 and the hGS preconditioner from Algorithm 3 with varying levels of truncation of the matrix-vector multiplications ($p_t = \{0, 1, 2\}$ and $p_t = 3$, i.e., no truncation). In both preconditioners we used Cholesky factorization of A_1 for the solves with M_1. We note that with $p_t = 0$ the hGS preconditioner reduces to the mean-based preconditioner. In both cases $CoV = 10\%$ and 25% the hGS preconditioner outperforms the mean-based preconditioner in terms of the number of PCG iterations for each of the five eigenpairs. Table 5.1 also shows that solving the eigenvalue problem with higher CoV leads to only a slight increase in the number of iterations.

Newton iteration. Next, we examine the inexact line-search Newton method from Algorithm 4 for computing the five smallest eigenvalues and corresponding eigenvectors of problem (5.4). For the line-search method, we set $\rho = 0.9$ for the backtracking and limit the maximum number of backtracks to 25, and $c = 0.05$. The initial guess for the nonlinear iteration is set using the (five smallest) eigenvalues and corresponding eigenvectors of the eigenvalue problem associated with the mean matrix A_1 as discussed in section 4.1. The nonlinear iteration terminates when the norm of the residual $\|r_n\|_2 < 10^{-10}$. The linear systems in Line 4 in Algorithm 4 are solved using either MINRES or GMRES with the mean-based preconditioner (Algorithm 5), the constraint mean-based preconditioner (Algorithm 6), and the contraint hierarchical Gauss–Seidel preconditioner (Algorithm 7–8). Figure 5.3 illustrates the convergence history of the inexact line-search Newton method in terms of norm of the residual $\|r_n\|_2$ with $CoV = 10\%$ (left panel) and 25% (right panel). The plots were generated using GMRES with the chGS preconditioner (Algorithm 7–8) with $p_t = 2$ ($\mathcal{I}_t = \{1, \ldots, 10\}$), but convergence with other preconditioners was virtually identical.

Next, we compare performance of MINRES and GMRES with the preconditioners from Algorithms 5–8 used to solve linear systems at line 4 in Algorithm 4 with zero initial guess and the stopping criterion (4.11). Table 5.2 shows the numbers of MINRES or GMRES iterations required by the inexact solves, averaged over the number of the nonlinear steps. Specifically, we compare the mean-based preconditioner (NMB) from Algorithm 5, the constraint mean-based preconditioner (cMB) from Algorithm 6, and the constraint hierarchical Gauss–Seidel preconditioner (chGS) from Algorithm 7–8. For all preconditioners, we need to select the vector $w^{s,(n)}$ as discussed in Algorithm 5. Choice (a) is referred to as *fixed* because the

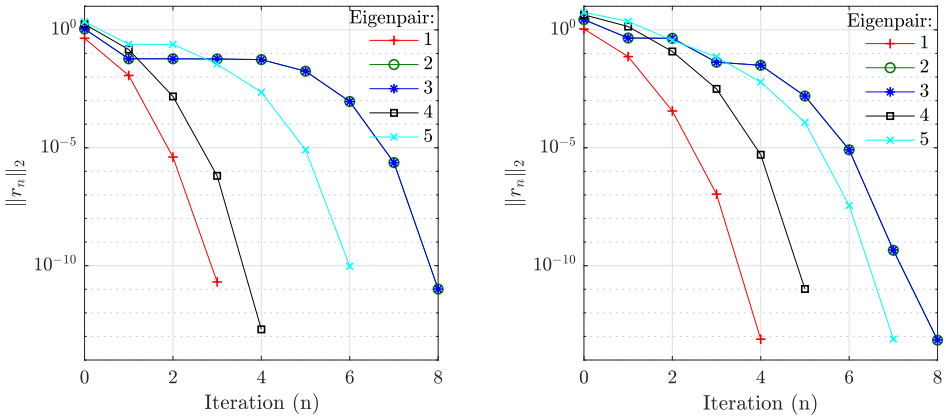

Figure 5.3. *Convergence history in terms of the nonlinear residual $\|r_n\|_2$ of the inexact line-search Newton method with CoV = 10% (left) and 25% (right).*

Table 5.2

Average number of MINRES/GMRES iterations for computing the five smallest eigenvalues and corresponding eigenvectors of the diffusion problem with CoV = 10% (left) and 25% (right) using the inexact line-search Newton method (Algorithm 4) with the stopping criteria $\|r_n\|_2 < 10^{-10}$.

	$CoV = 10\%$					$CoV = 25\%$				
Preconditioner	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
NMB (MINRES)	11.5	59.3	60.2	23.3	217.6	13.3	110.0	109.4	49.3	142.9
NMB (fixed)	11.3	71.5	59.9	29.6	120.5	15.2	79.3	79.5	43.8	101.1
NMB (updated)	13.3	28.9	27.8	16.2	43.0	19.0	68.9	64.5	87.3	122.9
cMB (fixed)	7.0	37.9	39.5	8.8	28.1	13.3	56.6	56.6	14.6	32.4
cMB (updated)	4.3	24.7	25.4	5.3	28.0	7.8	33.4	33.1	8.6	15.6
chGS($p_t = 1$)	2.3	17.9	17.1	2.8	15.4	3.3	18.3	18.1	2.8	18.9
chGS($p_t = 2$)	2.0	12.4	12.5	2.0	8.5	3.3	18.9	19.4	2.4	10.3
chGS(full)	2.0	13.8	13.5	2.0	12.3	3.3	15.1	15.1	2.8	14.4

vector $w^{s,(n)}$ is the corresponding eigenvector of the mean matrix A_1, and choice (b) is referred to as *updated* because the vector is updated after each step of Newton iteration. Only the variant (b) was used for the chGS preconditioner. We also need to specify (the solves with) the matrix M_1^s, in particular the choice of ϵ_M in (4.12). We report values of ϵ_M that, in our experience, worked best. For (both fixed) NMB and cMB, we set $\epsilon_M = 0.95$. For (updated) cMB and chGS, we set $\epsilon_M = 1$ and use the SVD decomposition as $\mathcal{M}_1 = \sum_{i=1}^{\text{rank}(\mathcal{M}_1)} d_i y_i z_i^T$ to solve linear systems in (4.15). If $\mathcal{M}_1$ appears to be numerically singular, the action of the inverse of $\mathcal{M}_1$ is replaced by a *pseudoinverse* $\sum_{i=1}^{\text{rank}(\mathcal{M}_1)} d_i^{-1} z_i y_i^T$. We note that with $p_t = 0$ the chGS preconditioner reduces to the (updated) cMB preconditioner. With all preconditioners the convergence was faster for simple eigenvalues, and the iteration counts increased in the course of Newton iteration. In both cases with $CoV = 10\%$ and 25% the constraint preconditioners outperform the mean-based preconditioners, and updating the vector $w^{s,(n)}$

Table 5.3

The number of GMRES iterations for computing the five smallest eigenvalues and corresponding eigenvectors of the diffusion problem with $CoV = 10\%$ (left) and 25% (right) using the inexact line-search Newton method (Algorithm 4) with preconditioners cMB (top) and chGS($p_t = 2$) (bottom) and with the stopping criteria $\|r_n\|_2 < 10^{-10}$.

	$CoV = 10\%$					$CoV = 25\%$				
	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
Nonlinear step	cMB (updated)									
1	2	2	2	1	1	2	2	2	1	1
2	4	8	8	3	35	5	8	8	3	3
3	7	10	10	6	35	9	11	11	6	10
4		13	14	11	21	15	18	17	12	12
5		23	26		17		34	34	21	16
6		45	46		23		75	74		22
7		72	72		41		86	86		45
8					51					
Nonlinear step	chGS($p_t = 2$)									
1	1	1	1	1	1	1	1	1	1	1
2	2	5	5	1	5	2	6	6	1	2
3	3	6	6	2	5	4	4	4	2	5
4		6	6	4	8	6	10	10	3	7
5		7	7		10		12	12	5	11
6		12	12		22		18	22		14
7		21	21				45	45		32
8		41	42				55	55		

improves the convergence. The lowest iteration counts were obtained with the chGS preconditioner, in particular with $p_t = 2$ and full, and we note that the computational cost with $p_t = 2$ is lower due to the truncation of the matrix-vector products. For these two preconditioners, Tables 5.2 and 5.3 show that solving the eigenvalue problem with higher CoV leads to only a slight increase in the number of iterations, and for simple eigenvalues the average iteration counts are only slightly larger than those of SISI.

A comparison of the inexact SISI and the inexact Newton iteration is provided by Figure 5.4, which shows the 2-norms of the residual indicator $\tilde{\tilde{r}}^{s,(n)} = [\tilde{r}_1^{s,(n)T}, \ldots, \tilde{r}_{n_\xi}^{s,(n)T}]^T$ from (3.11) and the part of the residual in the Newton method given by $F(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)})$; cf. (4.1). These quantities correspond to the residual of (2.9), through (2.12) and equivalent equation (2.14). It can be seen that it takes approximately the same number of steps for the Newton iteration to converge and for the SISI residuals to become flat in case of repeated eigenvalues, but more steps of SISI are needed for simple eigenvalues. With respect to the average number of Krylov iterations per a step of SISI and Newton iteration, the computational cost of the two methods is comparable for simple eigenvalues, but SISI is significantly more efficient for repeated eigenvalues. On the other hand, Newton iteration outperforms SISI in

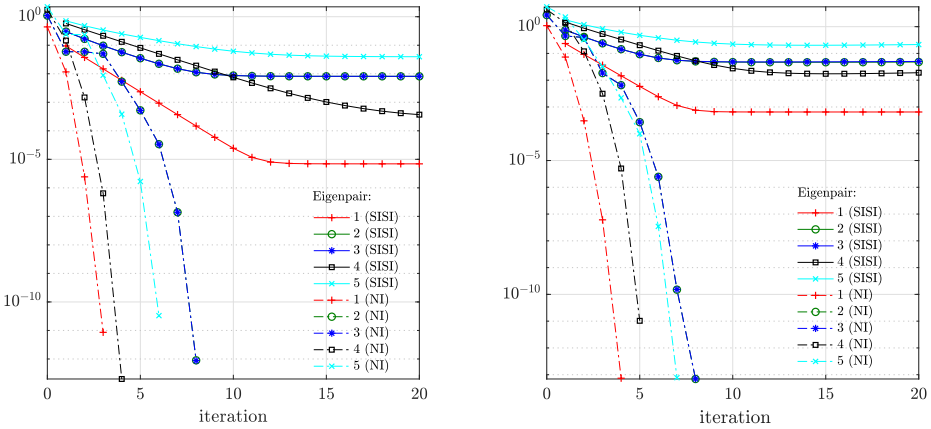

Figure 5.4. *Comparison of convergence of the inexact SISI in terms of residual indicator* $\|\bar{\tilde{r}}^{s,(n)}\|_2$ *and the inexact line-search Newton method in terms of* $\|F(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)})\|_2$ *with CoV* $= 10\%$ *(left) and* 25% *(right).*

Table 5.4

The first 10 *coefficients of the gPC expansion of the smallest eigenvalue of the diffusion problem with CoV* $= 10\%$ *(left) and* 25% *(right) using stochastic collocation (SC), SISI, and inexact line-search Newton method with the stopping criteria* $\|r_n\|_2 < 10^{-10}$. *Here* d *is the polynomial degree and* k *is the index of basis function in expansion* (2.5).

		$CoV = 10\%$			$CoV = 25\%$		
d	k	SC	SISI	NI	SC	SISI	NI
0	1	4.9431E+00	4.9431E+00	4.9431E+00	4.9052E+00	4.9052E+00	4.9052E+00
1	2	3.6197E-01	3.6197E-01	3.6197E-01	8.8127E-01	8.8127E-01	8.8127E-01
	3	1.4477E-13	-1.6489E-14	-7.9829E-16	2.0162E-13	-1.5964E-14	-7.3784E-16
	4	-6.6436E-13	-1.7135E-14	-1.3429E-15	9.9476E-14	-1.8588E-14	-1.4099E-15
2	5	1.8642E-02	1.8642E-02	1.8642E-02	1.1205E-01	1.1201E-01	1.1204E-01
	6	-5.4534E-13	-9.5178E-17	-7.4261E-17	-7.1498E-14	-2.9421E-15	-1.6150E-16
	7	-3.0909E-13	-1.1628E-15	-9.5249E-17	-9.4147E-14	-2.4433E-15	-3.7169E-16
	8	-1.5442E-03	-1.5442E-03	-1.5442E-03	-9.1479E-03	-9.1520E-03	-9.1493E-03
	9	-9.7700E-15	-1.1200E-15	1.3125E-18	-8.4643E-13	7.4442E-16	-1.2278E-17
	10	-1.5442E-03	-1.5442E-03	-1.5442E-03	-9.1479E-03	-9.1520E-03	-9.1493E-03

terms of accuracy of the solution residual, which is quite natural since Newton iteration is formulated as a minimization algorithm unlike SISI.

We also compare the gPC coefficients of eigenvalue expansions computed using the three different methods: the stochastic collocation method, the inexact SISI method, and the inexact line-search Newton method. In Table 5.4, we tabulate the first 10 coefficients of the gPC expansion of the smallest eigenvalues computed using the three methods. A good agreement of coefficients can be seen, in particular for coefficients with values much larger than zero, specifically with indices $k = 1, 2, 5, 8$, and 10. Figure 5.5 plots the probability density function (pdf) estimates of the five smallest eigenvalues obtained directly by Monte Carlo and the three

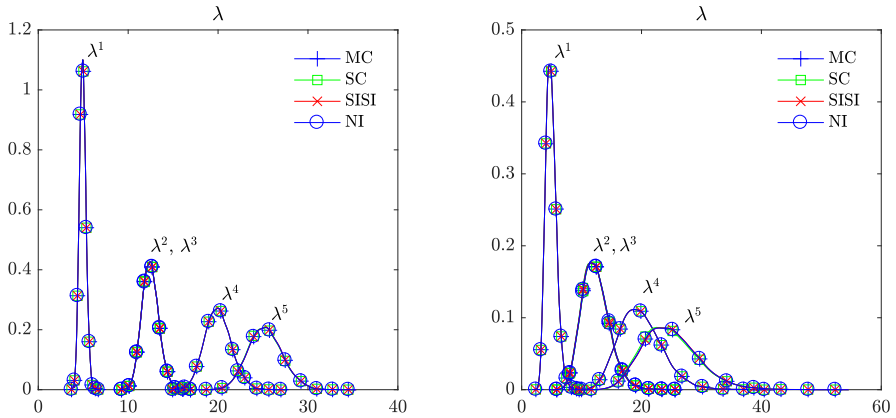

Figure 5.5. *pdf estimates of the five smallest eigenvalues with CoV = 10% (left) and 25% (right).*

Table 5.5

Average number of PCG iterations for computing the five smallest eigenvalues and corresponding eigenvectors of the diffusion problem with CoV = 10% (left) and 25% (right) using exact and inexact SISI (Algorithm 1).

		$CoV = 10\%$					$CoV = 25\%$				
		1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
MB	Inexact	6.45	3.90	3.90	4.60	3.75	8.60	5.55	5.55	6.05	4.75
	Exact	11.00	10.95	10.95	10.85	10.00	17.00	16.90	16.90	16.90	16.90
hGS (p_t=2)	Inexact	2.35	1.70	1.70	1.65	1.00	2.60	1.90	1.90	1.85	1.75
	Exact	3.00	3.00	3.00	3.00	3.00	5.00	5.00	5.00	4.00	4.00

methods, for which the estimates were obtained using MATLAB function `ksdensity` used for sampled gPC expansions. It can be seen that the pdf estimates overlap in all cases.

Inexact versus exact solves. We present numerical experiments that show the effectiveness of the inexact solvers by comparing them with the exact solvers, for which we fix the stopping tolerance of the PCG and GMRES methods to 10^{-12}. For the inexact methods we use the adaptive stopping tolerance given for SISI by (3.10) and for the Newton iteration by (4.11). A comparison of the inexact and exact solves in terms of the PCG iteration counts for computing the smallest five eigenvalues of the diffusion problem is shown in Table 5.5, and a comparison in terms of the GMRES iterations counts for computing the first and fourth smallest eigenvalues of the diffusion problem is shown in Table 5.6. In both cases, for given CoV and the choice of the preconditioner, we observe that the exact methods require more Krylov subspace iterations. It can be seen from Table 5.6 that virtually the same number of GMRES iterations is required in each nonlinear step of Newton iteration since the stopping tolerance of the exact solves is not adjusted to the nonlinear residual.

Effect of increasing the stochastic dimension. Table 5.7 shows the PCG iteration counts required to compute the smallest five eigenvalues of the diffusion problem for varying number of random variables $m_\xi = \{3, 5, 7\}$ with $CoV = 10\%$ and 25%, and Table 5.8 shows the GMRES iteration counts for computing the first and fourth smallest eigenvalues for the same

Table 5.6

The number of GMRES iterations for computing the first and fourth smallest eigenvalues and corresponding eigenvectors of the diffusion problem with CoV = 10% (left) and 25% (right) using exact and inexact line-search Newton methods (Algorithm 4) with preconditioners cMB (top) and chGS($p_t = 2$) (bottom), and with the stopping criteria $\|r_n\|_2 < 10^{-10}$.

	$CoV = 10\%$				$CoV = 25\%$			
	Inexact		Exact		Inexact		Exact	
	1st	4th	1st	4th	1st	4th	1st	4th
Nonlinear step	cMB (updated)							
1	2	1	13	16	2	1	22	39
2	4	3	13	15	5	3	22	27
3	7	6	14	16	9	6	22	27
4		11		16	15	12	22	27
5						21		27
Nonlinear step	chGS($p_t = 2$)							
1	1	1	5	7	1	1	7	16
2	2	1	5	7	2	1	8	15
3	3	2	6	7	4	2	8	11
4		4		7	6	3	8	11
5						5		10
6								10

Table 5.7

Average number of PCG iterations for computing the five smallest eigenvalues and corresponding eigenvectors of the diffusion problem with CoV = 10% (left) and 25% (right) for varying $m_\xi = \{3, 5, 7\}$ using inexact SISI (Algorithm 1).

		$CoV = 10\%$					$CoV = 25\%$				
m_ξ	Preconditioner	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
3	MB	6.45	3.90	3.90	4.60	3.75	8.60	5.55	5.55	6.05	4.75
	hGS ($p_t = 2$)	2.35	1.70	1.70	1.65	1.00	2.60	1.90	1.90	1.85	1.75
5	MB	6.50	3.90	3.90	4.50	3.85	8.00	4.85	4.85	6.50	4.70
	hGS ($p_t = 2$)	2.35	1.00	1.00	1.70	1.00	2.60	1.95	1.95	1.90	1.85
7	MB	6.40	3.95	3.95	4.55	3.85	8.00	4.85	4.85	6.50	4.70
	hGS ($p_t = 2$)	2.35	1.00	1.00	1.70	1.00	2.60	1.95	1.95	1.90	1.85

problem and setup. While in both cases we see a relatively small increase in iteration counts for larger CoV, increasing the stochastic dimension by setting larger m_ξ appears to have no effect on the iteration counts.

5.2. Stiffness of Mindlin plate with uniformly distributed Young's modulus. As the second example, we study eigenvalues of the stiffness of Mindlin plate with Young's modulus

Table 5.8

The number of GMRES iterations for computing the first and fourth smallest eigenvalues and corresponding eigenvectors of the diffusion problem with CoV = 10% (left) and 25% (right) for varying m_ξ using the inexact line-search Newton method (Algorithm 4) with preconditioners cMB (top) and chGS($p_t = 2$) (bottom), and with the stopping criteria $\|r_n\|_2 < 10^{-10}$.

	$CoV = 10\%$						$CoV = 25\%$					
m_ξ	3		5		7		3		5		7	
	1st	4th	1st	4th	1st	4th	1st	4th	1st	4th	1st	4th
Nonlinear step	cMB (updated)											
1	2	1	2	1	2	1	2	1	2	1	2	1
2	4	3	4	3	4	3	5	3	5	3	4	3
3	7	6	7	6	7	6	9	6	9	6	8	6
4		11		11		11	15	12	15	12	16	12
5								21		21		21
Nonlinear step	chGS($p_t = 2$)											
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	1	2	1	2	1	2	1	2	1
3	3	2	3	2	3	2	4	2	3	2	4	2
4		4		4		4	6	3	5	3	5	3
5								5		5		5

given by the stochastic expansion

$$E(x,\xi) = E_1 + \sum_{\ell=2}^{m_\xi+1} E_\ell \xi_{\ell-1}, \tag{5.6}$$

where $E_{\ell+1} = \sqrt{\lambda_\ell} v_\ell(x)$ with $\{(\lambda_\ell, v_\ell)\}_{\ell=1}^{m_\xi}$ are the eigenpairs of the eigenvalue problem associated with the covariance kernel,

$$C(X_1, X_2) = \frac{1}{3}\sigma_u^2 \exp\left(-\frac{|x_2 - x_1|}{L_x} - \frac{|y_2 - y_1|}{L_y}\right), \tag{5.7}$$

where L_x, L_y are as in (5.2), and σ_u is the standard deviation of the random field, the random variables ξ_ℓ are uniformly distributed over the interval $(-1, 1)$, $E_1 = 10920$ and other parameters are set as in [33]. The plate is discretized using 10×10 bilinear (Q4) finite elements with 243 physical degrees of freedom. We note that we consider only the stiffness matrix in the problem setup, and the mass matrix is taken as identity. For the uniform random variables, the set $\{\psi_k\}_{k=1}^{n_\xi}$ is given by Legendre polynomials and the Smolyak sparse grid with Gauss–Legendre quadrature is considered for the quadrature rule.

Table 5.9 shows the average numbers of PCG iterations required to solve linear system (3.3) with zero initial guess and the adaptive stopping criteria (3.10). As we observed in the results of the diffusion problem in Table 5.1, PCG with the hGS preconditioning requires less than half of the iteration counts with the MB preconditioner. Table 5.10 shows the average numbers

Table 5.9

Average number of PCG iterations for computing the five smallest eigenvalues and corresponding eigenvectors of the Mindlin plate problem with $CoV = 10\%$ (left) and 25% (right) using inexact SISI (Algorithm 1).

	$CoV = 10\%$					$CoV = 25\%$				
Preconditioner	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
MB	6.20	4.65	4.65	4.70	4.20	8.15	6.55	6.55	6.75	6.05
hGS ($p_t = 1$)	2.45	1.95	1.95	1.95	1.95	3.40	2.75	2.75	2.65	2.60
hGS ($p_t = 2$)	2.45	1.95	1.95	1.95	1.95	3.40	2.75	2.75	2.65	2.60
hGS (no trunc.)	2.45	1.95	1.95	1.95	1.95	3.40	2.75	2.75	2.65	2.60

Table 5.10

The average number of GMRES iterations for computing the first and fourth smallest eigenvalues and corresponding eigenvectors of the Mindlin plate problem with $CoV = 10\%$ and 25% for varying m_ξ (the number of random variables) using the inexact line-search Newton method (Algorithm 4) with preconditioners cMB (top) and chGS($p_t = 2$) (bottom), and with the stopping criteria $\|r_n\|_2 < 10^{-10}$.

	m_ξ	3		5		7		9	
		1st	4th	1st	4th	1st	4th	1st	4th
$CoV = 10\%$	NMB (fixed)	14.25	26.50	15.25	30.75	15.25	33.25	15.25	34.00
	NMB (updated)	12.00	12.00	15.00	13.75	15.00	14.00	15.00	14.25
	cMB (fixed)	10.25	10.25	10.75	11.25	11.00	11.50	11.00	11.75
	cMB (updated)	6.00	5.25	6.25	5.75	6.25	6.00	6.75	6.00
	chGS($p_t = 1$)	3.00	2.75	3.00	3.00	3.00	3.00	3.00	3.00
	chGS($p_t = 2$)	3.00	2.75	3.00	2.75	3.00	3.00	3.00	3.00
	chGS(full)	3.00	2.75	3.00	2.75	3.00	3.00	3.00	3.00
$CoV = 25\%$	NMB (fixed)	13.25	32.40	14.50	42.80	14.75	61.20	20.00	63.60
	NMB (updated)	14.75	16.60	19.75	29.17	26.40	40.00	27.60	42.67
	cMB (fixed)	11.25	18.17	12.50	22.33	12.50	28.83	17.40	29.50
	cMB (updated)	6.50	10.83	7.25	12.67	10.20	16.33	10.20	17.00
	chGS($p_t = 1$)	3.25	4.83	3.25	5.33	3.25	7.17	4.60	7.67
	chGS($p_t = 2$)	3.25	4.83	3.25	5.33	3.25	7.17	4.40	7.50
	chGS(full)	3.25	4.83	3.25	5.50	3.25	6.83	4.40	7.33

of GMRES iterations required to solve the linear systems at line 4 in Algorithm 4 with zero initial guess and the adaptive stopping criteria (4.11). As in the results of the diffusion problem in Table 5.2, we again observe that the updated versions of the preconditioners yield lower iteration counts compared to their fixed variants and the lowest counts are achieved with the chGS preconditioner. Increasing both CoV and stochastic dimension m_ξ leads to only a mild increase in iteration counts. Finally, Table 5.11 shows the first 10 coefficients of the gPC expansion of the smallest eigenvalue of the Mindlin plate. As for the solution coefficients of the diffusion problem shown in Table 5.4, a good agreement of coefficients can be seen also here.

Table 5.11

The first 10 *coefficients of the gPC expansion of the smallest eigenvalue of the Mindlin plate problem with* $CoV = 10\%$ *(left) and* 25% *(right) using stochastic collocation (SC), inexact SISI, and, the inexact line-search Newton method with the stopping criteria* $\|r_n\|_2 < 10^{-10}$. *Here* d *is the polynomial degree and* k *is the index of basis function in expansion* (2.5).

		$CoV = 10\%$			$CoV = 25\%$		
d	k	SC	SISI	NI	SC	SISI	NI
0	1	4.6271E-01	4.6271E-01	4.6271E-01	4.5784E-01	4.5784E-01	4.5784E-01
1	2	-2.2476E-02	-2.2476E-02	-2.2476E-02	-5.6737E-02	-5.6734E-02	-5.6735E-02
	3	6.6391E-14	-3.5389E-16	-8.0416E-18	-1.7453E-13	-6.5624E-16	-1.1174E-17
	4	3.2080E-13	-4.2037E-16	1.4672E-17	6.0396E-14	-4.8016E-16	2.5675E-17
2	5	-3.1659E-05	-3.1607E-05	-3.1634E-05	-2.5953E-04	-2.4582E-04	-2.5268E-04
	6	-7.8920E-14	1.7146E-16	-1.0762E-18	-2.2204E-16	9.0132E-16	4.9237E-18
	7	3.1186E-13	3.8511E-16	-4.5709E-19	-6.1270E-15	9.5916E-16	8.0412E-18
	8	-3.8995E-04	-3.8995E-04	-3.8995E-04	-2.5032E-03	-2.5021E-03	-2.5030E-03
	9	-2.8144E-14	-9.5150E-17	-5.8430E-19	1.1297E-13	-9.2077E-17	-1.5950E-18
	10	-3.8995E-04	-3.8995E-04	-3.8995E-04	-2.5032E-03	-2.5021E-03	-2.5030E-03

6. Conclusion.

We studied inexact methods for symmetric eigenvalue problems in the context of spectral stochastic finite element discretizations. The performance was compared using eigenvalue problems given by the stochastic diffusion equation with lognormally distributed diffusion coefficient and by the stiffness of Mindlin plate with Young's modulus depending on uniformly distributed random variables. Both problems were given in a two-dimensional physical domain. The methods were formulated on the basis of the SISI and the line-search Newton methods. In both formulations we obtained symmetric stochastic Galerkin matrices. In the first case the matrices were also positive definite, so the associated linear systems were solved using PCG method. For the PCG we used mean-based and hierarchical Gauss–Seidel preconditioners. The second preconditioner slightly decreased the overall iteration count, but in all cases only a handful of iterations were required for convergence per one step of SISI. The iteration count for PCG also did not appear to be sensitive to algebraic multiplicity of eigenvalues, but in terms of SISI we observed somewhat slower convergence for simple eigenvalues (i.e., those with algebraic multiplicity one). For the second method based on Newton iteration, we proposed several novel preconditioners adapted to the structure of the Jacobian matrices obtained from the stochastic Galerkin discretization. The linear systems were solved using the GMRES method (and in a few cases also MINRES) with various preconditioners. We analytically show that chGS preconditioner with a truncated matrix-vector product is the most efficient one for high-dimensional problems. The overall iteration count of GMRES was higher compared to PCG, in particular for eigenvalues with algebraic multiplicity larger than one. On the other hand, only a handful of iterations were required with the chGS preconditioner for simple eigenvalues. In terms of the iteration count of SISI and the Newton iteration, we observed that the two methods are comparable for simple eigenvalues, but SISI appeared more efficient for repeated eigenvalues. Increasing either the value of CoV or the stochastic dimension lead to only a slight increase of the number of iterations, in particular when the

constraint hierarchical preconditioners were used. Comparing the accuracy in terms of the solution residual, Newton iteration naturally outperformed SISI. Nevertheless both methods identified the coefficients of polynomial chaos expansion of the smallest eigenvalue in a close agreement and matched well those computed by the stochastic collocation. The probability density estimates of all eigenvalues matched, also with the direct Monte Carlo simulation.

From a user's perspective, SISI is straightforward to use and in combination with the stochastic modified Gram–Schmidt process allows us to compute coefficients of polynomial chaos expansions of several eigenvalues and eigenvectors, while the Newton iteration requires some setup of parameters for the line search and backtracking. On the other hand, Newton iteration may be more suitable when interior eigenvalues are sought, since SISI assumes that all smaller eigenvalues were deflated from the mean matrix.

Appendix A. Inexact Newton iteration. The inexact nonlinear iteration is based on the Newton–Krylov method, in which each step entails solving the linear system (4.10) by a Krylov subspace method followed by an update (4.9). But first, let us describe the evaluation of $F(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)})$ and $G(\bar{u}^{s,(n)})$. The vector $F(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)})$, defined by (4.2), consists of two terms: the first term is evaluated as

$$\mathbb{E}[\Psi\Psi^T \otimes A]\bar{u}^{s,(n)} = \sum_{\ell=1}^{n_a}(H_\ell \otimes A_\ell)\bar{u}^{s,(n)} = \operatorname{vec}\left(\sum_{\ell=1}^{n_a} A_\ell \bar{U}^{s,(n)} H_\ell^T\right),$$

which is the same as (3.4), and the second term is evaluated as

$$\mathbb{E}[((\bar{\lambda}^{s,(n)})^T\Psi)\Psi\Psi^T \otimes I_{n_x}]\bar{u}^{s,(n)} = \sum_{i=1}^{n_\xi}(\lambda_i^{s,(n)} H_i \otimes I_{n_x})\bar{u}^{s,(n)} = \operatorname{vec}\left(\sum_{i=1}^{n_\xi} \lambda_i^{s,(n)} \bar{U}^{s,(n)} H_i^T\right).$$

The vector $G(\bar{u}^{s,(n)})$, defined by (4.3), is evaluated as

$$G(\bar{u}^{s,(n)}) = \mathbb{E}\left[\Psi \otimes \left((\bar{u}^{s,(n)T}(\Psi\Psi^T \otimes I_{n_x})\bar{u}^{s,(n)}) - 1\right)\right],$$

where the ith row of $G(\bar{u}^{s,(n)})$ is

$$\begin{aligned}\left[G(\bar{u}^{s,(n)})\right]_i &= \mathbb{E}[\psi_i(\bar{u}^{s,(n)T}(\Psi\Psi^T \otimes I_{n_x})\bar{u}^{s,(n)}) - \psi_i] \\ &= \bar{u}^{s,(n)T}\mathbb{E}[\psi_i\Psi\Psi^T \otimes I_{n_x}]\bar{u}^{s,(n)} - \delta_{1i},\end{aligned}$$

and the first term above is evaluated as

$$\bar{u}^{s,(n)T}\mathbb{E}[\psi_i\Psi\Psi^T \otimes I_{n_x}]\bar{u}^{s,(n)} = \bar{u}^{s,(n)T}(H_i \otimes I_{n_x})\bar{u}^{s,(n)},$$

or, denoting the trace operator by tr, this term also can be evaluated as

$$\bar{u}^{s,(n)T}\mathbb{E}[\psi_i\Psi\Psi^T \otimes I_{n_x}]\bar{u}^{s,(n)} = \operatorname{tr}(\bar{U}^{s,(n)} H_i \bar{U}^{s,(n)T}) = \operatorname{tr}(\bar{U}^{s,(n)T}\bar{U}^{s,(n)} H_i).$$

Remark A.1. For completeness, let us describe a possible setup of the Jacobian matrices in (4.8) or (4.10). Block (4.5) can be set up as

$$\mathbb{E}[\Psi\Psi^T \otimes A] - \mathbb{E}[((\bar{\lambda}^{s,(n)})^T\Psi)\Psi\Psi^T \otimes I_{n_x}] = \sum_{i=1}^{n_a} H_i \otimes A_i - \sum_{i=1}^{n_\xi} (\lambda_i^{s,(n)} H_i \otimes I_{n_x}). \tag{A.1}$$

Block (4.6) can be set up as

$$\mathbb{E}[\Psi^T \otimes (\Psi\Psi^T \otimes I_{n_x})\bar{u}^{s,(n)}] = \mathbb{E}[(\psi_1\Psi\Psi^T \otimes I_{n_x})\bar{u}^{s,(n)}, \dots, (\psi_{n_\xi}\Psi\Psi^T \otimes I_{n_x})\bar{u}^{s,(n)}], \tag{A.2}$$

and the ith column of this block is

$$\mathbb{E}[(\psi_i\Psi\Psi^T \otimes I_{n_x})\bar{u}^{s,(n)}] = (H_i \otimes I_{n_x})\bar{u}^{s,(n)} = \operatorname{vec}\left(\bar{U}^{s,(n)} H_i^T\right). \tag{A.3}$$

Finally, block (4.7) is the transpose of (4.6) scaled by a factor of -2; cf. (4.10).

In implementation, the explicit setup described in Remark A.1 is avoided because Krylov subspace methods require only matrix-vector products. Let us write a product with Jacobian matrix from (4.10) at step n of the nonlinear iteration as

$$J(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}) \begin{bmatrix} \delta\bar{u} \\ \delta\bar{\lambda} \end{bmatrix}, \qquad \text{where } J(\bar{u}^{s,(n)}, \bar{\lambda}^{s,(n)}) = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}, \tag{A.4}$$

with A and B^T denoting the matrices in (4.5) and (4.6), resp. Then,

$$A\,\delta\bar{u} = \left(\sum_{\ell=1}^{n_a} H_\ell \otimes A_\ell - \sum_{i=1}^{n_\xi} H_i \otimes \lambda_i^{s,(n)} I_{n_x}\right)\delta\bar{u} = \operatorname{vec}\left(\sum_{\ell=1}^{n_a} A_\ell \delta\bar{U} H_\ell^T - \sum_{i=1}^{n_\xi} \lambda_i^{s,(n)} \delta\bar{U} H_i^T\right), \tag{A.5}$$

$$B^T\,\delta\bar{\lambda} = -\sum_{i=1}^{n_\xi} \delta\lambda_i \mathbb{E}[\Psi^T \otimes (\Psi\Psi^T \otimes I_{n_x})]\bar{u}^{s,(n)} = -\operatorname{vec}\left(\sum_{i=1}^{n_\xi} \delta\lambda_i \bar{U}^{s,(n)} H_i^T\right), \tag{A.6}$$

and

$$B\,\delta\bar{u} = -\mathbb{E}[\Psi \otimes (\bar{u}^{s,(n)T}(\Psi\Psi^T \otimes I_{n_x}))]\delta\bar{u} = -\begin{bmatrix} \bar{u}^{s,(n)T}(H_1 \otimes I_{n_x})\delta\bar{u} \\ \vdots \\ \bar{u}^{s,(n)T}(H_{n_\xi} \otimes I_{n_x})\delta\bar{u} \end{bmatrix}, \tag{A.7}$$

where the ith row can be equivalently evaluated as $\operatorname{tr}(H_i^T \bar{U}^{s,(n)T}\delta\bar{U})$.

Appendix B. Matrix-vector product in the chGS preconditioner. The matrix-vector product with subblocks of the stochastic Jacobian matrices is performed as in (A.4)–(A.7). For example, the matrix-vector product with a subblock of the A-part of the Jacobian matrix (cf. (A.5)) can be written as

$$\sum_{t\in\mathcal{I}_t} (\left[h_{t,(\ell)(k)}\right] \otimes A_t) v_{(k)}^s = \operatorname{vec}\left(\sum_{t\in\mathcal{I}_t} A_t V_{(k)}^s \left[h_{t,(k)(\ell)}\right]\right), \tag{B.1}$$

$$\sum_{t\in\mathcal{I}_t} (\left[h_{t,(\ell)(k)}\right] \otimes \lambda_t^{s,(n)} I_{n_x}) v_{(k)}^s = \operatorname{vec}\left(\sum_{t\in\mathcal{I}_t} \lambda_t^{s,(n)} V_{(k)}^s \left[h_{t,(k)(\ell)}\right]^T\right), \tag{B.2}$$

where $V^s_{(k)}$ is a subset of the columns of V^s specified by the index set (k). We note that the matrix-vector products in (B.2) depend on the eigenvalue approximation at step n of Newton iteration. The truncation of the matrix-vector products, indicated by summing up over index set $\mathcal{I}_t$, is performed using the same strategy as in Algorithm 3.

Appendix C. Computational cost. Here, we discuss the computational costs of the GMRES method with different preconditioners. The most computationally intensive operations in the GMRES are matrix-vector products and preconditioning. Each step of the GMRES thus requires cost of $c_{\text{mvp}} + c_{\text{prec}}$, where

c_{mvp}: cost of matrix-vector products described in eqs. (A.5)–(A.7),

c_{prec}: cost of preconditioning.

Then the total computational cost of the GMRES is $n_{\text{iter}}(c_{\text{mvp}}+c_{\text{prec}})$, where n_{iter} refers to the total iteration count. The cost of matrix-vector products is largely due to evaluating the first term, $\sum_{\ell=1}^{n_a} A_\ell \delta\bar{U} H_\ell^T$, in (A.5) and, thus, the cost can be approximately measured as $c_{\text{mvp}} \approx n_a(c_x+c_\xi)$, where c_x and c_ξ are the costs for matrix-matrix products associated with A_ℓ and H_ℓ, resp., in the expression $A_\ell \delta\bar{U} H_\ell^T$. For the preconditioning, we compare the two most efficient preconditioners, cMB and truncated chGS with $p_t < p$. Let us denote the computational cost of a solve with $\mathcal{M}_1$ in (4.15) by $c_{\mathcal{M}_1}$. The cMB preconditioner (Algorithm 5) requires $c_{\text{prec}} = c_{\mathcal{M}_1}$ and the computational cost of the GMRES with the cMB preconditioner can be approximated as

$$c_{\text{cMB}} = n_{\text{iter}}^{\text{cMB}}(\underbrace{n_a(c_x + c_\xi)}_{c_{\text{mvp}}} + \underbrace{c_{\mathcal{M}_1}}_{c_{\text{prec}}}).$$

The chGS preconditioner (Algorithm 7–8) requires two truncated matrix-vector products (B.1)–(B.2), where the truncation is specified by the set $\mathcal{I}_t$ and applications of the cMB preconditioners for $2p$ times (in the forward and the backward sweep of the Algorithm 7–8) and, thus, the cost can be assessed as $c_{\text{prec}} \approx 2n_t(c_x+c_\xi)+2pc_{\mathcal{M}_1}$, where $n_t = \dim(\mathcal{I}_t)$. Now we can write the total computational cost of the GMRES method with the chGS preconditioner as

$$c_{\text{chGS}} = n_{\text{iter}}^{\text{chGS}}(\underbrace{n_a(c_x + c_\xi)}_{c_{\text{mvp}}} + \underbrace{2n_t(c_x + c_\xi) + 2p\, c_{\mathcal{M}_1}}_{c_{\text{prec}}}).$$

From the analytic expressions of the costs, we can see that c_{prec} for chGS is larger than c_{prec} for cMB as chGS requires two truncated matrix-vector products $2n_t(c_x + c_\xi)$ at each GMRES iteration. On the other hand, typically, $n_{\text{iter}}^{\text{cMB}} \gg n_{\text{iter}}^{\text{chGS}}$, that is the cMB preconditioner requires more iterations. Specifically, the cMB preconditioner needs to perform extra $n_{\text{iter}}^{\text{cMB}} - n_{\text{iter}}^{\text{chGS}}$ matrix-vector products, with cost $n_a(c_x + c_\xi)$. To compare the computational costs of the two methods cMB and chGS($p_t = 2$) in practice, we tabulate the values of $n\xi$, n_a, and n_t for varying $m_\xi = \{3, 5, 7\}$ and $p = \{3, 4, 5\}$; see Table C.1. For problems with coefficients characterized by linear expansion in ξ such as (5.6), cMB could be less expensive since n_a is typically smaller than n_t. For problems with coefficients characterized by more general

Table C.1

The number of terms, n_a, in the expansion (2.2) modeling linear and nonlinear coefficient expansions such as (5.1) and (5.6), resp., and the number of terms n_t in the truncation set $\mathcal{I}_t$ with $p_t = 2$ for varying number of random variables m_ξ and the maximum polynomial degree p of the solution expansion (2.5).

m_ξ	3			5			7		
p	3	4	5	3	4	5	3	4	5
n_ξ	20	35	56	56	126	252	120	330	792
n_a (nonlinear)	84	165	286	462	1287	3003	1716	6435	19448
n_a (linear)	4			6			8		
n_t	10			21			36		

(nonlinear) expansions such as (5.1), chGS with truncated matrix-vector products become more cost efficient because n_a grows exponentially as m_ξ and p become larger, whereas n_t remains small. Note that an analogous comparison can be made for chGS and NMB.

Acknowledgments. We would like to thank Prof. Howard C. Elman for sharing his pearls of wisdom with us and many fruitful discussions. We would also like to thank the anonymous referees for insightful comments.

REFERENCES

[1] P. Benner, A. Onwunta, and M. Stoll, *A low-rank inexact Newton-Krylov method for stochastic eigenvalue problems*, Comput. Methods Appl. Math., to appear, https://doi.org/10.1515/cmam-2018-0030.

[2] D. Brockway, P. Soran, and P. Whalen, *Monte-Carlo eigenvalue calculation*, in Monte-Carlo Methods and Applications in Neutronics, Photonics and Statistical Physics, Lecture Notes in Phys. 240, R. Alcouffe, R. Dautray, A. Forster, G. Ledanois, and B. Mercier, eds., Springer, Berlin, 1985, pp. 378–387, https://doi.org/10.1007/BFb0049064.

[3] X. Chen, Y. Kawamura, and T. Okada, *Solution of stochastic eigenvalue problem by improved stochastic inverse power method (I-SIPM)*, J. Marine Sci. Technol., (2017), https://doi.org/10.1007/s00773-017-0513-3.

[4] H. C. Elman and T. Su, *Low-Rank Solution Methods for Stochastic Eigenvalue Problems*, arXiv:1803.03717v1, 2018.

[5] R. Ghanem, *The nonlinear Gaussian spectrum of log-normal stochastic processes and variables*, J. Appl. Mech., 66 (1999), pp. 964–973, https://doi.org/10.1115/1.2791806.

[6] R. G. 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.

[7] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer, New York, 1991.

[8] D. Ghosh, *Application of the random eigenvalue problem in forced response analysis of a linear stochastic structure*, Arch. Appl. Mech., 83 (2013), pp. 1341–1357.

[9] D. Ghosh and R. G. Ghanem, *Stochastic convergence acceleration through basis enrichment of polynomial chaos expansions*, Internat. J. Numer. Methods Engrg., 73 (2008), pp. 162–184.

[10] D. Ghosh and R. G. Ghanem, *An invariant subspace-based approach to the random eigenvalue problem of systems with clustered spectrum*, Internat. J. Numer. Methods Engrg., 91 (2012), pp. 378–396.

[11] G. H. Golub and Q. Ye, *Inexact inverse iteration for generalized eigenvalue problems*, BIT, 40 (2000), pp. 671–684, https://doi.org/10.1023/A:1022388317839.

[12] H. Hakula, V. Kaarnioja, and M. Laaksonen, *Approximate methods for stochastic eigenvalue problems*, Appl. Math. Comput., 267 (2015), pp. 664–681, https://doi.org/10.1016/j.amc.2014.12.112.

[13] H. Hakula and M. Laaksonen, *Asymptotic Convergence of Spectral Inverse Iterations for Stochastic Eigenvalue Problems*, arXiv:1706.03558, 2017.
[14] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.
[15] M. Kamiński, *The Stochastic Perturbation Method for Computational Mechanics*, Wiley, New York, 2013.
[16] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.
[17] M. Kleiber, *The Stochastic Finite Element Method: Basic Perturbation Technique and Computer Implementation*, Wiley, New York, 1992.
[18] O. Le Maître and O. M. Knio, *Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics*, Sci. Comput., Springer, Berlin, 2010.
[19] K. Lee, H. C. Elman, and B. Sousedík, *A Low-Rank Solver for the Navier-Stokes Equations with Uncertain Viscosity*, arXiv:1710.05812, 2017.
[20] G. J. Lord, C. E. Powell, and T. Shardlow, *An Introduction to Computational Stochastic PDEs*, Cambridge Texts Appl. Math. 50, Cambridge University Press, Cambridge, UK, 2014.
[21] H. G. Matthies and A. Keese, *Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 1295–1331, https://doi.org/10.1016/j.cma.2004.05.027.
[22] H. Meidani and R. G. Ghanem, *A stochastic modal decomposition framework for the analysis of structural dynamics under uncertainties*, in Proceedings of the 53rd Conference on Structures, Structural Dynamics, and Materials, Honolulu, HI, 2012.
[23] H. Meidani and R. G. Ghanem, *Spectral power iterations for the random eigenvalue problem*, AIAA J., 52 (2014), pp. 912–925, https://doi.org/10.2514/1.J051849.
[24] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972, https://doi.org/10.1137/S1064827599355153.
[25] M. P. Nightingale and C. J. Umrigar, *Monte Carlo Eigenvalue Methods in Quantum Mechanics and Statistical Mechanics*, Wiley, New York, 2007, pp. 65–115.
[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, Berlin, 1999.
[27] E. Pagnacco, E. Souza de Cursi, and R. Sampaio, *Subspace inverse power method and polynomial chaos representation for the modal frequency responses of random mechanical systems*, Comput. Mech., 58 (2016), pp. 129–149, https://doi.org/10.1007/s00466-016-1285-z.
[28] B. Pascual and S. Adhikari, *Hybrid perturbation-polynomial chaos approaches to the random algebraic eigenvalue problem*, Comput. Methods Appl. Mech. Engrg., 217–220 (2012), pp. 153–167, https://doi.org/10.1016/j.cma.2012.01.009.
[29] M. F. Pellissetti and R. G. Ghanem, *Iterative solution of systems of linear equations arising in the context of stochastic finite elements*, Adv. Eng. Softw., 31 (2000), pp. 607–616.
[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, https://doi.org/10.1093/imanum/drn014.
[31] H. Pradlwarter, G. Schuller, and G. Szekely, *Random eigenvalue problems for large systems*, Comput. Struct., 80 (2002), pp. 2415–2424, https://doi.org/10.1016/S0045-7949(02)00237-7.
[32] M. Shinozuka and C. J. Astill, *Random eigenvalue problems in structural analysis*, AIAA J., 10 (1972), pp. 456–462.
[33] 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, https://doi.org/10.1137/140999359.
[34] B. Sousedík and H. C. Elman, *Stochastic Galerkin methods for the steady-state Navier-Stokes equations*, J. Comput. Phys., 316 (2016), pp. 435–452, https://doi.org/10.1016/j.jcp.2016.04.013.
[35] B. Sousedík and R. G. Ghanem, *Truncated hierarchical preconditioning for the stochastic Galerkin FEM*, Int. J. Uncertain. Quantif., 4 (2014), pp. 333–348, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2014007353.
[36] B. Sousedík, R. G. Ghanem, and E. T. Phipps, *Hierarchical Schur complement preconditioner for the stochastic Galerkin finite element methods*, Numer. Linear Algebra Appl., 21 (2014), pp. 136–151, https://doi.org/10.1002/nla.1869.
[37] 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, https://doi.org/10.1002/nme.1712.

[38] J. vom Scheidt and W. Purkert, *Random Eigenvalue Problems*, North Holland Ser. Probab. Appl. Math., North Holland, Amsterdam, 1983.
[39] A. J. Wathen, *Preconditioning*, Acta Numer., 24 (2015), pp. 329–376, https://doi.org/10.1017/S0962492915000021.
[40] M. Williams, *A method for solving a stochastic eigenvalue problem applied to criticality*, Ann. Nuclear Energy, 37 (2010), pp. 894–897, http://www.sciencedirect.com/science/article/pii/S0306454910000861.
[41] M. Williams, *A method for solving stochastic eigenvalue problems*, Appl. Math. Comput., 215 (2010), pp. 3906–3928, http://www.sciencedirect.com/science/article/pii/S0096300309010285.
[42] M. Williams, *A method for solving stochastic eigenvalue problems* II, Appl. Math. Comput., 219 (2013), pp. 4729–4744, http://www.sciencedirect.com/science/article/pii/S0096300312011058.
[43] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*, Princeton University Press, Princeton, NJ, 2010.
[44] D. Xiu and G. E. Karniadakis, *The Wiener-Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.