AN INEXACT NEWTON-KRYLOV METHOD FOR STOCHASTIC EIGENVALUE PROBLEMS

PETER BENNER[*], AKWUM ONWUNTA[†] AND MARTIN STOLL[‡]

Abstract. This paper aims at the efficient numerical solution of stochastic eigenvalue problems. Such problems often lead to prohibitively high dimensional systems with tensor product structure when discretized with the stochastic Galerkin method. Here, we exploit this inherent tensor product structure to develop a globalized low-rank inexact Newton method with which we tackle the stochastic eigenproblem. We illustrate the effectiveness of our solver with numerical experiments.

Key words. Stochastic Galerkin system, Krylov methods, eigenvalues, eigenvectors, low-rank solution, preconditioning.

AMS subject classifications. 35R60, 60H15, 60H35, 65N22, 65F10, 65F50

1. Introduction.

In many areas of computational science and engineering, eigenvalue problems play an important role. This is, for example, the case in structural mechanics, where eigenvalue problems typically appear in the context of vibrations and buckling. For deterministic problems, there are currently well-established algorithms dedicated to the computation of eigenvalues and eigenvectors, see, e.g., [20]. However, in many cases of practical interest, physical characteristics are not always completely deterministic. For instance, the stiffness of a plate can locally be reduced by material imperfections, or the velocity of a flow can be influenced by turbulence. In recent times, an increasingly important way to model such problems is by describing the uncertain problem characteristics more realistically using random variables. By doing so, one would then gain more insight regarding the effect of the uncertainties on the model. This approach then leads to a stochastic eigenvalue problem (SEVP).

It is worth pointing out that the consequence of modeling the input parameters of a physical problem as random variables is that the desired output naturally inherits the stochasticity in the model. Generally speaking, there are two broad techniques for analyzing and quantifying uncertainty in a given model: simulation-based methods and expansion-based methods. In the simulation- (or sampling-) based methods, the stochastic moments of the eigenvalues and eigenvectors are obtained by generating ensembles of random realizations for the prescribed random inputs and utilizing repetitive deterministic solvers for each realization. Prominent among this class of methods is the classical Monte Carlo method. This method has been applied to many problems and its implementations are straightforward. It is (formally) independent of the dimensionality of the random space; that is, it is independent of the number of random variables used to characterize the random inputs. It does, however, exhibit a very

[*] Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg, Germany, (benner@mpi-magdeburg.mpg.de)

[†] Corresponding author; Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg, Germany, (onwunta@mpi-magdeburg.mpg.de)

[‡] Numerical Linear Algebra for Dynamical Systems Group, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg, Germany, (stollm@mpi-magdeburg.mpg.de); Technische Universität Chemnitz, Faculty of Mathematics, Professorship Scientific Computing, 09107 Chemnitz, Germany, (martin.stoll@mathematik.tu-chemnitz.de)

slow convergence rate [39]. To accelerate its convergence, several techniques have been developed: the multilevel Monte Carlo method [10], the quasi-Monte Carlo method [26], the Markov chain Monte Carlo method [19], etc. Although these methods can improve the efficiency of the traditional Monte Carlo method, additional restrictions are imposed based on their specific designs and their applicability is limited.

The expansion-based methods for uncertainty analysis and quantification are often designed to retain the advantages of Monte Carlo simulations; in particular, they enable one to compute the full statistical characteristics of the solution, while reducing the simulation time. A typical example of the expansion-based methods are the spectral stochastic finite element methods (SFEM) [18, 30]; they rely on the approximation of the random eigenvalues and eigenvectors by projecting them onto a global basis and are considerably less expensive than the simulation-based methods. We will, in particular, employ mainly SFEM in this paper.

During the last two decades, there has been a lot of research on SFEM for uncertainty analysis and quantification for solutions of partial differential equations [3, 4, 30]. However, SFEM for SEVPs has been so far much less addressed in the literature. To a great extent, most research on SEVPs has, in fact, focused more on simulation-based techniques [31, 35]. Nevertheless, relatively few attempts have been made to approximate the stochastic moments of both the eigenvalues and eigenvectors through the use of spectral methods [17, 21, 42]. In [42], the authors propose algorithms based on the inverse power method together with spectral methods for computing approximate eigenpairs of both symmetric and non-symmetric SEVPs. The method proposed in [17] essentially rewrites the eigenvalue problem resulting from a spectral discretization (which we henceforth refer to as stochastic Galerkin method (SGM)) as a set of nonlinear equations with tensor product structure, which are then solved using the Newton-Raphson method. In the spirit of [17], this paper presents an algorithm to determine the spectral expansions of the eigenvalues and the eigenvectors based on a Newton's method and SGM. However, unlike [17], this work specifically focuses on the use of a *globalized low-rank inexact Newton method* to tackle the eigenproblem.

Now, recall that under certain conditions, the iterates produced by the Newton's method converge quadratically to a solution x^* of a given nonlinear system, and those of the inexact Newton method can obtain super-linear convergence [1, 14, 36]. Both cases, however, assume an initial guess x_0 sufficiently close to x^*. Generally speaking, globalizing the inexact Newton method means augmenting the method with additional conditions on the choices of iterates $\{x_k\}$ to enhance the likelihood of convergence to x^*, see e.g. [36] for details of different globalization techniques. The advantages of globalization notwithstanding[1], a drawback of Newton-type methods is that for fairly large eigenproblems such as the SEVPs considered in this work, they require considerable computational effort to solve the linear system arising from each Newton step. The aim of this paper is therefore to mitigate this computational challenge by exploiting the inherent tensor product structure in the SEVP to tackle the stochastic eigenproblem. More precisely, we combine low-rank Krylov solvers with a globalized inexact Newton method to efficiently solve SEVPs.

The rest of the paper is organized as follows. In Section 2, we present the problem that we would like to solve in this paper. Next, Section 3 gives an overview of the

[1] It is important to note that no globalization strategy determines a sequence that converges to a solution for every problem; rather, globalization techniques are essentially used only to enhance the likelihood of convergence to some solution of the problem.

stochastic Galerkin method on which we shall rely to discretize our model problem. After discussing our globalized low-rank inexact Newton solver in Section 4, we proceed to Section 5 to provide the numerical results to buttress the efficiency of the proposed solver, while Section 6 draws some conclusions on the findings in this work.

2. Problem statement. Let the triplet $(\Omega, \mathcal{F}, \mathbb{P})$ denote a complete probability space, where Ω is the set of elementary events, $\mathcal{F} \subset 2^{\Omega}$ is a σ-algebra on Ω and $\mathbb{P} : \mathcal{F} \to [0,1]$ is an appropriate probability measure. Let $\mathcal{D} \subset \mathbb{R}^d$ with $d \in \{1,2,3\}$, be a bounded physical domain. In this paper, we consider the following eigenvalue problem for an N_x-dimensional real symmetric random matrix

$$\mathcal{A}(\omega)\varphi(\omega) = \lambda(\omega)\varphi(\omega), \tag{2.1}$$

subject to the normalization condition

$$\varphi(\omega)^T\varphi(\omega) = 1, \tag{2.2}$$

where

$$\lambda(\omega) \in \mathbb{R}, \quad \varphi(\omega) \in \mathbb{R}^{N_x}, \quad \mathcal{A}(\omega) \in \mathbb{R}^{N_x \times N_x}, \quad \omega \in \Omega.$$

The matrix $\mathcal{A}(\omega)$ represents, for example, the stiffness matrix in a structural mechanics problem [17]. In this case, the stochasticity in $\mathcal{A}(\omega)$ is often inherited from the randomness in the underlying physical system such as elastic and dynamic parameters. Moreover, we assume that the randomness in the model is induced by a prescribed finite number of random variables $\xi := \{\xi_1, \xi_2, \ldots, \xi_m\}$, where $m \in \mathbb{N}$ and $\xi_i(\omega) : \Omega \to \Gamma_i \subseteq \mathbb{R}$. We also make the simplifying assumption that each random variable is independent and characterized by a probability density function $\rho_i : \Gamma_i \to [0,1]$. If the distribution measure of the random vector $\xi(\omega)$ is absolutely continuous with respect to the Lebesgue measure, then there exists a joint probability density function $\rho : \Gamma \to \mathbb{R}^+$, where $\rho(\xi) = \prod_{i=1}^m \rho_i(\xi_i)$, and $\rho \in L^\infty(\Gamma)$. Furthermore, we can now replace the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with $(\Omega, \mathbb{B}(\Gamma), \rho(\xi)d\xi)$, where $\mathbb{B}(\Gamma)$ denotes the Borel σ-algebra on Γ and $\rho(\xi)d\xi$ is the finite measure of the vector ξ. Then, the expected value of the product of measurable functions on Γ determines the Hilbert space $L^2_\rho(\Omega, \mathbb{B}(\Gamma), \rho(\xi)d\xi)$, with inner product

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

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

In this paper, we assume that the random matrix $\mathcal{A}(\omega)$ in (2.1) admits the representation

$$\mathcal{A}(\omega) = A_0 + \sum_{k=1}^{m} \xi_k(\omega)A_k, \quad m \in \mathbb{N}, \;\; A_k \in \mathbb{R}^{N_x \times N_x}, \;\; k = 0, 1, \ldots, m, \tag{2.3}$$

where $\{\xi_k\}$ are independent random variables. This is indeed the case if a Karhunen-Loève expansion (KLE) is used to discretize random stiffness properties; see, e.g., [21, 30, 27]. Furthermore, the stochastic eigenvalues and eigenvectors in this work are approximated using the so-called *generalized polynomial chaos expansion* (gPCE) [3, 27, 43]. More precisely, the ℓth random eigenvalue and eigenvector are given, respectively, as

$$\lambda_\ell(\omega) = \sum_{k=0}^{N_\xi - 1} \lambda_k^{(\ell)}\psi_k(\xi(\omega)), \quad \lambda_k^{(\ell)} \in \mathbb{R}, \tag{2.4}$$

and

$$\varphi_\ell(\omega) = \sum_{k=0}^{N_\xi-1} \varphi_k^{(\ell)} \psi_k(\xi(\omega)), \quad \varphi_k^{(\ell)} \in \mathbb{R}^{N_x}, \tag{2.5}$$

where $\{\psi_i\}$ are multidimensional Legendre basis polynomials expressed as functions of the random vector ξ, with properties

$$\mathbb{E}(\psi_k) = \delta_{k0} \quad \text{and} \quad \mathbb{E}(\psi_j \psi_k) = \delta_{jk}\mathbb{E}(\psi_k^2).$$

The spectral expansions (2.4) and (2.5) are the gPCE of the random quantities $\lambda_\ell(\omega)$ and $\varphi_\ell(\omega)$, respectively. Throughout this paper, we use normalized Legendre basis polynomials in which case $\mathbb{E}(\psi_i^2) = 1$, so that $\mathbb{E}(\psi_i\psi_j) = \delta_{ij}$. We remark here that N_ξ in (2.4) and (2.5) is chosen in such a way that $N_\xi > m$. In particular, using total degree Legendre polynomials ψ_i yields

$$N_\xi = (m+r)!/m!r!, \tag{2.6}$$

where r is the degree of ψ_i, see e.g. [30].

In what follows, we will, for notational convenience, omit the index ℓ associated with the ℓth eigenpair. It is pertinent to note here the difference between the structure of a deterministic and a random eigenproblem. In the deterministic case, a typical eigenpair is of the form (λ, φ), where $\lambda \in \mathbb{R}$ and $\varphi \in \mathbb{R}^{N_x}$, with N_x denoting the size of the deterministic matrix $\mathcal{A}$. In the stochastic case, however, the eigenpair corresponding to ℓth physical mode consists of the set

$$x := \{\lambda_0, \lambda_1, \ldots, \lambda_{N_\xi-1}, \varphi_0, \varphi_1, \ldots, \varphi_{N_\xi-1}\}. \tag{2.7}$$

3. Stochastic Galerkin method.

The stochastic Galerkin method is based on the projection

$$\langle \mathcal{A}\varphi, \psi_k \rangle = \langle \lambda\varphi, \psi_k \rangle, \quad k = 0, \ldots, N_\xi - 1, \quad \ell = 1, \ldots N_x. \tag{3.1}$$

Substituting (2.3), (2.4), and (2.5) into (3.1) yields the nonlinear algebraic equations

$$\sum_{i=0}^{m-1} \sum_{j=0}^{N_\xi-1} \mathbb{E}(\xi_i \psi_j \psi_k) A_i \varphi_j = \sum_{i=0}^{N_\xi-1} \sum_{j=0}^{N_\xi-1} \mathbb{E}(\psi_i \psi_j \psi_k) \lambda_i \varphi_j, \; k = 0, \ldots, N_\xi - 1, \tag{3.2}$$

which can be rewritten in Kronecker product notation as

$$\underbrace{\left[G_0 \otimes A_0 + \sum_{k=1}^{m} G_k \otimes A_k \right]}_{:=A} \Phi = \left[\sum_{k=0}^{N_\xi-1} \lambda_k \underbrace{(H_k \otimes \mathbf{I})}_{:=B_k} \right] \Phi, \tag{3.3}$$

where $\mathbf{I}$ is the identity matrix and

$$\begin{cases} G_0 = \operatorname{diag}\left(\langle \psi_0^2 \rangle, \langle \psi_1^2 \rangle, \ldots, \left\langle \psi_{N_\xi-1}^2 \right\rangle \right), \\ G_k(i,j) = \langle \psi_i \psi_j \xi_k \rangle, \quad k = 1, \ldots, m, \\ H_k(i,j) = \langle \psi_i \psi_j \psi_k \rangle, \quad k = 0, \ldots, N_\xi - 1, \\ \Phi = \left(\varphi_0, \varphi_1, \ldots, \varphi_{N_\xi-1} \right) \in \mathbb{R}^{N_x N_\xi}. \end{cases} \tag{3.4}$$

Here, the block A_0 (as well as A itself) is symmetric and positive definite; it captures the mean information in the model and appears on the diagonal blocks of A, whereas the other blocks $A_k,\ k = 1, \ldots, m$, represent the fluctuations in the model. Moreover, the random variables $\{\xi_k\}_{k=1}^m$ are centered, normalized and independent; see e.g., [30].

Recalling that $N_\xi > m$, we see that (3.3) can also be expressed as

$$\underbrace{\sum_{k=0}^{N_\xi - 1} \left[(G_k \otimes A_k) - \lambda_k (H_k \otimes \mathbf{I})\right] \Phi}_{:=E} = 0, \quad G_k = A_k = 0, \text{ for } k > m. \tag{3.5}$$

Now, observe that the problem (3.3) can be considered as an *eigentuple-eigenvector* problem:

$$A\Phi = \sum_{k=0}^{N_\xi - 1} \lambda_k B_k \Phi, \tag{3.6}$$

in which one needs to find an eigentuple $\Lambda := (\lambda_0, \ldots, \lambda_{N_\xi - 1}) \in \mathbb{R}^{N_\xi}$ and an eigenvector $\Phi \in \mathbb{R}^{N_x N_\xi}$, where $A := \sum_{k=0}^m G_k \otimes A_k$ and $B_k := \lambda_k (H_k \otimes \mathbf{I})$. Note that $B_0 := H_0 = G_0 = \mathbf{I}$. Thus, the case $k = 0$ in (3.6) corresponds to the standard deterministic eigenproblem

$$A\Phi = \lambda_0 \Phi, \tag{3.7}$$

which has already been studied extensively [33]. For $k = 1$ (that is, $N_\xi = 2$), we obtain

$$(A - \lambda_1 B_1)\Phi = \lambda_0 B_0 \Phi, \tag{3.8}$$

which yields a standard eigenproblem for each fixed value of λ_1. Moreover, since A, B_0 and B_1 are symmetric matrices (with B_0 being positive definite), we have a continuum of real solutions $\lambda_0(\lambda_1)$ parameterized by λ_1. The existence of the continuum of real solutions is not surprising since there are $2N_x + 2 = 2(N_x + 1)$ unknowns (that is, λ_0, λ_1 and the components of Φ) in only $2N_x$ equations. To circumvent this situation, it is proposed in [17] to prescribe an additional condition via the normalization of the eigenvectors as in (2.1). This could then make it feasible to determine λ_1 and thereby reduce the two-parameter problem (3.8) to a one-parameter eigenproblem (3.7). Thus, the existence of a continuum of real solutions could make (3.8) numerically tractable by permitting its reduction to a sequence of solutions of (3.7), see e.g. [6] for details.

The two-parameter eigenproblem has been considered by Hochstenbach and his co-authors in [22, 23] following a Jacobi-Davidson approach. However, unlike the problem under consideration in this work, for which the resulting system is coupled, these authors focused on decoupled systems. Moreover, the approach that the authors adopted is quite complicated for two-parameter problems and can hardly be applied to multi-parameter eigenproblems considered in this paper. The approach considered here follows closely the framework of [17]. More specifically, our method relies on a Newton-Krylov solution technique, which we proceed to discuss in Section 4.

4. Newton-Krylov approaches.

4.1. The Newton system for stochastic eigenvalue problem. As we already pointed out in Section 3, the problem (3.6) contains more unknowns than equations. As suggested in [17], we incorporate the normalization condition of the eigenvectors so that the random eigenproblem is posed as a set of

$$N_x N_\xi + N_\xi = (N_x + 1)N_\xi \tag{4.1}$$

non-linear deterministic equations for each physical mode of the stochastic system. To this end, observe that SGM discretization of (2.2) yields [17]

$$\sum_{i=0}^{N_\xi-1}\sum_{j=0}^{N_\xi-1} \mathbb{E}(\psi_i\psi_j\psi_k)\varphi_i^T\varphi_j = \delta_{k0}, \quad k = 0, \ldots, N_\xi - 1, \tag{4.2}$$

or, equivalently,

$$\Phi^T(H_k \otimes \mathbf{I})\Phi = \delta_{k0}, \quad k = 0, 1, \ldots, N_\xi - 1. \tag{4.3}$$

The Newton's method is a well-established iterative method. For a well-chosen initial iterate, the method exhibits local quadratic convergence. In this method, (3.5) and (4.3) are simultaneously expressed in the form $F(x) = 0$, where $x = (\Lambda, \Phi) \in \mathbb{R}^{(N_x+1)N_\xi}$ is a vector containing the solution set defined in (2.7). More precisely, we have

$$F(x) = \begin{bmatrix} \sum_{k=0}^{N_\xi-1}\left[(G_k \otimes A_k) - \lambda_k(H_k \otimes \mathbf{I})\right]\Phi \\ \Phi^T(H_0 \otimes \mathbf{I})\Phi - 1 \\ \Phi^T(H_1 \otimes \mathbf{I})\Phi \\ \vdots \\ \Phi^T(H_{N_\xi-1} \otimes \mathbf{I})\Phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4.4}$$

The Newton iteration for $F(x) = 0$ results from a multivariate Taylor expansion about a current point x_k :

$$F(x_{k+1}) = F(x_k) + F'(x_k)(x_{k+1} - x_k) + \text{higher-oder terms.}$$

Setting the left-hand side to zero and neglecting the terms of higher-order curvature yields a Newton method; that is, given an initial iterate x_0, we obtain an iteration over a sequence of linear systems (or the Newton equations)

$$F(x_k) + F'(x_k)s_k = 0, \tag{4.5}$$

where x_k is the current iterate. Moreover, $F(x)$ is the vector-valued function of non-linear residuals and $\mathcal{J} := F'$ is the associated Jacobian matrix, x is the state vector to be found, and k is the iteration index. Forming each element of $\mathcal{J}$ requires taking analytic or discrete derivatives of the system of equations with respect to x_k. The solution $s_k := \delta x_k = x_{k+1} - x_k$ is the so-called Newton step. Once the Newton step is obtained, then the next iterate is given by $x_{k+1} = x_k + s_k$ and the procedure is repeated until convergence with respect to the prescribed tolerance is achieved. More specifically, given an initial approximation, say, $(v, \theta) := (v_0, v_1, \ldots, v_{N_\xi}, \theta_0, \theta_1, \ldots, \theta_{N_\xi}) \approx (\Phi, \Lambda)$, the next approximation (v^+, θ^+) in the Newton's method is given by

$$\begin{bmatrix} v^+ \\ \theta^+ \end{bmatrix} = \begin{bmatrix} v \\ \theta \end{bmatrix} - \underbrace{\begin{bmatrix} T(\theta) & T'(\theta)v \\ Q'(v) & 0 \end{bmatrix}^{-1}}_{\mathcal{J}:=F'} \underbrace{\begin{bmatrix} T(\theta)v \\ Q(v) \end{bmatrix}}_{F}, \tag{4.6}$$

where [17]

$$T(\theta) = \sum_{k=0}^{N_\xi - 1} \left[(G_k \otimes A_k) - \theta_k (H_k \otimes \mathbf{I}) \right] \in \mathbb{R}^{N_x N_\xi \times N_x N_\xi}, \tag{4.7}$$

$$T(\theta)v = \sum_{k=0}^{N_\xi - 1} \left[(G_k \otimes A_k) - \theta_k (H_k \otimes \mathbf{I}) \right] v \in \mathbb{R}^{N_x N_\xi}, \tag{4.8}$$

$$T'(\theta)v = - \sum_{k=0}^{N_\xi - 1} (H_k \otimes v_k) \in \mathbb{R}^{N_x N_\xi \times N_\xi}, \tag{4.9}$$

$$Q(v) = \mathbf{d} := \left[v^T (H_0 \otimes \mathbf{I}) v - 1, \cdots, v^T (H_{N_\xi - 1} \otimes \mathbf{I}) v \right]^T \in \mathbb{R}^{N_\xi}, \tag{4.10}$$

and

$$Q'(v) = 2 \sum_{k=0}^{N_\xi - 1} (H_k \otimes v_k^T) \in \mathbb{R}^{N_x N_\xi \times N_\xi}. \tag{4.11}$$

4.2. Inexact Newton method. Notwithstanding the locally quadratic convergence and simplicity of implementation of the Newton's method, it involves enormous computational cost, particularly when the size of the problem is large. In order to reduce the computational complexity associated with the method, Dembo, Eisenstat and Steihaug proposed in [11] the *inexact Newton method* as given by Algorithm 1, which is a generalization of the Newton's method.

The condition in line 5 of the algorithm is the inexact Newton condition. Note that the real number η_k in Algorithm 1 is the so-called forcing term for the k-th iteration step. At each iteration step of the inexact Newton method, η_k should be chosen first, and then an inexact Newton step s_k is obtained by solving the Newton equations (4.5) approximately with an efficient solver for systems of linear equations. Quite often, the linear system to be solved at each inexact Newton step is so large that it cannot be solved by direct methods. Instead, modern iterative solvers such as Krylov subspace methods [32] are typically used to solve the linear systems approximately. This leads to a special kind of inexact Newton method, commonly referred to as *inexact Newton-Krylov subspace method,* which is very popular in many application areas [1, 25, 36].

We point out here that it is nevertheless hard to choose a good sequence of forcing terms. More precisely, there may be a trade-off between the effort required to solve the linear system to a tight tolerance and the resulting required number of nonlinear iterations. Too large a value for η_k results in less work for the Krylov method but more nonlinear iterations, whereas too small a value for η_k results in more Krylov iterations per Newton iteration. Examples of this trade-off between total nonlinear iterations and execution time can be found in, for instance, [25] in the context of solution of Navier-Stokes equations. Several strategies for optimizing the computational work with a variable forcing term η_k are given in [1, 14]. At any rate, it is important to note that that choice of the forcing terms should be related to specific problems and the information of $F(x)$ should be used effectively [1].

Algorithm 1 Inexact Newton Method (INM)

```
Given $x_0 \in \mathbb{R}^{(N_x+1)N_\xi}$
for $k = 0, 1, \ldots$ (until $\{x_k\}$ convergence) do
    Choose some $\eta_k \in [0, 1)$.
    Solve the Newton equations (4.5) approximately to obtain a step $s_k$ such that
    $||F(x_k) + F'(x_k)s_k|| \leq \eta_k ||F(x_k)||$.
    Set $x_{k+1} = x_k + s_k$.
end for
```

For practical computations, there are some concrete strategies, one of which was proposed originally by Dembo and Steihaug in [12], namely,

$$\eta_k = \min\{1/(k+2), ||F(x_k)||\}. \tag{4.12}$$

Moreover, Cai et. al in [9] propose the following constant forcing terms:

$$\eta_k = 10^{-4}. \tag{4.13}$$

Two other popular adaptive strategies were proposed by Eisenstat and Walker in [14]:

(a) Given some $\eta_0 \in [0, 1)$, choose

$$\eta_k = \begin{cases} \zeta_k, & \eta_{k-1}^{(1+\sqrt{5})/2} \leq 0.1, \\ \max\left\{\zeta_k, \eta_{k-1}^{(1+\sqrt{5})/2}\right\}, & \eta_{k-1}^{(1+\sqrt{5})/2} > 0.1, \end{cases}$$

where

$$\zeta_k = \frac{||F(x_k) - F(x_{k-1}) - F'(x_{k-1})s_{k-1}||}{||F(x_{k-1})||}, \quad k = 1, 2 \ldots$$

or

$$\zeta_k = \frac{|\,||F(x_k)|| - ||F(x_{k-1}) + F'(x_{k-1})s_{k-1}||\,|}{||F(x_{k-1})||}, \quad k = 1, 2 \ldots$$

(b) Given some $\tau \in [0, 1),\ \omega \in [1, 2),\ \eta_0 \in [0, 1)$, choose

$$\eta_k = \begin{cases} \zeta_k, & \tau\eta_{k-1}^{\omega} \leq 0.1, \\ \max\left\{\zeta_k, \tau\eta_{k-1}^{\omega}\right\}, & \tau\eta_{k-1}^{\omega} > 0.1, \end{cases}$$

where

$$\zeta_k = \tau\left(\frac{||F(x_k)||}{||F(x_{k-1})||}\right)^{\omega}, \quad k = 1, 2 \ldots$$

The numerical experiments in [14] show that the above two choices (a) and (b) can effectively overcome the 'over-solving' phenomenon, and thus improve the efficiency of the inexact Newton method[2]. In particular, the authors added safeguards (bounds) to

[2] The concept of 'over-solving' implies that at early Newton iterations η_k is too small. Then one may obtain an accurate linear solution to an inaccurate Newton correction. This may result in a poor Newton update and degradation in the Newton convergence. In [40] it has been demonstrated that in some situations the Newton convergence may actually suffer if η_k is too small in early Newton iterations.

choice (a) and (b) to prevent the forcing terms from becoming too small too quickly, so that more concrete strategies are obtained. Besides, choice (a) and choice (b) with $\tau \geq 0.9$ and $\omega \geq (1+\sqrt{5})/2$ have the best performances. We adopt choice (b) in our numerical experiments.

The inexact Newton method is locally convergent as shown in the following result from [11].

Theorem 4.1. *[11, Theorem 2.3] Assume that $F : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable, $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$ and $F'(x^*)$ is nonsingular. Let $0 < \eta_k < \eta_{\max} < t < 1$ be given constants. If the forcing terms $\{\eta_k\}$ in the inexact Newton method satisfy $\eta_k \leq \eta_{\max} < t < 1$ for all k, then there exists $\varepsilon > 0$, such that for any $x_0 \in N_\varepsilon(x^*) := \{x : ||x - x^*|| < \varepsilon\}$, the sequence $\{x_k\}$ generated by the inexact Newton method converges to x^* , and*

$$||x_{k+1} - x_k||_* \leq t||x - x^*||_*,$$

where $||y||_* = ||F'(x^*)y||$.

By Theorem 4.1, if the forcing terms $\{\eta_k\}$ in the inexact Newton method are uniformly strict less than 1, then the method is locally convergent. The convergence rate of the inexact Newton method is, moreover, established in the following result from [11].

Theorem 4.2. *[11, Corollary 3.5] Assume that $F : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable, $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$ and $F'(x^*)$ is nonsingular. If the sequence $\{x_k\}$ generated by inexact Newton method converges to x^*, then*

- *$\{x_k\} \to x^*$ super-linearly when $\eta_k \to 0$.*
- *$\{x_k\} \to x^*$ quadratically when $\eta_k = \mathcal{O}(||F(x_k)||)$ and $||F'(x)||$ is Lipschitz continuous at x^*.*

For more details of local convergence theory and the role played by the forcing terms in inexact Newton methods, see e.g., [1, 14]. We proceed next to give an overview of Krylov subspace methods.

4.3. Krylov subspace methods.

Krylov subspace methods are probably the most popular methods for solving large, sparse linear systems (see e.g. [15] and the references therein). The basic idea behind Krylov subspace methods is the following. Consider, for arbitrary $A \in \mathbb{R}^{m\times m}$ and $b \in \mathbb{R}^m$, the linear system

$$Ax = b. \tag{4.14}$$

Suppose now that x_0 is an initial guess for the solution x of (4.14), and define the initial residual $r_0 = b - Ax_0$. Krylov subspace methods are iterative methods whose kth iterate x_k satisfies [3]

$$x_k \in x_0 + \mathbb{K}_k(A, x_0), \quad k = 1, 2, \ldots, \tag{4.15}$$

where

$$\mathbb{K}_k(A, x_0) := \text{span}\left\{r_0, Ar_0, \ldots, A^{k-1}r_0\right\} \tag{4.16}$$

denotes the kth Krylov subspace generated by A and r_0. The Krylov subspaces form a nested sequence that ends with dimension $d = \dim(\mathbb{K}_m(A, r_0)) \leq m$, i.e.,

$$\mathbb{K}_1(A, r_0) \subset \ldots \subset \mathbb{K}_d(A, r_0) = \cdots = \mathbb{K}_m(A, r_0).$$

[3] Krylov methods require only matrix-vector products to carry out the iteration (not the individual elements of A) and this is key to their use with the Newton's method, as will be seen below.

In particular, for each $k \leq d$, the Krylov subspace $\mathbb{K}_k(A, r_0)$ has dimension k. Because of the k degrees of freedom in the choice of the iterate x_k, k constraints are required to make x_k unique. In Krylov subspace methods this is achieved by requiring that the kth residual $r_k = b - Ax_k$ is orthogonal (with respect to the Euclidean inner product) to a k-dimensional space $\mathcal{C}_k$, called the constraints space:

$$r_k = b - Ax_k \in r_0 + A\mathbb{K}_k(A, r_0), \tag{4.17}$$

where $r_k \perp \mathcal{C}_k$. It can be shown [5] that there exists a uniquely defined iterate x_k of the form (4.15) and for which the residual $r_k = b - Ax_k$ satisfies (4.17) if

(a) A is symmetric positive definite and $\mathcal{C}_k = \mathbb{K}_k(A, r_0)$, or

(b) A is nonsingular and $\mathcal{C}_k = A\mathbb{K}_k(A, r_0)$.

In particular, (a) characterizes the conjugate gradient (CG) method [15] whereas (b) characterizes the minimal residual (MINRES) method [28], the generalized minimal residual (GMRES) method [34], and the bi-conjugate gradient stabilized (BiCGstab) method [41].

A vast majority of fully coupled nonlinear applications of primary interest (including the one considered herein) result in Jacobian matrices that are non-symmetric. A further point of discrimination is whether the method is derived from the long-recurrence Arnoldi orthogonalization procedure, which generates orthonormal bases of the Krylov subspace, or the short-recurrence Lanczos bi-orthogonalization procedure, which generates non-orthogonal bases for non-symmetric matrices A.

Note that GMRES is an Arnoldi-based method. In GMRES, the Arnoldi basis vectors form the trial subspace out of which the solution is constructed. One matrix-vector product is required per iteration to create each new trial vector, and the iterations are terminated based on a by-product estimate of the residual that does not require explicit construction of intermediate residual vectors or solutions – a major beneficial feature of the algorithm. GMRES has a residual minimization property in the Euclidean norm (easily adaptable to any inner-product norm) but requires the storage of all previous Arnoldi basis vectors. Full restarts, seeded restarts, and moving fixed sized windows of Arnoldi basis vectors are all options for fixed-storage versions. Full restart is simple and historically the most popular, though seeded restarts show promise. The BiCGstab methods [41] are Lanczos-based alternatives to GMRES for non-symmetric problems. In BiCGstab methods, the Lanczos basis vectors are normalized, and two matrix-vector products are required per iteration. However, these methods enjoy a short recurrence relation, so there is no requirement to store many Lanczos basis vectors. These methods do not guarantee monotonically decreasing residuals. We refer to [15] for more details on Krylov methods, and for preconditioning for linear problems.

4.4. Inexact Newton-Krylov method with backtracking.

In practice, *globalization strategies* leading from a convenient initial iterate into the ball of convergence of Newton's method around the desired root are often required to enhance the robustness of the inexact Newton method. More precisely, globalization implies augmenting Newton's method with certain auxiliary procedures that increase the likelihood of convergence when good initial approximate solutions are not available. Newton-Krylov methods, like all Newton-like methods, must usually be globalized. Globalizations are typically structured to test whether a step gives satisfactory progress towards a solution and, if necessary, to modify it to obtain a step that does give satisfactory

Algorithm 2 Inexact Newton Backtracking Method (INBM)

```
Let $x_0 \in \mathbb{R}^{(N_x+1)N_\xi}$, $\eta_{\max} \in [0,1)$, $t \in (0,1)$, and $0 < \theta_{\min} < \theta_{\max} < 1$, be given.
for $k = 0, 1, \dots$ (until $\{x_k\}$ convergence) do
    Choose initial $\eta_k \in [0, \eta_{\max})$ and solve (4.5) approximately to obtain $s_k$ such that
    $||F(x_k) + F'(x_k)s_k|| \leq \eta_k ||F(x_k)||$.
    While $||F(x_k + s_k)|| > [1 - t(1 - \eta_k)]||F(x_k)||$
    Choose $\theta \in [\theta_{\min}, \theta_{\max}]$.
    Update $s_k \leftarrow \theta s_k$ and $\eta_k \leftarrow 1 - \theta(1 - \eta_k)$.
    Set $x_{k+1} = x_k + s_k$.
end for
```

progress [29]. A major class of globalization approaches[4] which we consider in this paper are the *backtracking (line-search, damping) methods*. In these methods, the step lengths are adjusted (usually shortened) to obtain satisfactory steps. On the one hand, backtracking methods have the attrative feature of the relative ease with which they can be implemented; on the other hand, each step direction in these methods is restricted to be that of the initial trial step, which may be a weak descent direction, especially if the Jacobian is ill-conditioned [36].

The inexact Newton backtracking method (INBM) is given in Algorithm 2. In this algorithm, the backtracking globalization resides in the while-loop, in which steps are tested and shortened as necessary until the acceptability condition

$$||F(x_k + s_k)|| \leq [1 - t(1 - \eta_k)]||F(x_k)||, \tag{4.18}$$

holds. As noted in [13], if F is continuously differentiable, then this globalization produces a step for which (4.18) holds after a finite number of passes through the while-loop; furthermore, the inexact Newton condition (cf. line 5 in Algorithm 1) still holds for the final s_k and η_k. The condition (4.18) is a 'sufficient-decrease' condition on $||F(x_k + s_k)||$.

In [14], the authors show with experiments that backtracking globalization significantly improves the robustness of a Newton-GMRES method when applied to nonlinear problems, especially when combined with adaptively determined forcing terms. In this work, we combine the backtracking globalization with low-rank techniques to tackle the high-dimensional stochastic eigenproblem. Our motivation for employing low-rank techniques stems from the fact that despite the advantages of the INKM with backtracking in solving nonlinear problems, for the stochastic problem (2.1) – (2.2) under consideration, the dimensions of the Jacobian quickly become prohibitively large with respect to the discretization parameters. As a consequence, one expects overwhelming memory and computational time requirements, as the block-sizes of the Jacobian matrix become vast. This is a major drawback of the SGM. In this paper, we propose to tackle this *curse of dimensionality* with a low-rank version of INKM. Low-rank strategies have proven to be quite efficient in solving problems of really high computational complexity arising, for instance, from deterministic and stochastic time-dependent optimal control problems [2, 4, 38], PDEs with random coefficients [3, 27], etc. The low-rank technique presented here only needs to store a small portion of the vectors in comparison to the full problem and we want present this approach in the sequel.

[4] See e.g. [29, 36] for a detailed discussion on other globalization strategies such as trust-region methods.

4.5. Low-rank inexact Newton-Krylov method. As we have already noted earlier, we will use a Krylov solver algorithm as an optimal solver for the Newton equation (cf. (4.5) in step 3 in Algorithm 2) in each INKM iteration. In particular, our approach is based on the low-rank version of the chosen Krylov solver. Although the low-rank method discussed herein can be easily extended to other Krylov solvers [3, 4, 38], we focus mainly on BiCGstab [24]. In this section, we proceed first to give a brief overview of this low-rank iterative solver. Now, recall first that

$$\operatorname{vec}(WXV) = (V^T \otimes W)\operatorname{vec}(X), \tag{4.19}$$

where $\operatorname{vec}(X) = (x_1, \ldots, x_p)^T \in \mathbb{R}^{np\times 1}$ is a column vector obtained by stacking the columns of the matrix $X = [x_1, \ldots, x_p] \in \mathbb{R}^{n\times p}$ on top of each other. Observe then that, using (4.19), each Newton equation (4.5) can be rewritten as $\mathcal{J}\mathcal{X} = \mathcal{R}$, where

$$\mathcal{J} := F' = \begin{bmatrix} \sum\limits_{i=0}^{N_\xi-1} [(G_i - \lambda_i H_i) \otimes (A_i - I_{N_x})] & -\sum\limits_{i=0}^{N_\xi-1} H_i \otimes v_i \\ 2\sum\limits_{i=0}^{N_\xi-1} H_i \otimes v_i^T & 0 \end{bmatrix},$$

$$\mathcal{X} := s = \begin{bmatrix} \operatorname{vec}(Y) \\ \operatorname{vec}(Z) \end{bmatrix}, \quad \mathcal{R} := -F = \begin{bmatrix} \operatorname{vec}(R_1) \\ \operatorname{vec}(R_2) \end{bmatrix},$$

and

$$R_1 = \operatorname{vec}^{-1}\left(\sum_{i=0}^{N_\xi-1} [(G_i - \lambda_i H_i) \otimes (A_i - I_{N_x})]\, v \right), \quad R_2 = \operatorname{vec}^{-1}(\mathbf{d}),$$

where $\mathbf{d}$ is as given by (4.10). Hence, (4.19) implies that

$$\mathcal{J}\mathcal{X} = \operatorname{vec}\left(\sum_{i=0}^{N_\xi-1} \begin{bmatrix} (A_i - I_{N_x})Y(G_i - \lambda_i H_i)^T - v_i Z H_i^T \\ 2v_i^T Y H_i^T \end{bmatrix} \right) = \operatorname{vec}\left(\begin{bmatrix} R_1 \\ R_2 \end{bmatrix} \right). \tag{4.20}$$

Our approach is essentially based on the assumption that both the solution matrix $\mathcal{X}$ admits a low-rank representation; that is,

$$\begin{cases} Y = W_Y V_Y^T, & \text{with } W_Y \in \mathbb{R}^{(N_x+1)\times k_1}, \quad V_Y \in \mathbb{R}^{N_\xi \times k_1} \\ Z = W_Z V_Z^T, & \text{with } W_Z \in \mathbb{R}^{(N_x+1)\times k_2}, \quad V_Z \in \mathbb{R}^{N_\xi \times k_2} \end{cases} \tag{4.21}$$

where $k_{1,2,3}$ are small relative to N_ξ. Substituting (4.21) in (4.20) and ignoring the vec operator, we then obtain[5]

$$\sum_{i=0}^{N_\xi-1} \begin{bmatrix} (A_i - I_{N_x})W_Y V_Y^T (G_i - \lambda_i H_i)^T - v_i W_Z V_Z^T H_i^T \\ 2v_i^T W_Y V_Y^T H_i^T \end{bmatrix} = \begin{bmatrix} R_{11}R_{12}^T \\ R_{21}R_{22}^T \end{bmatrix}, \tag{4.22}$$

where $R_{11}R_{12}^T$ and $R_{21}R_{22}^T$ are the low-rank representations of R_1 and R_2, respectively.

[5] Note that v_i in (4.22) comes from the previous low-rank iterate of the nonlinear Newton solver.

Algorithm 3 Jacobian-vector multiplication in low-rank format `Amult`

```
Input: $W_{11}, W_{12}, W_{21}, W_{22}$
Output: $X_{11}, X_{12}, X_{21}, X_{22}$
$X_{11} = \sum_{i=0}^{N_\xi - 1} [\ (A_i - I)W_{11} \quad - v_i W_{21}\ ]$
$X_{12} = [\ (G_i - \lambda_i H_i)W_{12} \quad \cdots \quad H_i W_{22}\ ],\ \ i = 0, \cdots, N_\xi - 1.$
$X_{21} = \sum_{i=0}^{N_\xi - 1} [\ 2v_i^T W_{11}\ ]$
$X_{22} = [\ H_i W_{12}\ ],\ \ i = 0, \cdots, N_\xi - 1.$
```

The attractiveness of this approach lies therefore in the fact that one can rewrite the three block rows in the left hand side in (4.22), respectively, as

$$\left\{ \begin{array}{l} \text{(first block row)} \sum_{i=0}^{N_\xi - 1} [\ (A_i - I)W_Y \quad - v_i W_Z\ ] \begin{bmatrix} V_Y^T (G_i - \lambda_i H_i)^T \\ V_Z^T H_i^T \end{bmatrix}, \\ \\ \text{(second block row)} \sum_{i=0}^{N_\xi - 1} [\ 2v_i W_Y\ ] \begin{bmatrix} V_Y^T H_i^T \end{bmatrix}, \end{array} \right. \tag{4.23}$$

so that the low-rank nature of the factors guarantees fewer multiplications with the submatrices while maintaining smaller storage requirements. More precisely, keeping in mind that

$$x = \mathrm{vec}\left(\begin{bmatrix} X_{11} X_{12}^T \\ X_{21} X_{22}^T \end{bmatrix} \right) \tag{4.24}$$

corresponds to the associated vector x from a vector-based version of the Krylov solver, matrix-vector multiplication in our low-rank Krylov solver is given by Algorithm 3.

Note that an important feature of low-rank Krylov solvers is that the iterates of the solution matrices Y and Z in the algorithm are truncated by a truncation operator $\mathcal{T}_\epsilon$ with a prescribed tolerance ϵ. This could be accomplished via QR decomposition as in [24] or truncated singular value decomposition (SVD) as in [3, 38]. The truncation operation is necessary because the new computed factors could have increased ranks compared to the original factors in (4.23). Hence, a truncation of all the factors after the matrix-vector products, is used to construct new factors; for instance,

$$[\tilde{X}_{11}, \tilde{X}_{12}] := \mathcal{T}_\epsilon\left([X_{11}, X_{12}]\right) = \mathcal{T}_\epsilon\left(\sum_{i=0}^{N_\xi - 1} [\ (A_i - I)W_Y \quad - v_i W_Z\ ] \begin{bmatrix} V_Y^T (G_i - \lambda_i H_i)^T \\ V_Z^T H_i^T \end{bmatrix} \right).$$

Moreover, in order to ensure that the inner products within the iterative low-rank solver are computed efficiently, we use the fact that

$$\langle x, y \rangle = \mathrm{vec}\,(X)^T\, \mathrm{vec}\,(Y) = \mathrm{trace}\left(X^T Y\right)$$

to deduce that

$$\begin{aligned} \mathrm{trace}\left(X^T Y\right) &= \mathrm{trace}\left(\underbrace{\left(X_{11} X_{12}^T\right)^T}_{\text{Large}} \underbrace{\left(Y_{11} Y_{12}^T\right)}_{\text{Large}} + \underbrace{\left(X_{21} X_{22}^T\right)^T}_{\text{Large}} \underbrace{\left(Y_{21} Y_{22}^T\right)}_{\text{Large}} \right) \\ &= \mathrm{trace}\left(\underbrace{Y_{12}^T X_{12}}_{\text{Small}} \underbrace{X_{11}^T Y_{11}}_{\text{Small}} + \underbrace{Y_{22}^T X_{22}}_{\text{Small}} \underbrace{X_{21}^T Y_{11}}_{\text{Small}} \right), \end{aligned} \tag{4.25}$$

Algorithm 4 Preconditioner implementation in low-rank Krylov solver

```
Input: $W_{11}, W_{12}, W_{21}, W_{22}$
Output: $X_{11}, X_{12}, X_{21}, X_{22}$
Solve: $(A_0 - I_{N_x})X_{11} = W_{11}$
Solve: $(1 - \lambda_0)X_{12} = W_{12}$
Solve: $\left[v_0(A_0 - I_{N_x})^{-1}v_0^T\right] X_{21} = W_{21}$
Solve: $2(1 - \lambda_0)^{-1}X_{22} = W_{12}$
```

where X and Y are as given in (4.24), which allows us to compute the trace of small matrices rather than of the ones from the full model.

For more details on implementation issues, we refer the interested reader to [3, 38].

4.6. Preconditioning. The purpose of preconditioning the INBM is to reduce the number of Krylov iterations, as manifested by efficiently clustering eigenvalues of the iteration matrix. Traditionally, for linear problems, one chooses a few iterations of a simple iterative method (applied to the system matrix) as a preconditioner. Throughout this paper, we will focus mainly on mean-based block-diagonal preconditioners. More specifically, we precondition the Jacobian matrix J (cf. (4.6)) in the INBM algorithm with a preconditioner $\mathcal{P}$ of the form

$$\mathcal{P} := \begin{bmatrix} E & 0 \\ 0 & S \end{bmatrix}, \tag{4.26}$$

where

$$S = CE^{-1}B \tag{4.27}$$

is the (negative) *Schur complement*. Moreover, $E := T(\Lambda)$, $B := T'(\Lambda)$ and $C := Q'(\Phi)$ as given, respectively, by (4.7), (4.9) and (4.11). We note here that (4.26) is only an ideal preconditioner for the Jacobian in the sense that it is not cheap to solve the system with it. In practice, one often has to approximate its two diagonal blocks in order to use $\mathcal{P}$ with Krylov solvers. Here, we propose to approximate the $(1, 1)$ blocks with $(G_0 - \lambda_0 H_0) \otimes (A_0 - I_{N_x})$ which is easy to invert: if we use the normalized Legendre polynomial chaos to compute the matrices G_i and H_i, then $(G_0 - \lambda_0 H_0) = (1 - \lambda_0)I_{N_\xi}$ so that action of the approximated $(1, 1)$ block is just N_ξ copies of $(A_0 - I_{N_x})$. To approximate the Schur complement S, that is, block $(2, 2)$, poses more difficulty, however. One possibility is to approximate S by dropping all but the first terms in B, C and E to obtain

$$\begin{aligned} S_0 &:= 2(1 - \lambda_0)^{-1}(I_{N_\xi} \otimes v_0)(I_{N_\xi} \otimes (A_0 - I_{N_x})^{-1})(I_{N_\xi} \otimes v_0)^T \\ &= 2(1 - \lambda_0)^{-1} I_{N_\xi} \otimes \left[v_0(A_0 - I_{N_x})^{-1}v_0^T\right]. \end{aligned} \tag{4.28}$$

This is the version we use in our experiments, and its implementation details are provided in Algorithm 4.

5. Numerical results. In this section, we present some numerical results obtained with the proposed inexact Newton-Krylov solver for the stochastic eigenproblems (2.1). The numerical experiments were performed on a Linux machine with 80 GB RAM using MATLAB® 7.14 together with a MATLAB version of the algebraic multigrid (AMG) code HSL MI20 [7]. We implement our mean-based preconditioner using one V-cycle of AMG with symmetric Gauss-Seidel (SGS) smoothing to approximately invert $A_0 - I_{N_x}$. We remark here that we apply the method as a black-box in each experiment and the set-up of the approximation to $A_0 - I_{N_x}$ only needs to be performed once. Unless otherwise stated, in all the simulations, BiCGstab is terminated when the relative residual error is reduced to $tol = 10^{-5}$. Note that tol should be chosen such that the truncation tolerance $trunctol \leq tol$; otherwise, one would be essentially iterating on the 'noise' from the low-rank truncations. In particular, we have chosen herein $trunctol = 10^{-6}$. We have used the Frobenius norm throughout our numerical experiments.

Before proceeding to present our numerical example, it is perhaps pertinent to highlight certain factors that often influence the convergence of the inexact Newton method [16]:

- the proximity of the initial guess. Here, we have employed uniformly distributed samples for our initial guess.
- The globalization technique employed, (e.g. backtracking, or trust region). In this paper, we have used only backtracking and it worked quite well for our considered problem.
- The discretization of the SEVPs – failure of the spatial discretization to adequately reflect the underlying physics of the continuous problem can cause convergence difficulties for globalized Newton-Krylov methods.
- The convergence of the Krylov solver and preconditioning strategy employed – using nonlinear preconditioning techniques can be an alternative [8].

For our numerical example, let $\mathcal{D} = (0,1) \times (0,1)$. We consider the stochastic eigenproblem of finding the functions $\lambda : \Omega \to \mathbb{R}$ and $\varphi : \Omega \times D \to \mathbb{R}$ such that, $\mathbb{P}$-almost surely, the following holds:

$$\begin{cases} -\nabla \cdot (a(\cdot,\omega)\nabla\varphi(\cdot,\omega)) = \lambda(\omega)\varphi(\cdot,\omega), \text{ in } \mathcal{D}\times\Omega, \\ \varphi(\cdot,\omega) = 0, \text{ on } \partial\mathcal{D}\times\Omega, \end{cases} \tag{5.1}$$

where $a : \mathcal{D}\times\Omega \to \mathbb{R}$ is a random coefficient field. We assume that there exist positive constants $a_{\min}$ and $a_{\max}$ such that

$$\mathbb{P}\left(\omega \in \Omega : a(\mathbf{x},\omega) \in [a_{\min}, a_{\max}], \forall \mathbf{x} \in \mathcal{D}\right) = 1. \tag{5.2}$$

Here, the random input $a(\cdot,\omega)$ admits a KLE and has a covariance function given by

$$C_a(\mathbf{x},\mathbf{y}) = \sigma_a^2 \exp\left(-\frac{|x_1 - y_1|}{\ell_1} - \frac{|x_2 - y_2|}{\ell_2}\right), \quad \forall(\mathbf{x},\mathbf{y}) \in [-1,1]^2,$$

with correlation lengths $\ell_1 = \ell_2 = 1$ and mean of the random field a in the model $\mathbb{E}[a] = 1$. The forward problem has been extensively studied in, for instance, [30]. The eigenpairs of the KLE of the random field a are given explicitly in [18]. Note then that discretising in space yields the expression (2.1) with the random matrix $\mathcal{A}(\omega)$ having the explicit expression (2.3). In particular, the stiffness matrices $A_k \in \mathbb{R}^{N_x \times N_x}$, $k = 0, 1, \ldots, m$, in (2.3) are given, respectively, by

$$A_0(i,j) = \int_{\mathcal{D}} \mathbb{E}[a](x)\nabla\phi_i(x)\nabla\phi_j(x)\,dx, \tag{5.3}$$

$$A_k(i,j) = \sigma_a\sqrt{\gamma_k}\int_{\mathcal{D}} \vartheta_k(x)\nabla\phi_i(x)\nabla\phi_j(x)\,dx, \;\; k > 0, \tag{5.4}$$

where σ_a is the standard deviation of a. Here, $\{\gamma_k\}$ and $\{\vartheta_k(x)\}$ are, respectively, the eigenvalues and eigenfunctions corresponding to a covariance function associated with a. Also, $\{\phi_j(x)\}$ are $\mathbf{Q}_1$ spectral elements which we have used to discretize the problem (5.1) in the spatial domain $\mathcal{D}$. Moreover, we choose $\xi = \{\xi_1, \ldots, \xi_m\}$ such that $\xi_k \sim \mathcal{U}[-1, 1]$, and $\{\psi_k\}$ are m-dimensional Legendre polynomials with support in $[-1, 1]^m$. In particular, we have $N_\xi = 210$ (with $m = 6$ and $r = 4$; cf. (2.6)).

In what follows, we consider two cases. First, in Table 5.2, we set $\sigma_a = 0.01$ and $N_x = 49$, so that from (4.1) and (4.6), we have a Jacobian matrix $\mathcal{J}$ of dimension $\dim(\mathcal{J}) := (N_x + 1)N_\xi = 10,500$. Admittedly, one would argue that this dimension of the Jacobian is small, and as such can as well be handled without the low-rank method proposed in this paper! Such an arguement is understandably valid. However, this is basically intended to provide a first and simple insight as to how the algorithm works. A more difficult case is provided in Table 5.3 where we have increased σ_a and N_x to $\sigma_a = 0.1$ and $N_x = 392,704$, respectively. Hence, we obtain a Jacobian of size $\dim(\mathcal{J}) := (N_x + 1)N_\xi = 82,468,050$ at each inexact Newton iteration! We note here that, besides the increased dimension, increasing the variability (σ_a) in the problem equally increases the complexity of the linear system to be solved [3].

The methods discussed in the previous sections have many parameters that must be set, e.g., the maximum BiCGstab iterations, maximum forcing term, etc. These parameters affect the performance of the methods. We chose parameters commonly used in the literature. In particular, for the forcing terms η_k, we set $\eta_0 = 0.9, \eta_{\max} = 0.9$, $\eta_{\min} = 0.1$. For the backtracking parameters, we used $\theta_{\max} = 0.1$, $\theta_{\max} = 0.5$; the maximum number of backtracks allowed is 20.

Now, we consider the first case; that is, when $\dim(\mathcal{J}) := (N_x+1)N_\xi = 10,500$. We note here that the INBM algorithm presented in this paper computes one eigenvalue nearest to the initial guess. To compute two or more distinct (or multiple) roots of $F(x) = 0$ for an SEVP would require some specialized techniques, which can be a lot more involved. Nevertheless, this task is currently under investigation, and an algorithm for the computation of other eigenvalues will be presented in a subsequent paper.

All the eigenvalues of the deterministic matrix (i.e. A_0) are shown in Figure 5.1. The first six smallest eigenvalues of A_0 are $0.5935, 1.4166, 1.4166, 2.1143, 2.6484, 2.6484$. We note here that most of the eigenvalues of the matrix A_0 are either repeated or quite clustered. Observe in particular that 1.4166 and 2.6484 are repeated eigenvalues.

In Figure 5.2, we show the convergence of the low-rank INBM to the second stochastic eigenvalue $\lambda_2(\omega)$. The figure confirms the super-linear convergence of the inexact Newton algorithm as we reported earlier. In Table 5.1 and Figure 5.3, we have shown the first eight coefficients of the spectral expansion gPCE and the probability density function (pdf) of the second eigenvalue, respectively. Observe here that the pdf is as expected centered at the mean of the stochastic eigenvalue, i.e 1.4121. This quantity can also be obtained from the first coefficient of the gPCE in Table 5.1, since from (2.4), we have

$$\mathbb{E}(\lambda_2(\omega)) = \sum_{k=0}^{N_\xi - 1} \lambda_k^{(2)} \mathbb{E}(\psi_k(\xi(\omega))) = \lambda_0^{(2)},$$

due to the orthogonality of the basis polynomials $\{\psi_k\}$. We remark here also that this mean value of the second eigenvalue is quite close to the eigenvalue computed from the associated deterministic problem, i.e., 1.4166. If we increased the order of the Legendre polynomials, then the INBM would converge to the exact deterministic

Fig. 5.1: Eigenvalues of the deterministic matrix A_0

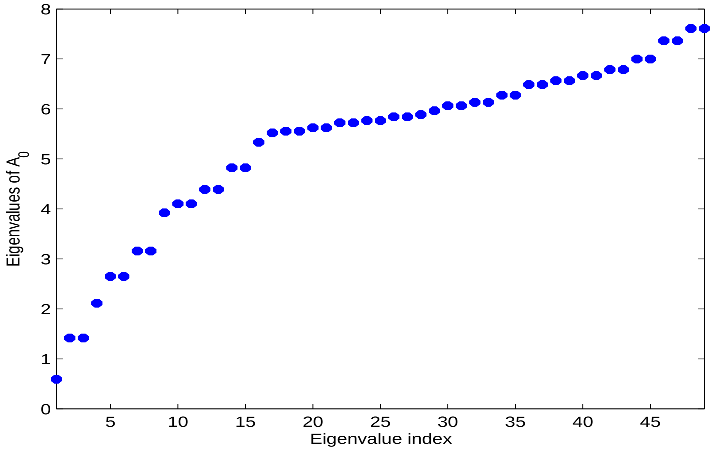
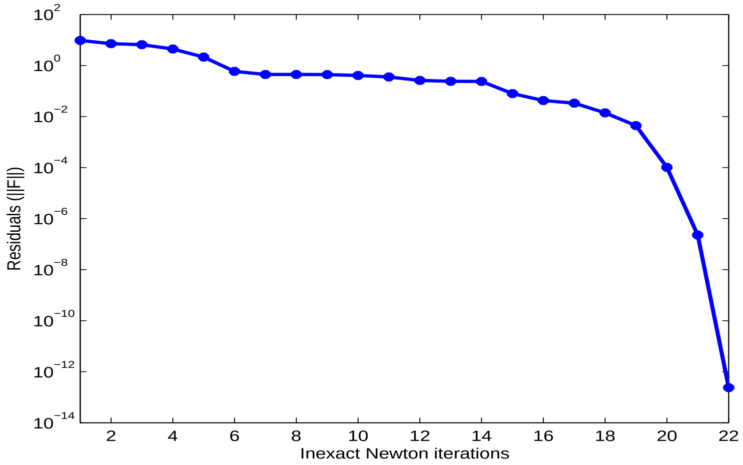


Fig. 5.2: Convergence of low-rank INBM for the second stochastic eigenvalue $\lambda_2(\omega)$.

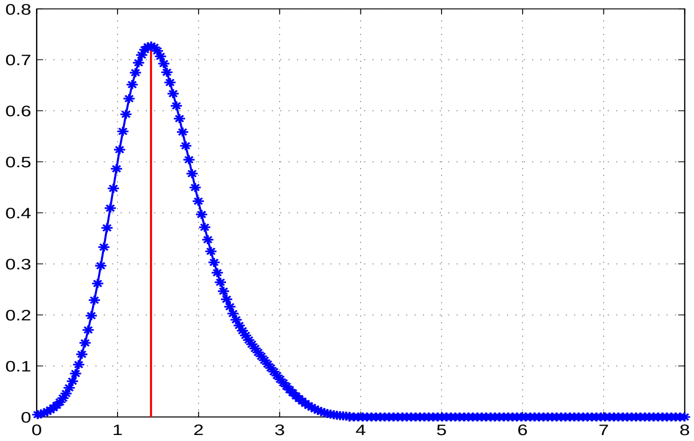


value. However, this would come at a higher computational expense, as the quantity N_ξ would also need to be increased accordingly. This kind of observation has earlier been extensively verified by the authors in the context of linear systems arising from PDEs with uncertain inputs [3, 27].

Table 5.1: The first eight coefficients of the spectral expansion gPCE of the second eigenvalue with using INBM. Here, k stands for the index of the basis function in the expansion (2.4).

k	0	1	2	3	4	5	6	7
$\lambda_k^{(2)}$	1.4121	0.5492	0.7009	0.5492	-0.02013	0.0952	-0.03537	0.0594

Fig. 5.3: Probability density function (pdf) estimate of the second eigenvalue obtained with $\sigma_a = 0.1$



Next, in Tables 5.2 and 5.3, we show the performance of the INBM solver in the two cases; that is, for $\dim(\mathcal{J}) = 10,500$ and $\dim(\mathcal{J}) = 82,468,050$. Here, the Newton equations (cf. (4.5)) are solved using low-rank BiCGstab, as well as using the standard preconditioned BiCGstab method which we have denoted as full model (FM), that is, without low-rank truncation. The CPU times reported are the total time it took the solver to compute the spectral coefficients of the eigenpair $(\lambda_2(\omega), \varphi_2(\omega))$. Here, for each choice of the forcing terms $\{\eta_k\}$ discussed in Section 4.2, we report inexact Newton steps (INS), backtracks per inexact Newton step (BINS), total BiCGstab iterations (iter), total CPU times (t) in seconds, ranks of the solution (R), memory in kilobytes of the low-rank solution (LR) and full method solution (FM). By the memory requirement of a low-rank solution $X = WV^T$, we mean the sum of the two separate computer memories occupied by its factors W and V^T, since X is computed and stored in this format, unlike the solution from FM. From the two tables, we see that for this problem, the performance of the INBM algorithm is independent of the the choice of the forcing terms $\{\eta_k\}$. In particular, Table 5.2 shows that the algorithm could compute the solution within a few seconds in the first case. Besides, the INBM algorithm reduces the storage requirement of the solution to one-quarter of memory required to compute the full solution. In fact, as shown in [3, 4], for a fixed N_ξ, low-rank Krylov solvers typically provide more storage benefits as $N_x \to \infty$.

Finally, as in the first case, we have also computed only the second stochastic eigenvalue $\lambda_2(\omega)$ of the matrix $\mathcal{A}(\omega)$ (cf. (2.3)) for the case where $\dim(\mathcal{J}) = 82,468,050$. Again, the mean $\lambda_0^{(2)}$ of this stochastic eigenvalue corresponds to the second eigenvalue of the deteministic matrix A_0, which in this case is 0.003. Note in particular from Table 5.3 that with the FM, MATLAB indeed fails as the size of the Jacobian matrix $\mathcal{J}$ at each inexact Newton step is now increased to more than 82 million degrees of freedom. Yet, INBM handles this task in about 200 minutes; that is, roughly 6 minutes per Newton step. Here, the solution from FM terminates with ‘out of memory’, which we have denoted as ‘OoM’.

Table 5.2: Performance of the INBM solver for $\dim(\mathcal{J}) = 10,500$ with $\sigma_a = 0.01$. Here, I, II, and III represent the different forcing parameter choices (4.12), (4.13), and (b) in Section 4.2.

η_k	INS	BINS	# iter	t	R	mem(LR)	mem(FM)
I	22	1.5	22	16.5	9	18.7	84
II	22	1.5	22	17.5	10	20.8	84
III	22	1.5	23	17.2	10	20.8	84

Table 5.3: Performance of the INBM solver for $\dim(\mathcal{J}) = 82,468,050$ with $\sigma_a = 0.1$. Here, I, II, and III represent the different forcing parameter choices (4.12), (4.13), and (b) in Section 4.2.

η_k	INS	BINS	# iter	t	R	mem(LR)	mem(FM)
I	34	3.6	39	12123.4	51	156551.7	OoM
II	32	3.5	43	12112.8	51	156551.7	OoM
III	33	3.5	42	12200.1	51	156551.7	OoM

6. Conclusions. In computational science and engineering, there are certain problems of growing interest for which random matrices are considered as random perturbations of finite-dimensional operators. These random matrices are usually not obtained from a finite-dimensional representation of a partial differential operator, and in a number of interesting cases, closed-form expressions of the statistical moments and probability density functions of their eigenvalues and eigenvectors are available; see e.g., [37]. The matrices of interest in the present paper, on the other hand, are the result of a finite-dimensional approximation of an underlying continuous system and their stochasticity is intrinsically tied to the uncertainty in the parameters of this system. For such systems, closed-form expressions are generally not available for the solution of the SEVPs.

In this paper, we have presented a low-rank Newton-type algorithm for approximating the eigenpairs of SEVPs. The numerical experiments confirm that the proposed solver can mitigate the computational complexity associated with solving high dimensional Newton systems in the considered SEVPs. More specifically, the low-rank approach guarantees significant storage savings [3, 4, 38], thereby enabling the solution of large-scale SEVPs that would otherwise be intractable.

Acknowledgement. The authors would like to thank Sergey Dolgov for fruitful discussions in the course of writing this paper. The work was performed while Martin Stoll was at the Max Planck Institute for Dynamics of Complex Technical Systems.

REFERENCES

[1] H.-B. Ana, Z.-Y Mob, and X.-P. Liua, *A choice of forcing terms in inexact Newton method*, Journal of Computational and Applied Mathematics, 200 (2007), pp. 47 – 60.

[2] P. Benner, S. Dolgov, A. Onwunta, and M. Stoll, *Low-rank solvers for unsteady Stokes-Brinkman optimal control problem with random data*, Computer Methods in Applied Mechanics and Engineering, 304 (2016), pp. 26–54.

[3] P. Benner, A. Onwunta, and M. Stoll, *Low-rank solution of unsteady diffusion equations with stochastic coefficients*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 622 – 649.

[4] ———, *Block-diagonal preconditioning for optimal control problems constrained by PDEs with uncertain inputs*, SIAM Journal on Matrix Analysis and Applications, 37 (2016), pp. 491 – 518.

[5] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1 – 137.

[6] E. K. Blum and A. R. Curtis, *A convergent gradient method for matrix eigenvector-eigentuple problems*, Numerische Mathematik, 31 (1978), pp. 247 – 263.

[7] J. Boyle, M. D. Mihajlovic, and J. A. Scott, *HSL MI20: An efficient AMG preconditioner for finite element problems in 3D*, International Journal for Numerical Methods in Engineering, 82 (2010), pp. 64–98.

[8] P. Brune, M. G. Knepley, B. Smith, and X. Tu, *Composing scalable nonlinear algebraic solvers*, Tech. Report ANL/MCS-P2010-0112, Argonne National Laboratory, Argonne, IL, 2013.

[9] X. C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriti, *Newton-Krylov-Schwarz methods in CFD*, Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations, (1995), pp. 17 – 30.

[10] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, *Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients*, Computing and Visualization in Science, 14 (2011), pp. 3–15.

[11] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400 – 408.

[12] R. S. Dembo and T. Steihaug, *Truncated Newton algorithms for large-scale optimization*, Mathematical Programming, 26 (1983), pp. 190 – 212.

[13] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM Journal on Optimization, 4 (1994), pp. 393 – 422.

[14] ———, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17(1) (1996), pp. 16 – 32.

[15] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*, vol. Second Edition, Oxford University Press, 2014.

[16] P. E. Farrell, A. Birkisson, and S. W. Funke, *Deflation techniques for finding distinct solutions of nonlinear partial differential equations*, SIAM Journal on Scientific Computing, 37 (2015), pp. A2026 – A2045.

[17] R. Ghanem and D. Ghosh, *Efficient characterization of the random eigenvalue problem in a polynomial chaos decomposition*, International Journal for Numerical Methods in Engineering, 72 (2007), pp. 486 – 504.

[18] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag: New York, 1991.

[19] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman & Hall, London, 1995.

[20] G. H. Golub and C. F. van Loan, *Matrix Computations*, vol. Third Edition, Johns Hopkins University Press, 1996.

[21] H. Hakula, V. Kaarnioja, and M. Laaksonen, *Approximate methods for stochastic eigenvalue problems*, Applied Mathematics and Computation, 267 (2015), pp. 664 – 681.

[22] M. Hochstenbach, T. Kosir, and B. Plestenjak, *A Jacobi-Davidson type method for the two-parameter eigenvalue problem*, SIAM Journal on Matrix Analysis and Applications, 24 (2005), pp. 477 – 497.

[23] M. Hochstenbach and B. Plestenjak, *A Jacobi-Davidson type method for a right definite two-parameter eigenvalue problem*, SIAM Journal on Matrix Analysis and Applications, 24 (2002), pp. 392 – 410.

[24] D. Kressner and C. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1288–1316.

[25] P. R. McHugh and D. A. Knoll, *Fully implicit finite volume solutions of the incompressible Navier-Stokes and energy equations using inexact Newton's method*, International Journal for Numerical Methods in Fluids, 18 (1994), pp. 439 – 455.

[26] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.

[27] A. Onwunta, *Low-Rank Iterative Solvers for Stochastic Galerkin Linear Systems*, PhD thesis, Otto-von-Guericke Universität, Magdeburg, 2016.

[28] C. C. Paige and M. A. Saunders, *Solutions of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629.

[29] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker, *Globalization techniques*

for Newton-Krylov methods and applications to the fully coupled solution of the Navier-Stokes equations, SIAM Review, 48 (2006), pp. 700 – 721.

[30] C. E. Powell and H. Elman, *Block-diagonal preconditioning for spectral stochastic finite-element systems*, IMA Journal of Numerical Analysis, 29 (2009), pp. 350–375.

[31] H. J. Pradlwarter, G. I. Schuaeller, G. S, and Szekely, *Random eigenvalue problems for large systems*, Computers and Structures, 80 (20 - 30) (2002), pp. 2415 – 2424.

[32] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[33] ———, *Numerical Methods for Large Eigenvalue Problems: Revised Edition*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2011.

[34] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[35] G. I. Schuaeller, G. S, and Szekely, *Computational procedure for a fast calculation of eigenvectors and eigenvalues of structures with random properties*, Computer Methods in Applied Mechanics and Engineering, 191 (8 - 10) (2001), pp. 799 – 816.

[36] J. N. Shadid, R. S. Tuminaro, and H. F. Walker, *An inexact Newton method for fully coupled solution of the Navier-Stokes equations with heat and mass transport*, Journal of Computational Physics, 137 (1997), pp. 155 – 185.

[37] C. Soize, *Random matrix theory for modeling uncertainties in computational mechanics*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 1333 – 1366.

[38] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM Journal on Scientific Computing, 37 (2015), pp. B1 – B29.

[39] H. Tiesler, R. M. Kirby, D. Xiu, and T. Preusser, *Stochastic collocation for optimal control problems with stochastic PDE constraints*, SIAM Journal on Control and Optimization, 50 (2012), pp. 2659 – 2682.

[40] R. S. Tuminaro, H. F. Walker, and J.N. Shadid, *On backtracking failure in Newton-GMRES methods with a demonstration for the Navier-Stokes equations*, Journal of Computational Physics, 180 (2002), pp. 549 – 558.

[41] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 13 (1992), pp. 631 – 644.

[42] C. V. Verhoosel, M. A. Gutierrez, and S.J. Hulshoff, *Iterative solution of the random eigenvalue problem with application to spectral stochastic finite element systems*, International Journal for Numerical Methods in Engineering, 68 (2006), pp. 401 – 424.

[43] D. Xiu and J. Shen, *Efficient stochastic Galerkin methods for random diffusion*, Journal of Computational Physics, 228 (2009), pp. 266–281.