BIT Numerical Mathematics (2022) 62:965–994
https://doi.org/10.1007/s10543-021-00903-x

BIT

Enhanced alternating energy minimization methods for stochastic galerkin matrix equations

Kookjin Lee[1] · **Howard C. Elman**[2] · **Catherine E. Powell**[3] · **Dongeun Lee**[4]

Received: 13 March 2021 / Accepted: 18 November 2021 / Published online: 3 January 2022

Abstract

In uncertainty quantification, it is commonly required to solve a forward model consisting of a partial differential equation (PDE) with a spatially varying uncertain coefficient that is represented as an affine function of a set of random variables, or parameters. Discretizing such models using stochastic Galerkin finite element methods (SGFEMs) leads to very high-dimensional discrete problems that can be cast as linear multiterm matrix equations (LMTMEs). We develop efficient computational methods for approximating solutions of such matrix equations in low rank. To do this, we follow an alternating energy minimization (AEM) framework, wherein the solution is represented as a product of two matrices, and approximations to each component are sought by solving certain minimization problems repeatedly. Inspired by proper generalized decomposition methods, the iterative solution algorithms we present are based on a rank-adaptive variant of AEM methods that successively computes a rank-one solution component at each step. We introduce and evaluate new enhancement procedures to improve the accuracy of the approximations these algorithms deliver. The efficiency and accuracy of the enhanced AEM methods is demonstrated through numerical exper-

Communicated by Lothar Reichel.

✉ Kookjin Lee
kookjin.lee@asu.edu

Howard C. Elman
elman@cs.umd.edu

Catherine E. Powell
c.powell@manchester.ac.uk

Dongeun Lee
dongeun.lee@tamuc.edu

1 School of Computing and Augmented Intelligence, Arizona State University, Tempe, USA

2 Department of Computer Science, University of Maryland, College Park, USA

3 Department of Mathematics, University of Manchester, Manchester, UK

4 Department of Computer Science, Texas A&M University-Commerce, Commerce, USA

iments with LMTMEs associated with SGFEM discretizations of parameterized linear elliptic PDEs.

Keywords Low-rank approximation · Alternating energy minimization · Stochastic Galerkin methods · Matrix equations · PDEs with random inputs · Uncertainty quantification

Mathematics Subject Classification 15A24 · 65C30 · 65F10 · 65M60

1 Introduction

We are interested in computing low-rank approximate solutions of linear systems $Au = b$ with the Kronecker-product structure

$$\left(\sum_{i=0}^{m} G_i \otimes K_i\right) u = \sum_{i=0}^{\hat{m}} g_i \otimes f_i, \tag{1.1}$$

where $A = \sum_{i=0}^{m} G_i \otimes K_i$ is symmetric positive definite, $\otimes$ is the Kronecker product, $\{K_i\}_{i=0}^{m} \in \mathbb{R}^{n_1 \times n_1}$, $\{G_i\}_{i=0}^{m} \in \mathbb{R}^{n_2 \times n_2}$, $\{f_i\}_{i=0}^{\hat{m}} \in \mathbb{R}^{n_1}$, and $\{g_i\}_{i=0}^{\hat{m}} \in \mathbb{R}^{n_2}$, and we assume that $m, \hat{m} \ll n_1, n_2$. The solution vector $u \in \mathbb{R}^{n_1 n_2}$ consists of n_2 subvectors of dimension n_1, i.e., $u = [u_1^{\mathsf{T}}, \ldots, u_{n_2}^{\mathsf{T}}]^{\mathsf{T}}$, where $\{u_i\}_{i=1}^{n_2} \in \mathbb{R}^{n_1}$. The solution also has an alternative representation in matrix format, $U = [u_1, \ldots, u_{n_2}] \in \mathbb{R}^{n_1 \times n_2}$. Exploiting this, and using standard properties of the Kronecker product, one can show that the linear system (1.1) is equivalent to a linear multi-term matrix equation (LMTME) [36]

$$\sum_{i=0}^{m} K_i U G_i^{\mathsf{T}} = B, \quad \text{where} \quad B = \sum_{i=0}^{\hat{m}} f_i g_i^{\mathsf{T}} \in \mathbb{R}^{n_1 \times n_2}. \tag{1.2}$$

Systems with such structure arise, for example, in the discretization of deterministic linear elliptic PDEs on high-dimensional domains [2,20–22] as well as in the discretization, via stochastic Galerkin finite element methods (SGFEMs) [13,18,25,28,47], of linear elliptic PDEs parameterized with random or unknown coefficients (see Eqs. (1.4)–(1.5) below). When the matrices K_i and G_i are sparse, then for moderately large values of n_1 and n_2 it is feasible to solve (1.1) using standard iterative methods. Indeed, in the case of parameterized PDEs, standard *Krylov subspace methods* [34,35] and *multigrid methods* [4,10,24] have been considered. However, the dimensions of the system matrices can grow rapidly when the discretization is refined and, in the case of parameterized PDEs, when the number m of input random variables is increased.

For large n_1 and n_2, direct application of standard iterative methods may be computationally prohibitive and storing or explicitly forming the matrix U may be prohibitive in terms of memory. Motivated by this, we are interested in inexpensive computation of approximate solutions of LMTMEs of the form (1.2) of low rank, using methods

that do not require constructions of matrices of size $n_1 \times n_2$. We begin by introducing a factored representation of $U \in \mathbb{R}^{n_1 \times n_2}$,

$$U = V W^{\mathsf{T}},$$

where, if U is of full rank $r := \min(n_1, n_2)$, $V \in \mathbb{R}^{n_1 \times r}$ and $W \in \mathbb{R}^{n_2 \times r}$. Our aim is then to find a low-rank approximation to this factored matrix of the form

$$U_p = V_p W_p^{\mathsf{T}} \in \mathbb{R}^{n_1 \times n_2}, \tag{1.3}$$

where $V_p = [v_1, \ldots, v_p] \in \mathbb{R}^{n_1 \times p}$ and $W_p = [w_1, \ldots, w_p] \in \mathbb{R}^{n_2 \times p}$ and $p \ll r$, and we want to derive solution algorithms for computing U_p that operate only on the smaller factors V_p and W_p without explicitly forming the large matrix U_p.

One such solution algorithm has been developed for matrix completion/sensing problems [16,17], which, at the pth iteration, computes V_p and W_p by alternately solving certain minimization problems. Although the algorithm computes highly accurate approximations, it can become very expensive as p increases (see Sect. 2.4). Another approach is to use successive rank-one approximations and successively compute pairs of vectors $\{(v_i, w_i)\}_{i=1}^{p}$ to build the factors V_p and W_p of (1.3) until a stopping criterion is satisfied. The pth iteration starts with V_{p-1} and W_{p-1} and constructs v_p and w_p as the solutions of certain minimization problems. This approach for solving parameterized PDEs is one component of a methodology known as Proper Generalized Decomposition (PGD) [30,31,46]. As observed in those works, using only successive rank-one approximations is less expensive but may not be practical because it typically results in approximations with an unnecessarily large value of p.

Our goal in this study is to develop solution algorithms that preserve only the good properties of the above two types of solution strategies, i.e., algorithms that compute an accurate low-rank approximate solution in a computationally efficient way. In developing such algorithms, we take our cue from PGD methods, in which, to improve accuracy, the successive rank-one constructions are supplemented with an *updating procedure* that is performed intermittently during the iteration. Inspired by this approach, we propose a solution algorithm that adaptively computes approximate solutions in an inexpensive way via the successive rank-one approximation method. This is supplemented by an enhancement procedure, which effectively improves the accuracy of the resulting approximate solutions. We propose two novel enhancement procedures developed by modifying some ideas in [17] used for matrix completion problems [38]. An algebraic formulation of PGD methods corresponds to alternating minimizations of errors in a particular norm, e.g., the energy norm or the ℓ^2 norm. Since we are considering linear systems with symmetric and positive definite matrices, we use the energy norm, and refer to the resulting methods as alternating energy minimization (AEM) methods.

Some other rank-adaptive approaches for approximating solutions of LMTMEs in low-rank format are as follows. An approach close to the ideas considered in this paper is a greedy low-rank algorithm, developed in [20], where the successive rank-one algorithm is followed by an enhancement procedure. This method is also used

in [23] for a regression problem arising in a neuroscientific model of synaptic connections. A method in [5], called the AMEn algorithm, uses AEM techniques in combination with tensor-train (TT) decompositions [32]; AMEn is designed for high-dimensional problems with multiple terms in the Kronecker product format. Another class of approaches includes incrementally computing rank-p solution pairs by solving a residual minimization problem, an approach known as alternating least-squares (ALS). This has been used to compute low-rank approximate solutions of parameterized PDEs in [6,7], and to solve matrix recovery problems, matrix sensing and completion problems, [15–17,38]. In [36], an adaptive iterative procedure to solve LMTMEs (1.2) associated with SGFEM discretizations is given, which incrementally computes a set of orthonormal basis vectors to form V_p which represents the spatial part of the solution. Two other classes of iterative low-rank algorithms are low-rank Krylov subspace methods [2,3,21,22,29,33,40,45] and low-rank multigrid methods [12,44]. These methods operate on iterates represented in Kronecker product format (e.g., $\sum_{i=1}^{\tilde{p}} w_i \otimes v_i$ for representing a vector with $\tilde{p}$ terms) and employ so-called truncation operators (e.g., based on singular value decomposition) to keep the iterates in low rank. See also [43] for a comprehensive overview of computational approaches for solving linear matrix equations.

Although we only exploit the abstract structure (1.2) to derive solution algorithms, we are motivated by the need to solve LMTMEs associated with SGFEM discretizations of parameterized PDEs arising in forward uncertainty quantification. Here we briefly mention their key features; more details are given in Sect. 4. Consider the model problem

$$-\nabla \cdot (a(x,\xi)\nabla u(x,\xi)) = f(x) \quad (x,\xi) \in D \times \Gamma, \tag{1.4}$$

where $D \subset \mathbb{R}^{2,3}$ is the *spatial domain* and the diffusion coefficient has the form

$$a(x,\xi) = a_0(x) + \sum_{i=1}^{m} a_i(x)\xi_i, \tag{1.5}$$

where $\xi = [\xi_1, \ldots, \xi_m]$ is a vector of m independent random variables taking values in a *parameter domain* $\Gamma \subset \mathbb{R}^m$. In the SGFEM approach, the solution $u(x,\xi)$ to (1.4) is approximated in a finite-dimensional space with tensor product structure $X_h \otimes S_d$ where X_h is a standard finite element space associated with D and S_d is a space of (usually, global) polynomials on Γ. Applying such a scheme leads to a LMTME (1.2) in which the K_i matrices are associated with the chosen finite element discretization, and the G_i matrices are associated with the polynomial approximation on the m-dimensional parameter domain. In particular, K_i is a finite element stiffness matrix weighted by the coefficient $a_i(x)$ appearing in (1.5). These matrices are ill-conditioned with respect to the mesh parameter h, and due to the properties of the coefficients in (1.5), they have decaying importance in terms of their contribution to the sum in (1.1). Moreover, the first term $G_0 \otimes K_0$ usually dominates and serves as an effective and computationally efficient preconditioner [35]. We will exploit this fact in numerical experiments in Sect. 4.

An outline of the rest of the paper is as follows. In Sect. 2, we introduce and derive alternating energy minimization (AEM) methods for (1.2) using the well-known general projection framework and discuss a collection of methods developed for constructing low-rank approximate solutions of the form (1.3). In Sect. 3, we discuss enhancement procedures and derive two new approaches for performing such updates. In Sect. 4, we perform extensive numerical experiments and measure the effectiveness and the efficiency of the enhanced AEM methods for LMTMEs associated with SGFEM discretizations of parameterized elliptic PDEs of the form (1.4). We also compare our methods with the greedy low-rank algorithm [20] on the same benchmark problems as that method shares many features with the proposed methods and can easily be described within the enhanced AEM framework. Finally, in Sect. 5, we draw some conclusions.

2 Alternating energy minimization (AEM) methods

In this section, we derive AEM methods for solving the matrix Eq. (1.2) from the optimal projection framework and review two variants of such methods. We first introduce some notation. Upper-case and lower-case letters are used to denote matrices and vectors, respectively. An inner product between two matrices $X, Y \in \mathbb{R}^{n_1 \times n_2}$ is defined as $\langle X, Y\rangle \equiv \mathrm{tr}(X^{\mathsf{T}}Y) = \mathrm{tr}(XY^{\mathsf{T}}) = \sum_{i,j} X_{ij}Y_{ij}$, where tr is the trace operator, and $\mathrm{tr}(X) = \sum_{i=1}^{n} x_{ii}$ if $X \in \mathbb{R}^{n\times n}$. The norm induced by $\langle\cdot,\cdot\rangle$ is the Frobenius norm $\|X\|_{\mathrm{F}} = \sqrt{\langle X, X\rangle}$. For shorthand notation, we introduce a linear operator $\mathscr{A}(X) = \sum_{i=0}^{m} K_i X G_i^{\mathsf{T}}$ for $X \in \mathbb{R}^{n_1\times n_2}$. Using this, we can define the weighted inner product $\langle X, Y\rangle_A = \langle \mathscr{A}(X), Y\rangle = \langle X, \mathscr{A}(Y)\rangle$ and the induced A-norm $\|\cdot\|_A$. Finally, vec denotes a vectorization operator, $\mathrm{vec}(X) = x$, where $X = [x_1, \ldots, x_{n_2}] \in \mathbb{R}^{n_1\times n_2}$ and $x = [x_1^{\mathsf{T}}, \ldots, x_{n_2}^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{n_1 n_2}$, for $x_i \in \mathbb{R}^{n_1}$, $i = 1, \ldots, n_2$.

2.1 General projection framework

For the computation of V_p and W_p in (1.3), we rely on the classical theory of orthogonal (Galerkin) projection methods [39]. Let $\mathscr{K} \subset \mathbb{R}^{n_1\times n_2}$ be a *search space* in which an approximate solution $U_p \in \mathbb{R}^{n_1\times n_2}$ is sought, and let $\mathscr{L}$ be a *constraint space* onto which the residual $B - \mathscr{A}(U_p)$ is projected. Following [39, Proposition 5.2], if the system matrix A is symmetric positive definite and $\mathscr{L} = \mathscr{K}$, then a matrix U_p^* is the result of an orthogonal projection onto $\mathscr{L}$ if and only if it minimizes the A-norm of the error over $\mathscr{K}$, i.e.,

$$U_p^* = \underset{U_p\in\mathscr{K}}{\arg\min}\, J_A(U_p),$$

where the objective function is

$$J_A(U_p) = \frac{1}{2}\|U - U_p\|_A^2. \tag{2.1}$$

Because we seek a factored representation of U_p, we slightly modify (2.1) to give

$$J_A(V_p, W_p) = \frac{1}{2}\|U - V_p W_p^{\mathsf{T}}\|_A^2, \tag{2.2}$$

and a new minimization problem

$$\min_{V_p \in \mathbb{R}^{n_1 \times p}, W_p \in \mathbb{R}^{n_2 \times p}} J_A(V_p, W_p). \tag{2.3}$$

Since J_A is quadratic, gradients with respect to V_p and W_p can be easily obtained as

$$\nabla_{V_p} J_A = \left(\mathscr{A}(V_p W_p^{\mathsf{T}}) - B\right) W_p = \sum_{i=0}^{m} (K_i V_p W_p^{\mathsf{T}} G_i^{\mathsf{T}}) W_p - B W_p, \tag{2.4}$$

$$\nabla_{W_p} J_A = \left(\mathscr{A}(V_p W_p^{\mathsf{T}}) - B\right)^{\mathsf{T}} V_p = \sum_{i=0}^{m} (K_i V_p W_p^{\mathsf{T}} G_i^{\mathsf{T}})^{\mathsf{T}} V_p - B^{\mathsf{T}} V_p. \tag{2.5}$$

Employing the first-order optimality condition on (2.4)–(2.5) (i.e., setting (2.4) and (the transpose of) (2.5) to be zero) results in the set of equations

$$\sum_{i=0}^{m} (K_i V_p W_p^{\mathsf{T}} G_i^{\mathsf{T}}) W_p = B W_p \in \mathbb{R}^{n_1 \times p}, \tag{2.6}$$

$$\sum_{i=0}^{m} V_p^{\mathsf{T}} (K_i V_p W_p^{\mathsf{T}} G_i^{\mathsf{T}}) = V_p^{\mathsf{T}} B \in \mathbb{R}^{p \times n_2}. \tag{2.7}$$

These equations can be interpreted as projections of the residual $B - \mathscr{A}(V_p W_p^{\mathsf{T}})$ onto the spaces spanned by the columns of W_p and V_p, respectively.

Given (2.6)–(2.7), a widely used strategy for solving the minimization problem (2.3) is to compute each component of the solution pair (V_p, W_p) alternately [5–7,15–17]. That is, one can fix W_p and solve the system of equations of order $n_1 p$ in (2.6) for V_p, and then one can fix V_p and solve the system of equations of order $n_2 p$ in (2.7) for W_p. However, in this approach, suitable choices of p for satisfying a fixed error tolerance are typically not known *a priori*. Thus, adaptive schemes that incrementally compute solution pairs (v_i, w_i) have been introduced [17,30,31,46]. All of these schemes are based on alternately solving two systems of equations for two types of variables in an effort to minimize a certain error measure. In this study, we employ alternating methods for minimizing the energy norm of the error (2.3) and, thus, we refer to approaches of this type as *alternating energy minimization* (AEM) methods. In the following sections, we present two adaptive variants of AEM methods: a Stage-p AEM method and a successive rank-one AEM method.

2.2 Stage-*p* AEM method

In [17], an ALS method that entails solving a sequence of least-squares problems whose dimensions increase with p was developed for solving matrix-recovery problems [15–17]. We adapt this approach to the energy minimization problem (2.3) and refer to it as the Stage-p AEM method. It is an iterative method that runs until an approximate solution satisfies a stopping criterion (e.g., the relative difference of two consecutive iterates $\|V_p W_p^{\mathsf{T}} - V_{p-1} W_{p-1}^{\mathsf{T}}\|_{\mathrm{F}} \leq \epsilon \|V_p W_p^{\mathsf{T}}\|_{\mathrm{F}}$ with a user-specified stopping tolerance ϵ.) At the pth iteration, called a "stage" in [17], this method seeks p-column factors V_p and W_p determining an approximate solution by initializing $W_p^{(0)}$ and solving the following systems of equations in sequence:

$$\sum_{i=0}^{m} (K_i) V_p^{(k)} (W_p^{(k-1)\mathsf{T}} G_i W_p^{(k-1)})^{\mathsf{T}} = B W_p^{(k-1)}, \tag{2.8}$$

$$\sum_{i=0}^{m} (V_p^{(k)\mathsf{T}} K_i V_p^{(k)}) W_p^{(k)\mathsf{T}} (G_i^{\mathsf{T}}) = V_p^{(k)\mathsf{T}} B, \tag{2.9}$$

for $k = 1, \ldots, k_{\max}$, where the superscript indicates the number of alternations between the two systems of Equations (2.8)–(2.9). Note that the method can also begin by initializing $V_p^{(0)}$ and alternating between (2.9) and (2.8). Algorithm 1 summarizes the entire procedure. The CHECKCONVERGENCE procedure (line 9) is detailed in Sect. 3. Terms of the form V_{p-1} or W_{p-1} that appear in several places for $p = 1$ (for example, in line 3 of Algorithm 1) correspond to null or "zero-column" matrices.

Algorithm 1 Stage-p AEM method

INPUT: $p_{\max}$: the maximum number of solution pairs,
$k_{\max}$: the maximum number of alternations in each stage,
ϵ: a parameter for checking convergence,

1: **function** STAGEPAEM($p_{\max}, k_{\max}, \epsilon$)
2: **for** $p = 1, \ldots, p_{\max}$ **do**
3: Set a random initial guess for $w_p^{(0)}$ and $W_p^{(0)} \leftarrow [W_{p-1}, w_p^{(0)}]$
4: **for** $k = 1, \ldots, k_{\max}$ **do**
5: $V_p^{(k)} \leftarrow$ solve (2.8)
6: $W_p^{(k)} \leftarrow$ solve (2.9)
7: **end for**
8: $V_p \leftarrow V_p^{(k)}$ and $W_p \leftarrow W_p^{(k)}$
9: $V_p, W_p \leftarrow$ CHECKCONVERGENCE(V_p, W_p, ϵ)
10: **end for**
11: **end function**

Systems of equations for "vectorized" versions of the matrix factors V_p and W_p can be derived[1] from (2.8) and (2.9) as follows

[1] The left-hand sides of (2.10)–(2.11) are derived using $\mathrm{vec}(KUG^{\mathsf{T}}) = (G \otimes K)\mathrm{vec}(U)$. Note that (2.11) is derived by first transposing (2.9) and then vectorizing the resulting equation.

$$\sum_{i=0}^{m}[(W_p^{(k-1)\mathsf{T}} G_i W_p^{(k-1)}) \otimes K_i]\,\mathrm{vec}(V_p^{(k)}) = \mathrm{vec}(B W_p^{(k-1)}), \tag{2.10}$$

$$\sum_{i=0}^{m}[(V_p^{(k)\mathsf{T}} K_i V_p^{(k)}) \otimes G_i]\,\mathrm{vec}(W_p^{(k)}) = \mathrm{vec}(B^{\mathsf{T}} V_p^{(k)}). \tag{2.11}$$

Thus, solving (2.8) entails solving a linear system of dimension $n_1 p \times n_1 p$, and solving (2.9) entails solving a system of dimension $n_2 p \times n_2 p$. Both systems are smaller than the original system (1.2) when p is small. However, the blocks of the reduced matrices of size $p \times p$ such as ($W_p^{\mathsf{T}} G_i W_p$ and $V_p^{\mathsf{T}} K_i V_p$) are dense, even if the original ones are sparse, and so as p increases, the computational costs for solving (2.8)–(2.9) increase and the Stage-p AEM method may be impractical for large-scale problems.

2.3 Successive rank-one AEM method

We now describe a successive rank-one (S-rank-1) approximation method which, at each iteration, adds a rank-one correction to the current iterate. This is a basic component of PGD methods [30,31,46] for solving parameterized PDEs. The method only requires solutions of linear systems with coefficient matrices of size $n_1 \times n_1$ and $n_2 \times n_2$ rather than coupled systems like those in the Stage-p AEM method that grow in size with the step counter p.

Assume that $p-1$ pairs of solutions are computed, giving V_{p-1} and W_{p-1}. The next step is to compute a new solution pair (v_p, w_p) by choosing the objective function

$$J_A(v_p, w_p) = \frac{1}{2}\|U - V_{p-1} W_{p-1}^{\mathsf{T}} - v_p w_p^{\mathsf{T}}\|_A^2,$$

and solving the following minimization problem

$$\min_{v_p \in \mathbb{R}^{n_1}, w_p \in \mathbb{R}^{n_2}} J_A(v_p, w_p).$$

The gradients of J_A with respect to v_p and w_p are

$$\nabla_{v_p} J_A = \left(\mathscr{A}(v_p w_p^{\mathsf{T}}) + \mathscr{A}(V_{p-1} W_{p-1}^{\mathsf{T}}) - B\right) w_p, \tag{2.12}$$

$$\nabla_{w_p} J_A = \left(\mathscr{A}(v_p w_p^{\mathsf{T}}) + \mathscr{A}(V_{p-1} W_{p-1}^{\mathsf{T}}) - B\right)^{\mathsf{T}} v_p. \tag{2.13}$$

Employing the first-order optimality conditions (setting (2.12) and (the transpose of) (2.13) to zero) results in systems of equations for which, in a succession of steps $k = 1, \ldots, k_{\max}$, v_p is updated using fixed w_p and then w_p is updated using fixed v_p:

$$\sum_{i=0}^{m}(K_i) v_p^{(k)} (w_p^{(k-1)\mathsf{T}} G_i w_p^{(k-1)})^{\mathsf{T}} = B w_p^{(k-1)} - \mathscr{A}(V_{p-1} W_{p-1}^{\mathsf{T}}) w_p^{(k-1)}, \tag{2.14}$$

Algorithm 2 Successive rank-one AEM method

INPUT: $p_{\max}$, $k_{\max}$, and ϵ
function SRANKONEAEM($p_{\max}, k_{\max}, \epsilon$)
  for $p = 1, \ldots, p_{\max}$ **do**
    Set a random initial guess for $w_p^{(0)}$.
    for $k = 1, \ldots, k_{\max}$ **do**
      $v_p^{(k)} \leftarrow$ solve (2.14)
      $w_p^{(k)} \leftarrow$ solve (2.15)
    end for
    $v_p \leftarrow v_p^{(k)}$ and $w_p \leftarrow w_p^{(k)}$
    Add to solution matrices, $V_p \leftarrow [V_{p-1}, v_p]$, $W_p \leftarrow [W_{p-1}, w_p]$
    $V_p, W_p \leftarrow$ CHECKCONVERGENCE(V_p, W_p, ϵ)
  end for
end function

$$\sum_{i=0}^{m} (v_p^{(k)\mathsf{T}} K_i v_p^{(k)}) w_p^{(k)\mathsf{T}} (G_i^{\mathsf{T}}) = v_p^{(k)\mathsf{T}} B - v_p^{(k)\mathsf{T}} \mathscr{A}(V_{p-1} W_{p-1}^{\mathsf{T}}). \tag{2.15}$$

Algorithm 2 summarizes this procedure, which randomly initializes $w_p^{(0)}$ and then alternately solves (2.14)–(2.15). Like the Stage-p AEM method, the algorithm can start with either $w_p^{(0)}$ or $v_p^{(0)}$.

2.4 Algebraic interpretation of the methods

Algorithms 1 and 2 both entail an "outer iteration" with counter p and an "inner iteration" with counter k, and both are designed to minimize the objective function (2.2). It is instructive to see the difference between the two methods in vectorized format. To this end, let

$$\mathscr{A}_w(w_i, w_j) = \sum_{l=0}^{m} K_l (w_j^{\mathsf{T}} G_l^{\mathsf{T}} w_i) \in \mathbb{R}^{n_1 \times n_1}, \quad \mathscr{A}_v(v_i, v_j) = \sum_{l=0}^{m} G_l (v_j^{\mathsf{T}} K_l^{\mathsf{T}} v_i) \in \mathbb{R}^{n_2 \times n_2},$$

and let us assume $p = 2$ for simplifying the presentation.

Both methods seek solution pairs $(V_2, W_2) = ([v_1, v_2], [w_1, w_2])$ satisfying the systems of Eqs. (2.6)–(2.7), which can be written in a vectorized form:

$$\begin{bmatrix} A_w(w_1, w_1) & A_w(w_1, w_2) \\ A_w(w_2, w_1) & A_w(w_2, w_2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} B w_1 \\ B w_2 \end{bmatrix}, \tag{2.16}$$

$$\begin{bmatrix} A_v(v_1, v_1) & A_v(v_1, v_2) \\ A_v(v_2, v_1) & A_v(v_2, v_2) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}} v_1 \\ B^{\mathsf{T}} v_2 \end{bmatrix}. \tag{2.17}$$

In the second outer iteration, the Stage-p AEM method alternately solves fully coupled linear systems (2.8)–(2.9) specified by $W_2^{(k-1)}$ and $V_2^{(k)}$, respectively , which can be

written in vectorized form as in (2.16) and (2.17), respectively:

$$\begin{bmatrix} A_w(w_1^{(k-1)}, w_1^{(k-1)}) & A_w(w_1^{(k-1)}, w_2^{(k-1)}) \\ A_w(w_2^{(k-1)}, w_1^{(k-1)}) & A_w(w_2^{(k-1)}, w_2^{(k-1)}) \end{bmatrix} \begin{bmatrix} v_1^{(k)} \\ v_2^{(k)} \end{bmatrix} = \begin{bmatrix} Bw_1^{(k-1)} \\ Bw_2^{(k-1)} \end{bmatrix},$$
$$\begin{bmatrix} A_v(v_1^{(k)}, v_1^{(k)}) & A_v(v_1^{(k)}, v_2^{(k)}) \\ A_v(v_2^{(k)}, v_1^{(k)}) & A_v(v_2^{(k)}, v_2^{(k)}) \end{bmatrix} \begin{bmatrix} w_1^{(k)} \\ w_2^{(k)} \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}} v_1^{(k)} \\ B^{\mathsf{T}} v_2^{(k)} \end{bmatrix}. \tag{2.18}$$

In contrast, the S-rank-1 method seeks approximate solutions of (2.16)–(2.17) by solving systems of equations associated with the diagonal blocks. In the first outer iteration, the method alternates between the following equations to find v_1 and w_1:

$$\begin{bmatrix} A_w(w_1^{(k-1)}, w_1^{(k-1)}) \end{bmatrix} \begin{bmatrix} v_1^{(k)} \end{bmatrix} = \begin{bmatrix} Bw_1^{(k-1)} \end{bmatrix},$$
$$\begin{bmatrix} A_v(v_1^{(k)}, v_1^{(k)}) \end{bmatrix} \begin{bmatrix} w_1^{(k)} \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}} v_1^{(k)} \end{bmatrix}.$$

In the second outer iteration, the method alternately solves the systems of equations in the second rows of the following equations to find v_2 and w_2:

$$\begin{bmatrix} A_w(w_1, w_1) & \\ A_w(w_2^{(k-1)}, w_1) & A_w(w_2^{(k-1)}, w_2^{(k-1)}) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2^{(k)} \end{bmatrix} = \begin{bmatrix} Bw_1 \\ Bw_2^{(k-1)} \end{bmatrix},$$
$$\begin{bmatrix} A_v(v_1, v_1) & \\ A_v(v_2^{(k)}, v_1) & A_v(v_2^{(k)}, v_2^{(k)}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2^{(k)} \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}} v_1 \\ B^{\mathsf{T}} v_2^{(k)} \end{bmatrix}.$$

Because v_1 and w_1 are fixed, the (2,1)-block matrices are multiplied by v_1 and w_1 and the resulting vectors are moved to the right-hand sides. Then solving the equations associated with the (2,2)-block matrices gives $v_2^{(k)}$ and $w_2^{(k)}$. As illustrated in this example, the S-rank-1 AEM method approximately solves (2.16)–(2.17) by taking the matrices in the lower-triangular blocks to the right-hand sides and solving only the systems associated with the diagonal blocks, as opposed to solving fully coupled systems as in the Stage-p AEM method.

The system matrices that arise in Algorithm 1 have reduced components that are dense but small (of size $p \times p$) whereas the "non-reduced" components are large but sparse. In Algorithm 2, the system matrices are sparse and of order n_1 and n_2 (as the reduced components are of size 1×1). Thus in both cases, we may use Krylov subspace methods to solve the systems. Then, with the iteration counter p, the cost of the Stage-p AEM method grows quadratically (since the reduced components are dense), whereas that of the S-rank-1 AEM method grows linearly with p. Thus, using the Stage-p AEM method can be impractical for large-scale applications. On the other hand, as the S-rank-1 AEM method employs only the lower-triangular part of the system matrices, convergence tends to be slow and the level of accuracy that can be achieved in a small number of steps is limited. To overcome these shortcomings, we will consider several ways to modify and enhance them to improve accuracy.

Remark 2.1 The Stage-p AEM and S-rank-1 AEM methods can be seen as two extreme versions of AEM methods. The former solves fully coupled systems and the latter sequentially solves systems associated with the diagonal blocks. Although it has not been explored in this study, in an intermediate approach, more than one consecutive pair of solution vectors $(\{v_p, \ldots, v_{p+\ell}\}, \{w_p, \ldots, w_{p+\ell}\})$, with $\ell \in \mathbb{N}$, can be computed in a coupled manner at each outer iteration.

3 Enhancements

We now describe variants of the S-rank-1 AEM method that perform extra computations to improve accuracy. The general strategy is to compute an enhancement of the approximate solution at every n_{update} outer iterations as specified in Algorithms 3–5.

Algorithm 3 Enhanced AEM method

INPUT: $p_{\max}$, $k_{\max}$, n_{update}, and ϵ
function ENHANCEDAEM($p_{\max}, k_{\max}, n_{\text{update}}, \epsilon$)
  for $p = 1, \ldots, p_{\max}$ **do**
    $v_p, w_p \leftarrow$ RANKONECORRECTION($V_{p-1}, W_{p-1}, k_{\max}$)
    Add to solution matrices, $V_p \leftarrow [V_{p-1}, v_p]$, $W_p \leftarrow [W_{p-1}, w_p]$
    if $p \mod n_{\text{update}} == 0$ **then**
      $V_p, W_p \leftarrow$ ENHANCEMENT(V_p, W_p)
    end if
    $V_p, W_p \leftarrow$ CHECKCONVERGENCE(V_p, W_p, ϵ)
  end for
end function

Algorithm 4 Rank one correction

INPUT: V_{p-1}, W_{p-1}, and $k_{\max}$
function RANKONECORRECTION($V_{p-1}, W_{p-1}, k_{\max}$)
  Set a random initial guess for $w_p^{(0)}$.
  for $k = 1, \ldots, k_{\max}$ **do**
    $v_p^{(k)} \leftarrow$ solve (2.14)
    $w_p^{(k)} \leftarrow$ solve (2.15)
  end for
  $v_p \leftarrow v_p^{(k)}$ and $w_p \leftarrow w_p^{(k)}$
end function

We present three enhancement procedures, one taken from the literature and two new ones. These are (i) a procedure adopted from an updating technique developed in [46, Section 2.5], which defines one variant of PGD methods; (ii) a refined version of this approach, which only solves systems associated with the diagonal blocks of the system matrices but incorporates information (upper-triangular blocks) in a manner similar to Gauss-Seidel iterations; and (iii) an adaptive enhancement of the Stage-p AEM method that decreases costs with negligible impact on accuracy. In discussing

Algorithm 5 Checking for convergence

INPUT: V_p, W_p, and ϵ
function CHECKCONVERGENCE(V_p, W_p, ϵ)
  if $\|V_p W_p^\mathsf{T} - V_{p-1} W_{p-1}^\mathsf{T}\|_\mathrm{F} \leq \epsilon \|V_p W_p^\mathsf{T}\|_\mathrm{F}$ **then**
    $V_p, W_p \leftarrow$ ENHANCEMENT(V_p, W_p)
    if $\|V_p W_p^\mathsf{T} - V_{p-1} W_{p-1}^\mathsf{T}\|_\mathrm{F} \leq \epsilon \|V_p W_p^\mathsf{T}\|_\mathrm{F}$ **then** Stop
    end if
  end if
end function

these ideas, we distinguish updated solutions using the notation, $\overline{v}_i$, $\overline{w}_i$ (for vectors), and $\overline{V}_p = [\overline{v}_1, \ldots, \overline{v}_p]$, $\overline{W}_p = [\overline{w}_1, \ldots, \overline{w}_p]$ (for matrices). In addition, we also review the method proposed in [23].

Before we detail each method, we first elaborate on the CHECKCONVERGENCE procedure in Algorithm 5. This checks the relative difference between the current iterate and the previous iterate $\|V_p W_p^\mathsf{T} - V_{p-1} W_{p-1}^\mathsf{T}\|_\mathrm{F} \leq \epsilon \|V_p W_p^\mathsf{T}\|_\mathrm{F}$ in the Frobenius norm. To compute $\|V_p W_p^\mathsf{T}\|_\mathrm{F}^2$ while avoiding explicitly forming the large matrix $V_p W_p^\mathsf{T}$, we form $X = (V_p^\mathsf{T} V_p) \odot (W_p^\mathsf{T} W_p) \in \mathbb{R}^{p \times p}$, where $\odot$ is the Hadamard product, and then sum up all the elements of X. The product $V_p W_p^\mathsf{T}$ is never explicitly formed. If this condition is met, we apply the ENHANCEMENT procedure and check the convergence with the same criterion. The purpose of this extra enhancement is to help prevent Algorithm 3 from terminating prematurely (i.e., the stopping condition can be met when Algorithm 3 stagnates).

3.1 PGD-updated AEM

Suppose the factors V_p and W_p obtained from RANKONECORRECTION do not satisfy the first-order optimality conditions (2.6)–(2.7). An enhancement like that of the PGD update [30,31,46] modifies one of these factors (e.g., the one corresponding to the smaller dimension n_1 or n_2) by solving the associated minimization problem for V_p (given W_p, when $n_1 < n_2$) or for W_p (given V_p when $n_1 > n_2$) so that one of the first-order conditions holds. We outline the procedure for approximating W_p; the procedure for V_p is analogous. The basic procedure is to solve the optimization problem $\min_{W_p \in \mathbb{R}^{n_2 \times p}} J_A\left(V_p, W_p\right)$ every n_{update} steps. In place of V_p, an orthonormal matrix $\widetilde{V}_p$ is used, so that the construction entails solving

$$\overline{W}_p = \underset{W_p \in \mathbb{R}^{n_2 \times p}}{\arg\min}\ J_A\left(\widetilde{V}_p, W_p\right), \tag{3.1}$$

where J_A is the quadratic objective function defined in (2.2). The gradient of the objective function J_A with respect to W_p can be computed as

$$\nabla_{W_p} J_A = \left(\mathscr{A}(\widetilde{V}_p W_p^\mathsf{T}) - B\right)^\mathsf{T} \widetilde{V}_p = \sum_{i=0}^{m} (K_i \widetilde{V}_p W_p^\mathsf{T} G_i^\mathsf{T})^\mathsf{T} \widetilde{V}_p - B^\mathsf{T} \widetilde{V}_p.$$

Algorithm 6 PGD-update enhancement

Input: V_p and W_p
function PGDUPDATE(V_p, W_p)
  if $n_1 < n_2$ **then**
    $\widetilde{W}_p \leftarrow$ orthonormalize W_p.
    $\overline{V}_p \leftarrow$ solve $\sum_{i=0}^{m}(K_i)\overline{V}_p(\widetilde{W}_p^{\mathsf{T}}G_i\widetilde{W}_p)^{\mathsf{T}} = B\widetilde{W}_p$
    $V_p \leftarrow \overline{V}_p$
  else
    $\widetilde{V}_p \leftarrow$ orthonormalize V_p.
    $\overline{W}_p \leftarrow$ solve $\sum_{i=0}^{m}(\widetilde{V}_p^{\mathsf{T}}K_i\widetilde{V}_p)\overline{W}_p^{\mathsf{T}}(G_i^{\mathsf{T}}) = \widetilde{V}_p^{\mathsf{T}}B$
    $W_p \leftarrow \overline{W}_p$
  end if
end function

Thus, solving the minimization problem (3.1) by employing the first-order optimality condition is equivalent to solving a system of equations similar in structure to (2.7),

$$\sum_{i=0}^{m}(\widetilde{V}_p^{\mathsf{T}}K_i\widetilde{V}_p)\overline{W}_p^{\mathsf{T}}(G_i^{\mathsf{T}}) = \widetilde{V}_p^{\mathsf{T}}B \in \mathbb{R}^{p\times n_2}. \tag{3.2}$$

Compared to the original system (1.2), the dimension of this matrix is reduced via a "single-sided" reduction; in (3.2), the reduction is on the side of the first dimension, i.e., n_1 is reduced to p. The vectorized form of this system, for $p = 2$, is

$$\begin{bmatrix} A_v(\tilde{v}_1,\tilde{v}_1)A_v(\tilde{v}_1,\tilde{v}_2) \\ A_v(\tilde{v}_2,\tilde{v}_1)A_v(\tilde{v}_2,\tilde{v}_2) \end{bmatrix}\begin{bmatrix} \overline{w}_1 \\ \overline{w}_2 \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}}\tilde{v}_1 \\ B^{\mathsf{T}}\tilde{v}_2 \end{bmatrix},$$

which has structure like that of the second system in (2.18) of the Stage-p AEM method. We summarize this single-sided enhancement method in Algorithm 6.

Remark 3.1 Another approach for computing a set of orthonormal basis vectors and computing a low-rank solution by solving a reduced system of type (3.2) is given in [36]. The MultiRB method of [36] incrementally computes a set of orthonormal basis vectors for the spatial part of the solution (i.e., $\widetilde{V}_p \in \mathbb{R}^{n_1\times p}$) using *rational Krylov subspace methods* and solves a reduced system for $\overline{W}_p$ and, consequently, $U_p = \widetilde{V}_p\overline{W}_p^{\mathsf{T}}$.

3.2 PGD/Gauss–Seidel-updated AEM

The second strategy for enhancement, like Algorithm 2 (and in contrast to PGD-updated AEM), only requires solutions of linear systems with coefficient matrices of dimensions $n_1 \times n_1$ and $n_2 \times n_2$, independent of p. As observed in Sect. 2.4, the S-rank-1 AEM method loosely corresponds to solving lower block-triangular systems of equations. We modify these computations by using more information (from the upper triangular part), as soon as it becomes available. This leads to a method that resembles the (block) Gauss–Seidel method for linear systems [14]. Suppose $\{(v_i, w_i)\}_{i=1}^{p}$ are

Algorithm 7 PGD/GS enhancement

Input: V_p and W_p
1: **function** PGD/GS(V_p, W_p)
2: **for** $l = 1, \ldots, p$ **do**
3: $\overline{v}_l \leftarrow$ solution of Eq. (3.3)
4: $\overline{w}_l \leftarrow$ solution of Eq. (3.4)
5: **end for**
6: $V_p \leftarrow \overline{V}_p$, $W_p \leftarrow \overline{W}_p$
7: **end function**

obtained from p iterations of Algorithm 3. When the condition on line 5 of Algorithm 3 is met, these quantities will be updated in sequence to produce $\{(\overline{v}_i, \overline{w}_i)\}_{i=1}^{p}$ using the most recently computed quantities. In particular, suppose the updated pairs $\{(\overline{v}_i, \overline{w}_i)\}_{i=1}^{l-1}$ have been computed. Then the lth pair (v_l, w_l) is updated as follows. First, given w_l, the update $\overline{v}_l$ is computed by solving

$$\mathscr{A}_w(w_l, w_l)\overline{v}_l = Bw_l - \sum_{i=1}^{l-1} \mathscr{A}_w(w_l, \overline{w}_i)\overline{v}_i - \sum_{i=l+1}^{p} \mathscr{A}_w(w_l, w_i)v_i. \tag{3.3}$$

Then given $\overline{v}_l$, $\overline{w}_l$ is computed by solving

$$\mathscr{A}_v(\overline{v}_l, \overline{v}_l)\overline{w}_l = B^{\mathsf{T}}\overline{v}_l - \sum_{i=1}^{l-1} \mathscr{A}_v(\overline{v}_l, \overline{v}_i)\overline{w}_i - \sum_{i=l+1}^{p} \mathscr{A}_v(\overline{v}_l, v_i)w_i. \tag{3.4}$$

With $p = 2$ as an example, in vector format, the first step of this enhancement is to update (v_1, w_1) to $(\overline{v}_1, \overline{w}_1)$ by solving the following equations:

$$\begin{bmatrix} A_w(w_1, w_1) & A_w(w_1, w_2) \end{bmatrix} \begin{bmatrix} \overline{v}_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} Bw_1 \end{bmatrix},$$
$$\begin{bmatrix} A_v(\overline{v}_1, \overline{v}_1) & A_v(\overline{v}_1, v_2) \end{bmatrix} \begin{bmatrix} \overline{w}_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}}\overline{v}_1 \end{bmatrix},$$

and the second step is to update (v_2, w_2) to $(\overline{v}_2, \overline{w}_2)$ by solving the second row of the following equations:

$$\begin{bmatrix} A_w(\overline{w}_1, \overline{w}_1) & A_w(\overline{w}_1, w_2) \\ A_w(w_2, \overline{w}_1) & A_w(w_2, w_2) \end{bmatrix} \begin{bmatrix} \overline{v}_1 \\ \overline{v}_2 \end{bmatrix} = \begin{bmatrix} B\overline{w}_1 \\ Bw_2 \end{bmatrix},$$
$$\begin{bmatrix} A_v(\overline{v}_1, \overline{v}_1) & A_v(\overline{v}_1, \overline{v}_2) \\ A_v(\overline{v}_2, \overline{v}_1) & A_v(\overline{v}_2, \overline{v}_2) \end{bmatrix} \begin{bmatrix} \overline{w}_1 \\ \overline{w}_2 \end{bmatrix} = \begin{bmatrix} B^{\mathsf{T}}\overline{v}_1 \\ B^{\mathsf{T}}\overline{v}_2 \end{bmatrix}.$$

This strategy, which we call the PGD/GS enhancement, is summarized in Algorithm 7. It is an alternative to Algorithm 6 and is also applied every n_{update} outer iterations. For a comparison of Algorithms 6 and 7, note that Algorithm 6 (PGD-update) works with a larger system but it can exploit the matricized representation (3.2). Once the system matrices $\widetilde{G}_i = \widetilde{W}_p^{\mathsf{T}} G_i \widetilde{W}_p$ or $\widetilde{K}_i = \widetilde{V}_p^{\mathsf{T}} K_i \widetilde{V}_p$ are formed, if it is not too large,

the system in (3.2) (of order $n_2 p$ in this example) can be approximately solved using an iterative method such as the preconditioned conjugate gradient (PCG) method. In contrast, Algorithm 7 (PGD/GS) requires sequential updates of individual components in Eqs. (3.3)–(3.4), but with smaller blocks, of order n_1 and n_2. As we will show in Sect. 4, the PGD/GS-updated AEM method exhibits better performance in some error measures.

We have found that in practice, the enhancement procedure can be improved by updating only a chosen subset of solution pairs rather than all the solution pairs $\{(v_i, w_i)\}_{i=1}^{p}$. We discuss a criterion to choose such a subset next.

3.3 Reduced stage-*p* AEM method

The third enhancement procedure excerpts and modifies certain computations in the Stage-p AEM method (Lines 5 and 6 in Algorithm 1) in a computationally efficient way. The procedure adaptively chooses solution pairs to be updated and solves reduced systems to update only those pairs. Let us assume for now that a subset of the solution pairs to be updated has been chosen. Denote the set of indices of those solution pairs by $\ell(p) \subseteq \{1, \ldots, p-1\}$ and the remaining indices by $\ell^{\mathrm{c}}(p) = \{1, \ldots, p-1\} \setminus \ell(p)$. Then the update is performed by solving the following equations for $\overline{V}_{\ell(p)}$ and $\overline{W}_{\ell(p)}$:

$$\sum_{i=0}^{m} (K_i) \overline{V}_{\ell(p)} (\widetilde{W}_{\ell(p)}^{\mathsf{T}} G_i \widetilde{W}_{\ell(p)})^{\mathsf{T}} = B \widetilde{W}_{\ell(p)} - \sum_{i=0}^{m} (K_i) V_{\ell^{\mathrm{c}}(p)} (\widetilde{W}_{\ell(p)}^{\mathsf{T}} G_i W_{\ell^{\mathrm{c}}(p)})^{\mathsf{T}}, \tag{3.5}$$

where $\widetilde{W}_{\ell(p)}$ is obtained by orthonormalizing the columns of $W_{\ell(p)}$, and

$$\sum_{i=0}^{m} (\widetilde{V}_{\ell(p)}^{\mathsf{T}} K_i \widetilde{V}_{\ell(p)}) \overline{W}_{\ell(p)}^{\mathsf{T}} (G_i^{\mathsf{T}}) = \widetilde{V}_{\ell(p)}^{\mathsf{T}} B - \sum_{i=0}^{m} (\widetilde{V}_{\ell(p)}^{\mathsf{T}} K_i V_{\ell^{\mathrm{c}}(p)}) W_{\ell^{\mathrm{c}}(p)}^{\mathsf{T}} (G_i^{\mathsf{T}}), \tag{3.6}$$

where $\widetilde{V}_{\ell(p)}$ is obtained by orthonormalizing the columns of $\overline{V}_{\ell(p)}$. Then, $V_{\ell(p)}$ and $W_{\ell(p)}$ are updated to $\overline{V}_{\ell(p)}$ and $\overline{W}_{\ell(p)}$, while $V_{\ell^{\mathrm{c}}(p)}$ and $W_{\ell^{\mathrm{c}}(p)}$ remain the same.

We now describe a criterion to choose a subset of the solution pairs to be updated. Let us assume that $p-1$ iterations of Algorithm 3 have been performed, and V_{p-1} and W_{p-1} have been computed. The pth solution pair (v_p, w_p) is then computed via Algorithm 4. If $p \bmod n_{\text{update}} = 0$, then a subset of the previous $p-1$ solution pairs is chosen by inspecting the angles between v_p and the columns of V_{p-1} and similarly for w_p and W_{p-1}. We normalize all vectors $\tilde{v}_i = \frac{v_i}{\|v_i\|_2}$ and compute $\beta_V = \widetilde{V}_{p-1}^{\mathsf{T}} \tilde{v}_p \in \mathbb{R}^{p-1}$ (the vector of cosines of the angles), and an analogous vector β_W using w_p and W_{p-1}. The entries of β_V and β_W indicate how far from orthogonal all previous vectors are to v_p and w_p. Ideally, we want the method to compute p left and right singular vectors of the solution U (i.e., $\beta_V = \beta_W = 0$). As the aim is to find good basis vectors for approximating U, it is undesirable to keep vectors that are far from being orthogonal to v_p and w_p. To resolve this, we choose a subset of columns of V_{p-1} and

Algorithm 8 Reduced stage-p enhancement

Input: V_p, W_p, and τ
function RSTAGEP(V_p, W_p, τ)
  Normalize the columns: $\tilde{v}_i = \frac{v_i}{\|v_i\|_2}$, $\tilde{w}_i = \frac{w_i}{\|w_i\|_2}$ for $i = 1, \ldots, p$
  Compute $\beta_V = \widetilde{V}_{p-1}^{\mathsf{T}} \tilde{v}_p$, $\beta_W = \widetilde{W}_{p-1}^{\mathsf{T}} \tilde{w}_p$
  Select $\ell(p) = \{i \in [1, \ldots, p-1] \mid |[\beta_V]_i| > \tau \text{ or } |[\beta_W]_i| > \tau\}$
  $\widetilde{W}_{\ell(p)} \leftarrow$ orthonormalize $W_{\ell(p)}$
  $\overline{V}_{\ell(p)} \leftarrow$ solve (3.5)
  $\widetilde{V}_{\ell(p)} \leftarrow$ orthonormalize $\overline{V}_{\ell(p)}$
  $\overline{W}_{\ell(p)} \leftarrow$ solve (3.6)
  $V_{\ell(p)} = \overline{V}_{\ell(p)}$, $W_{\ell(p)} = \overline{W}_{\ell(p)}$
end function

W_{p-1} for which the entries of β_V and β_W are too large; we fix $\tau > 0$ and choose

$$\ell(p) = \{i \in \{1, \ldots, p-1\} \mid |[\beta_V]_i| > \tau \text{ or } |[\beta_W]_i| > \tau\}. \tag{3.7}$$

Algorithm 8 summarizes the resulting reduced stage-p (R-stage-p) enhancement.

3.4 PGD-greedy AEM

As a baseline for comparison, we review the greedy low-rank method proposed in [20] and further examined in [23], which can be interpreted as another variant of the S-rank-1 AEM method with ENHANCEMENT. We denote this method as PGD-greedy AEM in this study. The method seeks an approximate solution in a three-factor form $U_p = \widetilde{V}_p Z_p \widetilde{W}_p^{\mathsf{T}}$, where the columns of $\widetilde{V}_p$ and $\widetilde{W}_p$ are orthonormal; this can be achieved by (i) slightly modifying RANKONECORRECTION (Algorithm 4) and (ii) employing a particular enhancement procedure. The modified version of RANKONECORRECTION computes a new basis pair $(\tilde{v}_p, \tilde{w}_p)$, where each vector has unit norm, by setting $\tilde{w}_p^{(0)}$ to have unit norm and alternately performing the following procedure:

$$\begin{aligned}
&\text{Solve } \mathscr{A}_w(\tilde{w}_p^{(k-1)}, \tilde{w}_p^{(k-1)}) v_p^{(k)} = B\tilde{w}_p^{(k-1)} - \mathscr{A}(U_{p-1}^{\mathsf{T}})\tilde{w}_p^{(k-1)}, && \tilde{v}_p^{(k)} \leftarrow v_p^{(k)}/\|v_p^{(k)}\|_2,\\
&\text{Solve } \mathscr{A}_v(\tilde{v}_p^{(k)}, \tilde{v}_p^{(k)}) w_p^{(k)} = B^{\mathsf{T}}\tilde{v}_p^{(k)} - \mathscr{A}(U_{p-1}^{\mathsf{T}})^{\mathsf{T}} v_p^{(k)}, && \tilde{w}_p^{(k)} \leftarrow w_p^{(k)}/\|w_p^{(k)}\|_2.
\end{aligned}$$

This inner iteration continues until a stopping criterion $\left| \|v_p^{(k)}\|_2 / \|w_p^{(k)}\|_2 - 1 \right| \le \delta$ is satisfied, where δ is a stopping tolerance. Then (at each outer iteration) the approximate solution U_p is computed by solving a reduced linear system of equations, which is obtained via a "double-sided" reduction, for Z_p such that

$$\widetilde{V}_p^{\mathsf{T}} \mathscr{A}(\widetilde{V}_p Z_p \widetilde{W}_p^{\mathsf{T}}) \widetilde{W}_p = \widetilde{V}_p^{\mathsf{T}} B \widetilde{W}_p \in \mathbb{R}^{p \times p}. \tag{3.8}$$

We refer readers to [20,23] for more details on this method.

At the pth outer iteration, this double-sided reduction technique requires the computation of the solution of Eq. (3.8) which is of size $p \times p$, whereas the PGD-update

method employs a single-sided reduction requiring the computation of solutions of Eq. (2.7) which are of size $\min(n_1, n_2) \times p$. The R-stage-p method, on the other hand, updates only the solution pairs chosen based on the criterion (3.7) and, thus, the size of problems arising in that approach is affected by the value of τ.

4 Numerical experiments

In this section, we present the results of numerical experiments with the algorithms described in Sects. 2 and 3. For benchmark problems, we consider stochastic diffusion problems, where the stochasticity is assumed to be characterized by a prescribed set of m real-valued random variables. We apply suitable SGFEM discretizations to these problems, resulting in LMTMEs of the form (1.2) whose system matrices are symmetric positive-definite. All experiments are performed on an INTEL 3.1 GHz i7 CPU, with 16 GB RAM, using MATLAB R2019b.

4.1 Stochastic diffusion problems

Let $(\Omega, \mathscr{F}, P)$ be a probability space and let $D = [0, 1] \times [0, 1]$ be the spatial domain. Next, let $\xi_i : \Omega \to \Gamma_i \subset \mathbb{R}$, for $i = 1, \ldots, m$, be independent and identically distributed random variables and define $\xi = [\xi_1, \ldots, \xi_m]$. Then, $\xi : \Omega \to \Gamma$ where $\Gamma \equiv \prod_{i=1}^{m} \Gamma_i$ denotes the image. Given a second-order random field $a : D \times \Gamma \to \mathbb{R}$, we consider the following boundary value problem with constant forcing term $f(x) = 1$. Find $u : D \times \Gamma \to \mathbb{R}$ such that

$$\begin{cases} -\nabla \cdot (a(x, \xi)\nabla u(x, \xi)) = f(x) & \text{in } D \times \Gamma, \\ u(x, \xi) = 0 & \text{on } \partial D \times \Gamma, \end{cases} \tag{4.1}$$

where $a(x, \xi)$ has the form (1.5) and the ξ_i are chosen to be independent uniform random variables. Note that (1.5) has the same structure as a truncated Karhunen-Loève (KL) expansion [27]. If we denote the joint probability density function of ξ by $\rho(\xi)$ then the expected value of a random function $v(\xi)$ on Γ is $\langle v \rangle_\rho = \int_\Gamma v(\xi)\rho(\xi)d\xi$.

For the discretization, we consider the stochastic Galerkin method [1,13,28,47], which seeks an approximation to the solution of the following weak formulation of (4.1): Find $u(x, \xi)$ in $V = H_0^1(D) \otimes L_\rho^2(\Gamma)$ such that

$$\left\langle \int_D a(x, \xi)\nabla u(x, \xi) \cdot \nabla v(x, \xi)dx \right\rangle_\rho = \left\langle \int_D f(x)v(x, \xi)dx \right\rangle_\rho, \quad \forall v \in V. \tag{4.2}$$

In particular, we seek an approximation of form $\tilde{u}(x, \xi) = \sum_{s=1}^{n_\xi} \sum_{r=1}^{n_x} u_{rs}\phi_r(x)\psi_s(\xi)$, where $\{\phi_r\}_{r=1}^{n_x}$ is a set of standard finite element basis functions, which arises from using continuous piecewise bilinear approximation on a uniform mesh of square elements (Q1 elements) and n_x is related to the refinement level of the spatial mesh (our implementation uses the Incompressible Flow & Iterative Solver Software (IFISS) [11,42]). In addition, $\{\psi_s\}_{s=1}^{n_\xi}$ is chosen to be a finite subset of the set of orthonor-

mal polynomials that provides a basis for $L^2_\rho(\Gamma)$ (also known as a generalized polynomial chaos (gPC), [48]). As the random variables are uniformly distributed, we use m-variate normalized Legendre polynomials $\{\psi_s\}_{s=1}^{n_\xi}$, which are constructed as products of univariate Legendre polynomials, $\psi_s(\xi) = \prod_{i=1}^m \pi_{d_i(s)}(\xi_i)$. Here, $d(s) = (d_1(s), \ldots, d_m(s))$ is a multi-index and $\pi_{d_i(s)}$ is the $d_i(s)$-order univariate Legendre polynomial in ξ_i. A set of multi-indices $\{d(s)\}_{s=1}^{n_\xi}$ is specified as a set $\Lambda_{m,\,d_{\text{tot}}} = \{d(s) \in \mathbb{N}_0^m : \|d(s)\|_1 \leq d_{\text{tot}}\}$, where $\mathbb{N}_0$ is the set of non-negative integers, and $\|d(s)\|_1 = \sum_{j=1}^m d_j(s)$. With this construction, $\text{span}(\{\psi_s(\xi)\}_{s=1}^{n_\xi})$ is the set of polynomials in ξ of total degree d_{tot} or less, and with dimension $n_\xi = \dim(\Lambda_{m,\,d_{\text{tot}}}) = \frac{(m+d_{\text{tot}})!}{m!d_{\text{tot}}!}$.

Galerkin projection of (4.2) onto the chosen finite-dimensional space (i.e., using the same test basis functions as the trial basis functions), with the coefficients of the solution expansion ordered as $u = [u_{11}, \ldots, u_{n_x 1}, u_{12}, \ldots, u_{n_x n_\xi}]^\mathsf{T}$ results in

$$\left(\sum_{i=0}^{m} G_i \otimes K_i\right) u = g_0 \otimes f_0, \tag{4.3}$$

where the system matrices are defined as

$$[G_0]_{st} = \langle \psi_s(\xi)\psi_t(\xi)\rangle_\rho\,, \qquad [K_0]_{k\ell} = \int_D a_0(x)\nabla\phi_k(x)\cdot\nabla\phi_\ell(x)dx,$$
$$[G_i]_{st} = \langle \xi_i\,\psi_s(\xi)\psi_t(\xi)\rangle_\rho\,, \qquad [K_i]_{k\ell} = \int_D a_i(x)\nabla\phi_k(x)\cdot\nabla\phi_\ell(x)dx,$$

for $i = 1, \ldots, m$, $s, t = 1, \ldots, n_\xi$ and $k, \ell = 1, \ldots n_x$. Due to the deterministic forcing term $f(x) = 1$, the right-hand side has a rank-one structure (i.e., $\hat{m} = 0$ in (1.1)), with $[f_0]_k = \int_D f(x)\phi_k(x)dx$, and $[g_0]_s = \langle\psi_s(\xi)\rangle_\rho$. Matricizing (4.3) gives an LMTME of the form (1.2) with $n_1 = n_x$ and $n_2 = n_\xi$, where m is the number of terms in Eq. (1.5), and we can now apply the AEM methods to compute an approximate solution.

4.2 Benchmark problem 1: separable exponential covariance

In this problem, we assume that the random field $a(x, \xi)$ is a truncated KL expansion

$$a(x,\xi) = \mu + \sigma \sum_{i=1}^{m} \sqrt{\lambda_i}\varphi_i(x)\xi_i, \tag{4.4}$$

where μ is the mean of $a(x, \xi)$, $\{(\varphi_i(x), \lambda_i)\}_{i=1}^m$ are eigenpairs of the integral operator associated with the covariance kernel $C(x,\ y) \equiv \exp\left(-\frac{|x_1-y_1|}{c} - \frac{|x_2-y_2|}{c}\right)$, c is the associated correlation length, and σ^2 is the variance of the untruncated random field. In addition, each $\xi_i \sim U(-\sqrt{3}, \sqrt{3})$ and so has mean zero and variance one.

In the following sections, we compare the six AEM variants: Stage-p (Algorithm 1), S-rank-1 (Algorithm 2), PGD-updated (Algorithm 6), PGD/GS-updated (Algorithm

7), reduced stage-p (Algorithm 8), and PGD-greedy from [20,23]. For orthonormalization in PGD-updated (Algorithm 6) and reduced stage-p (Algorithm 8), we use MATLAB's `qr` function to implement the so-called skinny QR method. For assessing performance, we explore two key aspects. The first is the accuracy of the computed solutions, which we assess by computing two error metrics: cosines of angles between the truth singular vectors and the columns of the computed factors (Sect. 4.2.1), and errors between the truth solution and the computed solution measured in three different norms (Sect. 4.2.2). The second aspect is timings and scalability (Sect. 4.2.3). As the assessment of the first aspect requires the ground truth solution of (4.3), which is computed using MATLAB's backslash operator, and its singular vectors, we choose small-sized problems in Sects. 4.2.1–4.2.2. When making comparisons with the truth solution, we force all the AEM methods to execute $p_{\max} = \min(n_x, n_\xi) = 56$ iterations. Larger problems are considered in Sect. 4.2.3, where scalability matters and finding the truth solution is impossible with the available resources.

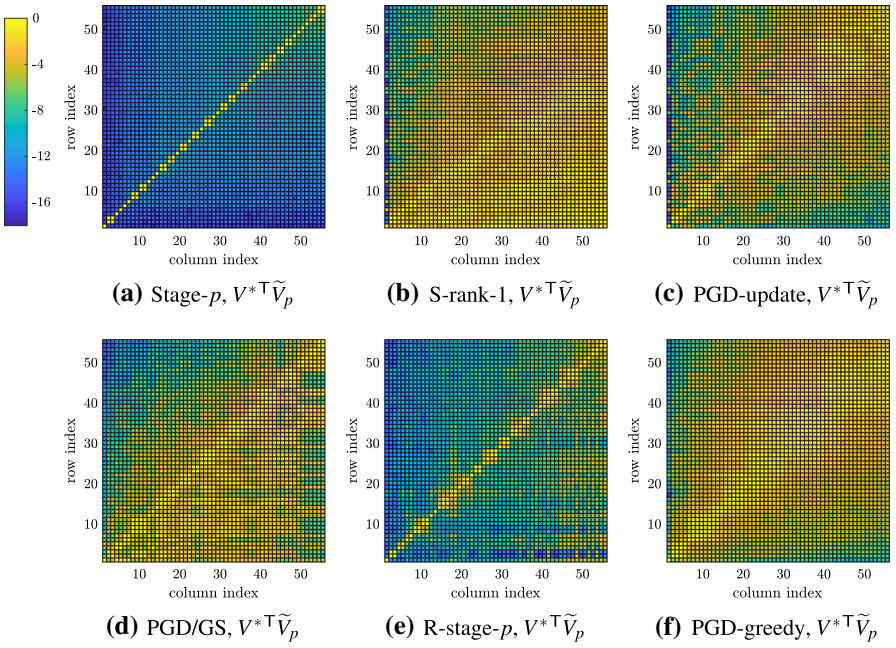
In producing experimental results in Sects. 4.2.1–4.2.2, we attempt to see each method's best possible results without considering the computational costs. Hence, we set $k_{\max} = 5, n_{\text{update}} = 1$ (i.e., enhancements are performed at every outer iteration) in Algorithm 3. For the same reason, we set PGD/GS to update all the solution pairs and, for R-stage-p, we set $\tau = .001$. For PGD-greedy we use $\delta = .1$ (the inner iteration stopping tolerance) following [23].[2] All linear systems that arise in each AEM method are solved using PCG. The stopping criterion for these iterations is for the relative residual to be less than the stopping tolerance 10^{-12}. We also apply so-called mean-based preconditioners, which we will discuss in detail in Sect. 4.2.3.

4.2.1 Relation to singular vectors

We begin by exploring how the factors in the approximate solutions constructed by each of the methods compare with the left and right singular vectors of the true solution matrix U. This is important because (i) singular vectors represent the most effective choice with respect to the Frobenius norm for approximating a matrix U. That is, the minimum error over all rank-p approximations is $\|U - \widetilde{V}_p \Sigma_p \widetilde{W}_p^{\mathsf{T}}\|_{\mathrm{F}}$, where $U = \widetilde{V} \Sigma \widetilde{W}^{\mathsf{T}}$ is the singular value decomposition [8], and (ii) in some applications such as collaborative filtering for recommendation systems, computing singular vectors accurately is very important for precise predictions [17,19,49]. For these tests, the diffusion coefficient is given by (4.4) with $(\mu, \sigma) = (1, .1)$ and $c = 2$. We use a spatial discretization with grid level 4 (i.e., grid spacing $\frac{1}{2^4}$, and $n_x = 225$) and we truncate the expansion (4.4) at $m = 5$. For the SGFEM approximation, we choose $d_{\text{tot}} = 3$ which gives $n_\xi = 56$.

For any approximation of the form (1.3), let $\widetilde{V}_p$ and $\widetilde{W}_p$ be normalized versions of the factors, i.e., each column of $\widetilde{V}_p$ and $\widetilde{W}_p$ is scaled to have unit norm. From the ground truth solution U, the matrices V^* and W^* of left and right singular vectors are computed. The entries of $V^{*\mathsf{T}} \widetilde{V}_p$, the cosines of the angles between the left singular vectors of the true solution and the left vectors defining the approximate solution,

[2] We also tested a variant of PGD-greedy that performed $k_{\max} = 5$ inner iteration steps (as done for the other EnhancedAEM methods) and observed only minor differences.



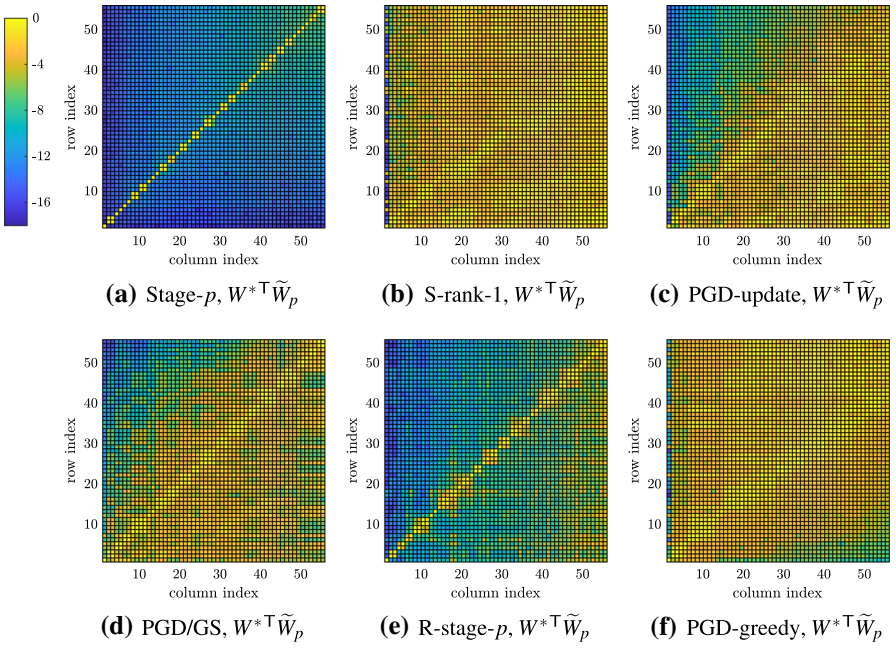
(a) Stage-p, $V^{*\mathsf{T}}\widetilde{V}_p$ **(b)** S-rank-1, $V^{*\mathsf{T}}\widetilde{V}_p$ **(c)** PGD-update, $V^{*\mathsf{T}}\widetilde{V}_p$

(d) PGD/GS, $V^{*\mathsf{T}}\widetilde{V}_p$ **(e)** R-stage-p, $V^{*\mathsf{T}}\widetilde{V}_p$ **(f)** PGD-greedy, $V^{*\mathsf{T}}\widetilde{V}_p$

Fig. 1 Cosines of angles (plotted in log scale) between the left singular vectors V^* and $\widetilde{V}_p$, where $\widetilde{V}_p$ are computed using the Stage-p and S-rank-1 AEM methods, and the EnhancedAEM methods with PGD-update, PGD/GS, R-stage-p enhancements, and PGD-greedy

together with the analogous angles for the right vectors, $W^{*\mathsf{T}}\widetilde{W}_p$, give insight into the quality of the approximate solution. Figs. 1a and 2a and Figs. 1b and 2b depict the cosines of the angles between the singular vectors and the columns of $\widetilde{V}_p$ and $\widetilde{W}_p$ computed using the Stage-p AEM and S-rank-1 AEM methods discussed in Sect. 2. It can be seen from these results (in Figs. 1a and 2a) that the Stage-p AEM method does a good job of approximating the singular vectors of the solution. That is, the values of the diagonal entries are close to one and the values of the off-diagonal entries are close to zero. On the other hand, the S-rank-1 AEM method (see Figs. 1b and 2b) is far less effective. The 2×2 blocks on the diagonals in Figs. 1a and 1a reflect the presence of equal singular values.

Figures 1c–f and 2c–f show analogous results for the four enhanced AEM methods. With PGD-update, the spatial component gets reduced (i.e., we form $\widetilde{K}_i = \widetilde{V}_p^{\mathsf{T}} K_i \widetilde{V}_p$) and W_p is updated. Figures 1c and 2c show that this computation improves the quality of the resulting factor W_p (and V_p as well) as approximate singular vectors, compared to those obtained with the S-rank-1 method. It is evident that PGD/GS further improves the quality of $\widetilde{V}_p$ and $\widetilde{W}_p$ (Figs. 1d and 2d) as approximate singular vectors, and R-stage-p is nearly as effective as Stage-p (Figs. 1e and 2e). The PGD-greedy AEM method is less effective in finding the left and the right singular vectors (Figs. 1f and 2f) than PGD/GS and R-stage-p.



(a) Stage-p, $W^{*\mathsf{T}}\widetilde{W}_p$ **(b)** S-rank-1, $W^{*\mathsf{T}}\widetilde{W}_p$ **(c)** PGD-update, $W^{*\mathsf{T}}\widetilde{W}_p$

(d) PGD/GS, $W^{*\mathsf{T}}\widetilde{W}_p$ **(e)** R-stage-p, $W^{*\mathsf{T}}\widetilde{W}_p$ **(f)** PGD-greedy, $W^{*\mathsf{T}}\widetilde{W}_p$

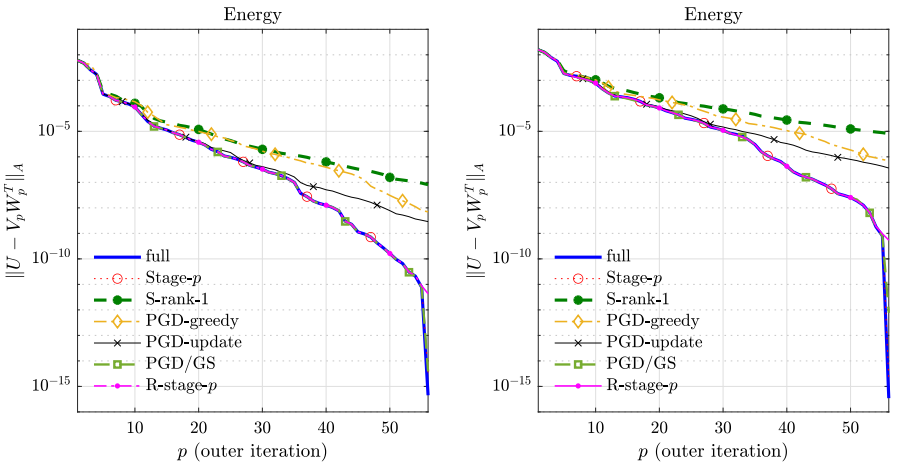
Fig. 2 Cosines of angles (plotted in log scale) between the right singular vectors W^* and $\widetilde{W}_p$, where $\widetilde{W}_p$ are computed using the Stage-p and S-rank-1 AEM methods, and the EnhancedAEM methods with PGD-update, PGD/GS, R-stage-p enhancements, and PGD-greedy

4.2.2 Assessment of solution accuracy

We now compare the convergence behavior of the variants of the AEM methods introduced in Sects. 2 and 3. We use two different settings for the stochastic diffusion coefficient: [exp1] $(\mu, \sigma) = (1, .1)$, $c = 2$ and [exp2] $(\mu, \sigma) = (1, .2)$, $c = .5$. We again truncate the series (4.4) at $m = 5$ and, for the Legendre basis polynomials, we consider $d_{\text{tot}} = 3$ which gives $n_\xi = 56$. We deliberately keep the same value for m and d_{tot} for both settings so that we can keep the dimensions of the problem the same and, thus, directly compare the behavior of each method in different problem settings.

For each method, the approximate solution U_p is computed and we measure the accuracy compared to the reference solution U. We did this using three different metrics: the energy norm error $\|U - U_p\|_A$, the error in the Frobenius norm $\|U - U_p\|_{\text{F}}$, and the residual in the Frobenius norm $\|B - \mathscr{A}(U_p)\|_{\text{F}}$. Note that direct computation of the residual norm is only possible due to the small size of the problem. See [22] for a method to compute this norm using skinny QR factorization in cases where m, p and $\hat{m}$ are not too large. Here, we only report the energy norm errors (in Fig. 3), as behavior for the other two metrics is virtually identical. For comparison, a rank-p reference solution (referred to as "full" in Fig. 3) is obtained directly from the first p singular values and singular vectors of U.

For both settings, as expected, the convergence behavior of the S-rank-1 AEM method is significantly worse than that of the rank-p reference solution, whereas



(a) Energy norm - Exp1 **(b)** Energy norm - Exp2

Fig. 3 Solution errors measured in the energy norm

that of the Stage-p AEM method is virtually the same as for the full direct solver. The EnhancedAEM method with PGD-update converges well until a certain level of accuracy is achieved, but it fails to achieve a high level of accuracy. The EnhancedAEM methods with PGD/GS and R-stage-p enhancement are more effective than with the PGD-update. The accuracy that those two methods achieve is virtually the same as that of the Stage-p AEM method and the full direct solver.

4.2.3 Computational timings

The above results do not account for computational costs; we now investigate timings under various experimental settings with a more practical outer stopping criterion. This is important for large-scale applications, and so we now consider a finer spatial grid, with grid level 6 (i.e., grid spacing $\frac{1}{2^6}$, and $n_x = 3969$), as well as larger parameter spaces, with $m = \{20, 24\}$ (the number of random variables in (4.4)) and $d_{\text{tot}} = 4$, which results in $n_\xi = \{10626, 20475\}$. We use the same settings for the stochastic diffusion coefficient [exp1] $(\mu, \sigma) = (1, .1)$, $c = 2$ and [exp2] $(\mu, \sigma) = (1, .2)$, $c = .5$. Again, we set m and d_{tot} to be the same for both problems, as we want to keep the dimensions fixed so that we can make direct and fair comparisons.

Before we present these results, we summarize the systems of equations to be solved for each of the EnhancedAEM methods and the adjustable parameters that affect the performances of the methods.[3] We first describe how we solve the systems arising at the pth outer iteration when the condition for applying the enhancement is met, as well as the systems arising in RankOneCorrection (Algorithm 4). We use PCG to solve each system of equations using mean-based preconditioners [35], which are

[3] The results of using the Stage-p and S-rank-1 AEM methods are not reported because the Stage-p AEM method is computationally too expensive and the S-rank-1 AEM method exhibits poor convergence behavior and, indeed, fails to satisfy the given convergence criterion.

Table 1 System matrices and preconditioners for each ENHANCEMENT procedure

Name	X	$\widetilde{K}_i$	$\widetilde{G}_i$	M_x	M_ξ	Eqs
S-rank-1	v_p	K_i	$w_p^\mathsf{T} G_i w_p$	K_0	1	(2.14)
(Alg. 4)	w_p^T	$v_p^\mathsf{T} K_i v_p$	G_i	1	G_0	(2.15)
PGD-update	V_p	K_i	$\widetilde{W}_p^\mathsf{T} G_i \widetilde{W}_p$	K_0	$\widetilde{W}_p^\mathsf{T} G_0 \widetilde{W}_p$	
(Alg. 6)	W_p^T	$\widetilde{V}_p^\mathsf{T} K_i \widetilde{V}_p$	G_i	$\widetilde{V}_p^\mathsf{T} K_0 \widetilde{V}_p$	G_0	(3.2)
PGD/GS	v_l	K_i	$w_l^\mathsf{T} G_i w_l$	K_0	1	(3.3)
(Alg. 7)	w_l^T	$\bar{v}_l^\mathsf{T} K_i \bar{v}_l$	G_i	1	G_0	(3.4)
R-stage-p	$V_{\ell(p)}$	K_i	$\widetilde{W}_{\ell(p)}^\mathsf{T} G_i \widetilde{W}_{\ell(p)}$	K_0	$\widetilde{W}_{\ell(p)}^\mathsf{T} G_0 \widetilde{W}_{\ell(p)}$	(3.5)
(Alg. 8)	$W_{\ell(p)}^\mathsf{T}$	$\widetilde{V}_{\ell(p)}^\mathsf{T} K_i \widetilde{V}_{\ell(p)}$	G_i	$\widetilde{V}_{\ell(p)}^\mathsf{T} K_0 \widetilde{V}_{\ell(p)}$	G_0	(3.6)
PGD-greedy	Z	$\widetilde{V}_p^\mathsf{T} K_i \widetilde{V}_p$	$\widetilde{W}_p^\mathsf{T} G_i \widetilde{W}_p$	$\widetilde{V}_p^\mathsf{T} K_0 \widetilde{V}_p$	$\widetilde{W}_p^\mathsf{T} G_0 \widetilde{W}_p$	(3.8)

constructed using reduced versions of the matrices K_0 and G_0, that are adapted to each method. For all systems, each PCG iteration requires matrix-vector products in the matricized form (see [2,22,26] for detailed matrix operations)

$$\sum_{i=0}^{m} (M_x^{-1} \widetilde{K}_i) X (M_\xi^{-1} \widetilde{G}_i)^\mathsf{T},$$

where X is a quantity to be updated, $\widetilde{K}_i$ and $\widetilde{G}_i$ are reduced matrices, and M_x and M_ξ are the preconditioner factors. Table 1 summarizes each system matrix and preconditioner.[4]

Now, we discuss adjustable parameters. The EnhancedAEM methods (Algorithms 3–5) require parameters $p_{\max}$, $k_{\max}$, n_{update}, and ϵ. We set $p_{\max} = 1000$ to prevent excessive computations. We found that choosing $k_{\max} > 2$ results in negligible difference in accuracy, but requires extra computations and, thus, we use $k_{\max} = \{1, 2\}$. For n_{update}, which determines how often the enhancement procedure is called, we vary n_{update} as $\{5, 10, 20, 30\}$. Next, we use ϵ to check the convergence of the outer iteration in Algorithm 5), and we vary ϵ as $\{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}\}$. Finally, for PGD/GS and R-stage-p, we empirically found that choosing $\tau > 0.05$ results in decreased accuracy in the approximate solution and, thus, we set $\tau = 0.05$. Again, for PGD-greedy we use $\delta = .1$ (the inner iteration stopping tolerance) following [23].

Next, we set parameters for the PCG method. For all systems, the stopping criterion uses the relative residual in the Frobenius norm. We use two different tolerances: τ_{basis} for solving systems that arise in RANKONECORRECTION and PGD/GS, and τ_{coupled} for solving systems that arise in PGD-update, R-stage-p, and PGD-greedy. Table 2 summarizes the parameters used for the experiments. We found that larger values of

[4] Note that, for PGD-update, one can always choose the smallest solution component to update. In practice, however, updating the W_p component (i.e., reduction in $\{K_i\}_{i=0}^m$) always requires the smallest computational costs and, thus, we only report the result of updating W_p.

Table 2 Parameters used in the experiments for measuring timings

The maximum number of outer iterations	$p_{\max} = 1000$
The maximum number of inner iterations	$k_{\max} = \{1, 2\}$
The frequency of the enhancement procedure	$n_{\text{update}} = \{5, 10, 20, 30\}$
The stopping tolerance for outer iterations	$\epsilon = \{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}\}$
PCG stopping tolerance for RankOneCorrection and PGD/GS	$\tau_{\text{basis}} = 10^{-5}$
PCG stopping tolerance for PGD-update and R-stage-p	$\tau_{\text{coupled}} = 10^{2}\epsilon$

the tolerances τ_{basis} and τ_{coupled} led to poor performance and smaller values did not improve accuracy.

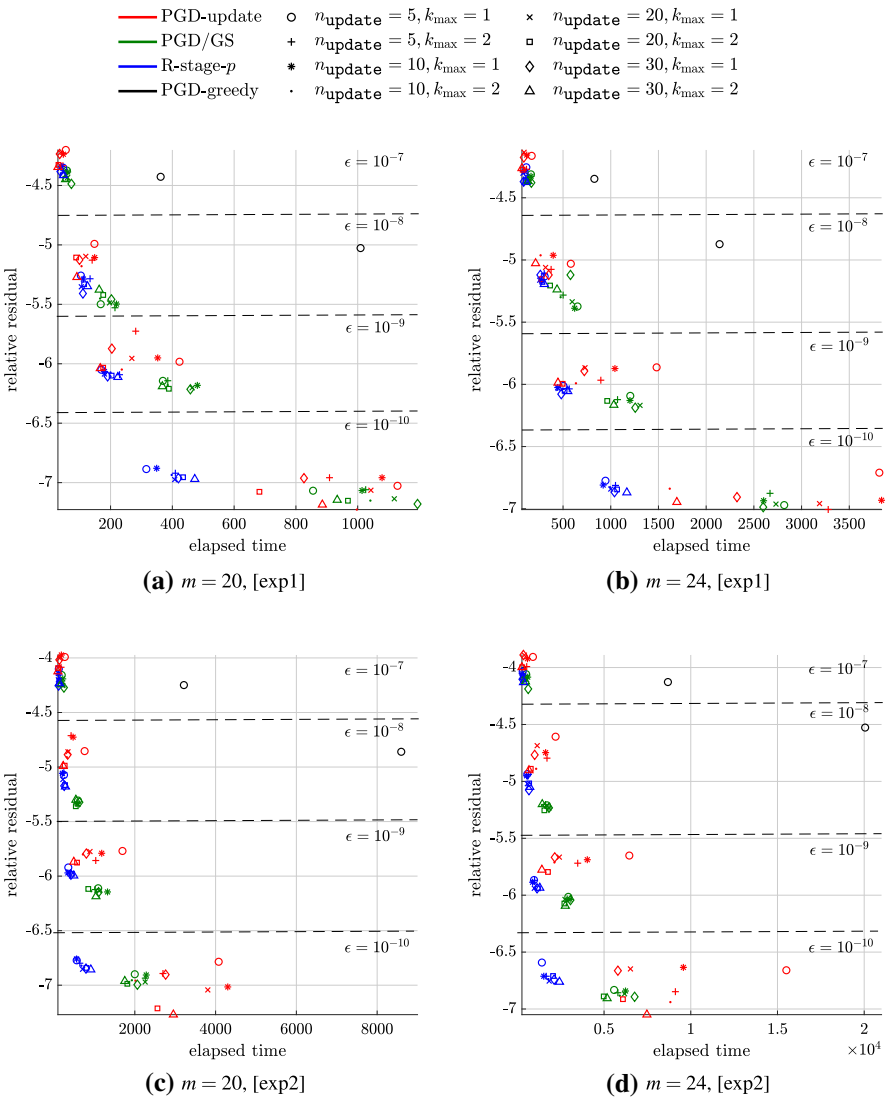
In Fig. 4, we plot elapsed time (in seconds) against the relative residual error in the Frobenius norm of the final iterate for both [exp1] and [exp2]. Recall that we use the stopping condition in Algorithm 5 for the outer iteration, which has a lower computational cost and lower storage requirements. Here, we compute the final relative residual error in a separate post-processing step, simply for comparison. For these experiments, the relative residual error is observed to be up to three orders of magnitude larger than the value of ϵ used for the chosen stopping condition (see Algorithm 5). A discussion about using the relative residual error and the backwards error as stopping criteria for smaller-scale stochastic Galerkin matrix equations is given in [37].

Results obtained with the EnhancedAEM methods with PGD-update, PGD/GS, R-stage-p, and PGD-greedy are marked in red, green, blue, and magenta respectively, and each configuration of n_{update} and $k_{\max}$ is marked with a different symbol. It can be seen from the figures that

- the costs of R-stage-p and PGD/GS are less sensitive to n_{update} and $k_{\max}$ than those of PGD-update;
- the costs of PGD-greedy, which solves a reduced system at every outer iteration, tend to be larger then other methods;
- R-stage-p is more efficient for smaller values of n_{update} whereas PGD/GS and PGD-update are better with larger n_{update};
- for PGD-update and PGD/GS, relatively large $n_{\text{update}} > 10$ and $k_{\max} = 2$ results in better performances, and, for R-stage-p, relatively small $n_{\text{update}} \leq 10$ and $k_{\max} = 1$ results in better performances.

Table 3 reports the number of outer iterations p required to achieve the stopping tolerance ϵ for problems [exp1] and [exp2] when PGD-update, PGD/GS, and R-stage-p are used. The benefit of using R-stage-p becomes more pronounced as we seek highly accurate solutions with smaller ϵ. Our general observation is that among the four enhancement approaches, the R-stage-p method is less sensitive to the choice of algorithm parameter inputs, scales better for larger problem sizes, and is the most effective of the four approaches.

We now briefly consider a second benchmark problem whose solution matrix has different rank characteristics and for which low-rank solvers ought to perform well.



(a) $m = 20$, [exp1] **(b)** $m = 24$, [exp1]

(c) $m = 20$, [exp2] **(d)** $m = 24$, [exp2]

Fig. 4 Computational timings (in seconds) of four EnhancedAEM methods for varying $k_{\max}$ and n_{update}. Timings of each method with each parameter set-up are averaged over 5 testing runs

4.3 Benchmark problem 2: fast decay coefficients

We define the random field $a(x, \xi)$ as in (1.5) but now we choose $\xi_i \sim U(-1, 1)$ and the functions $a_i(x)$ have coefficients that decay more rapidly than in the first benchmark problem. The details of this problem can be found in [9]. Specifically, the coefficients of the expansion are

$$a_0 = 1, \qquad a_i(x) = \alpha_i \cos(2\pi \varrho_1(i) x_1) \cos(2\pi \varrho_2(i) x_2), \quad i = 1, 2, \ldots, m$$

Table 3 The number of outer iterations p required to achieve the stopping tolerance ϵ for solving the problems [exp1] and [exp2] when PGD-update, PGD/GS, and R-stage-p are used. The reported values of p are computed by averaging values of p obtained with the eight different combinations of n_{update} and $k_{\max}$ shown in the legend of Fig. 4

[exp1]	$m = 20$			$m = 24$		
	PGD-update	PGD/GS	R-stage-p	PGD-update	PGD/GS	R-stage-p
$\epsilon = 10^{-7}$	163.8	160.4	152.9	184.9	177.8	173.0
$\epsilon = 10^{-8}$	264.6	273.9	259.5	306.6	312.3	296.7
$\epsilon = 10^{-9}$	356.3	363.7	340.1	415.0	421.5	397.3
$\epsilon = 10^{-10}$	531.1	520.6	486.0	609.4	593.7	563.9
[exp2]	$m = 20$			$m = 24$		
	PGD-update	PGD/GS	R-stage-p	PGD-update	PGD/GS	R-stage-p
$\epsilon = 10^{-7}$	293.1	287.7	282.1	344.0	334.9	330.6
$\epsilon = 10^{-8}$	414.6	422.7	397.7	492.8	506.7	478.3
$\epsilon = 10^{-9}$	569.8	544.6	511.6	673.7	640.5	616.7
$\epsilon = 10^{-10}$	821.6	716.4	677.1	933.1	848.1	810.1

where $\alpha_i = \bar{\alpha} i^{-\sigma}$ with $\sigma > 1$ and $\bar{\alpha}$ satisfies $0 < \bar{\alpha} < 1/\zeta(\sigma)$, where ζ is the Riemann zeta function. Furthermore, $\varrho_1(i) = i - k(i)(k(i)+1)/2$ and $\varrho_2(i) = k(i) - \varrho_1(i)$ where $k(i) = \lfloor -1/2 + \sqrt{1/4 + 2i} \rfloor$. For computing the coefficients, we use the MATLAB software package S- IFISS [41]. In the following experiment, we choose $m = 20$, $\sigma = 4$ and $\bar{\alpha} = 0.832$. The parameter σ controls the rate of algebraic decay of the coefficients. The specific choice $\sigma = 4$ leads to fast decay and this causes the true solution matrix to have a lower rank than in the first benchmark problem.

We investigate computational timings of the EnhancedAEM methods with the same experimental settings used in Sect. 4.2.3. Here, we vary the stopping tolerance for the outer iterations as $\epsilon = \{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}\}$ and we choose the same values of n_{update} and $k_{\max}$ as before. Figure 5 reports elapsed time (in seconds) against relative residual error. In nearly all cases, our observations agree with the findings in Fig. 4. However, the impact of n_{update} is slightly less clear for these tests. The R-stage-p method is generally still less sensitive than the other two methods to the choices of n_{update} and $k_{\max}$, with one exception, indicated by the blue triangle marker, which is located to the far right in Fig. 5. With $n_{\text{update}} = 30$, $k_{\max} = 2$, and $\epsilon = 10^{-9}$ (giving the right-most blue triangle), the R-stage-p method does not meet the stopping criterion until $p \approx 125$, which is larger than the value $p \approx 90$ needed for the other choices of algorithm inputs. We attribute this to the large number of steps (30) between enhancements; in this case, the method fell just short of the stopping criterion after 90 steps. Finally, we report the number of outer iterations p required to achieve the stopping tolerance ϵ in Table 4. As the true solution matrix has an intrinsic low-rank structure, the reported values of p are much smaller than those shown in Table 3.

Remark 4.1 We found the solvers to be largely insensitive to the choice of the parameters $k_{\max}$ and n_{update}; a similar observation was made in [30].

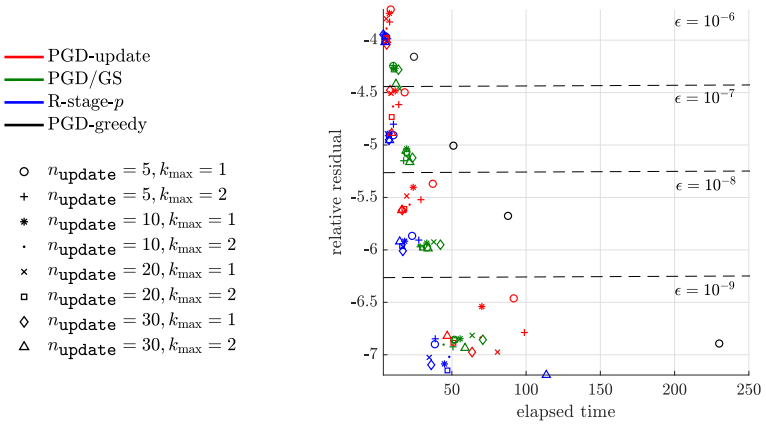


Fig. 5 Computational timings (in seconds) of four EnhancedAEM methods for varying $k_{\max}$ and n_{update}. Timings of each method with each parameter set-up are averaged over 5 testing runs

Table 4 The number of outer iterations p required to achieve the stopping tolerance ϵ for solving the second benchmark problem when PGD-update, PGD/GS, R-stage-p, and PGD-greedy are used. The reported values of p are computed by averaging values of p obtained with the eight different combinations of n_{update} and $k_{\max}$ shown in the legend of Fig. 5

	PGD-update	PGD/GS	R-stage-p	PGD-greedy
$\epsilon = 10^{-6}$	43.7	49.0	30.1	44.8
$\epsilon = 10^{-7}$	58.3	68.3	41.4	73.0
$\epsilon = 10^{-8}$	81.7	91.7	61.3	102.8
$\epsilon = 10^{-9}$	130.9	121.6	91.6	162.8

4.4 Further extensions

We tested all the AEM methods on matrix equations obtained from SGFEM discretizations of stochastic convection-diffusion problems [26, Section 5.2], where the randomness is in the diffusion coefficient as in Sect. 4.2. Although the energy norm cannot be defined for this problem as it has a non-symmetric operator, the same projection framework described herein can be applied to compute approximate solutions. Experiments (not reported here) were conducted similar to the ones in Sects. 4.2.1–4.2.2. We observed that the R-stage-p method produces qualitatively better approximate factors V_p and W_p, as measured in the error metrics used in Sects. 4.2.1–4.2.2, than the S-rank-1 AEM method and the other two EnhancedAEM methods. We also note that for problems with a non-symmetric operator, rank-one update ALS algorithms such as the ones in [20,23] can be used.

5 Conclusions

In this study, we have investigated several variants of alternating minimization methods to compute low-rank solutions of matrix equations that arise from SGFEM discretizations of parameterized elliptic PDEs. Using a general formulation of alternating minimization of errors in the energy norm, derived from the well-known general projection method, our starting point was a variant of the stagewise ALS method, a technique for building rank-p approximate solutions developed for matrix completion and matrix sensing. Our main contribution consists of a combination of this approach with so-called enhancement procedures of the type used for PGD methods [30,31] in which rank-one approximate solutions are enhanced by adaptive use of higher-rank quantities that improve solution quality but limit costs by adaptively restricting the rank of updates. Experimental results demonstrate that the proposed PGD/GS and R-stage-p methods produce accurate low-rank approximate solutions built from good approximations of the singular vectors of the matricized parameter-dependent solutions. Moreover, the results show that the R-stage-p method scales better for larger problems, is less sensitive to algorithm inputs, and produces approximate solutions in the fastest times.

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

References

1. Babuška, I., Tempone, R., Zouraris, G.E.: Galerkin finite element approximations of stochastic elliptic partial differential equations. SIAM J. Numer. Anal. **42**(2), 800–825 (2004)
2. Ballani, J., Grasedyck, L.: A projection method to solve linear systems in tensor format. Numer. Linear Algebra Appl. **20**(1), 27–43 (2013)
3. Benner, P., Breiten, T.: Low rank methods for a class of generalized Lyapunov equations and related issues. Numerische Mathematik **124**(3), 441–470 (2013)
4. Corveleyn, S., Rosseel, E., Vandewalle, S.: Iterative solvers for a spectral Galerkin approach to elliptic partial differential equations with fuzzy coefficients. SIAM J. Sci. Comput. **35**(5), S420–S444 (2013)
5. Dolgov, S.V., Savostyanov, D.V.: Alternating minimal energy methods for linear systems in higher dimensions. SIAM J. Sci. Comput. **36**(5), A2248–A2271 (2014)
6. Doostan, A., Iaccarino, G.: A least-squares approximation of partial differential equations with high-dimensional random inputs. J. Comput. Phys. **228**(12), 4332–4345 (2009)
7. Doostan, A., Validi, A., Iaccarino, G.: Non-intrusive low-rank separated approximation of high-dimensional stochastic models. Comput. Methods Appl. Mech. Eng. **263**, 42–55 (2013)
8. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika **1**(3), 211–218 (1936)
9. Eigel, M., Pfeffer, M., Schneider, R.: Adaptive stochastic Galerkin FEM with hierarchical tensor representations. Numerische Mathematik **136**(3), 765–803 (2017)
10. Elman, H.C., Furnival, D.: Solving the stochastic steady-state diffusion problem using multigrid. IMA J. Numer. Anal. (2007)
11. Elman, H.C., Ramage, A., Silvester, D.J.: IFISS: a computational laboratory for investigating incompressible flow problems. SIAM Rev. **56**(2), 261–273 (2014)
12. Elman, H.C., Su, T.: A low-rank multigrid method for the stochastic steady-state diffusion problem. SIAM J. Matrix Anal. Appl. **39**(1), 492–509 (2018)
13. Ghanem, R.G., Spanos, P.D.: Stochastic finite elements: a spectral approach. Dover Publications, New York (2003)

14. Golub, G.H., Van Loan, C.F.: Matrix computations. JHU Press, Maryland (2012)
15. Haldar, J.P., Hernando, D.: Rank-constrained solutions to linear matrix equations using powerfactorization. IEEE Signal Process. Lett. **16**(7), 584–587 (2009)
16. Hardt, M.: Understanding alternating minimization for matrix completion. In: Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on, pp. 651–660. IEEE (2014)
17. Jain, P., Netrapalli, P., Sanghavi, S.: Low-rank matrix completion using alternating minimization. In: Proceedings of the forty-fifth annual ACM Symposium on Theory of Computing, pp. 665–674. ACM (2013)
18. Khoromskij, B.N., Schwab, C.: Tensor-structured galerkin approximation of parametric and stochastic elliptic pdes. SIAM J. Sci. Comput. **33**(1), 364–385 (2011)
19. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Comput. **8**, 30–37 (2009)
20. Kressner, D., Sirković, P.: Truncated low-rank methods for solving general linear matrix equations. Numer. Linear Algebra Appl. **22**(3), 564–583 (2015)
21. Kressner, D., Tobler, C.: Krylov subspace methods for linear systems with tensor product structure. SIAM J. Matrix Anal. Appl. **31**(4), 1688–1714 (2010)
22. Kressner, D., Tobler, C.: Low-rank tensor Krylov subspace methods for parametrized linear systems. SIAM J. Matrix Anal. Appl. **32**(4), 1288–1316 (2011)
23. Kürschner, P., Dolgov, S., Harris, K.D., Benner, P.: Greedy low-rank algorithm for spatial connectome regression. J. Math. Neurosci. **9**(1), 1–22 (2019)
24. Le Maître, O.P., Knio, O.M., Debusschere, B.J., Najm, H.N., Ghanem, R.G.: A multigrid solver for two-dimensional stochastic diffusion equations. Comput. Methods Appl. Mech. Eng. **192**(41), 4723–4744 (2003)
25. Lee, K., Carlberg, K., Elman, H.C.: Stochastic least-squares Petrov-Galerkin method for parameterized linear systems. SIAM/ASA J. Uncertain. Quantif. **6**(1), 374–396 (2018)
26. Lee, K., Elman, H.C.: A preconditioned low-rank projection method with a rank-reduction scheme for stochastic partial differential equations. SIAM J. Sci. Comput. **39**(5), S828–S850 (2017)
27. Loève, M.: Probability theory, Vol. II, vol. 46. Springer (1978)
28. Lord, G.J., Powell, C.E., Shardlow, T.: An introduction to computational stochastic PDEs. Cambridge University Press, Cambridge (2014)
29. Matthies, H.G., Zander, E.: Solving stochastic systems with low-rank tensor compression. Linear Algebra Appl. **436**(10), 3819–3838 (2012)
30. Nouy, A.: A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. Comput. Methods Appl. Mech. Eng. **196**(45), 4521–4537 (2007)
31. Nouy, A.: Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems. Arch. Comput. Methods Eng. **17**(4), 403–434 (2010)
32. Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
33. Palitta, D., Kürschner, P.: On the convergence of Krylov methods with low-rank truncations. Numerical Algorithms pp. 1–35 (2021)
34. Pellissetti, M.F., Ghanem, R.G.: Iterative solution of systems of linear equations arising in the context of stochastic finite elements. Adv. Eng. Softw. **31**(8), 607–616 (2000)
35. Powell, C.E., Elman, H.C.: Block-diagonal preconditioning for spectral stochastic finite-element systems. IMA J. Numer. Anal. **29**, 350–375 (2009)
36. Powell, C.E., Silvester, D.J., Simoncini, V.: An efficient reduced basis solver for stochastic Galerkin matrix equations. SIAM J. Sci. Comput. **39**(1), A141–A163 (2017)
37. Pranesh, S.: Backward error and condition number of a generalized Sylvester equation, with application to the stochastic Galerkin method. Linear Algebra Appl. **594**, 95–116 (2020)
38. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Review **52**(3), 471–501 (2010)
39. Saad, Y.: Iterative methods for sparse linear systems. SIAM, New Delhi (2003)
40. Schwab, C., Gittelson, C.J.: Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs. Acta Numerica **20**, 291–467 (2011)
41. Silvester, D.J., Bespalov, A., Powell, C.E.: S-IFISS, available online at http://www.manchester.ac.uk/ifiss/s-ifiss1.0.tar.gz
42. Silvester, D.J., Elman, H.C., Ramage, A.: Incompressible flow and iterative solver software (IFISS) version 3.5 (2016). http://www.manchester.ac.uk/ifiss/
43. Simoncini, V.: Computational methods for linear matrix equations. SIAM Rev. **58**(3), 377–441 (2016)

44. Sterck, H.D., Miller, K.: An adaptive algebraic multigrid algorithm for low-rank canonical tensor decomposition. SIAM J. Sci. Comput. **35**(1), B1–B24 (2013)
45. Stoll, M., Breiten, T.: A low-rank in time approach to PDE-constrained optimization. SIAM J. Sci. Comput. **37**(1), B1–B29 (2015)
46. Tamellini, L., Le Maître, O.P., Nouy, A.: Model reduction based on proper generalized decomposition for the stochastic steady incompressible Navier-Stokes equations. SIAM J. Sci. Comput. **36**(3), A1089–A1117 (2014)
47. Xiu, D.: Numerical methods for stochastic computations: a spectral method approach. University Press, Princeton (2010)
48. Xiu, D., Karniadakis, G.E.: The Wiener-Askey polynomial chaos for stochastic differential equations. SIAM J. Sci. Comput. **24**(2), 619–644 (2002)
49. Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-scale parallel collaborative filtering for the Netflix prize. In: International conference on algorithmic applications in management, pp. 337–348. Springer (2008)