C. R. Acad. Sci. Paris, Ser. I 357 (2019) 561–570

Contents lists available at ScienceDirect

C. R. Acad. Sci. Paris, Ser. I

www.sciencedirect.com

Numerical analysis

Sparse approximate solutions to stochastic Galerkin equations

Approximations creuses pour la méthode de Galerkin stochastique

Christophe Audouze, Prasanth B. Nair

University of Toronto, Institute for Aerospace Studies, 4925 Dufferin Street, Ontario, M3H 5T6, Canada

ARTICLE INFO

Article history:
Received 26 June 2018
Accepted after revision 6 June 2019
Available online 2 July 2019

Presented by the Editorial Board

ABSTRACT

In this Note, we formulate a sparse Krylov-based algorithm for solving large-scale linear systems of algebraic equations arising from the discretization of randomly parametrized (or stochastic) elliptic partial differential equations (SPDEs). We analyze the proposed sparse conjugate gradient (CG) algorithm within the framework of inexact Krylov subspace methods, prove its convergence and study its abstract computational cost. Numerical studies conducted on stochastic diffusion models show that the proposed sparse CG algorithm outperforms the classical CG method when the sought solutions admit a sparse representation in a polynomial chaos basis. In such cases, the sparse CG algorithm recovers almost exactly the sparsity pattern of the exact solutions, which enables accelerated convergence. In the case when the SPDE solution does not admit a sparse representation, the convergence of the proposed algorithm is very similar to the classical CG method.

RÉSUMÉ

Dans cette Note, nous formulons une méthode de Krylov creuse pour la résolution de grands systèmes linéaires issus de la discrétisation d'équations aux dérivées partielles elliptiques paramétrées aléatoirement. Nous analysons l'algorithme du gradient conjugué (GC) creux dans le cadre des méthodes de sous-espaces de Krylov inexactes, montrons sa convergence et étudions sa complexité algorithmique. Les études numériques réalisées sur des modèles de diffusion stochastiques montrent que la méthode du GC creux converge plus rapidement que le GC classique lorsque la solution recherchée admet une représentation creuse dans une base de chaos polynomial. Dans ce cas, l'algorithme du GC creux retrouve presque intégralement la structure creuse de la solution exacte, permettant d'accélérer la convergence. Lorsque la solution exacte est dense, l'algorithme du GC creux fournit des résultats de convergence similaires à ceux obtenus avec le GC classique.

E-mail addresses: c.audouze@utoronto.ca (C. Audouze), pbn@utias.utoronto.ca (P.B. Nair).

https://doi.org/10.1016/j.crma.2019.05.009
1631-073X/

1. Introduction

In computational engineering sciences, different sources of randomness (which typically appear in the physical parameters, boundary conditions and/or initial conditions of a partial differential equation) are routinely taken into account for uncertainty quantification purposes, for example for applications in fluid dynamics or heat transfer. The sources of randomness are commonly modeled using a finite number of random variables, leading to randomly parameterized (or stochastic) partial differential equations (SPDEs). In the present work, we focus on the numerical solution to large-scale linear algebraic systems of equations arising from spatial and stochastic discretization of SPDEs. As a typical example, consider the following stochastic steady-state diffusion equation with homogeneous Dirichlet boundary conditions

$$\begin{aligned} -\nabla\cdot\big(\kappa(\mathbf{x};\boldsymbol{\xi})\nabla u(\mathbf{x};\boldsymbol{\xi})\big) &= f(\mathbf{x};\boldsymbol{\xi}) \quad \text{a.s. in } \mathcal{D}\times\Gamma,\\ u(\mathbf{x};\boldsymbol{\xi}) &= 0 \qquad\quad \text{a.s. on } \partial\mathcal{D}\times\Gamma, \end{aligned} \tag{1}$$

where $\mathcal{D}$ is an open bounded domain of $\mathbb{R}^d$ $(d=2,3)$ with boundary $\partial\mathcal{D}$, and $\mathbf{x}\in\mathcal{D}$ is the spatial variable. We denote the probability space by the triplet $(\Omega,\mathcal{F},\mathcal{P})$, where $\Omega\subset\mathbb{R}^q$ is the sample space, $\mathcal{F}$ is the σ-algebra associated with Ω and $\mathcal{P}:\mathcal{F}\to[0,1]$ is a probability measure. The components of the vector $\boldsymbol{\xi}=(\xi_1(\omega),\dots,\xi_M(\omega))^\intercal\in\mathbb{R}^M$, where $\omega\in\Omega$, are assumed to be i.i.d. random variables. The joint probability density function (pdf) of $\boldsymbol{\xi}$ is denoted by $\rho(\boldsymbol{\xi})$ and can be written as the product of its marginal densities, that is $\rho(\boldsymbol{\xi})=\prod_{j=1}^M\rho_j(\xi_j)$. We denote by $\Gamma=\Gamma_1\times\cdots\times\Gamma_M$ the joint image of $\boldsymbol{\xi}$. The diffusivity field is classically discretized using a truncated Karhunen–Loève (KL) expansion scheme [17] as $\kappa(\mathbf{x};\boldsymbol{\xi})=\kappa_0(\mathbf{x})+\sum_{m=1}^M\xi_m\kappa_m(\mathbf{x})$, where κ_0 is the mean diffusivity and $(\kappa_m)_{m\geq1}$ forms an orthogonal basis of $L^2(\mathcal{D})$. The functions κ_m are defined as $\kappa_m=\sigma\sqrt{\lambda_m}\psi_m$, where σ denotes the standard deviation of κ and $\{\lambda_m,\psi_m\}$ are the eigenpairs of the two-point correlation function used for modeling the diffusivity field. Using Doob–Dynkin's lemma [20], it can be shown that the SPDE solution can be described with the same set of random variables used for representing κ, that is, $u=u(\mathbf{x};\xi_1(\omega),\dots,\xi_M(\omega))$. For more details about stochastic diffusion models, see for example [2,21].

In recent years, various numerical schemes have been developed for solving SPDEs, including polynomial chaos (PC) stochastic Galerkin projection schemes [13,32] and stochastic collocation schemes based on sparse tensor product quadrature rules [31]. In the present work, we specifically focus on stochastic Galerkin methods where approximate solutions are sought in finite-dimensional tensor product subspaces $\mathbb{P}_{\mathbf{x},h}\otimes\mathbb{P}_{\boldsymbol{\xi},p_\xi}$, where

$$\mathbb{P}_{\mathbf{x},h}=\operatorname{span}\{\phi_i(\mathbf{x})\}_{i=1,\dots,N_\mathbf{x}}\subset H_0^1(\mathcal{D}),\ \ \mathbb{P}_{\boldsymbol{\xi},p_\xi}=\operatorname{span}\{\psi_i(\boldsymbol{\xi})\}_{i=1,\dots,N_\boldsymbol{\xi}}\subset L^2(\Gamma). \tag{2}$$

Typically ϕ_i denote piecewise linear continuous finite element (FE) basis functions defined on a triangulation of $\mathcal{D}$ and vanishing on $\partial\mathcal{D}$. Using the multi-index notation $\boldsymbol{\alpha}=(\alpha_1,\dots,\alpha_M)\in\mathbb{N}^M$, PC basis functions are given by $\psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})=\prod_{i=1}^M\psi_{\alpha_i}(\xi_i)$. The set of multivariate PC basis functions of total degree p_ξ is then defined as $\mathbb{P}_{\boldsymbol{\xi},p_\xi}=\operatorname{span}\{\psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}),\ |\boldsymbol{\alpha}|=\alpha_1+\cdots+\alpha_M\leq p_\xi\}$, which is of dimension $N_\boldsymbol{\xi}=\frac{(M+p_\xi)!}{M!p_\xi!}$. For notational convenience, PC basis functions are often reordered using a one-to-one mapping as $\psi_i(\boldsymbol{\xi})$, $i=1,\dots,N_\boldsymbol{\xi}$. The application of stochastic Galerkin projection schemes (also referred to as spectral FE methods) leads to deterministic $N_\mathbf{x}N_\boldsymbol{\xi}\times N_\mathbf{x}N_\boldsymbol{\xi}$ linear algebraic equations of the form

$$\mathbf{A}\mathbf{u}=\mathbf{b}, \tag{3}$$

where $\mathbf{A}$ has the characteristic Kronecker product structure

$$\mathbf{A}=\mathbf{G}_0\otimes\mathbf{A}_0+\sum_{m=1}^M\mathbf{G}_m\otimes\mathbf{A}_m, \tag{4}$$

with $\mathbf{G}_m\in\mathbb{R}^{N_\boldsymbol{\xi}\times N_\boldsymbol{\xi}}$ and $\mathbf{A}_m\in\mathbb{R}^{N_\mathbf{x}\times N_\mathbf{x}}$. In the case of the stochastic diffusion models described above, the matrix entries are given by $(\mathbf{G}_0)_{ij}=\langle\psi_i\psi_j\rangle$, $(\mathbf{G}_m)_{ij}=\langle\xi_m\psi_i\psi_j\rangle$, $(\mathbf{A}_0)_{ij}=\int_\mathcal{D}\kappa_0(\mathbf{x})\nabla\phi_i(\mathbf{x})\nabla\phi_j(\mathbf{x})\,\mathrm{d}\mathbf{x}$ and $(\mathbf{A}_m)_{ij}=\int_\mathcal{D}\kappa_m(\mathbf{x})\nabla\phi_i(\mathbf{x})\nabla\phi_j(\mathbf{x})\,\mathrm{d}\mathbf{x}$, where the notation $\langle\cdot\rangle$ stands for the expectation operator with respect to ρ, i.e. $\langle\cdot\rangle=\int_\Gamma\cdot\,\rho(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi}$.

From a theoretical point of view, there has been a growing interest in the analysis of PC-based Galerkin projection schemes with the derivation of *a priori* rates of convergence; see, for example, [2,3,8,29]. Randomly parametrized linear algebraic equations (corresponding to the spatial semi-discretization of SPDEs) have been analyzed and studied numerically in [1,7,9,10]. In realistic engineering applications (for example, 3D stochastic diffusion models where vast numbers of random variables are needed to accurately model the diffusivity field), the expanded matrix equations (3) become extremely large, since both $N_\mathbf{x}$ and $N_\boldsymbol{\xi}$ are large ($N_\boldsymbol{\xi}$ increases exponentially with M and p_ξ). This has motivated the development of efficient numerical techniques (e.g., [18,19,23]) and efficient preconditioning strategies (e.g., [21,22,28,30]).

In the present work, we focus on sparse approximations of large-scale stochastic Galerkin matrix equations, that is, we seek a sparse approximate solution $\widehat{\mathbf{u}}$ such that

$$\left\|\left(\mathbf{G}_0\otimes\mathbf{A}_0+\sum_{m=1}^M\mathbf{G}_m\otimes\mathbf{A}_m\right)\widehat{\mathbf{u}}-\mathbf{b}\right\|_2\leq\varepsilon, \tag{5}$$

where ε is a given threshold. Sparse approximation algorithms based on ℓ_1 or ℓ_2 regularizations are widely used in machine learning and compressive sensing while solving multivariate regression problems with noisy datasets, the sparsity being used to filter the noise in the training data [6,11,12,16]. The idea of sparse approximations has also been used in various other contexts to alleviate the curse of dimensionality, for example, sparse quadrature rules [31] based on Smolyak's construction, sparse approximations for transport-dominated diffusion problems [24], and stochastic Galerkin approximation based on sparse wavelet bases for the Boltzmann equation with uncertainties [25].

In the context of stochastic Galerkin equations, our goal is to formulate a sparse version of the classical Krylov methods, namely the conjugate gradient (CG) algorithm. Since the expanded matrix equations (3) are deterministic, the sparsity feature is not used to filter the noise, but to improve computational efficiency and reduce memory requirements. Using sparse Krylov-based algorithms to solve stochastic Galerkin matrix equations also allows to exploit the sparsity feature that is inherent to some SPDE solutions and accelerate the convergence speed. It is worth noting that PC approximations of some stochastic diffusion models can be shown to be sparse in nature [5]. We carry out a convergence analysis of the proposed sparse CG algorithm by interpreting it as an inexact Krylov method. Numerical studies are presented for a set of test problems to illustrate that the proposed algorithm can recover almost exactly the sparsity pattern of the exact solution, while incurring computational expense that is lower than with the classical CG algorithm. In cases where the exact solution does not admit a sparse representation in a PC basis, the performance of the proposed algorithm coincides with the classical CG algorithm.

2. Sparse CG algorithm

We propose a sparse CG algorithm for solving (3)–(4) in the case when $\mathbf{A} \in \mathbb{R}^{N_\mathbf{x}N_\xi \times N_\mathbf{x}N_\xi}$ is a symmetric positive-definite (SPD) matrix. The main idea is to start with a sparse initial guess $\mathbf{u}^0 \in \mathbb{R}^{N_\mathbf{x}N_\xi}$ and generate iterates of the form $\mathbf{u}^{k+1} = \mathbf{u}^k + \alpha_k \mathbf{p}^k$, where $\mathbf{p}^k \in \mathbb{R}^{N_\mathbf{x}N_\xi}$ are (very) sparse search directions and α_k denotes the step length. Typically, a random initialization of the $N_\mathbf{x}$ first components is used, which corresponds to considering a deterministic approximation to the SPDE solution. Since $\mathbf{A}$ is SPD, solving (3) is equivalent to minimizing the quadratic objective function $F(\mathbf{u}) = \frac{1}{2}(\mathbf{Au}, \mathbf{u})_2 - (\mathbf{u}, \mathbf{b})_2$, which justifies the natural choice of the step length:

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}} F(\mathbf{u}^k + \alpha \mathbf{p}^k) = \frac{(\mathbf{p}^k, \mathbf{r}^k)_2}{(\mathbf{Ap}^k, \mathbf{p}^k)_2}, \tag{6}$$

where $\mathbf{r}^k = \mathbf{b} - \mathbf{Au}^k$. At the k-th iteration, the search direction is defined as $\mathbf{p}^k = \mathbf{m}^k \circ \mathbf{z}^k$, where the preconditioned residual $\mathbf{z}^k \in \mathbb{R}^{N_\mathbf{x}N_\xi}$ is known to be a descent direction, $\mathbf{m}^k \in \mathbb{N}^{N_\mathbf{x}N_\xi}$ is a sparse mask (i.e. a vector that only contains zeros and ones), and $\circ$ denotes the Hadamard (or element-wise) product between vectors. The initial sparse mask is defined with ones for its first $N_\mathbf{x}$ components. It is worth mentioning that the $\mathbf{A}$-conjugacy property for the search directions as well as the residual orthogonality with respect to the Krylov subspace do not hold anymore since a sparse mask is applied to the search direction at each iteration. Consequently, it is not possible to use classical CG updates of the form $\mathbf{p}^{k+1} \leftarrow \mathbf{z}^{k+1} + \beta_k \mathbf{p}_k$. Hence, a modified Gram–Schmidt (MGS) procedure is used to enforce the conjugacy of $\mathbf{p}^{k+1}$ with respect to the preceding l search directions $(\mathbf{p}^j)_{j=k-l,\dots,k}$, where l is kept small for memory and computational savings.

A key ingredient of the sparse CG algorithm is the strategy used to define new search directions. The simplest way to determine a new index (or a set of indices) in the sparse mask is to find the largest component(s) in the current residual, among the indices that have not yet been selected. However, numerical studies suggest that this criterion is problem-dependent – the number of new entries to add in the sparse mask that ensures fast convergence can vary significantly, depending on the test case; also, setting the "right" number of entries beforehand is not straightforward in practice. For more flexibility, we choose new entries in the sparse mask by selecting among a set of indices the residual components that are larger than a given threshold, where the threshold parameter is dynamically updated across iterations. More precisely, if $\mathcal{J}^k$ denotes a set of indices that have not yet been selected, the new entries are defined as $\mathcal{I}_{k+1} = \{i \in \mathcal{J}^k,\ |\mathbf{r}_i^{k+1}| > \theta\}$. The threshold θ is initialized with a large value and then automatically decreased when no (or not enough) additional entries can be selected, that is, if $|\mathcal{I}_{k+1}| < \varepsilon_{lv}$. Since the classical CG stopping criterion $||\mathbf{r}^k||_2 > \varepsilon ||\mathbf{r}^0||_2$ is ensured if

$$||\mathbf{r}^k||_2 = \left(\sum_{i=1}^{N_\mathbf{x}N_\xi} (\mathbf{r}_i^k)^2 \right)^{1/2} \geq \sqrt{N_\mathbf{x}N_\xi} \min_i |\mathbf{r}_i^k| > \varepsilon ||\mathbf{r}^0||_2, \tag{7}$$

it naturally provides the lower bound $\theta_{cg} = \varepsilon ||\mathbf{r}^0||_2 / \sqrt{N_\mathbf{x}N_\xi} \leq \theta$. The main steps of the sparse CG algorithm (using a preconditioner $\mathbf{M}$) are outlined below.

Preconditioned sparse CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{M}$, $N_\mathbf{x}$, N_ξ, ε, θ_{init}, l)

Define $\mathbf{u}^0 = (u_1, u_2 \dots, u_{N_\mathbf{x}}, 0, 0, \dots, 0)^\mathsf{T}$ and $\mathbf{r}^0 = \mathbf{b} - \mathbf{Au}^0$.
$\mathbf{m}^0 \leftarrow (1, 1, \dots, 1, 0, 0, \dots, 0)^\mathsf{T}$
$\mathcal{J}^0 \leftarrow \{N_\mathbf{x} + 1, \dots, N_\mathbf{x}N_\xi\}$

Solve $\mathbf{M}\mathbf{z}^0 = \mathbf{r}^0$
$\mathbf{p}^0 = \mathbf{m}^0 \circ \mathbf{z}^0$
$\theta \leftarrow \theta_{\text{init}}$
$\theta_{cg} = \varepsilon ||\mathbf{r}^0||_2 / \sqrt{N_{\mathbf{x}} N_{\boldsymbol{\xi}}}$
$k \leftarrow 0$
while $\left(||\mathbf{r}^k||_2 / ||\mathbf{r}^0||_2 > \varepsilon\right)$ **do**
  $\alpha_k = \frac{(\mathbf{p}^k, \mathbf{r}^k)_2}{(\mathbf{A}\mathbf{p}^k, \mathbf{p}^k)_2}$
  $\mathbf{u}^{k+1} \leftarrow \mathbf{u}^k + \alpha_k \mathbf{p}^k$
  $\mathbf{r}^{k+1} \leftarrow \mathbf{r}^k - \alpha_k \mathbf{A}\mathbf{p}^k$
  Solve $\mathbf{M}\mathbf{z}^{k+1} = \mathbf{r}^{k+1}$
  Select $\mathcal{I}_{k+1} = \{i \in \mathcal{J}^k,\ |\mathbf{r}_i^{k+1}| > \theta\}$, $|\mathcal{I}_{k+1}| = m_k$
  $\mathbf{m}^{k+1} \leftarrow \mathbf{m}^k + \sum_{j \in \mathcal{I}_{k+1}} \mathbf{e}_j$
  $\mathcal{J}^{k+1} \leftarrow \mathcal{J}^k \setminus \mathcal{I}_{k+1}$
  if $|\mathcal{I}_{k+1}| < \varepsilon_{lv}$ **then**
    $\theta \leftarrow \theta / 10$
    if $\theta < \theta_{cg}$ **then**
      $\theta \leftarrow \theta_{cg}$
    end if
  end if
  $\mathbf{p}^{k+1} \leftarrow \mathbf{m}^{k+1} \circ \mathbf{z}^{k+1}$
  $\mathbf{p}^{k+1} \leftarrow \text{MGS}(\mathbf{p}^{k+1}, (\mathbf{p}^j)_{j=k-l,\dots,k})$
  $k \leftarrow k + 1$
end while

3. Computational aspects and analysis

3.1. Computational cost and storage

We compare the classical and sparse CG algorithms in terms of abstract computational cost and memory requirements. For simplicity, we consider non-preconditioned versions of both algorithms, where the computational cost at each iteration is dominated by matrix–vector products. Since $\mathbf{A}$ has a Kronecker product structure (see (4)), the matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{p}^k$ can be efficiently computed as

$$\mathbf{y} = \left(\sum_{m=0}^{M} \mathbf{G}_m \otimes \mathbf{A}_m\right)\mathbf{p}^k = \text{vec}\left(\sum_{m=0}^{M} \mathbf{A}_m \mathbf{P}^k \mathbf{G}_m^{\mathsf{T}}\right), \tag{8}$$

where $\mathbf{P}^k = [\mathbf{p}_1^k, \dots, \mathbf{p}_{N_{\boldsymbol{\xi}}}^k] \in \mathbb{R}^{N_{\mathbf{x}} \times N_{\boldsymbol{\xi}}}$ contains the $N_{\boldsymbol{\xi}}$-dimensional PC component vectors of $\mathbf{p}^k$, and vec is the columnwise vectorization of matrices. In classical CG algorithms, the cost per iteration is essentially $\mathcal{O}\big(\sum_{m=0}^{M}\big(nnz(\mathbf{A}_m)N_{\boldsymbol{\xi}} + nnz(\mathbf{G}_m)N_{\mathbf{x}}\big)\big)$. Assuming that $\mathbf{A}_m$ and $\mathbf{G}_m$ have the same sparsity profile, that is, $nnz(\mathbf{A}_m) = \gamma_1 N_{\mathbf{x}}^2$ and $nnz(\mathbf{G}_m) = \gamma_2 N_{\boldsymbol{\xi}}^2$ with $0 < \gamma_1, \gamma_2 \ll 1$, the cost per iteration of the classical CG method scales as

$$\mathcal{O}\big((M+1)\big(\gamma_1 N_{\mathbf{x}}^2 N_{\boldsymbol{\xi}} + \gamma_2 N_{\boldsymbol{\xi}}^2 N_{\mathbf{x}}\big)\big). \tag{9}$$

In sparse CG algorithms, the cost at iteration k is essentially related to three matrix-vector products, $\mathbf{A}\mathbf{p}^k$ plus two extra products in the MGS step. In contrast to classical CG algorithms, the sparse direction $\mathbf{p}^k$ is a (very) sparse vector that has $N_{\mathbf{x}} + S_k$ non-zero entries with $S_k = m_1 + m_2 + \cdots + m_k$ for $k \geq 1$ and $S_0 = 0$. Since $\mathbf{p}^k$ has $N_{\mathbf{x}} + S_k$ non-zeros (with $N_{\mathbf{x}}$ first non-zeros), the number of non-zero columns in $\mathbf{P}^k$ is equal to $C_k \leq \min(1 + S_k, N_{\boldsymbol{\xi}})$. For k large enough, the sparse mask may be entirely filled, in which case sparse CG becomes classical CG with $C_k = N_{\boldsymbol{\xi}}$. When k is relatively small, it is expected that $\mathbf{P}^k$ is sparse, in which case the cost of $\mathbf{A}_m\mathbf{P}^k$ is $\mathcal{O}(nnz(\mathbf{A}_m)C_k)$ with $C_k \ll N_{\boldsymbol{\xi}}$. Since $\mathbf{A}_m$ is sparse and $\mathbf{P}^k$ is very sparse, the matrix $\mathbf{A}_m\mathbf{P}^k$ will also be very sparse. Denoting the number of non-zero rows in $\mathbf{A}_m\mathbf{P}^k$ by R_k, the cost of the matrix product $(\mathbf{A}_m\mathbf{P}^k)\mathbf{G}_m^{\mathsf{T}}$ is $\mathcal{O}(nnz(\mathbf{G}_m)R_k)$ with $R_k \ll N_{\mathbf{x}}$. As a result, the cost of the sparse CG algorithm at a (relatively small) iteration k is

$$\mathcal{O}\big(3(M+1)\big(\gamma_1 N_{\mathbf{x}}^2 C_k + \gamma_2 N_{\boldsymbol{\xi}}^2 R_k\big)\big). \tag{10}$$

The additional inner products used in the MGS step are not taken into account as their cost is significantly lower than matrix–vector products. Depending on the level of sparsity of the matrices $\mathbf{A}_m$ and $\mathbf{G}_m$, the ratio of costs at the k-th sparse CG iteration scales as follows in Table 1.

Table 1
Ratio of costs (9) and (10) at the k-th sparse CG iteration for small values of k.

	$nnz(\mathbf{G}_m) \ll nnz(\mathbf{A}_m)$	$nnz(\mathbf{A}_m) \ll nnz(\mathbf{G}_m)$	$nnz(\mathbf{G}_m) \sim nnz(\mathbf{A}_m)$
Ratio of costs	$\frac{1}{3}\frac{N_\xi}{C_k} \gg 1$	$\frac{1}{3}\frac{N_\mathbf{x}}{R_k} \gg 1$	$\frac{1}{3}\frac{N_\mathbf{x}+N_\xi}{R_k+C_k} \gg 1$

Table 2
Storage requirements at each iteration of classical/sparse CG algorithms.

	classical CG	sparse CG
# of vectors	5 dense	$(l+1)$ dense and $(l+2)$ sparse
# of non-zero entries	$5N_\mathbf{x}N_\xi$	$(l+1)N_\mathbf{x}N_\xi + (l+2)N_\mathbf{x} + 3S_k + S_{k-1} + \cdots + S_{k-l}$

We next discuss the storage requirements of both algorithms. At each iteration, the classical CG algorithm requires five dense vectors ($\mathbf{u}^k$, $\mathbf{p}^k$, $\mathbf{A}\mathbf{p}^k$, $\mathbf{r}^k$, $\mathbf{r}^{k-1}$) to be saved, meaning that $5N_\mathbf{x}N_\xi$ entries need to be stored. In the sparse CG algorithm, we have to save $2l+3$ vectors, of which $l+1$ dense vectors ($\mathbf{r}^k$ and $(\mathbf{A}\mathbf{p}^j)_{j=k-l,\ldots,k}$) and $l+2$ sparse vectors ($\mathbf{u}^k$, $\mathbf{m}^k$ and $(\mathbf{p}^j)_{j=k-l,\ldots,k}$). The storage requirements are summarized in Table 2.

In practice, the parameter l is kept small. In our numerical studies, $l=3$, meaning that 4 dense and 5 sparse vectors are stored at each sparse CG iteration, instead of 5 dense vectors in classical CG algorithms, which roughly represents 20% memory saving.

3.2. Convergence analysis

It is of interest to note that the sparse CG algorithm is convergent. If $\overline{m}$ denotes the average number of new entries added in the sparse mask at each iteration, then $\lfloor \frac{N_\mathbf{x}N_\xi - N_\mathbf{x}}{\overline{m}} \rfloor$ iterations are needed to fill up entirely the sparse mask (assuming the $N_\mathbf{x}$ first entries of the initial sparse mask are non-zeros). Once the sparse mask is fully dense, the sparse CG algorithm becomes a classical CG method, which is known to converge in at most $N_\mathbf{x}N_\xi$ iterations. As a result, the sparse CG algorithm converges in at most $\lfloor \frac{N_\mathbf{x}N_\xi - N_\mathbf{x}}{\overline{m}} \rfloor + N_\mathbf{x}N_\xi = \lfloor \frac{N_\mathbf{x}(N_\xi(\overline{m}+1)-1)}{\overline{m}} \rfloor$ iterations.

We now provide some theoretical analysis of the proposed sparse CG algorithm using the framework of inexact Krylov subspace methods studied in [26,27]. To this end, we first show that the sparse CG algorithm is a projection method on inexact Krylov subspaces. The sparse CG algorithm generates subspaces of the form

$$\text{span}\left\{\mathbf{m}^0 \circ \mathbf{r}^0, \mathbf{m}^1 \circ (\mathbf{A}(\mathbf{m}^0 \circ \mathbf{r}^0)), \mathbf{m}^2 \circ \left(\mathbf{A}(\mathbf{m}^1 \circ (\mathbf{A}(\mathbf{m}^0 \circ \mathbf{r}^0)))\right), \ldots\right\}. \tag{11}$$

Defining $\mathbf{v} = \mathbf{m}^0 \circ \mathbf{r}^0$, it can be seen that $\mathbf{m}^1 \circ (\mathbf{A}(\mathbf{m}^0 \circ \mathbf{r}^0)) = \mathbf{A}_1(\mathbf{v})$ with $\mathbf{A}_1 = (\mathbf{m}^1 \otimes \mathbf{1}^\intercal) \circ \mathbf{A}$, where $\mathbf{1} = (1, 1, \ldots, 1)^\intercal \in \mathbb{N}^{N_\mathbf{x}N_\xi}$. Similarly, $\mathbf{m}^2 \circ \left(\mathbf{A}(\mathbf{m}^1 \circ (\mathbf{A}(\mathbf{m}^0 \circ \mathbf{r}^0)))\right) = \mathbf{m}^2 \circ \mathbf{A}(\mathbf{A}_1\mathbf{v}) = \mathbf{A}_2(\mathbf{A}_1(\mathbf{v}))$ with $\mathbf{A}_2 = (\mathbf{m}^2 \otimes \mathbf{1}^\intercal) \circ \mathbf{A}$. Hence, the sparse CG algorithm generates subspaces of the form $\text{span}\left\{\mathbf{v}, \mathbf{A}_1(\mathbf{v}), \mathbf{A}_2(\mathbf{A}_1(\mathbf{v})), \ldots\right\}$. Since sparse masks have their entries $\mathbf{m}_i^k$ equal to zero (resp. one) for $i \in \mathcal{J}^k$ (resp. $i \notin \mathcal{J}^k$), each matrix $\mathbf{A}_k$ can be written as $\mathbf{A}_k = \mathbf{A} + \mathbf{E}_k$ as in [26,27], with perturbation matrices given by

$$\mathbf{E}_k(i,j) = \begin{cases} 0 & \text{if } i \notin \mathcal{J}^k, \\ -a_{ij} & \text{if } i \in \mathcal{J}^k. \end{cases}$$

As iterations k increase $\mathbf{m}^k \to \mathbf{1}$, meaning that $\mathcal{J}^k \to \{\emptyset\}$, $\mathbf{E}_k \to \mathbf{0}$ and $\mathbf{A}_k \to \mathbf{A}$. As such, the sparse CG algorithm generates inexact Krylov subspaces as in [27], where sequences of perturbed matrices $\mathbf{A}_{\varepsilon_i}$ such that $\lim_{\varepsilon \to 0} \mathbf{A}_\varepsilon = \mathbf{A}$ were considered.

Since the classical CG method is equivalent to the Full Orthogonalization Method (FOM) used within Arnoldi's procedure in the symmetric case, the sparse CG algorithm can be analyzed within the framework of inexact Krylov subspace methods. Following [26], we consider the inexact Arnoldi procedure

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_{m+1}\mathbf{H}_m - [\mathbf{E}_1\mathbf{v}_1, \mathbf{E}_2\mathbf{v}_2, \ldots, \mathbf{E}_m\mathbf{v}_m], \tag{12}$$

where $\mathbf{H}_m \in \mathbb{R}^{(m+1)\times m}$ and $\mathbf{V}_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m] \in \mathbb{R}^{N_\mathbf{x}N_\xi \times m}$ is such that $\mathbf{V}_m^\intercal \mathbf{V}_m = \mathbf{I}$. When $\mathbf{A}$ is symmetric, the Arnoldi process coincides with the Lanczos process, that is $\widehat{\mathbf{H}}_m = (\mathbf{H}_m)_{1:m,1:m}$ is symmetric tridiagonal. At the m-th iteration ($m \geq 1$) the "true residual" is defined as $\mathbf{r}^m = \mathbf{b} - \mathbf{A}\mathbf{u}^m = \mathbf{b} - \mathbf{A}(\mathbf{u}^0 + \mathbf{V}_m\mathbf{y}_m) = \mathbf{r}^0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m$ with $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{u}^0$ and $\mathbf{v}_1 = \mathbf{r}^0/||\mathbf{r}^0||_2$. By contrast, the "truncated residual" is given by $\widetilde{\mathbf{r}}^m = \mathbf{r}^0 - \mathbf{V}_{m+1}\mathbf{H}_m\mathbf{y}_m$, which yields $\mathbf{r}^m = \widetilde{\mathbf{r}}^m + [\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m$ for $m \geq 1$, with $\widetilde{\mathbf{r}}^0 = \mathbf{r}^0$. A natural assumption used in [26] is that the perturbation terms satisfy $||[\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m||_2 < \gamma\, ||\mathbf{A}||_2$, where $||\mathbf{A}||_2 = \sup\{||\mathbf{A}\mathbf{x}||_2,\ ||\mathbf{x}||_2 = 1\}$ and $\gamma \in (0,1)$. When applied to vectors, $||\cdot||_2$ denotes the Euclidean norm, i.e. $||\mathbf{x}||_2 = (\mathbf{x}^\intercal\mathbf{x})^{1/2}$.

We now provide a bound for the perturbation terms when $\mathbf{y}_m$ is computed using FOM as $\mathbf{y}_m^{\text{fom}} = \widehat{\mathbf{H}}_m^{-1}(\beta\mathbf{e}_1)$, where $\beta = ||\mathbf{r}_0||_2$, $\mathbf{e}_1 = (1, 0, \ldots, 0)^\intercal \in \mathbb{R}^m$.

Lemma 3.1. *If the eigenvalues of* $\mathbf{A}$ *are such that* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N_\mathbf{x}N_\xi}$ *without being all equal, the perturbation terms* $||[\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m^{\text{fom}}||_2$ *in the sparse CG algorithm satisfy* $||[\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m||_2 \leq \gamma\, ||\mathbf{A}||_2$ *with*

$$\gamma = \frac{m(c_{m-1}(1+2\rho_1))^4}{(\lambda_1 - \lambda_{N_\mathbf{x}N_\xi})^2 \tan^4(\theta_1)} \left(\sum_{k=1}^{m} \left|\left| \widetilde{\mathbf{r}}_{\text{fom}}^{k-1} \right|\right|_2^2 \right)^{1/2}, \tag{13}$$

where c_{m-1} *is the Chebyshev polynomial of the first kind of degree* $m-1$, $\rho_1 = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_{N_\mathbf{x}N_\xi}}$ *and* θ_1 *is the angle between* $\mathbf{v}_1$ *and the eigenvector of* $\mathbf{A}$ *corresponding to* λ_1.

Proof. Since $\mathbf{v}_i$ are orthonormal vectors, it holds

$$\begin{aligned} ||[\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m||_2 &= \left|\left| \left(\sum_{k=1}^{m} \mathbf{E}_k \mathbf{v}_k \mathbf{v}_k^\mathsf{T} \right) \mathbf{V}_m \mathbf{y}_m \right|\right|_2 \leq \sum_{k=1}^{m} ||\mathbf{E}_k||_2 \left|\left| \mathbf{v}_k \mathbf{v}_k^\mathsf{T} \right|\right|_2 ||\mathbf{V}_m \mathbf{y}_m||_2 \\ &= \left(\sum_{k=1}^{m} ||\mathbf{E}_k||_2 \right) ||\mathbf{y}_m||_2. \end{aligned}$$

Using [26, Lemma 5.2] for estimating the FOM components in the inexact Lanczos procedure, and using the fact that $||\mathbf{E}_k||_2 \leq ||\mathbf{A}||_2$ for any symmetric matrix $\mathbf{A}$, it follows that

$$||[\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m||_2 \leq m\, ||\mathbf{A}||_2\, ||\mathbf{y}_m^{\text{fom}}||_2 \leq \frac{m||\mathbf{A}||_2}{\lambda_{\max}^2(\widehat{\mathbf{H}}_m)} \left(\sum_{k=1}^{m} \left|\left| \widetilde{\mathbf{r}}_{\text{fom}}^{k-1} \right|\right|_2^2 \right)^{1/2},$$

where we use $\sigma_{\max}(\widehat{\mathbf{H}}_m) = \lambda_{\max}^2(\widehat{\mathbf{H}}_m)$ for symmetric matrices. According to Kaniel–Paige's convergence theory [14], the largest eigenvalue of $\widehat{\mathbf{H}}_m$ in the Lanczos procedure can be estimated as

$$\lambda_1 \geq \lambda_{\max}(\widehat{\mathbf{H}}_m) \geq \frac{(\lambda_1 - \lambda_{N_\mathbf{x}N_\xi}) \tan^2(\theta_1)}{(c_{m-1}(1+2\rho_1))^2},$$

which leads to (13).

Remark 1. The coefficient γ is expected to be small since $\left|\left| \widetilde{\mathbf{r}}_{\text{fom}}^{k} \right|\right|_2$ is globally (but not necessarily monotonically) decreasing as long as $\text{Im}(\mathbf{V}_{m+1}\mathbf{H}_m)$ keeps growing, see [26, Remark 3.4]. In exact arithmetic $||\widetilde{\mathbf{r}}_{\text{fom}}^{k}||_2 \to 0$ since $||\widetilde{\mathbf{r}}_{\text{fom}}^{k}||_2 = ||\mathbf{r}_{\text{fom}}^{k}||_2$ and the true residual norm is known to tend to zero in the CG case.[1]

Remark 2. We discuss the finiteness of the coefficient γ. First, $\lambda_1 = \lambda_{N_\mathbf{x}N_\xi}$ would correspond to the case when $\mathbf{A} = \lambda_1 \mathbf{I}$, for which solving (3) is not of interest. Second, expanding $\mathbf{v}_1$ in the set of (orthonormal) eigenvectors as $\mathbf{v}_1 = \sum_i d_i \mathbf{z}_i$ with $d_i = \mathbf{v}_1^\mathsf{T}\mathbf{z}_i$, it holds $1 = ||\mathbf{v}_1||_2^2 = \sum_i d_i^2$. Since $\cos(\theta_1) = \mathbf{z}_1^\mathsf{T}\mathbf{v}_1 = d_1$ and $\tan^2(\theta_1) = \frac{1-d_1^2}{d_1^2}$, then $\tan^2(\theta_1) = 0 \Leftrightarrow \mathbf{v}_1 = \pm\mathbf{z}_1$. As such, $\tan^2(\theta_1) = 0$ never holds as it would mean that $\mathbf{v}_1$ and $\mathbf{z}_1$ are collinear, which is highly unlikely in practice. Lastly, when considering stochastic Galerkin matrix equations corresponding to stochastic diffusion models (1), the coefficient ρ_1 is extremely small, since the largest eigenvalues of $\mathbf{A}$ are tightly clustered for those models. As a result, the terms $c_{m-1}(1+2\rho_1)$ increase very slowly with m. In summary, γ does not diverge for those stochastic Galerkin matrix equations.

Remark 3. The idea proposed here can be extended to nonsymmetric stochastic Galerkin matrix equations using a sparse variant of the GMRES method. When using GMRES, that is $\mathbf{y}_m^{gm} = \arg\min_{\mathbf{y}\in\mathbb{R}^m} ||\mathbf{H}_m\mathbf{y} - \beta\mathbf{e}_1||_2$, the norm of the perturbation is bounded as

$$||[\mathbf{E}_1\mathbf{v}_1, \ldots, \mathbf{E}_m\mathbf{v}_m]\mathbf{y}_m||_2 \leq m\, ||\mathbf{A}||_2\, ||\mathbf{y}_m^{gm}||_2 \leq \frac{m||\mathbf{A}||_2}{\sigma_{\max}(\mathbf{H}_m)} \left(\sum_{k=1}^{m} \left|\left| \widetilde{\mathbf{r}}_{gm}^{k-1} \right|\right|_2^2 \right)^{1/2}$$

using [26, Lemma 5.1].

[1] The CG residual norm is such that $||\mathbf{r}^k||_2 \leq C(\mathbf{A},\mathbf{b},\mathbf{u})||\mathbf{u}^k - \mathbf{u}||_\mathbf{A}$, where $\mathbf{u}$ is the exact solution, $\mathbf{b}$ is the right-hand side, $||\cdot||_\mathbf{A}$ is the energy norm $||\mathbf{x}||_\mathbf{A} = (\mathbf{x}^\intercal\mathbf{A}\mathbf{x})^{1/2}$ and $C(\mathbf{A},\mathbf{b},\mathbf{u})$ is a constant that is independent of $\mathbf{u}^k$. Since $||\mathbf{u}^k - \mathbf{u}||_\mathbf{A} \leq 2\left(\frac{\kappa_\mathbf{A}^{1/2}-1}{\kappa_\mathbf{A}^{1/2}+1}\right)^k ||\mathbf{u}^0 - \mathbf{u}||_\mathbf{A}$, the CG residual norm tends to zero.

Table 3
S-IFISS problems #2, #3 and #4 with $M = 5$ and $N_\xi = 126$ and sparse manufactured solutions: timings and numbers of iterations of preconditioned classical/sparse CG algorithms ($\varepsilon_{lv} = 10$). Speed-ups are 21, 4, and 6 for the three test problems.

	Problem #2 ($N_\mathbf{x} = 961$)		Problem #3 ($N_\mathbf{x} = 1023$)		Problem #4 ($N_\mathbf{x} = 1023$)	
	Classical CG	Sparse CG	Classical CG	Sparse CG	Classical CG	Sparse CG
# iterations	849	136	141	41	138	26
timings (s)	383	18	20	5	19	3

Since the proposed sparse CG algorithm fits within the framework of inexact Krylov-based methods, we can use some of the results derived in [26]. For example, the distance between the true and truncated residuals after m iterations of the sparse CG algorithm is such that

$$||\mathbf{r}^m - \widetilde{\mathbf{r}}^m||_2 \leq \sum_{k=1}^{m} \left|\mathbf{e}_k^\mathsf{T} \mathbf{y}_m^{\text{fom}}\right| ||\mathbf{E}_k||_2. \tag{14}$$

In contrast to the classical Arnoldi method, the true residual in sparse CG algorithms is not orthogonal to the basis vectors $(\mathbf{v}_i)_{i=1,\dots,m}$. The loss of orthogonality can be estimated as

$$\left|\left|\mathbf{V}_m^\mathsf{T} \mathbf{r}^m\right|\right|_2 \leq \sum_{k=1}^{m} \left|\mathbf{e}_k^\mathsf{T} \mathbf{y}_m^{\text{fom}}\right| ||\mathbf{E}_k||_2. \tag{15}$$

Equations (14) and (15) suggest that if the components $\left|\mathbf{e}_k^\mathsf{T} \mathbf{y}_m^{\text{fom}}\right|$ tend to decrease across iterations, then the perturbations $||\mathbf{E}_k||_2$ are allowed to grow (see [26] for more details).

4. Numerical experiments

To illustrate, we consider randomly parametrized diffusion models (1) defined on two-dimensional spatial domains with homogeneous Dirichlet boundary conditions and deterministic source terms using the S-IFISS Matlab toolbox [4]. For different test cases, we compare the sparse CG algorithm to the classical CG method in terms of convergence and timings. For a fair comparison, we use the same block-diagonal preconditioner $\mathbf{M} = \mathbf{G}_0 \otimes \mathbf{A}_0$ (see [21]) in both CG algorithms. Simulations were performed using sequential Matlab solvers on a single core of a machine with dual quad Intel Core i5-7500U processors and 8 Gb RAM.

First, we focus on favorable cases when the sought discretized solutions are sparse. To create a sparse solution, we define $\mathbf{w} \in \mathbb{R}^{N_\mathbf{x} N_\xi}$ with components chosen randomly from $[-1, 1]$ and then nullify the $\mathbf{w}_i$ that are smaller than a given value (say, 0.95) for $i > N_\mathbf{x}$. This provides a sparse vector $\mathbf{u}$ from which a right-hand side $\mathbf{b} = \mathbf{A}\mathbf{u}$ is computed, where $\mathbf{A}$ is the discretized diffusion matrix provided by S-IFISS. We manufacture sparse solutions in this way for several S-IFISS test cases, namely problem #2 (where $\mathcal{D} = [-1, 1]^2$, standard deviation $\sigma = 0.525$ in the KL expansion) and problems #3 and #4 (where $\mathcal{D} = [-2, 2] \times [-1, 1]$ and $\sigma = 0.515$). For each problem, we consider $M = 5$ random variables and 4-th PC order, meaning that $N_\xi = 126$.

Timings and numbers of iterations are reported in Table 3, showing that the sparse CG algorithm runs faster than the classical CG method with up to roughly twenty times reduction in computational cost. To illustrate, the convergence of the classical/sparse CG algorithm is shown in Figs. 1 and 2. It is to be noted that the choice made for ε_{lv} generally affects the convergence of the sparse CG algorithm (see, for example, Fig. 1).

Another important aspect of the sparse CG algorithm concerns its ability to recover the sparsity pattern (if any) of the sought solution. Recasting discretized sparse CG solutions $\widehat{\mathbf{u}} \in \mathbb{R}^{N_\mathbf{x} N_\xi}$ as $\widehat{\mathbf{U}} = [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_{N_\xi}] \in \mathbb{R}^{N_\mathbf{x} \times N_\xi}$ with $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \dots, \alpha_{i,N_\mathbf{x}})^\mathsf{T}$, stochastic solutions are built at each FE node $\mathbf{x}_j$ as $\widehat{u}(\mathbf{x}_j, \boldsymbol{\xi}) = \sum_{i=1}^{N_\xi} \alpha_{i,j} \psi_i(\boldsymbol{\xi})$. For each index $j = 1, 2, \dots, N_\mathbf{x}$, the γ-sparsity of $\widehat{u}$ at $\mathbf{x}_j$ is defined as the number of significant PC coefficients

$$\#\{i \text{ such that } |\alpha_{i,j}| > \gamma\}. \tag{16}$$

In Fig. 3, the γ-sparsity of the sparse CG solution as a function of the spatial coordinates is represented for S-IFISS problem #2, showing that the sparse structure of the exact solution is almost recovered by the sparse CG algorithm. This attractive feature (referred to as the "bet of sparsity principle" [15] in the literature) of sparse CG algorithms explains its computational efficiency compared to that of classical CG methods, since it naturally exploits the underlying sparsity (if any) of the sought solution. As another illustration of this feature, we compare the exact sparsity recovery ($\gamma = 0$) of sparse manufactured solutions for various S-IFISS test cases. The preconditioned sparse CG algorithm recovers almost exactly the sparse structure of the exact solutions, by contrast with the classical CG method, which always provides dense solutions (see Table 4).

We finally compare both CG algorithms while approximating the solution to the stochastic diffusion equation. We consider the most challenging S-IFISS test-case, namely problem #2 with the source term given by $f(\mathbf{x}) = \frac{1}{8}(2 - x_1^2 - x_2^2)$. We use

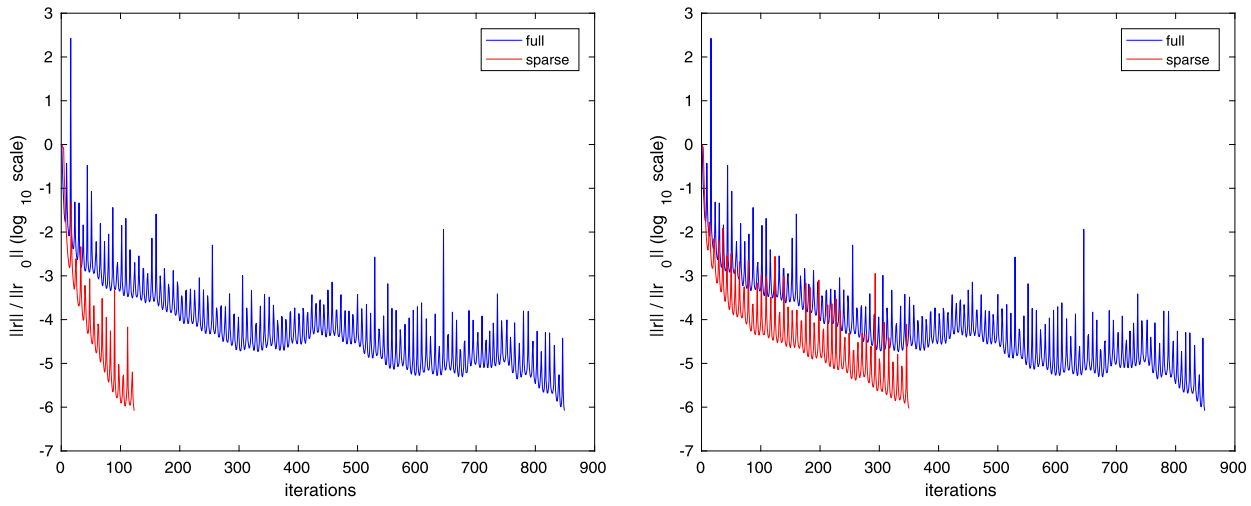

Fig. 1. S-IFISS problem #2 with $M = 5$, $N_{\mathbf{x}} = 961$, $N_{\boldsymbol{\xi}} = 126$, $\sigma = 0.525$ and sparse manufactured solutions: convergence of preconditioned classical/sparse CG algorithms with $\varepsilon = 10^{-6}$, $\theta_{\text{init}} = 10$ and different values of ε_{lv}.

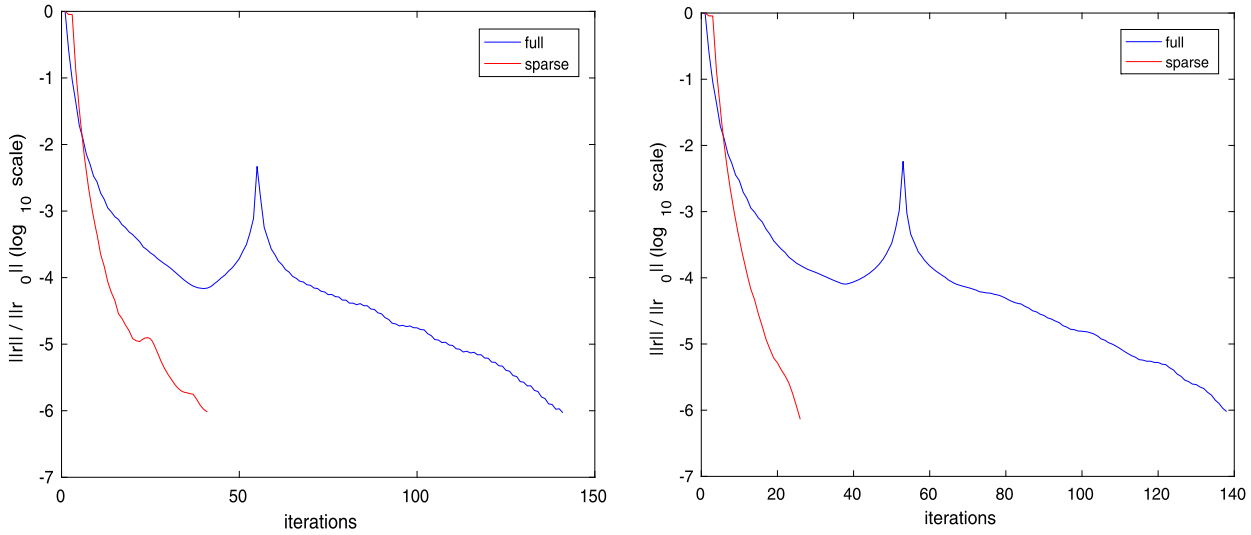

Fig. 2. S-IFISS problems #3 (left) and #4 (right) with sparse manufactured solutions, $M = 5$, $N_{\mathbf{x}} = 1023$, and $\sigma = 0.515$: Convergence of preconditioned classical/sparse CG algorithms with $\varepsilon = 10^{-6}$, $\theta_{\text{init}} = 10$ and $\varepsilon_{lv} = 10$ in sparse CG.

Table 4
S-IFISS test cases with sparse manufactured solutions, $M = 5$, $N_{\boldsymbol{\xi}} = 126$, refined values for $N_{\mathbf{x}}$ and default values for other parameters: exact sparsity recovery ($\gamma = 0$) of the preconditioned sparse CG algorithm with $\varepsilon_{lv} = 10$. For each problem, the classical CG returns a dense solution with 100% of non-zeros.

S-IFISS problem	$N_{\mathbf{x}}$	nnz($\mathbf{u}$)	nnz($\widehat{\mathbf{u}}$)	S-IFISS problem	$N_{\mathbf{x}}$	nnz($\mathbf{u}$)	nnz($\widehat{\mathbf{u}}$)
# 1	705	3.26%	3.29%	# 4	1023	3.29%	7.73%
# 2	961	3.28%	3.47%	# 5	961	3.26%	3.53%
# 3	1023	3.29%	7.76%	# 6	961	3.29%	6.32%

$M = 5$ random variables and 4-th PC order (i.e. $N_{\boldsymbol{\xi}} = 126$), a stretched mesh grid made of $N_{\mathbf{x}} = 961$ points and $\sigma = 0.525$. As can be seen in Fig. 4, both CG algorithms converge similarly, indicating the fact that there is no underlying sparsity in the sought solution. This suggests that the sparse CG algorithm reduces to the classical CG method when considering SPDE models with non-sparse solutions, which is in agreement with the convergence property of sparse CG (see section 3.2). It is worth mentioning that even if the convergence of both algorithms looks comparable, the sparse CG algorithm still runs three times faster than the classical CG method.

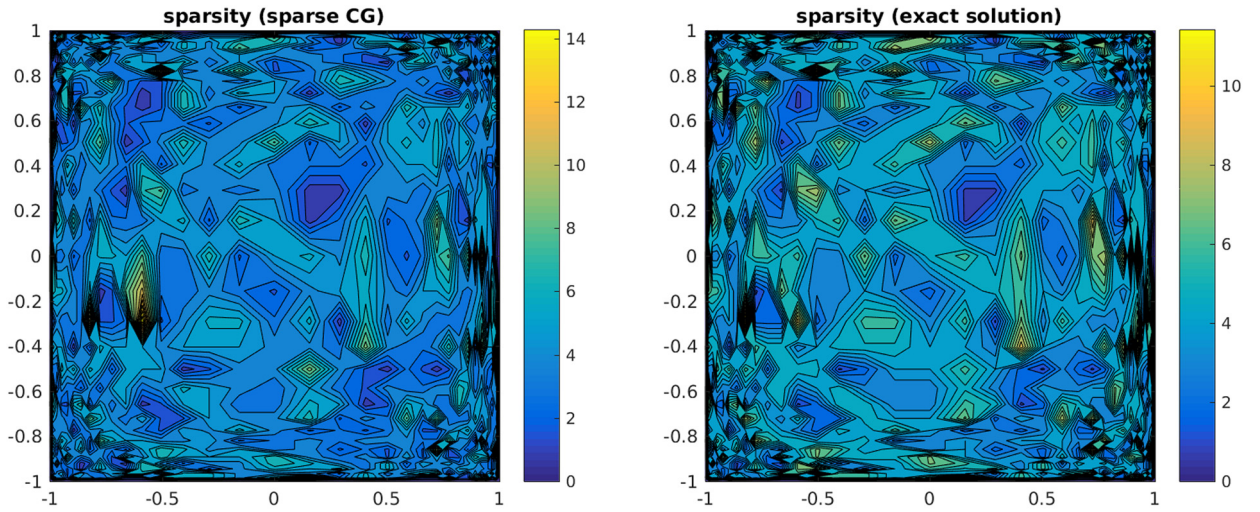

Fig. 3. S-IFISS problem #2 with sparse manufactured solution: γ-sparsity of sparse CG (left) and exact (right) solutions with $\gamma = 10^{-4}$.

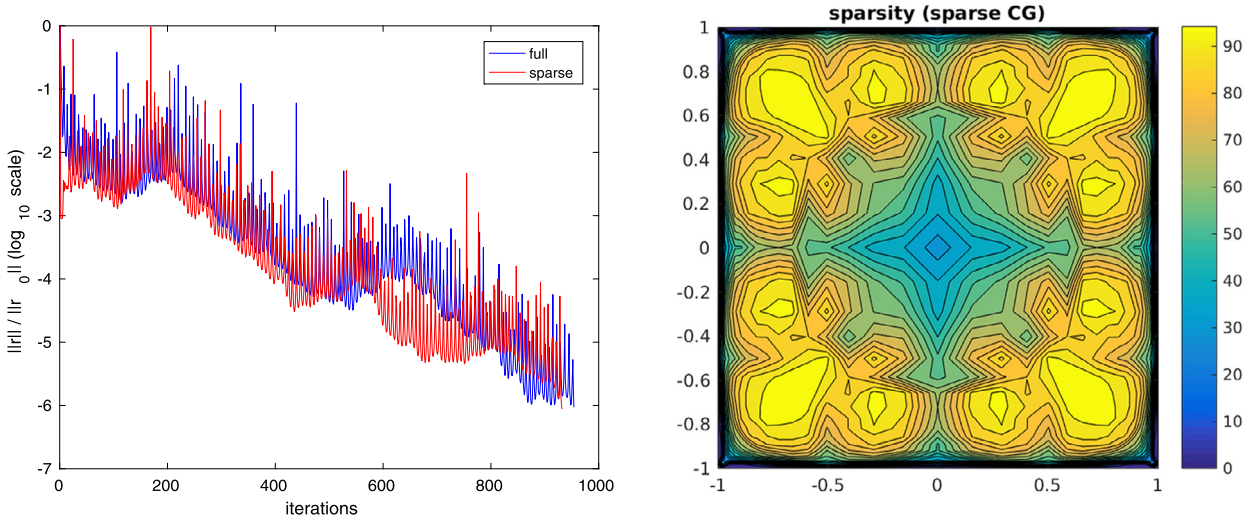

Fig. 4. S-IFISS problem #2 with stochastic diffusion solution, $M = 5$, $N_{\mathbf{x}} = 961$, $N_{\boldsymbol{\xi}} = 126$, $\sigma = 0.525$. Left: convergence of preconditioned classical/sparse CG algorithms with $\varepsilon = 10^{-6}$, $\theta_{\text{init}} = 10$, and $\varepsilon_{lv} = N_{\mathbf{x}} N_{\boldsymbol{\xi}}/100 = 1210$ in sparse CG algorithm. Classical and sparse CG algorithms run in 434 s (919 iterations) and 134 s (933 iterations), respectively. Right: γ-sparsity of the sparse CG solution with $\gamma = 10^{-2}$.

5. Conclusions

In this work, we proposed a sparse CG algorithm for efficiently solving matrix equations arising from stochastic Galerkin projection schemes. The sparse CG algorithm is proved to be convergent and has a lower abstract computational cost compared to that of the classical CG method. Numerical studies conducted on stochastic diffusion models show that the sparse CG algorithm can provide substantial speedups when the exact solution has a sparse representation in a PC basis set. In such cases, the sparse structure of the SPDE solutions is almost recovered, which allows for significant reductions in computational cost compared to the classical CG method. This suggests that the proposed sparse CG algorithm has the potential to efficiently approximate solutions to SPDE models with localized uncertainties. For non-sparse solutions, the sparse CG algorithm displays similar convergence trends as the classical CG method, which is in agreement with the convergence property of the sparse CG algorithm.

Further numerical studies would be required to compare more precisely sparse and classical CG algorithms for a wider class of SPDEs. The idea proposed here can also be used to formulate a sparse variant of the GMRES algorithm with applications to nonsymmetric stochastic Galerkin matrix equations. Interestingly, the sparse CG algorithm can also be used for solving nonlinear SPDEs. The full discretization of such SPDEs leads to systems of non-linear equations of the form $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ that can be solved in a Newton-like fashion. At each iteration, the sparse CG algorithm can be used for solving the (potentially large-scale) matrix equations for the Newton step, $\mathbf{f}'(\mathbf{u}^k)\boldsymbol{\delta}^k = -\mathbf{f}(\mathbf{u}^k)$, where $\mathbf{f}'(\mathbf{u})$ is the Jacobian matrix of $\mathbf{f}$. Using this

approach, it is expected to obtain sparse approximate solutions that would be computationally advantageous for nonlinear SPDE models whose solution admits a sparse representation in a PC basis set.

Acknowledgements

This research is funded by an NSERC Discovery Grant and the Canada Research Chairs program.

References

[1] C. Audouze, P. Håkansson, P.B. Nair, A stopping criterion for iterative solution of stochastic Galerkin matrix equations, Int. J. Uncertain. Quantificat. 6 (3) (2016) 245–269.
[2] I. Babuška, R. Tempone, G.E. Zouraris, Galerkin finite element approximations of stochastic elliptic partial differential equations, SIAM J. Numer. Anal. 42 (2004) 800–825.
[3] J. Bäck, F. Nobile, L. Tamellini, R. Tempone, Convergence of quasi-optimal Stochastic Galerkin methods for a class of PDES with random coefficients, Comput. Math. Appl. 67 (4) (2014) 732–751.
[4] A. Bespalov, C.E. Powell, D. Silvester, Energy norm a posteriori estimation for parametric operator equations, SIAM J. Sci. Comput. 36 (2014) 339–363.
[5] M. Bieri, C. Schwab, Sparse High Order FEM for Elliptic SPDEs, Research Report No. 2008-22, ETH, Zürich, Switzerland, 2008.
[6] T. Blumensath, Accelerated iterative hard thresholding, Signal Process. 92 (2012) 752–756.
[7] T. Butler, P. Constantine, T. Wildey, A posteriori error analysis of parameterized linear systems using spectral methods, SIAM J. Matrix Anal. Appl. 33 (1) (2012) 195–209.
[8] A. Cohen, R. DeVore, C. Schwab, Convergence Rates of Best N-Term Galerkin Approximations for a Class of Elliptic SPDEs, Research Report No. 2009-02, University of Zürich, Switzerland, 2009.
[9] P.G. Constantine, D.F. Gleich, G. Iaccarino, Spectral methods for parameterized matrix equations, SIAM J. Matrix Anal. Appl. 31 (5) (2010) 2681–2699.
[10] O.G. Ernst, E. Ullmann, Stochastic Galerkin matrices, SIAM J. Matrix Anal. Appl. 31 (4) (2010) 1848–1872.
[11] M.A.T. Figueiredo, R.D. Nowak, Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, IEEE J. Sel. Top. Signal Process. 1 (4) (2008) 586–597.
[12] S. Gazzola, J.G. Nagy, Generalized Arnoldi–Tikhonov method for sparse reconstruction, SIAM J. Sci. Comput. 36 (2) (2014) B225–B247.
[13] R. Ghanem, P. Spanos, Stochastic Finite Elements: A Spectral Approach, Springer Verlag, New York, 1991.
[14] G.H. Golub, C.F. Van Loan, Matrix Computations, 4th edition, Johns Hopkins University Press, Baltimore, MD, USA, 2013.
[15] T. Hastie, R. Tibshirani, M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations, Monographs on Statistics and Applied Probability, vol. 143, Chapman and Hall/CRC, 2015.
[16] A. Lanza, S. Morigi, L. Reichel, F. Sgallari, A generalized Krylov subspace method for l_p–l_q minimization, SIAM J. Sci. Comput. 37 (5) (2015) S30–S50.
[17] M. Loève, Probability Theory, 4th edition, Springer, 1977.
[18] A. Nouy, A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations, Comput. Methods Appl. Mech. Eng. 196 (2007) 4521–4537.
[19] A. Nouy, O.P. Le Maître, Generalized spectral decomposition for stochastic nonlinear problems, J. Comput. Phys. 228 (2009) 202–235.
[20] B. Øksendal, Stochastic Differential Equations. An Introduction with Applications, Springer-Verlag, Berlin, 1998.
[21] C.E. Powell, H.C. Elman, Block-diagonal preconditioning for spectral stochastic finite-element systems, IMA J. Numer. Anal. 29 (2009) 350–375.
[22] C.E. Powell, D. Silvester, Preconditioning steady-state Navier–Stokes equations with random data, SIAM J. Sci. Comput. 34 (5) (2012) A2482–A2506.
[23] C.E. Powell, D. Silvester, V. Simoncini, An Efficient Reduced Basis Solver for Stochastic Galerkin Matrix Equations, MIMS Report 2015.64, University of Manchester, UK, 2015.
[24] C. Schwab, E. Süli, R.A. Todor, Sparse finite element approximation of high-dimensional transport-dominated diffusion problems, ESAIM: Math. Model. Numer. Anal. 42 (2008) 777–819.
[25] R. Shu, J. Hu, S. Jin, A stochastic Galerkin method for the Boltzmann equation with multi-dimensional random inputs using sparse wavelet bases, Numer. Math., Theory Methods Appl. 10 (2) (2017) 1–23.
[26] V. Simoncini, D.B. Szyld, Theory of inexact Krylov subspace methods and applications to scientific computing, SIAM J. Sci. Comput. 25 (2) (2003) 454–477.
[27] V. Simoncini, D.B. Szyld, Relaxed Krylov subspace approximation, Proc. Appl. Math. Mech. 5 (2005) 797–800.
[28] B. Sousedik, R.G. Ghanem, E.T. Phipps, Hierarchical Schur complement preconditioner for the stochastic Galerkin finite element methods, Numer. Linear Algebra Appl. 21 (1) (2014) 136–151.
[29] R.A. Todor, C. Schwab, Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients, IMA J. Numer. Anal. 27 (2007) 232–261.
[30] E. Ullmann, A Kronecker product preconditioner for stochastic Galerkin finite element discretizations, SIAM J. Sci. Comput. 32 (2) (2010) 923–946.
[31] D. Xiu, J.S. Hesthaven, High order collocation methods for differential equations with random inputs, SIAM J. Sci. Comput. 27 (2005) 1118–1139.
[32] D. Xiu, G.E. Karniadakis, Modeling uncertainty in steady-state diffusion problems via generalized polynomial chaos, Comput. Methods Appl. Mech. Eng. 191 (2003) 4927–4948.