Available online at www.sciencedirect.com

ScienceDirect

Comput. Methods Appl. Mech. Engrg. 276 (2014) 627–658

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

A new perspective on the solution of uncertainty quantification and reliability analysis of large-scale problems

George Stavroulakis [*], Dimitris G. Giovanis, Manolis Papadrakakis, Vissarion Papadopoulos

Institute of Structural Analysis and Antiseismic Research, National Technical University of Athens, Iroon Polytechniou 9, Zografou Campus, Athens 15780, Greece

Received 22 November 2013; received in revised form 17 March 2014; accepted 23 March 2014
Available online 26 April 2014

Abstract

This work revisits the computational performance of non-intrusive Monte Carlo versus intrusive Galerkin methods of large-scale stochastic systems in the framework of high performance computing environments. The purpose of this work is to perform an assessment of the range of the relative superiority of these approaches with regard to a variety of stochastic parameters. In both approaches, the solution of the resulting algebraic equations is performed with a combination of primal and dual domain decomposition methods implementing specifically tailored preconditioners. The solution of repeated simulations of the Monte Carlo method is accelerated with an A-orthogonalization procedure aiming at reducing the iterations of subsequent simulations, while the solution of the augmented equations of the stochastic Galerkin method is enhanced with preconditioners which combine the block diagonal features of the resulting matrices as well as the sparsity pattern of the off block-diagonal terms. Numerical results are presented, demonstrating the efficiency of the proposed implementations on a large-scale 3D problem with different stochastic characteristics and useful conclusions are derived regarding the ranges of stochastic parameters in which non-intrusive solvers have a superior performance compared to intrusive ones and vice versa.
© 2014 Published by Elsevier B.V.

Keywords: Spectral stochastic finite element method; Monte Carlo simulation; Polynomial chaos; Karhunen–Loève decomposition; Primal–dual domain decomposition; FETI method

1. Introduction

The most straightforward technique of solving stochastic partial differential equations (PDE) are the widely applicable non-intrusive Monte Carlo (MC) methods. They can handle any type of problems (linear, nonlinear,

[*] Corresponding author. Tel.: +30 2107721694.
E-mail address: stavroulakis@nessos.gr (G. Stavroulakis).

http://dx.doi.org/10.1016/j.cma.2014.03.009
0045-7825/© 2014 Published by Elsevier B.V.

dynamic) as well as any kind of uncertainty in the load or in the system properties and they can be implemented in a non-intrusive manner in the framework of existing deterministic solvers. In particular, when dealing with deterministic external loading, Monte Carlo methods feature the solution of successive linear systems with multiple left-hand sides, since only the coefficient matrix $\mathbf{K}$ changes in every simulation. Due to the fact that the solution process has to start from the beginning, a new stiffness matrix needs to be formed at each simulation. Thus, the repeated solutions of the system of equations for each newly formed solution becomes a major computational task that hinders the stochastic assessment of large-scale problems with MC methods. Such solution can be performed either with a standard direct method based on Cholesky factorization or with iterative methods. The solution with a direct method has two major drawbacks: poor performance in 3D large scale problems and in parallel and distributed computing environments, as well as incapability of exploiting the near-by nature of the successive simulations.

In order to alleviate the incapability of direct schemes to exploit the proximity of the resulting systems of equations, iterative solvers have been proposed which are customized to the particular properties of the equilibrium equations arising in the context of MC methods. Such iterative solvers have been presented for sequential [1–5] as well as for parallel computing environments [6–11]. The resulting near-by problems can be effectively solved using the preconditioned conjugate gradient (PCG) algorithm equipped with a preconditioner following the rationale of incomplete Cholesky preconditioning. This solution procedure consists of utilizing the deterministic $\mathbf{K}_0$ stiffness matrix as its preconditioner throughout the entire simulation process for the solution of the near-by problems. The repeated solutions required for the preconditioning step of the MC-PCG-Skyline algorithm can be treated as problems with multiple right-hand sides, since the entries in the residual vector are updated at each PCG iteration of each MC simulation. A direct solver for this procedure is proposed in [12] and the dual decomposition FETI method is proposed in [11].

On the other hand, recently proposed approaches, such as stochastic collocation and Galerkin methods, are intrusive and are using tensor product spaces for the spatial and stochastic discretizations. In the case where the uncertain input parameters are modeled via the Karhunen–Loève (KL) expansion and the system response is projected on a polynomial chaos (PC) basis, the method is called spectral stochastic finite element method (SSFEM). SSFEM approach applies a Galerkin minimization in order to transform a stochastic PDE into a coupled set of deterministic PDEs. Thus, the solution of stochastic problems using the SSFEM approach has to be performed on augmented linear equation systems which can be up to orders of magnitude larger than the corresponding deterministic ones [13–21].

For large-scale problems the solution of such augmented algebraic systems can become quite challenging due to the increased memory and computational resources required. Solution techniques for addressing these problems are based on adaptive methods [22,23] and on iterative solvers like the block Gauss–Jacobi [8,24–26] and the PCG [2,5,8,18,27–32]. The variant proposed in [32] is an extension of the domain decomposition FETI-DP method to SSFEM problems where the Lagrange multipliers are enhanced in order to take into account the interaction forces occurring at both the deterministic and stochastic parts. Although such an implementation might prove efficient when compared to the aforementioned iterative solvers, its performance can be degraded when dealing with large KL and PC expansion orders. The two-level variant proposed in [29] proves to be quite efficient since it exploits the block-sparsity structure of the augmented stiffness matrix while utilizing a domain decomposition solver, optimized for multiple right-hand sides.

The present work revisits the computational performance of non-intrusive Monte Carlo versus intrusive Galerkin methods for large-scale stochastic systems in the framework of high performance computing environments. The purpose of this work is to perform an assessment of the range of the relative superiority of these approaches with regard to a variety of stochastic parameters. In both approaches, the solution of the resulting algebraic equations is performed with a combination of primal and dual domain decomposition methods implementing specifically tailored preconditioners. The solution of repeated simulations of the Monte Carlo method is accelerated with an A-orthogonalization procedure aiming at reducing the iterations of subsequent simulations, while the solution of the augmented equations of the stochastic Galerkin method is enhanced with preconditioners which combine the block diagonal features of the resulting matrices as well as the sparsity pattern of the off block-diagonal terms. Numerical results are presented demonstrating the efficiency of the proposed implementations on a variety of problems with different stochastic characteristics and useful conclusions are derived regarding the ranges of stochastic parameters in which non-intrusive solvers have a superior performance compared to intrusive ones and vice versa.

2. Monte Carlo simulation in high performance computing environments

MC methods require the solution of problems of the form

$$\mathbf{K}_i\mathbf{u}_i = \mathbf{f}_i \quad i(i = 1, \ldots, n_{sim}) \tag{1}$$

where $\mathbf{K}_i$ is the stiffness matrix corresponding to the stochastic realization of the i-th simulation, $\mathbf{u}_i$ is the corresponding vector of unknown nodal displacements, n_{sim} is the number of Monte Carlo simulations and $\mathbf{f}$ is the vector of nodal loads. The size of the stiffness matrix and the corresponding vectors is equal to the size of the equivalent deterministic problem. Thus, if $\mathbf{K}_0$ is the stiffness matrix of the deterministic problem with dimensions $N \times N$, Eq. (1) can be written as

$$(\mathbf{K}_0 + \Delta\mathbf{K}_i)\mathbf{u}_i = \mathbf{f}, \quad i = 1, \ldots, n_{sim} \tag{2}$$

which specifies a set of near-by problems.

If the uncertainties in the input parameters are modeled by Gaussian random fields then the truncated KL expansion is defined as [33,34]:

$$\widehat{a}(\mathbf{x}, \vartheta) = a_0(\mathbf{x}) + \sum_{i=1}^{M} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) \tag{3}$$

where, $a_0(\mathbf{x})$ denotes the mean value of the random field, $\xi_i(\theta)$ is a set of uncorrelated zero mean Gaussian random variables, θ being the random event. λ_i and $\phi_i(\mathbf{x})$ are the eigenvalues and mutually orthogonal eigenfunctions of its covariance function $C(\mathbf{x}_1, \mathbf{x}_2)$ which may be calculated in the domain D of the random field $a(\mathbf{x}, \theta)$, from the solution of the homogeneous Fredholm integral equation of the second kind given by

$$\int_D C(\mathbf{x}_1, \mathbf{x}_2)\phi_i(\mathbf{x}_1) = \lambda_i\phi_i(\mathbf{x}_2) \tag{4}$$

Thus, in the case of Gaussian random fields Eq. (2) can be written as

$$\left(\mathbf{K}_0 + \sum_{j=1}^{M} \mathbf{K}_j\xi_j(\theta)\right)\mathbf{u}(\theta) = \mathbf{f} \tag{5}$$

$\mathbf{K}_j$ are deterministic and are given by:

$$\mathbf{K}_j = \mathbf{K}\left(\sqrt{\lambda_j}\phi_j(\mathbf{x})\right) \quad j = 1, \ldots, M \tag{6}$$

These repeated solutions can be performed either with a standard direct method based on Cholesky factorization or with preconditioned iterative methods. In high performance computing environments which feature computing systems with multicore processors and distributed memory architectures, iterative schemes are more advantageous since they manage to harness the computational power of such environments while being more easily custom tailored to the particular properties of the equilibrium equations arising in the context of the numerical simulation used. In this work, we have further improved two variants of the MC-PCG-Skyline method previously proposed in [11,12].

2.1. The MC-PCG-Skyline method

The PCG algorithm, when solving a linear system of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$ with a preconditioner $\tilde{\mathbf{A}}$, is depicted in Table 1 for iteration k.

Table 1
The PCG algorithm.

Solution estimate		$\mathbf{x}^k = \mathbf{x}^{k-1} + \eta^{k-1}\mathbf{p}^{k-1}$
Residual vector		$\mathbf{r}^k = \mathbf{r}^{k-1} - \eta^{k-1}\mathbf{q}^{k-1}$
Preconditioned residual vector		$\mathbf{z}^k = \tilde{\mathbf{A}}^{-1}\mathbf{r}^k$
Search vector	Using re-orthogonalization	$\mathbf{p}^k = \mathbf{z}^k - \sum_{i=0}^{k-1} \frac{\mathbf{z}^{k^T}\mathbf{q}^i}{\mathbf{p}^{k^T}\mathbf{q}^i}\mathbf{p}^i$
A matrix product vector		$\mathbf{q}^k = \mathbf{A}\mathbf{p}^k$
η Estimation	Using re-orthogonalization	$\eta^k = \frac{\mathbf{p}^{k^T}\mathbf{r}^k}{\mathbf{p}^{k^T}\mathbf{q}^k}$

- Initialization phase: $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0\mathbf{z}^0 = \tilde{\mathbf{A}}^{-1}\mathbf{r}^0,\ \mathbf{p}^0 = \mathbf{z}^0,\ \mathbf{q}^0 = \mathbf{A}\mathbf{p}^0,\ \eta^0 = \frac{\mathbf{p}^{0^T}\mathbf{r}^0}{\mathbf{p}^{0^T}\mathbf{q}^0},$
- Repeat for $k = 1, 2\ldots$ until convergence:

The PCG algorithm equipped with a preconditioner following the rationale of incomplete Cholesky preconditioning features an error matrix $\mathbf{E}_i$. This matrix is dependent on the discarded elements of the lower triangular matrix produced by the incomplete Cholesky factorization procedure, which do not satisfy a specified magnitude or position criterion [35]. Considering the near-by problems of the form (2), if matrix $\mathbf{E}_i$ is taken as $\Delta\mathbf{K}_i$, the preconditioning matrix becomes the initial matrix $\tilde{\mathbf{A}} = \mathbf{K}_0$. The PCG algorithm equipped with the latter preconditioner throughout the entire solution process constitutes the MC-PCG-Skyline method for the solution of the n_{sim} near-by problems of Eq. (2).

With the preconditioning matrix $\tilde{\mathbf{A}} = \mathbf{K}_0$ remaining the same during the successive Monte Carlo simulations, the repeated solutions required for the evaluation of the preconditioned residual vector $\mathbf{z}^k = \tilde{\mathbf{A}}^{-1}\mathbf{r}^k$ can be treated as problems with multiple right-hand sides, since this vector needs to be evaluated at each PCG iteration k of each simulation i. In order for this evaluation to be efficient, a solution scheme capable of solving efficiently problems with multiple right-hand sides is required.

The original MC-PCG-Skyline algorithm proposed in [12] uses a Cholesky direct solver for performing the proconditioning step, where $\mathbf{K}_0$ is factorized to LL^T at the beginning of the Monte Carlo simulation procedure and each evaluation of the preconditioned residual vector is carried out by a forward substitution, a vector operation and a backward substitution. Another implementation for obtaining the preconditional residual vector $\mathbf{z}^k$ was proposed in [11] where the dual domain decomposition FETI method is applied to perform the repeated solutions in parallel computing environment. In the present work, each evaluation of the preconditioned residual vector is carried out using a PFETI solver [36], optimized for multiple right-hand sides [37], as described in the following section, adhering to the rationale of the PCG method where the preconditioning step is performed with the FETI method [11].

2.2. The PSM, FETI and PFETI methods

2.2.1. The primal substructuring method (PSM)

The primal substructuring method (PSM) is based on the elimination of the internal dof of the subdomains and solving for the interface dof. The interface problem is of the form:

$$\mathbf{S}\mathbf{u}_b = \hat{\mathbf{f}}_b \tag{7}$$

where

$$\mathbf{S} = \mathbf{L}_b\mathbf{S}^s\mathbf{L}_b \tag{8}$$

$$\mathbf{u}_b^s = \mathbf{L}_b\mathbf{u}_b \tag{9}$$

$$\hat{\mathbf{f}}_b = \mathbf{L}_b^T\hat{\mathbf{f}}_b^s \tag{10}$$

and

$$\mathbf{S}^s = \begin{bmatrix} \mathbf{S}^{(1)} & & & & \\ & \ddots & & & \\ & & \mathbf{S}^{(s)} & & \\ & & & \ddots & \\ & & & & \mathbf{S}^{(N_s)} \end{bmatrix}$$

$$\hat{\mathbf{f}}_b^s = \begin{bmatrix} \hat{\mathbf{f}}_b^{(1)} & \cdots & \hat{\mathbf{f}}_b^{(N_s)} \end{bmatrix}$$

N_s is the number of subdomains s of the global domain, $\mathbf{L}$ is the Boolean global to local mapping operator, while index b denoting that the vectors are applied to interface dof. Moreover, the Schur complement matrix $\mathbf{S}^{(s)}$ and $\hat{\mathbf{f}}_b^{(s)}$ are defined as follows:

$$\mathbf{S}^{(s)} = \mathbf{K}_{bb}^{(s)} - \mathbf{K}_{bi}^{(s)} \left(\mathbf{K}_{ii}^{(s)}\right)^{-1} \mathbf{K}_{ib}^{(s)} \tag{11}$$

$$\hat{\mathbf{f}}_b^{(s)} = \mathbf{f}_b^{(s)} - \mathbf{K}_{bi}^{(s)} \left(\mathbf{K}_{ii}^{(s)}\right)^{-1} \mathbf{f}_i^s \tag{12}$$

with subscript i denoting the restriction of the matrices and vectors to internal degrees of freedom (dof).

All $\mathbf{K}_{xy}$, with $x = b,\ i$ and $y = b,\ i$, can be obtained by rewriting the local subdomain problem to the form:

$$\begin{bmatrix} \mathbf{K}_{bb}^{(s)} & \mathbf{K}_{bi}^{(s)} \\ \mathbf{K}_{ib}^{(s)} & \mathbf{K}_{ii}^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_b^{(s)} \\ \mathbf{u}_i^{(s)} \end{bmatrix} - \begin{bmatrix} \mathbf{B}_b^T \\ 0 \end{bmatrix} \cdot \lambda \tag{13}$$

with λ being the traction forces between each subdomain and $\mathbf{B}_b$ is a signed Boolean matrix. Displacement and force vectors are defined as:

$$\mathbf{u}_b^s = \begin{bmatrix} \mathbf{u}_b^{(1)^T} & \cdots & \mathbf{u}_b^{(s)^T} & \cdots & \mathbf{u}_b^{(N_s)^T} \end{bmatrix}^T \tag{14}$$

$$\mathbf{f}_b^s = \begin{bmatrix} \mathbf{f}_b^{(1)^T} & \cdots & \mathbf{f}_b^{(s)^T} & \cdots & \mathbf{f}_b^{(N_s)^T} \end{bmatrix}^T \tag{15}$$

2.2.2. The dual substructing method (FETI)

The dual substructing FETI method proposed by Farhat and Roux [38] is based on solving the following dual problem after the elimination of all internal and boundary dof:

$$\begin{bmatrix} \mathbf{F}_I & -\mathbf{G} \\ -\mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{e} \end{bmatrix} \tag{16}$$

where

$$\mathbf{F}_I = \mathbf{B}\mathbf{K}^{s^+}\mathbf{B}^T \tag{17}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{B}^{(1)}\mathbf{R}^{(1)} & \cdots & \mathbf{B}^{(N_s)}\mathbf{R}^{(N_s)} \end{bmatrix}^T \tag{18}$$

$$\mathbf{d} = \mathbf{B}\mathbf{K}^{s^+}\mathbf{f}^s \tag{19}$$

$$\mathbf{e} = \mathbf{R}^{s^T}\mathbf{f}^s \tag{20}$$

$$\mathbf{R}^{(s)} = \mathrm{null}\left(\mathbf{K}^{(s)}\right) \tag{21}$$

and

$$\mathbf{K}^{s^+} = \begin{bmatrix} \mathbf{K}^{(1)^+} & & & & \\ & \ddots & & & \\ & & \mathbf{K}^{(s)^+} & & \\ & & & \ddots & \\ & & & & \mathbf{K}^{(N_s)^+} \end{bmatrix} \tag{22}$$

$$\mathbf{R}^s = \left[\mathbf{R}^{(1)^T} \quad \cdots \quad \mathbf{R}^{(N_s)^T} \right]^T \tag{23}$$

$$\mathbf{f}^s = \left[\mathbf{f}^{(1)^T} \quad \cdots \quad \mathbf{f}^{(N_s)^T} \right]^T \tag{24}$$

with the superscript $(+)$ denoting the generalized inverse of a given matrix. The FETI method is usually implemented with a preconditioned conjugate projected gradient (PCPG) algorithm for the solution of the Lagrange multipliers in Eq. (16).

2.2.3. The primal–dual substructuring method (PFETI)

The PFETI belongs to the primal–dual substructuring methods family, introduced by Fragakis and Papadrakakis [36]. These methods are derived by the PSM with the following rationale: The PSM solves Eq. (7) in order to evaluate the interface displacements $\mathbf{u}_b$, using the PCG algorithm. An efficient preconditioner of PSM treats the residual vector $\mathbf{r} = \hat{\mathbf{f}}_b - \mathbf{S}\mathbf{u}_b$ as an external force vector applied on the subdomain interface in order to provide an accurate estimation of the interface displacements caused by the aforementioned force vector. The preconditioner which estimates the subdomain interface displacements after performing the first iteration of a dual domain decomposition method is the one proposed at [36]. The PCG algorithm applied to Eq. (7) with the following preconditioner

$$\tilde{\mathbf{S}}^{-1} = \mathbf{L}_{p_b}^T \mathbf{H}_b^T \mathbf{S}^{s^+} \mathbf{H}_b \mathbf{L}_{p_b} \tag{25}$$

where

$$\mathbf{H}_b = \mathbf{I} - \mathbf{B}_b \mathbf{G} \left(\mathbf{G}^T \mathbf{G}\right)^{-1} \mathbf{R}_b^{s^T} \tag{26}$$

constitutes the PFETI method.

One of the main differences between the FETI and PFETI methods is their interface problem. The interface problem of PFETI is based on the primal displacements variables at the interface nodes, while for FETI the interface problem is dual as it is based on the interaction forces, the Lagrange multipliers, between the subdomains. This difference leads to a significant increase of the interface problem size of the FETI method when compared to PFETI, particularly in 3D problems and irregular meshes where the domain decomposition produces a large number of crosspoints at the intersections of the subdomains which lead to an increase of the number of Lagrange multipliers. This difference results to superior performance with respect to iteration count of the PFETI method, as shown in [36], when compared to the FETI method equipped with either the lumped or Dirichlet preconditioners. On the other hand, each preconditioning step of the FETI method, when equipped with the lumped preconditioner, is cheaper compared to the PFETI method since it only involves a forward and backward substitution whereas for PFETI and the Dirichlet-preconditioned FETI, a series of matrix–vector multiplications are also required.

2.3. Optimizing the solution with multiple right-hand sides

When using the PCG algorithm to solve problems with multiple right-hand sides, convergence can be accelerated by utilizing appropriately information accumulated during the previous solutions. In particular, given a sequence of linear systems with a constant left-hand side matrix **A** and multiple right-hand side vectors of the form

$$\mathbf{A}\mathbf{x}_i = \mathbf{b}_i, \quad i = 1, \ldots, j, j+1, \ldots, n_a \tag{27}$$

where n_a is the number of solutions required, the number of PCG iterations required for each linear system may be reduced using the Krylov subspaces generated from search vectors $\mathbf{p}$ during the previous solutions. For the solution of the linear system $j+1$, the following first solution estimate is considered:

$$\mathbf{x}^0_{j+1} = \mathbf{P}_{np}\mathbf{x}_p \tag{28}$$

with

$$\begin{aligned} &\mathbf{P}_{np} = [\mathbf{p}_1 \ldots \mathbf{p}_{n0}] \\ &\mathbf{x}_p = \left(\mathbf{Q}^T_{n_p}\mathbf{P}_{n_p}\right)^{-1}\mathbf{P}^T_{n_p}\mathbf{b}_{j+1}\mathbf{Q}_{n_p} \\ &\mathbf{Q}_{n_p} = \mathbf{A}\mathbf{P}_{n_p} = [\mathbf{A}\mathbf{p}_1 \ldots \mathbf{A}\mathbf{p}_{n0}] = [\mathbf{q}_1 \ldots \mathbf{q}_{n0}] \end{aligned} \tag{29}$$

Given that search vector $\mathbf{p}$ and matrix product vector $\mathbf{q}$ using a re-orthogonalization procedure are ensured to be A-orthogonal, the evaluation of $\mathbf{x}_p$ is trivial since $\mathbf{Q}^T_{n_p}\mathbf{P}_{n_p}$ has values only in its diagonal. Moreover, the search vector evaluation step can be carried out using not necessarily all but a fraction of the vectors stored from all the accumulated solutions.

3. SSFEM in high performance computing environment

In the SSFEM approach, the system response is projected in a PC basis as follows

$$\mathbf{u}(\theta) = \sum_{j=0}^{Q-1}\mathbf{u}_j\Psi_j(\boldsymbol{\xi}) \tag{30}$$

where $\{\Psi_j(\boldsymbol{\xi})\}_{j=0}^{Q-1} = \{\Psi_j((\xi_1(\theta), \ldots, \xi_M(\theta)))\}_{j=0}^{Q-1}$ is the PC basis, consisting of the M-dimensional zero mean and orthogonal Hermite polynomials of order p, satisfying:

$$\begin{aligned} &\Psi_0 = 1 \\ &E[\Psi_j] = 0 \quad j > 0 \\ &E[\Psi_j(\theta)\Psi_k(\theta)] = 0 \quad j \neq k \end{aligned} \tag{31}$$

The value of Q in Eq. (30) is determined by the following formula

$$Q = \binom{M+p}{p} \tag{32}$$

In this case Eq. (5) can be written as

$$\left(\sum_{i=0}^{M}\mathbf{K}_i\xi_i(\theta)\right) \cdot \left(\sum_{j=0}^{Q-1}\mathbf{u}_j\Psi_j(\theta)\right) = \mathbf{f} \tag{33}$$

with the following residual due to the truncation error

$$\boldsymbol{\epsilon}_{M,Q} = \sum_{i=0}^{M}\sum_{j=0}^{Q-1}\mathbf{K}_i\mathbf{u}_j\xi_i(\theta)\Psi_j(\theta) - \mathbf{f} \tag{34}$$

The best approximation of the exact solution $\mathbf{u}(\theta)$ spanned by $\{\Psi_k\}_{k=0}^{Q-1}$ is obtained by minimizing this residual in a mean square sense:

$$E[\boldsymbol{\epsilon}_{M,Q} \cdot \Psi_k] = 0 \quad k = 0, 1, \ldots, Q-1 \tag{35}$$

By introducing the following notation:

$$c_{ijk} = E[\xi_i\Psi_j\Psi_k] \tag{36}$$

$$\mathbf{f}_k = E[\Psi_k\mathbf{f}] \tag{37}$$

Eq. (33) can be rewritten as

$$\sum_{i=0}^{M}\sum_{j=0}^{Q-1} c_{ijk}\mathbf{K}_i\mathbf{u}_j = \mathbf{f}_k \quad k = 0,\ldots,Q-1 \tag{38}$$

and by defining for the sake of simplicity

$$\mathbf{K}_{jk} = \sum_{i=0}^{M+1} c_{ijk}\mathbf{K}_i \tag{39}$$

Eq. (38) rewrites as follows:

$$\sum_{j=0}^{Q-1} \mathbf{K}_{jk}\mathbf{u}_j = \mathbf{f}_k \tag{40}$$

In the above equation, each $\mathbf{u}_j$ is a $N-$dimensional vector and each $\mathbf{K}_{jk}$ is a matrix of size $N \times N$. The Q different equations can be cast in a matrix form of size $(N \cdot Q \times N \cdot Q)$ as follows:

$$\boldsymbol{K} \cdot \boldsymbol{u} = \boldsymbol{f} \tag{41}$$

where,

$$\boldsymbol{K} = \begin{bmatrix} \sum_{i=0}^{M} c_{i,0,0}\mathbf{K}_i & \sum_{i=0}^{M} c_{i,1,0}\mathbf{K}_i & \cdots & \sum_{i=0}^{M} c_{i,Q-1,0}\mathbf{K}_i \\ \sum_{i=0}^{M} c_{i,0,1}\mathbf{K}_i & \sum_{i=0}^{M} c_{i,1,1}\mathbf{K}_i & \cdots & \sum_{i=0}^{M} c_{i,Q-1,1}\mathbf{K}_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{M} c_{i,0,Q-1}\mathbf{K}_i & \sum_{i=0}^{M} c_{i,1,Q-1}\mathbf{K}_i & \cdots & \sum_{i=0}^{M} c_{i,Q-1,Q-1}\mathbf{K}_i \end{bmatrix}$$

and

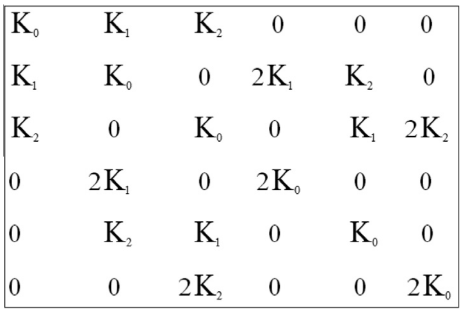
$$\boldsymbol{u} = [\mathbf{u}_0, \quad \mathbf{u}_1, \quad \cdots, \quad \mathbf{u}_{Q-1}]^T$$

$$\boldsymbol{f} = [\mathbf{f}_0, \quad \mathbf{f}_1, \quad \cdots, \quad \mathbf{f}_{Q-1}]^T$$

The sparsity patterns of the $(N \times N)$ non-zero block sub-matrices of $\boldsymbol{K}$ for two dimensional elasticity cases is shown in Figs. 1 and 2.

After solving the augmented system for $\boldsymbol{u} = \{\mathbf{u}_k, k = 0,\ldots,Q-1\}$, the required $\mathbf{u}(\theta)$ is computed from:

$$\mathbf{u}(\theta) = \sum_{j=0}^{Q-1} \mathbf{u}_j \Psi_j(\theta) \tag{42}$$



K_0	K_1	K_2	0	0	0
K_1	K_0	0	$2K_1$	K_2	0
K_2	0	K_0	0	K_1	$2K_2$
0	$2K_1$	0	$2K_0$	0	0
0	K_2	K_1	0	K_0	0
0	0	$2K_2$	0	0	$2K_0$

Fig. 1. Sparsity pattern of $\boldsymbol{K}$ for the Gaussian case $(M = 2, p = 2)$.

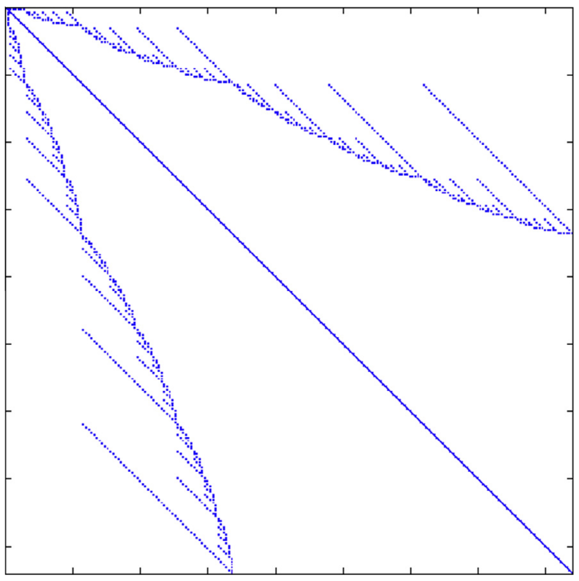


Fig. 2. Sparsity pattern of $\boldsymbol{K}$ for the Gaussian case ($M = 6, p = 4$).

Once the coefficients $\mathbf{u}_j$ of the expansion are computed, approximate statistics of the solution can be derived by MC simulations. In this case however, the MC simulation computational effort is trivial since it is applied directly to the polynomial representation of Eq. (42) without the need of solving a system of equations at each simulation.

3.1. Log-normal random fields

If the random field is considered to be non-Gaussian the probabilistic structure of $\xi_i(\theta)$ in Eq. (3) cannot be determined so the KL expansion is of no practical interest. For the case of log-normal random fields $l(\mathbf{x},\theta)$ the truncated KL expansion of an underlying Gaussian field can be defined as follows

$$\hat{l}(\mathbf{x},\theta) = e^{\left\{g_0(\mathbf{x})+\sum_{i=1}^{M}\sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x})\right\}} \tag{43}$$

Ghanem proposed in [39] an expansion of $l(\mathbf{x},\theta)$ into a polynomial basis in order to be able to include these fields in the context of the spectral stochastic finite element method. The truncated PC expansion of a log-normal random field $l(\mathbf{x},\theta)$ reads

$$\hat{l}(\mathbf{x},\theta) = \sum_{i=0}^{Q-1} l_i(\mathbf{x})\Psi_i(\boldsymbol{\xi}) \tag{44}$$

In this case the stiffness matrix of Eq. (5) becomes

$$\mathbf{K}(\theta) = \sum_{i=0}^{Q-1}\mathbf{K}_i\Psi_i(\boldsymbol{\xi}) \tag{45}$$

where $\Psi_i(\boldsymbol{\xi})$ is the PC basis. Substituting Eq. (45) in Eq. (33), the stochastic equilibrium equation can be written as follows:

$$\left(\sum_{i=0}^{Q-1}\mathbf{K}_i\Psi_i(\theta)\right)\cdot\left(\sum_{j=0}^{Q-1}\mathbf{u}_j\Psi_j(\theta)\right) = \mathbf{f} \tag{46}$$

The Galerkin minimization of error leads to a system of linear equations similar to Eq. (38) where the coefficients c_{ijk} are replaced by:

$$d_{ijk} = E[\Psi_i \Psi_j \Psi_k] \quad i, j, k = 0, \ldots Q-1 \tag{47}$$

The matrix $\boldsymbol{K}$ retains the block-sparsity nature and its diagonally block-dominance albeit to a lesser extent than the Gaussian case.

When dealing with Gaussian stochastic fields, each block of the diagonal is comprised of the deterministic stiffness matrix $\mathbf{K}_0$ scaled by an integer. In log-normal stochastic fields, each block of the diagonal is comprised either of the deterministic stiffness matrix $\mathbf{K}_0$ scaled by an integer or of a linear combination of the deterministic stiffness matrix $\mathbf{K}_0$ and stochastic matrices $\mathbf{K}_1$ to $\mathbf{K}_n$. Matrices $\mathbf{K}_1$ to $\mathbf{K}_n$ may differ up to orders of magnitude when compared to $\mathbf{K}_0$, so matrix $\boldsymbol{K}$ is also considered block-dominant. Figs. 3 and 4 depict the sparsity pattern of the non-zero block sub-matrices of $\boldsymbol{K}$ in the case of a log-normal distribution for the same M and p values considered in the Gaussian case of Figs. 1 and 2. It can be seen that the augmented matrix $\boldsymbol{K}$ in the case of a log-normal random field is far more dense than the one obtained in the case of a Gaussian input (see Figs. 2 and 4).

3.2. Solution of the augmented systems

The augmented systems that are generated when using SSFEM are suitable candidates for iterative solvers since they are flexible enough to be custom tailored to their particular architecture of the augmented systems

$$\begin{array}{cccccc}
K_0 & K_1 & K_2 & 2K_3 & K_4 & 2K_5 \\
K_1 & (K_0+2K_3) & K_4 & 2K_1 & K_2 & 0 \\
K_2 & K_4 & (K_0+2K_5) & 0 & K_1 & 2K_2 \\
2K_3 & 2K_1 & 0 & (2K_0+8K_3) & K_4 & 0 \\
K_4 & K_2 & K_1 & K_4 & (K_0+2K_3) & K_4 \\
2K_5 & 0 & 2K_2 & 0 & K_4 & (2K_0+8K_5)
\end{array}$$

Fig. 3. Sparsity pattern of $\boldsymbol{K}$ for the non-Gaussian case ($M = 2, p = 2$).

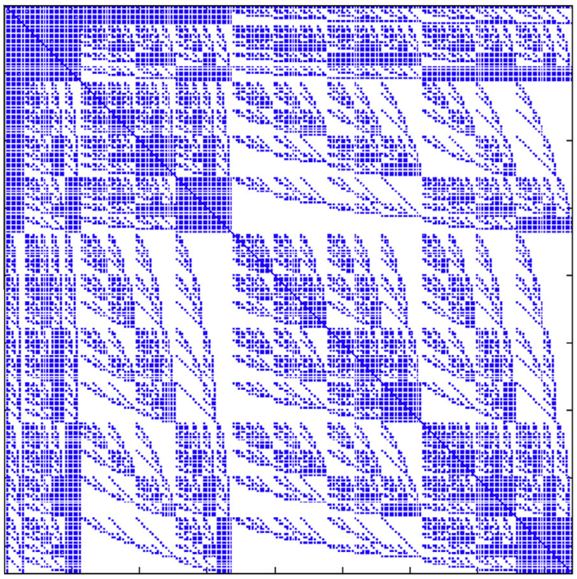


Fig. 4. Sparsity pattern of $\boldsymbol{K}$ for the non-Gaussian case ($M = 6, p = 4$).

while they are amenable to be efficiently implemented in high performance computing environments. A number of solution procedures for solving Eq. (41) has been proposed addressing small to medium problems. However, as the problem size grows, such a solution can become quite challenging due to the enormous memory and computational resources required. Solution techniques are based on either Gauss–Jacobi [8,24–26] or PCG [2,5,8,18,27,28,30,31,40] iterative solvers for addressing this problem. In this work two specialized preconditioners that take advantage of the properties of the augmented SSFEM linear systems are proposed which were found to be effective for both Gaussian and log-normal distributions.

Consider the preconditioning matrix for the case of Gaussian distribution of the form

$$\tilde{\mathbf{A}} = \begin{bmatrix} a_1\mathbf{K}_0 & 0 & \cdots & 0 \\ 0 & a_2\mathbf{K}_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n\mathbf{K}_0 \end{bmatrix} \tag{48}$$

where a_n are the coefficients as calculated from the PC bases (see Eq. (36)). For each evaluation of the preconditioned residual vector, the same $\mathbf{K}_0$ matrix needs to be "inverted" n times, as in the case of the MC-PCG-Skyline method. This matrix "inversion" is implemented as the solution of n linear systems. Since matrix $\tilde{\mathbf{A}}$ is block diagonal, the solution process can be pipelined as the successive solution of n linear systems with multiple right-hand sides. The PCG algorithm equipped with preconditioning matrix $\tilde{A}$ and utilizing the FETI method for solving the successive linear systems is proposed in [29]. A variant of this approach is tested in this work by employing PFETI for the solution of the repeated linear systems involved in the preconditioning steps of PCG. This algorithm is abbreviate as SSFEM-PCG-B for the solution of the augmented linear system that occurs from SSFEM.

The second preconditioner is based on the SSOR-type preconditioning matrix. In particular, the augmented matrix $\boldsymbol{K}$ is decomposed into the diagonal component $\mathbf{D}$ as it appears in Eq. (48) strictly lower triangular component $\mathbf{L}$ of the form:

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \mathbf{K}_{21} & 0 & \cdots & 0 \\ \vdots & \mathbf{K}_{m2} & \ddots & \vdots \\ \mathbf{K}_{n1} & \mathbf{K}_{n2} & \cdots & 0 \end{bmatrix} \tag{49}$$

Using this decomposition, the SSOR-type preconditioner is of the form:

$$\tilde{\mathbf{A}} = (\mathbf{D}-\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}-\mathbf{L}^T) \iff \tilde{\mathbf{A}}^{-1} = (\mathbf{D}-\mathbf{L}^T)^{-1}\mathbf{D}(\mathbf{D}-\mathbf{L})^{-1} \tag{50}$$

The evaluation of the preconditioned residual vector of the PCG algorithm is implemented as follows:

1. Solve:

$$(\mathbf{D}-\mathbf{L})\mathbf{z}_1^k = \mathbf{r}^k \tag{51}$$

2. Evaluate:

$$\mathbf{z}_2^k = \mathbf{D}\mathbf{z}_1 \tag{52}$$

3. Solve:

$$(\mathbf{D}-\mathbf{L}^T)\mathbf{z}^k = \mathbf{z}_2^k \tag{53}$$

The matrix of the linear system of Eq. (51) is lower triangular and its solution involves the implementation of a forward substitution algorithm in block form, as follows:

$$\begin{aligned}
&a_{11}\mathbf{K}_0\mathbf{x}_1 && = \mathbf{b}_1 \\
&a_{21}\mathbf{K}_1\mathbf{x}_2 + a_{22}\mathbf{K}_0\mathbf{x}_2 && = \mathbf{b}_2 \\
&\vdots \qquad\qquad \ddots \qquad \vdots && = \vdots \\
&a_{m1}\mathbf{K}_1\mathbf{x}_m + a_{m2}\mathbf{K}_2\mathbf{x}_2 + \cdots + a_{mm}\mathbf{K}_0\mathbf{x}_m && = \mathbf{b}_m
\end{aligned} \tag{54}$$

where $\mathbf{r}^k = [\mathbf{b}_1^T \cdots \mathbf{b}_m^T]$, $\mathbf{z}_1^k = [\mathbf{x}_1^T \cdots \mathbf{x}_m^T]$ and $a_{xy}\mathbf{K}_z$ are the various block matrices as they occur from the formulation of the SSFEM augmented system. The evaluation of these block equations are executed in a sequential manner and are implemented as the successive solution of the following linear systems:

$$\begin{aligned}
\mathbf{K}_0\mathbf{x}_1 &= \frac{\mathbf{b}_1}{a_{11}} \\
\mathbf{K}_0\mathbf{x}_2 &= \frac{\mathbf{b}_2 - a_{21}\mathbf{K}_1\mathbf{x}_1}{a_{22}} \\
\vdots &= \vdots \\
\mathbf{K}_0\mathbf{x}_m &= \frac{\mathbf{b}_m - \sum_{i=1}^{m-1} a_{mi}\mathbf{K}_i\mathbf{x}_i}{a_{mm}}
\end{aligned} \tag{55}$$

Similarly, the linear system of Eq. (53) is upper triangular and its solution involves the implementation of a backward substitution algorithm in block form, as follows:

$$\begin{aligned}
a_{11}\mathbf{K}_0\mathbf{y}_1 + a_{12}\mathbf{K}_2\mathbf{y}_2 + \cdots + a_{1m}\mathbf{K}_m\mathbf{y}_m &= \mathbf{c}_1 \\
a_{22}\mathbf{K}_2\mathbf{y}_2 + \cdots + a_{2m}\mathbf{K}_m\mathbf{y}_m &= \mathbf{c}_2 \\
\vdots &= \vdots \\
a_{mm}\mathbf{K}_0\mathbf{y}_m &= \mathbf{c}_m
\end{aligned} \tag{56}$$

where $\mathbf{z}_2^k = \left[\mathbf{c}_1^T \cdots \mathbf{c}_m^T\right]^T$, $\mathbf{z}^k = \left[\mathbf{y}_1^T \cdots \mathbf{y}_m^T\right]^T$. The evaluation of these block equations are executed as the successive solution of the following linear systems:

$$\begin{aligned}
\mathbf{K}_0\mathbf{y}_m &= \frac{\mathbf{c}_m}{a_{mm}} \\
\mathbf{K}_0\mathbf{y}_{m-1} &= \frac{\mathbf{c}_{m-1} - a_{m-1,m-1}\mathbf{K}_{m-1}\mathbf{y}_{m-1}}{a_{m-1,m-1}} \\
\vdots &= \vdots \\
\mathbf{K}_0\mathbf{y}_1 &= \frac{\mathbf{c}_1 - \sum_{i=1}^{m-1} a_{1i}\mathbf{K}_i\mathbf{y}_i}{a_{11}}
\end{aligned} \tag{57}$$

The PCG algorithm equipped with the above SSOR preconditioner and utilizing the PFETI method for solving the linear systems occurring at each step of the aforementioned forward and backward substitutions, constitutes the SSFEM-PCG-S method for the solution of the augmented linear system that occurs from SSFEM. It is worth noting that, in contrast to the point SSOR preconditioner applied within an iterative solver, the evaluation of the preconditioned residual vector of the SSFEM-PCG-S method is parallel and scalable due to the fact that all block matrices that take part in the matrix–vector multiplication operations of the forward and backward substitutions are already decomposed into subdomains. This means that each operation at each step of these forward and backward substitutions, including both the matrix–vector multiplications and the solution process, are carried out in parallel, exhibiting the scalability of the PFETI method [36].

3.3. Implementing the A matrix–vector product

The augmented systems that are generated from the application of SSFEM involve large coefficient matrices that feature a block form. Each block is comprised of a linear combination of stiffness matrix realizations that have identical sparsity pattern and bandwidth with the deterministic matrix $\mathbf{K}_0$.

Both the SSFEM-PCG-B and SSFEM-PCG-S methods need the computation of the $\tilde{A}$ matrix–vector product at each iteration i. In this work, this computation is performed as a series of matrix–vector multiplications where the resulting vectors are linearly combined in order to form the final vector. This is accomplished by forming and storing all the linear combination of the aforementioned stiffness matrix realizations in a symmetric sparse storage format in order to minimize storage requirements. Moreover, matrix structure identity is exploited by generating and storing only one data structure for the location of non-zero elements, reducing memory storage requirements and improving cache locality (see Appendix A). Due to the nature of the augmented stiffness matrix as it occurs for log-normal input fields, a caching scheme has been developed for the log-normal case in this work, in order to minimize the computational burden of this evaluation, which is described subsequently.

3.4. A caching scheme for the log-normal case

Stochastic problems modeled with input random fields featuring a log-normal distribution, produce augmented stiffness matrices that have three major differences compared to those produced with a Gaussian distribution input field. The coefficient matrix sparsity in a log-normal case is:

i. Much denser compared to the Gaussian ones, as shown in Figs. 2 and 4 and in Table 2.
ii. Each block position might be the result of a linear combination of the stochastic matrices, while in Gaussian matrices each block position is occupied by one stochastic matrix multiplied by an integer coefficient.
iii. A number of block diagonal matrices is composed of a linear combination of the deterministic matrix and some of the stochastic matrices, while in Gaussian matrices each block position is occupied by the deterministic matrix multiplied by an integer coefficient.

Table 2 presents an indication of how many block positions are occupied for each p from 1 to 6 and the relative sparsity (0% is fully dense) for $M = 4$ and the computational effort ratio of log-normal vs Gaussian formulation for each PCG iteration. These differences affect the performance of the solvers greatly, with respect to the number of iterations necessary for convergence and the amount of computations required to perform each PCG iteration. The computational effort is further magnified due to a more cumbersome implementation of the A matrix–vector product computations that are needed in each iteration of the SSFEM-PCG-B and SSFEM-PCG-S methods.

As shown in the previous section, the augmented matrix is never formulated as a whole and each product is being evaluated by multiplying every block matrix with its corresponding block vector and accumulating the partial results to the corresponding position of the resulting vector. This means that if a block position of the coefficient matrix is comprised of a linear combination of n terms, n matrix–vector products must be evaluated, for the complete calculation of the resulting vector, while for a Gaussian field only one matrix–vector product for each block position needs to be evaluated.

In order to alleviate this additional computational effort, a caching scheme has been applied where each unique linear combination that occurs in every block position is being pre-calculated and stored in order to

Table 2
Sparsity of the resulting augmented stochastic matrix for KL expansion of order $M = 4$ and for various polynomial chaos orders p and the computational effort ratio of log-normal vs Gaussian formulation for each PCG iteration.

p	Block size	Gaussian		Log-normal		Computational effort ratio
		Filled blocks	Sparsity (%)	Filled blocks	Sparsity (%)	
1	5×5	13	48	13	48	1.00
2	15×15	45	80	135	40	1.05
3	35×35	125	97.14	725	40.81	1.35
4	70×70	285	94.18	3090	36.94	2.48
5	126×126	565	96.44	10158	36.02	5.41
6	210×210	1013	97.70	29448	33.22	11.82

reduce the computation cost of each A matrix–vector evaluation at each iteration of the SSFEM-PCG-B and SSFEM-PCG-S methods.

In order to demonstrate this technique, we consider again the case of $M = 2,\ p = 2$ for the log-normal case as depicted in Fig.3, where the linear combination $\mathbf{K}_0 + 2\mathbf{K}_3$ is found two times. At the uncached case, we would need to perform 4 matrix–vector operations (the augmented stiffness matrix is symmetric so we need to perform 2 matrix vector operations twice) and 2 linear scaling operations (it is the operation of multiplying a vector or matrix with a scalar and is performed also twice). However, for the cached case only 2 matrix–vector operations need to be performed since the linear combination is stored as a separate matrix.

Table 3 compares the amount of matrix–vector products needed for a KL expansion per PCG iteration, of order $M = 4$ and a PC order varying from $p = 4$ to $p = 6$.

It has to be mentioned that this caching scheme requires one to two orders of magnitude more computer memory resources for storing the corresponding stiffness matrices but it can offer significant performance benefits as it is shown in the numerical examples section. If memory requirements of this caching scheme cannot be met, only a portion of the re-occurring matrices can be cached in order to decrease memory resources consumption, at the benefit of speedup.

3.5. A full block preconditioning scheme for the log-normal case

The existence of a number of linear combinations of the deterministic matrix with stochastic ones at the block diagonal part of the augmented stiffness matrix, as shown in Fig. 3 for the log-normal case, can really deteriorate the convergence rate of both the preconditioner of the SSFEM-PCG-B method and the preconditioner of the SSFEM-PCG-S method, especially at large input covariances where the magnitude of the stochastic matrices is comparable to the magnitude of the deterministic one.

In order to overcome this deficiency, a MC-PCG-PFETI solver is used instead of a regular PFETI solver, in order to evaluate the preconditioned residual of each iteration of the SSFEM-PCG-B solver as well as to solve the linear systems Eqs. (54) and (57) of the SSFEM-PCG-S solver. Thus the MC-PCG-PFETI solver takes into account the full linear combination of the block diagonal, enhancing the convergence rate of the SSFEM-PCG-B and SSFEM-PCG-S solvers, instead of taking into account only K_0 matrix which gives an approximation to the preconditioned residual.

As in the case of the Monte Carlo simulations, the repeated solutions required for the preconditioning step of the MC-PCG-PFETI algorithm can be treated as problems with multiple right-hand sides, since the entries in the residual vector are updated at each PCG iteration k of each block diagonal part of the coefficient matrix. In order to illustrate this technique, we consider once more the augmented stiffness matrix of Fig. 3. For the SSFEM-PCG-B method the preconditioner is of the form:

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{K}_0 & & & & & \\ & \mathbf{K}_0 + 2\mathbf{K}_3 & & & & \\ & & \mathbf{K}_0 + 2\mathbf{K}_5 & & & \\ & & & \mathbf{K}_0 + 8\mathbf{K}_3 & & \\ & & & & \mathbf{K}_0 + 2\mathbf{K}_3 & \\ & & & & & 2\mathbf{K}_0 + 8\mathbf{K}_5 \end{bmatrix} \tag{58}$$

Table 3
Normal and cached schemes.

p	Non-zero block submatrices	Matrices stored		Total MV multipliers	
		Normal	Cached	Normal	Cached
4	3.090	70	710	4.937	3.090
5	10.158	126	2109	19.542	10.158
6	29.448	210	6064	70.952	29.448

This means that for each iteration k of the SSFEM-PCG-B method, the preconditioned residual vector involves the solution of the following linear systems:

$$\begin{aligned}
&\mathbf{K}_0\mathbf{z}_0^k = \mathbf{r}_0^k \\
&(\mathbf{K}_0 + 2\mathbf{K}_3)\mathbf{z}_1^k = \mathbf{r}_1^k \\
&(\mathbf{K}_0 + 2\mathbf{K}_5)\mathbf{z}_2^k = \mathbf{r}_2^k \\
&(\mathbf{K}_0 + 8\mathbf{K}_3)\mathbf{z}_3^k = \mathbf{r}_3^k \\
&(\mathbf{K}_0 + 2\mathbf{K}_3)\mathbf{z}_4^k = \mathbf{r}_4^k \\
&(2\mathbf{K}_0 + 8\mathbf{K}_5)\mathbf{z}_5^k = \mathbf{r}_5^k
\end{aligned} \tag{59}$$

with

$$\mathbf{z}^k = \begin{bmatrix} \mathbf{z}_0^{k^T} & \mathbf{z}_1^{k^T} & \mathbf{z}_2^{k^T} & \mathbf{z}_3^{k^T} & \mathbf{z}_4^{k^T} & \mathbf{z}_5^{k^T} \end{bmatrix}^T$$

and

$$\mathbf{r}^k = \begin{bmatrix} \mathbf{r}_0^{k^T} & \mathbf{r}_1^{k^T} & \mathbf{r}_2^{k^T} & \mathbf{r}_3^{k^T} & \mathbf{r}_4^{k^T} & \mathbf{r}_5^{k^T} \end{bmatrix}^T$$

The SSFEM-PCG-B algorithm equipped with preconditioner $\tilde{\mathbf{A}}$ of Eq. (58) and utilizing the MC-PCG-PFETI method for solving the linear systems at each step of Eq. (59) constitutes the SSFEM-PCG-BF variant for the solution of the augmented linear system that occurs from the SSFEM.

In the same fashion, the SSFEM-PCG-S method requires the successive solution of systems (51) and (53) which for the case considered in connection to Fig. 3 and Eqs. (28) and (50) are of the form:

$$\begin{aligned}
&\mathbf{K}_0\mathbf{x}_1 = \mathbf{b}_1 \\
&(\mathbf{K}_0 + 2\mathbf{K}_3)\mathbf{x}_2 = \mathbf{b}_2 \\
&(\mathbf{K}_0 + 2\mathbf{K}_5)\mathbf{x}_3 = \mathbf{b}_3 \\
&(\mathbf{K}_0 + 8\mathbf{K}_3)\mathbf{x}_4 = \mathbf{b}_4 \\
&(\mathbf{K}_0 + 2\mathbf{K}_3)\mathbf{x}_5 = \mathbf{b}_5 \\
&(2\mathbf{K}_0 + 8\mathbf{K}_5)\mathbf{x}_6 = \mathbf{b}_6
\end{aligned} \tag{60}$$

and

$$\begin{aligned}
&(2\mathbf{K}_0 + 8\mathbf{K}_5)\mathbf{y}_6 = \mathbf{c}_6 \\
&(\mathbf{K}_0 + 2\mathbf{K}_3)\mathbf{y}_5 = \mathbf{c}_5 \\
&(\mathbf{K}_0 + 8\mathbf{K}_3)\mathbf{y}_4 = \mathbf{c}_4 \\
&(\mathbf{K}_0 + 2\mathbf{K}_5)\mathbf{y}_3 = \mathbf{c}_3 \\
&(\mathbf{K}_0 + 2\mathbf{K}_3)\mathbf{y}_2 = \mathbf{c}_2 \\
&\mathbf{K}_0\mathbf{y}_1 = \mathbf{c}_1
\end{aligned} \tag{61}$$

The PCG algorithm equipped with the block SSOR preconditioner of the SSFEM-PCG-S method and utilizing the MC-PCG-Skyline method for solving the linear systems occurring at the preconditioned residual vector evaluation, constitutes the SSFEM-PCG-SF variant for the solution of the augmented linear system that occurs from the SSFEM.

4. Numerical test

Numerical tests are performed for stochastic finite element and reliability analysis implementing the proposed versions of SSFEM and MC. For the case of Gaussian fields we will examine the performance of

SSFEM-PCG-B, and SSFEM-PCG-S. For log-normal fields we will test the performance of SSFEM-PCG-B, SSFEM-PCG-BF, SSFEM-PCG-S, SSFEM-PCG-SF and their variants with caching SSFEM-PCG-BC, SSFEM-PCG-BFC, SSFEM-PCG-SC, SSFEM-PCG-SFC, respectively. For the case of the MC method, we will examine the performance of the MC-PCG-Skyline, MC-PCG-FETI and MC-PCG-PFETI solvers. The computer platform used is an Intel Core i7 X980 with 6 physical cores at 3.33 GHz with 24 GB of RAM.

In order to assess the computational efficiency of the MC and SSFEM methods for the analysis of systems with uncertain properties, a soil cube of $10 \times 10 \times 20$ m under load in the center of its upper surface due to a large footing was considered, resulting to a finite element mesh of 10k dof approximately. This mesh is decomposed into 16 subdomains featuring a cubic aspect ratio each, as shown in Fig. 5.

A multi-parametric study has been carried out first, considering both Gaussian and log-normal stochastic fields. One dimensional stochastic fields are used to describe the spatial variation of the system's modulus of elasticity E around its mean as $E = E_0 \cdot (1 + f(\mathbf{x}))$, where E_0 is the mean value of E and $f(\mathbf{x})$ a zero mean homogeneous stochastic field with standard deviation σ_E. The covariance function of the random field $f(\mathbf{x})$ is assumed to be exponential:

$$C(\mathbf{x}_1, \mathbf{x}_2) = \sigma_E^2 e^{-\frac{|\Delta \mathbf{x}|}{b}} \tag{62}$$

where $\Delta\mathbf{x} = \mathbf{x}_2 - \mathbf{x}_1$. Three test cases regarding coefficients σ_E are examined: (a) $\sigma_E = 15\%$ (Gaussian), (b) $\sigma_E = 30\%$ (log-normal) and (c) $\sigma_E = 80\%$ (log-normal), which is a typical value for the standard deviation in soil mechanics problems. Moreover, four correlation length values are assumed: (a) $\mathrm{b} = 0.1a$, (b) $\mathrm{b} = 1a$, (c) $\mathrm{b} = 10a$ and (d) $\mathrm{b} = 100a$, with a being the height of the cube. For all these test cases, two separate problems are addressed: evaluation of the second moments of the response field and a reliability analysis with 0.1% probability of failure. Setting $a = 20$ m, the correlation lengths that were examined for this example were 2 m, 20 m, 200 m and 2000 m.

4.1. Solver assessment procedure

In order to set an objective basis for assessing the computational performance of the numerical algorithms discussed, a parametric study was conducted, regarding different values for standard deviation σ_E and correlation length b. For the computation of the second moments of the response field, the following procedure was followed:

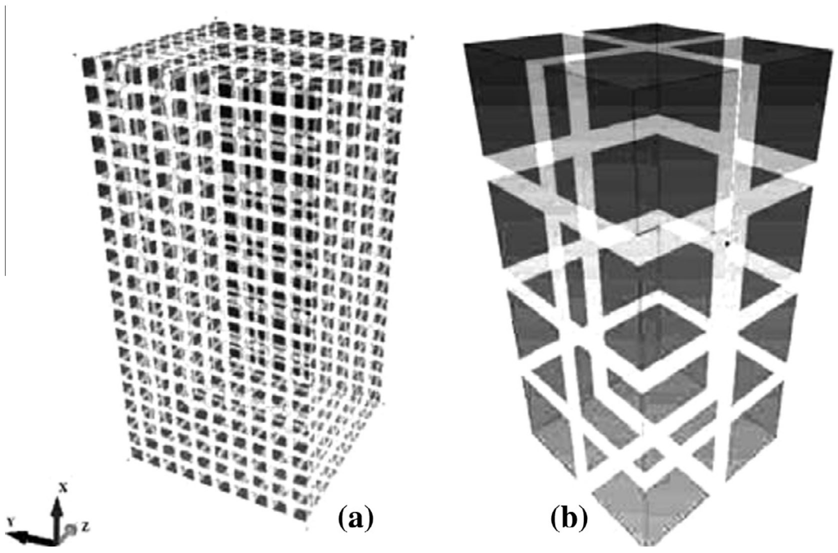


Fig. 5. Domain decomposition of a quarter of the deterministic soil problem with 10k dof. (a) Element mesh (b) subdomain mesh.

Step 1. A series of Monte Carlo analyses of 100k simulations was carried out, using $M = 1$ as the order of the KL expansion, in order to estimate the necessary number of simulations for a convergence error of less than 1% for each value of σ_E and b examined. This error is computed as the normalized difference of the $COV(\%)$ at each simulation with respect to the $COV(\%)$ computed at the end of the 100k simulations.

Step 2. Assuming that the convergence behavior of the previous step remains invariant for increasing M, another series of Monte Carlo analyses was carried out, in the range of $M = 2$ to $M = 12$, in order to estimate the appropriate order of the KL expansion for a convergence error of less than 1%. In this case an "exact" solution was assumed at $M = 12$ in order to compute the relative error (%) for different M.

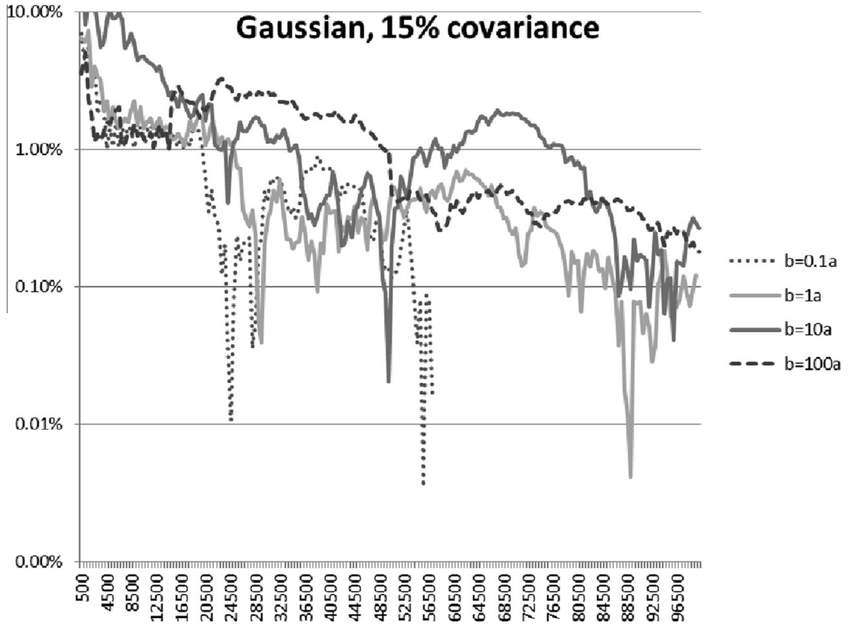


Fig. 6. Step 1: $COV(\%)$ convergence error of MC for the Gaussian field with $\sigma_E = 15\%$.

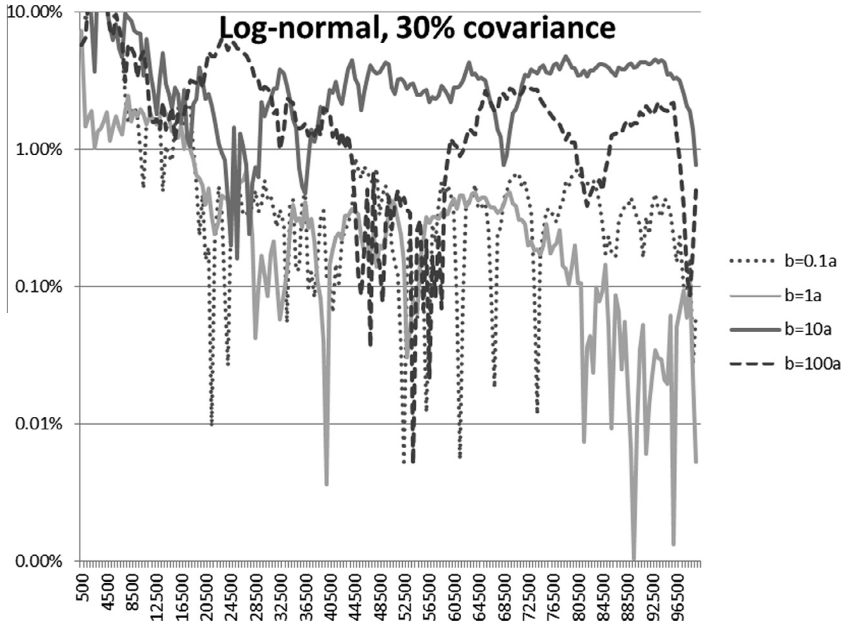


Fig. 7. Step 1: $COV(\%)$ convergence error of MC for the log-normal field with $\sigma_E = 30\%$.

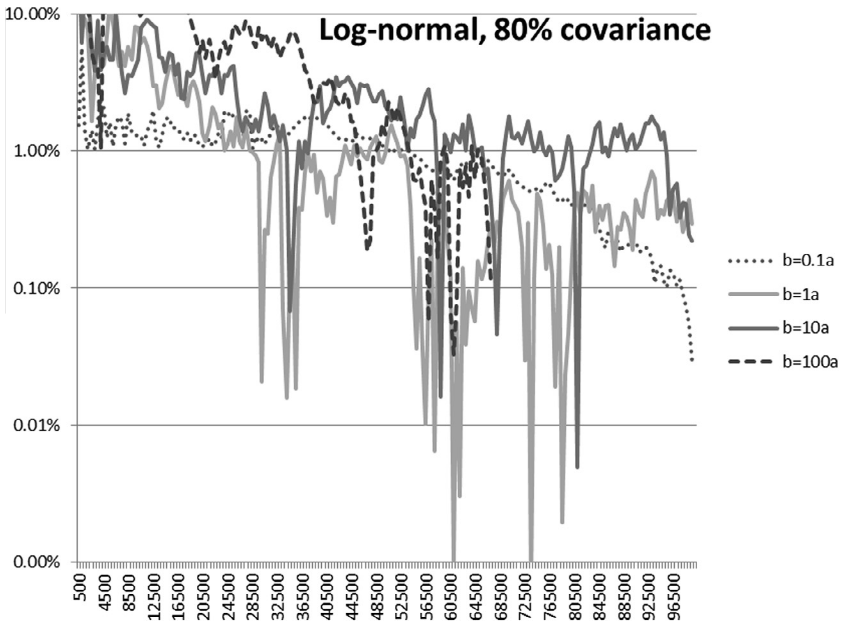


Fig. 8. Step 1: $COV(\%)$ convergence error of MC for the log-normal field with $\sigma_E = 80\%$.

Table 4
Required number of MC simulations for achieving a COV error less than 1%.

Correlation length b	$\sigma_E = 15\%$	$\sigma_E = 30\%$	$\sigma_E = 80\%$
0.1a	20.000	10.000	53.000
1a	25.000	18.000	28.000
10a	23.000	23.000	34.000
100a	50.000	43.000	45.000

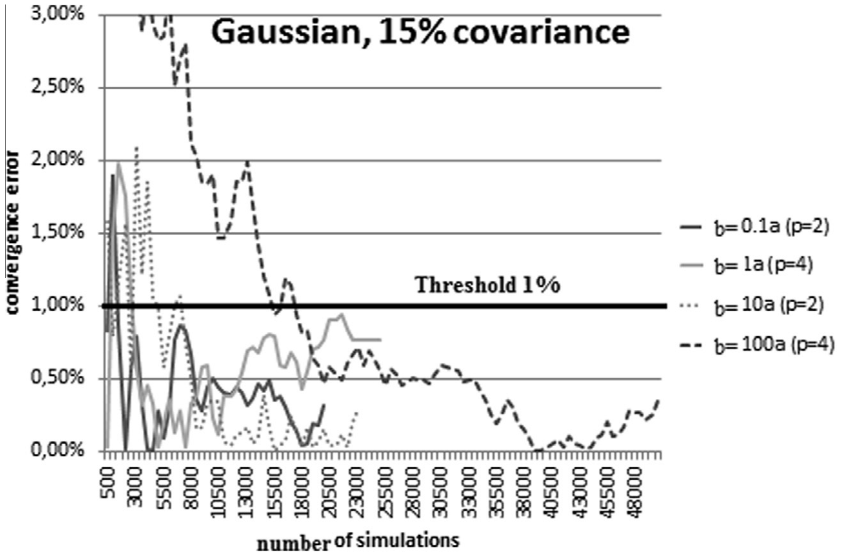


Fig. 9. Step 3: $COV(\%)$ convergence error of the SSFEM for the Gaussian field with $\sigma_E = 15\%$ and $p = 2, 4$.

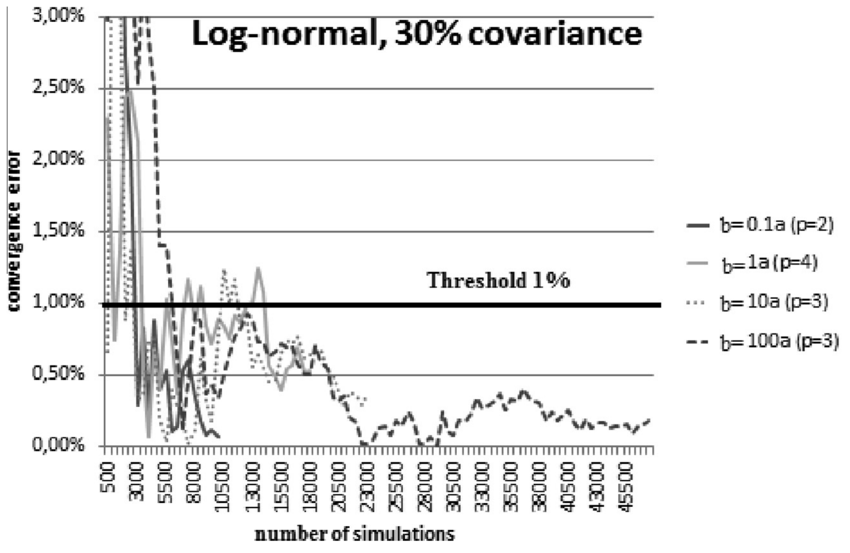


Fig. 10. Step 3: $COV(\%)$ convergence error of the SSFEM for the log-normal field with $\sigma_E = 30\%$ and $p = 2, 3, 4$.

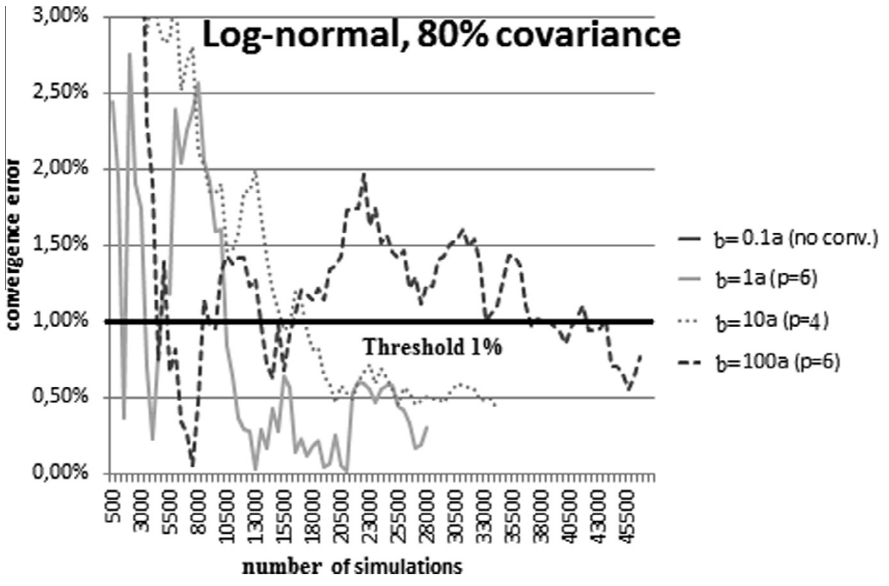


Fig. 11. Step 3: $COV(\%)$ convergence error of the SSFEM for the log-normal field with $\sigma_E = 80\%$ and $p = 2, 3, 4$.

Table 5
Step 2: $COV(\%)$ convergence errors for the various KL expansion orders.

Correlation length b	$\sigma_E = 15\%$		$\sigma_E = 30\%$		$\sigma_E = 80\%$	
	M	Error (%)	M	Error (%)	M	Error (%)
0.1a	12	“exact”	10	0.43	4	0.75
1a	6	0.93	4	0.75	4	0.57
10a	2	0.36	2	0.85	4	0.26
100a	2	0.48	2	0.53	4	0.96

Step 3. Using the results of step 2, the same procedure as in step 2 was carried out performing SSFEM analyses, in order to estimate the appropriate order of the PC expansion required for convergence to the corresponding MC results.

Step 4. For the case of reliability analysis with 0.1% target probability of failure, the order of the PC expansion is being modified, with respect to step 3 (convergence in *COV*%), in order to reach a convergence error in the estimation of the probability of failure of less than 10%, compared to the corresponding MC results. The number of simulations for both MC and SSFEM is in this case 100k.

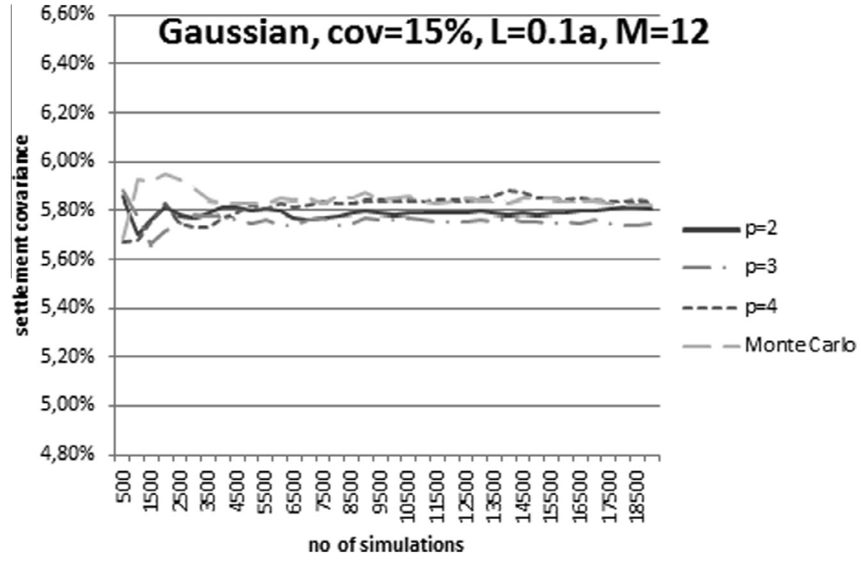


Fig. 12. Settlement covariance for $\sigma_E = 15\%$, correlation length 2 m, and $M = 12$ for MC and SSFEM.

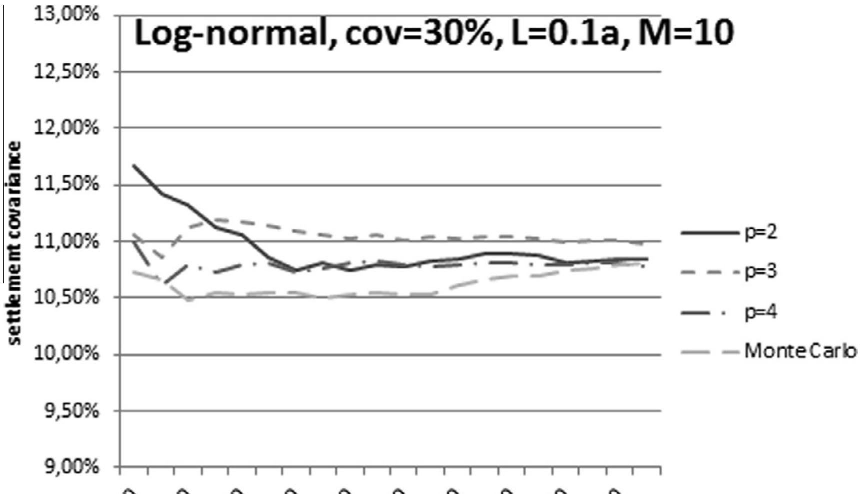


Fig. 13. Settlement covariance for $\sigma_E = 30\%$, correlation length 2 m, and $M = 10$ for MC and SSFEM.

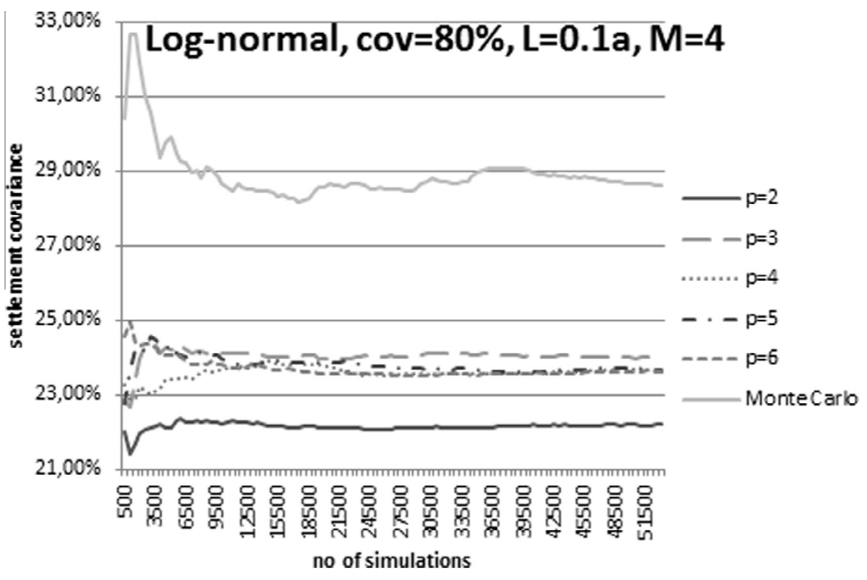


Fig. 14. Settlement covariance for $\sigma_E = 80\%$, correlation length 2 m, and $M = 4$ for MC and SSFEM.

Table 6
Convergence errors for the SSFEM.

Correlation length b	$\sigma_E = 15\%$		$\sigma_E = 30\%$		$\sigma_E = 80\%$	
	p	Error (%)	p	Error (%)	p	Error (%)
0.1a	2	0.23	2	0.07	6	30.00
1a	4	0.09	4	0.69	6	0.52
10a	2	0.03	3	0.36	4	0.68
100a	4	0.36	3	0.74	6	0.88

Table 7
Settlements with 0.1% probability of failure.

Correlation length b	MC		
	$\sigma_E = 15\%$	$\sigma_E = 30\%$	$\sigma_E = 80\%$
0.1a	0.019405	0.023468	0.076088
1a	0.024796	0.032947	0.118410
10a	0.028214	0.038950	0.166543
100a	0.029439	0.039841	0.168306

Table 8
Probability of failure as computed by SSFEM for the settlements of Table 7 for the limit states.

Correlation length b	SSFEM		
	$\sigma_E = 15\%$	$\sigma_E = 30\%$	$\sigma_E = 80\%$
0.1a	0.06	0.04	–
1a	0.09	0.09	0.07
10a	0.03	0.09	0.01
100a	0.07	0.10	0.10

4.2. Computation of the second moments of the response field

Figs. 6–8 show the convergence error for each field as per step 1 of the assessment procedure. Based on these figures, the number of simulations necessary for evaluating the second moments of the response field are shown in Table 4.

Following step 2, Figs. 9–11 show the convergence error for each field as per step 3 of the assessment procedure for the selection of PC expansion order (p) required for the SSFEM to converge at an error less than

Table 9
Probability of failure as computed by SSFEM for the settlements of Table 7 and the necessary PC order (values marked in bold correspond to analyses that needed an increase of the PC order expansion).

Correlation length b	$\sigma_E = 15\%$		$\sigma_E = 30\%$		$\sigma_E = 80\%$	
	p	Prob	p	Prob	p	Prob
0.1a	**4**	**0.09**	**5**	**0.09**	–	–
1a	4	0.09	4	0.09	**7**	0.09
10a	**4**	**0.09**	3	0.09	**8**	**0.10**
100a	**6**	**0.09**	3	0.10	6	0.10

Table 10
Performance of the various MC-PCG-Skyline variants for the MC for evaluating the second moments of the response field for $\sigma_E = 15\%$ in sequential and parallel implementation.

$\sigma_E = 15\%$	Correlation length b	0.1a	1a	10a	100a
	MC simulations	20.000	25.000	23.000	50.000
	PCG iterations	184.398	196.875	114.715	144.592
MC-PCG-Skyline	Time (s)-sequential	31.759	85.930	163.202	224.143
	Time (s)-parallel	4.670	12.637	24.000	32.962
MC-PCG-FETI	FETI iterations	455.016 (2.950.368)	134.198 (3.150.000)	22.015 (1.835.440)	39.060 (2.313.472)
	Time (s)-sequential	48.001	36.752	21.203	39.827
	Time (s)-parallel	7.059	5.405	3.118	5.857
MC-PCG-PFETI	PFETI iterations	425.281 (2.950.368)	124.133 (3.150.000)	20.157 (1.835.440)	35.761 (2.313.472)
	Time (s)-sequential	36.771	28.076	16.443	30.590
	Time (s)-parallel	5.407	4.129	2.418	4.498

Table 11
Performance of the various MC-PCG-Skyline variants for the MC for evaluating the second moments of the response field for $\sigma_E = 30\%$ in sequential and parallel implementation.

$\sigma_E = 30\%$	Correlation length b	0.1a	1a	10a	100a
	MC simulations	10.000	18.000	23.000	43.000
	PCG iterations	110.100	221.531	114.541	153.825
MC-PCG-Skyline	Time (s)-sequential	18.922	105.391	161.203	241.235
	Time (s)-parallel	2.783	15.499	23.706	35.476
MC-PCG-FETI	FETI iterations	314.475 (1.761.600)	107.198 (3.544.496)	23.442 (1.832.656)	37.087 (2.461.200)
	Time (s)-sequential	33.053	32.437	22.189	38.625
	Time (s)-parallel	4.861	4.770	3.263	5.680
MC-PCG-PFETI	PFETI iterations	294.235 (1.761.600)	99.478 (3.544.496)	21.777 (1.832.656)	34.375 (2.461.200)
	Time (s)-sequential	25.337	24.956	17.393	30.080
	Time (s)-parallel	3.726	3.670	2.558	4.423

1% using the KL expansion orders M shown in Table 5. This relative error is computed with respect to the corresponding MC simulations with the same parameter M.

Figs. 12–14 depict some indicative graphs of the convergence behavior of the SSFEM in specific cases. Table 6 summarizes the convergence of the SSFEM (relative error %) with respect to MC, for all cases considered.

It is worth noting that for the case of $b = 0.1a$, the SSFEM failed to provide a solution within the acceptable error margin when compared to the MC solution. While increasing the p-order of the PC expansion, the SFFEM method was asymptotically converging to a solution which exhibited a 30% error when compared to the corresponding Monte Carlo solution.

4.3. Reliability analysis

Utilizing the values of M obtained at step 2 of the solver assessment procedure for the computation of the second moments of the response field, we performed reliability analysis on the same test problem. Table 7 shows the settlement values which correspond to a probability of failure of 0.1%, as estimated by MC with

Table 12
Performance of the various MC-PCG-Skyline variants for the MC for evaluating the second moments of the response field for $\sigma_E = 80\%$ in sequential and parallel implementation.

$\sigma_E = 80\%$	Correlation length	$0.1a$	$1a$	$10a$	$100a$
	MC simulations	53.000	28.000	34.000	45.000
	PCG iterations	1.193.825	695.100	272.340	253.350
MC-PCG-Skyline	Time (s)-sequential	205.082	68.030	370.320	400.378
	Time (s) -parallel	30.159	10.004	54.459	58.879
MC-PCG-FETI	FETI iterations	3.413.444 (19.101.200)	1.624.400 (11.121.600)	72.507 (4.357.440)	56.799 (4.053.600)
	Time (s)-sequential	358.806	97.472	65.108	60.159
	Time (s)-parallel	52.765	14.334	9.575	8.847
MC-PCG-PFETI	PFETI iterations	3.265.860 (19.101.200)	1.530.760 (11.121.600)	67.320 (4.357.440)	52.650 (4.053.600)
	Time (s)-sequential	281.182	75.286	50.653	46.932
	Time (s) -parallel	41.350	11.071	7.449	6.902

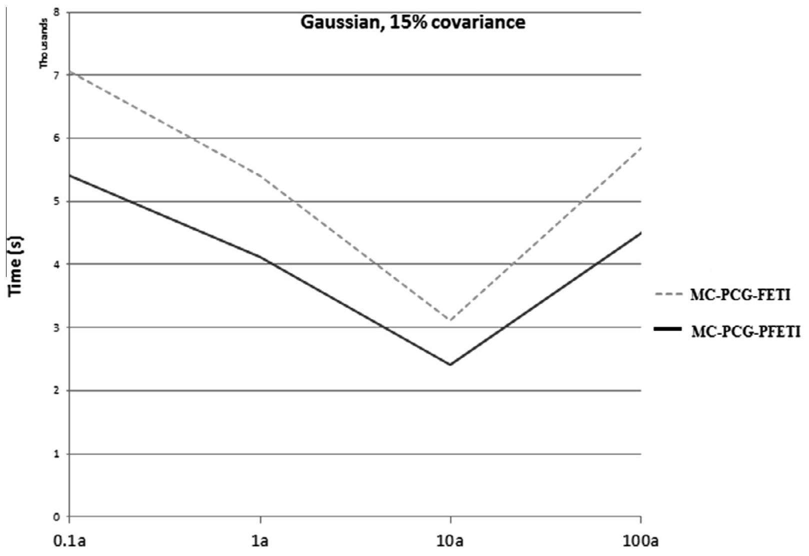


Fig. 15. Performance of the MC-PCG-PFETI and MC-PCG-FETI for Gaussian $\sigma_E = 0.15\%$.

100k simulations, for various stochastic parameters (σ_E and b) considered. Table 8 shows the probability of failure for the corresponding limit state settlements of Table 7 using SSFEM with the KL and PC order expansions used for the second order moments analysis as shown in Tables 5 and 6, respectively. The settlement values of Table 7 are used as reference values, i.e. as the limit states that correspond to a probability of failure 0.1% for all cases considered.

While Table 9 shows the same probability of failure with the PC order expansion needed to reach almost the same accuracy with the "reference" MC solution. Values marked in bold correspond to analyses that needed an increase of the PC order expansion.

For the case of $b = 0.1a$ and $\sigma_E = 80\%$, a series of analyses were performed with various expansion orders M and p, going up to $M = 12$ and $p = 6$. However, SSFEM failed to converge to an acceptable solution resulting to a minimum convergence error of 20%.

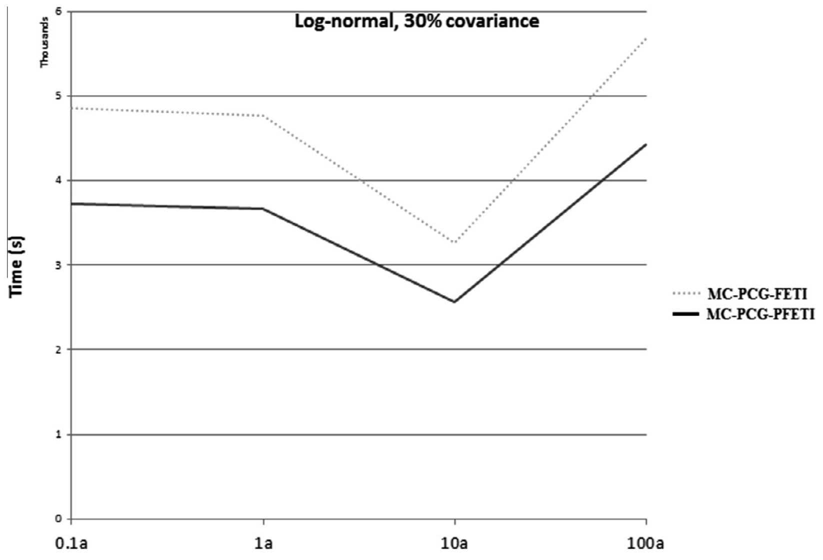


Fig. 16. Performance of the MC-PCG-PFETI and MC-PCG-FETI for log-normal $\sigma_E = 0.30\%$.

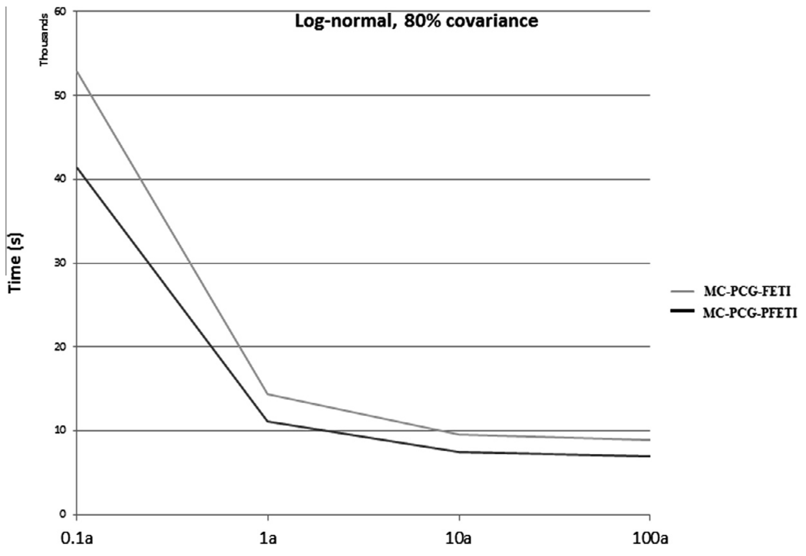


Fig. 17. Performance of the MC-PCG-PFETI and MC-PCG-FETI for log-normal $\sigma_E = 0.80\%$.

4.4. Performance of the proposed solution procedures

Using all previous numerical data (number of simulations, KL expansion order and PC expansion order), a series of numerical tests were performed in order to assess the performance of the various solution techniques discussed and proposed in this work. For all cases considered the normalized solution accuracy was set to 10^{-7} while for the computation of the preconditioned residual vector, the required accuracy was set to 10^{-3}. Also, the first 600 search vectors were used, achieving a reduction between 90 and 95% of the required iterations for each PFETI solutions when compared to the non re-orthogonalization procedure.

Tables 10–12 show the performance of proposed MC-PCG-PFETI solver for the evaluation of the second order moments of the response field using the MC method, in comparison to MC-PCG-Skyline and MC-PCG-FETI and are visually depicted in Figs. 15–17. The PFETI and FETI iterations correspond to the sum of the PFETI and FETI iterations needed for all the MC simulations using the A-orthogonalization technique, while in parentheses the corresponding PFETI and FETI iterations without A-orthogonalization

Table 13
Performance metrics for the Gaussian case ($\sigma_e = 15\%$ covariance).

Correlation length b		0.1a	1a	10a	100a
Gaussian $\sigma_E = 15\%$					
SSFEM-PCG-B	MC simulations	20.000	25.000	23.000	50.000
	PCG iterations	7	10	5	8
	PFETI iterations	650	702	96	123
	Total time (s)-sequential	2.417	3.339	105	168
	Total time (s)-parallel	422	586	18	29
SSFEM-PCG-S	PCG iterations	3	4	3	4
	PFETI iterations	377	269	74	74
	Total time (s)-sequential	1.026	1.178	77	100
	Total time (s)-parallel	179	204	13	17

Table 14
Performance metrics for the log-normal case ($\sigma_E = 30\%$ covariance).

Correlation length b		0.1a	1a	10a	100a
Log-normal 30%					
SSFEM-PCG-B	MC simulations	10.000	18.000	23.000	43.000
	PCG iterations	10	15	12	11
	PFETI iterations	551	367	120	98
	Total time (s)-sequential	2.786	6.568	244	256
	Total time (s)-parallel	488	1.149	43	43
SSFEM-PCG-BF	PCG iterations	10	17	13	11
	PFETI iterations	577	328	132	91
	Total time (s)-sequential	2.930	7.191	260	273
	Total time (s)-parallel	516	1.241	44	46
SSFEM-PCG-S	PCG iterations	4	5	4	4
	PFETI iterations	403	260	100	79
	Total time (s)-sequential	1.872	4.317	172	142
	Total time (s)-parallel	330	745	31	24
SSFEM-PCG-SF	PCG iterations	4	5	5	4
	PFETI iterations	577	263	108	85
	Total time (s)-sequential	2.728	4.347	200	150
	Total time (s)-parallel	480	752	35	25

are given. These numbers show a drastic decrease of iterations ranging from one to two orders of magnitude as a result of the A-orthogonalization procedure. Moreover, from these tables, it is evident that the PFETI variant outperforms the FETI one in all tests, showing a 1.25× speedup. This performance increase occurs for two reasons: (i) PFETI needs ∼10% less iterations when compared to FETI. (ii) The cost for each reorthogonalization of the PFETI method is about 35% less when compared to the FETI method. This stems from the fact that the interface problem of the PFETI method is based on the boundary dof of each subdomain while the interface problem of the FETI method is based on the lagrange multipliers which, due to the existence of a considerable number of subdomains crosspoints, are significantly larger in quantity than the boundary dof.

The Skyline variant seems to be more efficient for $b = 0.1a$ but this happens due to the relatively small size of the deterministic model. For large deterministic models, the Skyline variant is outperformed by domain decomposition methods, particularly in massively parallel computation environments.

Tables 13–15 depict the performance of the proposed SSFEM solution methods. Table 13 shows the performance of SSFEM methods, using the information gathered from steps 1–3 with respect to the necessary number of simulations, KL expansion order (M) and PC expansion order (p) for the Gaussian case (see Tables 4–6).

For the Gaussian case, only the SSFEM-PCG-B and SSFEM-PCG-S solvers are used for the SSFEM. It can be seen that the proposed SSFEM-PCG-S outperforms SSFEM-PCG-B achieving a 2.8× speedup with respect to the SSFEM-PCG-B.

Table 14 shows performance metrics for the log-normal case with 30% covariance, where all solver variants are implemented within SSFEM. As in the case of the Gaussian field SSFEM-PCG-S outperforms SSFEM-PCG-B achieving a 2.3× speedup when compared to the SSFEM-PCG-B.

Table 15
Performance metrics for the log-normal case ($\sigma_E = 80\%$ covariance).

Correlation length b		$0.1a$	$1a$	$10a$	$100a$
Log-normal 80%					
SSFEM-PCG-B	MC simulations	53.000	28.000	34.000	45.000
	PCG iterations	48	89	33	58
	PFETI iterations	685	1.010	523	584
	Total time (s)-sequential	266.702	272.836	14.171	321.161
	Total time (s)-parallel	46.799	49.175	2.447	55.380
	Total time cached (s)-sequential	112.142	114.721	9.443	134.402
	Total time cached (s)-parallel	1988	2079	131	2393
SSFEM-PCG-BF	PCG iterations	47	117	47	82
	PFETI iterations	2.827	12.393	423	467
	Total time (s)-sequential	273.786	280.083	19.072	452.675
	Total time (s)-parallel	48.047	50.475	3.291	78.050
	Total time cached (s)-sequential	12.246	12.563	1.237	1.885
	Total time cached (s)-parallel	21.495	22.580	2.130	32.538
SSFEM-PCG-S	PCG iterations	16	186	36	59
	PFETI iterations	528	1.167	414	338
	Total time (s)-sequential	177.970	368.937	28.790	651.255
	Total time (s)-parallel	31.236	66.484	4.968	112.289
	Total time cached (s)-sequential	74.930	154.271	18.473	276.205
	Total time cached (s)-parallel	13.180	27.807	3.189	47.624
SSFEM-PCG-SF	PCG iterations	12	25	13	20
	PFETI iterations	461	449	273	228
	Total time (s)-sequential	133.533	276.818	10.529	220.894
	Total time (s)-parallel	23.430	49.887	1.818	38.096
	Total time cached (s)-sequential	56.253	115.818	6.803	92.094
	Total time cached (s)-parallel	9.879	2.087	1.175	15.888

Table 15 presents the performance metrics for the log-normal case with 80% covariance. As previously, the SSFEM-PCG-S and SSFEM-PCG-SF variants outperform the SSFEM-PCG-B and SSFEM-PCG-BF methods, showing a speedup up to 2.8×. For this covariance of the log-normal case, the proposed caching scheme proves to be quite efficient, providing up to 3× speedup when compared to the corresponding uncached method.

Tables 16–18 compare the performance of the MC and SSFEM when using the most computationally efficient solution method for evaluating the second order moments of the response field. It can be seen that for the Gaussian input field that SSFEM outperforms Monte Carlo method. The same conclusion can be reached for the log-normal case with 30% covariance.

Table 18 shows performance metrics for the log-normal case with 80% covariance, where all solver variants are used for the SSFEM. In contrast to the log-normal case with 30% covariance, MC method outperforms

Table 16
Monte Carlo vs. SSFEM for the Gaussian case (15% covariance).

Correlation length b		0.1*a*	1*a*	10*a*	100*a*
Gaussian 15%					
MC	PCG iterations	184.398	196.875	114.715	144.592
	PFETI iterations	425.281	124.133	20.157	35.761
	Time (s)-sequential	36.771	28.076	16.443	30.590
	Time (s)-parallel	5.407	4.129	2.418	4.498
SSFEM	PCG iterations	3	4	3	4
	PFETI iterations	377	269	74	74
	Time (s)-sequential	1.026	1.178	77	100
	Time (s)-parallel	179	204	13	17

Table 17
Monte Carlo vs. SSFEM for the log-normal case (30% covariance).

Correlation length b		0.1*a*	1*a*	10*a*	100*a*
Log-normal 30%					
MC	PCG iterations	110.100	221.531	114.541	153.825
	PFETI iterations	294.235	99.478	21.777	34.375
	Time (s)-sequential	25.337	24.956	17.393	30.080
	Time (s)-parallel	3.726	3.670	2.558	4.423
SSFEM	PCG iterations	4	5	4	4
	PFETI iterations	403	260	100	79
	Time (s)-sequential	1.568	2.884	162	132
	Time (s)-parallel	277	498	28	23

Table 18
Monte Carlo vs. SSFEM for the log-normal case (80% covariance).

Correlation length b		0.1*a*	1*a*	10*a*	100*a*
Log-normal 80%					
MC	PCG iterations	1.194.328	695.100	272.340	253.350
	PFETI iterations	3.265.860	1.530.760	67.320	52.650
	Time (s)-sequential	281.182	75.286	50.653	46.932
	Time (s)-parallel	41.350	11.071	7.449	6.902
SSFEM	PCG iterations	–	89	13	20
	PFETI iterations	–	1.010	273	228
	Time (s)-sequential	–	114.721	6.803	92.094
	Time (s)-parallel	–	20.679	1.175	15.888

SSFEM in all cases except for the b = 10a correlation length. This is due to the small order of $p = 4$ required by the PC expansion, compared to the other cases which required an expansion of order $p = 6$. For the b = 0.1a case, SSFEM fails to converge.

Tables 19–21 present performance comparisons of the MC and SSFEM when using the most efficient solution method for carrying out a reliability analysis. It can be seen that for the Gaussian input field, SSFEM outperforms MC by more than 2 orders of magnitude, while for the log-normal input field with 30% covariance, SSFEM outperforms MC for all cases except for the case of 0.1a correlation length. However, for the log-normal input field with 80% covariance, SSFEM is inferior to the MC while being unable to converge for the case of 0.1a correlation length.

Table 19
Reliability analysis: MC vs. SSFEM for the Gaussian case ($\sigma_E = 15\%$ covariance).

Correlation length b		0.1a	1a	10a	100a
Gaussian 15%					
MC	PCG iterations	899.678	763.678	490.980	271.804
	PFETI iterations	2.022.934	469.789	469.789	72.059
	Time (s)-sequential	179.401	112.023	68.632	59.793
	Time (s)-parallel	26.382	16.474	10.093	8.793
		$p = 4$	$p = 4$	$p = 4$	$p = 6$
SSFEM	PCG iterations	3	4	5	6
	PFETI iterations	473	269	180	448
	Time (s)-sequential	6.797	1.178	291	1.276
	Time (s)-parallel	1.156	204	53	203

Table 20
Reliability analysis: MC vs. SSFEM for the log-normal case ($\sigma_E = 30\%$ covariance).

Correlation length b		0.1a	1a	10a	100a
Log-normal 30%					
MC	PCG iterations	1.087.678	1.143.034	490.236	356.205
	PFETI iterations	287.4353	512.199	93.203	79.748
	Time (s)-sequential	250.318	128.625	74.246	69.648
	Time (s)-parallel	36.811	18.915	10.919	10.242
		$p = 5$	$p = 4$	$p = 3$	$p = 3$
SSFEM	PCG iterations	5	5	4	4
	PFETI iterations	621	260	100	79
	Time (s) -sequential	302.369	2.884	162	132
	Time (s)-parallel	54.508	498	28	23

Table 21
Reliability analysis: MC vs. SSFEM for the log-normal case ($\sigma_E = 80\%$ covariance).

Correlation length b:		0.1a	1a	10a	100a
Log-normal 80%					
MCS	PCG iterations	2.223.683	2.478.497	792.509	557.113
	PFETI iterations	6.057.138	5.441.639	195.228	116.438
	Time (s)-sequential	522.998	267.753	147.731	103.245
	Time (s)-parallel	76.912	39.375	21.725	15.183
		–	$p = 7$	$p = 8$	$p = 6$
SSFEM	PCG iterations	-	25	15	20
	PFETI iterations	–	449	309	228
	Time (s)-sequential	–	472.607	517.627	92.094
	Time (s)-parallel	–	79.033	86.419	15.888

5. Conclusions

An investigation of the computational performance of non-intrusive Monte Carlo versus intrusive Galerkin methods of large-scale stochastic systems has been made, in the framework of high performance computing environments. The range of the relative superiority of these approaches was assessed with regard to a variety of stochastic parameters. In both approaches, the resulting algebraic equations were solved utilizing primal and dual domain decomposition methods with enhanced preconditioners, custom tailored to the specific numerical properties of the each formulation of the stochastic problem with no loss of parallel scalability.

For the case of the Monte Carlo method, the standard PCG method equiped with a PFETI sover for the preconditioning step is compared with the skyline and FETI solvers. For the case of the FETI and PFETI methods, the preconditioning steps were accelerated using an A-orthogonalization procedure which substantially reduced the number of iterations for each preconditioning step needed for the solution of the resulting near-by problems. The PFETI variant, was shown to be the most efficient preconditioning method providing enhanced numerical performance.

For the case of the spectral stochastic finite element method, a set of specialized preconditioners, customized to the properties of the augmented stiffness matrices were developed and applied, along with a caching scheme which exploits the structure of the augmented stiffness matrices of log-normal input fields. In particular, besides the block-diagonal preconditioner, a block-SSOR preconditioner was implemented which was shown to greatly enhance the solution convergence. Moreover, for the case of log-normal input fields, variants of the block diagonal and block-SSOR solvers were implemented particularly suited to the diagonal form of the augmented matrix. Furthermore, a caching scheme was proposed that greatly reduced the matrix–vector multiplication operations needed for the solution process, thus reducing the overall processing power necessary for solution at the expense of requiring more computer memory resources.

When comparing the novel solution techniques proposed for the MC procedure, a speedup of $1.25\times$ was exhibited while for the SSFEM, a speedup of $3\times$ was exhibited when utilizing the block-SSOR preconditioning combined with caching techniques with respect to the diagonally block preconditioning, making the SSFEM even more attractive for solving large scale stochastic problems in high performance computing environments.

The efficiency comparison of the Monte Carlo method and SSFEM was based on the computation of the second order moments of the response field as well as on reliability analysis. For the first case, SSFEM proved to be more efficient when dealing with input fields exhibiting small to medium covariance. However, for the case of large covariance, the Monte Carlo outperformed the SSFEM in most cases with the latter being unable to converge in one of the problems cases. Results are similar for the reliability analysis tests with the addition that the SSFEM needed a greater polynomial chaos order to converge to the MC solution, compared to the first uncertainty analysis tests.

Acknowledgments

This work has been supported by the European Research Council Advanced Grant MASTER - Mastering the computational challenges in numerical modeling and optimum design of CNT reinforced composites (ERC-2011-ADG-20110209). The authors would also like to acknowledge the fruitful discussions with George Stefanou on the contents of this paper.

Appendix A. This appendix describes the process discussed in Section 3.3 for the case of the SSFEM augmented linear system of Fig. 3, where the A matrix–vector product $\mathbf{q}^k = \mathbf{A}\mathbf{p}^k$ for iteration k is evaluated in 6 consecutive steps, as described below:

Initialization phase:

$$\mathbf{p}^k = \begin{bmatrix} \mathbf{p}_0^{k^T} & \mathbf{p}_1^{k^T} & \mathbf{p}_2^{k^T} & \mathbf{p}_3^{k^T} & \mathbf{p}_4^{k^T} & \mathbf{p}_5^{k^T} \end{bmatrix}^T$$

$${}_0\mathbf{q}^k = \begin{bmatrix} {}_0\mathbf{q}_0{}^{k^T} & {}_0\mathbf{q}_1^{k^T} & {}_0\mathbf{q}_2^{k^T} & {}_0\mathbf{q}_3^{k^T} & {}_0\mathbf{q}_4^{k^T} & {}_0\mathbf{q}_5^{k^T} \end{bmatrix}^T = 0$$

Step $i = 1$: Calculate ${}_iq_j^k,\ j = 0,1,2,3,4,5$

$$\begin{aligned}
{}_i\mathbf{q}_0^k &= {}_{i-1}\mathbf{q}_0^k + \mathbf{K}_0 \cdot \mathbf{p}_0^k \\
{}_i\mathbf{q}_1^k &= {}_{i-1}\mathbf{q}_1^k + \mathbf{K}_0 \cdot \mathbf{p}_1^k \\
{}_i\mathbf{q}_2^k &= {}_{i-1}\mathbf{q}_2^k + \mathbf{K}_0 \cdot \mathbf{p}_2^k \\
{}_i\mathbf{q}_3^k &= {}_{i-1}\mathbf{q}_3^k + \mathbf{K}_0 \cdot 2\mathbf{p}_3^k \\
{}_i\mathbf{q}_4^k &= {}_{i-1}\mathbf{q}_4^k + \mathbf{K}_0 \cdot \mathbf{p}_4^k \\
{}_i\mathbf{q}_5^k &= {}_{i-1}\mathbf{q}_5^k + \mathbf{K}_0 \cdot 2\mathbf{p}_5^k
\end{aligned} \tag{A.1}$$

Step $i = 2$: Calculate ${}_iq_j^k,\ j = 0,1,2,3,4$

$$\begin{aligned}
{}_i\mathbf{q}_0^k &= {}_{i-1}\mathbf{q}_0^k + \mathbf{K}_1 \cdot \mathbf{p}_0^k \\
{}_i\mathbf{q}_1^k &= {}_{i-1}\mathbf{q}_1^k + \mathbf{K}_1 \cdot \mathbf{p}_0^k + \mathbf{K}_1 \cdot 2\mathbf{p}_3^k \\
{}_i\mathbf{q}_2^k &= {}_{i-1}\mathbf{q}_2^k + \mathbf{K}_1 \cdot \mathbf{p}_4^k \\
{}_i\mathbf{q}_3^k &= {}_{i-1}\mathbf{q}_3^k + \mathbf{K}_1 \cdot 2\mathbf{p}_1^k \\
{}_i\mathbf{q}_4^k &= {}_{i-1}\mathbf{q}_4^k + \mathbf{K}_1 \cdot \mathbf{p}_2^k
\end{aligned} \tag{A.2}$$

Step $i = 3$: Calculate ${}_iq_j^k,\ j = 0,1,2,4,5$

$$\begin{aligned}
{}_i\mathbf{q}_0^k &= {}_{i-1}\mathbf{q}_0^k + \mathbf{K}_2 \cdot \mathbf{p}_2^k \\
{}_i\mathbf{q}_1^k &= {}_{i-1}\mathbf{q}_1^k + \mathbf{K}_2 \cdot \mathbf{p}_4^k \\
{}_i\mathbf{q}_2^k &= {}_{i-1}\mathbf{q}_2^k + \mathbf{K}_2 \cdot \mathbf{p}_1^k + \mathbf{K}_2 \cdot 2\mathbf{p}_5^k \\
{}_i\mathbf{q}_4^k &= {}_{i-1}\mathbf{q}_4^k + \mathbf{K}_2 \cdot 2\mathbf{p}_1^k \\
{}_i\mathbf{q}_5^k &= {}_{i-1}\mathbf{q}_5^k + \mathbf{K}_2 \cdot 2\mathbf{p}_2^k
\end{aligned} \tag{A.3}$$

Step $i = 4$: Calculate ${}_iq_j^k,\ j = 0,1,3,4$

$$\begin{aligned}
{}_i\mathbf{q}_0^k &= {}_{i-1}\mathbf{q}_0^k + \mathbf{K}_3 \cdot 2\mathbf{p}_3^k \\
{}_i\mathbf{q}_1^k &= {}_{i-1}\mathbf{q}_1^k + \mathbf{K}_3 \cdot 2\mathbf{p}_1^k \\
{}_i\mathbf{q}_3^k &= {}_{i-1}\mathbf{q}_3^k + \mathbf{K}_3 \cdot 2\mathbf{p}_0^k + \mathbf{K}_3 \cdot 8\mathbf{p}_3^k \\
{}_i\mathbf{q}_4^k &= {}_{i-1}\mathbf{q}_4^k + \mathbf{K}_3 \cdot 2\mathbf{p}_4^k
\end{aligned} \tag{A.4}$$

Step $i = 5$: Calculate ${}_iq_j^k,\ j = 0,1,2,3,4,5$

$$\begin{aligned}
{}_i\mathbf{q}_0^k &= {}_{i-1}\mathbf{q}_0^k + \mathbf{K}_4 \cdot \mathbf{p}_4^k \\
{}_i\mathbf{q}_1^k &= {}_{i-1}\mathbf{q}_1^k + \mathbf{K}_4 \cdot \mathbf{p}_2^k \\
{}_i\mathbf{q}_2^k &= {}_{i-1}\mathbf{q}_2^k + \mathbf{K}_4 \cdot \mathbf{p}_1^k \\
{}_i\mathbf{q}_3^k &= {}_{i-1}\mathbf{q}_3^k + \mathbf{K}_4 \cdot 2\mathbf{p}_4^k \\
{}_i\mathbf{q}_4^k &= {}_{i-1}\mathbf{q}_4^k + \mathbf{K}_4 \cdot \mathbf{p}_0^k + \mathbf{K}_4 \cdot 2\mathbf{p}_5^k \\
{}_i\mathbf{q}_5^k &= {}_{i-1}\mathbf{q}_5^k + \mathbf{K}_4 \cdot 2\mathbf{p}_4^k
\end{aligned} \tag{A.5}$$

Step $i = 6$: Calculate ${}_iq_j^k,\ j = 0,2,5$

$$\begin{aligned}
{}_i\mathbf{q}_0^k &= {}_{i-1}\mathbf{q}_0^k + \mathbf{K}_5 \cdot 2\mathbf{p}_5^k \\
{}_i\mathbf{q}_2^k &= {}_{i-1}\mathbf{q}_2^k + \mathbf{K}_5 \cdot 2\mathbf{p}_2^k \\
{}_i\mathbf{q}_5^k &= {}_{i-1}\mathbf{q}_5^k + \mathbf{K}_5 \cdot 2\mathbf{p}_0^k + \mathbf{K}_5 \cdot 8\mathbf{p}_5^k
\end{aligned} \tag{A.6}$$

with ${}_6\mathbf{q}^k = \mathbf{q}^k$. By examining these steps, it is evident that the computation of A matrix–vector product is computationally intensive since for a 6x6 matrix, 34 matrix–vector (MV) products and 19 linear scalings are being computed.

References

[1] Y.T. Feng, C.F. Li, D.R.J. Owen, A directed Monte Carlo solution of linear stochastic algebraic system of equations, Finite Elem. Anal. Des. 46 (2010) 462–473.

[2] R. Ghanem, R.M. Kruger, Numerical solution of spectral stochastic finite element systems, Comput. Methods Appl. Mech. Eng. 129 (1996) 289–303.

[3] C.F. Li, S. Adhikari, C. Song, Y.T. Feng, D.R.J. Owen, A joint diagonalisation approach for linear stochastic systems, Comput. Struct. 88 (2010) 1137–1148.

[4] M. Papadrakakis, V. Papadopoulos, Robust and efficient methods for stochastic finite element analysis using Monte Carlo simulation, Comput. Methods Appl. Mech. Eng. 134 (1996) 325–340.

[5] M.F. Pellissetti, R. Ghanem, Iterative solution of systems of linear equations arising in the context of stochastic finite elements, Adv. Eng. Software 31 (2000) 607–616.

[6] D.C. Charmpis, Incomplete factorization preconditioners for the iterative solution of stochastic finite element equations, Comput. Struct. 88 (3–4) (2010) 178–188.

[7] D.C. Charmpis, M. Papadrakakis, G.S. Stefanou, A domain decomposition solution for the stochastic finite element analysis of shells, in: H.A. Mang, F.G. Rammerstorfer, J. Eberhardsteiner (Eds.), Proceedings of the Fifth World Congress on Computational Mechanics (WCCM V), Vienna University of Technology, Austria, 2002.

[8] D.B. Chung, M.A. Guttierez, L.L. Graham-Brady, Lingen F-J, Efficient numerical strategies for spectral stochastic finite element models, Int. J. Numer. Methods Eng. 64 (2005) 1334–1349.

[9] M.F. Pellissetti, Parallel processing in structural reliability, Struct. Eng. Mech. 32 (1) (2009) 95–126.

[10] V. Papadopoulos, D.C. Charmpis, M. Papadrakakis, A computationally efficient method for the buckling analysis of shells with stochastic imperfections, Comput. Mech. 43 (5) (2009) 687–700.

[11] M. Papadrakakis, Kotsopulos A, Parallel solution methods for stochastic finite element analysis using Monte Carlo simulation, Comput. Methods Appl. Mech. Eng. 168 (1999) 305–320.

[12] D.C. Charmpis, M. Papadrakakis, Improving the computational efficiency in finite element analysis of shells with uncertain properties, Comput. Methods Appl. Mech. Eng. 194 (2005) 1447–1478.

[13] M. Anders, M. Hori, Three-dimensional stochastic finite element method for elasto-plastic bodies, Int. J. Numer. Methods Eng. 51 (2001) 449–478.

[14] N.Z. Chen, C. Guedes Soares, Spectral stochastic finite element analysis for laminated composite plates, Comput. Methods Appl. Mech. Eng. 197 (2008) 4830–4839.

[15] D.B. Chung, M.A. Gutierrez, R. de Borst, Object-oriented stochastic finite element analysis of fibre metal laminates, Comput. Methods Appl. Mech. Eng. 194 (2005) 1427–1446.

[16] R.G. Ghanem, P.D. Spanos (Eds.), Stochastic Finite Elements: A Spectral Approach, Springer-Verlag, 1991.

[17] A. Keese, A review of recent developments in the numerical solution of stochastic partial differential equations (stochastic finite elements), Working Report 2003-06, Technical University, Braunschweig, Germany, 2003.

[18] H.G. Matthies, A. Keese, Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations, Comput. Methods Appl. Mech. Eng. 194 (2005) 1295–1331.

[19] N. Wiener, The homogeneous chaos, Am. J. Math. 60 (1938) 897–936.

[20] M.M.R. Williams, Polynomial chaos functions and stochastic differential equations, Ann. Nucl. Energy 33 (2006) 774–785.

[21] D. Xiu, G.E. Karniadakis, The Wiener–Askey polynomial chaos for stochastic differential equations, SIAM J. Sci. Comput. 24 (2002) 619–644.

[22] G. Blatman, B. Sudret, An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis, Probab. Eng. Mech. 25 (2) (2010) 183–197.

[23] S. Adhikari, A reduced spectral function approach for the stochastic finite element analysis, Comput. Methods Appl. Mech. Eng. 200 (21–22) (2011) 1804–1821.

[24] M. Anders, M. Hori, Three-dimensional stochastic finite element method for elasto-plastic bodies, Int. J. Numer. Methods Eng. 51 (2001) 449–478.

[25] R. Ghanem, P.D. Spanos, Polynomial chaos in stochastic finite elements, J. Appl. Mech., ASME (1990) 197–202.

[26] C.F. Li, Y.T. Feng, D.R.J. Owen, Explicit solution to the stochastic system of linear algebraic equations $({}_1a_1 + {}_2a_2 + \cdots + {}_ma_m)\mathrm{x} = \mathrm{b}$, Comput. Methods Appl. Mech. Eng. 195 (44–47) (2006) 6560–6576.

[27] C. Desceliers, R. Ghanem, C. Soize, Polynomial chaos representation of a stochastic preconditioner, Int. J. Numer. Methods Eng. 64 (5) (2005) 618–634.

[28] P. Fraunfelder, C. Schwab, R.A. Todor, Finite elements for elliptic problems with stochastic coefficients, Comput. Methods Appl. Mech. Eng. 194 (2005) 205–228.

[29] D. Ghosh, P. Avery, C. Farhat, A FETI-preconditioned conjugate gradient method for large-scale stochastic finite element problems, Int. J. Numer. Methods Eng 80 (2009) 914–931.

[30] A. Keese, H.G. Matthies, Hierarchical parallelisation for the solution of stochastic finite element equations, Comput. Struct. 83 (2005) 1033–1047.

[31] H.M. Panayirci, Efficient solution of Galerkin-based polynomial chaos expansion systems, Adv. Eng. Software 41 (2010) 1277–1286.

[32] W. Subber, A. Sarkar, Dual-primal domain decomposition method for uncertainty quantification, Comput. Methods Appl. Mech. Eng. 266 (2013) 112–124.

[33] M. Grigoriu, Evaluation of Karhunen–Loève spectral and sampling representations for stochastic processes, J. Eng. Mech. 132 (2006) 179–189.

[34] S.P. Huang, S.T. Quek, K.K. Phoon, Convergence study of the truncated Karhunen–Loève expansion for simulation of stochastic processes, Int. J. Numer. Methods Eng. 52 (2001) 1029–1043.

[35] M. Papadrakakis, Solving Large-Scale Linear Problems in Solid and Structural Mechanics, John Wiley & Sons, 1993.

[36] Y. Fragakis, M. Papadrakakis, The mosaic of high performance domain decomposition methods for structural mechanics: formulation, interrelation and numerical efficiency of primal and dual methods, Comput. Methods Appl. Mech. Eng. 192 (2003) 35–36.

[37] Y. Fragakis, M. Papadrakakis, The mosaic of high performance domain decomposition methods for structural mechanics – Part II: Formulation enhancements, multiple right-hand sides and implicit dynamics, Comput. Methods Appl. Mech. Eng. 193 (2004) 4611–4662.

[38] C. Farhat, F.-X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, Int. J. Numer. Methods. Eng. 32 (1991) 1205–1227.

[39] R.-G. Ghanem, The nonlinear Gaussian spectrum of log-normal stochastic processes and variables, J. Appl. Mech. 66 (1999) 964–973.

[40] D. Ghosh, P. Avery, C. Farhat, A method to solve spectral stochastic finite element problems for large-scale systems, Int. J. Numer. Methods. Eng. 00 (2008) 1–6.