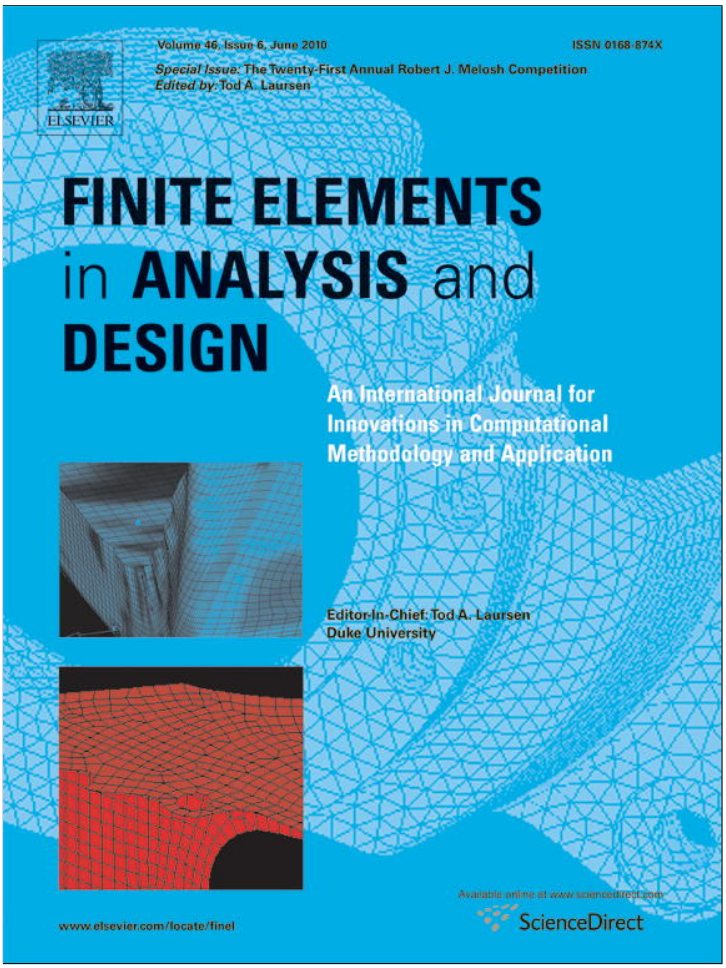

Finite Elements in Analysis and Design 46 (2010) 462–473

Contents lists available at ScienceDirect

Finite Elements in Analysis and Design

journal homepage: www.elsevier.com/locate/finel

A directed Monte Carlo solution of linear stochastic algebraic system of equations

Y.T. Feng *, C.F. Li, D.R.J. Owen

Civil and Computational Engineering Centre, School of Engineering, Swansea University, SA2 8PP, UK

ARTICLE INFO

Article history:
Received 17 November 2009
Accepted 16 December 2009
Available online 12 February 2010

Keywords:
Stochastic finite elements
Stochastic linear algebraic system of equations
Monte Carlo simulations
Hyper-spherical transformation
Iterative solver
Preconditioning

ABSTRACT

This paper proposes a modified Monte Carlo simulation method for the solution of a linear stochastic algebraic system of equations arising from the stochastic finite element modelling of linear elastic problems. The basic idea is to direct Monte Carlo samples along straight lines and then utilise their spatial proximity or order to provide high quality initial approximations in order to significantly accelerate the convergence of iterative solvers at each sample. The method, termed the directed Monte Carlo (DMC) simulation, is developed first for one random variable using the preconditioned conjugate gradient equipped with an initial approximation prediction scheme, and then extended to multiple Gaussian random variable cases by the adoption of a general hyper-spherical transformation. The eigenproperties of the linear system are also briefly discussed to reveal the suitability of several preconditioning schemes for iterative solvers. Two numerical examples with up to around 6000 DOFs are provided to assess the performance of the proposed solution strategy and associated numerical techniques in terms of computational costs and solution accuracy.

1. Introduction

The main objective of this paper is to propose a Monte Carlo based simulation method for the solution of the following linear stochastic algebraic system of equations arising from the stochastic finite element modelling (SFEM) of linear elastic problems:

$$[\mathbf{K}_0+\mathbf{K}(\boldsymbol{\varepsilon})]\mathbf{u}(\boldsymbol{\varepsilon})=\mathbf{b} \tag{1}$$

where

$$\mathbf{K}(\boldsymbol{\varepsilon})=\varepsilon_1\mathbf{K}_1+\cdots+\varepsilon_n\mathbf{K}_n;\quad \boldsymbol{\varepsilon}=(\varepsilon_1,\ldots,\varepsilon_n)$$

in which $\mathbf{K}_0$ and $\mathbf{K}_i$ $(i=1,\ldots,n)$ are $N\times N$ deterministic matrices; ε_i are n mutually un-correlated/independent random variables with certain probability distributions and collectively form an n-dimensional random vector $\boldsymbol{\varepsilon}$; $\mathbf{b}$ is a deterministic external loading vector; and $\mathbf{u}(\boldsymbol{\varepsilon})$ is the random displacement vector to be sought.

The development of effective solution strategies for Eq. (1) to obtain various statistical properties of the solution $\mathbf{u}(\boldsymbol{\varepsilon})$ and the associated results, such as strain and stress, is one of the central issues in SFEM and becomes increasingly more important when larger scale problems with many random variables need to be considered. Various solution approaches have been proposed over the past decades. See [1,2] for detailed reviews and the references therein. Among all existing solution methods, Monte Carlo (MC) simulations are considered to be the most versatile approach and are in fact almost always used at some stage in most stochastic solution procedures. The main disadvantage of Monte Carlo simulations, however, is the intensive computational cost involved, particularly for large scale problems with many random variables, since a large number of samples are normally required to achieve a reasonable solution accuracy and a new system of equations needs to be solved at each MC sample.

In addition to those schemes traditionally proposed to improve the sampling strategies of Monte Carlo methods, including importance sampling, stratified sampling, recursive stratified sampling and adaptive Monte Carlo and especially VERGAS [3] (see [4] for a brief review on the topic), some progress has also been made in SFEM where the focus is on the solution of Eq. (1) by employing iterative approaches, such as preconditioned conjugate gradients (PCG), so that large scale problems could be tackled. Recent developments in this aspect can be found, e.g. in [5–8]. One example is the recent work of [9] which proposes an explicit stochastic incomplete Cholesky (IC) preconditioner for $\mathbf{K}_0+\mathbf{K}(\boldsymbol{\varepsilon})$ based on the polynomial chaos expansion concept [1] to achieve a fast solution convergence of PCG at each Monte Carlo sampling point. It has to be pointed out, however, that in contrast to the solution of deterministic linear systems of equations, existing stochastic solution procedures are generally far from

* Corresponding author.
E-mail addresses: Y.T.Feng@swansea.ac.uk (Y.T. Feng), D.R.J.Owen@swansea.ac.uk (D.R.J. Owen).

0168-874X/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.finel.2010.01.004

computationally adequate to handle large scale problems with many random variables, and therefore new and more advanced solution techniques are urgently needed in order to greatly improve the modelling capability of SFEM for practical applications.

The key feature of Monte Carlo samples is the randomness and unpredictability in the sequence of their spatial positions, in the space formed by the random variables, except for their statistical property. Therefore, it appears necessary to solve a separate system of equations of (1) at each Monte Carlo sampling $\boldsymbol{\varepsilon}_s$. Note that this is always the case when Eq. (1) is solved by a direct solver. However, after all the Monte Carlo samples are generated, many of them may be closely positioned spatially, particularly in the high probability regions and when the number of the sampling points is large. This spatial proximity of Monte Carlo samples suggests that if an iterative solver is employed, the solution obtained at one or more MC sample may be used to provide a good initial approximation for their immediate neighbours which could result in a significant reduction in the number of iterations required at these points. It is this observation that will be fully exploited in this work in order to improve the computational efficiency of MC simulations for the solution of Eq. (1). Note that an adaptively preconditioned iterative method has been successfully developed in [12] to solve one random variable system, the special case of Eq. (1).

In the next section, the SFEM formulation for the generation of the matrices $\mathbf{K}_i$ associated with the random variables in (1) and their underlying joint eigenproperties with the deterministic matrix $\mathbf{K}_0$ will be briefly discussed to identify a possible very effective preconditioning scheme to be used later in PCG. A modified Monte Carlo approach, termed the directed Monte Carlo Method and based on the utilisation of the spatial proximity of MC sampling points, will be developed first for one random variable in Section 3. Two integrated numerical techniques essential for the success of the new method, including preconditioning and initial approximation prediction, are discussed in detail. The extension of the approach to multiple Gaussian random variable cases is undertaken by the adoption of a general hyper-spherical concept in Section 4. Numerical experiments are conducted in Section 5 to assess the performance of the proposed solution strategy and associated numerical techniques in terms of computational costs and solution accuracy.

2. Stochastic finite element modelling of random elastic media

2.1. Explicit representation of a stochastic medium

It is assumed that the stochastic source of (1) is due to the irregular variation of the material properties, mainly Young's module $\mathbf{E}$, present in the problem concerned. This randomness may be described either by a set of un-correlated random variables or by a continuous stochastic field. In the former case, individual random variables may be employed to represent the randomness in different sub-domains of the problem and therefore may have different statistical properties. In the latter case, a real-valued stochastic function can be used to describe the random feature for the whole domain or for each sub-domain. In this situation, the so called Karhunen–Loève (K–L) expansion, originally developed in the theory of probability [10] and first introduced to SFEM by Ghanem and Spanos [1], has so far proved to be the most elegant mathematical representation for general second-order stochastic fields.

Suppose that the Young's modulus can be expressed by a real-valued stochastic function $\mathbf{E}(\mathbf{x},\varpi)$, where $\mathbf{x}$ is the coordinate vector of an arbitrary point in the problem domain Ω and ϖ represents a random event, and let $E(f(\cdot,\varpi))$ denote the mathematical expectation of a random function f. Given the expectation function $E_m(\mathbf{x})=E(\mathbf{E}(\mathbf{x},\varpi))$ and a covariance function $\mathbf{R}(\mathbf{x},\mathbf{y})\triangleq \mathrm{Cov}(\mathbf{E}(\mathbf{x},\varpi),\mathbf{E}(\mathbf{y},\varpi))$ where $\mathbf{x},\mathbf{y}\in\Omega$, the K–L expansion theorem states that $\mathbf{E}(\mathbf{x},\varpi)$ can be expressed as [10,1]

$$\mathbf{E}(\mathbf{x},\varpi)=E_m(\mathbf{x})+\sum_{i=1}^{\infty}\varepsilon_i(\varpi)\sqrt{\lambda_i^E}\,\boldsymbol{\phi}_i^E(\mathbf{x}) \tag{2}$$

where λ_i^E and $\boldsymbol{\phi}_i^E(\mathbf{x})$ are a pair of eigenvalue and eigenfunction of the following characteristic equation:

$$\int_\Omega \mathbf{R}(\mathbf{x},\mathbf{y})\boldsymbol{\phi}(\mathbf{y})d\mathbf{y}=\lambda\boldsymbol{\phi}(\mathbf{x}) \tag{3}$$

with the properties

$$\text{(a) } \lambda_i^E>0 \quad\text{and}\quad \text{(b) } \int_\Omega \boldsymbol{\phi}_i^E(\mathbf{x})\boldsymbol{\phi}_j^E(\mathbf{x})\,d\mathbf{x}=\delta_{ij} \tag{4}$$

where δ_{ij} is the Kronecker delta; and ε_i are mutually un-correlated random variables with zero mean and unit variance, i.e.

$$E(\varepsilon_i(\varpi))=0 \quad\text{and}\quad E(\varepsilon_i(\varpi)\varepsilon_j(\varpi))=\delta_{ij} \tag{5}$$

In essence, the above K–L expansion (2) transforms an implicitly defined random field into the combination of a (infinite) discrete set of un-correlated random variables. These variables possess the same probability distribution depending on the type of problem considered. In many engineering applications, ε_i can be assumed to be standard Gaussian random variables with zero mean and unit standard derivation, and therefore are mutually independent. Issues related to the effective computation of both λ_i^E and $\boldsymbol{\phi}_i^E$ are addressed in [11] with the proposal of the Fourier–Karhunen–Loève (F–K–L) scheme. In addition, more subtle issues concerned with the determination of probabilistic properties of the random variables ε_i are also discussed in [11].

In numerical simulations, Expression (2) is truncated into a finite series after sorting λ_i^E in descending order, and the number of the retained terms/random variables, n, can be determined with a required accuracy τ_E based on the so called trace relation [1]

$$\sum_{i=1}^{\infty}\lambda_i^E=\int_\Omega \sigma_E^2(\mathbf{x})\,d\mathbf{x} \tag{6}$$

where $\sigma_E^2(\mathbf{x})=\mathbf{R}(\mathbf{x},\mathbf{x})$. The fact that the eigenfunction $\boldsymbol{\phi}_i^E$ is normalised (refer to Eq. (4b)) makes λ_i^E a good indicator for the scale of variation that the associated random variable ε_i contributes to $\mathbf{E}(\mathbf{x})$.

The K–L expansion (2) admits, in a strict sense, that $\mathbf{E}$ may achieve any value, including negative and very large positive. Even though these situations occur only with a very small probability and thus statistically acceptable, they are not physical and might occasionally cause numerical difficulties at later stages. In engineering practice, the minimum and maximum value of $\mathbf{E}$, E_{min} and E_{max}, can often be specified and thus it is possible to determine the physically allowed value range for each random variable ε_i.

Suppose that the sample range of ε_i is a 1-D interval $\mathcal{D}\subset\mathcal{R}$ where $\mathcal{R}$ denotes the whole real number space, and that all ε_i have the same probability distribution. These random variables $\varepsilon_i(n=1,\ldots,n)$ form a sample volume $\mathbb{D}^n=\mathcal{D}\times\mathcal{D}\times\cdots\times\mathcal{D}$ in the n-dimensional real space $\mathbb{R}^n$.

2.2. Stiffness matrices associated with random variables and their properties

After an explicit representation of the random distribution of Young's modulus is achieved, the matrices $\mathbf{K}_0$ and $\mathbf{K}_i$ in Eq. (1) can

be computed following the standard stochastic finite element formulation (see for instance [1]). $\mathbf{K}_0$ is equivalent to the global stiffness matrix in the corresponding deterministic case with $\mathbf{E}=E_m$ and therefore is symmetric and positive definite (SPD), while $\mathbf{K}_i$ are symmetric but normally indefinite.

In addition to the statistical behaviour of the random vector $\boldsymbol{\varepsilon}$, the relationship between $\mathbf{K}_0$ and $\mathbf{K}(\boldsymbol{\varepsilon})$ also plays a paramount role in the solution of Eq. (1), $\mathbf{u}(\boldsymbol{\varepsilon})$. Particularly, to gain a fundamental understanding of their underlying relationship will shed a light onto the development of effective solution strategies for Eq. (1). This important issue is briefly addressed here.

As $\mathbf{K}_0$ is SPD, it permits a standard LL^T Cholesky decomposition:

$$\mathbf{K}_0=\mathbf{L}_0\mathbf{L}_0^T \tag{7}$$

and, together with symmetry of $\mathbf{K}(\boldsymbol{\varepsilon})$, ensures that the following generalised eigenproblem formed by $\mathbf{K}_0$ and $\mathbf{K}(\boldsymbol{\varepsilon})$ has real eigenvalues $\boldsymbol{\Lambda}_\varepsilon=\mathrm{diag}\{\lambda_1^\varepsilon,\ldots,\lambda_N^\varepsilon\}$ and corresponding real eigenvectors $\boldsymbol{\Phi}_\varepsilon=[\boldsymbol{\phi}_1^\varepsilon,\ldots,\boldsymbol{\phi}_N^\varepsilon]$:

$$\mathbf{K}(\boldsymbol{\varepsilon})\boldsymbol{\Phi}_\varepsilon=\mathbf{K}_0\boldsymbol{\Phi}_\varepsilon\boldsymbol{\Lambda}_\varepsilon \tag{8}$$

with

$$\boldsymbol{\Phi}_\varepsilon^T\mathbf{K}(\boldsymbol{\varepsilon})\boldsymbol{\Phi}_\varepsilon=\boldsymbol{\Lambda}_\varepsilon;\quad \boldsymbol{\Phi}_\varepsilon^T\mathbf{K}_0\boldsymbol{\Phi}_\varepsilon=\mathbf{I}\quad\text{(Identity matrix)}$$

and

$$\lambda_1^\varepsilon\le\lambda_2^\varepsilon\le\cdots\le\lambda_N^\varepsilon$$

The eigenpairs $\boldsymbol{\Lambda}_\varepsilon(\boldsymbol{\varepsilon})$ and $\boldsymbol{\Phi}_\varepsilon(\boldsymbol{\varepsilon})$ also possess the following properties:

$$\boldsymbol{\Lambda}_\varepsilon(0)=0;\quad \boldsymbol{\Phi}_\varepsilon(0)=\mathbf{L}_0^{-T}\quad\text{and}\quad \boldsymbol{\Lambda}_\varepsilon(-\boldsymbol{\varepsilon})=-\boldsymbol{\Lambda}_\varepsilon(\boldsymbol{\varepsilon}) \tag{9}$$

With the aid of the above eigendecomposition, the original equation (1) can be decoupled to

$$[\mathbf{I}+\boldsymbol{\Lambda}_\varepsilon]\boldsymbol{\Phi}_\varepsilon^{-1}\mathbf{u}(\boldsymbol{\varepsilon})=\boldsymbol{\Phi}_\varepsilon^T\mathbf{b} \tag{10}$$

which leads to an explicit solution of $\mathbf{u}(\boldsymbol{\varepsilon})$:

$$\mathbf{u}(\boldsymbol{\varepsilon})=\boldsymbol{\Phi}_\varepsilon[\mathbf{I}+\boldsymbol{\Lambda}_\varepsilon]^{-1}\boldsymbol{\Phi}_\varepsilon^T\mathbf{b}=\sum_{i=1}^{N}\frac{\mathbf{b}^T\boldsymbol{\phi}_i^\varepsilon}{1+\lambda_i^\varepsilon}\boldsymbol{\phi}_i^\varepsilon \tag{11}$$

Clearly, the eigenvalues λ_i^ε represent the variation scale caused by the random variables at the structural response level. The solution at each sampling point of $\boldsymbol{\varepsilon}$ will depend on the actual spectrum of λ_i^ε. Especially those extreme eigenvalues at both ends of the spectrum will have major contributions to the random variation of $\mathbf{u}$. Again, mathematically, λ_i^ε can take any value within $(-\infty,+\infty)$, e.g. when ε_i are the standard Gaussian random variables. Then when one of λ_i^ε is very close to -1 at one sample, for instance, it will give rise to a (nearly) singular situation with the possible consequence that a numerical difficulty may be encountered when solving Eq. (1), and/or that the final statistical properties of the solution may be contaminated by this single sample. The occurrence of such an undesired situation normally increases when a large number of sample points is required.

However, the physically imposed condition $E_{min}\le\mathbf{E}\le E_{max}$, which restricts the possible sample ranges of ε_i, also ensures that $\mathbf{K}_0+\mathbf{K}(\boldsymbol{\varepsilon})$ is SPD, i.e.

$$1+\lambda_i^\varepsilon>0\quad\text{or}\quad\lambda_i^\varepsilon>-1 \tag{12}$$

which, when taking into consideration of the property $\lambda_i^\varepsilon(\boldsymbol{\varepsilon})=-\lambda_i^\varepsilon(-\boldsymbol{\varepsilon})$, further leads to

$$\lambda_i^\varepsilon<1 \tag{13}$$

It follows from (12) and (13) that

$$|\lambda_i^\varepsilon|<1\quad(i=1,\ldots,N) \tag{14}$$

or

$$-1<\lambda_1^\varepsilon\le\lambda_2^\varepsilon\le\cdots\le\lambda_N^\varepsilon<1 \tag{15}$$

Thus unless E_{min} is too small compared to the mean value E_m, the previously mentioned numerical difficulty associated with the (near) singularity of $\mathbf{K}_0+\mathbf{K}(\boldsymbol{\varepsilon})$ can be avoided.

Furthermore, using $\mathbf{K}_0=\mathbf{L}_0\mathbf{L}_0^T$ as a preconditioner to Eq. (1) results in

$$(\mathbf{I}+\overline{\mathbf{K}}_\varepsilon)\mathbf{L}_0^T\mathbf{u}=\mathbf{L}_0^{-1}\mathbf{b} \tag{16}$$

with

$$\overline{\mathbf{K}}_\varepsilon=\mathbf{L}_0^{-1}\mathbf{K}(\varepsilon)\mathbf{L}_0^{-T}$$

It is easy to prove that $\boldsymbol{\Lambda}_\varepsilon$ is also the eigenvalue matrix of $\overline{\mathbf{K}}_\varepsilon$:

$$\overline{\boldsymbol{\Phi}}_\varepsilon^T\overline{\mathbf{K}}_\varepsilon\overline{\boldsymbol{\Phi}}_\varepsilon=\boldsymbol{\Lambda}_\varepsilon;\quad\overline{\boldsymbol{\Phi}}_\varepsilon^T\overline{\boldsymbol{\Phi}}_\varepsilon=\mathbf{I} \tag{17}$$

with the eigenvector matrix $\overline{\boldsymbol{\Phi}}_\varepsilon=\mathbf{L}_0^{-T}\boldsymbol{\Phi}_\varepsilon$. Thus, the condition number of $\mathbf{I}+\overline{\mathbf{K}}_\varepsilon$, κ, can be estimated as

$$\kappa=\max\left\{\frac{1+|\lambda_N^\varepsilon|}{1-|\lambda_1^\varepsilon|},\frac{1+|\lambda_1^\varepsilon|}{1-|\lambda_N^\varepsilon|}\right\}<\max\left\{\frac{2}{1-|\lambda_1^\varepsilon|},\frac{2}{1-|\lambda_N^\varepsilon|}\right\} \tag{18}$$

When the extreme eigenvalues $|\lambda_1^\varepsilon|$ and $|\lambda_N^\varepsilon|$ are not very close to 1, which is normally the case for the majority of samples, κ will be in order of $O(1)$. Thus it is concluded that $\mathbf{K}_0$ is in general a good preconditioner for Eq. (1). It is also worth highlighting that at problem scales that SFEM can currently handle effectively, the Cholesky decomposition is still very computationally competitive in terms of both memory and CPU time costs. Particularly, with the continuing advance of computer hardware, the scale of problems that a direct solver can effectively solve is also increasing. Furthermore, in the current problem concerned, a large number of repeated uses of the triangular decomposition $\mathbf{L}_0$ of $\mathbf{K}_0$ will significantly offset the overhead associated with the initial computation of $\mathbf{L}_0$ which further increases the effectiveness of the Cholesky decomposition. Thus the lower triangular matrix $\mathbf{L}_0$ and the deterministic solution $\mathbf{u}_d$

$$\mathbf{u}_d=\mathbf{u}(0)=\mathbf{K}_0^{-1}\mathbf{b} \tag{19}$$

are assumed available when needed.

It is normally difficult to compute $\boldsymbol{\Lambda}_\varepsilon$ and $\boldsymbol{\Phi}_\varepsilon$ as the general (random) functions of random variables $\boldsymbol{\varepsilon}$, but it is computationally feasible to obtain the joint eigenproperties of each matrix $\mathbf{K}_i$ with $\mathbf{K}_0$, particularly the two extreme eigenvalues, λ_1^i and λ_N^i. Investigating the eigenstructures of these individual matrices can reveal certain features of $\boldsymbol{\Lambda}_\varepsilon$ and $\boldsymbol{\Phi}_\varepsilon$. This issue, however, will not be pursued further in the present work.

3. Modified Monte Carlo simulation: one random variable

Suppose that the probability distribution function (PDF) of each random variable ε_i is $p(\varepsilon_i)$. As all ε_i are assumed independent, their joint PDF $p(\boldsymbol{\varepsilon})$ can be expressed as

$$p(\boldsymbol{\varepsilon})=p(\varepsilon_1)p(\varepsilon_2)\cdots p(\varepsilon_n)=\prod_{i=1}^{n}p(\varepsilon_i) \tag{20}$$

A statistical characteristic, such as the expectation, of an arbitrary function f of $\mathbf{u}(\boldsymbol{\varepsilon})$ can be generally written as

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))]=\int_{\mathbb{D}^n}f(\mathbf{u}(\boldsymbol{\varepsilon}))p(\boldsymbol{\varepsilon})\,dV \tag{21}$$

where the infinitesimal volume element $dV=d\varepsilon_1\cdot d\varepsilon_2\cdots d\varepsilon_n$. The standard Monte Carlo simulation estimates the above integral by taking the arithmetic means of the integral function $f(\mathbf{u}(\boldsymbol{\varepsilon}))$ over $\mathrm{M_c}$ points $\{\boldsymbol{\varepsilon}_1,\ldots,\boldsymbol{\varepsilon}_{M_c}\}$ sampled according to the required

PDF $p(\boldsymbol{\varepsilon})$:

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] \approx \frac{1}{M_c}\sum_{i=1}^{M_c} f(\mathbf{u}(\boldsymbol{\varepsilon}_i)) \tag{22}$$

The most expensive operation in the Monte Carlo simulation is that involved in the solution of (1) at each Monte Carlo point. The new Directed Monte Carlo approach, aimed at improving the computational efficiency of solving the equations on the basis of the utilisation of the spatial proximity of Monte Carlo sampling points, will be developed, first for one random variable cases in this section and then extended to general multiple random variable cases in the next section. This modified Monte Carlo approach consists of two essential ingredients: (1) an iterative algorithm is employed to solve the linear system of equations at each Monte Carlo sample and (2) the sampling points are first spatially sorted so that a high quality initial solution to the current sample can be obtained based on the solutions at its neighbours already solved.

3.1. Preconditioned conjugate gradient method

The iterative solution of a linear system of equations is well established, and many solution algorithms have been proposed. Among all iterative solvers developed, the preconditioned conjugate gradient algorithm is the most popular iterative solver for a SPD system. It is interesting to note that the PCG algorithm has been recently established to be equivalent to a second-order time integration scheme [13]. The standard PCG algorithm for solving a linear system of

$$\mathbf{K}\mathbf{x} = \mathbf{b}$$

with given preconditioning matrix $\mathbf{M}_p$ (also assumed SPD), initial guess $\mathbf{x}_0$, required solution accuracy τ and maximum iterations *maxite*, is provided in Box 1 for reference.

As all the computational aspects of the standard PCG have been thoroughly investigated, no new development will be offered in this work. However, there are two integrated issues that are essential to the success of the DMC method, and therefore deserve further discussions: (1) preconditioning and (2) initial guess. The latter issue will be elaborated upon in the next subsection.

Preconditioning is the most important technique for PCG to achieve a fast convergence for practical problems by providing a possible significant reduction in the condition number of the original matrix. Incomplete Cholesky (IC) decomposition and its variants are most commonly used preconditioning schemes [14]. Let the IC decomposition of a matrix $\mathbf{K}$ be denoted as IC($\mathbf{K}$). For the current problem concerned, since the matrix $\mathbf{K}(\boldsymbol{\varepsilon})$ varies at every sample point, there exist several options for the preconditioning matrix $\mathbf{M}_p$:

Option 1 (IC0): $\mathbf{M}_p = \mathrm{IC}(\mathbf{K}_0)$, the IC decomposition of $\mathbf{K}_0$;

Option 2 (SIC): $\mathbf{M}_p = \mathrm{IC}(\mathbf{K}_0 + \mathbf{K}(\boldsymbol{\varepsilon}))$, the IC decomposition of $\mathbf{K}(\boldsymbol{\varepsilon})$;

Option 3 (K0): $\mathbf{M}_p = \mathbf{K}_0 = \mathbf{L}_0\mathbf{L}_0^T$, the complete Cholesky decomposition of $\mathbf{K}_0$.

The first option, denoted IC0, uses the IC of $\mathbf{K}_0$ as the preconditioning matrix for all the sample points. The apparent attractive feature is its computational efficiency as it is generated only once. It is argued, however, that its performance may not be sufficient when large scale random variations are present. The second choice, denoted SIC, seems desirable because a new IC decomposition is generated for $\mathbf{K}(\boldsymbol{\varepsilon})$ at each sample point $\boldsymbol{\varepsilon}$. The obvious disadvantage of this option is the relatively expensive computational costs involved in the generation of IC($\mathbf{K}(\boldsymbol{\varepsilon})$) at the sample points. It is this concern that is addressed in [9] which proposes to express IC($\mathbf{K}(\boldsymbol{\varepsilon})$) as a series of $\boldsymbol{\varepsilon}$ based on the polynomial chaos expansion [1] to reduce the computational costs. Note that both IC and SIC decompositions may fail when the diagonal terms become negative during the decomposition. Although artificial amendments can be made to resume the procedure, the resulting decomposition often does not perform well.

The third option, denoted K0, uses the complete Cholesky decomposition of $\mathbf{K}_0$ as the preconditioning matrix. This option is highly recommended in the present work supported by the analysis conducted in Section 2 for the following reasons: (1) unlike the first two schemes, this option always achieves a good convergence regardless of the properties of $\mathbf{K}_0$, unless the random variation is very high and (2) the computational procedure involved in the decomposition is very stable while the other IC decompositions can fail. Note that the K0 scheme with further improvement has already been adopted in [6]. In Section 5, numerical examples will be provided to compare the performance of these three preconditioning schemes.

Generally a good initial approximation can also reduce the number of PCG iterations. However, it may not be easy to provide such an approximation apart from some special cases, as will be further discussed later. If no other information is available, the deterministic solution $\mathbf{u}_d$ may always be considered as a fairly good candidate.

Note that the above discussion is applied to both one and multiple random variable cases.

3.2. Exploitation of spatial proximity of Monte Carlo points

When only one random variable ε is present, Eq. (1) is reduced to

$$(\mathbf{K}_0 + \varepsilon\mathbf{K})\mathbf{u}(\varepsilon) = \mathbf{b} \tag{23}$$

Suppose that the total number of Monte Carlo samples required is M_c and all the sample points, $\mathcal{M} = \{\varepsilon_1^s, \ldots, \varepsilon_{M_c}^s\}$, for the random variable are generated before the solution of Eq. (23) is performed.

Box 1–Standard Preconditioned Conjugate Gradient Algorithm: $\mathbf{x} = \mathrm{PCG}(\mathbf{K}, \mathbf{b}, \mathbf{x}_0, \mathbf{M}_p, \tau, \mathit{maxite})$

- Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{K}\mathbf{x}_0$. If $\|\mathbf{r}_0\| < \tau\|\mathbf{b}\|$ return with $\mathbf{x} = \mathbf{x}_0$; else compute initial search direction $\mathbf{p}_0 = \mathbf{M}_p^{-1}\mathbf{r}_0$
- For $i = 0, 1, 2, \ldots$ *maxite*:
  1. Update solution: $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\mathbf{p}_i$ where $\alpha_i = \frac{\mathbf{p}_i^T\mathbf{r}_i}{\mathbf{p}_i^T\mathbf{K}\mathbf{p}_i} = \frac{\mathbf{r}_i^T\mathbf{M}_p^{-1}\mathbf{r}_i}{\mathbf{p}_i^T\mathbf{K}\mathbf{p}_i}$
  2. Update residual: $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i\mathbf{K}\mathbf{p}_i$
  3. Check convergence: if $\|\mathbf{r}_{i+1}\| < \tau\|\mathbf{b}\|$, return with $\mathbf{x} = \mathbf{x}_{i+1}$
  4. Compute new search direction: $\mathbf{p}_{i+1} = \mathbf{M}_p^{-1}\mathbf{r}_{i+1} + \beta_i\mathbf{p}_i$ where $\beta_i = -\frac{\mathbf{p}_i^T\mathbf{K}\mathbf{M}_p^{-1}\mathbf{r}_{i+1}}{\mathbf{p}_i^T\mathbf{K}\mathbf{p}_i} = \frac{\mathbf{r}_{i+1}^T\mathbf{M}_p^{-1}\mathbf{r}_{i+1}}{\mathbf{r}_i^T\mathbf{M}_p^{-1}\mathbf{r}_i}$

Normally these points bear no spatial sequence within the 1-D interval $\mathcal{D}$ while generated. However, when they are *sorted* in ascending order, their spatial relationship become apparent. Therefore, these sample points in the set $\mathcal{M}$ can be assumed to be in order

$$\varepsilon_1^s < \varepsilon_2^s < \cdots < \varepsilon_{M_c}^s \tag{24}$$

This *reorganisation* of the sample points makes no difference to the standard Monte Carlo simulation, but it is crucial for our modified Monte Carlo approach proposed here.

Following the re-order, the key to the effective exploitation of the localisation is how to provide a better initial guess for the current sample if the solutions at previous or neighbouring samples are known. This can be approached in two slightly different ways.

First, suppose that the solution $\mathbf{u}(\varepsilon)$ to (23) at ε is obtained and the solution $\mathbf{u}(\varepsilon+\Delta\varepsilon)$ at $\varepsilon+\Delta\varepsilon$ is sought:

$$[\mathbf{K}_0+(\varepsilon+\Delta\varepsilon)\mathbf{K}]\mathbf{u}(\varepsilon+\Delta\varepsilon)=\mathbf{b} \tag{25}$$

By employing a first order Taylor series approximation, $\mathbf{u}(\varepsilon+\Delta\varepsilon)$ has the following approximation

$$\mathbf{u}(\varepsilon+\Delta\varepsilon)\approx\mathbf{u}(\varepsilon)+\frac{d\mathbf{u}(\varepsilon)}{d\varepsilon}\Delta\varepsilon \tag{26}$$

where the gradient $d\mathbf{u}/d\varepsilon$ can be attained by the direction differentiation of (23) as

$$\frac{d\mathbf{u}(\varepsilon)}{d\varepsilon}=-(\mathbf{K}_0+\varepsilon\mathbf{K})^{-1}\mathbf{K}\mathbf{u}(\varepsilon) \tag{27}$$

However, to evaluate the gradient is equivalent to the solution of a new linear system, and therefore using (26) to find an approximation to $\mathbf{u}(\varepsilon+\Delta\varepsilon)$ is not very attractive. Nevertheless, it is straightforward to compute the gradient at $\varepsilon=0$ since it is assumed that the Cholesky decomposition of $\mathbf{K}_0$ is available:

$$\left.\frac{d\mathbf{u}(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0}=-\mathbf{K}_0^{-1}(\mathbf{K}\mathbf{u}_d) \tag{28}$$

A better approach is to utilise additional information to approximate the gradient $d\mathbf{u}/d\varepsilon$. Now suppose that the solutions $\mathbf{u}(\varepsilon^s)$ at the sample points from ε_1^s to ε_i^s have been found and the aim is to use these results to obtain a high quality initial guess $\mathbf{u}_0(\varepsilon^s)$ at ε_{i+1}^s for the corresponding PCG iterations. This is basically an interpolation/extrapolation problem, and many standard numerical schemes are available. Here the following two schemes are proposed for use:

Scheme 1. *Linear interpolation/extrapolation*: In this scheme, the gradient $d\mathbf{u}/d\varepsilon$ at ε_i^s is approximated by a forward Euler difference as

$$\left.\frac{d\mathbf{u}(\varepsilon)}{d\varepsilon}\right|_{\varepsilon=\varepsilon_i^s}\approx\frac{1}{\varepsilon_i^s-\varepsilon_{i-1}^s}[\mathbf{u}(\varepsilon_i^s)-\mathbf{u}(\varepsilon_{i-1}^s)] \tag{29}$$

then it follows that

$$\mathbf{u}_0(\varepsilon_{i+1})\approx\mathbf{u}(\varepsilon_i)+\frac{\varepsilon_{i+1}-\varepsilon_i^s}{\varepsilon_i^s-\varepsilon_{i-1}^s}[\mathbf{u}(\varepsilon_i^s)-\mathbf{u}(\varepsilon_{i-1}^s)] \tag{30}$$

Scheme 2. *Cubic spline interpolation/extrapolation*: In this scheme, a cubic spline is generated first to interpolate all the existing solutions (componentwise) and then is used to compute an initial approximation solution at ε_{i+1}^s.

The first linear scheme is very simple to implement. Its performance is, however, dependent on the spacings $\varepsilon_i^s-\varepsilon_{i-1}^s$ and $\varepsilon_{i+1}^s-\varepsilon_i^s$. At the expense of more computational costs, the spline scheme usually provides better approximation if the spacings between the sample points are all fairly uniform. However, the random samples cannot always satisfy this requirement and occasionally behave irrationally. Particularly when $\varepsilon_i^s-\varepsilon_{i-1}^s$ is significantly larger or smaller than the previous spacings, the predicated initial solution at ε_{i+1} is normally poor, while on the other hand the linear scheme is not sensitive to the spacings. In general, as will be demonstrated numerically in Section 5, the spline scheme exhibits better performance when a relatively small number of sample points are present, while the linear scheme behaves satisfactory when a large number of samples are present.

In both schemes, the first two solutions need to be computed differently. The first solution $\mathbf{u}(\varepsilon_1^s)$ may be obtained probably with $\mathbf{u}_d$ as the initial guess, which is then used as the initial approximation for the next solution $\mathbf{u}(\varepsilon_2^s)$. When the sample value interval $\mathcal{D}$ of the random variable ε is (roughly) symmetric about the origin, which is true for most cases, there exists an alternative sample processing strategy. The sorted sample sequence (24) can be split into two parts so that the samples in the first part are all smaller than zero and larger than zero in the second part:

$$\varepsilon_1^s < \cdots < \varepsilon_j^s < 0 < \varepsilon_{j+1}^s < \cdots < \varepsilon_{M_c}^s \tag{31}$$

Now treat $\mathbf{u}_d=\mathbf{u}(0)$ as the first solution, and since the gradient $d\mathbf{u}/d\varepsilon$ at $\varepsilon=0$ is available from (29), (26) can be used to compute the initial guess for ε_j to obtain the solution $\mathbf{u}(\varepsilon_j)$. Then the solutions from ε_{j-1} down to ε_1 can be found following the above two interpolation/extrapolation schemes. The second part can be processed, starting from ε_{j+1} until ε_{M_c}, in a similar manner. This latter strategy is numerically proved to be slightly more effective.

The main steps involved in the Directed Monte Carlo approach proposed above are summarised in Box 2, where the sample points are assumed to be processed from ε_1. Note that in the

Box 2–Directed Monte Carlo simulation—one random variable: DMC1($\mathbf{K}_0, \mathbf{K}, \mathbf{b}, f, \mathscr{M}, M_c, p(\varepsilon)$)

- If $\mathscr{M}$ is empty, generate M_c random samples $\mathscr{M}=\{\varepsilon_1^s,\ldots,\varepsilon_{M_c}^s\}$ according to the given PDF $p(\varepsilon)$; Sort the samples in $\mathscr{M}$ in ascendent order.
- Compute $\mathbf{u}_d=\mathbf{K}_0^{-1}\mathbf{b}$ (and $\mathbf{u}_0'=-\mathbf{K}_0^{-1}(\mathbf{K}\mathbf{u}_d)$ if required).
- Set initial preconditioning matrix $\mathbf{M}_p$ (and update it for the second point if required).
- Compute the first two solutions:$\mathbf{u}_1=\mathrm{PCG}(\mathbf{K}_0+\varepsilon_1^s\mathbf{K},\mathbf{b},\mathbf{u}_d,\mathbf{M}_p,\tau,\mathit{maxite})\mathbf{u}_2=\mathrm{PCG}(\mathbf{K}_0+\varepsilon_2^s\mathbf{K},\mathbf{b},\mathbf{u}_1,\mathbf{M}_p,\tau,\mathit{maxite})$
- For$i=3,\ldots M_c$:vbox
  1. Update preconditioning matrix $\mathbf{M}_p$ if required
  2. Compute initial guess $\mathbf{u}_0(\varepsilon_i^s)$ based on the linear or spline scheme
  3. Obtain solution:$\mathbf{u}_i=\mathrm{PCG}(\mathbf{K}_0+\varepsilon_i^s\mathbf{K},\mathbf{b},\mathbf{u}_i,\mathbf{u}_0(\varepsilon_i^s),\mathbf{M}_p,\tau,\mathit{maxite})$
- Calculate the expectation:$E[f(\mathbf{u})]=(1/M_c)\sum_{i=1}^{M_c}f(\mathbf{u}_i)$

current DMC approach, the random variable can have any probability distribution.

Note that the above scheme is further enhanced in [12] whereby after the solution is obtained at each MC point, the preconditioning matrix is adaptively updated by a Rank-one matrix which is purely dependent on the incremental solution related to the previous solution. This additional step further increase the computational efficiency of the Directed Monte Carlo approach.

4. Modified Monte Carlo simulation: multiple random variables

The directed Monte Carlo solution strategy proposed in the previous section cannot be readily extended to multiple random variables. The main challenge is the fact that spatial points in an n-dimensional domain do not have an in-built '*order*' as in the one dimensional case.

In this section, this difficulty is overcome by further modifying the standard Monte Carlo method with the introduction of a *hyper-spherical transformation* in the simulation. To this end, it is required that the random variables ε_i be the standard Gaussian distribution. Note that such an assumption imposes no great restriction to problems that can be considered, since many random phenomena in practical applications can be well approximated by the standard Gaussian process. Now all the variables have the following PDF:

$$p(\varepsilon) = \frac{1}{\sqrt{2\pi}}\mathbf{e}^{-\varepsilon^2/2} \tag{32}$$

and their joint probability distribution is

$$p(\boldsymbol{\varepsilon}) = \prod_{i=1}^{n} p(\varepsilon_i) = \frac{1}{(2\pi)^{n/2}}\mathbf{e}^{-\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}/2} \tag{33}$$

Without imposing the restriction to the sample value, the sampling domain of $\boldsymbol{\varepsilon}$ is the whole real space $\mathcal{R}^n$, and possesses an in-built hyper-spherical symmetry. The expectation of an arbitrary function f of $\mathbf{u}(\boldsymbol{\varepsilon})$ can then be evaluated as

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \int_{\mathbb{R}^n} f(\mathbf{u}(\boldsymbol{\varepsilon}))\frac{1}{(2\pi)^{n/2}}\mathbf{e}^{-\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}/2}\,dV \tag{34}$$

Now define an n-dimensional hyper-sphere of unit radius, termed 'unit n-sphere' in this work and denoted by $\mathbb{S}_n$, on which any point $\overline{\boldsymbol{\varepsilon}} = \{\overline{\varepsilon}_1, \overline{\varepsilon}_2, \ldots, \overline{\varepsilon}_n\}$ satisfies

$$\overline{\boldsymbol{\varepsilon}}^T\overline{\boldsymbol{\varepsilon}} = \overline{\varepsilon}_1^2 + \overline{\varepsilon}_2^2 + \cdots + \overline{\varepsilon}_n^2 = 1 \tag{35}$$

The total 'surface area' of this unit n-sphere is

$$A_n = \frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} \tag{36}$$

where Γ is the Gamma function.

4.1. Hyper-spherical transformation

Given a random vector $\boldsymbol{\varepsilon}$ in $\mathbb{R}^n$, introduce a new random variable ε_r and a random vector $\overline{\boldsymbol{\varepsilon}}$ on $\mathbb{S}_n$ as follows:

$$\varepsilon_r^2 = \varepsilon_1^2 + \varepsilon_2^2 + \cdots + \varepsilon_n^2 = \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} \tag{37}$$

and

$$\overline{\boldsymbol{\varepsilon}} = \boldsymbol{\varepsilon}/\varepsilon_r \tag{38}$$

Then $\boldsymbol{\varepsilon}$ can be expressed by ε_r and $\overline{\boldsymbol{\varepsilon}}$ as

$$\boldsymbol{\varepsilon} = \varepsilon_r\,\overline{\boldsymbol{\varepsilon}} \tag{39}$$

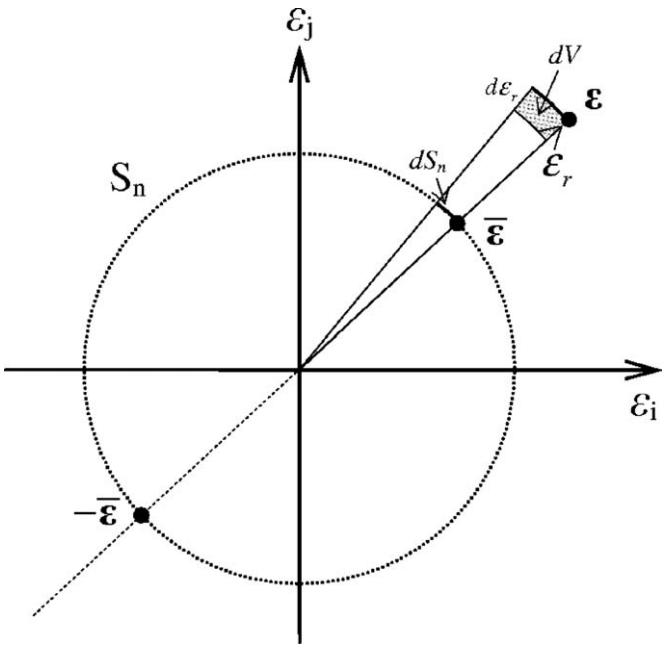

Fig. 1. 2D illustration of hyper-spherical transformation.

This is basically a general hyper-spherical transformation where ε_r and $\overline{\boldsymbol{\varepsilon}}$ represent, respectively, the 'magnitude' or 'radius' and the unit 'direction' of $\boldsymbol{\varepsilon}$, as illustrated in Fig. 1 in a polar coordinate sense. In other words, ε_r now represents the collective variation scale of all the random variables involved. Thus any random vector in the random space $\mathbb{R}^n$ can be generated by randomly giving a global variation scale ε_r and a point $\overline{\boldsymbol{\varepsilon}}$ on $\mathbb{S}_n$ with correct probability distributions. Obviously, ε_r and $\overline{\boldsymbol{\varepsilon}}$ are *independent* and $\overline{\boldsymbol{\varepsilon}}$ is *uniformly distributed* on $\mathbb{S}_n$ with a constant PDF

$$\overline{p}(\overline{\boldsymbol{\varepsilon}}) = 1/A_n \tag{40}$$

For ε_r, as it is well known that ε_r^2 is a χ^2-distribution:

$$\chi^2(x,n) = \frac{1}{2^{n/2}\Gamma(n/2)}x^{n/2-1}e^{-x/2} \quad (x>0) \tag{41}$$

the PDF of ε_r can be readily given as

$$p_r(\varepsilon_r) = \frac{1}{2^{n/2-1}\Gamma(n/2)}\,\varepsilon_r^{n-1}\mathbf{e}^{-\varepsilon_r^2/2} \quad (\varepsilon_r>0)$$

As $\varepsilon_r \in \mathcal{R}^+$, where $\mathcal{R}^+$ denotes the space formed by zero and all positive real numbers, the introduced hyper-spherical transformation (39) permits the following decomposition of the original 'Cartesian' space $\mathbb{R}^n$

$$\mathbb{R}^n = \mathcal{R}^+ \times \mathbb{S}_n \tag{42}$$

Without giving a detailed derivation, the volume element dV in the random space $\mathbb{R}^n$ has the following relationship with an infinitesimal 'line' element $d\varepsilon_r$ on the 'radial' direction and an infinitesimal 'surface' element dS_n on $\mathbb{S}_n$:

$$dV = \varepsilon_r^{n-1}\,d\varepsilon_r\,dS_n \tag{43}$$

With the above preparations, the expectation of the function $f(\mathbf{u}(\boldsymbol{\varepsilon}))$ can now be cast in the hyper-spherical space as

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \int_{\mathbb{S}_n}\int_{\mathcal{R}^+} f[\mathbf{u}(\varepsilon_r,\overline{\boldsymbol{\varepsilon}})]\frac{1}{(2\pi)^{n/2}}\mathbf{e}^{-\varepsilon_r^2/2}\,\varepsilon_r^{n-1}\,d\varepsilon_r\,dS_n \tag{44}$$

which can be further simplified in terms of the probability density functions $p_r(\varepsilon_r)$ and $\overline{p}(\overline{\boldsymbol{\varepsilon}})$:

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \int_{\mathbb{S}_n}\overline{p}(\overline{\boldsymbol{\varepsilon}})\left\{\int_{\mathcal{R}^+} f[\mathbf{u}(\varepsilon_r,\overline{\boldsymbol{\varepsilon}})]p_r(\varepsilon_r)\,d\varepsilon_r\right\}dS_n \tag{45}$$

and finally

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \int_{\mathbb{S}_n} f_r(\overline{\boldsymbol{\varepsilon}})\overline{p}(\overline{\boldsymbol{\varepsilon}})\, dS_n \tag{46}$$

where

$$f_r(\overline{\boldsymbol{\varepsilon}}) = \int_{\mathcal{R}^+} f[\mathbf{u}(\varepsilon_r, \overline{\boldsymbol{\varepsilon}})] p_r(\varepsilon_r)\, d\varepsilon_r \tag{47}$$

It is also very important to highlight that, for a given vector $\overline{\boldsymbol{\varepsilon}}$ in $\mathbb{S}_n$, $\mathbf{u}(\varepsilon_r, \overline{\boldsymbol{\varepsilon}})$ is the solution to the following linear equations with one random variable ε_r:

$$(\mathbf{K}_0 + \varepsilon_r \mathbf{K}_{\overline{\varepsilon}})\mathbf{u}(\varepsilon_r, \overline{\boldsymbol{\varepsilon}}) = \mathbf{b} \tag{48}$$

in which

$$\mathbf{K}_{\overline{\varepsilon}} = \overline{\varepsilon}_1\mathbf{K}_1 + \overline{\varepsilon}_2\mathbf{K}_2 + \cdots + \overline{\varepsilon}_n\mathbf{K}_n$$

is a deterministic matrix. Eqs. (46)–(48) readily suggest a new Monte Carlo simulation procedure in which the solution strategy developed in the previous section for the system with one random variable can now play an important part.

4.2. Modified Monte Carlo simulation in hyper-spherical space

Now the Monte Carlo simulation can be undertaken in two different spaces as follows:

(i) Generate a set of the required number (M_s) of sample points on $\mathbb{S}_n$:

$$\mathcal{M}_s = \{\overline{\boldsymbol{\varepsilon}}_1, \ldots, \overline{\boldsymbol{\varepsilon}}_i, \ldots, \overline{\boldsymbol{\varepsilon}}_{M_s}\}$$

(ii) At each sample $\overline{\boldsymbol{\varepsilon}}_i \in \mathcal{M}_s$, compute

$$f_r(\overline{\boldsymbol{\varepsilon}}_i) = \int_{\mathcal{R}^+} f[\mathbf{u}(\varepsilon_r, \overline{\boldsymbol{\varepsilon}}_i)] p_r(\varepsilon_r)\, d\varepsilon_r \tag{49}$$

which can be performed effectively by employing the above one-dimensional Directed Monte Carlo simulation using M_r samples from the set $\mathcal{M}_r = \{\varepsilon_r^1, \ldots, \varepsilon_r^k, \ldots, \varepsilon_r^{M_r}\}$ for ε_r:

$$f_r(\overline{\boldsymbol{\varepsilon}}_i) = \frac{1}{M_r}\sum_{k=1}^{M_r} f[\mathbf{u}(\varepsilon_r^k, \overline{\boldsymbol{\varepsilon}}_i)] \tag{50}$$

(iii) Then the final result can be computed as

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \frac{1}{M_s}\sum_{i=1}^{M_s} f_r(\overline{\boldsymbol{\varepsilon}}_i) \tag{51}$$

The above procedure is referred as to the Directed Monte Carlo simulation for multiple Gaussian variables.

Note that in the above discussion no coordinate system is explicitly specified for the computation associated with $\mathbb{S}_n$. Such a system may be needed when generating sample points on $\mathbb{S}_n$. In fact, any valid coordinate system can be chosen in principle. For instance, the so called hyper-spherical coordinate system, a generalisation of the 3D spherical coordinate system, can be used, in which the independent variables are $n-1$ 'angles' $\{\theta_1, \ldots, \theta_{n-1}\}$, and any point $\overline{\boldsymbol{\varepsilon}}$ on $\mathbb{S}_n$ can be expressed as

$$\overline{\boldsymbol{\varepsilon}} = \begin{pmatrix} \overline{\varepsilon}_1 \\ \overline{\varepsilon}_2 \\ \cdots \\ \overline{\varepsilon}_k \\ \cdots \\ \overline{\varepsilon}_{n-1} \\ \overline{\varepsilon}_n \end{pmatrix} = \begin{pmatrix} \cos\theta_1 \\ \sin\theta_1\cos\theta_2 \\ \cdots \quad \cdots \\ (\prod_{i=1}^{k-1}\sin\theta_i)\cos\theta_k \\ \cdots \quad \cdots \\ \sin\theta_1\cdots\sin\theta_{n-2}\cos\theta_{n-1} \\ \sin\theta_1\cdots\sin\theta_{n-2}\sin\theta_{n-1} \end{pmatrix} \tag{52}$$

where $\theta_i \in [0, \pi]$ for $i = 1, \ldots, n-2$ and $\theta_{n-1} \in [0, 2\pi]$. $\overline{\boldsymbol{\varepsilon}}$ will be uniformly distributed on $\mathbb{S}_n$ when all the angles are uniformly distributed in their value ranges.

However, the use of a coordinate system in $\mathbb{S}_n$ can be totally avoided. Suppose that the required number M_s of samples on $\mathbb{S}_n$ is larger than the required number M_r of the samples for ε_r. First generate M_s sample points of $\boldsymbol{\varepsilon}$ from which two sets of M_s samples for both ε_r and $\overline{\boldsymbol{\varepsilon}}$, $\mathcal{M}_r$ and $\mathcal{M}_s$, can be obtained following Eqs. (37) and (38). Then when computing $f_r(\overline{\boldsymbol{\varepsilon}}_i)$ at each sample point $\overline{\boldsymbol{\varepsilon}}_i \in \mathcal{M}_s$, randomly choose any M_r consecutive samples from $\mathcal{M}_r$ for ε_r.

Although it is assumed that $\varepsilon_r \in \mathcal{R}^+$, it can be extended to the whole real number axis $\mathcal{R}$ by modifying (37) as

$$\varepsilon_r = \pm\sqrt{\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}} \tag{53}$$

where the positive/negative sign can be chosen randomly. The corresponding PDF of $\varepsilon_r \in \mathcal{R}$ is

$$\overline{p}_r(\varepsilon_r) = \tfrac{1}{2}p_r(|\varepsilon_r|) \tag{54}$$

and the expectation of f is now of the form

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \int_{\mathbb{S}_n} \overline{f}_r(\overline{\boldsymbol{\varepsilon}})\overline{p}(\overline{\boldsymbol{\varepsilon}})\, dS_n \tag{55}$$

where

$$\overline{f}_r(\overline{\boldsymbol{\varepsilon}}) = \int_{\mathcal{R}} f[\mathbf{u}(\varepsilon_r, \overline{\boldsymbol{\varepsilon}})]\overline{p}_r(\varepsilon_r)\, d\varepsilon_r \tag{56}$$

It is clear from (48) that

$$\mathbf{u}(\varepsilon_r, \overline{\boldsymbol{\varepsilon}}) = \mathbf{u}(-\varepsilon_r, -\overline{\boldsymbol{\varepsilon}}) \tag{57}$$

Since if $\boldsymbol{\varepsilon} \in \mathbb{S}_n$, $-\boldsymbol{\varepsilon} \in \mathbb{S}_n$, and it is not difficult to prove that

$$\overline{f}_r(-\overline{\boldsymbol{\varepsilon}}) = \int_{\mathcal{R}} f[\mathbf{u}(\varepsilon_r, -\overline{\boldsymbol{\varepsilon}})]\overline{p}_r(\varepsilon_r)\, d\varepsilon_r = \int_{\mathcal{R}} f[\mathbf{u}(-\varepsilon_r, \overline{\boldsymbol{\varepsilon}})]\overline{p}_r(\varepsilon_r)\, d\varepsilon_r = \overline{f}_r(\overline{\boldsymbol{\varepsilon}}) \tag{58}$$

i.e. $\overline{f}_r(\overline{\boldsymbol{\varepsilon}})$ is symmetric about the origin, which thus leads to

$$E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = 2\int_{\mathbb{S}_n^+} \overline{f}_r(\overline{\boldsymbol{\varepsilon}})\overline{p}(\overline{\boldsymbol{\varepsilon}})\, dS_n \tag{59}$$

where $\mathbb{S}_n^+$ denotes any 'half' of the 'unit n-sphere' $\mathbb{S}_n$.

The benefits of extending ε_r to $\mathcal{R}$ are twofold: (1) to reduce the number of MC sample points required by half, thereby reducing the total MC simulation time and (2) to increase the number of

Box 3–Directed Monte Carlo simulation—multiple Gaussian variables: DMCn($\mathbf{K}_0, \{\mathbf{K}_i\}, \mathbf{b}, f, n, M_s, M_r$)

- Generate M_c Gaussian random vector samples $\mathscr{M}_s = \{\boldsymbol{\varepsilon}_1, \ldots, \boldsymbol{\varepsilon}_{M_s}\}$ and create two sets of sample points for $\mathscr{M}_r = \{\varepsilon_r^1, \ldots, \varepsilon_r^{M_s}\}$ and $\mathscr{M}_s = \{\overline{\boldsymbol{\varepsilon}}_1, \ldots, \overline{\boldsymbol{\varepsilon}}_{M_s}\}$ accordingly;
- Loop over all sample points $\overline{\boldsymbol{\varepsilon}}_i = \{\overline{\varepsilon}_1, \ldots, \overline{\varepsilon}_n\} \in \mathscr{M}_s$:
  1. Set $\mathbf{K}_{\overline{\varepsilon}}^i = \overline{\varepsilon}_1^i\mathbf{K}_1 + \cdots + \overline{\varepsilon}_n^i\mathbf{K}_n$
  2. Choose randomly M_r consecutive samples from $\mathscr{M}_r$ to form a subset $\mathscr{M}_r^s \subset \mathscr{M}_r$
  3. Compute: $\overline{f}_r(\overline{\boldsymbol{\varepsilon}}_i) = \mathrm{DMC1}(\mathbf{K}_0, \mathbf{K}_{\overline{\varepsilon}}^i, \mathbf{b}, f, \mathscr{M}_r^s, M_r, \overline{p}_r)$
- Calculate the expectation: $E[f(\mathbf{u}(\boldsymbol{\varepsilon}))] = \frac{1}{M_s}\sum_{i=1}^{M_s}\overline{f}_r(\overline{\boldsymbol{\varepsilon}}_i)$

sample points of ε_r used at each $\bar{\boldsymbol{\varepsilon}}$ to compute $\bar{f}_r$, thereby maximising the computational gain of the one-dimensional Directed Monte Carlo simulation. Another added benefit is that the important issue of fully understanding the eigenstructure of the general matrix $\mathbf{K}(\boldsymbol{\varepsilon})$ becomes more tractable, although this is outside the scope of the current work.

The DMC method for multiple random variables, however, has implications with regard to the solution accuracy in comparison with the standard Monte Carlo simulation, since both the Monte Carlo formulation and the sampling strategy are now essentially changed. This issue will be discussed further in conjunction with numerical simulations in the next section.

Finally, Box 3 outlines the main steps involved in the directed Monte Carlo simulation for multiple Gaussian random variables.

5. Numerical experiments

In this section, two examples will be used to provide a full assessment to several numerical aspects of the proposed directed Monte Carlo simulation methodology for the solution of Eq. (1). First the performance of PCG using different preconditioners and prediction schemes for initial approximations are investigated in terms of both PCG iterations and CPU costs mainly for one random variable cases. Then the solution accuracy of the DMC approach against the standard MC simulation in terms of the number of MC sample points is examined.

5.1. Problem descriptions

Both examples are elastic plane stress problems with a constant Poisson ratio ($\mu = 0.3$) and stochastic Young's modulus variations. Both Young's modulus are assumed to be stationary Gaussian stochastic fields which are normalised to have unit mean value E_m=1. The following two covariance functions $\mathbf{R}_1$ and $\mathbf{R}_2$ are given, respectively, for the two problems:

$$\mathbf{R}_1(\mathbf{x},\mathbf{y}) = 0.0729 \times \sigma_E^2 \mathbf{e}^{-(\mathbf{x}-\mathbf{y})^2/2.0^2} \tag{60}$$

$$\mathbf{R}_2(\mathbf{x},\mathbf{y}) = 0.0729 \times \sigma_E^2 \mathbf{e}^{-(\mathbf{x}-\mathbf{y})^2/0.3^2} \tag{61}$$

in which σ_E is a free parameter that can be used to vary the scale of the random variation. By employing the Karhunen–Loéve expansion with a Fourier series based solution scheme [11], each stochastic field of $\mathbf{E}$ is explicitly expanded into a finite function series as outlined in Section 2. The number of the terms, i.e. the mutually independent standard Gaussian random variables, retained is arbitrarily chosen in the present work without considering the accuracy of the expansion. The sample value range of each normalised Gaussian random variable is restricted to $[-3,3]$.

The finite element discretisations of the two problems, with a total number DOFs of 274 and 5986, respectively, are shown in Fig. 2. The maximum dimensions of the two problems in metres are 16×16 and 6×5, respectively.

The first example has a very small number of DOFs. It is used mainly for investigating the convergence features of the numerical techniques developed earlier. The small scale also makes it possible to examine the solution accuracy of the new method compared with the standard procedure within a reasonable time scale.

For this example, the first nine terms in the K–L expansion are considered, resulting in nine Gaussian random variables ε_i and associated matrices $\mathbf{K}_i$. The corresponding eigenvalue λ_i^E of each random variable ε_i and the two extreme eigenvalues of the matrix $\mathbf{K}_i$ are listed in Table 1 at the variance scale of $\sigma_E = 1.0$. It is clear that for the given covariance function, the eigenvalues of all the nine random variables have a similar value and that, except for $\mathbf{K}_2$ which is SPD, the remaining matrices are all indefinite, but their minimum and maximum eigenvalues are almost the same in magnitude.

The second example has a more complex structural configuration with in total 5640 elements and 5986 DOFs, much larger than the first example. The main purpose of this example is to further confirm the results obtained in the first example and particularly to investigate the efficiency of the proposed methodology in terms of CPU time costs. For this example, the first five stochastic variables and the corresponding matrices are involved in the computation. Their eigenvalues at the variance scale of $\sigma_E = 1.0$ are listed in Table 2, and similar features to the first example can be observed.

In both examples, a randomly generated loading vector $\mathbf{B}$ is considered.

Table 1
Eigenvalues of random variables/associated matrices: Example 1.

Variable/ Matrix no.	Eigenvalue of variable (λ^E)	Min eigenvalue of matrix (λ_1)	Max eigenvalue of matrix (λ_N)
1	0.08889	0.0439	0.1101
2	0.08483	−0.1108	0.1113
3	0.08483	−0.1107	0.1110
4	0.07884	−0.1088	0.1088
5	0.07884	−0.0988	0.1008
6	0.07048	−0.1001	0.1008
7	0.07048	−0.0991	0.0995
8	0.06206	−0.0942	0.0942
9	0.06206	−0.0884	0.0855

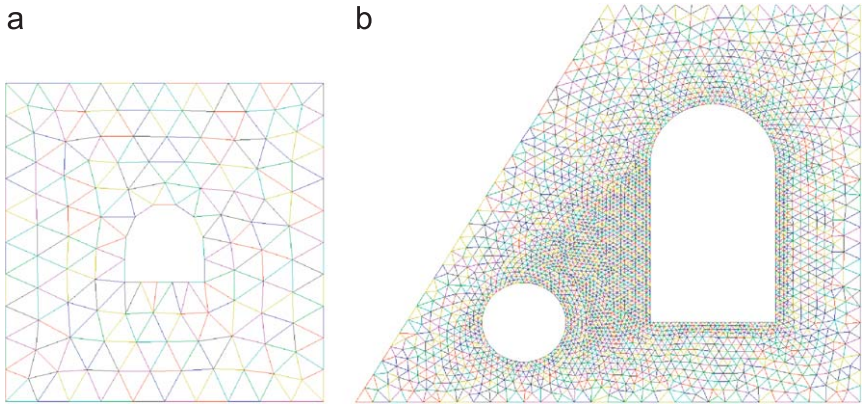

Fig. 2. Finite element discretisations of two examples: (a) Example 1 (274 DOFs) and (b) Example 2 (5986 DOFs).

Table 2
Eigenvalues of random variables and associated matrices: Example 2.

Variable/ Matrix no.	Eigenvalue of variable (λ^E)	Min eigenvalue of matrix (λ_1)	Max eigenvalue of matrix (λ_N)
1	0.02211	0.0022371	0.0105175
2	0.02091	−0.0100223	0.0099748
3	0.02091	−0.0085306	0.0098457
4	0.02062	−0.0099338	0.0107368
5	0.02062	−0.0098814	0.0107376

5.2. Performance of DMC and the associated numerical techniques

The aim of the first investigation is to use the two examples to establish the convergence properties of PCG, in the context of the DMC method for one random variable, when employing the different associated numerical techniques for solving the linear system of equations of (48). The equations are created as follows: A number of random variables and the corresponding matrices are arbitrarily selected from all the matrices available in each example, and a random vector $\overline{\boldsymbol{\varepsilon}}$ is given randomly to construct $\mathbf{K}(\overline{\boldsymbol{\varepsilon}})$. M_r sample values for ε_r are generated following the procedure outlined in the previous section.

5.2.1. Convergence of preconditioning and initial approximation schemes in PCG

The first batch of tests uses Example 1 to assess the performance of PCG with different preconditioning and initial approximation prediction schemes in terms of PCG iterations.

The average numbers of iterations required to achieve the prescribed convergence of PCG per MC sample are recorded in Tables 3 and 4 for the following different combinations of the integrated numerical techniques and parameters in DMC: (1) the number of random matrices included, n; (2) the variance scale σ_E; (3) the three preconditioning schemes; (4) the two interpolation/extropolation schemes for the prediction of the initial approximation; (5) the given convergence tolerance τ; and (5) different numbers of Monte Carlo samples $M_r = 10$, 24, 50, and 100 used for ε_r. Note that the same number of negative and positive samples are used and these two sets of samples are separately processed starting from the smallest absolute value of the samples as outlined in Section 3.2.

Tables 3 and 4 list the results for n=3 and 9, respectively, with six cases included in total. For each case, the performance of the standard MC simulation with PCG as the solver using the three different preconditioning schemes is also included for comparison as a benchmark.

In Table 3 (n=3), three matrices $\mathbf{K}_3$, $\mathbf{K}_6$ and $\mathbf{K}_9$ are used. Considering Case 1 ($\sigma_E = 1$, $\tau = 10^{-5}$) as a basic case, Cases 2 and 4 increase σ_E to 2.5 to represent large scale random variation conditions, and Cases 3 and 4 increase the solution accuracy to $\tau = 10^{-6}$. For both Cases 1 and 3 ($\sigma_E = 1$), the two minimum and maximum eigenvalues of $\mathbf{K}_r$ with $\mathbf{K}_0$ are, respectively, −0.1104 and 0.1139, while for Cases 2 and 4 ($\sigma_E = 2.5$), the two eigenvalues proportionally increase in magnitude to −0.2760 and 0.2847.

In Table 4, all the nine matrices are used, but only $\sigma_E = 1$ is considered since a further increase of σ_E will be very likely to result in an indefinite $\mathbf{K}(\overline{\boldsymbol{\varepsilon}})$. Two levels of solution accuracy are considered in Cases 5 and 6, respectively.

By examining the results presented in Tables 3 and 4, the following conclusions can be made:

(1) *Preconditioning schemes*: As expected, the incomplete Cholesky decomposition of $\mathbf{K}_0$, IC0, is the least effective scheme.

Table 3
Average PCG iterations per MC sample—$n = 3 : \{\mathbf{K}_3, \mathbf{K}_6, \mathbf{K}_9\}$.

Preconditioning	IC		SIC		K0	
Interpolation	Linear	Spline	Linear	Spline	Linear	Spline
Case 1	$\sigma_E = 1.0, \tau = 10^{-5}$, $\lambda \in [-0.1104, 0.1139]$					
M_c=10	16.0	13.7	15.4	12.7	3.00	2.40
M_c=24	13.6	8.54	13.5	8.75	1.66	1.33
M_c=50	8.44	9.50	8.00	9.58	1.34	1.20
M_c=100	7.22	11.5	7.02	10.9	1.06	1.32
Standard MC	26.5		26.3		4.75	
Case 2	$\sigma_E = 2.5, \tau = 10^{-5}$, $\lambda \in [-0.2760, 0.2847]$					
M_c=10	24.9	22.5	22.6	19.3	6.80	5.70
M_c=24	23.0	18.8	20.9	16.4	5.21	7.64
M_c=50	18.5	14.4	16.0	12.7	3.80	2.44
M_c=100	16.0	14.3	13.8	12.2	3.13	2.50
Standard MC	30.1		27.2		8.63	
Case 3	$\sigma_E = 1.0, \tau = 10^{-6}$, $\lambda \in [-0.1104, 0.1139]$					
M_c=24	19.7	12.4	19.1	12.4	3.20	1.67
M_c=50	16.3	10.1	15.3	9.76	2.24	1.30
M_c=100	14.1	11.1	13.7	11.3	1.90	1.43
Standard MC	28.7		28.6		5.68	
Case 4	$\sigma_E = 2.5, \tau = 10^{-6}$, $\lambda \in [-0.2760, 0.2847]$					
M_c=24	27.3	21.5	24.3	19.0	7.13	4.91
M_c=50	23.7	16.7	20.6	14.8	5.34	4.18
M_c=100	22.4	15.0	20.0	13.1	4.62	2.79
Standard MC	32.5		29.3		10.2	

Table 4
Average PCG iterations per MC sample—$n = 9 : \{\mathbf{K}_2, \ldots, \mathbf{K}_{10}\}$.

Preconditioning	IC		SIC		K0	
Interpolation	Linear	Spline	Linear	Spline	Linear	Spline
Case 5	$\sigma_E = 1.0, \tau = 10^{-5}$, $\lambda \in [-0.1104, 0.1139]$					
M_c=10	20.1	17.9	18.6	16.4	4.60	3.90
M_c=24	16.2	12.0	14.3	11.3	3.25	2.46
M_c=50	14.7	11.1	12.9	9.98	2.81	2.14
M_c=100	11.5	12.0	10.2	11.2	2.31	1.90
Standard MC	28.6		26.8		7.66	
Case 6	$\sigma_E = 2.5, \tau = 10^{-5}$, $\lambda \in [-0.1104, 0.1139]$					
M_c=10	25.0	21.9	23.2	20.4	6.00	5.00
M_c=24	22.5	15.3	21.2	15.2	4.50	2.96
M_c=50	21.4	13.8	19.8	12.7	4.28	2.50
M_c=100	19.1	12.9	17.4	12.5	3.69	2.14
Standard MC	30.9		28.9		9.17	

Although the 'stochastic' preconditioning scheme, SIC, can normally enhance the performance but the improvement is marginal and in general no more than 15% of reduction in iterations is achieved in all the test cases. On the contrary, the K0 scheme using the complete Cholesky decomposition of $\mathbf{K}_0$ exhibits a very good performance, with a reduction of iterations ranging from around 3 to 10 times compared to both IC0 and SIC schemes, and a typical reduction factor is around 5 or 6. A better performance of the K0 scheme is achieved at relatively smaller variation scales (Cases 1 and 3) and degrades slightly at larger variations (Cases 2 and 4), while the increase of the level of solution accuracy requirement τ makes the comparison more favourable to the K0 scheme. These results very positively support our earlier analytical analysis on the eigenvalue properties of the

stochastic matrix $\mathbf{K}(\varepsilon)$ made in Section 2.2. It is highlighted that the above conclusion is applied not only to one random variable but general cases. It is also noticed that the K0 preconditioning scheme can also be further enhanced by, for instance, the use of the Newmann expansion of Eq. (23) as proposed in [6].

(2) *Initial approximation schemes*: The ability to provide a good initial approximation for PCG solvers is the key to the success of the proposed DMC method and this is confirmed by all the test cases. In fact, compared with the standard MC method, the current PCG with an initial approximation achieves a reduction of iterations by 2–4 times, depending on the number of Monte Carlo samples used. Generally speaking, the required PCG iterations decrease with increase of the number of MC samples. It is possible that for a sufficiently large M_r the average iterations may be reduced to less than 1, thereby achieving even a greater improvement over the standard MC method. For the two interpolation/extrapolation approaches proposed, the spline scheme outperforms, in most cases, the linear scheme by an amount ranging from 10% up to 40% in terms of iterations, but the scheme becomes less effective in several cases. This phenomenon is purely dependent on the distribution of MC sample points. When the samples are fairly evenly distributed, which is often the case when the number of samples is small, the spline extrapolation will be superior to the linear extrapolation. However, when some clusters occur in the samples, which is an inherent feature with a large number of random numbers, the accuracy of the spline extrapolation at points immediately after the clusters will suffer, leading to an increase of PCG iterations. Fig. 3 illustrates this phenomenon by showing the accuracy of the initial approximations provided by the two extrapolation schemes for two different numbers of samples $M_r = 24$ and M_r=50 in Case 1. Note that only the positive samples are used and the positions of these samples are marked along the x-axis, from which the irregular spacing pattern of the samples is clearly demonstrated. The defect of the spline scheme may be eliminated, however, by enhancing the standard spline interpolation algorithm for the current situation.

In summary, the proposed DMC can indeed significantly enhance the computational effectiveness of the Monte Carlo simulation for general one random variable problems. When equipped with the K0 preconditioning and linear/spline extrapolation schemes, it can typically achieve performance around 3 times faster than the standard Monte Carlo simulation when the number of samples involved is not too small.

5.2.2. CPU time cost comparisons

The above observations are further examined by the second batch of tests using the second example. All the five matrices associated with the five variables are considered, i.e. n=5. Table 5 lists the average number of PCG iterations per MC sample with the same test conditions as in the previous batch of tests except for the scales of variation σ_E which are 10 times larger.

In essence, the new tests confirm all the observations made in the previous tests for the three preconditioning schemes and the two extrapolation schemes. As a matter of fact, the K0 preconditioning performs even better as it converges over 10 times faster than the other two counterparts at both normal and larger variance scales (Cases 7 and 8). Also the average number of iterations of the K0 preconditioning is reduced to less than 1 at M_r=100 in Case 7. These reveal that the features of the proposed methodology are universal regardless of the finite element scale of problems to be modelled.

Although it is clear that the K0 preconditioning is superior to the other two preconditioning schemes in terms of PCG iterations, it is more costly at each iteration as the size of its Cholesky decomposition matrix $\mathbf{L}_0$ is larger then the other two IC decompositions, if ignoring the extra cost associated with the re-generation of a new incomplete decomposition for SIC at each sample. Therefore, it is necessary to examine the CPU time costs of

Table 5
Average PCG iterations per MC sample—Example 2 ($n = 5$).

Preconditioning	IC		SIC		K0	
Interpolation	Linear	Spline	Linear	Spline	Linear	Spline
Case 7	$\sigma_E = 10.0, \tau = 10^{-5}, \lambda \in [-0.0830, 0.1300]$					
M_c=10	41.9	36.4	39.6	34.9	2.50	2.20
M_c=24	33.5	29.8	31.5	28.7	1.95	1.18
M_c=50	16.7	26.2	15.2	23.8	1.20	1.12
M_c=100	9.89	34.6	9.84	32.8	0.77	1.30
Standard MC	77.74		74.90		4.14	
Case 8	$\sigma_E = 25.0, \tau = 10^{-5}, \lambda \in [-0.2175, 0.3250]$					
M_c=10	68.8	63.0	58.2	53.7	6.20	5.30
M_c=24	60.1	46.4	50.3	38.3	4.67	3.25
M_c=50	41.6	37.9	19.8	33.1	3.30	2.10
M_c=100	27.6	38.4	22.0	36.5	2.46	2.39
Standard MC	91.0		80.0		7.57	

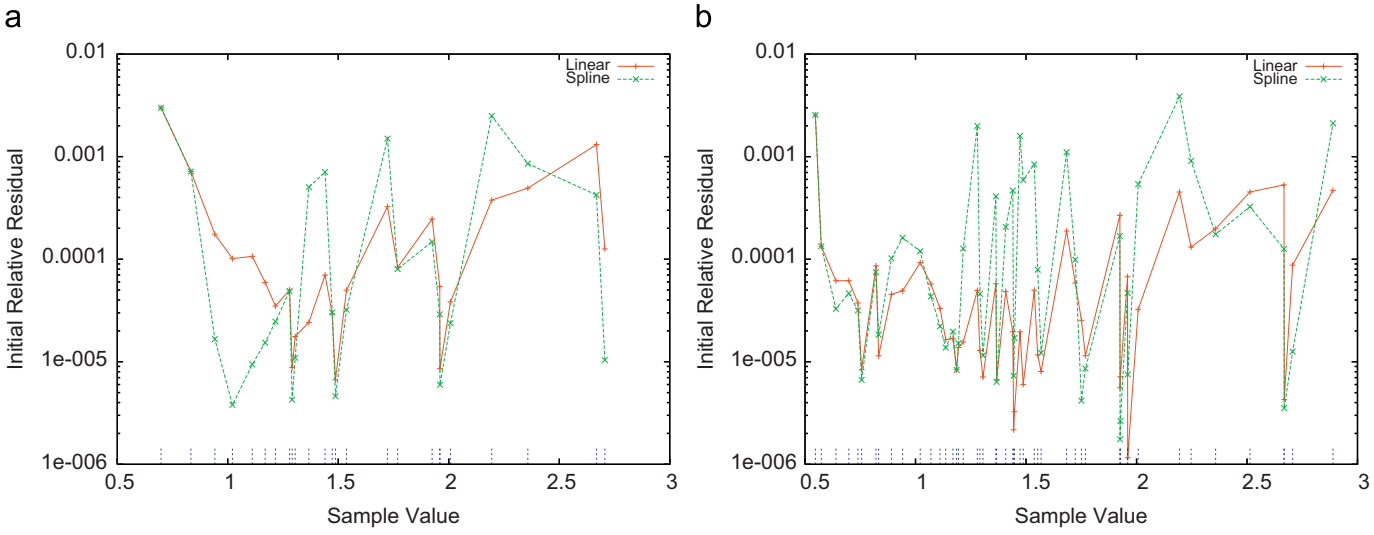

Fig. 3. Accuracy of initial approximations provided by linear and spline schemes in Case 1: (a) $M_r = 24$ and (b) $M_r = 50$.

all the numerical techniques at the same time. The relevant results are collected in Table 6.

It is evident that although the overall speed-up of the K0 preconditioning in terms of CPU costs over the other two schemes is smaller than the speed-up in iterations, it is still about 4 to 5 times faster, confirming that $\mathbf{K}_0$ is indeed a very effective and robust preconditioner, at least for not very large scale matrices. As expected, a small reduction in iterations of the SIC scheme over the IC0 scheme cannot compensate for the extra costs occurred for the generation of the IC decomposition at each MC sample, making it an unattractive option unless further developments are undertaken, such as the work in [9].

It is noticeable that the spline extrapolation scheme becomes slower than the linear scheme in all cases. This is due to the fact that in the standard B-spline interpolation algorithm currently implemented, all the previous solutions are used to predict the initial solution for the current MC sample. It is possible, however, to modify the B-spline formulation so that only a few solutions are required for the interpolation and extrapolation, thereby making its computational cost comparable to that of the linear interpolation/extrapolation.

Table 6
Total CPU times (s)—Example 2 (n=5).

Preconditioning	IC		SIC		K0	
Interpolation	Linear	Spline	Linear	Spline	Linear	Spline
Case 7	$\sigma_E=10.0, \tau=10^{-5}, \lambda\in[-0.0830, 0.1300]$					
M_c=10	9.64	8.56	16.9	15.1	1.29	1.26
M_c=24	18.6	17.0	32.7	35.1	2.37	1.93
M_c=50	19.9	32.3	40.1	64.7	3.75	5.60
M_c=100	23.9	91.1	66.2	169.	5.54	14.1
Standard MC	178.1		261.3		23.2	
Case 8	$\sigma_E=25.0, \tau=10^{-5}, \lambda\in[-0.2175, 0.3250]$					
M_c=10	15.73	14.64	24.13	23.26	3.17	3.10
M_c=24	33.43	26.51	52.18	47.17	6.03	5.72
M_c=50	48.95	45.56	86.64	87.81	10.26	10.65
M_c=100	66.78	101.2	147.7	189.2	17.47	26.21
Standard MC	210.0		283.2		34.73	

5.3. Solution accuracy of the DMC method

The proposed DMC simulation method for multiple Gaussian random variables utilises the hyper-spherical transformation to convert the original Monte Carlo simulation into two parts: a uniform distribution simulation on $\mathbb{S}_n$ and a one-dimensional χ^2-type distribution along the ε_r direction, representing the global variation scale. The previous tests have established that the DMC provides a very effective procedure to deal with the solution along the ε_r direction, i.e. the probability distribution of the solution at each point on $\mathbb{S}_n$ can be obtained efficiently. However, as the original goal is to solve Eq. (1) for any number of random variables, it is necessary to examine the solution accuracy of DMC for general cases in comparison with the standard MC simulation where the sampling is performed in real space $\mathcal{R}^n$.

The solution accuracy of DMC in terms of the total number of Monte Carlo points is assessed using Example 1. Following the procedure outlined in Section 4, for a given n, different numbers of Monte Carlo samples, M_s, are generated on $\mathbb{S}_n$. At each point $\bar{\boldsymbol{\varepsilon}}$, different numbers of sample points M_r are used for ε_r to determine $\bar{f}_r(\bar{\boldsymbol{\varepsilon}})$ (refer to (47)). In the simulation, the expectation and standard derivation of the total strain energy of the structure $\mathbf{b}^T\mathbf{u}(\bar{\boldsymbol{\varepsilon}})$, normalised by the deterministic energy $\mathbf{b}^T\mathbf{u}_d$, are considered.

The computed expectation (mean value) and standard derivation of the normalised strain energy via the total number of Monte Carlo points $M_c = M_s \times M_r$ up to 50,000 are depicted in Fig. 4 for the cases of n=9 with M_r=20, 40 and 100, respectively. Note that the corresponding numbers for M_s are 2500, 1250, and 500. The results computed by the standard Monte Carlo simulation, which corresponds to the special case of M_r=1, are also shown in the figure for comparison.

The figure illustrates that the best convergence is achieved by the standard MC simulation, while the convergence of the DMC solution appears to be slightly less smooth and the oscillation at the small sample numbers increases with the increase of M_r. This is not surprising as for a fixed M_c, an increase of M_r is equivalent to the concentration of Monte Carlo samples along fewer directions, therefore reducing the randomness of these samples. This suggests that the current hyper-spherical transformation may lose, to a limited degree, the unique advantage of the standard MC simulation when the number of MC samples used is not sufficiently large. It can be argued that the improved

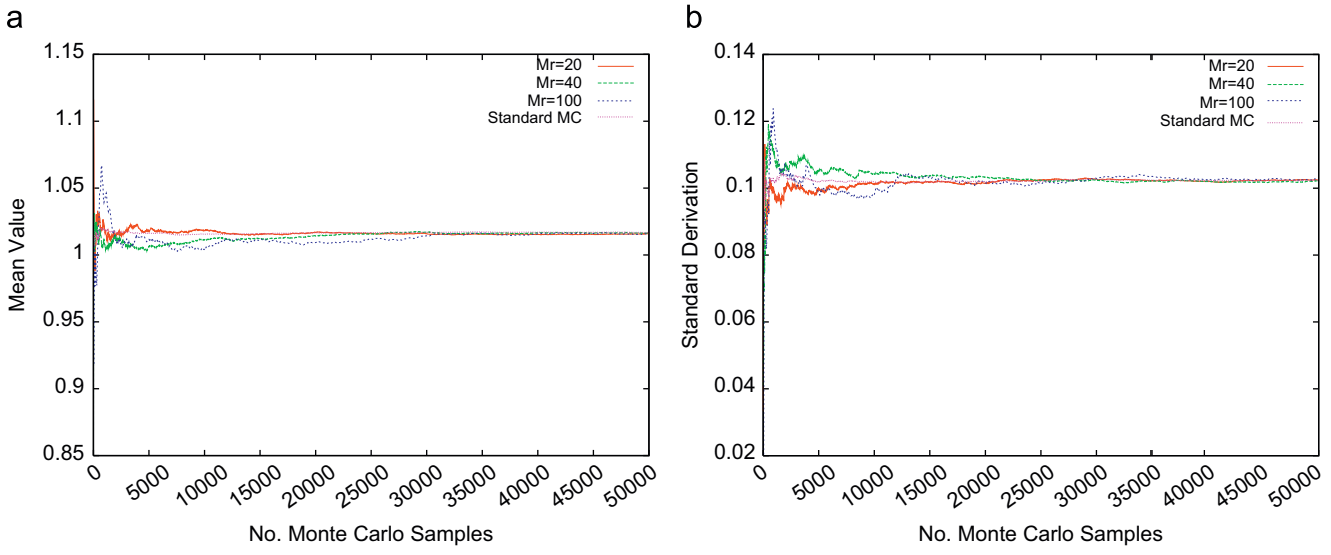

Fig. 4. Convergence histories of the normalised total strain energy versus the number of Monte Carlo samples: Example 1: (a) the mean value and (b) the standard derivation.

computational efficiency of DMC at each sample on $\mathbb{S}_n$ will permit the use of more sample points thereby compensating for the slight loss of solution accuracy.

However, it is more important to highlight that the hyperspherical transformation, together with the ability of effectively (and accurately) determining $\overline{f}_r(\overline{\varepsilon})$ at each point on $\mathbb{S}_n$, may provide a prospect of developing new simulation approaches within the general Monte Carlo solution framework that have potential to greatly improve the solution accuracy of the current DMC method. This is the line of research that is currently being pursued in order to further enhance the overall computational capability of the DMC method for general problems with multiple random variables.

6. Conclusions

The current work proposes a modified Monte Carlo simulation procedure, DMC, for solving a linear stochastic algebraic system of equations. The basic idea of DMC for one random variable cases is to order the Monte Carlo samples so that when the samples are processed in sequence the previous obtained solutions can be utilised to provide a high quality initial approximation for the current point thereby significantly accelerating the convergence of iterative solvers. In DMC, PCG plays a central role and the two essential numerical techniques crucial to the success of the method include preconditioning and initial proximation predictions. It is proposed on the basis of analytical analysis, and later confirmed numerically, that the deterministic matrix $\mathbf{K}_0$ can serve as a very effective preconditioning matrix. The numerical experiments conducted demonstrate that the proposed DMC can indeed significantly enhance the computational effectiveness of the Monte Carlo simulation for general one random variable problems. When employing the K0 preconditioning and linear/spline extrapolation schemes, DMC can typically perform around 3 times faster than the standard Monte Carlo simulation when the number of samples used is not too small.

The extension of the DMC method to multiply Gaussian random variable cases is realised by the adoption of a hyperspherical transformation whereby any n-dimensional random vector in $\mathcal{R}^n$ can be expressed by a random variable ε_r representing the global random variation scale, and a unit directional random vector on the unit 'n-sphere' $\mathbb{S}_n$. Such a transformation permits the Monte Carlo calculation of the solution to be undertaken as a one random variable case along the ε_r direction at each sample on $\mathbb{S}_n$. Although the overall computational costs of the Monte Carlo simulation can be reduced in this way, it is at the expense of slightly losing solution accuracy when the total number of samples used is not sufficiently large. This observation indicates the aspect to be further pursued in order to improve the overall performance of the proposed DMC method for general multiple random variable problems.

References

[1] R. Ghanem, P. Spanos, Stochastic Finite Elements—A Spectral Approach, revised ed., Dover Publications, New York, 2003.
[2] C.F. Li, Y.T. Feng, D.R.J. Owen, Explicit solution to the stochastic system of linear algebraic equations $(\alpha_1 A_1+\alpha_2 A_2+\cdots+\alpha_m A_m)x=b$, Computer Methods in Applied Mechanics and Engineering 195 (44–47) (2006) 6560–6576.
[3] G.P. Lepage, A new algorithm for adaptive multidimensional integration, Journal of Computational Physics 27 (1978) 192–203.
[4] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Random numbers in numerical recipes in C, The Art of Scientific Computing, second ed., Cambridge University Press, England, 1992 (Chapter 7).
[5] R. Ghanem, R.M. Kruger, Numerical solution of spectral stochastic finite element systems, Computer Methods in Applied Mechanics and Engineering 129 (1996) 289–303.
[6] M. Papadrakakis, V. Papadopoulos, Robust and efficient methods for stochastic finite element analysis using Monte Carlo simulation, Computer Methods in Applied Mechanics and Engineering 134 (3–4) (1996) 325–340.
[7] M. Papadrakakis, A. Kotsopulos, Parallel solution methods for stochastic finite element analysis using Monte Carlo simulation, Computer Methods in Applied Mechanics and Engineering 168 (1998) 305–320.
[8] R. Ghanem, Ingredients for a general purpose stochastic finite elements implementation, Computer Methods in Applied Mechanics and Engineering 168 (1999) 19–34.
[9] C. Desceliers, R. Ghanem, C. Soize, Polynomial chaos representation of a stochastic preconditioner, International Journal of Numerical Methods in Engineering 64 (2005) 618–634.
[10] M. Loéve, Probability Theory, vol. 2, fourth ed., Springer, Berlin, 1977.
[11] C.F. Li, Y.T. Feng, D.R.J. Owen, A semi-explicit solution to the Karhunen–Loéve expansions of wide-sense stationary stochastic fields, International Journal of Numerical Methods in Engineering 73 (13) (2008) 1942–1965.
[12] Y.T. Feng, Adaptive preconditioning of linear stochastic algebraic systems of equations, Communications in Numerical Methods in Engineering 23 (11) (2007) 1023–1034.
[13] Y.T. Feng, On the discrete dynamic nature of the conjugate gradient method, Journal of Computational Physics 211 (1) (2006) 91–98.
[14] M. Benzi, Preconditioning techniques for large linear systems: a survey, Journal of Computational Physics 182 (2002) 418–477.