INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2011; **87**:521–540
Published online 1 February 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.3116

Gaussian process emulators for the stochastic finite element method

F. A. DiazDelaO and S. Adhikari*, †

College of Engineering, Swansea University, Swansea, U.K.

SUMMARY

This paper explores a method to reduce the computational cost of stochastic finite element codes. The method, known as Gaussian process emulation, consists of building a statistical approximation to the output of such codes based on few training runs. The incorporation of emulation is explored for two aspects of the stochastic finite element problem. First, it is applied to approximating realizations of random fields discretized via the Karhunen–Loève expansion. Numerical results of emulating realizations of Gaussian and lognormal homogeneous two-dimensional random fields are presented. Second, it is coupled with the polynomial chaos expansion and the partitioned Cholesky decomposition in order to compute the response of the typical sparse linear system that arises due to the discretization of the partial differential equations that govern the response of a stochastic finite element problem. The advantages and challenges of adopting the proposed coupling are discussed. Copyright © 2011 John Wiley & Sons, Ltd.

Received 5 May 2010; Revised 11 November 2010; Accepted 17 November 2010

KEY WORDS: stochastic finite element method; Gaussian stochastic process; Bayesian statistics; Karhunen–Loève expansion; polynomial chaos; partitioned Cholesky decomposition

1. INTRODUCTION

The realistic modeling and prediction of engineering systems can be better achieved when their random aspects are taken into account. These might involve uncertain material properties, uncertain boundary conditions, and unknown manufacturing tolerances, among others. The associated uncertainty can be represented by stochastic coefficients in the system of stochastic partial differential equations that govern the system's response. Engineering systems can be quite complex to analyze, and addressing such complexity requires the employment of numerical algorithms with a sound theoretical basis. The finite element method (FEM) [1–5] has proved to be well suited for a large class of engineering problems, and thus it is a natural candidate to be extended in order to accommodate random functions. Such extension, known as the stochastic FEM (SFEM) [6–18], sets the framework to model the physical properties of a given engineering system as random fields. Broadly speaking, the SFEM aims at characterizing the global probabilistic structure of the system's random response. However, when a system with a large number of degrees of freedom is investigated, a computer code designed to study it may become prohibitively expensive to run. Such codes, as well as the underlying mathematical models are henceforth referred to as *simulators*. A simulator can be understood as a function $\eta : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ that, given an input $\mathbf{x}$, it returns an output $\mathbf{y} = \eta(\mathbf{x})$. There already exist several strategies that reduce the computational cost of

*Correspondence to: S. Adhikari, Chair of Aerospace Engineering, College of Engineering, Swansea University, Singleton Park, Swansea SA2 8PP, U.K.
†E-mail: S.Adhikari@swansea.ac.uk

expensive simulators by approximating their output. Based on different underlying methodologies, these strategies are referred to as metamodels, response surfaces, surrogates, auxiliary models, among many others [19]. They have been extensively applied in many engineering problems. For example, Craig *et al.* [20] performed variable screening and optimization in crashworthiness design based on a response surface methodology. Pérez *et al.* [21] solved non-linear optimization problems using quadratic response surfaces. Fan *et al.* [22] incorporated surrogate modeling to multi-objective optimization.

Yet, another type of metamodeling approach, which has been in constant development over the last two decades, is Gaussian process emulation. Based on the analysis and design of computer experiments [23, 24], and using concepts of Bayesian statistics, such technology consists in constructing a statistical approximation to the simulator's output, called an *emulator*. The main idea behind Gaussian process emulation is the following: A small and carefully selected set of code runs is treated as training data used to update the prior beliefs about the simulator's output. As it will be explained later, these prior beliefs take the form of a Gaussian stochastic process. After conditioning on the training data and updating, the mean of the resulting posterior distribution provides a fast approximation to the simulator's output at any untried input, whereas it returns the known value of the simulator at each of the initial runs. Gaussian process emulators (GPEs) have already been implemented in several scientific fields. For instance, Kennedy *et al.* [25] emulated a vegetation dynamic model, a model of ecosystem photosynthesis and water balance, and another one that estimates the UK's carbon budget. Challenor *et al.* [26] and Rougier [27] applied GPEs to climate prediction models. Xiong *et al.* [28] applied GPEs to a vehicle crash model used in safety optimization. Haylock and O'Hagan [29] emulated a model of doses to organs of the body after ingestion of a radioactive substance. DiazDelaO and Adhikari [30] used GPEs for structural dynamic analysis. Many other examples can be found in the literature.

The present paper explores the integration of GPEs into the SFEM in order to reduce the computational cost of studying engineering systems with random mechanical properties. The paper focuses on the problem of emulating random fields based on few evaluations of a simulator. First, discretized Gaussian and lognormal random fields are emulated. As it will be seen, the simulators of these random fields take inputs from a spatial domain that are trivially generated. Once it is shown that GPEs can efficiently approximate both Gaussian and non-Gaussian random fields, the problem of approximating the random response of an engineering system obtained by the SFEM is studied. The rationale is that since this response is itself a random field, it can be expected to be emulated satisfactorily. Nevertheless, the generation of the training runs upon which to build the GPE is not straightforward. Owing to this, an algorithm to obtain the necessary training runs is proposed. The paper is thus organized as follows. Section 2 provides an overview of the theory behind the implementation of GPEs. Section 3 explains how to discretize and emulate Gaussian and lognormal random fields. In Section 4, some SFEM techniques for the calculation of a system's response are reviewed with emphasis on the polynomial chaos expansion. A partitioned Cholesky algorithm coupled with a GPE is proposed to generate the training runs and an example in stochastic structural mechanics is presented. Section 5 offers some conclusions.

2. GAUSSIAN PROCESS EMULATORS

Suppose that n points, namely $\mathbf{x}_1, \ldots, \mathbf{x}_n$, are chosen in the input domain of the simulator $\eta(\cdot)$. Each of these is called a *design point*. The set $\{\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)\}$, resulting from the evaluation of $\eta(\cdot)$ in each design point, is called *training set*. According to O'Hagan [31], an emulator of $\eta(\cdot)$ should satisfy some minimal criteria:

1. Since by definition the output at each design point is known, the emulator should reproduce this output with no uncertainty.
2. At any $\mathbf{x}$ that is not a design point, the probability distribution provided by the emulator should produce a mean value that constitutes a plausible interpolation/extrapolation of the

DOI: 10.1002/nme

training data. The probability distribution around this predictive mean should also express the uncertainty about how the emulator might interpolate/extrapolate.

If $\eta(\cdot)$ is an expensive simulator, it can only be evaluated at a limited number of inputs. From the perspective of Bayesian statistics, $\eta(\cdot)$ is treated as a random variable in the sense that its value is unknown until the simulator is actually run. Thus, a characterization of the relationship between the input and the unknown output must first be selected. Subjective information about this relationship should be combined with objective information provided by the training set to predict the output of the simulator at untried inputs.

Begin by assuming that $\eta(\cdot)$ admits the following stochastic representation:

$$\eta(\mathbf{x}) = \mathbf{h}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\beta} + Z(\mathbf{x}) \tag{1}$$

where $\mathbf{h}(\cdot)$ is a vector of known functions of $\mathbf{x}$ and $\boldsymbol{\beta}$ is a vector of unknown coefficients. The function $Z(\cdot)$ is assumed to be a stochastic process with mean zero and some covariance function of $\mathbf{x}$. A common choice for $Z(\cdot)$ is the *Gaussian stochastic process*.

Definition 1 (*Gaussian stochastic process*)
Let $\mathscr{D} \subseteq \mathbb{R}^d$. Then $Z(\mathbf{x})$ for $\mathbf{x} \in \mathscr{D}$ is a Gaussian stochastic process if for any $n \geqslant 1$ and any choice $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathscr{D}$, the vector $[Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_n)]^{\mathrm{T}}$ has a multivariate normal distribution.

The Gaussian stochastic process is used in practice for much the same reasons that the normal distribution repeatedly appears in statistics: it is convenient, flexible and quite often realistic [32]. Let $\eta(\cdot)$ have a Gaussian process distribution with mean function $m(\cdot)$ and covariance $\sigma^2 C(\cdot, \cdot)$. This is expressed by

$$\eta(\cdot) \sim \mathscr{N}(m(\cdot), \sigma^2 C(\cdot, \cdot)) \tag{2}$$

If a linear structure of the form $m(\cdot) = \mathbf{h}(\cdot)^{\mathrm{T}}\boldsymbol{\beta}$ is chosen for the mean, then Equation (1) assumes that the random function $\eta(\cdot)$ deviates from the mean of its distribution following a Gaussian stochastic process.

Suppose $\eta(\cdot)$ is a smooth, continuous function of its inputs. It follows that if $\mathbf{x}$ and $\mathbf{x}'$ are close together, then the values of $Z(\mathbf{x})$ and $Z(\mathbf{x}')$ should also be close. It is therefore reasonable to think that the correlation between $\eta(\mathbf{x})$ and $\eta(\mathbf{x}')$ increases when the distance between $\mathbf{x}$ and $\mathbf{x}'$ decreases and vice versa. This assumption implies that each element of the training set provides considerable information about $\eta(\cdot)$ for inputs close to the corresponding design points. Hence, the uncertainty about the value of untried inputs decreases as the number of design points increases because the maximum distance from any design point is reduced (recall criterion 2 above). The use of this extra information is the feature that accounts for greater efficiency of the use of GPEs over Monte Carlo methods [31].

A common choice for covariance function, and the one that is adopted hereafter is

$$Cov(\eta(\mathbf{x}), \eta(\mathbf{x}')) = \sigma^2 e^{-(\mathbf{x}-\mathbf{x}')^{\mathrm{T}}\mathbf{B}(\mathbf{x}-\mathbf{x}')} \tag{3}$$

where $\mathbf{B}$ is a positive definite matrix of *smoothness parameters*. Observe that $C(\mathbf{x}, \mathbf{x}) = 1$ and that it decreases as the distance between two points increases, as required for a correlation function.

As a consequence of the above, the prior knowledge about $\eta(\cdot)$, given $\boldsymbol{\beta}$ and σ^2, is represented as having a Gaussian process distribution with linear mean function $\mathbf{h}(\cdot)^{\mathrm{T}}\boldsymbol{\beta}$ and covariance expressed by Equation (3). Mathematically

$$\eta(\cdot)|\boldsymbol{\beta}, \sigma^2 \sim \mathscr{N}(\mathbf{h}(\cdot)^{\mathrm{T}}\boldsymbol{\beta}, \sigma^2 C(\cdot, \cdot)) \tag{4}$$

This prior distribution contains subjective information about the relationship between the input and the unknown outputs. The next step is to update it by adding objective information. Suppose there are n design points that produce the vector of observations $\mathbf{y} = [\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)]^{\mathrm{T}}$ and the

DOI: 10.1002/nme

output of $\eta(\cdot)$ at an untried input $\mathbf{x}$ is to be estimated. Carrying out the suitable integration [29], it can be shown that the distribution that results from incorporating the observations $\mathbf{y}$ is of the form

$$\eta(\cdot)|\mathbf{y}, \sigma^2 \sim \mathcal{N}(m^{**}(\cdot), \sigma^2 C^{**}(\cdot,\cdot)) \tag{5}$$

Very conveniently, this posterior distribution is also a Gaussian process distribution. The complete expressions of $m^{**}(\cdot)$ and $C^{**}(\cdot,\cdot)$, as well as the procedure to obtain the posterior distribution (5) from the prior distribution (4), are given in the Appendix. There, it can be seen that $m^{**}(\cdot)$ does not include any terms involving $\eta(\cdot)$. Thus, it provides a fast approximation of $\eta(\mathbf{x})$ for any $\mathbf{x}$ (recall again criterion 2 above). The conditioning on σ^2 can also be eliminated. Haylock and O'Hagan [29] also show that

$$\frac{\eta(\mathbf{x}) - m^{**}(\mathbf{x})}{\widehat{\sigma}\sqrt{\dfrac{(n-q-2)C^{**}(\mathbf{x})}{n-q}}} \sim t_{n-q} \tag{6}$$

which is a t-distribution with $n-q$ degrees of freedom (not to be confused with the degrees of freedom in an FEM context). The expression for $\widehat{\sigma}$ and the meaning of q are also contained in the Appendix A.

In light of the above discussion, the algorithm to approximate a simulator $\eta(\cdot)$ is summarized in Algorithm 1.

Algorithm 1 Gaussian process emulation.

Input: Design points $\{\mathbf{x}_i\}_{i=1}^n$
Output: Predictive mean $m^{**}(\cdot)$
begin
1. Select n design points $\{\mathbf{x}_i\}_{i=1}^n$
2. Obtain the vector of observations $\mathbf{y} = [\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)]^{\mathrm{T}}$
3. Update the prior distribution (4) using $\mathbf{y}$ and obtain the posterior distribution (5)
4. Compute the predictive mean $m^{**}(\mathbf{x}) = \mathbf{E}[\eta(\mathbf{x})|\mathbf{y}]$ for any untried $\mathbf{x}$

end

3. DIRECT EMULATION OF RANDOM FIELDS

3.1. Random field discretization

Let $(\Theta, \mathscr{F}, \mathscr{P})$ be a probability space and let $\mathscr{L}^2(\Theta, \mathscr{F}, \mathscr{P})$ be the space of random variables with finite second moment. Thus, if $\mathscr{X}(\theta): \Theta \to \mathscr{D}_{\mathscr{X}} \subset \mathbb{R}$ is a random variable, then

$$\mathbf{E}[\mathscr{X}(\theta)] = \int_{\Theta} \mathscr{X}^2(\theta)\, \mathrm{d}\mathscr{P}(\theta) < +\infty \tag{7}$$

$\mathscr{L}^2(\Theta, \mathscr{F}, \mathscr{P})$ is a Hilbert space with respect to the inner product

$$\mathbf{E}[\mathscr{X}_1(\theta)\mathscr{X}_2(\theta)] = \int_{\Theta} \mathscr{X}_1(\theta)\mathscr{X}_2(\theta)\, \mathrm{d}\mathscr{P}(\theta) \tag{8}$$

A random field $\mathscr{H}(\mathbf{x}, \theta)$, with $\mathbf{x} \in \mathbb{R}^N$ and $\theta \in \Theta$, is a curve in $\mathscr{L}^2(\Theta, \mathscr{F}, P)$. For a given $\mathbf{x_0}$, $\mathscr{H}(\mathbf{x_0}, \theta)$ is a random variable; whereas for a given θ_0, $\mathscr{H}(\mathbf{x}, \theta_0)$ is a realization of the random field. A Gaussian random field is such that for any $n \geqslant 1$, the vector $[\mathscr{H}(\mathbf{x}_1, \theta_0), \ldots, \mathscr{H}(\mathbf{x}_n, \theta_0)]^{\mathrm{T}}$ is Gaussian. Moreover, it is *homogeneous* if its mean $\mu(\mathbf{x}, \theta_0)$ and variance $\varsigma^2(\mathbf{x}, \theta_0)$ are constant and its autocorrelation coefficient $\rho(\mathbf{x}, \mathbf{x}')$ is a function of $\mathbf{x}$ and $\mathbf{x}'$ only. An analogous definition

DOI: 10.1002/nme

applies for a lognormal random field. The autocorrelation function may take many distinct forms. One example is the exponential type

$$\rho(\mathbf{x},\mathbf{x}')=\mathrm{e}^{-\frac{|\mathbf{x}_{(1)}-\mathbf{x}'_{(1)}|}{\alpha_1}-\frac{|\mathbf{x}_{(2)}-\mathbf{x}'_{(2)}|}{\alpha_2}} \tag{9}$$

where $\mathbf{x}_{(i)}$ denotes the ith coordinate of $\mathbf{x}$ and α_1,α_2 are known as *correlation lengths*. Note that the variance of the random field is denoted by ς, such that it is not confused with the variance of the GPE introduced in Section 2, denoted by σ.

Random fields are a useful tool to model the distributed random mechanical properties of engineering systems, such as Poisson's ratio, Young's modulus, or yield stress. It is well established that, for a linear N-degree-of-freedom system, the deterministic FEM eventually yields a system of the form

$$\mathbf{Ku}=\mathbf{f} \tag{10}$$

where $\mathbf{K}\in\mathbb{R}^{N\times N}$ is known as the stiffness matrix, $\mathbf{u}\in\mathbb{R}^N$ is the response vector, and $\mathbf{f}\in\mathbb{R}^N$ is the forcing vector. If uncertainty is to be taken into account, then $\mathbf{K}$ becomes a random matrix. This means that the stiffness matrix becomes a function of the spatial coordinates and a random dimension, namely $\mathbf{K}(\mathbf{x},\theta)$. Analogously, the response vector becomes a random vector.

When implementing the deterministic FEM, functions are represented by a set of parameters, that is, the values of the function and its derivatives at the nodal points. In the case of the SFEM, the random field involved is discretized by representing it as a set of random variables. Therefore, $\mathscr{H}(\mathbf{x},\theta)$ needs to be discretized in order to solve the associated system of random algebraic equations. An advantageous alternative for discretizing $\mathscr{H}(\mathbf{x},\theta)$ is the Karhunen–Loève expansion (KLE), for which

$$\mathscr{H}(\mathbf{x},\theta)=\sum_{i=0}^{\infty}\sqrt{\lambda_i}\,\xi_i(\theta)\phi_i(\mathbf{x}) \tag{11}$$

where $\{\xi_i(\theta)\}_{i=0}^{\infty}$ is a set of random variables, $\{\lambda_i\}_{i=0}^{\infty}$ a set of constants, and $\{\phi_i(\mathbf{x})\}_{i=0}^{\infty}$ an orthonormal set of deterministic functions. In particular, $\{\lambda_i\}_{i=0}^{\infty}$ and $\{\phi_i(\mathbf{x})\}_{i=0}^{\infty}$ are the eigenvalues and eigenfunctions of the covariance kernel $K(\cdot,\cdot)$, that is, they arise from the solution of the integral equation

$$\int_{\mathbb{R}^N}K(\mathbf{x}_1,\mathbf{x}_2)\phi_i(\mathbf{x})\,\mathrm{d}\mathbf{x}_1=\lambda_i\phi_i(\mathbf{x}_2) \tag{12}$$

The truncated KLE of $\mathscr{H}(\mathbf{x},\theta)$ up to M terms is defined (see for example [14]) as

$$\mathscr{H}(\mathbf{x},\theta)=\mu(\mathbf{x})+\sum_{i=1}^{M}\sqrt{\lambda_i}\,\xi_i(\theta)\phi_i(\mathbf{x}) \tag{13}$$

Note that this is not the only available technique to discretize the random field $\mathscr{H}(\mathbf{x},\theta)$. There are more discretization methods that can be divided into three categories at least: point discretization, average discretization, and series expansion methods [14]. Of course, the KLE belongs to the third classification. Nevertheless, the KLE has uniqueness and error-minimization properties that make it a convenient choice over other available methods. See [33] for a detailed study of the cited and other KLE properties.

In particular, engineers are interested in obtaining the probability density function and the cumulative distribution function of the random response $\mathbf{u}(\mathbf{x},\theta)$ in order to assess the reliability of the system under study. However, this objective can prove to be difficult to achieve and the approach is limited to obtaining the first few statistical moments of $\mathbf{u}(\mathbf{x},\theta)$. There exist several strategies, such as perturbation and projection methods, or the random matrix approach, to deal with this problem. An account of these methods is given in [34–37].

Before discussing the incorporation of GPEs to the solution process of the stochastic version of Equation (10), it is worth focusing on the problem of efficiently generating realizations of the

DOI: 10.1002/nme

associated random field $\mathscr{H}(\mathbf{x}, \theta)$. This can be a computationally expensive task as the number of points in which the random field realization is evaluated increases. Once the convenience of the use of GPEs for this particular problem has been established, a method to solve the linear system (10) with the aid of GPEs will be proposed.

3.2. *Numerical example*: *random field emulation*

Consider a Gaussian homogeneous two-dimensional random field $\mathscr{H}(\mathbf{x}, \theta)$ with mean $\mu = 5$, variance $\varsigma^2 = 1$, and exponential autocorrelation function. Suppose the order of the KLE is fixed at $M = 100$. Note that the estimation of these three parameters is an interesting problem in its own right. They might represent material properties and be determined by, for example, experimental measurements. However, since the purpose of this paper is to explore the capabilities of emulators in the SFEM context, the parameters above are chosen freely and following no particular methodology. The corresponding simulator is thus

$$\eta(\mathbf{x}, \theta) = 5 + \sum_{i=1}^{100} \sqrt{\lambda_i}\, \xi_i(\theta) \phi_i(\mathbf{x}) \tag{14}$$

Let the spatial domain of the simulator be the square region $\mathscr{R}_L = [0, L] \times [0, L]$. Divide this domain into square elements such that the nodal points are $(L+1)^2$ in total. The truncated KLE can be calculated based on the results by Ghanem and Spanos [38], who provide analytical expressions for the eigenvalues and eigenfunctions corresponding to a random field with the above characteristics. Figure 1 shows a realization $\mathscr{H}(\mathbf{x}, \theta_\mathbf{0})$ of the random field with $L = 50$, which in terms of Equation (14) would be $\eta(\mathbf{x}, \theta_\mathbf{0})$ for every $\mathbf{x} \in \mathscr{R}_{50}$. The correlation lengths were chosen to be $[\alpha_1, \alpha_2]^\mathrm{T} = [0.02L, 0.02L]^\mathrm{T}$.

In order to apply Algorithm 1 to similarly generated random fields, n design points $(\mathbf{x}, \theta_\mathbf{0})_1, \ldots, (\mathbf{x}, \theta_\mathbf{0})_n$ were selected using Latin hypercube sampling [39] and a nearest neighbor algorithm such that every point in the Latin hypercube was translated into a node. For a very fine grid, the difference in the initial design should be expected to be small. The value of n was chosen to be 10% of the number of nodes. The emulator's predictive mean $m^{**}(\cdot)$ was calculated to infer the values of the random field realization in each unsampled node. As for the calculation of the smoothness parameters, a method suggested by Haylock [40] was implemented. Briefly speaking, if $\mathbf{b}$ is the diagonal of the matrix $\mathbf{B}$ in Equation (3) and the density function $f(\mathbf{B}|\mathbf{y})$ is derived, then a maximum likelihood estimator of $\mathbf{b}$ can be obtained. Figure 2(a) shows the values of $m^{**}(\cdot)$ compared with a realization of a Gaussian random field with correlation lengths $[0.2L, 0.02L]^\mathrm{T}$,

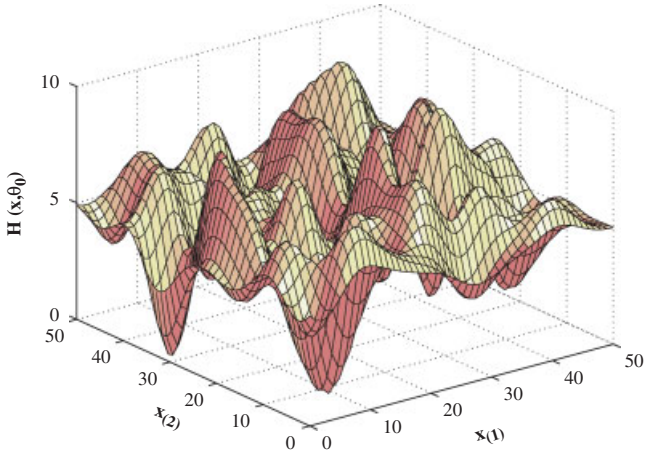

Figure 1. Realization of a Gaussian homogeneous random field $\mathscr{H}(\mathbf{x}, \theta)$ in $\mathscr{R}_{50} = [0, 50] \times [0, 50]$ with mean $\mu = 5$, variance $\varsigma^2 = 1$, and exponential autocorrelation function with correlation lengths $[\alpha_1, \alpha_2]^\mathrm{T} = [0.02L, 0.02L]^\mathrm{T}$. The KLE has $M = 100$ terms.

DOI: 10.1002/nme

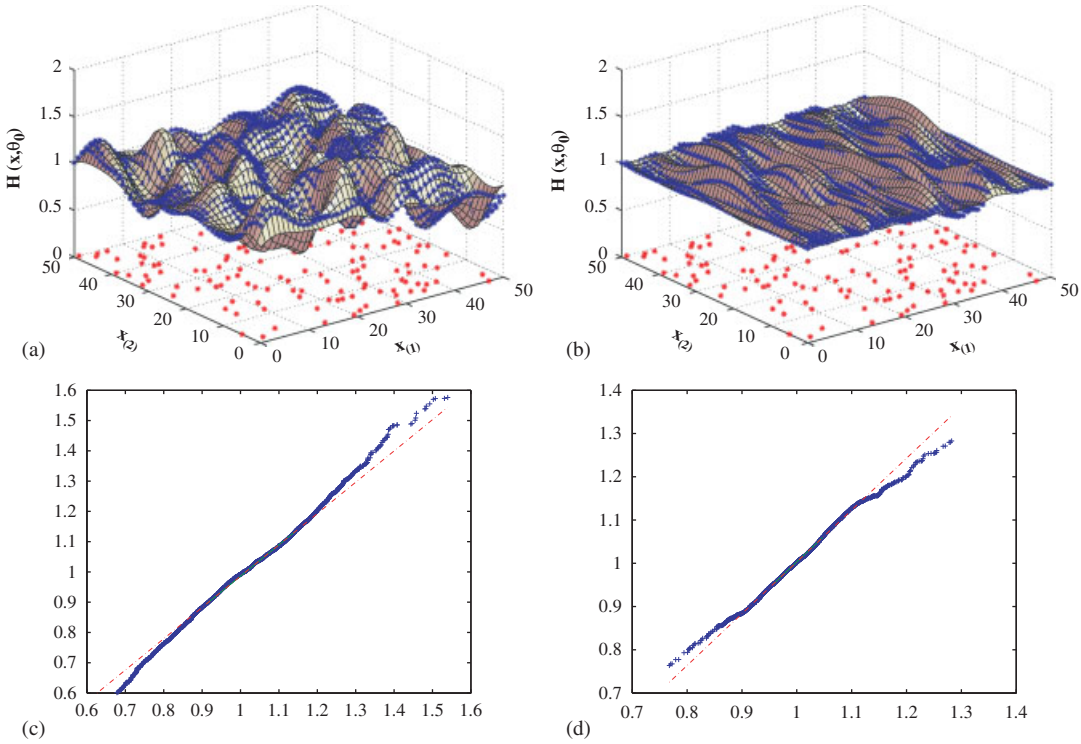

Figure 2. Realization $\mathscr{H}(\mathbf{x}, \theta_0)$ of a Gaussian homogeneous random field $\mathscr{H}(\mathbf{x}, \theta)$ with different correlation lengths. The adequacy of the emulator is represented by the Q–Q plots below: (a) correlation lengths $=(0.02L, 0.02L)^{\mathrm{T}}$; (b) correlation lengths $=(0.1L, 0.02L)^{\mathrm{T}}$; (c) Q–Q plot; and (d) Q–Q plot.

as well as the design points employed in the construction of the emulator. Figure 2(b) shows the emulation of a realization of another Gaussian random field, whose correlation lengths are $[0.1L, 0.02L]^{\mathrm{T}}$. The same design points were considered. Being $\mathscr{H}(\mathbf{x}, \theta)$ a two-dimensional random field, the accuracy of the approximation provided by the emulator might be difficult to appreciate. The adequacy of the emulator can be represented better in Figure 2(c) and (d), which are quantile–quantile plots (Q–Q plots) [41] of the emulated output against simulated output. The resulting plots are approximately linear, suggesting that the data sets come from the same distribution. The exercise was repeated for two lognormal random fields with correlation lengths $[0.02L, 0.02L]^{\mathrm{T}}$ and $[0.02L, 0.1L]^{\mathrm{T}}$. The lognormal random fields were obtained using the Nataf transformation [42–44]. The results of the emulation and the Q–Q plots are shown in Figure 3(a)–(d).

The exercise above was carried out for several values of L and thus an increasing number of nodes. The percentage of design points was varied to be as small as possible, taking care that the corresponding Q–Q plot remained approximately linear. The time employed to produce one realization of the corresponding random field for an increasing number of nodes is shown in Table I. Note that the time the emulator takes includes the time of evaluating the design points.

4. CALCULATION OF THE STOCHASTIC RESPONSE

The results in Section 3 show that realizations of both Gaussian and non-Gaussian random fields can be efficiently approximated using GPEs. This motivates the study of a non-Gaussian random field of particular importance, that is, the random field induced by the solution of the stochastic version of Equation (10) through SFEM analysis. This solution is important because it expresses the response of the engineering system under study. The problem of approximating it using GPEs is addressed in this section.

DOI: 10.1002/nme

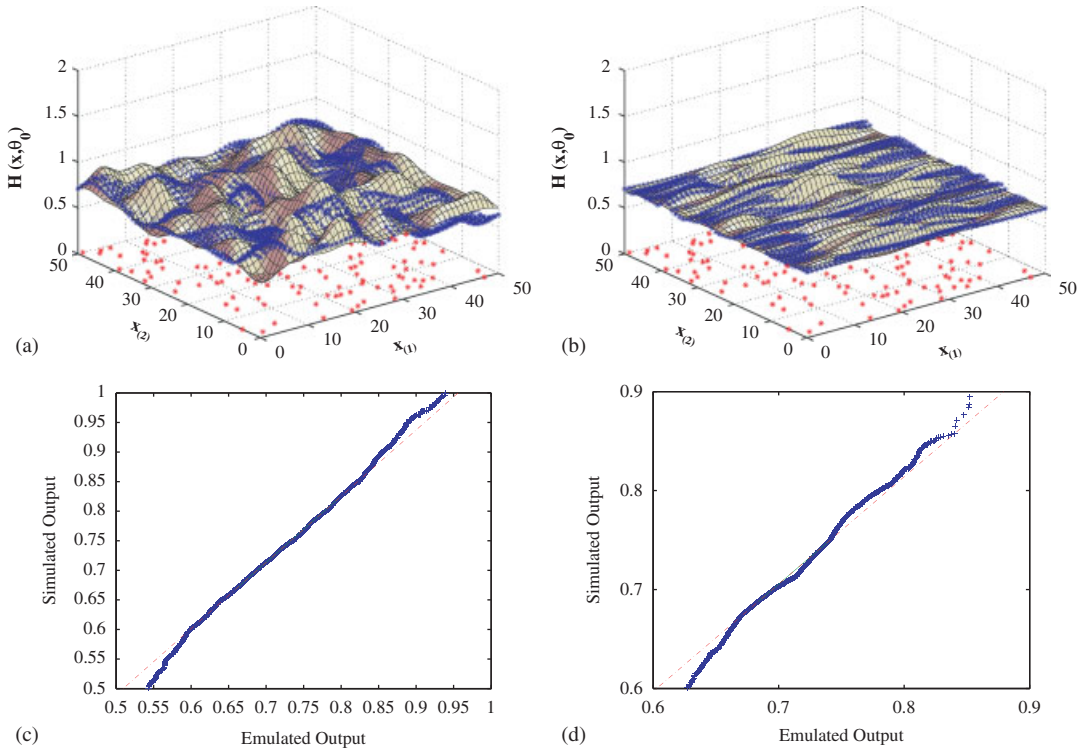

Figure 3. Realization $\mathscr{H}(\mathbf{x}, \theta_0)$ of a lognormal homogeneous random field $\mathscr{H}(\mathbf{x}, \theta)$ with different correlation lengths. The adequacy of the emulator is represented by the Q–Q plots below: (a) correlation lengths $= (0.02L, 0.02L)^{\mathrm{T}}$; (b) correlation lengths $= (0.02L, 0.1L)^{\mathrm{T}}$; (c) Q–Q plot; and (d) Q–Q plot.

Table I. Number of nodes vs CPU time employed (s).

No. of nodes	% Design points	Time simulator	Time emulator
121	10	1.03	0.20
441	10	3.58	1.26
961	5	7.77	2.36
1681	3	13.53	4.58
2601	3	21.73	6.42

4.1. Some available strategies

Several methods to solve Equation (10) and consequently calculate the statistics of the response $\mathbf{u}(\mathbf{x}, \theta)$ are available in the literature. These include Monte Carlo simulation techniques [45, 46], perturbation methods through Taylor series [6, 7, 47], expansion methods through Neumann series [48, 11], methods based on the eigenproperties of the finite element model [49, 50], and other analytical methods [51–53]. Despite their theoretical appeal, their implementation presents at least one of the following disadvantages: (a) lack of the geometrical appeal presented by the FEM; (b) limited applicability due to restrictive analytical constraints; (c) non-guaranteed convergence of the Taylor and Neumann series involved; and (d) potentially high computational cost. Additionally, since the covariance structure of the response random field is generally unknown, the KLE studied in Section 3 cannot be employed to represent $\mathbf{u}(\mathbf{x}, \theta)$.

Aiming to tackle some of the above-mentioned disadvantages, Ghanem and Spanos [38] proposed the *polynomial chaos expansion method*, also known simply as *polynomial chaos*. Later this approach was extended by several authors, for example [15, 17, 18, 54–56]. It essentially consists

DOI: 10.1002/nme

in representing each component of the random displacement vector $\mathbf{u}(\mathbf{x}, \theta)$ as a series of orthogonal polynomials $\{\Psi_j(\theta)\}_{j=0}^{\infty}$ in the standard normal variables $\{\xi_k(\theta)\}_{k=1}^{\infty}$, such that

$$\mathbf{u}(\mathbf{x}, \theta) = \sum_{j=0}^{P-1} \mathscr{U}_j(\mathbf{x})\Psi_j(\theta) \tag{15}$$

where each $\mathscr{U}_j$ is a deterministic vector in $\mathbb{R}^N$. Equation (15) is then substituted into (10), and $\mathbf{K}(\mathbf{x}, \theta)$ is discretized using the following truncated KLE:

$$\mathbf{K}(\mathbf{x}, \theta) = \sum_{i=0}^{M} \mathbf{K}_i(\mathbf{x})\xi_i(\theta) \tag{16}$$

where each $\mathbf{K}_i(\mathbf{x})$ is assembled from element matrices as in the deterministic FEM. The approximate solution is obtained by minimizing the underlying residual

$$\mathscr{R}(\mathbf{x}, \theta) = \sum_{i=0}^{M} \sum_{j=0}^{P-1} \mathbf{K}_i(\mathbf{x})\mathscr{U}_j(\mathbf{x})\xi_i(\theta)\Psi_j(\theta) - \mathbf{f} \tag{17}$$

by means of a projection onto the space spanned by $\{\Psi_j(\theta)\}_{j=0}^{P-1}$. It can be shown [14] that this projection can be determined by solving a linear system of the form

$$\mathscr{K} \cdot \mathscr{U} = \mathscr{F} \tag{18}$$

where each of the P components of $\mathscr{U} = [\mathscr{U}_0, \ldots, \mathscr{U}_{P-1}]^{\mathrm{T}}$ is N-dimensional and consequently the global stiffness matrix $\mathscr{K}$ is of size $NP \times NP$. Once every component in $\mathscr{U}$ is determined, the vector $\mathbf{u}(\mathbf{x}, \theta)$ can be computed as in Equation (15). Note that if v denotes the number of nodes in the finite element mesh and $\delta \geqslant 1$ is the number of degrees of freedom per node, then $N = \delta v$. This definition will become useful in the following subsections.

An important challenge in the implementation of polynomial chaos is its potentially high computational cost. The number P of basis polynomials in the expansion grows very rapidly for small changes in the degree of the polynomials p, and in the number of terms in the KLE M, as can be seen in Table II.

In general

$$P(p, M) = \sum_{k=0}^{p} \binom{M+k-1}{k} \tag{19}$$

This has an important implication, pointed out by Debusschere *et al.* [57]. Unless high-order polynomials are employed, the accuracy of the polynomial chaos representation may be inaccurate and unstable, that is, produce non-physical values for the parameters modeled. Unfortunately, the use of high values of p may be computationally intractable. Similarly, for a bigger number of terms in the KLE, the solution of the linear system (18) becomes increasingly expensive. Additionally, if the engineering system under study has a large number N of degrees of freedom, the computational effort can rapidly become prohibitive. A possible solution is offered by the recent development

Table II. Number of basis polynomials for different degrees (p) and number of terms in the Karhunen–Loève expansion (M).

M	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
2	3	6	10	15	21
3	4	10	20	35	56
4	5	15	35	70	126
5	6	21	56	126	252
6	7	28	84	210	462
10	11	66	286	1001	3003

DOI: 10.1002/nme

of non-intrusive methods [58–61]. The term non-intrusive refers to the fact that these strategies allow the implementation of the polynomial chaos without modifying the original computer model. In this setup, a GPE can be an inexpensive non-intrusive surrogate model of a polynomial chaos simulator. It can be seen from Equation (15) that

$$\mathbf{u}(\mathbf{x},\theta)=\sum_{j=0}^{P-1}\boldsymbol{\mathcal{U}}_j\ (\mathbf{x})\Psi_j(\theta)=\begin{bmatrix}\boldsymbol{\mathcal{U}}_0^{(1)}\ (\mathbf{x})\\ \vdots\\ \boldsymbol{\mathcal{U}}_0^{(N)}\ (\mathbf{x})\end{bmatrix}\Psi_0(\theta)+\cdots+\begin{bmatrix}\boldsymbol{\mathcal{U}}_{P-1}^{(1)}\ (\mathbf{x})\\ \vdots\\ \boldsymbol{\mathcal{U}}_{P-1}^{(N)}\ (\mathbf{x})\end{bmatrix}\Psi_{P-1}(\theta) \tag{20}$$

where the superscript in parenthesis numbers the row of the column vector it superscripts. Let $\Gamma=\{\gamma_1,\ldots,\gamma_{n\delta}\}\subset\{1,\ldots,N\}$ be an index set with $n\delta\ll N$. Suppose it is possible to obtain $n\delta$ rows from each of the vectors $\boldsymbol{\mathcal{U}}_0,\ldots,\boldsymbol{\mathcal{U}}_{P-1}$, that is, $[\boldsymbol{\mathcal{U}}_0^{(\gamma_1)},\ldots,\boldsymbol{\mathcal{U}}_0^{(\gamma_{n\delta})}]^{\mathrm{T}}$ from $\boldsymbol{\mathcal{U}}_0$, $[\boldsymbol{\mathcal{U}}_1^{(\gamma_1)},\ldots,\boldsymbol{\mathcal{U}}_1^{(\gamma_{n\delta})}]^{\mathrm{T}}$ from $\boldsymbol{\mathcal{U}}_1$, until $[\boldsymbol{\mathcal{U}}_{P-1}^{(\gamma_1)},\ldots,\boldsymbol{\mathcal{U}}_{P-1}^{(\gamma_{n\delta})}]^{\mathrm{T}}$ from $\boldsymbol{\mathcal{U}}_{P-1}$. Consequently, $n\delta$ rows of the response vector $\mathbf{u}(\mathbf{x},\theta)$ would be available, namely

$$\begin{aligned}\mathbf{u}^{(\gamma_1)}(\mathbf{x},\theta)&=\sum_{j=0}^{P-1}\boldsymbol{\mathcal{U}}_j^{(\gamma_1)}\ (\mathbf{x})\Psi_j(\theta)\\ \mathbf{u}^{(\gamma_2)}(\mathbf{x},\theta)&=\sum_{j=0}^{P-1}\boldsymbol{\mathcal{U}}_j^{(\gamma_2)}\ (\mathbf{x})\Psi_j(\theta)\\ &\vdots\\ \mathbf{u}^{(\gamma_{n\delta})}(\mathbf{x},\theta)&=\sum_{j=0}^{P-1}\boldsymbol{\mathcal{U}}_j^{(\gamma_{n\delta})}\ (\mathbf{x})\Psi_j(\theta)\end{aligned} \tag{21}$$

In that way, the set $\{\boldsymbol{\mathcal{U}}_0^{(\gamma_1)},\ldots,\boldsymbol{\mathcal{U}}_{P-1}^{(\gamma_1)},\ldots,\boldsymbol{\mathcal{U}}_0^{(\gamma_{n\delta})},\ldots,\boldsymbol{\mathcal{U}}_{P-1}^{(\gamma_{n\delta})}\}$ could be regarded as the design points that would be used to build a GPE in order to approximate the $N-n\delta$ remaining elements of $\mathbf{u}(\mathbf{x},\theta)$, namely $\{\mathbf{u}^{(\gamma_{n+1})}(\mathbf{x},\theta),\ldots,\mathbf{u}^{(\gamma_{N-n\delta})}(\mathbf{x},\theta)\}$, where $\{\gamma_{n+1},\ldots,\gamma_{N-n\delta}\}=\{1,\ldots,N\}\backslash\Gamma$. For notational purposes, let these sets be expressed as vectors, partitioning $\mathbf{u}(\mathbf{x},\theta)$ as

$$\mathbf{u}(\mathbf{x},\theta)=\begin{pmatrix}\mathbf{u}_1(\mathbf{x},\theta)\\ \\ \\ \mathbf{u}_2(\mathbf{x},\theta)\end{pmatrix}=\begin{pmatrix}\mathbf{u}^{(\gamma_{n\delta+1})}(\mathbf{x},\theta)\\ \vdots\\ \mathbf{u}^{(\gamma_{N-n\delta})}(\mathbf{x},\theta)\\ \mathbf{u}^{(\gamma_1)}(\mathbf{x},\theta)\\ \vdots\\ \mathbf{u}^{(\gamma_{n\delta})}(\mathbf{x},\theta)\end{pmatrix} \tag{22}$$

Note however that obtaining the design points is not as simple as it was when emulating the random fields in Section 3: in that case each design point represented a node in the finite element mesh and the training runs were obtained by simply evaluating the random field at the selected node. In the current setup, obtaining the required design points would imply solving the linear system (18) only partially, such that the corresponding training runs will be used as input for the

DOI: 10.1002/nme

emulation algorithm. The training runs can be expressed in terms of Equation (A7) in Appendix as $\mathbf{y}=[\mathbf{y}_1,\ldots,\mathbf{y}_\delta]\in\mathbb{R}^{n\times\delta}$, where

$$\begin{aligned}\mathbf{y}_1&=[\mathbf{u}^{(\gamma_1)},\ldots,\mathbf{u}^{(\gamma_\delta)}]^{\mathrm{T}}\\ \mathbf{y}_2&=[\mathbf{u}^{(\gamma_{\delta+1})},\ldots,\mathbf{u}^{(\gamma_{2\delta})}]^{\mathrm{T}}\\ &\vdots\\ \mathbf{y}_n&=[\mathbf{u}^{(\gamma_{n\delta-\delta+1})},\ldots,\mathbf{u}^{(\gamma_{n\delta})}]^{\mathrm{T}}\end{aligned}\tag{23}$$

The following subsection elaborates on this idea and proposes an algorithm to solve the linear system (18) partially in order to obtain the design points and the training runs $\mathbf{y}$.

4.2. Partitioned Cholesky decomposition

Let the stochastic finite element system $\mathscr{K}\cdot\mathscr{U}=\mathscr{F}$ in Equation (18) be partitioned as

$$\begin{pmatrix}\mathbf{A}&\mathbf{B}\\ \mathbf{B}^{\mathrm{T}}&\mathbf{C}\end{pmatrix}\begin{pmatrix}\mathscr{U}_1^*\\ \mathscr{U}_2^*\end{pmatrix}=\begin{pmatrix}\mathscr{F}_1\\ \mathscr{F}_2\end{pmatrix}\tag{24}$$

where $\mathbf{A}\in\mathbb{R}^{mP\times mP}$, $\mathbf{C}\in\mathbb{R}^{n\delta P\times n\delta P}$ with $m+n\delta=N$. Note that $\mathscr{U}$, $\mathscr{U}_1^*$ and $\mathscr{U}_2^*$, and $\mathscr{U}_0,\ldots,\mathscr{U}_{P-1}$ are related in the following way:

$$\mathscr{U}=\begin{pmatrix}\mathscr{U}_1^*\\ \\ \mathscr{U}_2^*\end{pmatrix}=\begin{pmatrix}\mathscr{U}_0\ (\mathbf{x})\\ \vdots\\ \mathscr{U}_{P-1}\ (\mathbf{x})\end{pmatrix}\tag{25}$$

Definition 2 (*partitioned Cholesky factor*)
Let $\mathscr{K}$ be a positive definite matrix partitioned as in Equation (24). The partitioned Cholesky factor $\mathbf{L}$ of $\mathscr{K}$ is given by

$$\mathbf{L}=\begin{pmatrix}\mathbf{L_A}&\mathbf{0}\\ \mathbf{W}^{\mathrm{T}}&\mathbf{L_D}\end{pmatrix}\tag{26}$$

with $\mathbf{L_A}$ and $\mathbf{L_D}$ the Cholesky factors of $\mathbf{A}$ and $\mathbf{D}=\mathbf{C}-\mathbf{W}^{\mathrm{T}}\mathbf{W}$ respectively, and $\mathbf{W}^{\mathrm{T}}$ such that

$$\mathbf{W}^{\mathrm{T}}\mathbf{W}=\mathbf{B}^{\mathrm{T}}\mathbf{L}_{\mathbf{A}}^{-\mathbf{T}}\mathbf{L}_{\mathbf{A}}^{-1}\mathbf{B}=\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{B}\tag{27}$$

The calculation of the partitioned Cholesky factor $\mathbf{L}$ of $\mathscr{K}$ is summarized in Algorithm 2.

Algorithm 2 Partitioned Cholesky factor L of $\mathscr{K}$.

Input: Matrices $\mathbf{A},\mathbf{B},\mathbf{C}$
Output: Partitioned Cholesky factor $\mathbf{L}$
begin
1. Factor $\mathbf{A}$ into $\mathbf{L_A}\mathbf{L_A^T}$
2. Solve the (triangular) linear systems $\mathbf{L_A}\mathbf{W}=\mathbf{B}$
3. Compute $\mathbf{D}=\mathbf{C}-\mathbf{W}^{\mathrm{T}}\mathbf{W}$
4. Factor $\mathbf{D}$ into $\mathbf{L_D}\mathbf{L_D^T}$
5. Assemble $\mathbf{L}$ as in Equation (26)

end

It can be shown [62] that the number of operations required to compute the partitioned factor $\mathbf{L}$ is the same as the required by the conventional, non-partitioned Cholesky decomposition. Once

DOI: 10.1002/nme

the matrix $\mathscr{K}$ is decomposed as $\mathbf{L}\mathbf{L}^{\mathrm{T}}$, the solution of the linear system (24) can be obtained by first solving the lower triangular linear system

$$\begin{pmatrix} \mathbf{L_A} & \mathbf{0} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{L_D} \end{pmatrix} \begin{pmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{pmatrix} = \begin{pmatrix} \mathscr{F}_1 \\ \mathscr{F}_2 \end{pmatrix} \tag{28}$$

and then using the auxiliary solution vectors $\mathbf{y_1}$ and $\mathbf{y_2}$ to solve the upper triangular linear system

$$\begin{pmatrix} \mathbf{L}_{\mathbf{A}}^{\mathrm{T}} & \mathbf{W} \\ \mathbf{0} & \mathbf{L}_{\mathbf{D}}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} \mathscr{U}_1^* \\ \mathscr{U}_2^* \end{pmatrix} = \begin{pmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{pmatrix} \tag{29}$$

The procedure is made explicit in Algorithm 3.

Algorithm 3 Solution of the partitioned system $\mathscr{K} \cdot \mathscr{U} = \mathscr{F}$.

Input: $\mathbf{L}, \mathscr{F}_1, \mathscr{F}_2$
Output: $\mathscr{U}_1^*, \mathscr{U}_2^*$
begin
Forward Solution:
1. Solve $\mathbf{L_A y_1} = \mathscr{F}_1$
2. Compute $\widetilde{\mathbf{f}}_2 = \mathscr{F}_2 - \mathbf{W}^{\mathrm{T}} \mathbf{y_1}$
3. Solve $\mathbf{L_D y_2} = \widetilde{\mathbf{f}}_2$

Backward Solution:
4. Solve $\mathbf{L}_{\mathbf{D}}^{\mathrm{T}} \mathscr{U}_2^* = \mathbf{y_2}$
5. Compute $\widetilde{\mathbf{y}}_1 = \mathbf{y_1} - \mathbf{W} \mathscr{U}_2^*$
6. Solve $\mathbf{L}_{\mathbf{A}}^{\mathrm{T}} \mathscr{U}_1^* = \widetilde{\mathbf{y}}_1$

end

In the worst case scenario, the computational complexity of implementing Algorithm 2 together with Algorithm 3 is the following: $O(N^3 P^3)$ operations are required for the Cholesky decomposition, $O(N^2 P^2)$ are necessary for the forward substitution and $O(N^2 P^2)$ are required for the backward substitution. The joint application of Algorithms 2 and 3 will hereafter be referred to as the symmetric algorithm.

From Table II, it can be seen that even for a system with few degrees of freedom, the number of operations $O(N^2 P^2)$ can be considerably large. To potentially reduce the cost of the symmetric algorithm, an alternative approach is proposed.

4.3. Coupling polynomial chaos with GPEs

As discussed in Section 4.1, the key to emulating the stochastic response $\mathbf{u}(\mathbf{x}, \theta)$ is obtaining the training runs contained in $\mathbf{u}_2(\mathbf{x}, \theta)$. From Equation (20) and the relationship provided by Equation (25), it can be seen that in order to obtain them, it is necessary to solve the linear system (18) for $n\delta P$ components of $\mathscr{U}$. Once these components are computed, the training runs would be obtained as in Equation (21). Note that if these $n\delta P$ components were all contained in the partition $\mathscr{U}_2^*$, then it would suffice to implement Algorithm 3 up to step 4. It is however unlikely that the components of $\mathscr{U}_2^*$ will be ordered such that the nodes associated with the training runs will be evenly spread throughout the finite element mesh. This would be detrimental to the predictive capability of the GPE, as it relies on design points that contain as much information as possible of the input domain. To solve this problem, the rows and columns of $\mathscr{K}$ can be permuted in order to group the desired components in $\mathscr{U}_2^*$. Note that due to the positive definiteness of $\mathscr{K}$, this permutation will render a matrix $\widetilde{\mathscr{K}}$ that is still symmetric and positive definite, such that Algorithm 3 can be readily applied. Once the design points are obtained, the training runs in Equation (23) are computed and a GPE can be built in order to approximate each of the remaining

DOI: 10.1002/nme

m components of $\mathbf{u}(\mathbf{x},\theta)$, denoted previously as $\mathbf{u}_1(\mathbf{x},\theta)$. For $\ell=1,\ldots,m$, this can be symbolically represented as

$$\mathbf{u}_1^{(\ell)}(\mathbf{x},\theta)=\mathscr{GPE}(\mathbf{u}_2(\mathbf{x},\theta)) \tag{30}$$

The proposed algorithm is summarized in Algorithm 4.

Algorithm 4 Solution of the partitioned system $\mathscr{K}\cdot\mathscr{U}=\mathscr{F}$ using GPEs.

Input: $\mathscr{K}$, $\mathscr{F}_\mathbf{1}$ and $\mathscr{F}_\mathbf{2}$
Output: $\mathscr{U}_1^*$, $\mathscr{U}_2^*$
begin
1. Choose a permutation $\widetilde{\mathscr{K}}$ of $\mathscr{K}$
2. Compute Cholesky factor $\mathbf{L}$ of $\widetilde{\mathscr{K}}$
Forward Solution:
3. Solve $\mathbf{L_A y_1}=\mathscr{F}_1$
4. Compute $\widetilde{\mathbf{f}}_2=\mathscr{F}_2-\mathbf{W}^\mathrm{T}\mathbf{y_1}$
5. Solve $\mathbf{L_D y_2}=\widetilde{\mathbf{f}}_\mathbf{2}$
Partial Backward Solution:
6. Solve $\mathbf{L}_\mathbf{D}^\mathrm{T}\mathscr{U}_2^*=\mathbf{y_2}$
Design points:
7. Compute $\mathbf{u}_2(\mathbf{x},\theta)$ as in Equation (21)
Training runs:
8. Compute the training runs $\mathbf{y}$ as in Equation (23)
Gaussian Process Emulation:
for $\ell=1,\ldots,m$
9. $\mathbf{u}_1^{(\ell)}(\mathbf{x},\theta)=\mathscr{GPE}(\mathbf{u}_2(\mathbf{x},\theta))$
end
end

4.4. Numerical complexity

Similar to the symmetric algorithm, the computational complexity of the proposed algorithm involves $O(N^3P^3)$ operations for the partitioned Cholesky decomposition and $O(N^2P^2)$ for the forward substitution. However, the $O(N^2P^2)$ operations required for the backward substitution are replaced by $O(n^2\delta^2P^2)$ operations for the partial backward substitution. Additionally, the GPE step requires $O(n^3\delta)$ operations. To see why, refer to Equation (A7) in the Appendix A, where $\mathbf{V}\in\mathbb{R}^{n\times n}$ and $\mathbf{y}\in\mathbb{R}^{n\times\delta}$. Therefore, the proposed algorithm would reduce the computational cost of solving the linear system (18) if and only if

$$O(N^2P^2)>O(n^2\delta^2P^2)+O(n^3\delta) \tag{31}$$

In Section 4.1, v was defined as the number of nodes and $N=\delta v$. Let n be a small fraction of v such that $n=\varepsilon v$ with $0<\varepsilon\ll 1$. Thus, $n=\varepsilon N/\delta$. This expression allows relating condition (31) to the number of nodes as follows:

$$\begin{aligned}
&N^2P^2>n^2\delta^2P^2+n^3\delta\\
\Leftrightarrow\ &N^2P^2>\varepsilon^2N^2P^2+(\varepsilon^3N^3)/\delta^2\\
\Leftrightarrow\ &P^2>\varepsilon^2P^2+(\varepsilon^3N)/\delta^2\\
\Leftrightarrow\ &1>\varepsilon^2+(\varepsilon^3N)/(P^2\delta^2)\\
\Leftrightarrow\ &1>\varepsilon^2+(\varepsilon^3v)/(P^2\delta)
\end{aligned} \tag{32}$$

As discussed before, P grows very rapidly for small changes in the number and order of basis polynomials. Naturally, P^2 grows even faster. This number is further scaled by $\delta\geqslant 1$. On the other

DOI: 10.1002/nme

hand, $\varepsilon^3 v$ is a very small fraction of v. Therefore, condition (31) is likely to be met, unless for example that the number of nodes v is considerably larger than the number of basis polynomials P.

4.5. *Efficient memory allocation*

In order to improve the memory usage when calculating the partitioned Cholesky factor $\mathbf{L}$, George [62] observes that the matrix $\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{B}$ in Algorithm 2 can be calculated not only by the conventional expressions

$$\mathbf{B}^{\mathrm{T}}\mathbf{L}_{\mathbf{A}}^{-\mathbf{T}}\mathbf{L}_{\mathbf{A}}^{-1}\mathbf{B}=\mathbf{W}^{\mathrm{T}}\mathbf{W} \tag{33}$$

but also by defining

$$\widetilde{\mathbf{W}}=\mathbf{L}_{\mathbf{A}}^{-\mathbf{T}}\mathbf{W} \tag{34}$$

In that way

$$\mathbf{B}^{\mathrm{T}}\widetilde{\mathbf{W}}=\mathbf{B}^{\mathrm{T}}(\mathbf{L}_{\mathbf{A}}^{-\mathbf{T}}\mathbf{W})=\mathbf{B}^{\mathrm{T}}(\mathbf{L}_{\mathbf{A}}^{-\mathbf{T}}(\mathbf{L}_{\mathbf{A}}^{-1}\mathbf{B})) \tag{35}$$

This asymmetric scheme can reduce storage requirements as $\mathbf{W}$ is not needed to compute the solution to the linear system as long as $\mathbf{B}$ is available. A product of the form $\mathbf{W}^{\mathrm{T}}\mathbf{z}$ or $\mathbf{W}\mathbf{z}$ can be computed by solving a triangular system and multiplying by a sparse matrix, namely $\mathbf{B}^{\mathrm{T}}(\mathbf{L}_{\mathbf{A}}^{-\mathrm{T}}\mathbf{z})$ or $\mathbf{L}_{\mathbf{A}}^{-1}(\mathbf{B}\mathbf{z})$. Storage is then saved if $\mathbf{B}$ is sparser than $\mathbf{W}$. Moreover, the storage of $\mathbf{W}$ can be avoided by computing $\mathbf{D}$ following the asymmetric scheme: $\mathbf{B}^{\mathrm{T}}\widetilde{\mathbf{W}}$ can be computed one column at a time, discarding each after modifying a column of $\mathbf{D}$. In that way, only a temporary vector is required.

4.6. *Numerical example*: *a cantilever plate*

The proposed algorithm is tested in an example of a square plate model clamped along one edge [38]. The analysis is based on a polynomial chaos simulator by Haukaas and Der Kiureghian [63]. The plate is subjected to a uniform tension on the two vertices on the opposite edge. Young's modulus is assumed to be a two-dimensional Gaussian random process with mean value $\overline{E}=2.0\times10^5$ MPa and known exponential covariance function. A finite element model of a plate with 15×15 elements (and $v=256$ nodes) is shown in Figure 4. The number of design points (shown as circles) was chosen to be 5% of the total number of free nodes. In order to reduce uncertainty, the nodes on free edge could have also been taken as design points. However, as mentioned in Section 2, this would have increased the number of design points dramatically. On the contrary, the nodes in the fixed edge were taken as design points as their displacement is known to be equal to zero. When the fixed edge is accounted for, the percentage of design points increased to around 11%.

A polynomial chaos analysis with third degree polynomials ($M=4$, $p=3$) was run, resulting in 35 basis polynomials ($P=35$). With 2 degrees of freedom per node ($\delta=2$), the system matrix $\mathscr{K}$ had a dimension of $17\,920\times17\,920$. The partitioned Cholesky factor had 31.62% non-zero elements (see Figure 5(a)). The rows and columns of $\mathscr{K}$ corresponding to the displacement of the design points were identified and a permutation was applied in order to apply Algorithm 4. The resulting Cholesky factor of $\widetilde{\mathscr{K}}$ had 8.26% non-zero elements (see Figure 5(b)). This is relevant since it could have been the case that the partitioned Choleksy factor suffered fill-in, with a potentially considerable increase in computer execution time and storage. At this point, the proposed algorithm cannot guarantee that the partitioned Cholesky factor of the permuted matrix $\widetilde{\mathscr{K}}$ will be at least as sparse as that of $\mathscr{K}$. This is a major research topic in numerical analysis, and for now beyond the scope of this paper. However, the important aspect to keep in mind is that, given a scheme that partially solves a linear system by determining a small part of the solution vector, a GPE can be employed to approximate the complement to that partial solution.

The displacements were computed for the design points, and a GPE was then built to predict the displacement of the remaining nodes. Condition (31) was satisfied, since $N^2P^2=321\,126\,400>3\,930\,290=n^2\delta^2P^2+n^3\delta$. Once the polynomial chaos analysis was complete, 10 000 samples

DOI: 10.1002/nme

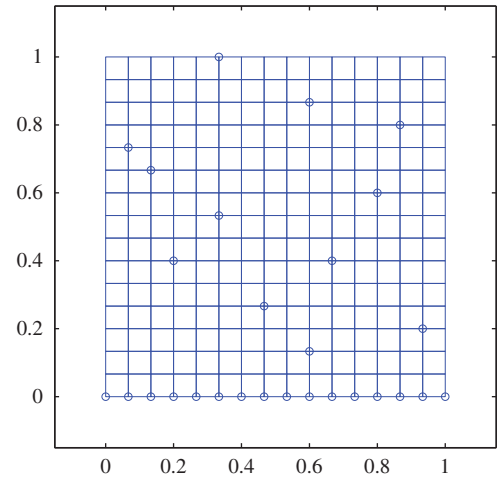

Figure 4. Cantilever plate model clamped along $y=0$. The design points are shown as circles.

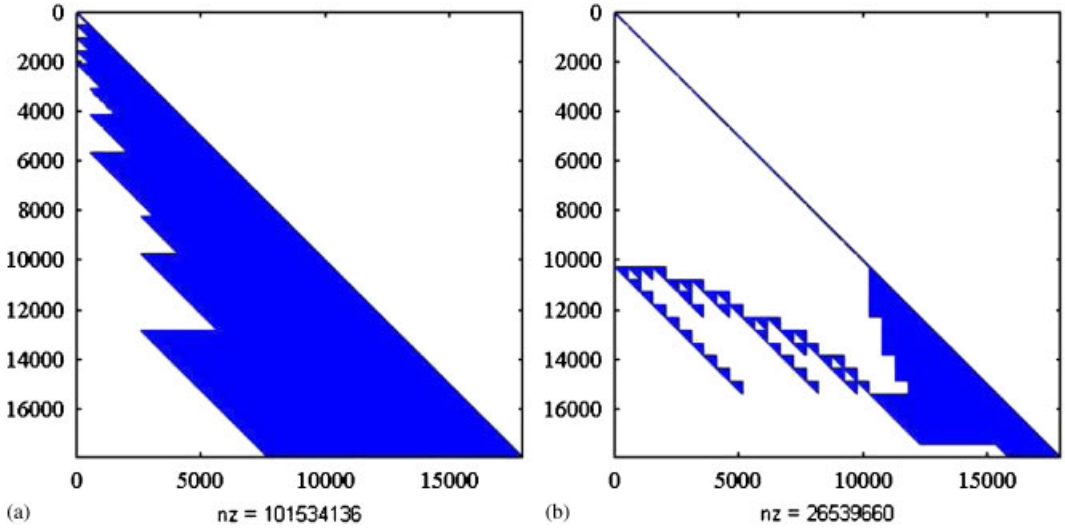

Figure 5. Matrix profiles showing the non-zero elements of the partitioned Cholesky factors for the original and the permuted matrix. The permuted factor of $\widetilde{\mathscr{K}}$ is much sparser in this case: (a) partitioned Cholesky factor of the original matrix $\mathscr{K}$ and (b) partitioned Cholesky factor of the permuted matrix $\widetilde{\mathscr{K}}$.

where generated and each of these displacement fields was emulated. The mean simulated and emulated displacements for each node are compared in Figure 6(a). The relative (Euclidean) distance between the mean simulated and emulated displacements is shown in Figure 6(b), where each distance is normalized by the maximum overall distance. Notice how the relative distance is greater on the free edge of the plate, reflecting the fact that the uncertainty in the prediction of a GPE is greater when extrapolating the training runs. The prediction seems to be reasonably accurate for the rest of the nodes, as can be seen by comparing the magnitude of the distance between the simulated and the emulated responses with respect to the overall maximum distance. The same type of analysis was carried out for the standard deviation of the displacements. The results are shown in Figure 7(a) and (b).

Finally, in order to treat the response $\mathbf{u}(\mathbf{x},\theta)$ statistically, both in the vertical and horizontal directions, the probability density functions of the displacement of three nodes across the plate was generated. As Figure 8 shows, the uncertainty in the GPE prediction seems to increase the

DOI: 10.1002/nme

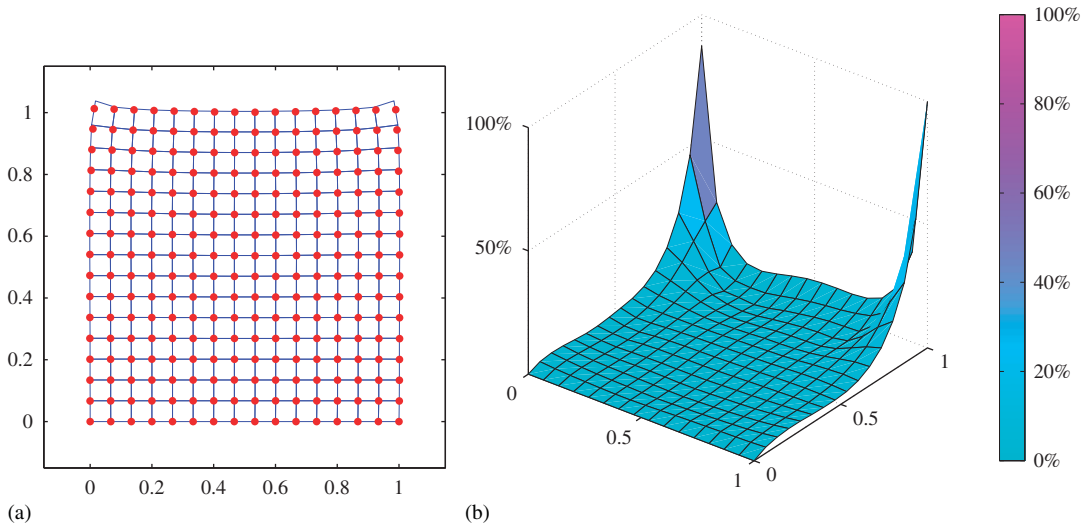

Figure 6. Comparison between the mean simulated (lines) and mean emulated (dots) displacement fields. The relative Euclidean distances are normalized by the overall maximum distance: (a) emulated and simulated displacements and (b) relative Euclidean distances.

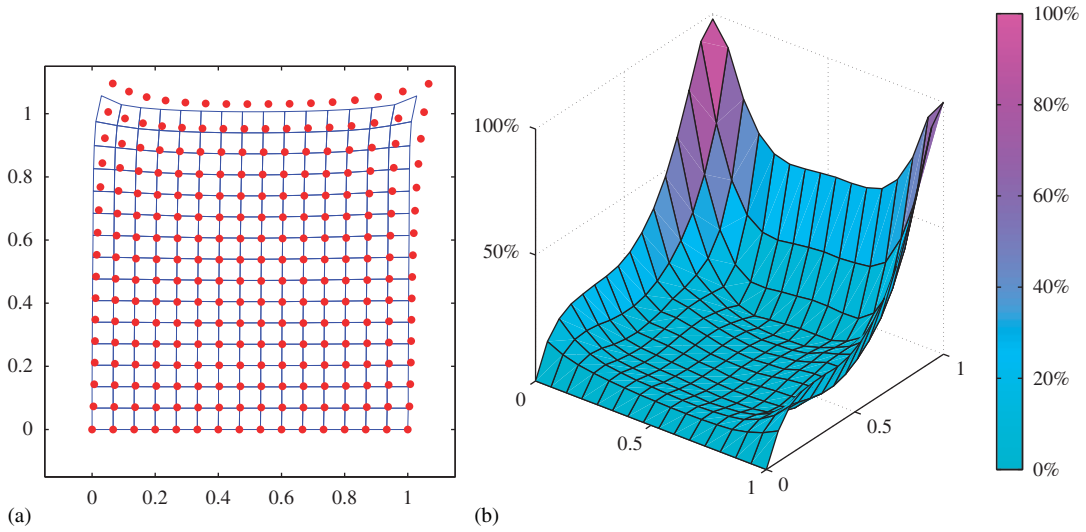

Figure 7. Comparison between the standard deviation of the simulated (lines) and mean emulated (dots) displacement fields. The relative Euclidean distances are normalized by the overall maximum distance. The standard deviation was scaled up by a factor of 10 in order to make them comparable to the magnitude of the mean displacements: (a) emulated and simulated displacements and (b) relative Euclidean distances.

farther away is the node from the fixed edge, since the magnitudes of the node displacements are expected to be larger.

5. CONCLUSIONS

The capabilities of GPEs to approximate computationally expensive random fields was explored. Assuming the covariance function is known, realizations of Gaussian and lognormal homogeneous random fields were emulated for an increasing number of input points in the input domain. For these cases, the generation of the training runs was automatic once the relevant simulator was

DOI: 10.1002/nme

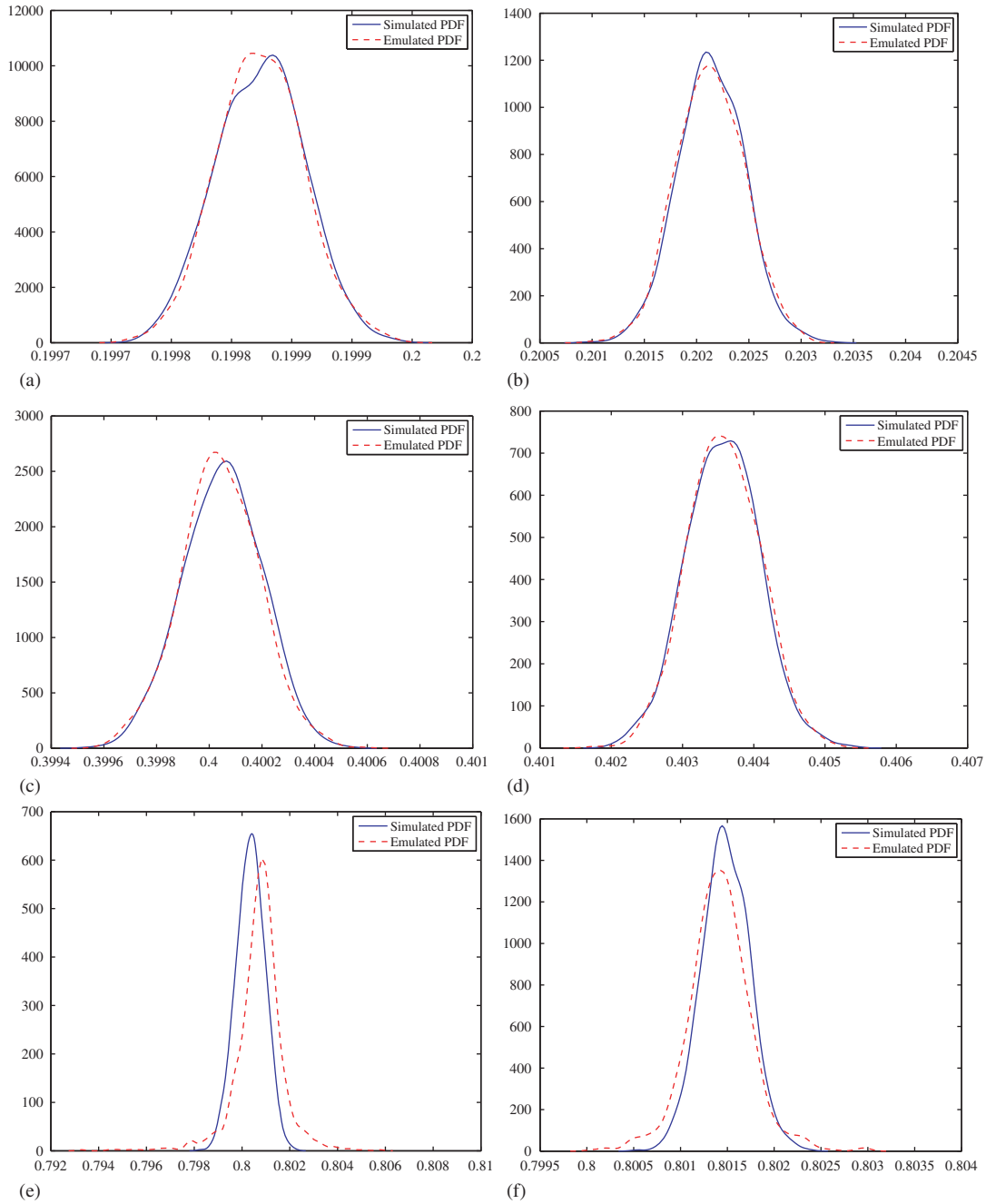

Figure 8. Probability density functions of the displacement of three nodes along the x and the y axes. The solid line corresponds to the simulator, whereas the dotted line corresponds to the emulator: (a) displacement of (0.2,0.2) along the x-axis; (b) displacement of (0.2,0.2) along the y-axis; (c) displacement of (0.4,0.4) along the x-axis; (d) displacement of (0.4,0.4) along the y-axis; (e) displacement of (0.8,0.8) along the x-axis; and (f) displacement of (0.8,0.8) along the y-axis.

defined and implemented using the KLE. Good agreement between the original and the emulated values was observed. The results show that a realization of a random field can be obtained from only a small number of training runs.

The random field induced by the solution of a stochastic finite element analysis was emulated by coupling of the polynomial chaos expansion method with GPEs. The main objective of doing so was

DOI: 10.1002/nme

to propose an inexpensive alternative to the computation of the response of an engineering system with random parameters. For that case, generating the training runs from the polynomial chaos simulator was not immediate. Owing to this, an algorithm to solve a partition of the main linear system was proposed. Good correspondence between the emulated and the simulated response was found for a stochastic mechanics example. It was shown that using the proposed approach it is possible to obtain a high-resolution random field and also to compute the complete response of a stochastic system using the response values at few points only.

APPENDIX A: UPDATING THE PRIOR DISTRIBUTION

In this section, the process of updating the prior distribution (4) to obtain the posterior distribution (5) is outlined. Define $\mathbf{H}=[\mathbf{h}(\mathbf{x}_1),\ldots,\mathbf{h}(\mathbf{x}_n)]^{\mathrm{T}}$ and $\mathbf{V}\in\mathbb{R}^{n\times n}$ with $\mathbf{V}_{\ell j}=C(x_\ell,x_j)\ \forall \ell, j\in\{1,\ldots,n\}$. Hence

$$\mathbf{y}|\boldsymbol{\beta},\sigma^2\sim\mathcal{N}(\mathbf{H}\boldsymbol{\beta},\sigma^2\mathbf{V}) \tag{A1}$$

To incorporate the objective information $\mathbf{y}$ and obtain the distribution of $\eta(\cdot)|\mathbf{y},\boldsymbol{\beta},\sigma^2$, use the following result, given in Krzanowski [64].

Proposition A1

Let $\mathbf{z}\in\mathbb{R}^n$ be a random vector such that $\mathbf{z}\sim\mathcal{N}(\boldsymbol{\mu},\Sigma)$. Partition $\mathbf{z}$ as $\begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix}$, where $\mathbf{z}_1\in\mathbb{R}^p$ and $\mathbf{z}_2\in\mathbb{R}^{n-p}$. Consequently, partition $\boldsymbol{\mu}=\begin{pmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{pmatrix}$ and $\Sigma=\begin{pmatrix}\Sigma_{11}&\Sigma_{12}\\ \Sigma_{21}&\Sigma_{22}\end{pmatrix}$, so that $\mathbf{E}[\mathbf{z}_j]=\boldsymbol{\mu}_j$ and $Cov(\mathbf{z}_j,\mathbf{z}_k)=\Sigma_{jk}$. Then, $\mathbf{z}_1|\mathbf{z}_2\sim\mathcal{N}(\widetilde{\boldsymbol{\mu}},\widetilde{\Sigma})$, where $\widetilde{\boldsymbol{\mu}}=\boldsymbol{\mu}_1+\Sigma_{12}\Sigma_{22}^{-1}(\mathbf{z}_2-\boldsymbol{\mu}_2)$ and $\widetilde{\Sigma}=\Sigma_{11}-\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$.

From this result, it follows that

$$\eta(\cdot)|\mathbf{y},\boldsymbol{\beta},\sigma^2\sim\mathcal{N}(m^*(\cdot),\sigma^2C^*(\cdot,\cdot)) \tag{A2}$$

where

$$m^*(x)=\mathbf{h}(x)^{\mathrm{T}}\boldsymbol{\beta}+\mathbf{t}(x)\mathbf{V}^{-1}(\mathbf{y}-\mathbf{H}\boldsymbol{\beta}) \tag{A3}$$

$$C^*(\mathbf{x},\mathbf{x}')=C(\mathbf{x},\mathbf{x}')-\mathbf{t}(\mathbf{x})^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{t}(\mathbf{x}') \tag{A4}$$

$$\mathbf{t}(\mathbf{x})=[C(\mathbf{x},\mathbf{x}_1),\ldots,C(\mathbf{x},\mathbf{x}_n)]^{\mathrm{T}} \tag{A5}$$

Removing the conditioning on $\boldsymbol{\beta}$ using standard integration techniques, obtain the posterior distribution

$$\eta(\cdot)|\mathbf{y},\sigma^2\sim\mathcal{N}(m^{**}(\cdot),\sigma^2C^{**}(\cdot,\cdot)) \tag{A6}$$

where

$$m^{**}(\mathbf{x})=\mathbf{h}(\mathbf{x})^{\mathrm{T}}\widehat{\boldsymbol{\beta}}+\mathbf{t}(\mathbf{x})\mathbf{V}^{-1}(\mathbf{y}-\mathbf{H}\widehat{\boldsymbol{\beta}}) \tag{A7}$$

$$C^{**}(\mathbf{x},\mathbf{x}')=C^*(\mathbf{x},\mathbf{x}')+(\mathbf{h}(\mathbf{x})^{\mathrm{T}}-\mathbf{t}(\mathbf{x})^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{H})(\mathbf{H}^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{H})^{-1}(\mathbf{h}(\mathbf{x})^{\mathrm{T}}-\mathbf{t}(\mathbf{x})^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{H})^{\mathrm{T}} \tag{A8}$$

$$\widehat{\boldsymbol{\beta}}=(\mathbf{H}^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{y} \tag{A9}$$

Regarding Equation (6), q is the rank of $\mathbf{H}$. The term $\widehat{\sigma}$ in the expression is equal to

$$\widehat{\sigma}^2=\frac{\mathbf{y}^{\mathrm{T}}(\mathbf{V}^{-1}-\mathbf{V}^{-1}\mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{V}^{-1}\mathbf{H})\mathbf{H}^{\mathrm{T}}\mathbf{V}^{-1})\mathbf{y}}{n-q-2} \tag{A10}$$

DOI: 10.1002/nme

ACKNOWLEDGEMENTS

FADO gratefully acknowledges the support of the Engineering and Physical Sciences Research Council (EPSRC) for the award of a studentship through an Ideas Factory grant and the Consejo Nacional de Ciencia y Tecnología (CONACYT) for the award of a scholarship from the Mexican government. SA gratefully acknowledges the support of the Leverhulme Trust through the award of the Philip Leverhulme Prize and the Royal Society of London through the Wolfson Research Merit award.

REFERENCES

1. Zienkiewicz OC, Taylor RL. *The Finite Element Method* (4th edn). McGraw-Hill: London, 1991.
2. Bathe KJ. *Finite Element Procedures*. Prentice-Hall Inc.: Englewood Cliffs, NJ, U.S.A., 1995.
3. Cook RD, Malkus DS, Plesha ME, Witt RJ. *Concepts and Applications of Finite Element Analysis* (4th edn). Wiley: New York, U.S.A., 2001.
4. Hughes TJR. *The Finite Element Method*: *Linear Static and Dynamic Finite Element Analysis*. Dover Publications: New York, U.S.A., 2000.
5. Petyt M. *Introduction to Finite Element Vibration Analysis*. Cambridge University Press: Cambridge, U.K., 1998.
6. Shinozuka M, Yamazaki F. In Stochastic finite element analysis: an introduction. In *Stochastic Structural Dynamics: Progress in Theory and Applications*, Ariaratnam ST, Schuëller GI, Elishakoff I (eds). Elsevier Applied Science: London, 1998.
7. Kleiber M, Hien TD. *The Stochastic Finite Element Method*. Wiley: Chichester, 1992.
8. Matthies HG, Brenner CE, Bucher CG, Soares CG. Uncertainties in probabilistic numerical analysis of structures and solids—stochastic finite elements. *Structural Safety* 1997; **19**(3):283–336.
9. Manohar CS, Adhikari S. Dynamic stiffness of randomly parametered beams. *Probabilistic Engineering Mechanics* 1998; **13**(1):39–51.
10. Manohar CS, Adhikari S. Statistical analysis of vibration energy flow in randomly parametered trusses. *Journal of Sound and Vibration* 1998; **217**(1):43–74.
11. Adhikari S, Manohar CS. Dynamic analysis of framed structures with statistical uncertainties. *International Journal for Numerical Methods in Engineering* 1999; **44**(8):1157–1178.
12. Adhikari S, Manohar CS. Transient dynamics of stochastically parametered beams. *ASCE Journal of Engineering Mechanics* 2000; **126**(11):1131–1140.
13. Haldar A, Mahadevan S. *Reliability Assessment Using Stochastic Finite Element Analysis*. Wiley: New York, U.S.A., 2000.
14. Sudret B, Der-Kiureghian A. Stochastic finite element methods and reliability. *Technical Report UCB/SEMM-2000/08*, Department of Civil & Environmental Engineering, University Of California, Berkeley, 2000.
15. Nair PB, Keane AJ. Stochastic reduced basis methods. *AIAA Journal* 2002; **40**(8):1653–1664.
16. Elishakoff I, Ren YJ. *Large Variation Finite Element Method for Stochastic Problems*. Oxford University Press: Oxford, U.K., 2003.
17. Sachdeva SK, Nair PB, Keane AJ. Comparative study of projection schemes for stochastic finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(19–22):2371–2392.
18. Sachdeva SK, Nair PB, Keane AJ. Hybridization of stochastic reduced basis methods with polynomial chaos expansions. *Probabilistic Engineering Mechanics* 2006; **21**(2):182–192.
19. Kleijnen JPC. Kriging metamodeling in simulation: a review. *European Journal of Operational Research* 2009; **192**(3):707–716.
20. Craig KJ, Stander N, Dooge DA, Varadappa S. Automotive crashworthiness design using response surface-based variable screening and optimization. *Engineering Computations* 2005; **22**(1):38–61.
21. Pérez VM, Renaud JE, Watson LE. Reduced sampling for construction of quadratic response surface approximations using adaptive experimental design. *Engineering Computations* 2008; **25**(8):764–782.
22. Fan H, Lampinen J, Levy Y. An easy-to-implement differential evolution approach for multi-objective optimizations. *Engineering Computations* 2006; **23**(2):124–138.
23. Santner T, Williams B, Notz W. *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. London, U.K., 2003.
24. Sacks J, Welch W, Mitchell T, Wynn H. Design and analysis of computer experiments. *Statistical Science* 1989; **4**(4):409–435.
25. Kennedy MC, Anderson CW, Conti S, O'Hagan A. Case studies in Gaussian process modelling of computer codes. *Reliability Engineering and System Safety* 2006; **91**(10–11):1301–1309.
26. Challenor P, Hankin R, Marsh R. *Avoiding Dangerous Climate Change*. Cambridge University Press: Cambridge, U.K., 2006.
27. Rougier J. Probabilistic inference for future climate using an ensemble of climate model evaluations. *Climatic Change* 2007; **81**(3):247–264.
28. Xiong Y, Chen W, Appley D, Ding X. A non-stationary covariance-based Kriging method for metamodelling in engineering design. *International Journal for Numerical Methods in Engineering* 2006; **71**:733–756.
29. Haylock R, O'Hagan A. *Bayesian Statistics*, vol. 5. Oxford University Press: Oxford, U.K., 1996.
30. DiazDelaO FA, Adhikari S. Structural dynamic analysis using Gaussian process emulators. *Engineering Computations* 2010; **27**(5):580–605.

DOI: 10.1002/nme

31. O'Hagan A. Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering and System Safety* 2006; **91**(10–11):1290–1300.
32. Kennedy MC, O'Hagan A. Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B—Statistical Methodology* 2001; **63**(3):425–450.
33. Devijver P, Kittler J. *Pattern Recognition*: *A Statistical Approach*. Prentice-Hall: London, U.K., 1982.
34. Adhikari S. Rates of change of eigenvalues and eigenvectors in damped dynamic system. *AIAA Journal* 1999; **37**(11):1452–1458.
35. Adhikari S. Matrix variate distributions for probabilistic structural mechanics. *AIAA Journal* 2007; **45**(7):1748–1762.
36. Adhikari S. On the quantification of damping model uncertainty. *Journal of Sound and Vibration* 2007; **306**(1–2):153–171.
37. Adhikari S. Wishart random matrices in probabilistic structural mechanics. *ASCE Journal of Engineering Mechanics* 2008; **134**(12):1029–1044.
38. Ghanem R, Spanos P. *Stochastic Finite Elements*: *A Spectral Approach*. Springer: New York, U.S.A., 1991.
39. McKay M, Conover W, Beckman R. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 1979; **21**(2):239–245.
40. Haylock R. Bayesian inference about outputs of computationally expensive algorithms with uncertainty on the inputs. *Ph.D. Thesis*, University of Nottingham, Nottingham, U.K., 1996.
41. Chambers J, Cleveland W, Kleiner B, Tuckey P. *Graphical Methods for Data Analysis*. Chapman & Hall: London, 1983.
42. Greene WH. *Econometric Analysis*. Prentice-Hall: New Jersey, 2002.
43. Der Kiureghian A, Li CC. Structural reliability under incomplete probability information. *Journal of Engineering Mechanics* (ASCE) 1986; **112**(1):85–104.
44. Ditlevsen O, Marsden H. *Structural Reliability Methods*. Wiley: Chichester, 1996.
45. Hurtado JE, Barbat AH. Monte Carlo techniques in computational stochastic mechanics. *Archives of Computational Methods in Engineering* 1998; **5**(1):3–29.
46. Papadrakakis M, Papadopoulos V. Robust and efficient methods for stochastic finite element analysis using Monte Carlo simulation. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**(3–4):325–340.
47. Elishakoff I, Ren YJ, Shinozuka M. Improved finite-element method for stochastic problems. *Chaos Solitons and Fractals* 1995; **5**(5):833–846.
48. Yamazaki F, Shinozuka M, Dasgupta G. Neumann expansion for stochastic finite element analysis. *Journal of Engineering Mechanics* (ASCE) 1988; **114**(8):1335–1354.
49. Falsone G, Impollonia N. A new approach for the stochastic analysis of finite element modelled structures with uncertain parameters. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**(44):5067–5085.
50. Falsone G, Impollonia N. A new approach for the stochastic analysis of finite element modelled structures with uncertain parameters (vol. 191, p. 5067, 2002). *Computer Methods in Applied Mechanics and Engineering* 2003; **192**(16–18):2187–2188.
51. Muscolino G, Ricciardi G, Impollonia N. Improved dynamic analysis of structures with mechanical uncertainties under deterministic input. *Probabilistic Engineering Mechanics* 2000; **15**(2):199–212.
52. Impollonia N, Muscolino G. Static and dynamic analysis of non-linear uncertain structures. *Meccanica* 2002; **37**(1–2):179–192.
53. Impollonia N, Ricciardi G. Explicit solutions in the stochastic dynamics of structural systems. *Probabilistic Engineering Mechanics* 2006; **21**(2):171–181.
54. Xiu DB, Karniadakis GE. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing* 2002; **24**(2):619–644.
55. Xiu DB, Karniadakis GE. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics* 2003; **187**(1):137–167.
56. Wan XL, Karniadakis GE. Beyond Wiener–Askey expansions: handling arbitrary pdfs. *Journal of Scientific Computing* 2006; **27**(3):455–464.
57. Debusschere B, Najm H, Pebay P, Knio O, Ghanem R, LeMaitre O. Numerical challenges in the use of polynomial chaos representations for stochastic processes. *SIAM Journal on Scientific Computing* 2005; **22**(2):698–719.
58. Berveiller M, Sudret B, Lemaire M. Stochastic finite element: a non-intrusive approach by regression. *Review of Européenne Mécanique Numérique* 2006; **15**(1–3):81–92.
59. Choi SK, Grandhi RV, Canfield RA. Structural reliability under non-Gaussian stochastic behavior. *Computers and Structures* 2004; **82**(13–14):1113–1121.
60. Sudret B. Uncertainty propagation and sensitivity analysis in mechanical models—contributions to structural reliability and stochastic spectral methods. *Habilitation à Diriger des Recherches*, Université Blaise Pascal, Clermont-Ferrand, France, 2007.
61. Blatman G, Sudret B. Sparse polynomial chaos expansions and adaptive stochastic finite elements using a regression approach. *Comptes Rendus Mecanique* 2008; **336**:518–523.
62. George A. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall: New Jersey, U.S.A., 1981.
63. Terje H, Der Kiureghian A. FERUM: Finite Element Reliability Using Matlab. Available from: http://www.ce.berkeley.edu/projects/ferum/.
64. Krzanowski W. *Principles of Multivariate Analysis*. Oxford University Press: Oxford, U.K., 2000.

DOI: 10.1002/nme