Abstract

A method for efficiently representing realizations of computationally expensive random fields is proposed. The approach, known as the emulator, consists of building a statistical approximation of realizations of such random fields. It is based on few runs of a code that performs the discretization of the random field via the Karhunen-Loève expansion. This study is aimed at the incorporating emulators to reduce the computational cost of stochastic finite element codes. Numerical results of emulating realizations of a Gaussian homogeneous two-dimensional random field are presented. It is shown that proposed approach can reduce simulation cost of random fields for stochastic finite element applications.

Keywords: stochastic finite element method, stochastic partial differential equations, Karhunen-Loeve expansion, gaussian stochastic process, Monte Carlo simulation, bayesian statistics.

1 Introduction

The realistic modeling and prediction of engineering systems can be better achieved when uncertainty is taken into account. This can be done by incorporating uncertainty into the partial differential equations that govern the system's response. The random aspects of modelling might involve uncertain material properties, uncertain boundary conditions, uncertain manufacturing tolerances. These uncertainties can be represented by stochastic coefficients in a system of stochastic partial differential equations.

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) 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 [1]. Such extension is known as the Stochastic Finite Element Method (SFEM), which incorporates random field models for the elastic, mass and damping properties of a given engineering system [2]. Mathematically, the problem is formulated as follows: find the response vector $\mathbf{u} \in \mathbb{R}^n$ such that

$$\mathbf{\Lambda u} = \mathbf{f} \tag{1}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is a stochastic differential operator and $\mathbf{f} \in \mathbb{R}^n$ is the (possibly random) excitation vector. Broadly speaking, the SFEM aims at characterizing the global probabilistic structure of the random response $\mathbf{u}$. However, when a system with a large number of degrees of freedom is investigated, a code designed to solve Equation (1) can become prohibitively expensive to run. Such cost can be measured, for example, in terms of the CPU time employed or the required computer capability. These codes, as well as the underlying mathematical models will be henceforth referred to as *simulators*. A simulator is a function $\eta : \mathbb{R}^n \rightarrow \mathbb{R}$ that, given an input $\mathbf{x}$, it returns an output $\mathbf{y} = \eta(\mathbf{x})$. There already exist several methods to reduce the execution cost of an expensive simulator $\eta(\cdot)$, such as local polynomial regression and neural networks. Another possible approach is the Bayesian Analysis of Computer Code Outputs [3]. This technology is based on the Analysis and Design of Computer Experiments [4] and uses concepts of Bayesian Statistics. It essentially consists in constructing an approximation to the simulator, called an *emulator*. More precisely, an emulator is a statistical approximation to the simulator $\eta(\cdot)$, in the sense that it provides a probability distribution for it. The main idea behind emulation is the following: Given some prior beliefs about $\eta(\cdot)$, an initial set of runs is treated as training data that will be used to update such beliefs. As it will be seen, these prior information will rely on a Gaussian stochastic process. The number of training runs will be small relative to the size of the input domain of the simulator, since by assumption it is computer intensive. Upon updating, the emulator will interpolate/extrapolate the available data at unsampled inputs, whereas it will return the known value of the simulator at each of the initial runs. Emulators have already been implemented in several scientific fields. A brief account of such applications can be found in [5].

The present paper will explore the use of emulators in the SFEM in order to reduce the computational cost of studying engineering systems with random mechanical properties. In Section 2, the problem posed by Equation (1) will be set in the context of Stochastic Mechanics. Emphasis will be put on explaining how to discretize the random field associated with the operator $\mathbf{\Lambda}$. Section 3 will provide an overview of the theory behind the implementation of emulators. In section 4, an emulator will be used to produce realizations of a discretized random field.

2 Random Field Discretization

Let $(\Theta, \mathcal{F}, P)$ be a probability space and let $\mathcal{L}^2(\Theta, \mathcal{F}, P)$ be the Hilbert space of random variables with finite second moment. A *random field* $\mathcal{H}(\mathbf{x}, \theta)$, with $\mathbf{x} \in \mathbb{R}^n$ and

$\theta \in \Theta$, is a curve in $\mathcal{L}^2(\Theta, \mathcal{F}, P)$, that is a collection of random variables indexed by $\mathbf{x}$ [6]. Hence, $\mathcal{H}(\mathbf{x_0}, \theta)$ for a given $\mathbf{x_0}$ is a random variable and $\mathcal{H}(\mathbf{x}, \theta_0)$ for a given θ_0 is a realization of the random field. A *Gaussian random field* is such that for any N, the vector $[\mathcal{H}(\mathbf{x}_1, \theta_0), \ldots, \mathcal{H}(\mathbf{x}_N, \theta_0)]^T$ is Gaussian. Moreover, it is *homogeneous* if its mean $\mu(\mathbf{x}, \theta_0)$ and variance $\sigma^2(\mathbf{x}, \theta_0)$ are constant and its autocorrelation coefficient $\rho(\mathbf{x}, \mathbf{x}')$ is a function of $\mathbf{x}$ and $\mathbf{x}'$ only. This autocorrelation function may take many distinct forms. One example is the one of the exponential type

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

where $\mathbf{x}_{(i)}$ denotes the i-th coordinate of $\mathbf{x}$ and α_1, α_2 are known as *correlation lengths*.

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

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{3}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is known as the stiffness matrix and $\mathbf{f} \in \mathbb{R}^n$ is the forcing vector. If uncertainty is to be taken into account, as in Equation (1), $\mathbf{K}$ becomes a random matrix. This means that the stiffness matrix becomes a function of the spatial coordinates and a random dimension, namely

$$\mathbf{K}(\mathbf{x}, \theta) = \mathcal{H}(\mathbf{x}, \theta)\mathbf{K_{det}} \tag{4}$$

where $\mathbf{K_{det}}$ is the deterministic part of the stiffness matrix. To keep notation simple, $\mathbf{K}(\mathbf{x}, \theta)$ will be denoted as $\mathbf{K}$ but understood as being a random matrix. An analogous treatment apply to $\mathbf{u}(\mathbf{x}, \theta)$ and $\mathbf{f}(\mathbf{x}, \theta)$.

Suppose one is interested in quantifying the uncertainty in Equation (3) induced by the random properties in the system. 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, $\mathcal{H}(\mathbf{x}, \theta)$ needs to be discretized in order to solve the associated system of random algebraic equations. The solution to this problem will enable uncertainty quantification. An advantageous alternative for discretizing $\mathcal{H}(\mathbf{x}, \theta)$ is the Karhunen-Loeve expansion (KLE), for which

$$\mathcal{H}(\mathbf{x}, \theta) = \sum_{i=0}^{\infty} \sqrt{\lambda_i} \xi_i(\theta) \phi_i(\mathbf{x}) \tag{5}$$

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

$$\int_{\mathbb{R}^n} K(\mathbf{x}_1, \mathbf{x}_2) \phi_i(\mathbf{x}) d\mathbf{x}_1 = \lambda_i \phi_i(\mathbf{x}_2) \tag{6}$$

According to Sudret and Der Kiureghian [6], the truncated KLE of $\mathcal{H}(\mathbf{x}, \theta)$ up to M terms is defined as

$$\mathcal{H}(\mathbf{x}, \theta) = \mu(\mathbf{x}) + \sum_{i=1}^{M} \sqrt{\lambda_i} \xi_i(\theta) \phi_i(\mathbf{x}) \tag{7}$$

They also show that applying the KLE, the system in Equation (3) becomes

$$[\mathbf{K}_0 + \sum_{i=0}^{\infty} \mathbf{K}_i \xi(\theta)] \mathbf{u} = \mathbf{f} \tag{8}$$

where each $\mathbf{K}_i$ is a deterministic matrix.

Note that this is not the only available technique to discretize the random field $\mathcal{H}(\mathbf{x}, \theta)$. There are more discretization methods that can be divided into three categories, at least: point discretization, average discretization and series expansion methods [6]. 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 [7] 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 $\mathbf{u}$ in order to assess the reliability of the system. However, this objective can prove to be difficult to achieve and the approach is limited to obtaining the first few statistical moments of $\mathbf{u}$. There exist several strategies, such as perturbation and projection methods, to deal with this problem. An account of these methods is given in [8].

Before attempting to incorporate emulators to the solution process of the above problems, this paper will focus on the problem of efficiently generating realizations of the random field $\mathcal{H}(\mathbf{x}, \theta)$. It will be shown that this can be a computationally expensive task as the number of points in which the random field realization is to be evaluated increases. The approach adopted here will prove useful compared against a Monte Carlo simulation setup, where the principle is to simulate a large number of realizations of the random field, then compute for each sample the response vector $\mathbf{u}$ and treat such response statistically.

3 Emulators

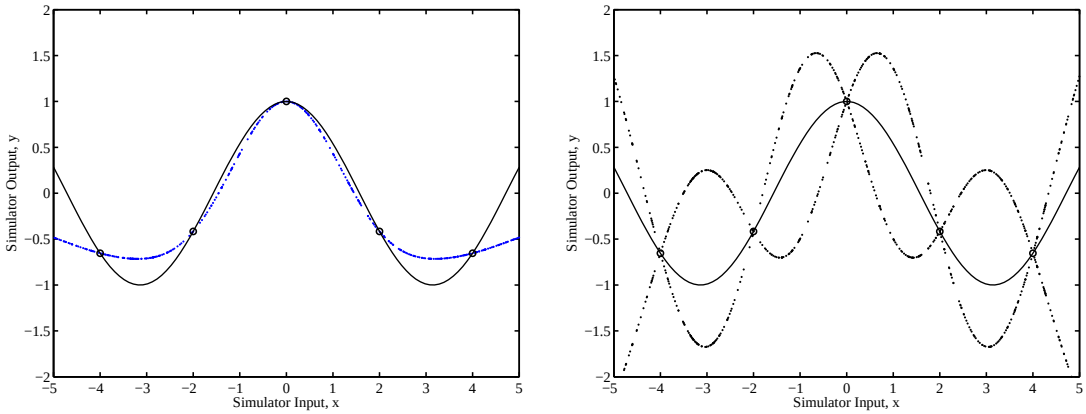
Suppose that N points, namely $\mathbf{x}_1, \ldots, \mathbf{x}_N$, are chosen in the input domain of the simulator $\eta(\cdot)$. Each of these will be known as a *design point*. The set $\{\mathbf{y}_1 = \eta(\mathbf{x}_1), \ldots, \mathbf{y}_N = \eta(\mathbf{x}_N)\}$, resulting from the evaluation of $\eta(\cdot)$ in each design point, will be called *training set*. According to O'Hagan [3], an emulator of $\eta(\cdot)$ should satisfy some minimal criteria:

1. It should produce the known value of the output for each design point, with no uncertainty.

2. At any **x** that is not a design point, it should provide a probability distribution whose mean value constitutes a plausible interpolation/extrapolation of the training data. Moreover, the probability distribution around this mean value should represent the uncertainty about how the emulator might interpolate/extrapolate.

Naturally, it is also desirable that emulation results at least as efficient as other available techniques, in terms of reducing the associated computational cost.

To illustrate the above criteria, take a trivial one-dimensional simulator and suppose for a moment it is computer intensive. Figure 1(a) depicts the case when five training runs (the circles) are used. The mean of the distribution provided by the emulator (the dots) approximates the real values of the simulator (the solid line) at several untried inputs across the input domain. As required, it returns the known value of the simulator at each training run. Figure 1(b) shows upper and lower bounds of two standard deviations for the mean of the distribution provided by the emulator. Note how uncertainty is equal to zero in each of the training runs, as it would be expected, since the emulator returns the true value of the simulator in these points.



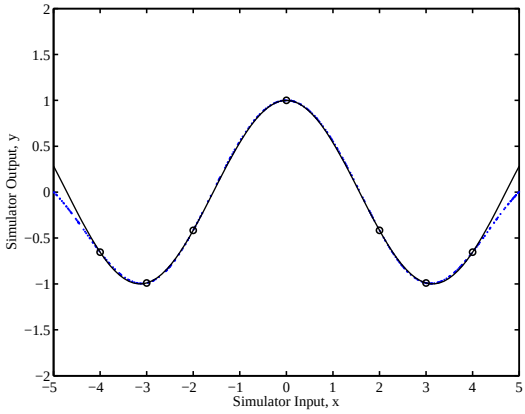
(a) Interpolation/extrapolation with 5 design points.

(b) The corresponding uncertainty. Two standard deviations.

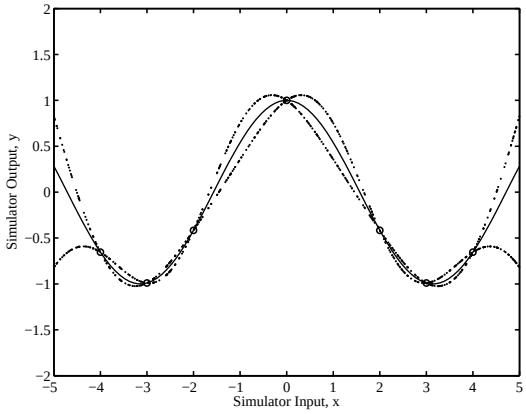
Figure 1: Approximation to the simulator $\mathbf{y} = \cos(\mathbf{x})$ and uncertainty about its mean, using 5 design points. The solid line is the true output of the simulator. The circles represent the training runs, and the dots are the mean of the posterior distribution and the uncertainty bounds, respectively.

Suppose the number of training runs is increased. Note how the approximation improves, as shown in Figure 2(a). In the same way, Figure 2(b) depicts how uncertainty is reduced. Observe however that despite the number of training points employed, uncertainty increases very rapidly when extrapolating the training set.

That $\eta(\cdot)$ is an expensive simulator means that it can only be evaluated at a limited number of inputs. From the perspective of Bayesian Statistics, $\eta(\cdot)$ is a random variable in the sense that its value is unknown until it is actually run. Thus, a characterization of the relationship between the input and the unknown output must first be



(a) Interpolation/extrapolation with 7 design points.



(b) The corresponding uncertainty. Two standard deviations.

Figure 2: Approximation to the simulator $\mathbf{y} = \cos(\mathbf{x})$ and uncertainty about its mean, using 7 design points. The solid line is the true output of the simulator. The circles represent the training runs, and the dots are the mean of the posterior distribution and the uncertainty bounds, respectively.

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 the random function $\eta(\cdot)$ is of the form

$$\eta(\mathbf{x}) = \mathbf{h}(\cdot)^T \boldsymbol{\beta}(\mathbf{x}) + Z(\mathbf{x}) \tag{9}$$

where $\mathbf{h}(\cdot) \in \mathbb{R}^N$ is a vector of known functions and $\boldsymbol{\beta}$ is a vector of unknown coefficients. The function $Z(\cdot)$ is assumed to be a stochastic process with mean zero and covariance function $Cov(Z(\mathbf{x}), Z(\mathbf{x}^{'}))$. A common choice for $Z(\cdot)$ is the *Gaussian stochastic process*. It is said that $Z(\cdot)$ is a Gaussian stochastic process if for any choice $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^n$, the vector $[Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_N)]^T$ has a multivariate normal distribution. As noted in [9], it 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. From this definition, the expression $\eta(\cdot) \sim N(m(\cdot), \sigma^2 C(\cdot, \cdot))$ means that $\eta(\cdot)$ has a Gaussian process distribution with mean function $m(\cdot)$ and covariance $\sigma^2 C(\cdot, \cdot)$. If a linear structure of the form $m(\cdot) = \mathbf{h}(\cdot)^T \boldsymbol{\beta}$ is assumed for the mean, the interpretation of Eq. (9) becomes clear, that is, the random function $\eta(\cdot)$ deviates from the mean of its distribution following a Gaussian stochastic process.

An important assumption is to consider $\eta(\cdot)$ to be 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 viceversa. 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 emulators over Monte Carlo methods, as pointed out by O'Hagan [3].

A common choice of a covariance function, and the one that will be adopted hereafter is

$$Cov(\eta(\mathbf{x}), \eta(\mathbf{x}^{'})) = \sigma^2 e^{-(\mathbf{x}-\mathbf{x}^{'})^T \mathbf{B} (\mathbf{x}-\mathbf{x}^{'})} \tag{10}$$

where $\mathbf{B}$ is a positive definite diagonal 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 $\mathbf{h}(\cdot)^T \boldsymbol{\beta}$ and covariance expressed by Equation (10). Mathematically,

$$\eta(\cdot)|\boldsymbol{\beta}, \sigma^2 \sim N(\mathbf{h}(\cdot)^T \boldsymbol{\beta}, \sigma^2 C(\cdot, \cdot)) \tag{11}$$

This prior distribution contains subjective information about the relationship between the input and the unknown outputs. This prior belief must be updated by adding objective information. Suppose there are N design points that produce the vector of observations $\mathbf{y} = [\mathbf{y}_1 = \eta(\mathbf{x}_1), \ldots, \mathbf{y}_N = \eta(\mathbf{x}_N)]^T$ and the output of $\eta(\cdot)$ at an untried input $\mathbf{x}$ is to be estimated. Carrying out the suitable integration as in [10], it can be shown that the distribution that results from incorporating the observations $\mathbf{y}$ is

$$\eta(\cdot)|\mathbf{y}, \sigma^2 \sim N(m^{**}(\cdot), \sigma^2 C^{**}(\cdot, \cdot)) \tag{12}$$

Conveniently enough, this posterior distribution is also a Gaussian process distribution. The complete expressions of $m^*(\cdot)$, $C^*(\cdot, \cdot)$, $m^{**}(\cdot)$ and $C^{**}(\cdot, \cdot)$, as well as the procedure to obtain the posterior distribution (12) from the prior distriburtion (11), 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 [10] also show that

$$\frac{\eta(\mathbf{x}) - m^{**}(\mathbf{x})}{\widehat{\sigma}\sqrt{\frac{(n-q-2)C^{**}(\mathbf{x})}{n-q}}} \sim t_{n-q} \tag{13}$$

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

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

Algorithm 1: Emulation

1. Select N initial design points $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

2. Obtain the vector of observations $\mathbf{y} = [\mathbf{y}_1 = \eta(\mathbf{x}_1), \ldots, \mathbf{y}_n = \eta(\mathbf{x}_n)]^T$.

3. Update the prior distribution (11), which contains subjective information, by adding the objective information $\mathbf{y}$ and obtaining the posterior distribution (12). Calculate $m^{**}(\cdot)$, the mean of the updated posterior distribution given the data $\mathbf{y}$. Such mean constitutes an approximation of $\eta(\mathbf{x})$ for any $\mathbf{x}$.

Step 1 in the algorithm above requires some consideration. The selection of the initial design must be done carefully when dealing with an expensive simulator. It would be ideal to extract the most information about $\eta(\cdot)$ out of the minimum number of evaluations possible. One strategy for selecting a set of inputs to evaluate the code is to choose the design points such that they are evenly spread throughout the input domain of $\eta(\cdot)$. In that case, random sampling from the distribution of the inputs could be a suitable strategy. Nevertheless, this scheme might have a disadvantage. Consider a case where the output is influenced by only a few components of each input vector. McKay et al. [11] proposed *Latin hypercube sampling* as a solution to this problem. Latin hypercube sampling can be viewed as an extension of Latin square designs to higher dimensions. Each dimension is guaranteed to be fully represented. Note that this is not the only existing strategy. An overview of some other approaches can be found in [8].

4 Application

Consider a Gaussian homogeneous two-dimensional random field $\mathcal{H}(\mathbf{x}, \theta)$ with mean $\mu = 50$, variance $\sigma^2 = 0.09$ and exponential autocorrelation function with correlation lengths $[0.03, 0.03]^T$. Let the order of the KLE be $M = 10$. The estimation of these parameters is an interesting problem in its own right. Since the purpose of the present study is to explore the capabilities of emulators in the SFEM context, the parameters above are chosen freely and following no particular methodology. Given this information, the relevant simulator is

$$\eta(\mathbf{x}, \theta) = \mu + \sum_{i=1}^{10} \sqrt{\lambda_i} \xi_i(\theta) \phi_i(\mathbf{x}) \tag{14}$$

The spatial domain of the simulator was chosen to be the square region $\mathcal{R}_L = [0, L] \times [0, L]$. This domain was then divided into square elements and the coordinates of the nodal points calculated ($(L+1)^2$ in total). The truncated KLE was then calculated based on the results by Ghanem and Spanos [1], who provide analytical expressions for the eigenvalues and eigenfunctions corresponding to a random field with the above characteristics. Let $L = 20$. Figure 3 shows a realization, $\mathcal{H}(\mathbf{x}, \theta_{\mathbf{0}})$, of the above random field, which in terms of Equation (14) would be $\eta(\mathbf{x}, \theta_{\mathbf{0}})$ for every $\mathbf{x} \in \mathcal{R}_{20}$. In order to apply Algorithm 1, the design points $(\mathbf{x}, \theta)_1, \ldots, (\mathbf{x}, \theta)_N$ were selected

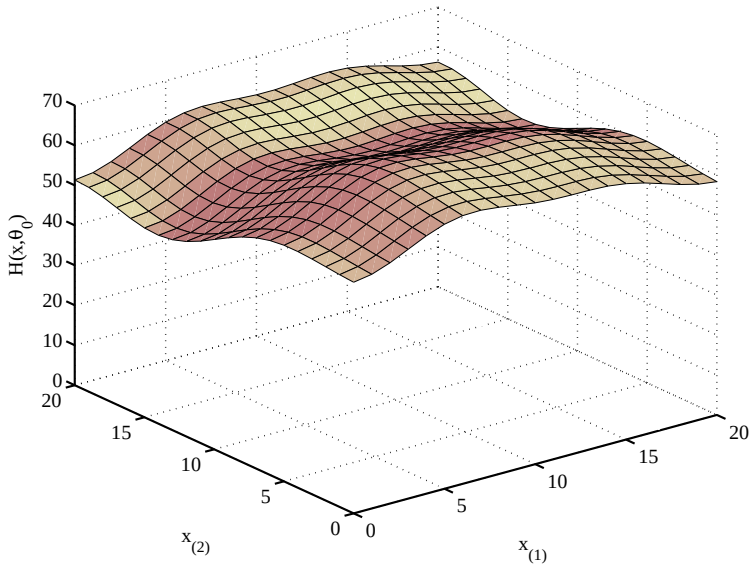


Figure 3: Realization of the Gaussian homogeneous random field $\mathcal{H}(\mathbf{x}, \theta)$.

according to a Latin hypercube design. The value of N was chosen to be 10% of the number of nodes. Note however that the design points do not necessarily, and in general do not coincide with any of the nodes, if a Latin hypercube design is to be selected. The training runs were used to construct the posterior distribution (12) and its mean $m^{**}(\cdot)$ was calculated to infer the values of the random field realization in each node. As for the calculation smoothness parameters, a method suggested by Haylock [12] was implemented. Briefly speaking, if **b** is the diagonal of the matrix **B** in Equation (10) then the density function $f(\mathbf{B}|\mathbf{y})$ was derived and a maximum likelihood estimator of **b** was obtained. Figure 4 shows the values of $m^{**}(\cdot)$ compared with the actual realization of the random field, as well as the design points employed in the construction of the emulator. Being $\mathcal{H}(\mathbf{x}, \theta)$ a two-dimensional random field, appreciation of the accuracy of the approximation provided by the emulator in Figure 4 can be difficult. Figure 5, on the contrary, offers a comparison between the real values of the random field realization and the emulated values. The exercise above was carried out for several values of L and thus an increasing number of nodes. The time employed to produce one realization of the corresponding random field for an increasing number of nodes is shown in Table 1.

5 Conclusions

The Stochastic Finite Element Method extends its deterministic counterpart, the Finite Element Method, by including random field models that account for the uncertainty present in real-life engineering systems. The discretization of such random fields,

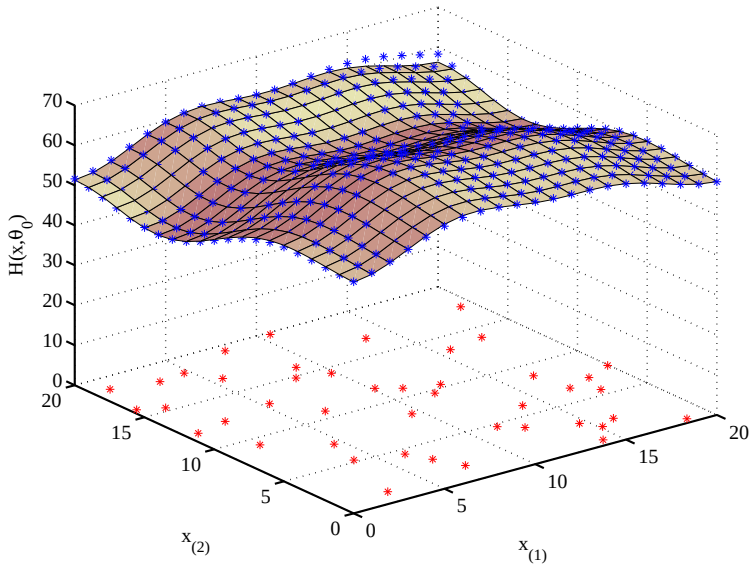


Figure 4: Emulation of the values of $\mathcal{H}(\mathbf{x}, \theta)$ at the nodal points. The initial design is shown lying on the lower plane.

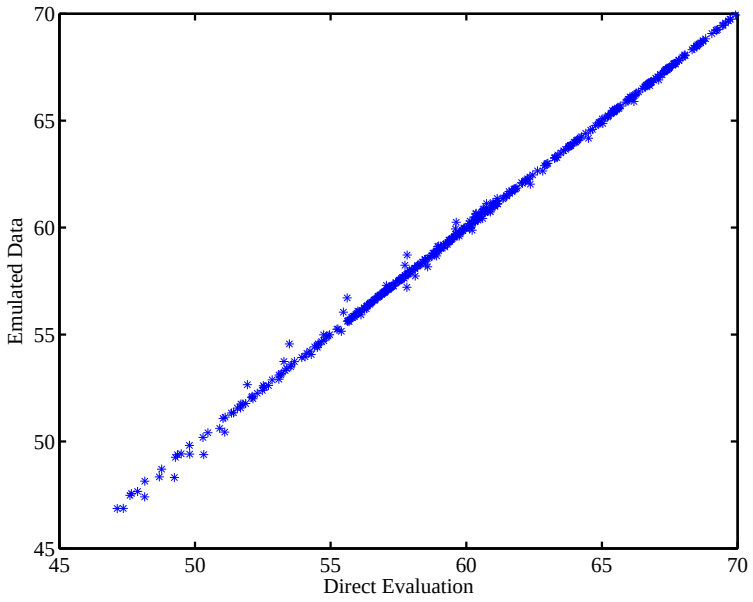


Figure 5: Emulation of the values at the nodal points.

necessary to solve the associated system of stochastic partial differential equations is usually carried out using the Karhunen-Loeve expansion. Obtaining a realization of a random field $\mathcal{H}(\mathbf{x}, \theta)$ can be time-consuming, especially if one is to evaluate this realization in a large number of points throughout the spatial dimension $\mathbf{x}$. To tackle

No. Nodes	Time (secs.) Direct	Time (secs.) Emulator
121	9.56	0.07
256	19.92	0.24
441	34.43	0.75
961	76.23	6.05
1681	131.29	17.76
2601	273.18	59.66

Table 1: Number of nodes vs. CPU time employed

this difficulty, the use of Gaussian process emulators has been proposed. Emulators provide a fast approximation to the values of a random field realization using only a relatively small set of such values as training data. A realization of a Gaussian homogeneous two-dimensional random field was emulated, increasing the number of nodal points to be evaluated, using the information provided by a set of training runs a tenth of the size of the training set. Good agreement between the original and the emulated values was observed.

The next step in the integration of emulators and the SFEM is the inclusion of more general non-Gaussian random fields that represent the mechanical properties of an engineering system in a more realistic manner. The main objective however, is to emulate random fields that model the response of the system and the evaluation of the efficiency of that approach against current available methodologies.

Appendix

A.1. Updating of the Prior Distribution

In this section the process of updating the prior distribution (11) to obtain the posterior distribution (12) is outlined. Define $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_N)]^T$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $\mathbf{A}_{\ell j} = C(x_\ell, x_j)\ \forall \ell, j \in \{1, \ldots, N\}$. Hence

$$\mathbf{y}|\boldsymbol{\beta}, \sigma^2 \sim N(\mathbf{H}\boldsymbol{\beta}, \sigma^2 \mathbf{A}) \tag{15}$$

To incorporate the objective information $\mathbf{y}$ and obtain the distribution of $\eta(\cdot)|\mathbf{y}, \boldsymbol{\beta}, \sigma^2$, use of the following result, given in Krzanowski [13].

Theorem. Let $\mathbf{z} \in \mathbb{R}^n$ be a random vector such that $\mathbf{z} \sim N(\boldsymbol{\mu}, \Sigma)$. 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, $\mathbf{z}_1|\mathbf{z}_2 \sim N(\widetilde{\boldsymbol{\mu}}, \widetilde{\Sigma})$, 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

$$\eta(\cdot)|\mathbf{y}, \boldsymbol{\beta}, \sigma^2 \sim N(m^*(\cdot), \sigma^2 C^*(\cdot,\cdot)) \tag{16}$$

where

$$m^*(x) = \mathbf{h}(x)^T\boldsymbol{\beta} + \mathbf{t}(x)\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\boldsymbol{\beta}) \tag{17}$$

$$C^*(\mathbf{x}, \mathbf{x}') = C(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T A^{-1}\mathbf{t}(\mathbf{x}') \tag{18}$$

$$\mathbf{t}(\mathbf{x}) = [C(\mathbf{x}, \mathbf{x}_1), \ldots, C(\mathbf{x}, \mathbf{x}_N)]^T \tag{19}$$

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

$$\eta(\cdot)|\mathbf{y}, \sigma^2 \sim N(m^{**}(\cdot), \sigma^2 C^{**}(\cdot,\cdot)) \tag{20}$$

where

$$m^{**}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\widehat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\widehat{\boldsymbol{\beta}}) \tag{21}$$

$$C^{**}(\mathbf{x}, \mathbf{x}') = C^*(\mathbf{x}, \mathbf{x}') + (\mathbf{h}(\mathbf{x})^T - \mathbf{t}(\mathbf{x})^T\mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})^{-1}(\mathbf{h}(\mathbf{x})^T - \mathbf{t}(\mathbf{x})^T\mathbf{A}^{-1}\mathbf{H})^T \tag{22}$$

$$\widehat{\boldsymbol{\beta}} = (\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{A}^{-1}\mathbf{y} \tag{23}$$

Regarding Eq. (13), q is the rank of $\mathbf{H}$. The term $\widehat{\sigma}$ in the expression is equal to

$$\widehat{\sigma}^2 = \frac{\mathbf{y}^T(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}(\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})\mathbf{H}^T\mathbf{A}^{-1})\mathbf{y}}{N - q - 2} \tag{24}$$

Acknowledgments

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 EPSRC through the award of an Advanced Research Fellowship.

References

[1] Ghanem, R. G. and Spanos, P. D., *Stochastic Finite Elements: A Spectral Approach*, Dover Publications, New York, USA, 2003.

[2] Kleiber, M. and Hien, T. D., *The Stochastic Finite Element Method*, John Wiley, Chichester, UK, 1992.

[3] O'Hagan, A., "Bayesian analysis of computer code outputs: A tutorial," *Reliability Engineering & System Safety*, Vol. 91, No. 10-11, 2006, pp. 1290–1300.

[4] Satner, T., Williams, B., and Notz, W., *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, London, UK, 2003.

[5] DiazDelaO, F. A. and Adhikari, S., "Bayesian emulator approach for complex dynamical systems," *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, AIAA, Schaumburg, IL, USA, April 2008.

[6] Sudret, B. and Der Kiureghian, A., "Stochastic Finite Element Methods and Reliability: A State-of-the-Art Report," Tech. Rep. UCB/SEMM-2000/08, University of California, Berkley, USA, 2000.

[7] Devijver, P. and Kittler, J., *Pattern Recognition: A Statistical Approach*, Prentice-Hall, London, UK, 1982.

[8] Adhikari, S., "Uncertainty quantification and propagation in structural dynamics," *International Conference on Civil Engineering in the New Millennium: Opportunities and Challenges*, Howrah, India, January 2007.

[9] Kennedy, M. C. and O'Hagan, A., "Bayesian calibration of computer models," *Journal of the Royal Statistical Society Series B-Statistical Methodology*, Vol. 63, 2001, pp. 425–450, 3.

[10] Haylock, R. and O'Hagan, A., *Bayesian Statistics 5*, Oxford University Press, Oxford, UK, 1996.

[11] McKay, M., Conover, W., and Beckman, R., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, 1979, pp. 239–245.

[12] Haylock, R., *Bayesian Inference About Outputs of Computationally Expensive Algorithms with Uncertainty on the Inputs*, Ph.D. thesis, University of Nottingham, Nottingham, UK, 1966.

[13] Krzanowski, W., *Principles of Multivariate Analysis*, Oxford University Press, Oxford, UK, 2000.