Gaussian process emulators for the analysis of complex models in engineering.

Diaz De la O, Francisco Alejandro

Gaussian Process Emulators for the Analysis of Complex Models in Engineering

Swansea University
Prifysgol Abertawe

Submitted to Swansea University in fulfillment of the requirements for the degree of Doctor of Philosophy

by

Francisco Alejandro Díaz De la O

Swansea University

April 2011

To the memory of Guillermo Miguel Díaz Zamorano

When you set out on your trip to Ithaca,

wish that the journey is long...

K. Kavafis

Tarda en llegar y al final hay recompensa.

G. Cerati

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed .
Date May 10th, 2011

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed .
Date May 10th, 2011

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..
Date May 10th, 2011

Abstract

This dissertation explores a metamodeling tool that aims at reducing the computational cost of expensive computer models broadly used in engineering: the finite element method and the stochastic finite element method. The metamodeling tool, known as Gaussian process emulation, consists of building a statistical approximation to the output of expensive computer codes. Following the Bayesian paradigm, a small and carefully selected set of code runs is treated as training data used to update the prior beliefs about the code's output. These beliefs are represented as a Gaussian stochastic process prior distribution. After conditioning on the training runs and updating the prior distribution, the mean of the resulting posterior distribution approximates the output of the simulator at any untried input, whereas it reproduces the known output of the code at each initial input.

The use of Gaussian process emulators is justified due to the fact that in the analysis of engineering systems, running a detailed high-resolution computer model can be expensive even for obtaining the response at few points in the input domain. The incorporation of emulation is explored for several problems in engineering, including damped structural dynamics (both in a deterministic and non-deterministic context), the assimilation of multi-fidelity finite element models and domain decomposition, the approximation of random field realizations discretised via de Karhunen-Loève expansion, and the reduction of the computational cost of the polynomial chaos expansion.

By proposing several novel ideas and algorithms, it is shown that Gaussian process emulators can be an efficient and effective tool both for prediction and uncertainty quantification in the analysis of engineering systems.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Sondipon Adhikari. Without his permanent guidance, support, and friendship, this dissertation would not have been possible. I feel very fortunate to have had the opportunity to work and learn from him.

I would like to thank Dr. Jeremy Oakley for supervising me at The University of Sheffield and for teaching me Gaussian process emulation. I am also thankful to Prof. Andrew Cliffe and Prof. Tim Bedford, for their support and feedback every time we met in Glasgow.

My deep gratitude to Prof. Y. T. Feng and Prof. Keith Worden, for taking the time to read this dissertation and making comments to improve its readability.

I wish to gratefully acknowledge the support of the EPSRC and CONACYT for funding my studies.

I would like to thank Residential Services at Swansea University, in particular Natasha Edmonds. The opportunity of working with them during my studies was a privilege.

During these years in Swansea, I have met excellent researchers who have become good friends. Thanks to Faruque Ali, Aurelio Arranz, Rajib Chowdhury, Mauro Innocente, Erick Saavedra, and Rubén Sevilla. I have always enjoyed our discussions about the present and especially about the future.

Mexico is always on my mind. Thanks to all the family and friends who have supported me throughout the years. In particular, thanks to Gloria M. Díaz Zamorano, David E. López Campos, and Héctor Pérez López.

My deepest gratitude is of course to Juan Francisco, Elvia, and Mariana. I think of them each and every day of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for this Work

The behaviour of complex systems (e.g. physical, chemical, biological, among many other types) is usually investigated constructing mathematical models. In the last decades, as computing power has steadily increased, the implementation of such models in computer programmes or codes has become more frequent. This is justified because many processes are extremely complex and physical experiments can become prohibitively expensive or simply impossible. Computer experiments can thus become an alternative to study intricate phenomena. A computer code implementation of a mathematical model, also referred to as *simulator* (O'Hagan, 2006), can be understood as a function $\eta : \Omega \rightarrow \mathbb{R}^d$ whose domain is a p-dimensional input space $\Omega = \Omega_1 \times \ldots \times \Omega_p$. A simulator is *deterministic* if evaluating the same input $\mathbf{x} = (x_1, \ldots, x_p)$ results in the same output $\mathbf{y} = \eta(\mathbf{x})$. Deterministic simulators are a common tool when studying intricate phenomena in a wide range of disciplines. In particular, many engineering systems are complex enough to render physical experimentation very costly. As a consequence, these systems are often investigated running simulators often involving the finite element method (Zienkiewicz and Taylor, 1991; Bathe, 1995; Cook *et al.*, 2001; Hughes, 2000; Petyt, 1998) and the stochastic finite element method (Ghanem and Spanos, 1991; Matthies *et al.*, 1997; Shinozuka and Yamazaki, 1998; Haldar and Mahadevan, 2000; Nair and Keane, 2002; Elishakoff and Ren, 2003; Sachdeva *et al.*, 2006a,b; Li *et al.*, 2006). However, these simulators can have a high cost of execution, measured in terms of employed CPU time, number of floating point operations performed or required computer capability. For instance, the analysis of large engineering systems such as automobiles, aircrafts, and space shuttles usually employs finite element models with well over several million degrees of freedom. Therefore, a prodigious computer power is needed to carry out, for example, crashworthiness design (Thomke *et al.*, 1999).

Several strategies have been devised to reduce the computational cost of expensive simulators. Based on different underlying methodologies, these strategies are referred to as metamodels, response surfaces, surrogate models, auxiliary models, among others (Kleijnen, 2009). Metamodels have been extensively applied in engineering. To list some few examples, Fan *et al.* (2006) incorporated surrogate modeling to multi-objective optimization; Sultan (2007) applied surrogate modeling to replace an expensive iterative procedure used to prevent rotor-housing interference in a fluid processing machine; Zhao *et al.* (2008) also applied metamodel-based design optimization. More examples of metamodels in engineering will be provided in the following sections, when some of the available strategies are discussed in more detail.

A particular type of metamodeling approach, which has been in constant development over the last two decades, is *Gaussian process emulators* (GPEs). Based on the analysis and design of computer experiments (Sacks *et al.*, 1989; Santner *et al.*, 2003) and using concepts of Bayesian statistics, this approach makes it possible to obtain a statistical approximation to the output of a simulator after evaluating a small and carefully selected set of *design points* $\{\mathbf{x}_i\}_{i=1}^n$ in the input domain Ω, hence reducing the required computer processing time. Broadly speaking, emulation works by generating a small set of *training runs* $\{\mathbf{x}_i, \eta(\mathbf{x}_i)\}_{i=1}^n$ that are treated as data used to update some prior beliefs about the simulator's output. These beliefs are represented by a Gaussian stochastic process prior distribution. After conditioning on the training runs and updating the prior distribution, the mean of the resulting posterior distribution approximates the output of the simulator at any untried $\mathbf{x} \in \Omega$, whereas it reproduces the known output at each design point.

GPEs have already been implemented in a number of different scientific fields in order to alleviate the computational burden of expensive simulators with encouraging results. Kennedy *et al.* (2006) presented three case studies related to environmental computer models. They emulated a vegetation dynamic model, a model of ecosystem photosynthesis and water balance, and finally a model that estimates the UK's carbon budget. Challenor *et al.* (2006) emulated what they consider to be a moderately complex climate model. Rougier (2007) presented another application to a climate model. Bates *et al.* (2006) emulated a model of a complete revolution of a piston's shaft. Haylock and O'Hagan (1996) emulated a model of doses to organs of the body after ingestion of a radioactive substance. Oakley and O'Hagan (2004) worked with a simulator of the cost resulting from bone fractures for patients suffering from osteoporosis. Other disciplines in which GPEs have been employed include biomechanics (Kolachalama *et al.*, 2007), reservoir forecasting (Busby, 2009), hydrogeology (Marrel *et al.*, 2009), heat transfer (McFarland and Mahadevan, 2008), and reliability analysis (Daneshkhah and Bedford, 2008), among others.

Bearing in mind all the above research and considering that complex finite element models are prone to the use of computer intensive simulators, it is natural to think about the potential benefit from the application of GPEs in an engineering context, both in deterministic and non-deterministic settings. The motivation of this dissertation is therefore the integration of GPEs with established computational and mathematical techniques in engineering, particularly the finite element method and the stochastic finite element method. The latter with the aim of establishing GPEs as an effective predictive and uncertainty quantification tool for several problems faced by computational modeling in engineering. We now briefly discuss the basic principles and strategies within the finite element method and the stochastic finite element method.

1.2 The Finite Element Method

A system's response on a domain Ω can be described in terms of partial differential equations with associated initial and boundary conditions. The finite element method discretises these partial differential equations replacing the geometry of Ω by a set of nodes in a mesh, whose components are known as finite elements. The displacement field is approximated by the nodal displacement vector $\mathbf{u}(\mathbf{x})$. For a linear N-degree-of-freedom system, the deterministic finite element method eventually yields a system of the form

$$\mathbf{K}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \mathbf{f} \tag{1.1}$$

where $\mathbf{K}(\mathbf{x}) \in \mathbb{R}^{N\times N}$ is known as the stiffness matrix, $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^N$ is the response vector, and $\mathbf{f} \in \mathbb{R}^N$ is the forcing vector. The stiffness matrix $\mathbf{K}(\mathbf{x})$ is obtained by assembling the element stiffness matrices

$$\mathbf{k}_\epsilon = \int_{\Omega_\epsilon} \mathbf{B}^\intercal \cdot \mathbf{D} \cdot \mathbf{B} d\Omega_\epsilon \tag{1.2}$$

where $\Omega_\epsilon \subset \Omega$, $\mathbf{B}$ is the matrix relating strains to nodal displacements and $\mathbf{D}$ is the elasticity matrix. Note that sometimes, for notational convenience, the stiffness matrix will simply be denoted by $\mathbf{K}$.

The literature on the finite element method is vast and a complete study of its foundations is beyond the scope of this dissertation. However, the response of all the systems studied here will be based on implementations of the finite element method and its stochastic counterpart.

1.3 The Stochastic Finite Element Method

A realistic and reliable mathematical model of a complex engineering system must incorporate uncertainty. No matter how sophisticated the constitutive model or powerful the computational tools employed are, the intrinsic randomness of the material or the loads involved may be such that deterministic models deliver a highly inaccurate representation of reality. Engineering systems can be quite complex to analyze, and addressing such complexity requires the employment of numerical algorithms with a sound theoretical basis. Since the finite element method has been extensively tested in the context of deterministic engineering mechanics, one natural extension of this approach is to include random parameters in the partial differential equations governing the system's response. The stochastic finite element method sets the framework to model the physical properties of a given engineering system as random fields. When uncertainty is introduced into the system, the stiffness matrix $\mathbf{K}(\mathbf{x})$ in Eq. (1.1) becomes a random matrix. It thus becomes a function of the spatial coordinates and a random parameter, namely $\mathbf{K}(\mathbf{x}, \theta)$. Also, note that if the linear system in Eq. (1.1) is solved, a random stiffness matrix would require the response vector $\mathbf{u}(\mathbf{x})$ to be random. The new equilibrium equation therefore becomes

$$\mathbf{K}(\mathbf{x}, \theta)\mathbf{u}(\mathbf{x}, \theta) = \mathbf{f} \tag{1.3}$$

The stochastic finite element method is aimed at designing strategies to solve Eq. (1.3) and consequently quantify and propagate the uncertainty in the random system response $\mathbf{u}(\mathbf{x}, \theta)$. We now present a brief introduction to random fields, their discretisation and the available theoretical tools for uncertainty propagation.

1.3.1 Uncertainty Modeling

Let $(\Theta, \mathcal{F}, \mathcal{P})$ be a probability space, where Θ is a sample space, $\mathcal{F}$ is a σ-algebra, and $\mathcal{P} : \mathcal{F} \to [0,1]$ a probability measure. Let $\mathcal{L}^2(\Theta, \mathcal{F}, \mathcal{P})$ be the space of random variables with finite second moment. Let $\mathbf{E}[\cdot]$ denote the expectation operator. Thus, if $\mathcal{X}(\theta) : \Theta \to \mathcal{D}_{\mathcal{X}} \subset \mathbb{R}$ is a random variable, then

$$\mathbf{E}[\mathcal{X}^2(\theta)] = \int_{\Theta} \mathcal{X}^2(\theta) d\mathcal{P}(\theta) < +\infty \tag{1.4}$$

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

$$\mathbf{E}[\mathcal{X}_1(\theta)\mathcal{X}_2(\theta)] = \int_{\Theta} \mathcal{X}_1(\theta)\mathcal{X}_2(\theta) d\mathcal{P}(\theta) \tag{1.5}$$

A random field $\mathcal{H}(\mathbf{x}, \theta)$, with $\mathbf{x} \in \mathbb{R}^N$ and $\theta \in \Theta$, is a curve in $\mathcal{L}^2(\Theta, \mathcal{F}, P)$. For a given $\mathbf{x_0}$, $\mathcal{H}(\mathbf{x_0}, \theta)$ is a random variable; whereas for a given θ_0, $\mathcal{H}(\mathbf{x}, \theta_0)$ is a realization of the random field. Random fields are a useful tool to model distributed random mechanical properties of engineering systems, such as Poisson's ratio, Young's modulus or yield stress.

When implementing the deterministic finite element method, 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 stochastic finite element method, the random fields involved are discretised by representing them as a finite set of random variables. Therefore, $\mathcal{H}(\mathbf{x}, \theta)$ needs to be discretised in order to solve the associated system of random algebraic equations. There are several discretisation strategies available in the literature. Some of the most common are listed below.

Point Discretisation Methods

In this type of methods, a spatial discretisation of the finite element mesh is employed for the approximation of the system's response. Some of the available strategies are listed here.

- **Shape function method** (Liu *et al.*, 1986). This method approximates the random field $\mathcal{H}(\mathbf{x}, \theta)$ in each element using nodal values $\mathbf{x}_i \in \Omega$ and polynomial shape functions N_i as follows
$$\hat{\mathcal{H}}(\mathbf{x}, \theta) \simeq \sum_{i=1}^{q} N_i(\mathbf{x})\mathcal{H}(\mathbf{x}_i, \theta) \tag{1.6}$$
where Ω is an open set describing the geometry and q is the number of nodes.

- **Mid-point method** (Kiureghian and Ke, 1988). For each element $\Omega_\epsilon \subset \Omega$, this method approximates the random field by a random variable equal to the value of the random field at the centroid $\mathbf{x}_c$ of the element. Mathematically,
$$\hat{\mathcal{H}}(\mathbf{x}, \theta) \simeq \mathcal{H}(\mathbf{x}_c, \theta) \tag{1.7}$$
where $\mathbf{x} \in \Omega_\epsilon$.

- **Optimal linear estimation (OLE) method** (Li and Der Kiureghian, 1993). Under this scheme, the approximation to the random field is done with random variables dependent on nodal values $\chi = \{\mathcal{H}(\mathbf{x}_1, \theta), \ldots \mathcal{H}(\mathbf{x}_q, \theta)\}$. The dependence is linear, such that
$$\hat{\mathcal{H}}(\mathbf{x}, \theta) \simeq a(\mathbf{x}, \theta) + \mathbf{b}_i^T(\mathbf{x}, \theta)\chi_i \tag{1.8}$$
The coefficients are calculated by minimizing the variance of the difference between the approximate and the exact random field, namely $\mathbf{Var}[\mathcal{H}(\mathbf{x}, \theta) - \hat{\mathcal{H}}(\mathbf{x}, \theta)]$, while restricting the mean of that difference to be equal to zero, namely $\mathbf{E}[\mathcal{H}(\mathbf{x}, \theta) - \hat{\mathcal{H}}(\mathbf{x}, \theta)] = 0$.

- **Integration point method** (Matthies *et al.*, 1997). This method associates a single random variable to each Gauss point of each element appearing in the finite element resolution scheme, given that every integration appearing in the finite element model is obtained from integrand evaluation at each Gauss point of each element. Unfortunately, the total number of random variables involved increases dramatically with the size of the problem.

Average Discretisation Methods

In this type of methods, the random variables χ_i are represented by weighted integrals of the random field over a domain Ω

$$\chi_i = \int_{\Omega_e} \mathcal{H}(\mathbf{x},\theta) w(\mathbf{x}) d\Omega \tag{1.9}$$

Some of the available methods are described below.

- **Spatial average method** (Vanmarcke and Grigoriu, 1983). Given a finite element mesh, this method defines an approximation to the random field in each element as a constant equal to the average of the original random field over the element. Thus,

$$\hat{\mathcal{H}}(\mathbf{x},\theta) \simeq \frac{\int_{\Omega_\epsilon} \mathcal{H}((\mathbf{x},\theta))}{|\Omega_\epsilon|} \tag{1.10}$$

  where $\mathbf{x} \in \Omega_\epsilon$. A disadvantage of this method is that the variance of the spatial average over an element under-represents the local variance of the random field (Kiureghian and Ke, 1988).

- **Weighted integral method** (Deodatis, 1991; Deodatis and Shinozuka, 1991). This method considers the element stiffness matrices as random quantities. Each random variable is the result of integrating the product of one of the monomials used in the finite element method by the random field over each element. The original random field is projected onto the space spanned by the shape functions of the finite elements.

Series Expansion Methods

The idea behind this type of discretisation methods is to expand any realization of the original random field over a complete set of deterministic functions $\{\phi_i(\cdot)\}_{i=0}^{\infty}$ and truncate the series after a finite number of terms

$$\hat{\mathcal{H}}(\mathbf{x},\theta) \simeq \sum_{i=1}^{M} \chi_i(\theta)\phi_i(\mathbf{x}) \tag{1.11}$$

- **Orthogonal series expansion (OSE) method** (Zhang and Ellingwood, 1993). This approach is based on the selection of a set of deterministic orthogonal functions $\{h_i(x)\}_{i=1}^{\infty}$ which are a basis of the space $\mathcal{L}^2(\Omega)$. The coefficients of the expansion are zero mean random variables

$$\chi_i(\theta) = \int_\Omega [H(\mathbf{x},\theta) - \mu(\mathbf{x})]h_i(\mathbf{x})d\Omega \tag{1.12}$$

  where $\mu(\mathbf{x})$ is the mean of the random field. The approximation to $\mathcal{H}(\mathbf{x},\theta)$ is obtained after computing the covariance matrix $\mathbf{\Sigma}_{\chi\chi}$, which completely characterizes the vector $\chi = \{\chi_1, \dots, \chi_M\}$. The approximation is of the form

$$H(\mathbf{x},\theta) = \mu(\mathbf{x}) + \sum_{i=1}^{\infty} \chi_i(\theta)h_i(\mathbf{x}) \tag{1.13}$$

- **Expansion optimal linear estimation method** (Li and Der Kiureghian, 1993). This method is an extension of OSE. It employs a spectral representation of χ, namely

$$\chi(\theta) = \boldsymbol{\mu}_\chi + \sum_{i=1}^{N} \sqrt{\lambda_i}\xi_i(\theta)\boldsymbol{\phi}_i \tag{1.14}$$

  where ξ_i are independent standard normal variables and $(\lambda_i, \boldsymbol{\phi}_i)$ are the eigenvalues and eigenvectors of the covariance matrix $\mathbf{\Sigma}_{\chi\chi}$.

- **Karhunen-Loève expansion method** (Karhunen, 1946; Loève, 1948).

  An advantageous alternative for discretising $\mathcal{H}(\mathbf{x},\theta)$ is the Karhunen-Loève expansion (KLE), for which

$$\mathcal{H}(\mathbf{x},\theta) = \sum_{i=0}^{\infty} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) \tag{1.15}$$

  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 eigenvalues and eigenfunctions. They arise from the solution of the following eigenvalue problem

$$\int_{\mathbb{R}^N} K(\mathbf{x}_1,\mathbf{x}_2)\phi_i(\mathbf{x}_1)d\mathbf{x}_1 = \lambda_i\phi_i(\mathbf{x}_2) \tag{1.16}$$

  where the function $K(\cdot,\cdot)$ is a covariance kernel, that is, a function which is bounded, symmetric and positive definite. Equation (1.16) is known as a Fredholm integral equation.

The truncated KLE of $\mathcal{H}(\mathbf{x},\theta)$ up to M terms is defined as (see for example Sudret and Der-Kiureghian (2000))

$$\mathcal{H}(\mathbf{x},\theta) = \mu(\mathbf{x}) + \sum_{i=1}^{M} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) \tag{1.17}$$

The KLE has uniqueness and error-minimization properties that make it a convenient choice over other available methods. See Devijver and Kittler (1982) for a detailed study of the cited and other KLE properties.

As an example, consider a random field $\mathcal{H}(\mathbf{x},\theta)$ with *exponential correlation function*

$$K(\mathbf{x}_1,\mathbf{x}_2) = e^{\|\mathbf{x}_1-\mathbf{x}_2\|/b} \tag{1.18}$$

where b is the correlation length. The eigenfunctions $\{\phi_i(\mathbf{x})\}_{i=0}^{\infty}$ and eigenvalues $\{\lambda_i\}_{i=0}^{\infty}$ are the solutions to equation

$$\int_{-a}^{+a} e^{|\mathbf{x}_1-\mathbf{x}_2|/b}\phi(\mathbf{x}_2)dx_2 = \lambda\phi(\mathbf{x}_1) \tag{1.19}$$

The explicit expressions of the eigenfunctions are

$$\phi_i(\mathbf{x}) = \frac{\cos(\omega_i\mathbf{x})}{\sqrt{a+\frac{\sin(2\omega_i a)}{2\omega_i}}} \quad \text{for } i \text{ even} \tag{1.20}$$

$$\phi_i^*(\mathbf{x}) = \frac{\sin(\omega_i^*\mathbf{x})}{\sqrt{a-\frac{sin(2\omega_i^* a)}{2\omega_i^*}}} \quad \text{for } i \text{ odd} \tag{1.21}$$

The corresponding eigenvalues are

$$\lambda_i = \frac{2c}{\omega_i^2+c^2} \quad \text{for } i \text{ even} \tag{1.22}$$

$$\lambda_i^* = \frac{2c}{\omega_i^{*2}+c^2} \quad \text{for } i \text{ odd} \tag{1.23}$$

where $c = 1/b$. ω_i, and ω^* are the solutions to

$$c - \omega\tan(\omega a) = 0 \quad \text{for } i \text{ even} \tag{1.24}$$

$$\omega^* + c\tan(\omega^* a) = 0 \quad \text{for } i \text{ odd} \tag{1.25}$$

The KLE of $\mathcal{H}(\mathbf{x},\theta)$ is therefore

$$\mathcal{H}(\mathbf{x},\theta) = \sum_{i=0}^{\infty} \left[\sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) + \sqrt{\lambda_i^*}\xi_i^*(\theta)\phi_i^*(\mathbf{x})\right] \tag{1.26}$$

1.3.2 Uncertainty Propagation

Several methods to solve Eq. (1.3) and consequently calculate the statistics of the response vector $\mathbf{u}(\mathbf{x},\theta)$ are available in the literature. Some of these methods are briefly discussed below. They include Monte Carlo simulation techniques (Hurtado and Barbat, 1998; Papadrakakis and Papadopoulos, 1996), perturbation methods through Taylor series (Shinozuka and Yamazaki, 1998; Kleiber and Hien, 1992; Elishakoff *et al.*, 1995), expansion methods through Neumann series (Yamazaki *et al.*, 1988; Shinozuka and Nomoto, 1980) and the spectral stochastic finite element method (Ghanem and Spanos, 1991). Note that in addition to these traditional methods, there exist a vast amount of literature in alternative analytical methods (Muscolino *et al.*, 2000; Impollonia and Muscolino, 2002; Falsone and Impollonia, 2002, 2003; Impollonia and Ricciardi, 2006) that will not be reviewed here.

Monte Carlo Simulation

Let $\{\theta_i\}_{i=1}^{S}$ be a series of realizations of a random parameter $\theta \in \Theta$ for a given engineering system. For every θ_i, the vector $\mathbf{u}_i = \mathbf{u}(\mathbf{x},\theta_i)$ is a solution to Eq. (1.3). Each solution expresses the system response given a specific value of θ. As a consequence, each $\mathbf{u}_i$ can be obtained by solving a deterministic finite element model.

In the probabilistic analysis of the response $\mathbf{u}(\mathbf{x},\theta)$, quantities of interest can be written as an expectation of a function φ of $\mathbf{u}(\mathbf{x},\theta)$ of the form

$$\mathbf{E}[\varphi(\mathbf{u}(\mathbf{x},\theta))] = \int_{\Theta} \varphi(\mathbf{u}(\mathbf{x},\theta))p_\theta(\theta)d\theta \tag{1.27}$$

where $p_\theta(\theta)$ denotes the probability function of θ. In practice, given a sample $\{\theta_i\}_{i=1}^{S}$, the expectation in Eq. (1.27) can be estimated by

$$\hat{\mathbf{E}}[\varphi(\mathbf{u}(\mathbf{x},\theta))] = \frac{1}{S}\sum_{i=1}^{S} \varphi(\mathbf{u}(\mathbf{x},\theta_i)) \tag{1.28}$$

The standard deviation of the estimator $\hat{\mathbf{E}}[\cdot]$ equals $S^{-1/2}\sigma_\varphi$, where σ_φ is the standard deviation of $\varphi(\cdot)$ (Nouy, 2009). The convergence of this estimator, in $O(S^{-1/2})$ is independent of the dimension of the data. Apart from its straightforward implementation, the independence of the dimension is probably the main advantage of the Monte Carlo method.

In particular, the mean and variance of the response can be approximated by

$$\mathbf{E}[\mathbf{u}_i] \simeq \frac{1}{S}\sum_{i=1}^{S}\mathbf{u}(\mathbf{x},\theta_i) \quad (1.29)$$

$$\mathbf{Var}[\mathbf{u}_i] \simeq \frac{1}{S-1}\sum_{i=1}^{S}(\mathbf{u}(\mathbf{x},\theta_i)-\mathbf{E}[\mathbf{u}_i])^2 \quad (1.30)$$

The obvious disadvantage of this approach is that its slow convergence might require an enormous number of samples to achieve a required accuracy. Several alternatives that increase the efficiency of the approach are available in the literature, such as importance sampling (Schuller *et al.*, 2004), adaptive sampling (Au and Beck, 1999), directed Monte Carlo simulation (Feng *et al.*, 2010), among many others.

Perturbation Method

The perturbation method works by expanding the stiffness matrix **K**, the response **u** and load vector **f** in Eq. (1.3) as Taylor series around their mean value. Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ be a zero mean random vector. The series expansions are

$$\mathbf{K} = \mathbf{K}_0 + \sum_{i=1}^{N}\mathbf{K}_i^I\alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{K}_{ij}^{II}\alpha_i\alpha_j + O(\|\boldsymbol{\alpha}\|^2) \quad (1.31)$$

$$\mathbf{u} = \mathbf{u}_0 + \sum_{i=1}^{N}\mathbf{u}_i^I\alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{u}_{ij}^{II}\alpha_i\alpha_j + O(\|\boldsymbol{\alpha}\|^2) \quad (1.32)$$

$$\mathbf{f} = \mathbf{f}_0 + \sum_{i=1}^{N}\mathbf{f}_i^I\alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{f}_{ij}^{II}\alpha_i\alpha_j + O(\|\boldsymbol{\alpha}\|^2) \quad (1.33)$$

where

$$\mathbf{K}_i^I = \left.\frac{\partial \mathbf{K}}{\partial \alpha_i}\right|_{\alpha=0} \quad (1.34)$$

$$\mathbf{K}_{ij}^{II} = \left.\frac{\partial^2 \mathbf{K}}{\partial \alpha_i \partial \alpha_j}\right|_{\alpha=0} \quad (1.35)$$

After substituting Eq. (1.31) and Eq. (1.32) into Eq. (1.33) and identifying the similar order coefficients, the following expressions are obtained

$$\mathbf{u}_0 = \mathbf{K}_0^{-1}.\mathrm{f}_0 \tag{1.36}$$

$$\mathbf{u}_i^I = \mathbf{K}_0^{-1}.(\mathbf{f}_i^I - \mathbf{K}_i^I.\mathbf{u}_0) \tag{1.37}$$

$$\mathbf{u}_{ij}^{II} = \mathbf{K}_0^{-1}.(\mathbf{f}_{ij}^{II} - \mathbf{K}_i^I.\mathbf{u}_j^I - \mathbf{K}_j^I.\mathbf{u}_i^I - \mathbf{K}_{ij}^{II}.\mathbf{u}_0) \tag{1.38}$$

The statistics of $\mathbf{u}$ are available from those of $\boldsymbol{\alpha}$ in equation (1.32), namely

$$\mathbf{E}[\mathbf{u}] \approx \mathbf{u}_0 + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{u}_{ij}^{II}\mathrm{Cov}[\alpha_i, \alpha_j] \tag{1.39}$$

$$\mathbf{Cov}[\mathbf{u},\mathbf{u}] \approx \sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{u}_i^I.(\mathbf{u}_j^I)^{\mathsf{T}}\mathrm{Cov}[\alpha_i, \alpha_j] \tag{1.40}$$

Neumann Expansion Method

The KLE of the random stiffness matrix in Eq. (1.3) can be shown (Sudret and Der-Kiureghian, 2000) to be

$$\mathbf{K}(\mathbf{x},\theta) = \sum_{i=0}^{\infty}\mathbf{K}_i(\mathbf{x})\xi_i(\theta) \tag{1.41}$$

where each $\mathbf{K}_i(\mathbf{x})$ is a deterministic matrix obtained by assembling the elementary stiffness matrix in an analogous way to the deterministic finite element method. Thus, Eq. (1.3) can be recast as

$$\left(\mathbf{K}_0 + \sum_{i=1}^{\infty}\mathbf{K}_i(\mathbf{x})\xi_i(\theta)\right)\mathbf{u}(\mathbf{x},\theta) = \mathbf{f} \tag{1.42}$$

Analytically, the vector of nodal displacements would be obtained by inverting the expanded matrix $\mathbf{K}(\mathbf{x},\theta)$. However, no closed form solution exists. An alternative solution is to rewrite Eq. (1.42) as

$$\mathbf{K}_0\left(\mathbf{I} + \sum_{i=1}^{\infty}\mathbf{K}_0^{-1}\mathbf{K}_i(\mathbf{x})\xi_i(\theta)\right)\mathbf{u}(\mathbf{x},\theta) = \mathbf{f} \tag{1.43}$$

Thus, if $\mathbf{u}_0 \equiv \mathbf{K}_0^{-1}\mathbf{f}$, then

$$\mathbf{u}(\mathbf{x},\theta) = \left(\mathbf{I} + \sum_{i=1}^{\infty}\mathbf{K}_0^{-1}\mathbf{K}_i(\mathbf{x})\xi_i(\theta)\right)^{-1}\mathbf{u}_0 \tag{1.44}$$

The *Neumann series expansion* is

$$\mathbf{u}(\mathbf{x},\theta) = \sum_{k=0}^{\infty}(-1)^k \left(\sum_{i=1}^{\infty} \mathbf{K}_0^{-1}\mathbf{K}_i(\mathbf{x})\xi_i(\theta)\right)^k \mathbf{u}_0 \tag{1.45}$$

Therefore, an approximate displacement vector $\mathbf{u}(\mathbf{x},\theta)$ can be obtained by truncating both the KLE of $\mathbf{K}(\mathbf{x},\theta)$ and the Neumann series expansion in Eq. (1.45).

Spectral Stochastic Finite Element Method

All the uncertainty propagation methods presented so far have a sound mathematical basis. However, despite this theoretical appeal, their implementation presents at least one of the following disadvantages: a) lack of the geometrical appeal presented by the deterministic finite element method; b) limited applicability due to restrictive analytical constraints; c) non-guaranteed convergence of the Taylor and Neumann series involved; d) potentially high computational cost. Additionally, since the covariance structure of the response random field is generally unknown, the KLE cannot be employed to represent the random displacement vector $\mathbf{u}(\mathbf{x},\theta)$.

Aiming to tackle some of the above mentioned disadvantages, Ghanem and Spanos (1991) applied and extended some of the ideas by Wiener (1938), namely the *polynomial chaos expansion method*, (also known simply as *polynomial chaos*). Later, this approach was further extended by several authors, for example Nair and Keane (2002); Sachdeva *et al.* (2006a,b); Xiu and Karniadakis (2002, 2003); Wan and Karniadakis (2006). This method essentially consists 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 $\{\xi_i(\theta)\}_{i=1}^{\infty}$. The polynomial chaos of order p is defined as the set of polynomials Γ_p in $\{\xi_i(\theta)\}_{i=1}^{\infty}$ of degree not exceeding p, orthogonal to the set of polynomials Γ_{p-1}. Ghanem and Spanos (1991) mention that any square-integrable random function $\mathbf{a}(\theta) \in \mathcal{L}^2(\Theta,\mathcal{F},P)$ can be approximated as closely as desired with the representation:

$$\begin{aligned}\mathbf{a}(\theta) &= a_0\Gamma_0 + \sum_{i_1=1}^{\infty} a_{i_1}\Gamma_1(\xi_{i_1}(\theta)) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2}\Gamma_2(\xi_{i_1}(\theta),\xi_{i_2}(\theta)) \\ &\quad + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2}\sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3}\Gamma_3(\xi_{\rho_1}(\theta),\xi_{\rho_2}(\theta),\xi_{\rho_3}(\theta)) + \ldots \\ &= \sum_{j=0}^{\infty} \hat{a}_j \boldsymbol{\Psi}_j[\boldsymbol{\xi}(\theta)]\end{aligned} \tag{1.46}$$

The vector of random displacements can be thus shown to be expressed as

$$\mathbf{u}(\mathbf{x},\theta) = \sum_{j=0}^{\infty} \mathcal{U}_j\,(\mathbf{x})\Psi_j(\theta) \tag{1.47}$$

where each $\mathcal{U}_j(\mathbf{x})$ is a deterministic vector. If the random stiffness matrix $\mathbf{K}(\mathbf{x},\theta)$ from Eq. (1.3) is expanded via the KLE, the new equilibrium equation that incorporates the polynomial chaos expansion of $\mathbf{u}(\mathbf{x},\theta)$ reads

$$\left(\sum_{i=0}^{\infty} \mathbf{K}_i(\mathbf{x})\xi_i(\theta)\right)\left(\sum_{j=0}^{\infty} \mathcal{U}_j(\mathbf{x})\Psi_j(\theta)\right) = \mathbf{f} \tag{1.48}$$

After truncation of both expansions, the residual is

$$\mathcal{R}(\mathbf{x},\theta) = \sum_{i=0}^{M}\sum_{j=0}^{P-1} \mathbf{K}_i(\mathbf{x})\mathcal{U}_j(\mathbf{x})\xi_i(\theta)\Psi_j(\theta) - \mathbf{f} \tag{1.49}$$

This residual can be minimized in a mean square sense, such that $\mathbf{E}[\mathcal{R}(\mathbf{x},\theta)\cdot\Psi_k] = 0$ for $k = 0,\dots P-1$. If we define $\mathbf{K}_{jk} = \sum_{i=0}^{M}\sum_{j=0}^{P-1} \mathbf{E}[\xi_i\Psi_j\Psi_k]\mathbf{K}_i$ and $\mathbf{f}_k = \mathbf{E}[\Psi_k\mathbf{f}]$, then the error minimizing procedure leads to the linear system

$$\begin{bmatrix} \mathbf{K}_{00} & \dots & \mathbf{K}_{0,P-1} \\ \mathbf{K}_{10} & \dots & \mathbf{K}_{1,P-1} \\ \vdots & & \vdots \\ \mathbf{K}_{P-1,0} & \dots & \mathbf{K}_{P-1,P-1} \end{bmatrix} \cdot \begin{bmatrix} \mathcal{U}_0 \\ \mathcal{U}_1 \\ \vdots \\ \mathcal{U}_{P-1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_{P-1} \end{bmatrix} \tag{1.50}$$

Which is an $NP \times NP$ system of the form

$$\mathcal{K}\cdot\mathcal{U} = \mathcal{F} \tag{1.51}$$

The polynomial chaos expansion is widely used in engineering. Its main disadvantage however is its potential cost of implementation. The method will be revisited in Chapter 6, when a strategy to reduce this cost incorporating GPEs is proposed.

1.4 Some Metamodels in Engineering

The following is a very brief overview of some of the metamodels that have been used in engineering. This overview is by no means exhaustive, and metamodeling is currently an active research area. An account of the following and some other metamodels can be found in Keane and Nair (2005).

1.4.1 Taylor Series Approximations

One of the simplest metamodels that can be employed are local approximations by a Taylor series. That is, in order to approximate the output of the simulator $\eta(\mathbf{x})$ in a neighborhood of $\mathbf{x}^*$, say $\|\mathbf{x}^* - \mathbf{x}\| \leq \epsilon$, one would typically use a metamodel of the form

$$\eta(\mathbf{x}) = f(\mathbf{x}^*) + \sum_{i=1}^{p}(x_i - x_i^*)\frac{\partial f(\mathbf{x}^*)}{\partial x_i} + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}(x_i - x_i^*)(x_j - x_j^*)\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_i \partial x_j} + \dots \qquad (1.52)$$

Despite the simplicity of the implementation, this metamodel presents two disadvantages. On the one hand, the radius of convergence ϵ might be small and therefore the approximation very poor (Storaasli and Sobieszczanski-Sobieski, 1974). On the other hand, the higher order terms may be expensive to compute.

1.4.2 Response Surface Methods

These metamodels are related to least-square regression techniques, in the sense that they assume independent and identically distributed measurement errors of computer experiments. A (quadratic) polynomial approximation to the output of a simulator $\eta(\cdot)$ at $\mathbf{x}$ is achieved by

$$\eta(\mathbf{x}) = c_0 + \sum_{1 \leq j \leq p} c_j x_j + \sum_{1 \leq j \leq p, k > j} c_{p-1+j+j} x_j x_k \qquad (1.53)$$

where $\{c_0, c_1, \dots, c_{m-1}\}$ are coefficients to be determined. Response surface methods have been employed extensively in engineering. For instance, Craig *et al.* (2005) performed variable screening and optimization in crashworthiness design based on a response surface methodology. Pérez *et al.* (2008) solved nonlinear optimization problems using quadratic response surfaces. Faravelli (1989) and Schuller *et al.* (1989) applied the methodology to reliability analysis. Despite being an intuitive and interpretable approach, response surface methods may lack effectiveness when modeling complex input/output relationships. Additionally, the radius or neighbourhood where the approximation is acceptable may also be limited.

1.4.3 Neural Networks

Neural networks are inspired in idealized models of brain structure whereby multiple neurons are connected following a configuration whose simplest form consists of an input layer, a hidden layer and an output layer. This configuration, known as feed-forward system, is mathematically represented by

$$\eta(\mathbf{x}) = \sum_{i=1}^{m} \lambda_i \phi(a_i) \tag{1.54}$$

where

$$a_i = \sum_{j=1}^{p} w_{ij} x_j + \delta_j \tag{1.55}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ are the weight parameters; $\boldsymbol{\delta}$ is a bias parameter; and $\phi(\cdot)$ is a transfer function. The number of parameters is determined by m neurons in the hidden layer and p inputs. Neural networks have been applied in engineering in areas such as damage detection (Pierce *et al.*, 2006) and fatigue lifetime prediction (Bezazi *et al.*, 2007; Pierce *et al.*, 2008). There is a copious amount of literature on neural networks. A study of their properties, different configurations, and the estimation of their parameters can be found in Bishop (1996).

1.4.4 Radial Basis Functions

A radial basis function is a function whose value depends on the distance to some center $\mathbf{x}_c$, that is $K(\mathbf{x}, \mathbf{x}_c) = K(||\mathbf{x} - \mathbf{x}_c||)$. In particular, if the design points $\{\mathbf{x}_i\}_{i=1}^n$ are selected to run a deterministic computer model, then the radial basis approximation is defined as

$$\eta(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(||\mathbf{x} - \mathbf{x}_i||) \tag{1.56}$$

where $\{\alpha_i\}_{i=1}^n$ are weights to be determined. Typical choices of radial basis functions are

Name	RBF
Linear spline	$\|\|\mathbf{x} - \mathbf{x}_c\|\|$
Cubic spline	$\|\|\mathbf{x} - \mathbf{x}_c\|\|^3$
Gaussian	$\exp(-\frac{\|\|\mathbf{x} - \mathbf{x}_c\|\|^2}{\theta})$
Multiquadrics	$(1 + \frac{\|\|\mathbf{x} - \mathbf{x}_c\|\|^2}{\theta})^{-1/2}$

Table 1.1: Examples of radial basis functions $K(\mathbf{x}, \mathbf{x}_c) = K(||\mathbf{x} - \mathbf{x}_c||)$, where $\mathbf{x}_c$ is called the center. The parameter θ is a smoothness parameter.

Radial basis functions have also been applied to solve engineering problems. For example, Rocha (2009) applied it to the construction of a wing weight estimation formula for the conceptual design of subsonic transports.

1.4.5 HDMR

The high dimensional model representation (HDMR) method approximates the response of a system in terms of functions of lower dimensions. The approximation can be denoted by

$$\eta(\mathbf{x}) = f_0 + \sum_i f_i(x_i) + \sum_{i<j} f_{ij}(x_i, x_j) + \ldots + f_{12\ldots N}(x_1, x_2, \ldots, x_N) \tag{1.57}$$

where f_0 is a constant, $f_i(x_i)$ is the first-order effect associated with the variable x_i upon the output, $f_{ij}(x_i, x_j)$ is the cooperative effects of variables x_i and x_j and analogously for the higher order terms. Finally, $f_{12\ldots N}(x_1, x_2, \ldots, x_N)$ is the residual dependence of all the variables affecting the output cooperatively.

HDMR efficiently approximates the system's response by representing the physical system hierarchically. Note however that the approximation is reliable if high-order variable correlations are weak, whereby the features of the system can be captured by the first low-order terms. The approach has been applied in areas such as reliability analysis (Rao and Chowdhury, 2009; Chowdhury and Rao, 2009), and stochastic finite element analysis (Chowdhury and Adhikari, 2010).

1.4.6 Kriging

A metamodeling approach closely related to Gaussian process emulation is kriging (Krige, 1951). This method was originally applied in geostatistics (see for example Cressie (1993)) and made its way into the design and analysis of computer experiments following the work of Sacks *et al.* (1989). Although kriging and Gaussian process emulator have similar computational procedures, authors like Pepelyshev and Oakley (2009) have noted that they have different motivations. For a more detailed study of kriging, refer to Forrester *et al.* (2008).

1.5 Areas of Opportunity

The above discussion shows that several metamodeling strategies have been widely used in engineering problems. However, GPEs have not been applied in this context, particularly when they involve the finite element method and the stochastic finite element method. This opens up the opportunity of investigating the possibility of employing GPEs as an efficient metamodeling tool in engineering. We have identified the following areas of opportunity:

1. The introduction of GPEs as an efficient and effective metamodeling tool to the engineering community.

2. The establishment of GPEs as an inexpensive approximation to the output of computer intensive simulators used in engineering, both in deterministic and non-deterministic settings. Such approximation should be efficient by minimizing the number of evaluations of the expensive simulator and thus reducing the computational cost involved.

3. The exploration of GPEs as an alternative metamodeling scheme for deterministic engineering problems that are expensive to solve computationally, such as damped structural dynamics and domain decomposition methods.

4. The proposal of new strategies for approximating the output of engineering models where parametric uncertainty needs to be taken into account in order to realistically model complex phenomena.

5. The exploration of GPEs as a new uncertainty quantification tool whose accuracy is comparable to Monte Carlo simulation, but much cheaper in terms of its implementation.

6. The efficient metamodeling of expensive random fields discretised via the Karhunen-Loève expansion.

7. The coupling of GPEs with established schemes of stochastic finite element analysis. Specifically, to achieve a coupling with the polynomial chaos expansion, which despite being a widespread technique, its application is limited by its computational cost.

1.6 Layout of the Dissertation

The dissertation is organized as follows. In Chapter 2, theoretical and practical aspects of the implementation of GPEs are introduced. Chapter 3 begins our study of GPEs for engineering models by applying them in the context of deterministic structural dynamic analysis. Chapter 4, proposes a novel methodology where GPEs are implemented jointly with domain decomposition methods in order to assimilate low-fidelity finite element models into high-fidelity models. In Chapter 5, the application of GPEs to structural dynamic analysis is revisited, assuming parametric uncertainty. Chapter 6 proposes a coupling between GPEs and the polynomial chaos expansion. Finally, Chapter 7 offers some conclusions and suggests some future research directions.

Chapter 2

Overview of Gaussian Process Emulators

2.1 Basic Definitions

Let $\eta(\cdot)$ be an expensive simulator, such that it is practical to evaluate it only at a limited number of inputs. This allows $\eta(\cdot)$ to be regarded as a random variable in the sense that the output is unknown until the simulator is actually run. A Bayesian treatment is followed throughout this dissertation, whereby prior beliefs about the relationship between the input and the unknown output are conditioned on a set of evaluations of $\eta(\cdot)$, thus combining subjective and objective information. Begin by assuming that $\eta(\cdot)$ admits the following stochastic representation

$$\eta(\mathbf{x}) = \mathbf{h}(\mathbf{x})^{\intercal}\boldsymbol{\beta} + Z(\mathbf{x}) \tag{2.1}$$

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 $\mathbf{x}$. An advantageous choice for $Z(\cdot)$ is the *Gaussian stochastic process.*

Definition. Let $\mathcal{X} \subseteq \mathbb{R}^d$. Then $Z(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X}$ is a **Gaussian stochastic process** if for any $L \geq 1$ and any choice $\{\mathbf{x}_1, \ldots, \mathbf{x}_L\} \subseteq \mathcal{X}$, the vector $[Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_L)]^{\intercal}$ has a multivariate normal distribution.

As noted by Kennedy and O'Hagan (2001), the choice of a Gaussian process is made for much the same reasons that the Gaussian distribution repeatedly appears in statistics: it is analytically tractable, flexible, and quite often realistic. Despite these advantages, they are careful to refer to alternative ways of expressing prior beliefs available in the literature. One such approach is to represent $\eta(\cdot)$ as a linear combination of basis functions such as splines, wavelets, and sigmoidal functions.

Suppose a linear structure of the form $\mathbf{h}(\cdot)^{\mathsf{T}}\boldsymbol{\beta}$, where $\mathbf{h}(\cdot)$ is a vector of regression functions and $\boldsymbol{\beta}$ is a vector of coefficients, is chosen to model the prior mean of $\eta(\cdot)$. Then, the interpretation of Eq. (2.1) becomes clearer. That is, $\eta(\cdot)$ is assumed to deviate from the mean of its distribution following a Gaussian stochastic process. Oakley and O'Hagan (2004) note that the choice of $\mathbf{h}(\cdot)$ is arbitrary, although it should be chosen to reflect the available information about the functional form of $\eta(\cdot)$. Authors such as Keane and Nair (2005) note that, for a sufficiently flexible correlation structure, $\mathbf{h}(\cdot) = 1$ is often found to be suitable for modeling highly complex input-output relationships. This is currently an area for further investigation.

An important assumption is to regard $\eta(\cdot)$ as a smooth function of its inputs. It follows that if $\mathbf{x}$ and $\mathbf{x}'$ are close together, then the values of $\eta(\mathbf{x})$ and $\eta(\mathbf{x}')$ 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 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 is reduced as the number of design points increases because the maximum distance from any design point decreases. The discussion on how to determine suitable covariance functions can become very technical and further details can be consulted in Santner *et al.* (2003). A popular choice for covariance function is the one that is adopted hereafter, namely

$$Cov(\eta(\mathbf{x}), \eta(\mathbf{x}')) = \sigma^2 C(\cdot, \cdot) \tag{2.2}$$

with the correlation function $C(\cdot, \cdot)$ such that

$$C(\cdot, \cdot) = e^{-(\mathbf{x}-\mathbf{x}')^{\mathsf{T}}\mathbf{B}(\mathbf{x}-\mathbf{x}')} \tag{2.3}$$

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

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 mean $\mathbf{h}(\cdot)^{\mathsf{T}}\boldsymbol{\beta}$ and covariance expressed by Eq. (2.2). The latter is symbolized by

$$\eta(\cdot)|\boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\mathbf{h}(\cdot)^{\mathsf{T}}\boldsymbol{\beta}, \sigma^2 C(\cdot, \cdot)) \tag{2.4}$$

The subjective information about the input and the unknown outputs is contained in this prior distribution. The next step is to update it by adding the objective information contained in a vector of observations, denoted here by $\mathbf{y} = [\eta(\mathbf{x}_1), \dots, \eta(\mathbf{x}_n)]^\intercal$.

Let $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_n)]^\intercal$, and $\mathbf{A} \in \mathbb{R}^{n\times n}$ such that $\mathbf{A}_{ij} = C(\mathbf{x}_i, \mathbf{x}_j) \ \ \forall i, j \in \{1, \dots, n\}$. Then

$$\mathbf{y}|\boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(\mathbf{H}\boldsymbol{\beta}, \sigma^2\mathbf{A}) \tag{2.5}$$

To incorporate the information $\mathbf{y}$ and obtain the distribution of $\eta(\cdot)|\mathbf{y}$, use the following result (Krzanowski, 2000).

Theorem. 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 $(\mathbf{z}_1, \mathbf{z}_2)^\intercal$, where $\mathbf{z}_1 \in \mathbb{R}^{N-n}$ and $\mathbf{z}_2 \in \mathbb{R}^n$. Consequently, partition $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)^\intercal$ 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 $\widetilde{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$.

It follows that

$$\eta(\cdot)|\mathbf{y}, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}(m^*(\cdot), \sigma^2 C^*(\cdot, \cdot)) \tag{2.6}$$

where

$$m^*(x) = \mathbf{h}(x)^\intercal\boldsymbol{\beta} + \mathbf{t}(x)\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\boldsymbol{\beta}) \tag{2.7}$$

$$C^*(\mathbf{x}, \mathbf{x}') = C(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^\intercal\mathbf{A}^{-1}\mathbf{t}(\mathbf{x}') \tag{2.8}$$

$$\mathbf{t}(\mathbf{x}) = [C(\mathbf{x}, \mathbf{x}_1), \dots, C(\mathbf{x}, \mathbf{x}_n)]^\intercal \tag{2.9}$$

Removing the conditioning on $\boldsymbol{\beta}$ using standard integration techniques (Haylock and O'Hagan, 1996), obtain the posterior distribution

$$\eta(\cdot)|\mathbf{y}, \sigma^2 \sim \mathcal{N}(m^{**}(\cdot), \sigma^2 C^{**}(\cdot, \cdot)) \tag{2.10}$$

where

$$m^{**}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^\intercal\widehat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})\mathbf{A}^{-1}(\mathbf{y} - \mathbf{H}\widehat{\boldsymbol{\beta}}) \tag{2.11}$$

$$C^{**}(\mathbf{x}, \mathbf{x}') = C^*(\mathbf{x}, \mathbf{x}') + (\mathbf{h}(\mathbf{x})^\intercal - \mathbf{t}(\mathbf{x})^\intercal\mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^\intercal\mathbf{A}^{-1}\mathbf{H})^{-1}(\mathbf{h}(\mathbf{x}')^\intercal - \mathbf{t}(\mathbf{x}')^\intercal\mathbf{A}^{-1}\mathbf{H})^\intercal \tag{2.12}$$

$$\widehat{\boldsymbol{\beta}} = (\mathbf{H}^\intercal\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^\intercal\mathbf{A}^{-1}\mathbf{y} \tag{2.13}$$

To estimate σ in Eq. (2.10), let q be the rank of $\mathbf{H}$. Then

$$\widehat{\sigma}^2 = \frac{\mathbf{y}^\mathsf{T}(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}(\mathbf{H}^\mathsf{T}\mathbf{A}^{-1}\mathbf{H})\mathbf{H}^\mathsf{T}\mathbf{A}^{-1})\mathbf{y}}{n-q-2} \tag{2.14}$$

The conditioning on σ^2 can also be eliminated, such that

$$\frac{\eta(\mathbf{x}) - m^{**}(\mathbf{x})}{\widehat{\sigma}\sqrt{\frac{(n-q-2)C^{**}(\mathbf{x})}{n-q}}} \sim t_{n-q} \tag{2.15}$$

which is a Student's t-distribution with $n-q$ degrees of freedom (not to be confused with the degrees of freedom in a finite element method context).

As it can be seen, Gaussian process emulation consists in updating the prior distribution (2.4), which contains subjective information, by adding the objective information $\mathbf{y}$ in order to obtain the posterior distribution (2.10). This enables the calculation of the predictive mean $m^{**}(\cdot)$ given the data $\mathbf{y}$. This mean is a fast approximation of $\eta(\mathbf{x})$ for any $\mathbf{x}$ in the domain of $\eta(\cdot)$. The complete process is summarized in Algorithm 1.

Alg. 1 Gaussian process emulation.

```
Input: Design points ${\mathbf{x}_i}_{i=1}^n$
Output: Predictive mean $\mathbf{E}[\eta(\mathbf{x})|\mathbf{y}] = m^{**}(\mathbf{x})$ and variance $\mathbf{Var}[\eta(\mathbf{x})|\mathbf{y}] = \sigma^2 C^{**}(\cdot,\cdot)$

    begin
        1. Select $n$ design points ${\mathbf{x}_i}_{i=1}^n$
        2. Obtain the vector of observations $\mathbf{y} = [\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)]^\mathsf{T}$
        3. Update the prior distribution (2.4) using $\mathbf{y}$ and obtain the posterior
             distribution (2.10)
        4. Compute the predictive mean $m^{**}(\mathbf{x})$ and variance $\sigma^2 C^{**}(\cdot,\cdot)$ for any untried $\mathbf{x}$
    end
```

Following O'Hagan (2006), a GPE 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 training data. The probability distribution around this predictive mean should also express the uncertainty about how the emulator might interpolate/extrapolate.

In addition to these criteria, Rougier *et al.* (2007) note that the key feature of an emulator is that it quantifies the uncertainty that arises from having a training set with limited size. Naturally, it is also desirable that emulation is at least as efficient as other available techniques, if it is to be worthy of implementation.

2.2 Introductory Example

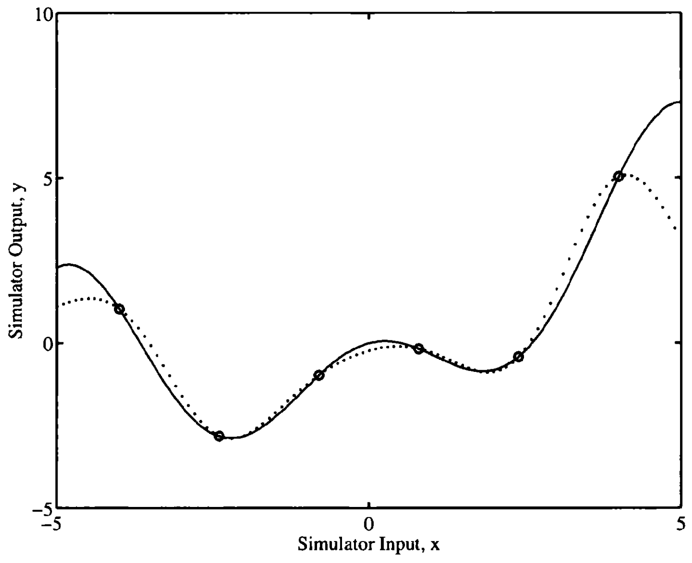
To illustrate the above criteria, take the trivial one-dimensional simulator $\eta(\mathbf{x}) = 0.5\mathbf{x} - \mathbf{x}\sin(\mathbf{x})$ and suppose for a moment it is computationally intensive. Figure 2.1(a) depicts the case when six training runs (the circles) are used. The mean of the distribution provided by the GPE (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. Note how the approximation improves when more training runs are used, as shown in Figure 2.1(b). On the other hand, Figure 2.2 shows upper and lower bounds of two standard deviations for the predictive mean of the GPE. As the number of training runs increases, there is a reduction of the uncertainty in the value of the predictive mean. Note how the uncertainty is equal to zero in each of the training runs, as it would be expected, since the GPE reproduces the simulator's output at these points. Observe however that for both cases, the uncertainty increases rapidly when extrapolating the training set.

2.3 Some Implementation Details

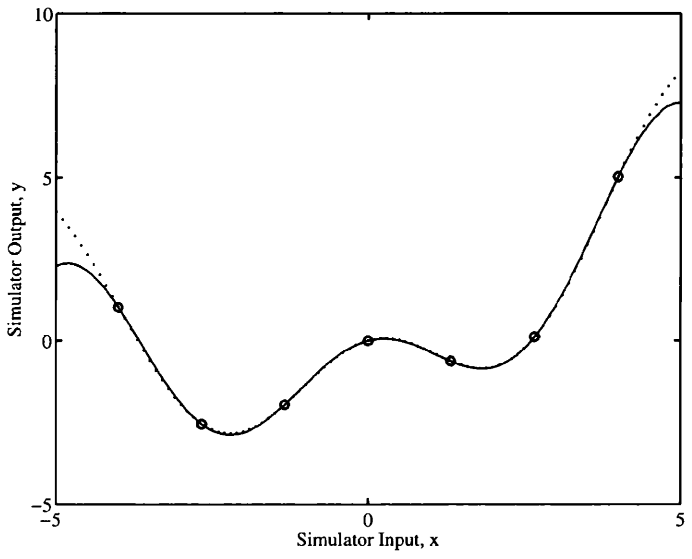
2.3.1 The Initial Design

In Algorithm 1, the selection of an initial design $\mathfrak{D} = \{\mathbf{x}_i\}_{i=1}^{n}$ in step 1 must be done carefully. It would be ideal to extract the most information about $\eta(\cdot)$ out of the minimum number of evaluations possible. The choice of the initial design is an active research area. A copious amount of literature on the subject is available. We now present a brief overview of different approaches to this issue. For the following discussion, recall Ω denotes the domain of $\eta(\cdot)$.

One natural strategy for selecting a set of inputs to evaluate the code at is to choose $\mathfrak{D}$ such that its elements are spread evenly throughout Ω. In that case, random sampling from the distribution of the inputs could be the suitable strategy. Nevertheless, this scheme might have a drawback. Consider the case of the output being influenced by only a few components of the input. In this situation, each dimension should be guaranteed to be fully represented. McKay *et al.* (1979) proposed *Latin hypercube sampling* as a solution to this problem. Furthermore, it is a computationally cheap method. Latin hypercube sampling can be viewed as an extension of Latin square designs to higher dimensions. The construction of a Latin hypercube design is carried out as shown in Algorithm 2.



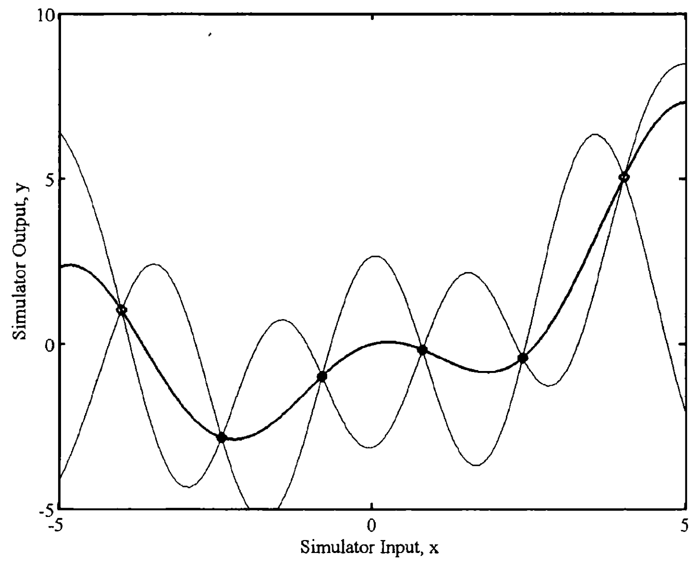
(a) Predictive mean using 6 design points



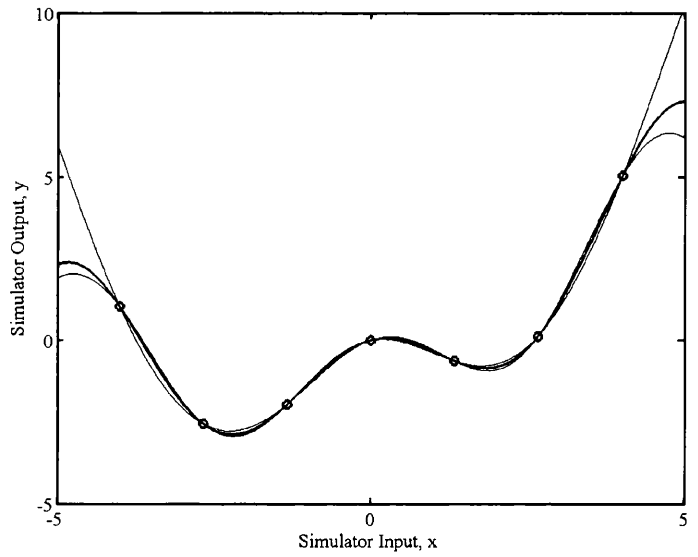
(b) Predictive mean using 7 design points

Figure 2.1: Approximation to the output of the simulator $\eta(\mathbf{x}) = 0.5\mathbf{x} - \mathbf{x}\sin(\mathbf{x})$ with the mean of a GPE; (-): output of the simulator, (o): training runs, $(\cdots)$: emulator's predictive mean. The approximation improves the more training runs available.

The notion of points spread evenly throughout Ω can have many other interpretations. Johnson *et al.* (1990) provide a criterion for quantifying this desirable property. Given an arbitrary design $\mathfrak{D}$ and a distance function $\rho : \Omega \times \Omega \longrightarrow \mathbb{R}$, they call $\mathfrak{D}^o$ a *maximin design* if no two points are too close together, that is, if the minimum distance between any two points is maximized. Formally,



(a) Uncertainty about the predictive mean using 6 design points



(b) Uncertainty about the predictive mean using 7 design points

Figure 2.2: Uncertainty bounds (two standard deviations) about the predictive mean of a GPE that approximates the output of the simulator $\eta(\mathbf{x}) = 0.5\mathbf{x} - \mathbf{x}\sin(\mathbf{x})$. The uncertainty is reduced the more training runs available.

$$\min_{\mathbf{x},\mathbf{x}' \in \mathfrak{D}^o} \rho(\mathbf{x},\mathbf{x}') = \max_{\mathfrak{D}} \min_{\mathbf{x},\mathbf{x}' \in \mathfrak{D}} \rho(\mathbf{x},\mathbf{x}') \tag{2.16}$$

In the same way, they call $\mathfrak{D}^*$ a *minimax design* if the maximum distance between any $\mathbf{x} \in \Omega$ and the candidate design $\mathfrak{D}^*$ is minimum over all possible designs $\mathfrak{D}$. That is,

Alg. 2 Latin hypercube.

```
Input: Number of design points $n$, input domain $\Omega \subseteq \mathbb{R}^p$
Output: Latin hypercube design $\mathfrak{D}$
  begin
    1. Divide the range of the $p$ components of $\mathbf{x} \in \Omega$ into $q$ regions of equal
       marginal probability.
    2. Sample once from each of these regions.
    3. To obtain $\mathfrak{D}$, sample without replacement from $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{iq}\}$, for $i = 1, \ldots, p$.
  end
```

$$\min_{\mathfrak{D}} \max_{\mathbf{x} \in \mathbb{R}^d} \rho(\mathbf{x}, \mathfrak{D}) = \max_{\mathbf{x} \in \mathbb{R}^d} \rho(\mathbf{x}, \mathfrak{D}^*) \tag{2.17}$$

Following the above ideas, Morris and Mitchell (1995) extended the definition of minimax design in the following way. Given n points in a design $\mathfrak{D}$, construct an increasing-order list $(d_1, \ldots, d_m)$ whose elements are the distinct values of the inter-element distances. Also, construct an index list $(J_1, \ldots, J_m)$ where J_j is the number of pairs of elements in the design separated by distance d_j. It is easy to see that $m \in \{1, \ldots, \binom{n}{2}\}$. Note that a design that maximizes d_1 and minimizes J_1 is a maximin design in the original sense. Morris and Mitchell (1995) call $\mathfrak{D}$ a maximin design if among the available designs it

- maximizes d_1, and among designs for which this is true
- minimizes J_1, and among designs for which this is true
- maximizes d_2, and among designs for which this is true
- minimizes J_2, and among designs for which this is true

  ⋮
- maximizes d_m, and among designs for which this is true
- minimizes J_m

The authors use a design construction algorithm based on a technique known as simulated annealing, for which we present a very brief summary. They start with a Latin hypercube design and randomly perturb it (e.g. changing the values of a randomly selected column of the design matrix). The perturbed design $\mathfrak{D}'$ is evaluated via a fitness function $\varphi : \mathfrak{D}' \longrightarrow \mathbb{R}$, and if it leads to an improvement, it is taken as the current design. On the other hand, if it is less fit, the replacement of $\mathfrak{D}$ with $\mathfrak{D}'$ occurs with probability

$$\pi = exp\{-[\varphi(\mathfrak{D}') - \varphi(\mathfrak{D})]/t\} \tag{2.18}$$

where t is a parameter (known as "temperature") that decreases with every iteration. That way, a less fit design is more likely to be chosen in early iterations, when t is high. The values of certain parameters, such as initial temperature or rate of cooling are selected heuristically and/or based on experience.

Both Latin hypercube sampling and minimax and maximin designs rely on the idea of a set of inputs being evenly spread throughout Ω. A different criterion is *maximum entropy sampling*, whose objective is to maximize the gain in information for prediction at unsampled sites. Shewry and Wynn (1987) used the Shannon entropy of a random vector Υ with density function $p(\cdot)$, namely

$$Ent(\Upsilon) = E_{\Upsilon}[-log\{p(\Upsilon)\}] \tag{2.19}$$

to show that the expected change in information is maximized by the design $\mathfrak{D}_E$ whose inputs maximize the entropy of the output. They called $\mathfrak{D}_E$ a maximum entropy design and established that it maximizes the determinant of the variance-covariance matrix of the output. Currin *et al.* (1988) applied the maximum entropy criterion to select designs for expensive computer experiments. They adopted the Bayesian approach and the Gaussian process model discussed previously.

As mentioned before, the choice of design points is an active research area. The overview presented here is by no means exhaustive. There exist a number of different strategies such as criterion based designs, sequential designs, combined designs and designs based on optimization procedures. A more complete account can be consulted in Santner *et al.* (2003). Throughout this dissertation, unless otherwise stated, the Latin hypercube sampling strategy proposed by McKay *et al.* (1979) is employed.

2.3.2 The Smoothness Parameters

It was previously assumed that $\eta(\cdot)$ is a smooth function of its inputs. Additionally, the correlation function between any two inputs, $\mathbf{x}$ and $\mathbf{x}'$, was defined by Eq. (2.3). A crucial component of such correlation function is the diagonal matrix $\mathbf{B}$, which contains what are known as *smoothness parameters*. Intuitively, these parameters specify how far an untried input needs to go from a design point before the uncertainty becomes appreciable. In other words, they quantify the rate at which the output varies as the one input changes.

There are at several techniques to estimate the smoothness parameters from the vector of observations $\mathbf{y} = [\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)]^{\intercal}$. One is called *cross-validation* and proceeds as follows. Define $\mathbf{y}_{-i}$ as the vector that excludes the i-th observation from $\mathbf{y}$. Let $\rho(\cdot)$ be a distance function defined as above and let $\mathbf{B}$ be given. Denote the diagonal of $\mathbf{B}$ by $\mathbf{b}$. The computation of the cross-validation estimator of $\widetilde{\mathbf{b}}$ is summarized in Algorithm 3 below.

Alg. 3 Cross-validation.

```
Input: Initial smoothness parameters $\mathbf{b}$
Output: Cross-validation estimator $\widetilde{\mathbf{b}}$
    begin
        for $i = 1, \ldots, n$
        1. Calculate the predictive mean $\mathrm{E}[\eta(\mathbf{x})|\mathbf{y}_{-i}] = m^{**}_{-i}(\mathbf{x})$
        2. Calculate the distance $d_i = \rho(m^{**}_{-i}(\cdot), \eta(\mathbf{x}_i))$
        end
        3. Compute $\widetilde{\mathbf{b}} = \min_{\mathbf{b}} \sum_{i=1}^{n} d_i$
    end
```

The technique to estimate the smoothness parameters used in this dissertation is to derive the density function $f(\mathbf{B}|\mathbf{y})$ and obtain a maximum likelihood estimator. Using the same definitions as in the prior-to-posterior analysis in the beginning of this chapter, Haylock (1996) shows that the posterior likelihood function is

$$f(\mathbf{B}|\mathbf{y}) \propto (\widehat{\sigma}^2)^{-\frac{(n-q)}{2}} |\mathbf{A}|^{-\frac{1}{2}} |\mathbf{H}^{\mathsf{T}}\mathbf{A}\mathbf{H}|^{-\frac{1}{2}} \tag{2.20}$$

Hankin (2005) observes that it is more convenient to work with the logarithms of the smoothness parameters, as this will force them to be positive and will lead B to be positive definite.

Another approach to estimate the smoothness parameters is to use Markov chain Monte Carlo (MCMC) (Neal, 1998). However, Oakley and O'Hagan (2004) point out the intensive computation required. More recently, Toal *et al.* (2011) have applied population-based methods such as particle swarm optimization.

2.4 Conclusions

In this chapter, the mathematical background of GPEs was presented. Basic definitions, assumptions and algorithms were presented. The output of a simple simulator was emulated in order to illustrate the properties of the metamodel. Additionally, some aspects of the implementation were detailed, such as the selection of the initial design and the estimation of the smoothness parameters. In the next chapter, we present a first application of GPEs applied to deterministic engineering systems by proposing them as an efficient predictive tool for expensive structural dynamic analysis.

Chapter 3

Deterministic Structural Dynamic Analysis [1]

3.1 Introduction

We begin our study of GPEs applied to deterministic engineering systems by proposing GPEs as a tool for reducing the computer cost of expensive structural dynamic analyses. In such cases, running a detailed high-resolution finite element model can be costly even for obtaining the dynamic response at few frequency points. To address this problem, this chapter investigates the possibility of representing the output of an expensive finite element code as a Gaussian stochastic process. GPEs are applied to both simulated and experimentally measured data from the frequency response of a cantilever plate excited by a harmonic force. The dynamic response over three frequency ranges is approximated using only a small number of response values, obtained both by running a finite element model at carefully selected frequency points and from experimental measurements. The results are then validated applying some adequacy diagnostics. It is shown that the GPE method can be an effective predictive tool for deterministic engineering systems, whenever the data is expensive to obtain, either from a computer-intensive code or a resource-consuming experiment. To the best of our knowledge, GPEs have not been implemented in structural dynamics and it can be seen it has good potential for this area of engineering.

[1]The ideas developed in this chapter have been published as DiazDelaO, F.A. & Adhikari, S. (2010), "Structural dynamic analysis using Gaussian process emulators", *Engineering Computations*, **27** (5) 580-605.

3.2 Damped Structural Dynamics

Consider the problem of modeling the response of a damped structural system subject to different frequency ranges of vibration. Viscous damping is the most common model for representing vibration damping in linear systems. First introduced by Rayleigh (1877), this model assumes that the instantaneous generalized velocities are the only relevant variables that determine damping. Viscous damping models are used widely for their simplicity and mathematical convenience even though the behavior of real structural materials is, at best, poorly mimicked by simple viscous models. For this reason it is well recognized that in general a physically realistic model of damping will not be viscous. Damping models in which the dissipative forces depend on any quantity other than the instantaneous generalized velocities are nonviscous damping models. Mathematically, any causal model which makes the energy dissipation functional nonnegative is a possible candidate for a nonviscous damping model. Clearly a wide range of choice is possible, either based on the physics of the problem, or by a priori selecting a model and fitting its parameters from experiments. For the sake of generality, we consider nonviscously (or viscoelastically) damped systems (see for example Torvik and Bagley (1987), Woodhouse (1998), Maia *et al.* (1998), and Adhikari and Woodhouse (2003)).

The equation of motion of a N-degree-of-freedom linear system with such damping can be expressed by

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \int_0^t \mathcal{G}(t-\tau)\,\dot{\mathbf{q}}(\tau)\,d\tau + \mathbf{K}\mathbf{q}(t) = \mathbf{f}(t) \tag{3.1}$$

where $\mathbf{q}(t) \in \mathbb{R}^N$ is the displacement vector, $\mathbf{f}(t) \in \mathbb{R}^N$ is the forcing vector, $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix, $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the stiffness matrix, and $\mathcal{G}(t) \in \mathbb{R}^{N\times N}$ is the matrix of damping kernel functions. The kernel functions $\mathcal{G}(t)$ are known as retardation functions, heredity functions, after-effect functions or relaxation functions in the context of different subjects. Early works in this area can be traced back to Biot (1958) in the context of viscoelastic materials. In the limit when $\mathcal{G}(t-\tau) = \mathbf{C}\,\delta(t-\tau)$, where $\delta(t)$ is the Dirac-delta function, Eq. (3.1) reduces to the case of viscous damping. Taking the Fourier transform of Eq. (3.1), the equation of motion in the frequency domain can be expressed in terms of the frequency level, $\omega \in [0, ..., \infty)$ as

$$\mathbf{D}(\omega)\overline{\mathbf{q}}(\omega) = \overline{\mathbf{f}}(\omega) \tag{3.2}$$

where $\overline{\mathbf{q}}(\omega)$ and $\overline{\mathbf{f}}(\omega)$ are the Fourier transforms of $\mathbf{q}(t)$ and $\mathrm{f}(t)$, respectively. The *dynamic stiffness matrix* $\mathbf{D}(\omega)$ is the complex symmetric matrix given by

$$\mathbf{D}(\omega) = -\omega^2\mathbf{M} + i\omega\mathbf{G}(\omega) + \mathbf{K} \tag{3.3}$$

where $\mathbf{G}(\omega)$ is the Fourier transform of $\mathcal{G}(t)$. Provided that $\mathbf{D}(\omega)^{-1}$ exists, the response vector becomes $\overline{\mathbf{q}}(\omega) = \mathbf{D}(\omega)^{-1}\overline{\mathbf{f}}(\omega)$. Suppose there is interest in working with some linear function of the elements of $\overline{\mathbf{q}}(\omega)$, namely

$$\phi(\omega) = \mathbf{Q}\overline{\mathbf{q}}(\omega) = \mathbf{Q}\mathbf{D}(\omega)^{-1}\overline{\mathbf{f}}(\omega) \tag{3.4}$$

where $\mathbf{Q}$ is a rectangular matrix. Since $\phi(\cdot)$ is a complex-valued function, only its modulus is relevant in practice. That way, $\eta(\omega)$ is expressed as

$$\eta(\omega) = \left|\mathbf{Q}\mathbf{D}(\omega)^{-1}\overline{\mathbf{f}}(\omega)\right| \tag{3.5}$$

For systems with general nonproportional damping as considered here, it is in general not possible to represent the response in terms of undamped modes. In such cases the response needs to be expressed in terms of the complex modes of the system (Adhikari, 2004). The computation of complex modes is numerically much more expensive as the size of the eigenvalue problem doubles (Newland, 1989) due to the use of the state-space approach. For systems with general frequency-dependent viscoelastic damping models, a higher-order nonlinear complex eigenvalue problem (Wagner and Adhikari, 2003) needs to be solved in order to obtain the dynamic response in terms of the modal series. The solution of such eigenvalue problems is significantly more expensive compared to even nonproportional viscously damped systems. Adhikari and Wagner (2004) showed that for such system a direct integration scheme in the time-domain can be more efficient compared to the modal approach. In this chapter, the alternative approach of obtaining the response by solving the linear system (3.2) for only few frequency points is explored. In such context, a GPE might be a convenient choice. In the following section an introductory example is discussed.

3.3 A Simple Example

Consider the simple three-degree-of-freedom spring-mass system shown in Figure 3.1. For purposes of illustration, the simulator of the corresponding frequency response function (FRF) is regarded as if it were computer intensive. Let the mass of each block be 1 kg, the stiffness of each spring be 1 N/m, and the viscous damping constant of the damper associated with each block be 0.8 Ns/m. The mass, stiffness, and damping matrices of this

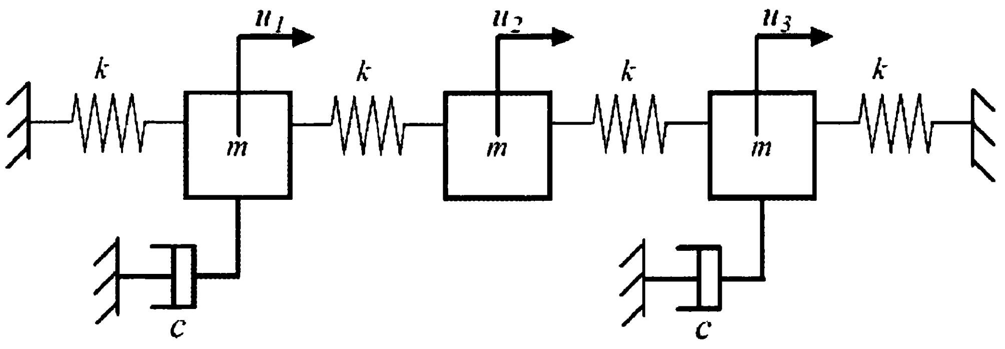


Figure 3.1: Three-degree-of-freedom non-viscously damped spring-mass system; $m = 1$ kg, $k = 1$ N/m, $c = 0.8$ Ns/m. The relaxation parameter λ is taken as the first natural frequency of the system, $\sqrt{2-\sqrt{2}}$ s^{-1}.

simple system can be can be obtained as:

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 2k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & 2k \end{bmatrix}$$

$$\text{and} \quad \mathbf{G}(\omega) = \frac{\lambda}{\lambda + i\omega} \begin{bmatrix} c & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & c \end{bmatrix}. \tag{3.6}$$

where $m = 1$, $k = 1$, $c = 0.8$, and λ, the relaxation parameter, is chosen to be equal to the first natural frequency of the system, $\sqrt{2-\sqrt{2}}$ s^{-1}. Note that the system has nonproportional damping. Let the forcing vector be $\mathbf{f} = [0, 1, 0]^{\mathsf{T}}$. In that case, the FRF corresponding to the ℓ-th degree of freedom has the following form

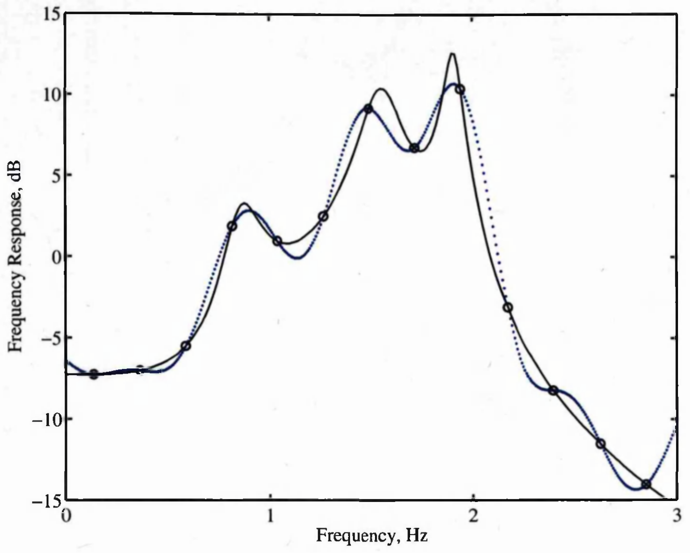
$$\phi_\ell(\omega) = \mathbf{D}(\omega)_\ell^{-1}\bar{\mathbf{f}}(\omega) \tag{3.7}$$

where $\mathbf{D}(\omega)_\ell^{-1}$ denotes the ℓ-th row of $\mathbf{D}(\omega)^{-1}$, for $\ell = 1, \ldots, 3$. Since $\phi_\ell(\cdot)$ is a complex-valued function, the simulator for ℓ fixed is the following single-variable function

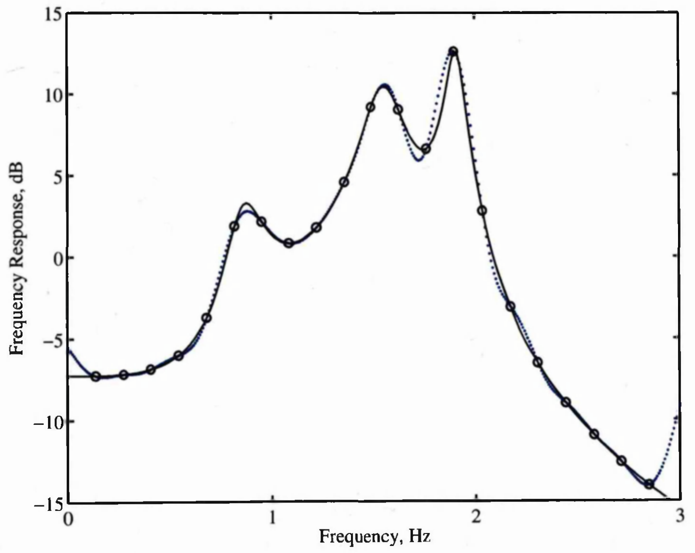
$$\eta(\omega) = |\phi_\ell(\omega)| = \left|\mathbf{D}(\omega)_\ell^{-1}\bar{\mathbf{f}}(\omega)\right| \tag{3.8}$$

Suppose that n design points, namely $\omega_1, \ldots, \omega_n$, are chosen in the input domain of the simulator $\eta(\cdot)$. Let $\{\eta(\omega_1), \ldots, \eta(\omega_n)\}$ be the training set resulting from the evaluation of $\eta(\cdot)$

in each of the design points. Figure 3.2(a) shows the case when $\ell = 3$ and $n = 13$ training runs (the circles) are used. As in the simple example in Chapter 2, the approximation improves when more training runs are used, as shown in Figure 3.2(b) with $n = 21$. Figure 3.3 shows



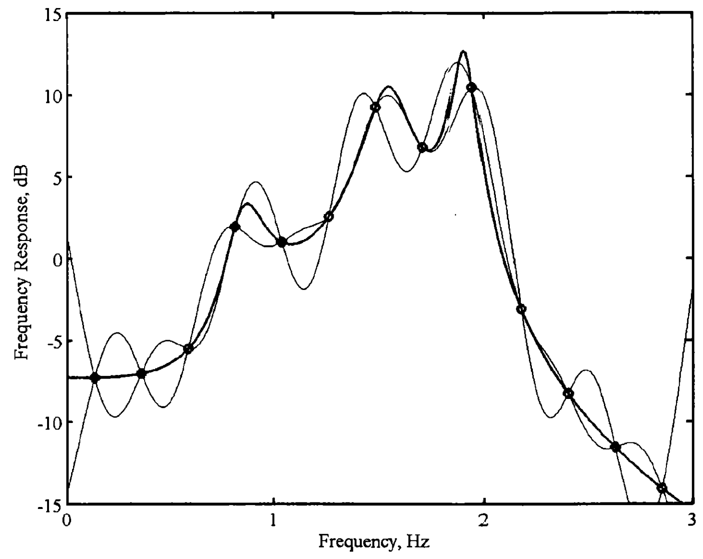
(a) Predictive mean using 13 design points.



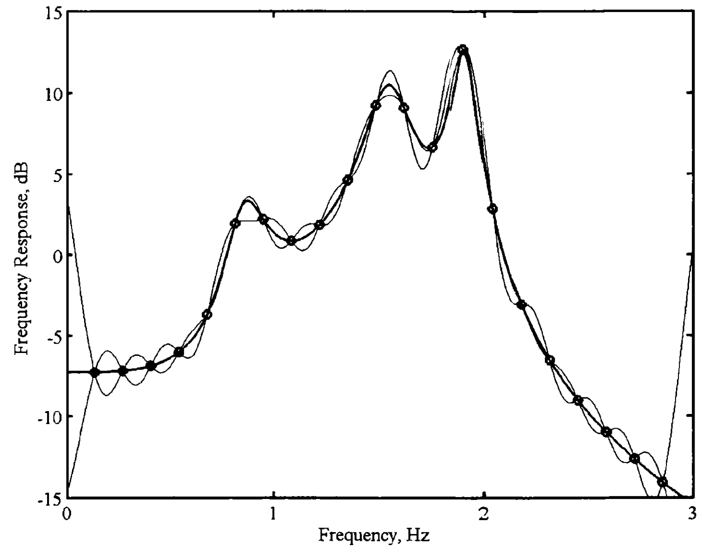
(b) Predictive mean using 21 design points.

Figure 3.2: Emulation of $\eta(\omega) = |\phi_3(\omega)|$ for a damped spring-mass system with 3 degrees of freedom. The forcing vector is $[0, 1, 0]^\intercal$; (-): simulator's output, (o): training runs, $(\cdots)$: emulator's predictive mean.

upper and lower bounds of two standard deviations for the predictive mean. The expected reduction of the uncertainty is observed when more training runs are employed.



(a) Uncertainty using 13 design points.



(b) Uncertainty using 21 design points.

Figure 3.3: Probability bounds (two standard deviations) for the predictive mean of the GPE that approximates the output of $\eta(\omega) = |\phi_3(\omega)|$ for a damped spring-mass system with 3 degrees of freedom. The forcing vector is $[0, 1, 0]^\mathsf{T}$; (-): simulator's output, (o): training runs.

3.4 Numerical Example: Frequency Response of a Cantilever Plate

Consider a finite element model of a rectangular steel plate 998 mm long, 530 mm wide, 3 mm thick, and with a mass of 12.47 kg. Suppose it is clamped along a short edge and that it has a damping patch attached to it, as shown in Figure 3.4. The resulting damping is nonproportional, since the corresponding damping matrix becomes a block matrix with some zeros along the diagonal. Hence, it cannot be represented as a positive linear combination of the mass and stiffness matrices whose diagonals are non-zero (Caughey and O'Kelly, 1965; Adhikari, 2001). Suppose that the plate is excited by a unit harmonic force and the frequency response is measured at one of the nodes. If the standard four-noded thin plate bending element model is assumed, it results in 12 degrees of freedom per element. As already mentioned in Section 3.2, the calculation of the frequency response for this kind of systems can be very expensive. Even for a relatively small case, say 25 $\times$ 15 elements, solving the linear system (3.2) for each frequency level can be very resource-consuming. Consider three frequency ranges, namely 0 - 1.0 kHz as the low-frequency range, 1.0 - 2.5 kHz as the medium-frequency range, and 2.5 - 4.0 kHz as the high-frequency range. Note that these frequency boundaries are selected on the basis of the qualitative nature of the response and devised purely for the presentation of the results.

An exploration of the applicability of GPEs for six dynamical systems was carried out. Keeping the aspect ratio of the plate, an increasing number of elements (up to 50 $\times$ 30 elements and 4650 degrees of freedom) were considered. Assuming the same boundaries for the low, medium, and high-frequency ranges and taking a resolution of 1 Hz, a simulator of the FRF was coded in Matlab$^{\text{TM}}$. For each frequency range, a training set whose size was chosen to be 5% the size of the corresponding level (50 design points for low, 75 design points for medium, 75 design points for high) was selected. The corresponding training runs were obtained by solving the linear system (3.2) at each of the design points. Using a machine with MS Windows Vista 64 bit, 2.66 GHz Quadcore Intel Xeon Processor, and 16.0 GB RAM, an emulator of the FRF was run for the six models in each frequency range and the time employed was registered. Following Keane and Nair (2005), $\mathbf{h}(\cdot) = 1$ was assumed due to the absence of prior knowledge of the mean. The smoothness parameters were obtained with the maximum likelihood method outlined in Chapter 2. The comparison of the time taken by the simulator and the emulator is shown in Table 3.1. The time taken purely by the emulator, that is, disregarding the time employed in obtaining the training runs is shown in parenthesis. Note how it remains approximately constant despite the increase in resolution. To illustrate the performance of emulation in the system with 50 $\times$ 30 elements, the predictive mean of the emulator and the corresponding probability bounds across each

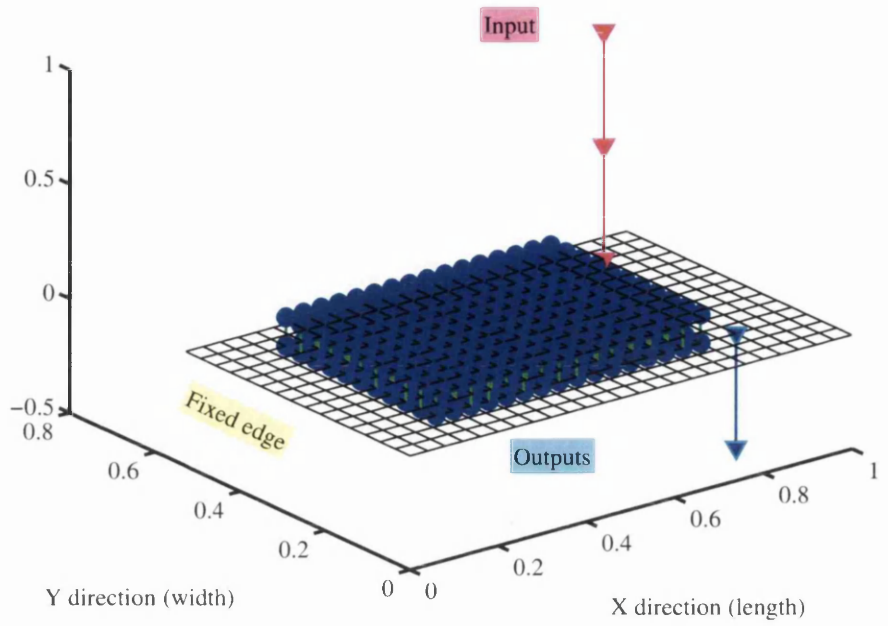
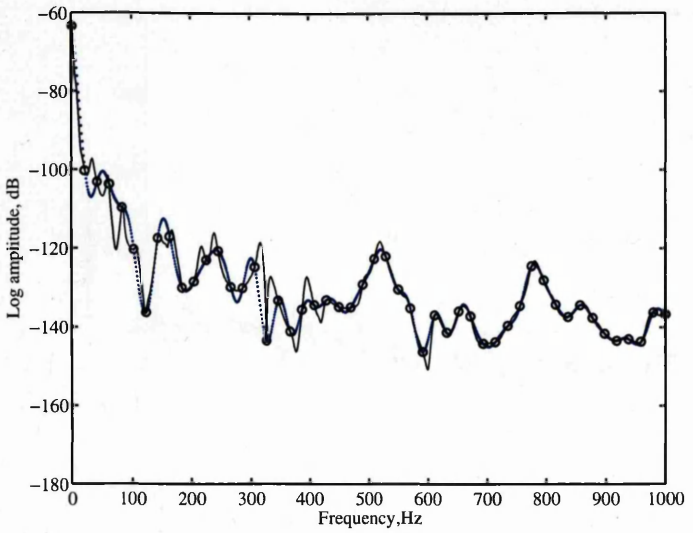


Figure 3.4: Finite element model of a steel cantilever plate with a damping patch. The material and geometric properties are: $L_x = 998$ mm, $L_y = 530$ mm, $t_h = 3$ mm, $\rho = 7860$ kg/m^3, $E = 2.0 \times 10^5$ MPa, $\mu_r = 0.3$, $W = 12.47$ kg.

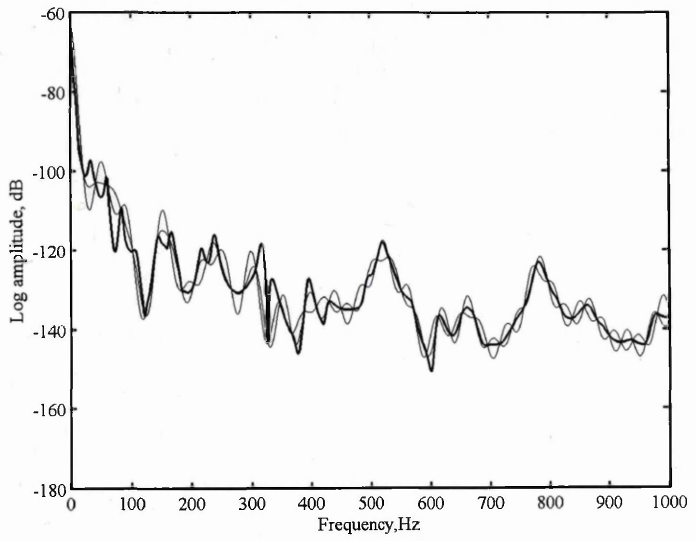
RESOLUTION		TIME (seconds)					
		Low Frequency		Medium Frequency		High Frequency	
No. elements	DOF	Simulator	Emulator	Simulator	Emulator	Simulator	Emulator
25×15	1200	794.28	40.47	1169.72	62.87	1201.85	65.71
			(0.91)		(4.74)		(7.37)
30×18	1710	2098.56	106.06	3149.57	160.64	3107.51	162.80
			(0.98)		(4.87)		(7.57)
35×21	2310	4830.03	244.50	7202.22	366.02	7207.08	365.68
			(0.96)		(4.88)		(7.21)
40×24	3000	10040.81	499.31	15025.99	749.04	14934.11	759.22
			(0.97)		(4.84)		(7.55)
45×27	3780	19253.25	992.63	29091.82	1429.92	29787.58	1495.28
			(0.94)		(4.86)		(7.62)
50×30	4650	35273.89	1763.70	53107.12	2437.01	56063.20	2781.38
			(0.97)		(4.83)		(7.49)

Table 3.1: Computation time in seconds of the simulator against the emulator, for the three frequency ranges and different resolutions. The numbers in parenthesis are the time employed purely by the emulator, without taking into account the computation of the training runs.

frequency range are shown in Figure 3.5, Figure 3.6 and Figure 3.7. Note that this numerical model is aimed at representing the experimental example studied in the next section.

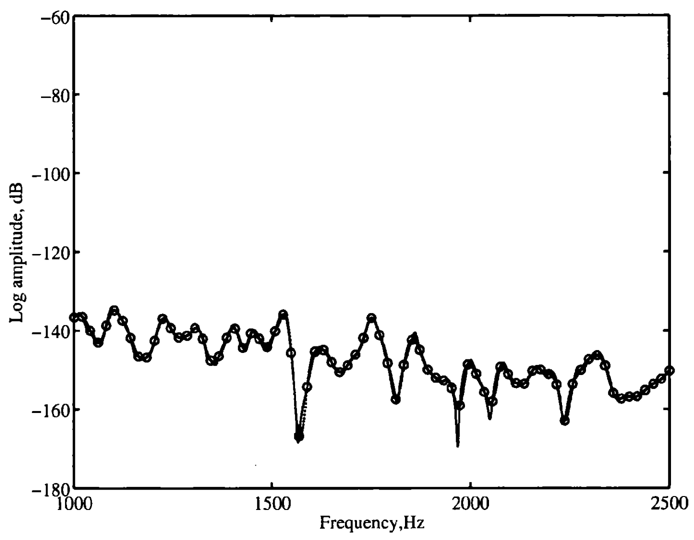


(a) Mean of the emulator

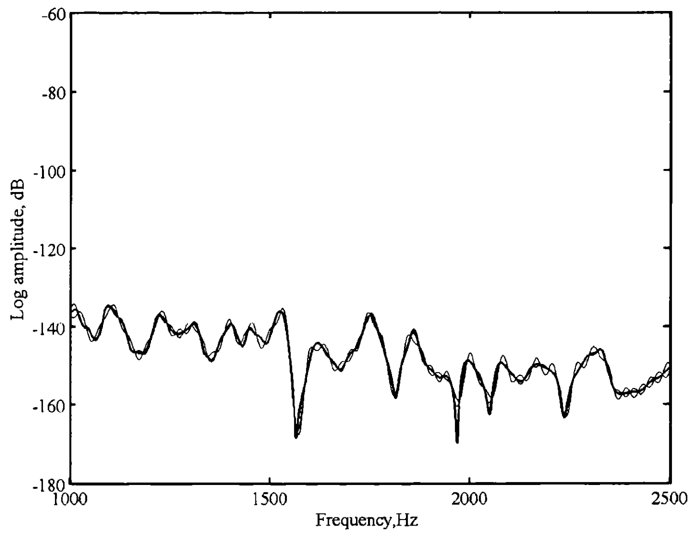


(b) Probability bounds for the mean (shaded areas)

Figure 3.5: Emulation of the response in the low-frequency range with 50 design points; (-): simulator's output, (o): training runs, (···): emulator's predictive mean. The shaded areas are probability bounds for the predictive mean (2 standard deviations).



(a) Mean of the emulator



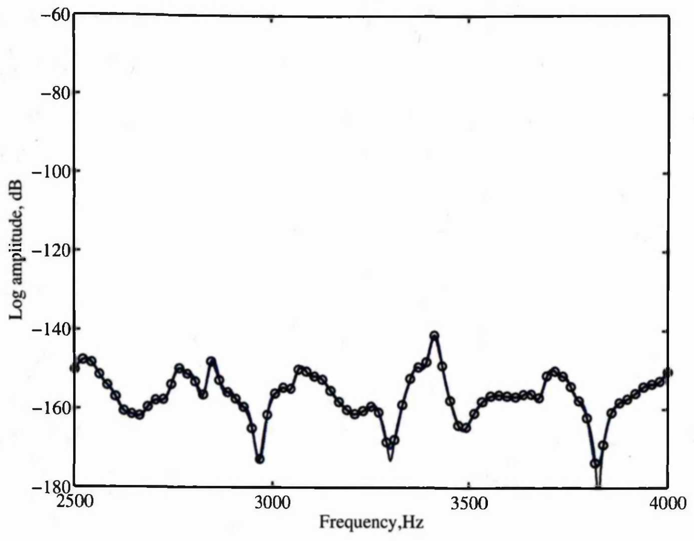
(b) Probability bounds for the mean (shaded areas)

Figure 3.6: Emulation of the response in the medium-frequency range with 75 design points; (-): simulator's output, (o): training runs, (···): emulator's predictive mean. The shaded areas are probability bounds for the predictive mean (2 standard deviations).

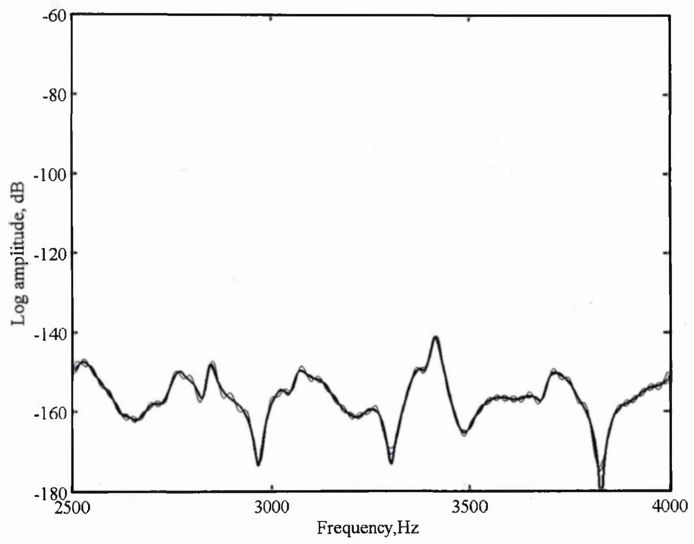
3.5 Experimental Example: Frequency Response of a Cantilever Plate

3.5.1 Experimental Setup

Suppose that the cost of obtaining measurements for a given experiment is such that they can only be generated for a very limited number of points. Also, suppose there is no suitable



(a) Mean of the emulator



(b) Probability bounds for the mean (shaded areas)

Figure 3.7: Emulation of the response in the high-frequency range with 75 design points; (-): simulator's output, (o): training runs, $(\cdots)$: emulator's predictive mean. The shaded areas are probability bounds for the predictive mean (2 standard deviations).

model to simulate the frequency response function, due perhaps to lack of knowledge about the physics of the system. In that case, the available data can be regarded as a training set upon which the emulation algorithm can be applied. In this section, an emulator is used to substitute the runs necessary to approximate experimental output (Adhikari *et al.*, 2009; Adhikari and Sarkar, 2009), whereby a cantilever plate was excited by a harmonic force and the frequency response was measured at different locations. A rectangular steel plate with the same physical and geometrical properties specified before was used, except that it had no damping patch attached. The plate was clamped along a short edge using a clamping device. The clamping device was attached to the top of a heavy concrete block and the whole assembly was placed on a steel table. Special care was taken to ensure its stability and to minimize vibration transmission. Six accelerometers were used as the response sensors. Their locations were selected such that they covered a broad area of the plate. Small holes were drilled into the plate and the accelerometers were attached by bolts through the holes. The test rig is shown in Figure 3.8.

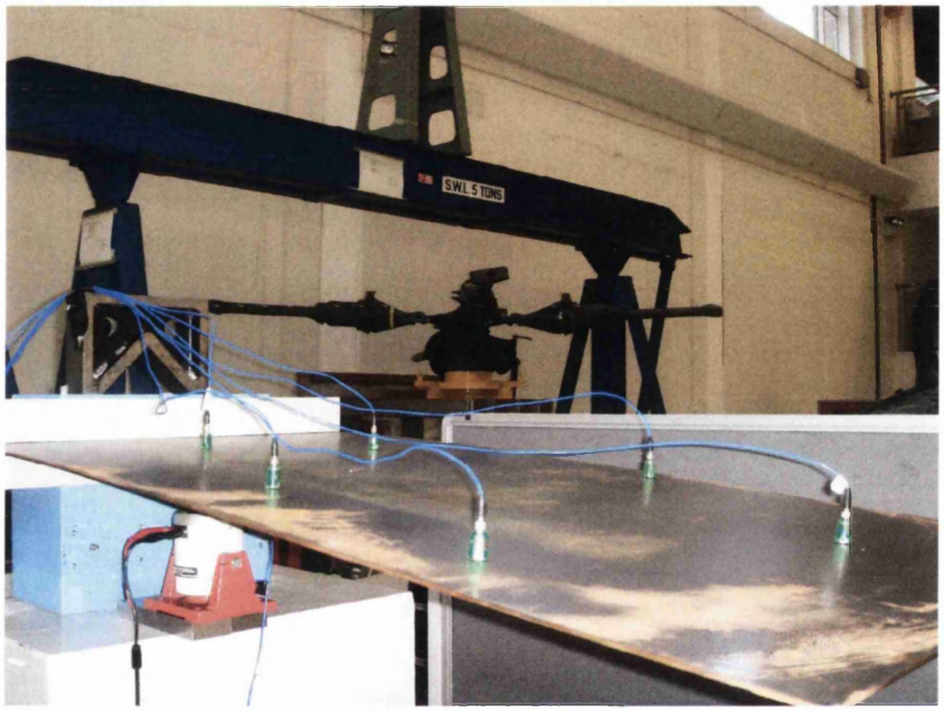


Figure 3.8: Experimental setup. A steel cantilever plate was excited by a harmonic force and the frequency response was measured. Six accelerometers were used as the response sensors. Their locations were selected to cover a broad area of the plate.

3.5.2 Experimental Methodology

Experimental modal analysis (Ewins, 2000; Maia and Silva, 1997; Silva and Maia, 1998) was used in the experiment. The three main components of the implemented experimental technique were (a) the excitation of the structure, (b) the sensing of the response, and (c) the data acquisition and processing. A shaker was used to act as an impulse hammer. The shaker generated impulses at a pulse interval of 20s and a pulse width of 0.01s. It was placed so that it impacted at a particular node of the plate. It was driven by a signal from a Simulink$^{\text{TM}}$ and dSpace$^{\text{TM}}$ system via a power amplifier. A hard steel tip is used for the hammer to increase the frequency range of excitation. The steel tip used in the experiment only gave clean data up to approximately 4500 Hz. Therefore, 4000 Hz was used as the upper limit of the frequency in the measured frequency response functions. The data logged beyond 4000 Hz were ignored. The data obtained are available on the world wide web for research purposes at http://engweb.swan.ac.uk/~adhikaris/uq/.

3.5.3 Emulation of Experimental Data

Emulation was performed to approximate the response of one of the nodes to vibration in the three frequency ranges, where 50, 75, and 75 design points were respectively employed. Analogously to the emulation of simulated data in Section 3.4, $\mathbf{h}(\cdot) = 1$ was assumed due to the absence of prior knowledge of the mean. The smoothness parameters were obtained with the maximum likelihood method mentioned in Chapter 2. The results are shown in Figure 3.9, Figure 3.10, and Figure 3.11.

Note that the number of design points for each frequency range was chosen arbitrarily. However, if it were truly expensive to carry out the experiment, the number of design points would depend on the cost of generating data. Moreover, the experimental data for every frequency level would not be available for comparison with the emulator's predictive mean. This brings up the problem of measuring if the simulator is a suitable representation of the emulator. This problem is addressed in Section 3.7.

3.6 Discussion of the Method and the Results

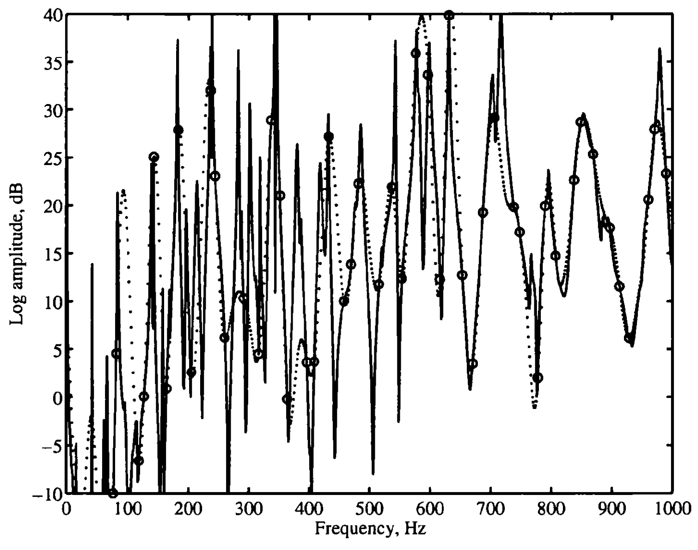
Based on the above results, it can be argued that the use of GPEs in the context of structural dynamics can help approach the following questions:

a) *Efficiency* - Can the output of a structural dynamics simulator be approximated using only a few trial runs?

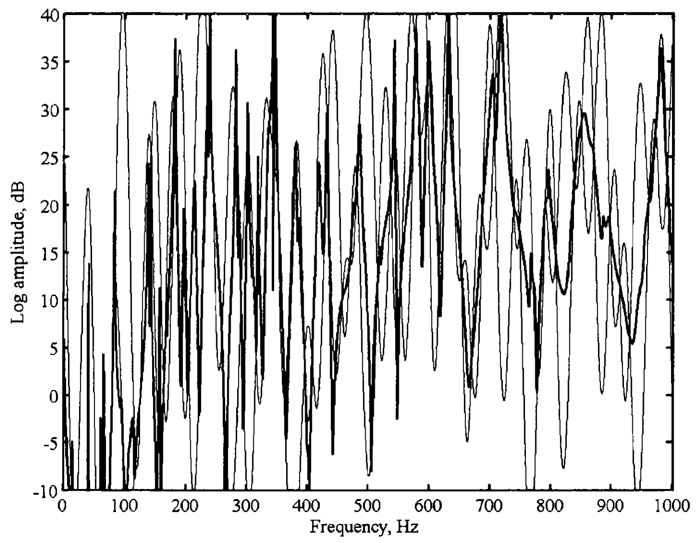
b) *Computational cost* - Can the number of floating point operations and the CPU time employed by an expensive simulator be dramatically reduced but still produce a satisfactory output?

c) *Interpolation of experimental data* - Can expensive experimental data be confidently interpolated to cope with the lack of a mathematical or computational model?

Regarding the first question, a GPE for the simple spring-mass system in Section 3.3 illustrates how the predictive mean can be a plausible approximation to the corresponding FRF. Related to to the second question, the dynamic response of systems with up to 4650 degrees of freedom was emulated in Section 3.4. Note that when the complete model was run, the



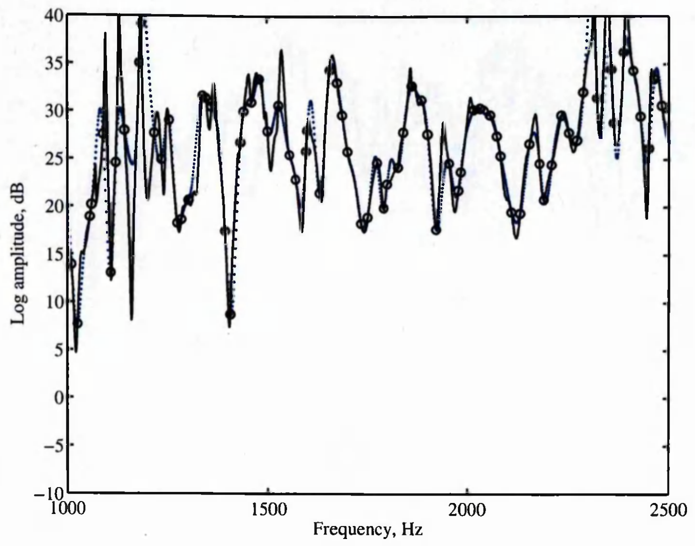
(a) Mean of the emulator



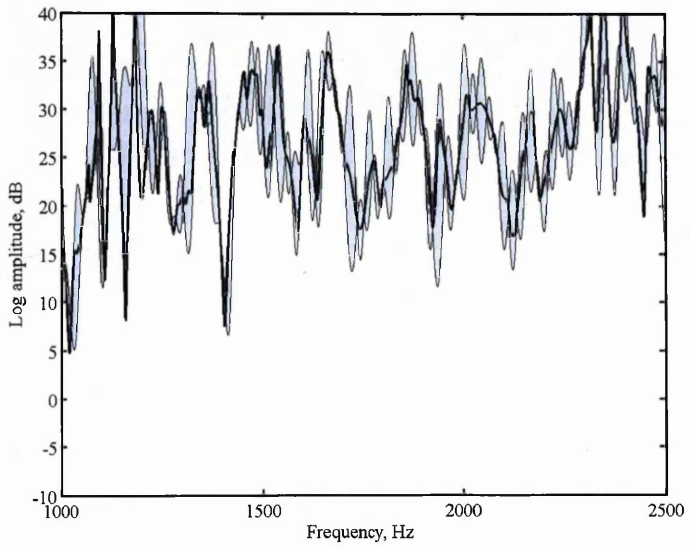
(b) Probability bounds for the mean (shaded areas)

Figure 3.9: Emulation of the response in the low-frequency range with 50 design points obtained from the experiment; (-): experimental results, (o): training runs, $(\cdots)$: emulator's predictive mean. The shaded areas are probability bounds for the predictive mean (2 standard deviations).

linear system (3.2) had to be solved 4000 times. Adopting the GPE approach, it had to be solved only 200 times, equivalent to the number of training runs necessary to approximate the output across the frequency range. The saving in CPU time was considerable and it was observed that the computational burden was mainly due to the calculation of the training runs, not to the emulation itself. From the figures presented, the approximation looks particularly appealing for the medium and high frequency ranges where the frequency re-



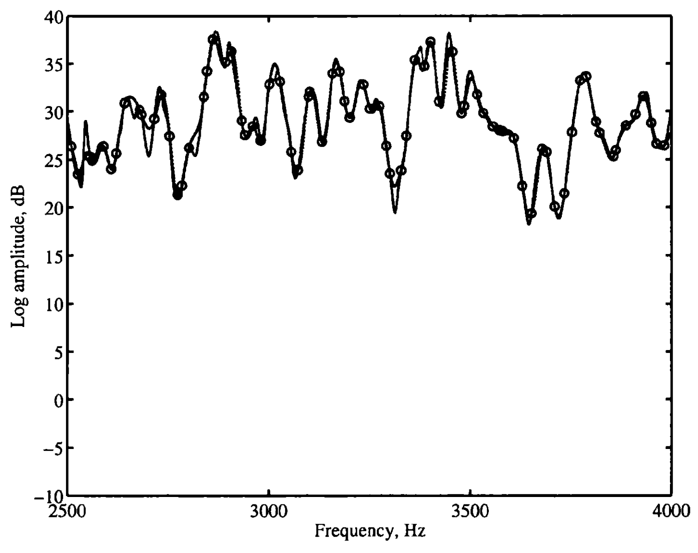
(a) Mean of the emulator



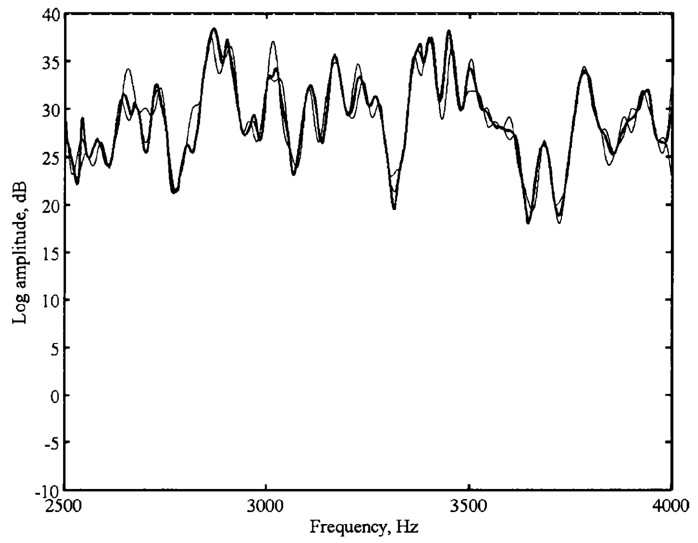
(b) Probability bounds for the mean (shaded areas)

Figure 3.10: Emulation of the response in the medium-frequency range with 75 design points obtained from the experiment; (-): experimental results, (o): training runs, (···): emulator's predictive mean. The shaded areas are probability bounds for the predictive mean (2 standard deviations).

sponse function is smoother compared to that in the low frequency region. This smoothness can be attributed to the phenomenon of modal overlap for high frequencies (see for example Manohar and Keane (1994)). Nevertheless, this is encouraging since the medium and high frequencies are in the computationally demanding ranges of vibration where numerical models of real-life systems can have several millions of degrees of freedom. With regards to the third question, an FRF obtained via experimental modal analysis was emulated in



(a) Mean of the emulator



(b) Probability bounds for the mean (shaded areas)

Figure 3.11: Emulation of the response in the high-frequency range with 75 design points obtained from the experiment; (-): experimental results, (o): training runs, (···): emulator's predictive mean. The shaded areas are probability bounds for the predictive mean (2 standard deviations).

Section 3.5. Real test data were used as the set of training runs necessary to construct a GPE and the experimental output was compared with the corresponding approximation.

As mentioned before, a GPE is a statistical approximation to the simulator. For both the simulated and the experimental examples presented here, the predictive mean and the corresponding uncertainty bounds were provided. This is one of the main differences between emulation and traditional interpolation techniques: emulation provides a probability distribution and thus a method of quantifying the uncertainty that arises from being able to evaluate the simulator very few times. In the examples presented, the emulator's output seemed to represent the simulator's output fairly well. Note, however, that visual inspection is not enough to rigourously determine the quality of the agreement between the simulator's output and the emulator's predictive mean. Despite the computation time for some of the simulators employed took tens of hours, it was still possible to obtain their output on the entire frequency range. For more expensive systems, the comparison of the simulator's output against the emulator's predictive mean would be, by definition, only possible in a very reduced number of points. Visual inspection would be unable to detect whether the emulator misrepresents the simulator's output, in which case, all inferences made using the emulator would be spurious. Moreover, in case of having more than two inputs, graphical comparison is not possible. It is therefore important to have a suitable means of assessing the adequacy of a GPE.

3.7 Emulator Validation

Although the Gaussian process is a flexible class of distributions to represent prior beliefs about a computer model, a GPE might poorly represent the simulator due to a wrong choice of the mean and covariance structures or a wrong choice of the training set that might induce an inappropriate estimation of parameters. To cope with these disadvantages, Bastos and O'Hagan (2009) proposed several methods for the validation of the emulators, some of which were applied to the results presented above.

Let $\Omega^* = \{\omega_1^*, \ldots, \omega_v^*\}$ be a set of *validation points* in the frequency domain, different from the already chosen design points, but similarly chosen using a criterion such as Latin hypercube sampling. The corresponding validation data are $\mathbf{y}^* = [\eta(\omega_1^*), \ldots, \eta(\omega_v^*)]^\intercal$. One possible diagnostic of $\mathbf{y}^*$ is to calculate the standardized prediction errors of the simulator's output and the predictive mean of the emulator given the training data. That is, for $j = 1, \ldots, v$

$$\delta_j^{se}(\mathbf{y}^*) = \frac{\mathbf{y}_j^* - m^{**}(\eta(\omega_j^*))}{\widehat{\sigma}\sqrt{C^{**}(\eta(\omega_j^*), \eta(\omega_j^*))}} \tag{3.9}$$

Each one of these standardized prediction errors has a Student's t-distribution, conditional on original vector of observations $\mathbf{y}$ (and on the smoothness parameters). Note that the size of the training data can be large enough such that $\delta_j^{se}(\mathbf{y}^*)$ approximates a standard normal distribution. Therefore, $|\delta_j^{se}(\mathbf{y}^*)| > 2$ for a high j would hint inadequacy of the GPE in a neighborhood of ω_j^*.

The emulation of experimental data in Section 3.5 was validated using this measure. The validation set was chosen to have 50 validation points. Figure 3.12(a), Figure 3.13 (a), and

Figure 3.14 (a) plot the predictive mean of the emulator against the standardized errors in the low, medium and high-frequency ranges. This type of graph can help to identify problems in the specification of the predictive mean, a situation that would be evident if, for example, most of the points lied in the positive area or viceversa. Additionally, Figure 3.12(b), Figure 3.13(b), and Figure 3.14(b) are the plots of the validation points against the standardized errors. This type of graph can be used to identify areas of the input domain for which the emulator misrepresents the simulator. In this case, a GPE appeared to be a sensible choice to represent the simulator, confirming what had been suggested by Figure 3.9, Figure 3.10, and Figure 3.11. Similarly satisfactory results were obtained for each of the frequency ranges.

A disadvantage of using the individual standardized errors is that correlation between the elements of the validation data set is not taken into account. Another possibility is to employ the Mahalanobis distance of $\mathbf{y}^*$, defined as

$$\delta_{MD}(\mathbf{y}^*) = \{\mathbf{y}^* - \mathbf{m}^{**}[\eta(\Omega^*)]\}^\mathsf{T}\mathbf{V}[\eta(\Omega^*)]^{-1}\{\mathbf{y}^* - \mathbf{m}^{**}[\eta(\Omega^*)]\} \tag{3.10}$$

where

$$\mathbf{m}^{**}[\eta(\Omega^*)] = [m^{**}(\eta(\omega_1^*)), \ldots, m^{**}(\eta(\omega_v^*))]^\mathsf{T} \tag{3.11}$$

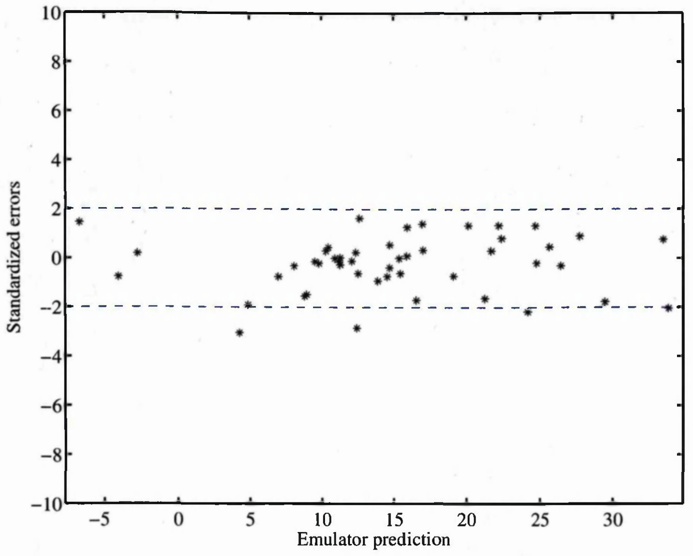
is the vector of individual predictive means and $\mathbf{V}[\eta(\Omega^*)]^{-1}$ is the inverse of the emulator's conditional covariance matrix expressed by Eq. (2.12) in Chapter 2. Note that it can be decomposed such that $\mathbf{V} = \mathbf{G}\mathbf{G}^\mathsf{T}$. That way, the vector of validation errors is

$$\boldsymbol{\delta}_\mathbf{G}(\mathbf{y}^*) = \mathbf{G}^{-1}\{\mathbf{y}^* - \mathbf{m}^{**}[\eta(\Omega^*)]\} \tag{3.12}$$

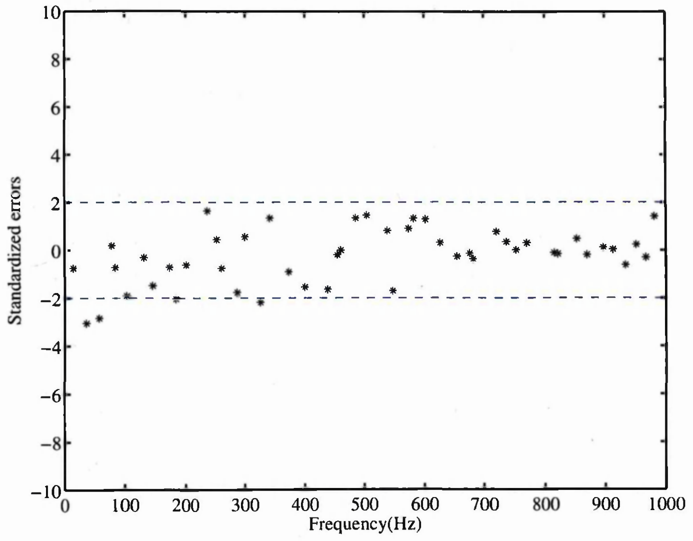
which implies that the Mahalanobis distance can be decomposed as

$$\delta_{MD}(\mathbf{y}^*) = \boldsymbol{\delta}_\mathbf{G}(\mathbf{y}^*)^\mathsf{T}\boldsymbol{\delta}_\mathbf{G}(\mathbf{y}^*) \tag{3.13}$$

One available strategy to obtain $\mathbf{G}$ is the pivoted Cholesky decomposition. It has the property of permuting the individual validation errors in Eq. (3.12), such that they are decreasingly ordered with respect to their variance. Figure 3.15, Figure 3.16, and Figure 3.17 show the individual validation errors, plotted against the index of the ordered validation data for the low, medium and high-frequency ranges. An implementation of the pivoted Cholesky decomposition is provided by Higham (2002) at http://www.maths.manchester.ac.uk/~higham/mctoolbox/. Again, the adequacy of the emulator was sought to be confirmed by the validation errors being uniformly spread around zero.



(a) Predictive mean vs. standardized errors

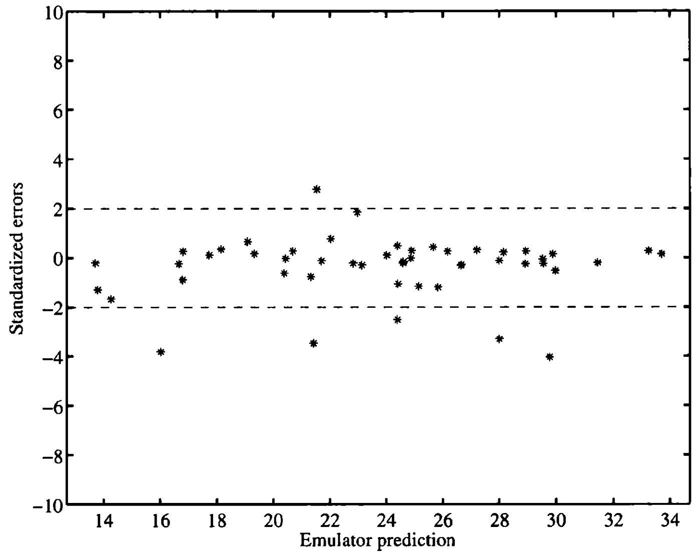


(b) Validation points vs. standardized errors

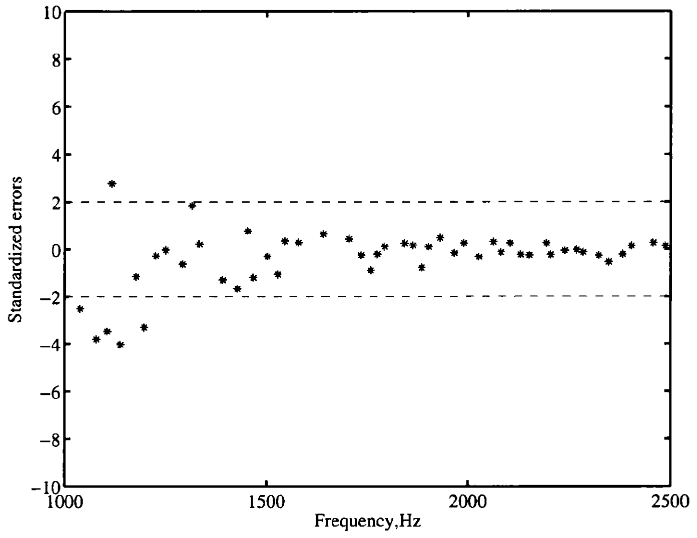
Figure 3.12: Individual standardized errors diagnostics. Low-frequency range.

3.8 Conclusions

In this chapter, the adoption of GPEs as an efficient predictive computational approach in deterministic structural dynamics was proposed. The capabilities of this computational tool were tested in both simulated and experimental contexts. Additionally, some diagnostics of adequacy were implemented, and the agreement between a simulator's and an emulator's output was verified. The main contribution of this chapter is that, although GPEs have been used in other disciplines, there is no precedent of them having been implemented for deterministic structural dynamic analysis. It was shown that the technique has good



(a) Predictive mean vs. standardized errors

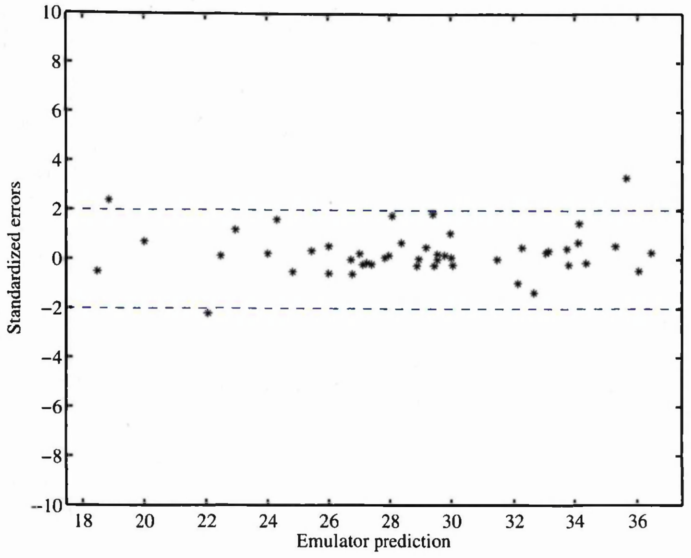


(b) Validation points vs. standardized errors

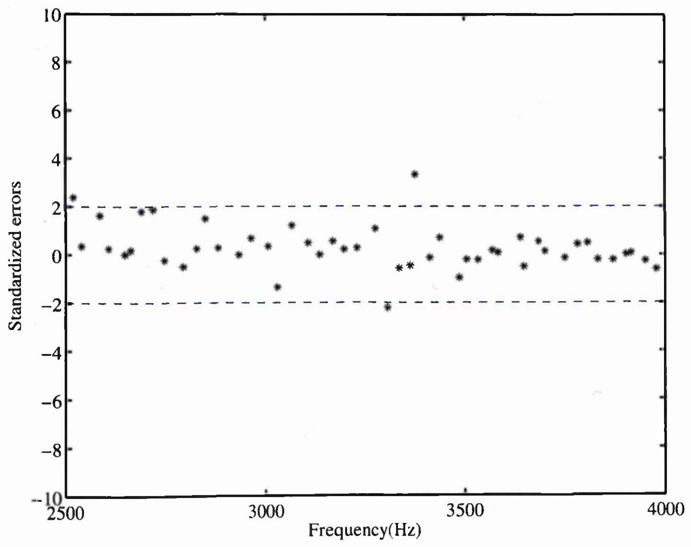
Figure 3.13: Individual standardized errors diagnostics. Medium-frequency range.

potential for this area of engineering.

The approach applied in this chapter will be extended to the case of stochastic structural dynamics in Chapter 5. Before that, the capabilities of emulators in the context of deterministic engineering systems are further explored. In Chapter 4, a novel coupling between GPEs and the domain decomposition method is proposed. Based on this coupling it will be shown how to assimilate a low-fidelity finite element model with a more expensive high-fidelity model.



(a) Predictive mean vs. standardized errors



(b) Validation points vs. standardized errors

Figure 3.14: Individual standardized errors diagnostics. High-frequency range.

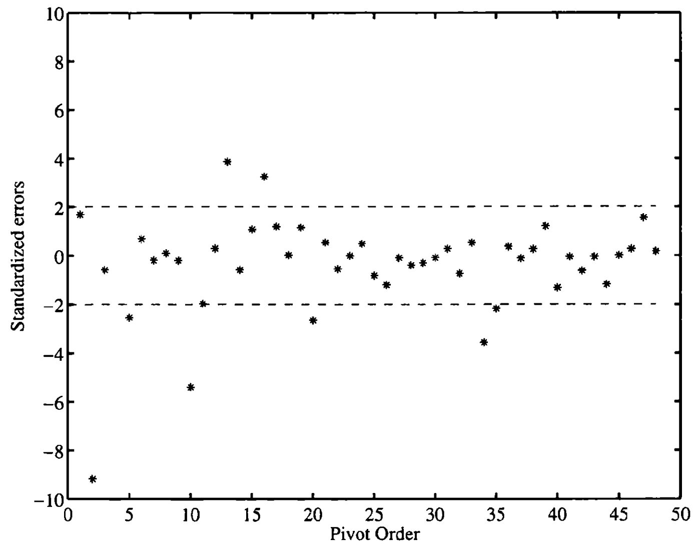


Figure 3.15: Pivoted Cholesky decomposition diagnostics. Low-frequency range.

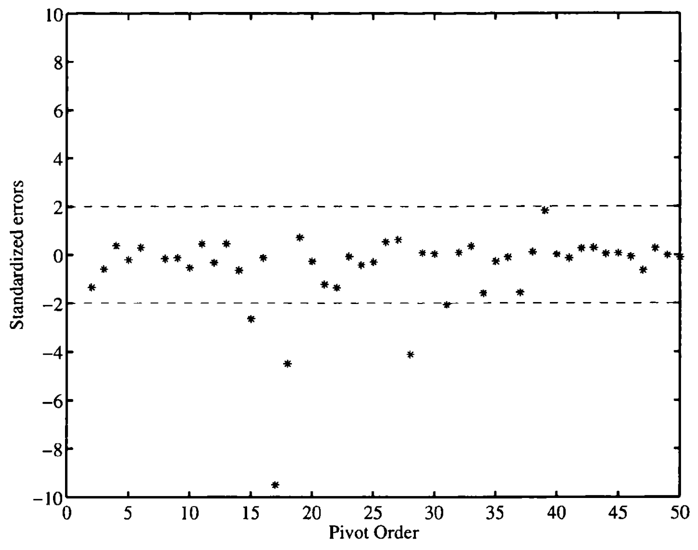


Figure 3.16: Pivoted Cholesky decomposition diagnostics. Medium-frequency range.

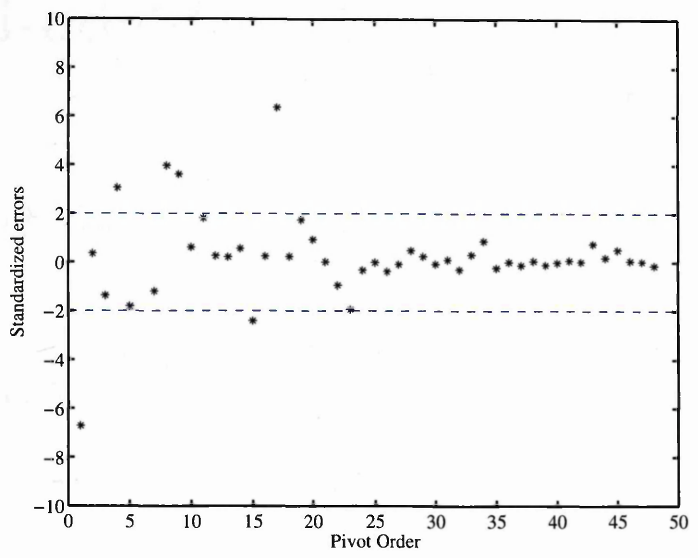


Figure 3.17: Pivoted Cholesky decomposition diagnostics. High-frequency range.

Chapter 4

Assimilation of Deterministic Multi-fidelity Finite Element Models

4.1 Introduction

The size of the finite element models has increased significantly over the past decades. For example, in the automotive and aerospace industries, models with millions of degrees of freedom are quite common nowadays. Such high-resolution models, combined with detailed physics can give good fidelity to experimental results. However, a potential disadvantage is that such large models may be computationally expensive. One alternative to address this problem is to use a low-resolution model. Although such low resolution models are often used during the design iteration, they are likely to be low-fidelity and may miss some crucial physics. The motivation of this chapter is therefore to investigate the possibility of improving the fidelity of a low-fidelity model without completely solving a detailed high-fidelity problem. A Bayesian approach that unifies GPEs and the domain decomposition method to solve the underlying boundary value problem of the finite element model is developed. Using this approach one can seamlessly assimilate a low-fidelity model with a more expensive high-fidelity model.

The Domain Decomposition Method (Schwarz, 1890; Glowinsky *et al.*, 1987; Glowinski and Wheeler, 1987; Bjørstad and Widlund, 1989; Babuška and Elman, 1989; Toselli and Windlund, 2005; Rao *et al.*, 2003; Gallimard and Sassi, 2010; Guibert and Tromeur-Dervout, 2007; Bendali *et al.*, 2007) is a divide-and-conquer algorithm aimed at solving partial differential equations (PDEs). With the develpment of more powerful computers and parallel architectures, it has become a common tool when solving expensive finite element models. Its main feature is that a linear system of discretised PDEs is recast as a set of smaller linear systems to be solved separately. A finite element model can thus be parallelized by partitioning the domain Ω in a number of subdomains. This allows an increase in the resolution of the model, along with a reduction in CPU requirements. There is, however, a potential disadvantage with this approach, since in order to obtain the finite element solution for each subdomain, the governing PDEs must be solved in the interface of each pair of subdomains. In this chapter, GPEs are used to approximate the solution to the interface problem.

4.2 Multi-fidelity Finite Element Modeling

Let Ω be a bounded domain in $\mathbb{R}^2$ with Lipschitz continuous boundary $\partial\Omega$, that is, there exist a finite number of covering open sets $\mathcal{O}_\ell$ such that, for every ℓ, $\partial\Omega \cap \mathcal{O}_\ell$ is the graph of a Lipschitz continuous function and $\Omega \cap \mathcal{O}_\ell$ lies on one side of this graph. The requirement of this type of continuity is important to control the smoothness of the boundary $\partial\Omega$. Let $\mathcal{T}^h$ be a family of conforming meshes (triangles) which are shape-regular as the mesh size $h \to 0$. Consider the elliptic PDE with the following Dirichlet boundary condition

$$\begin{aligned} -\nabla\big[\alpha(\mathbf{x})\nabla u(\mathbf{x})\big] + \beta(\mathbf{x})u(\mathbf{x}) &= f;\ \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= 0;\ \mathbf{x} \in \partial\Omega \end{aligned} \tag{4.1}$$

The Hilbert space $L^2(\Omega)$ and Sobolev space $H^k(\Omega)$ are respectively endowed with inner products $(u,v) = \int_\Omega u(\mathbf{x})v(\mathbf{x})d\mathbf{x}$ and $(u,v)_k = \int_\Omega u(\mathbf{x})v(\mathbf{x})d\mathbf{x} + \int_\Omega(\frac{du}{d\mathbf{x}})(\frac{dv}{d\mathbf{x}}) + \ldots + (\frac{d^k u}{d\mathbf{x}^k})(\frac{d^k v}{d\mathbf{x}^k})$. The aim is to obtain the function $u : \Omega \to \mathbb{R}$ which satisfies the conditions of problem (4.1) for a given $f : \Omega \to \mathbb{R}$.

As discussed in previous chapters, the standard finite element method can be applied in order to recast the PDE in problem (4.1) as

$$\mathbf{K}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \mathbf{f} \tag{4.2}$$

with $\mathbf{K} \in \mathbb{R}^{N\times N}$ and N the number of degrees of freedom in the underlying finite element mesh.

Suppose L_f and H_f are two finite element models to solve problem (4.1). Let $n_e^{(L_f)}$ and $n_e^{(H_f)}$ denote the number of elements in the finite element meshes of L_f and H_f induced by Eq. (4.2) and let $N^{(L_f)}$ and $N^{(H_f)}$ be the respective number of degrees of freedom. Also, let $h^{(L_f)}$ and $h^{(H_f)}$ be the respective element size. Finally, let $\mathbf{u}^{(r)}$ be a reference solution (e.g. experimental results) to (4.1). We call L_f a low-fidelity model and H_f a high-fidelity model if the following inequalities hold:

1. $\|\mathbf{u}^{(r)}(\mathbf{x}) - \mathbf{u}^{(L_f)}(\mathbf{x})\| > \|\mathbf{u}^{(r)}(\mathbf{x}) - \mathbf{u}^{(H_f)}(\mathbf{x})\|$ (Accuracy)
2. $h^{(L_f)} > h^{(H_f)}$ (Resolution)
3. $N^{(L_f)} < N^{(H_f)}$ (number of degrees of freedom)
4. $n_e^{(L_f)} < n_e^{(H_f)}$ (number of elements)

It is important to note that the concepts of low and high fidelity based on the above definition are relative. A simple refinement of a given low-fidelity mesh would imply a different increase in the fidelity of the model depending on the particular characteristics of the problem at hand. A more accurate description of L_f and H_f would therefore be "lower" and "higher" fidelity models respectively. Keeping this note in mind, the current low and high fidelity terminology is kept in the remainder of the chapter. Also, note that in the above definition it is implicitly assumed that both models L_f and H_f have same polynomial order p. A general version of the finite element method, known as *hp*-finite element method or *hp*-FEM (Babuška and Suri, 1990) would incorporate elements of variable size (h) and variable polynomial degree (p). This kind of method is beyond the scope of this dissertation. Note that a high-fidelity model can possess more features than simply more spatial resolution. For example, with higher resolution new geometrical details and more physics can be also added. The scope of multi-fidelity modeling therefore encompasses multi-scale and multi-physics modeling. Figure 4.1 shows two finite element models on a D-shaped domain, each with a different fidelity level.

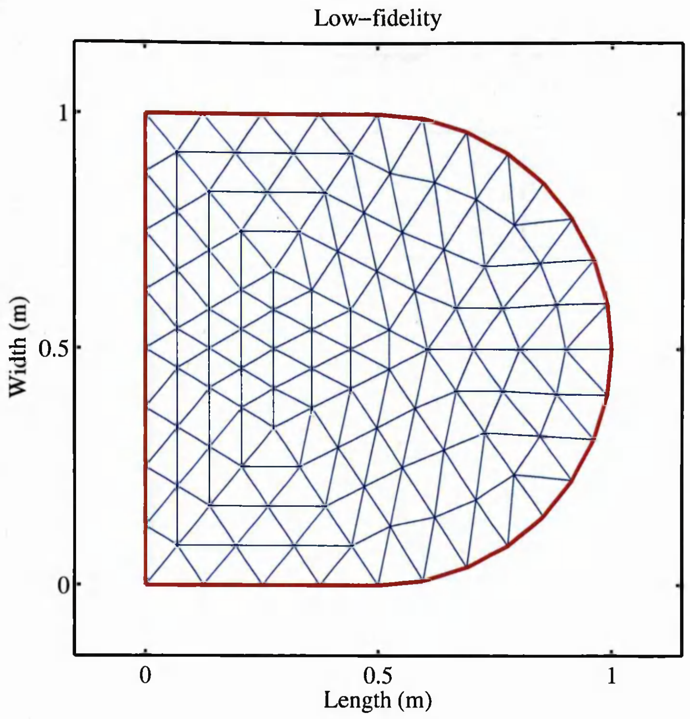
4.3 A Brief Overview of Domain Decomposition and Metamodeling

4.3.1 The Domain Decomposition Method

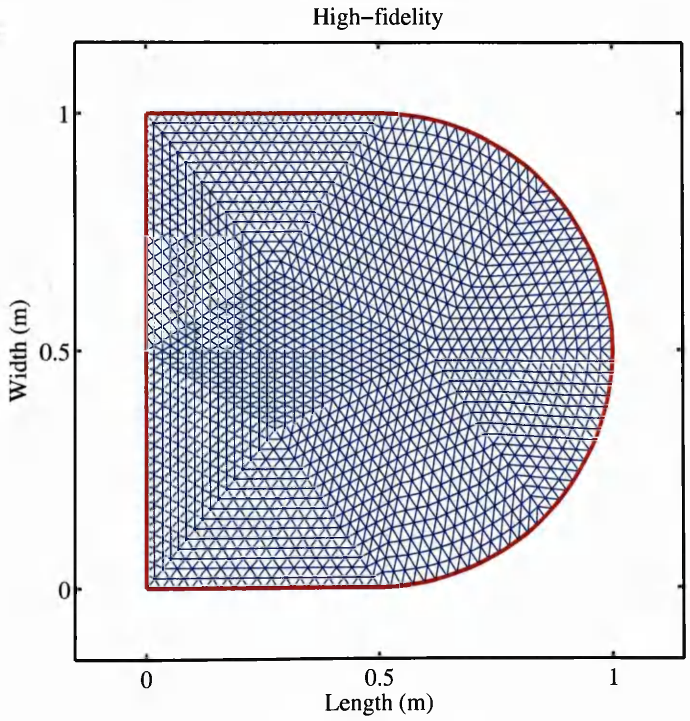
Let Ω be partitioned in $\mathcal{S}$ subdomains $\{\Omega_j : 1 \leq j \leq \mathcal{S}\}$, such that $\overline{\Omega} = \bigcup_j \overline{\Omega}_j$. Suppose these domains are non-overlapping, that is $\Omega_j \bigcap \Omega_k = \emptyset, \forall j \neq k$. The interface is denoted by Γ, where

$$\Gamma = \bigcup_{i,j} \left(\partial\Omega_i \cap \partial\Omega_j\right) \backslash \partial\Omega \tag{4.3}$$

In Figure 4.2, the D-shaped domain Ω from Figure 4.1 is partitioned into subdomains Ω_1 and Ω_2. The interface Γ separates both subdomains. Figure 4.3 shows the finite element mesh of the partitioned low-fidelity model L_f and the partitioned high-fidelity model H_f. In order to obtain the solution to these finite element models, it can be shown (Toselli and Windlund,



(a) Low-fidelity model L_f



(b) High-fidelity model H_f

Figure 4.1: Low and high-fidelity finite element models on the domain Ω.

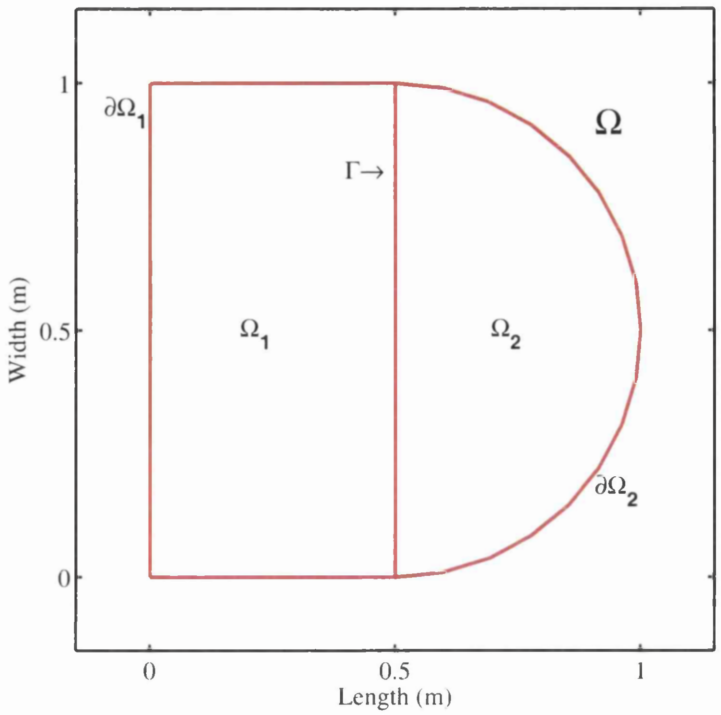


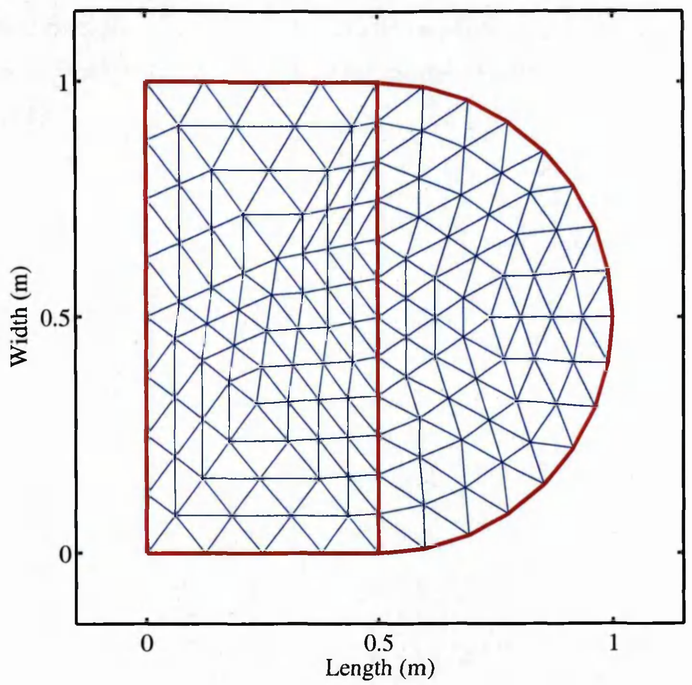
Figure 4.2: Domain decomposition of Ω into the subdomains Ω_1 and Ω_2. The borders of each subdomain are $\partial\Omega_1$ and $\partial\Omega_2$ respectively. The interface Γ separates the subdomains.

2005) that if Ω is the disjoint union of the subdomains $\Omega_1, \ldots, \Omega_S$, then the discretised PDEs governing the system's response can be recast as the following partitioned linear system

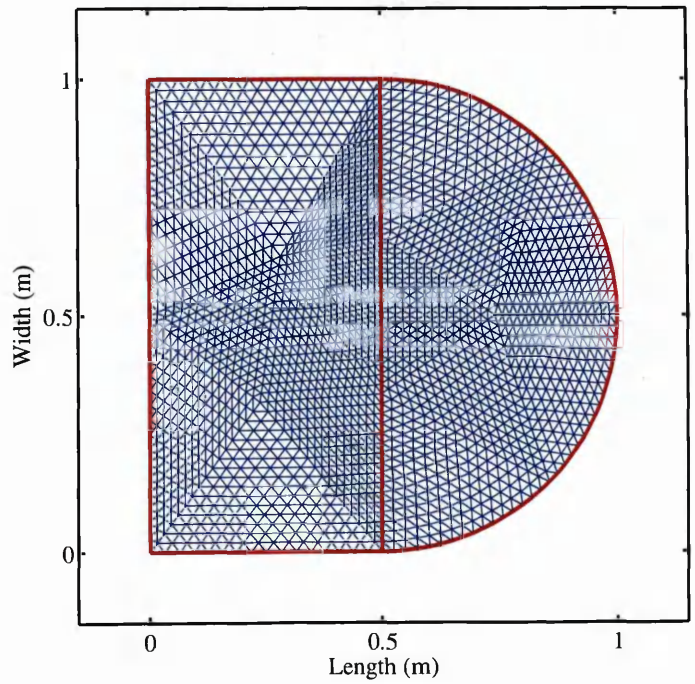
$$\begin{pmatrix} \mathcal{K}_1 & 0 & \ldots & 0 & B_1^\intercal \\ 0 & \mathcal{K}_2 & \ldots & 0 & B_2^\intercal \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \mathcal{K}_n & B_S^\intercal \\ B_1 & B_2 & \ldots & B_S & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ u_c \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_S \\ f_c \end{pmatrix} \tag{4.4}$$

The solution of the partitioned linear system (4.4) is obtained by solving the interface problem

$$\begin{aligned} &\left(C - B_1\mathcal{K}_1^{-1}B_1^\intercal - \ldots - B_S\mathcal{K}_S^{-1}B_S^\intercal\right) u_c \\ &= f_c - B_1\mathcal{K}_1^{-1}f_1 - \ldots - B_S\mathcal{K}_S^{-1}f_S \end{aligned} \tag{4.5}$$



(a) Low-fidelity model L_f



(b) High-fidelity model H_f

Figure 4.3: The finite element mesh of the low and high-fidelity finite element models on the domain Ω, which is decomposed into the subdomains Ω_1 and Ω_2.

and then solving in parallel

$$\begin{aligned} u_1 &= \mathcal{K}_1^{-1}(f_1 - B_1^{\mathsf{T}} u_c) \\ &\vdots \\ u_S &= \mathcal{K}_S^{-1}(f_S - B_S^{\mathsf{T}} u_c) \end{aligned} \tag{4.6}$$

Figure 4.4 shows the domain decomposition solution of problem (4.1) for H_f and for the values $\alpha(\mathbf{x}) = 1$, $\beta(\mathbf{x}) = 0$, and $f = 1$. The model represents the deformation of a membrane in the domain Ω.

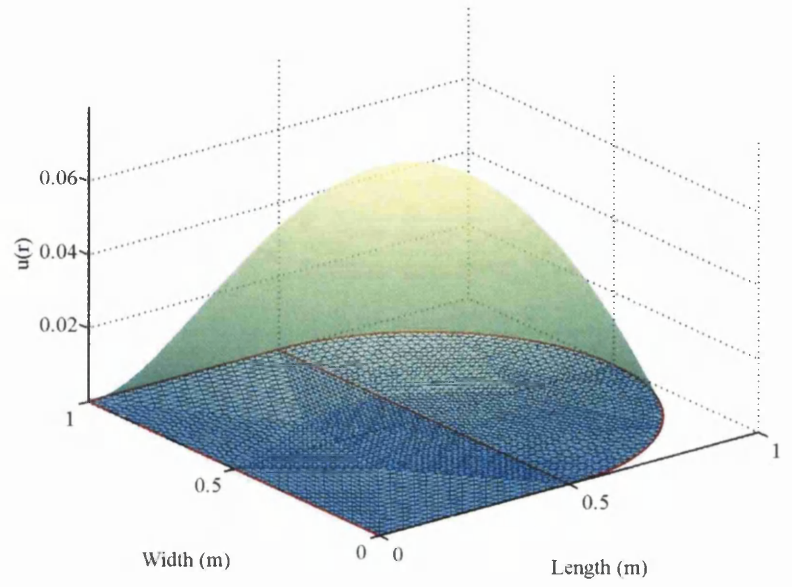


Figure 4.4: Domain decomposition solution of the high-fidelity finite element model H_f.

The main problem with solving a finite element model using domain decomposition is that the Schur complement matrix

$$\Sigma = C - B_1 \mathcal{K}_1^{-1} B_1^\intercal - \ldots - B_S \mathcal{K}_S^{-1} B_S^\intercal \tag{4.7}$$

is numerically expensive to obtain. Hence, solving the linear system (4.5) is likely to become a bottleneck of the domain decomposition strategy for a high-fidelity model H_f. A metamodeling approach based on GPEs is therefore proposed, whereby the solution to the interface problem (4.5) is approximated using only a few evaluations of a lower-fidelity model L_f.

4.3.2 Metamodeling Approach

Let $u^*(\mathbf{x}) : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ be a finite element solution to problem (4.1). If $\mathbf{x} = (x, y)$ and $u^*(\mathbf{x}) = z$, then u^* is a function that maps $(x, y) \mapsto z$. Adopting this notation, a level set of the domain Ω with respect to the x-axis is defined as

$$\mathcal{L}^x(c) = \Big\{ z \;\Big|\; u^*(c,y) = z\,; c \in \mathbb{R} \Big\} \tag{4.8}$$

and analogously for a level set with respect to the y-axis

$$\mathcal{L}^y(d) = \Big\{ z \;\Big|\; u^*(x,d) = z\,; d \in \mathbb{R} \Big\} \tag{4.9}$$

For $n_x, n_y \in \mathbb{Z}^+$, let Λ denote

$$\Lambda = \Big\{ \mathcal{L}^x(c_i), \mathcal{L}^y(d_j) \;\Big|\; 1 \le i \le n_x, 1 \le j \le n_y \Big\} \tag{4.10}$$

that is, the family of level sets with respect to x and with respect to y on Ω, determined by every number in $\mathcal{C} = \big\{ c_i, d_j \in \mathbb{R} \;\big|\; 1 \le i \le n_x, 1 \le j \le n_y \big\}$. Let $\Gamma^x(c_i)$ and $\Gamma^y(d_j)$ be the preimages of every $\mathcal{L}^x(c_i), \mathcal{L}^y(d_j) \in \Lambda$, that is

$$\Gamma^x(c_i) = \Big\{ y \in \Omega \;\Big|\; u^*(c_i, y) = z \Big\} \tag{4.11}$$

$$\Gamma^y(d_j) = \Big\{ x \in \Omega \;\Big|\; u^*(x, d_j) = z \Big\} \tag{4.12}$$

Notice that $u^*(\mathbf{x}) : \mathbb{R}^2 \longrightarrow \mathbb{R}$, whereas $\Gamma^x(c_i) \mapsto \mathcal{L}^x(c_i)$ and $\Gamma^y(d_j) \mapsto \mathcal{L}^y(d_j)$ are both mappings $\eta_i^x : \mathbb{R} \rightarrow \mathbb{R}$ and $\eta_j^y : \mathbb{R} \rightarrow \mathbb{R}$. Moreover, for every $c_i, d_j \in \mathcal{C}$, the set of points defined by $\bigcup_{ij} \Big\{ \mathcal{L}^x(c_i) \bigcup \mathcal{L}^y(d_j) \Big\}$ is the solution of the two-dimensional interface problem (4.5) whenever the interface between any two subdomains is parallel to any of the axis x and/or y. As mentioned before, this solution is the image of the mappings in $\mathcal{H} = \big\{ \eta_i^x(\cdot), \eta_j^y(\cdot) \;\big|\; 1 \le i \le n_x, 1 \le j \le n_y \big\}$. In order to reduce the computational cost of the interface problem, these mappings can be approximated using GPEs. The main idea is that the level sets defined by every interface problem in a low-fidelity finite element model can be emulated in order to approximate the interface problem in a corresponding high-fidelity finite element model.

Without loss of generality, suppose that n design points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are chosen in the input domain of any mapping $\eta(\cdot) \in \mathcal{H}$ (which will thus become the relevant simulator). Therefore, the set $\Big\{ \eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n) \Big\}$, will be regarded as the training set. Once the simulator $\eta(\cdot)$ is defined, the same principles used in Chapter 2 can be applied.

4.4 The Unification of Gaussian Process Emulation and Domain Decomposition

Let $u^*(\mathbf{x}) : \Omega \to \mathbb{R}$ be a finite element solution to the problem (4.1) for a low-fidelity finite element model L_f. Suppose that the solution $U^*(\mathbf{x}) : \Omega \to \mathbb{R}$ of a high-fidelity model H_f is to be obtained via domain decomposition. Assume the finite element mesh of H_f to be a refinement of the mesh of L_f. If a Delaunay triangulation is assumed for these meshes, it can be proved (George, 1991) that $\mathcal{N}(L_f) \subset \mathcal{N}(H_f)$, where $\mathcal{N}(L_f)$ and $\mathcal{N}(H_f)$ are the low and high-fidelity sets of nodes respectively. After being partitioned in subdomains, suppose the domain Ω contains n_x level sets with respect to the x-axis and n_y level sets with respect to the y-axis. As discussed in Section 4.3, a total of $n_x + n_y$ simulators of the form $\eta_i^x(\cdot) : \Gamma^x(c_i) \to \mathcal{L}^x(c_i)$ and $\eta_j^y(\cdot) : \Gamma^y(d_j) \to \mathcal{L}^y(d_j)$ can be specified for $u^*(\mathbf{x})$.

Notice that, since $\mathcal{N}(L_f) \subset \mathcal{N}(H_f)$, the sets $\mathcal{N}(H_f) \cap \Gamma^x(c_i)$ and $\mathcal{N}(H_f) \cap \Gamma^y(d_j)$ can be regarded as design points upon which GPEs can be built in order to infer the output of each simulator $\eta_i^x(\cdot)$ and $\eta_j^y(\cdot)$. That way, all the values of $U^*(\mathbf{x})$ that solve the interface problem for H_f would be approximated by the emulator.

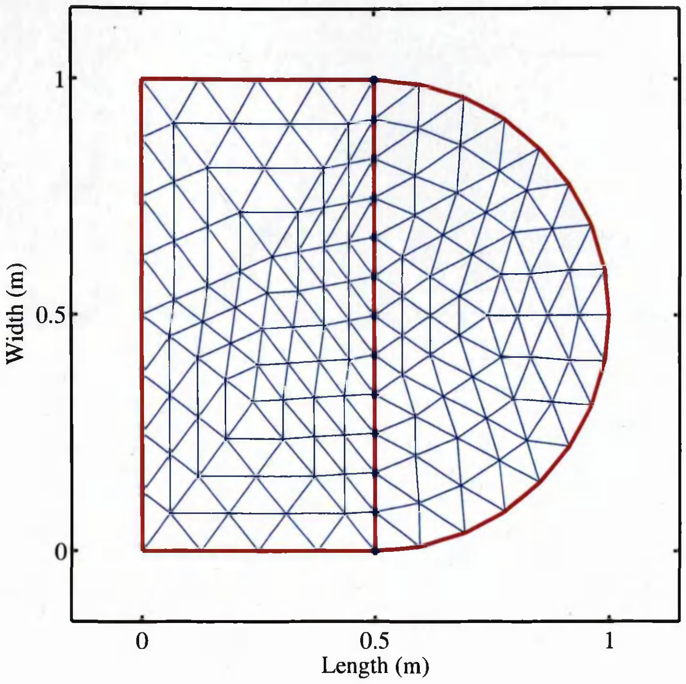
4.5 Numerical Examples

4.5.1 A Case of 2 Subdomains

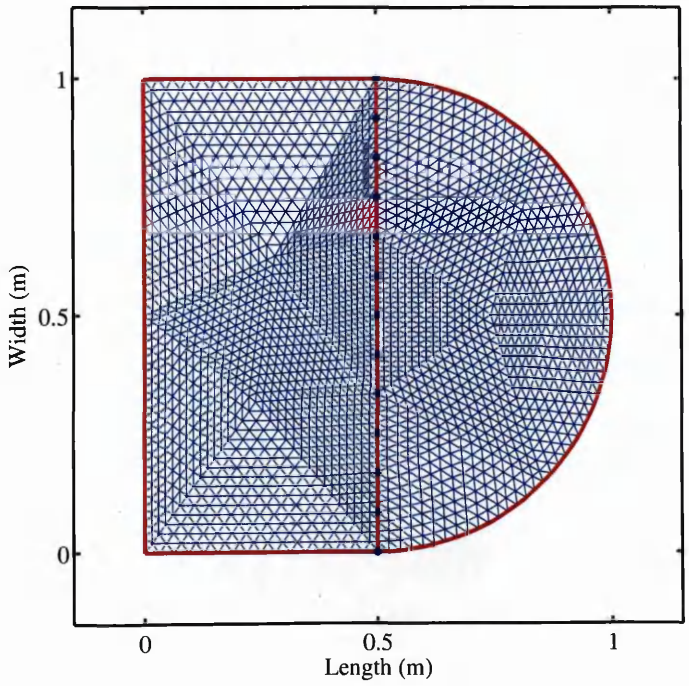
The model of the deforming membrane presented in Section 4.3 was considered again. The low-fidelity mesh consisted of 145 nodes, whereas a refinement yielded a high-fidelity mesh with 2113 nodes. The level set $\mathcal{L}^x(0.5)$ for L_f was determined by solving the interface problem, by means of Eq. (4.5). The set $\mathcal{N}(H_f) \cap \Gamma^x(0.5)$ determined the design points upon which to build the GPE to approximate the solution to the interface problem for H_f. Figure 4.5 illustrates how these design points are assimilated from L_f to H_f. Figure 4.6 shows the simulator defined by $\eta^x : \Gamma^x(0.5) \to \mathcal{L}^x(0.5)$. It also shows the approximation to $\mathcal{L}^x(0.5)$ provided by the predictive mean of the GPE, as well as probability bounds of 2 standard deviations about this mean.

Once the solution of the interface problem for H_f was emulated, the approximate solution for the subdomains Ω_1 and Ω_2 was obtained in parallel by solving Eqs. (4.6). Figure 4.7 shows the approximate solution to H_f for all the nodes in the high-fidelity mesh. The training runs obtained from the level set $\mathcal{L}^x(0.5)$, which were calculated by solving the low-fidelity model L_f, are also shown.

The computational advantage of the proposed approach can be appreciated by noting that 13 nodes lie in the interface of L_f. Thus, the cost of solving the interface problem for L_f was



(a) Low-fidelity model L_f



(b) High-fidelity model H_f

Figure 4.5: The finite element mesh of the low and high-fidelity finite element models on the domain Ω, which is decomposed into the subdomains Ω_1 and Ω_2. The design points on H_f are defined by the triangulation of L_f and therefore lie on the interface Γ.

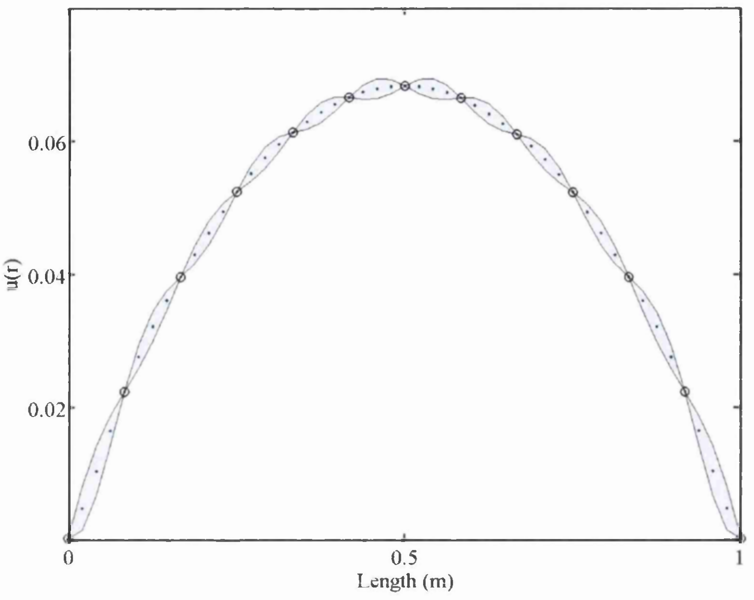


Figure 4.6: Approximation to the level set $\mathcal{L}^x(0.5)$ of H_f with the mean of a Gaussian process emulator and uncertainty (2 standard deviations) about this mean; (o): training runs, $(\cdots)$: emulator's predictive mean.

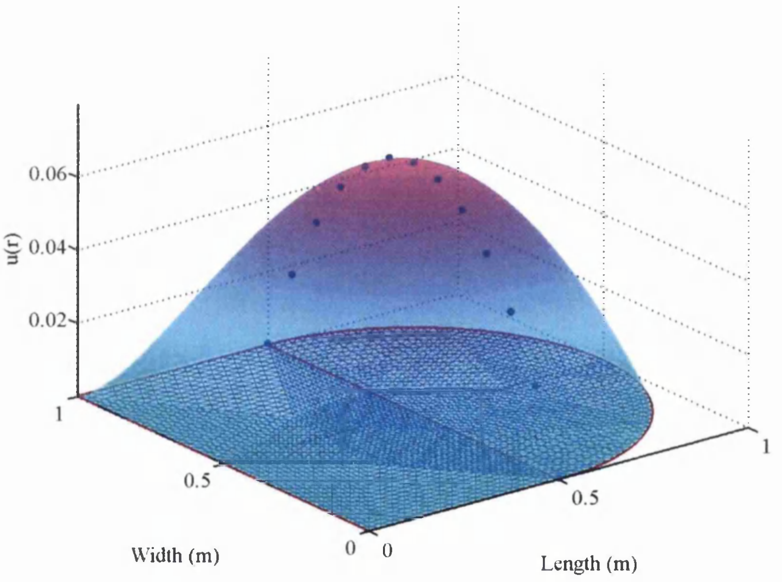


Figure 4.7: Solution of the high-fidelity finite element model H_f. The blue dots represent the training runs obtained from the level set $\mathcal{L}^x(0.5)$ from the low-fidelity model L_f. The interface problem for H_f was approximated by a GPE.

$\mathcal{O}(13^3)$ due to the solution of the Schur complement system in Eq. (4.5). The refinement of the Delaunay triangulation led to an increase of 4 times the number of nodes on the interface, which implied a cost of $\mathcal{O}(4^3 \times 13^3)$. In order to compare the approximate solution of H_f, denoted by $\widetilde{U}^*(\mathbf{x})$, with the domain decomposition solution $U^*(\mathbf{x})$ the absolute difference $|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})|$ was computed for every node. By definition, the maximum difference over all the nodes is the norm $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty$, which for this case had a magnitude of 8.12×10^{-4}. The contour defined by this comparison is shown in Figure 4.8. Note how, by construction, the difference in every design point is equal to 0. The same is true for the boundary, given the boundary conditions of problem (4.1).

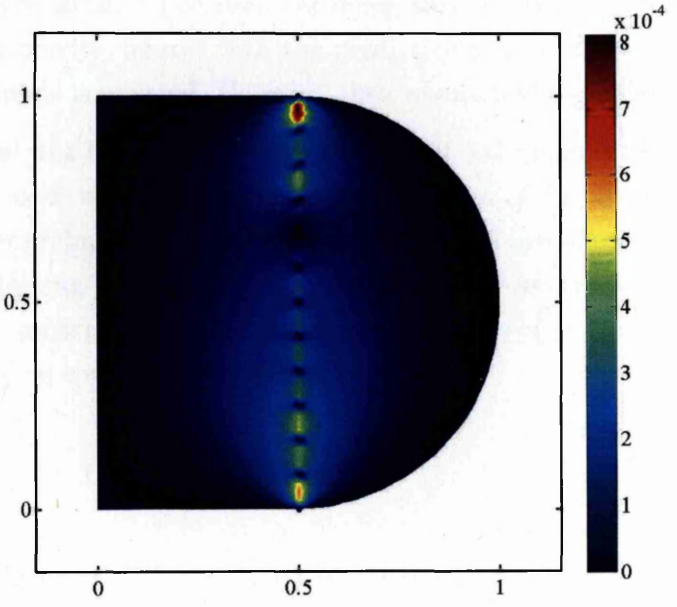


Figure 4.8: Absolute value of the difference between the domain decomposition solution of H_f, denoted by $U^*(\mathbf{x})$, and the solution with the proposed method, denoted by $\widetilde{U}^*(\mathbf{x})$. By definition, the maximum of this difference for all nodes is $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty \simeq 8.12 \times 10^{-4}$.

4.5.2 A Case of Two Non-convex Subomains

An interesting case arises when the domain Ω or any of its subdomains is not convex. As a reminder, a set Ω is convex if for any pair of points x and $y \in \Omega$, the point $\lambda x + (1-\lambda)y \in \Omega$ for all $\lambda \in [0,1]$. Consider the domain shown in Figure 4.9. Ω is clearly non-convex in $\mathbb{R}^2$, as it is trivial to construct a segment that connects two points belonging to the domain such that part of the segment does not belong to it. Due to this, the level set $\mathcal{L}^x(0.5)$ has a preimage $\Gamma^x(0.5)$ which is non-convex in $\mathbb{R}$, namely

$$\begin{aligned}\Gamma^x(0.5) &= \left\{y \in \Omega \;\middle|\; u^*(0.5, y) = z\right\} \\ &= \left\{y \in \Omega \;\middle|\; u^*(0.5, y) = z; 0 \le y \le 0.25\right\} \cup \left\{y \in \Omega \;\middle|\; u^*(0.5, y) = z; 0.75 \le y \le 1\right\} \\ &= \Gamma_1^x(0.5) \cup \Gamma_2^x(0.5) \end{aligned} \tag{4.13}$$

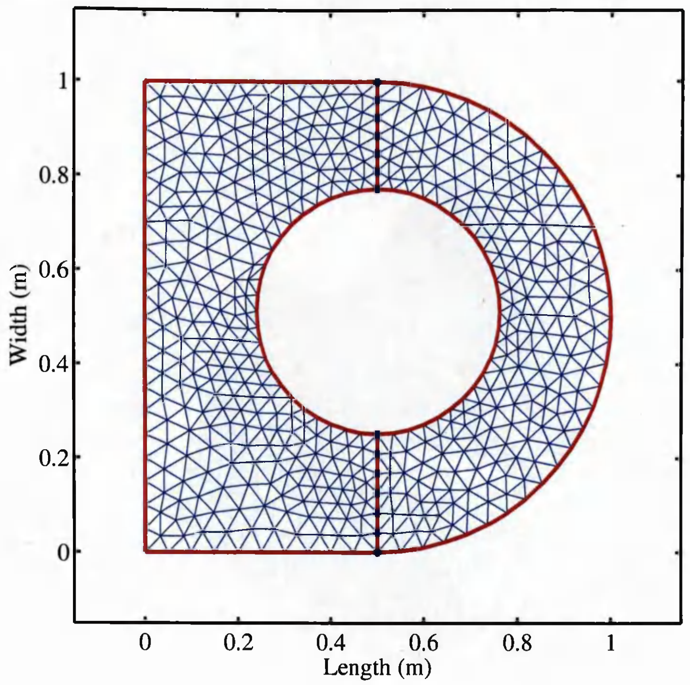
Hence, in order to approximate $\mathcal{L}^x(0.5)$, two GPEs were built, taking the respective design points from $\mathcal{N}(H_f) \cap \Gamma_1^x(0.5)$ and $\mathcal{N}(H_f) \cap \Gamma_2^x(0.5)$. The proposed method could thus be applied as previously. The result of doing this can be seen in Figure 4.10. Note that for the sake of brevity, figures with the predictive means of the emulators, as well as the uncertainty bounds is omitted. However, they resemble the parabolic curve in Figure 4.6.

For this case, the low-fidelity mesh consisted of 532 nodes, whereas a refinement yielded a high-fidelity mesh with 2008 nodes. The refinement of L_f into H_f yielded two nodes per design point per preimage, such that the computational cost of solving each interface problem for L_f and obtaining the training runs was $2 \times \mathcal{O}(7^3)$ as opposed to $2 \times \mathcal{O}(2^3 \times 7^3)$. The node-to-node comparison yielded a maximum difference of $||\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})||_\infty = 3.80 \times 10^{-5}$. The corresponding contour is shown in Figure 4.11.

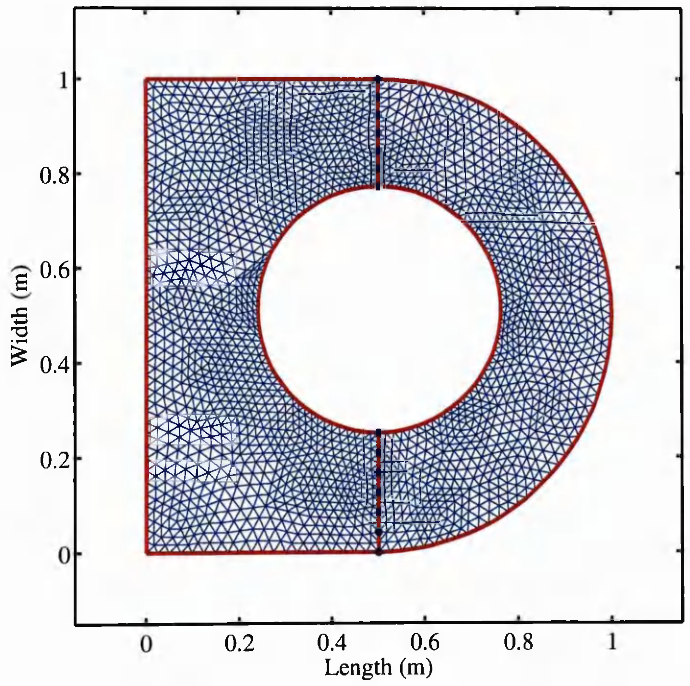
4.5.3 A Case of Three Subdomains

In this example, the proposed method is tested assuming a domain Ω that is partitioned into three subdomains. This partitioning determined two interfaces, whose intersection with the corresponding nodes produced two level sets, $\mathcal{L}^x(0.5)$ and $\mathcal{L}^y(0.5)$. The L-shaped domain Ω, as well as the low and high-fidelity meshes and the interfaces defining the design points are all depicted in Figure 4.12. In this example, the low-fidelity mesh consisted of 150 nodes, whereas a refinement yielded a high-fidelity mesh with 557 nodes.

The simulators $\eta^x : \Gamma^x(0.5) \longrightarrow \mathcal{L}^x(0.5)$ and $\eta^y : \Gamma^x(0.5) \longrightarrow \mathcal{L}^y(0.5)$ were emulated running two GPEs. The solution for the subdomains Ω_1, Ω_2, and Ω_3 was obtained in parallel following Eqs. (4.6). The solution is shown in Figure 4.13. Note that the refinement of L_f into H_f yielded again two nodes per design point on each interface, such that the computational cost of solving each interface problem for L_f and obtaining the training runs was $\mathcal{O}(8^3)$ as opposed to $\mathcal{O}(2^3 \times 8^3)$. Finally, The node-to-node comparison yielded a maximum difference of $||\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})||_\infty = 5.71 \times 10^{-3}$. The resulting contour is shown in Figure 4.14.



(a) Low-fidelity model L_f



(b) High-fidelity model H_f

Figure 4.9: The finite element mesh of the low and high-fidelity finite element models on the non-convex domain Ω, which is decomposed into the subdomains Ω_1 and Ω_2. The set of design points lies on a non-convex interface.

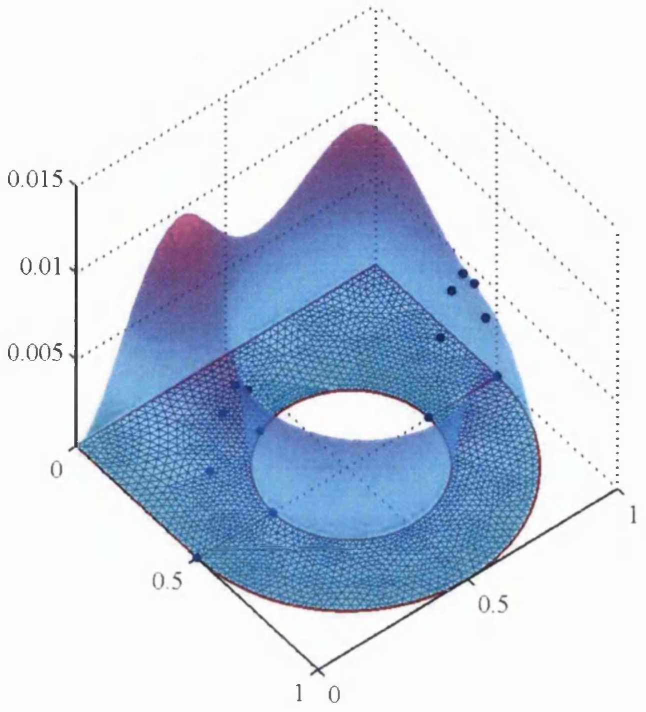


Figure 4.10: Solution of the high-fidelity finite element model H_f. The blue dots represent the training runs obtained from the level set $\mathcal{L}^x(0.5)$ from the low-fidelity model L_f. The interface problem for H_f was approximated by two GPEs, due to the non-convexity of the domain Ω.

4.6 Conclusions

In this chapter, a novel method based on GPEs to solve boundary value problems employing of domain decomposition was proposed. Given some conditions, the method can assimilate a low-fidelity finite element model with a computationally more expensive high-fidelity model. The computational cost of the domain decomposition solution of the high-fidelity model was reduced by solving the interface problem of the low-fidelity model. The main contribution of the chapter is the novel coupling between GPEs and domain decomposition methods to solve boundary value problems, reducing the computational cost and assimilating finite element models with different levels of fidelity.

This chapter concludes our study of GPEs for deterministic engineering systems. So far, we have proposed two methods for coupling GPEs with the finite element method. In the next chapters, the systems that will be analyzed will include parametric uncertainty, whereby the parameters of the system will be represented either by random variables or by random fields. As it will be seen, this will considerably increase the computational cost of

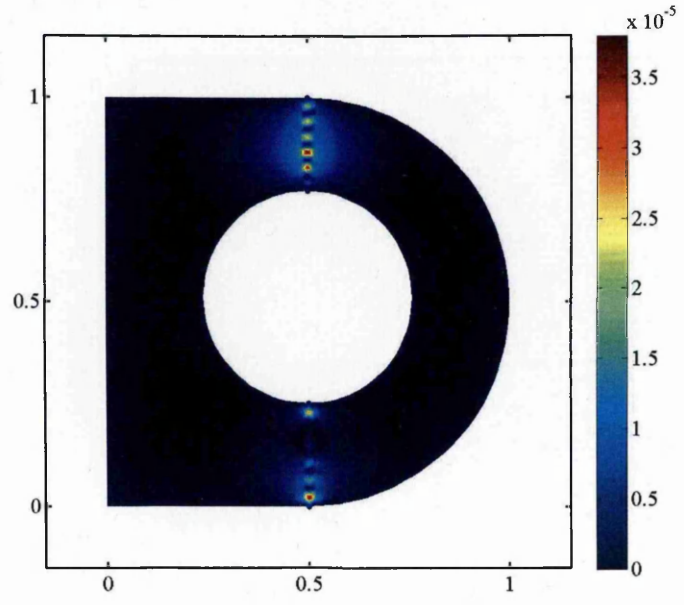
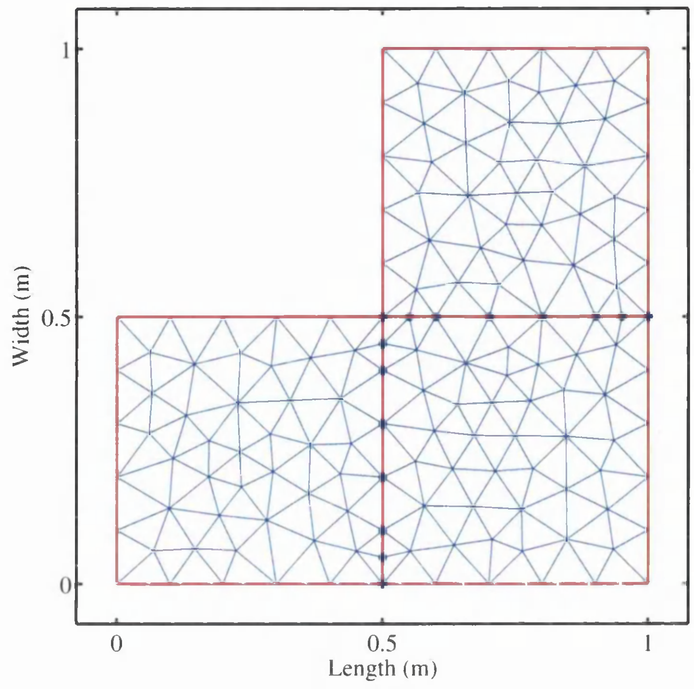
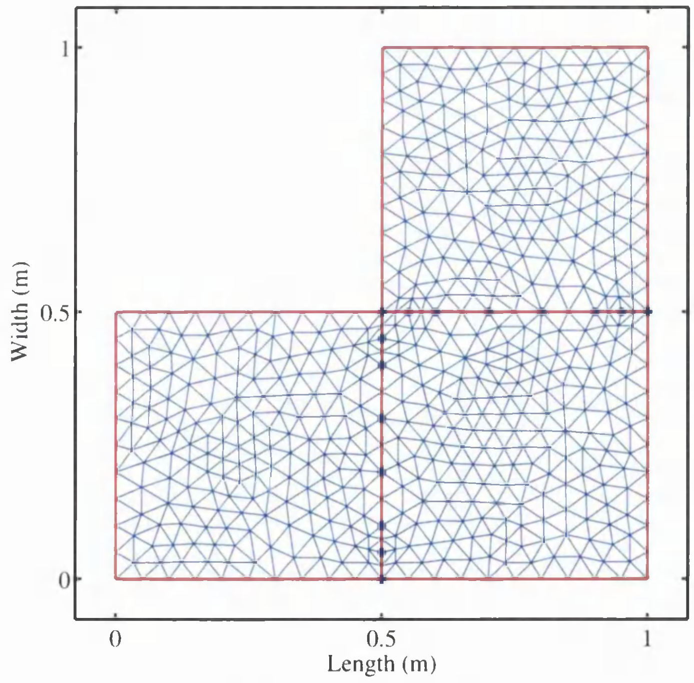


Figure 4.11: Absolute value of the difference between the domain decomposition solution of H_f, denoted by $U^*(\mathbf{x})$, and the solution with the proposed method, denoted by $\widetilde{U}^*(\mathbf{x})$. By definition, the maximum of this difference for all nodes is $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty \simeq 3.8 \times 10^{-5}$.

the simulators involved and the stochastic finite element methods available.



(a) Low-fidelity model L_f



(b) High-fidelity model H_f

Figure 4.12: The finite element mesh of a low fidelity model on an L-shaped domain Ω, decomposed in three subdomains Ω_1, Ω_2, and Ω_3, thus inducing two interface problems.

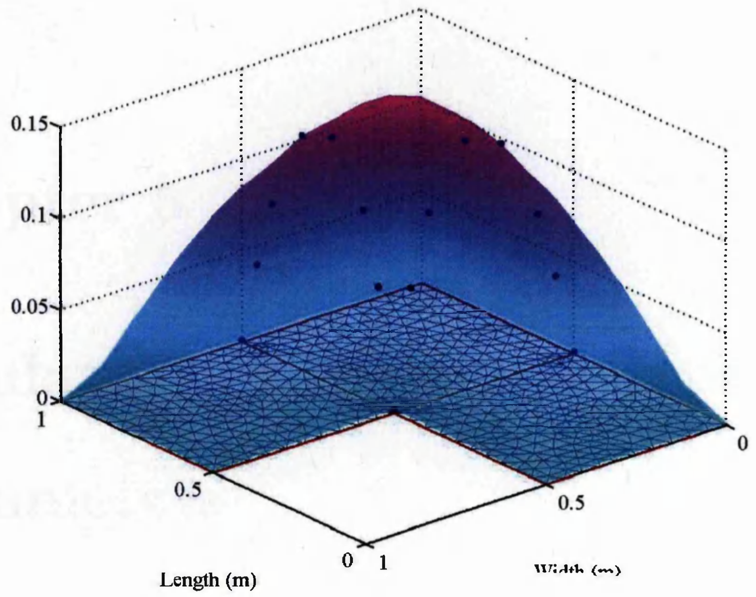


Figure 4.13: Solution of the high-fidelity finite element model H_f. The blue dots represent the training runs obtained from the level sets $\mathcal{L}^x(0.5)$ and $\mathcal{L}^y(0.5)$ from the low-fidelity model L_f. The interface problems for H_f was approximated by Gaussian process emulators.

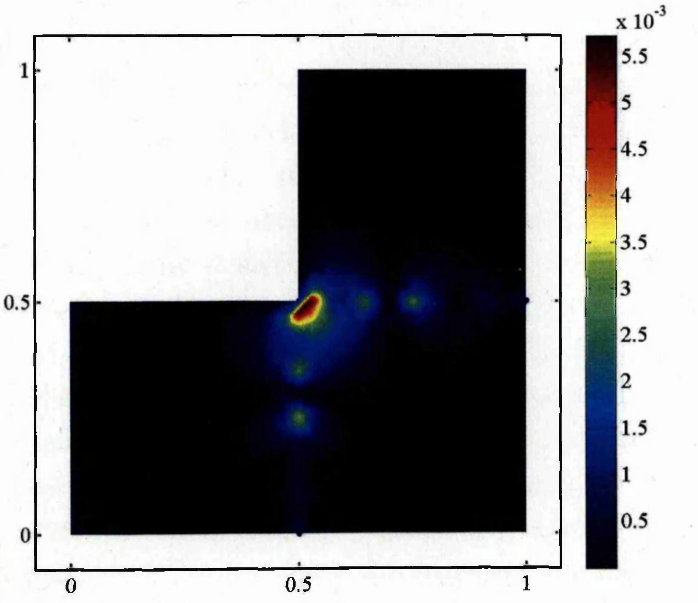


Figure 4.14: Absolute value of the difference between the domain decomposition solution of H_f, denoted by $U^*(\mathbf{x})$, and the solution with the proposed method, denoted by $\widetilde{U}^*(\mathbf{x})$. By definition, the maximum of this difference for all nodes is $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty \simeq 5.71 \times 10^{-3}$.

Chapter 5

Emulation of Systems with Random Parameters

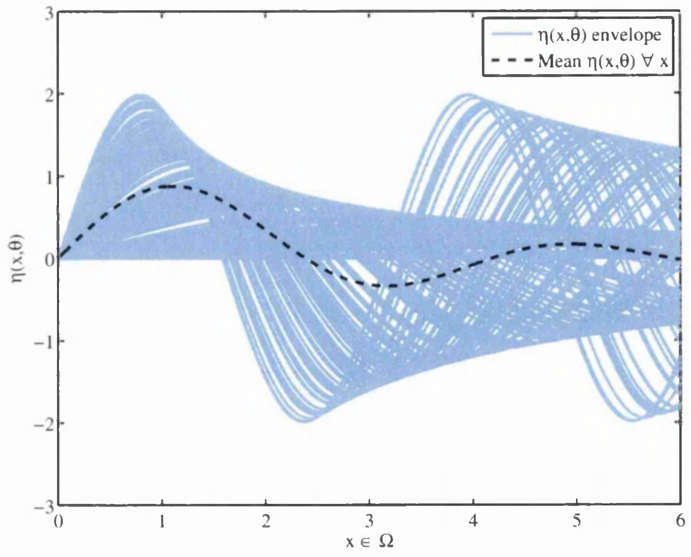
5.1 Introduction

In this chapter we begin our study of GPEs applied to engineering systems with parametric uncertainty by revisiting the structural dynamic analysis presented in Chapter 3 and extending it by including random parameters. The consideration of uncertainty in numerical models to obtain the probabilistic description of a system's response is becoming more desirable for industrial-scale finite element models. This is because uncertainty is practically unavoidable in the description of realistic physical systems. Randomness can be introduced due to uncertain material properties, uncertain boundary conditions or unknown manufacturing tolerances. Very large finite element models are used for complex engineering dynamical systems such as helicopters, automobiles and aircrafts. For such models, the consideration of uncertainty and a wide frequency-range of interest make the stochastic structural dynamics particularly challenging from the point of view of computational methods. Stochastic finite element based methods have been proposed for uncertainty propagation in static or low-frequency vibration problems. For dynamic problems, perturbation based approaches utilizing the random eigenvalue problems (Scheidt and Purkert, 1983; Adhikari and Friswell, 2007) have been used for the forced vibration response of linear dynamical systems. These methods are however, normally restricted to low level of random variation in the parameters. This problem can be avoided if a simulation based approach is adopted. The disadvantage is that direct Monte Carlo simulation based approaches are computationally expensive. This context is therefore suitable for the inclusion of GPEs as a tool for reducing the computational cost involved.

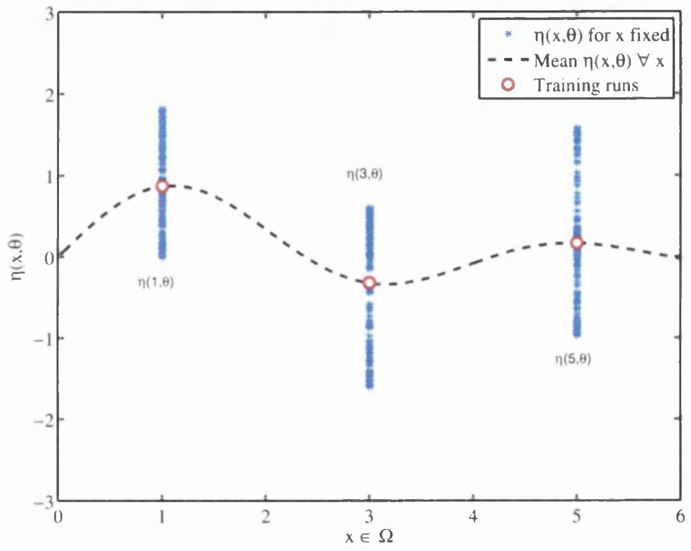
5.2 Preliminary Ideas

As mentioned in Chapter 2, emulation makes it possible to obtain a statistical approximation to the output of a simulator after evaluating a small number of design points in its domain, hence reducing the required computer processing time. However, there are cases when even obtaining the training runs can be computationally demanding. Consider a complex physical process that involves a random parameter θ. Let this process be investigated by running an expensive simulator $\eta : \Omega \times \Theta \to \mathbb{R}^d$, where Ω contains only non-random parameters (spatial coordinates, units of time, frequency levels as in Chapter 3, etc.) and Θ is the set of possible realizations of θ. Note that in this case the term "non-random" is preferred over "deterministic", since the distinction between random and non-random parameters is artificial: when implemented numerically, a set of realizations of θ is a sequence of pseudo-random numbers; for a fixed random seed such a sequence is deterministic. Suppose that a statistic of the output of $\eta(\cdot)$, such as the mean or the variance, is to be obtained. Since running $\eta(\cdot)$ is assumed to be expensive, a GPE can be employed to approximate the desired statistic. Note however that if the training runs upon which the emulator is built are based on design points chosen from Ω, then the task may easily become burdensome. That is because in order to obtain even a few instances of the desired statistic it would be necessary to evaluate $\eta(\cdot)$ at each design point repeatedly for a potentially large number of realizations of θ, say S. An example of this is shown in Figure 5.1, where $\eta(\mathbf{x}, \theta) = \theta sin(\theta \mathbf{x})$, $\mathbf{x} \in \Omega = \{0, 0.01, \ldots, 5.99, 6\}$, θ is uniformly distributed in $\Theta = [0, 2]$, and $S = 150$. Figure 5.1(a) shows the corresponding 150 realizations of $\eta(\cdot)$, as well as the mean output (the dotted line) for all $\mathbf{x} \in \Omega$. The direct calculation of the mean output would involve 150×601 evaluations of $\eta(\cdot)$. Suppose that in order to reduce the cost of calculating the mean output at every $\mathbf{x}$, 100 design points $\{\mathbf{x}_i\}_{i=1}^{100}$ are chosen, amongst which are $\mathbf{x}_{i_1} = 1, \mathbf{x}_{i_2} = 3, \mathbf{x}_{i_3} = 5$. Figure 5.1(b) shows the corresponding three mean output values (the circles), which would be part of the 100 training runs upon which a GPE would be built in order to approximate the mean output (the dotted line) for the remaining $\mathbf{x}$'s. In this case, $\eta(\cdot)$ would have to be evaluated 150×100 times, representing 16.64% of the original computational effort. Suppose that even this number of evaluations is still more than what can be afforded, so that less design points are chosen. As has been in seen in previous chapters, this reduces the accuracy of the approximation.

The aim of this chapter is to explore an alternative design point generation scheme for cases analogous to the one above, where not only does a computer model is expensive but also the approximation of its output via a GPE is hindered by the cost of generating the corresponding training runs. The proposed scheme is based on observing that the structure of the domain $\Omega \times \Theta$ offers the possibility of sampling the design points from Θ, whilst fixing the $\mathbf{x}$'s in Ω. That is, if the output of $\eta(\cdot)$ is treated as a function of θ only, then it is



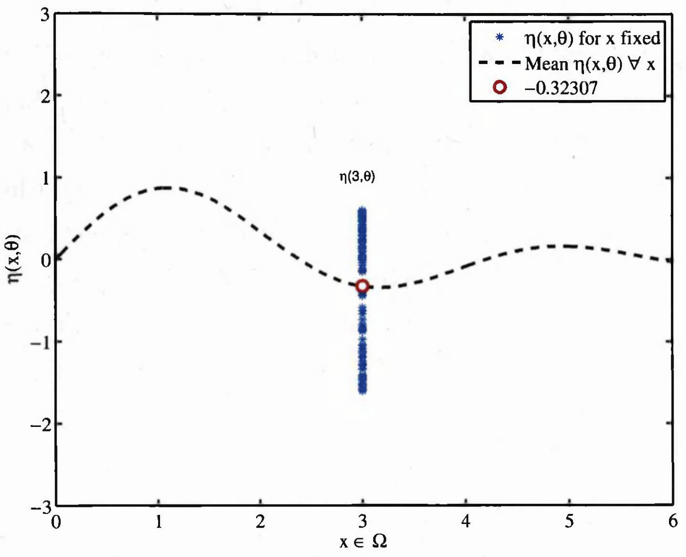
(a) 150 realizations of $\eta(\mathbf{x}, \theta)$ and mean output



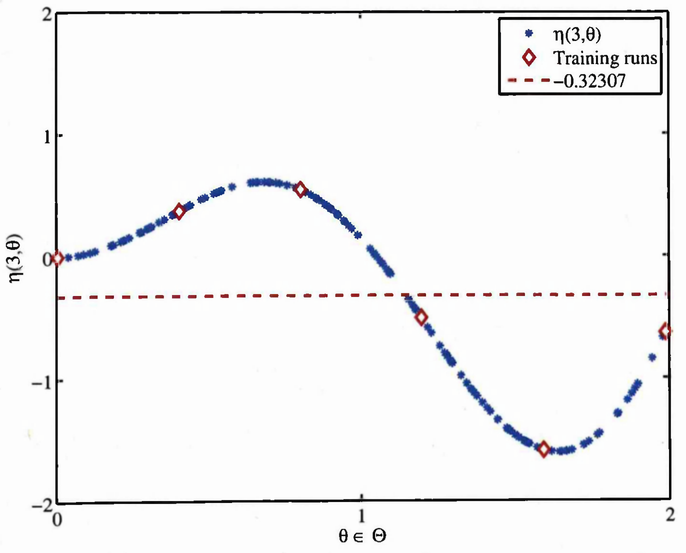
(b) Mean output of $\eta(\mathbf{x}, \theta)$ for $\mathbf{x} = 1, 3, 5$

Figure 5.1: 150 realizations of $\eta(\mathbf{x}, \theta) = \theta sin(\theta \mathbf{x})$ and mean output; $\mathbf{x} \in \Omega = \{0, 0.01, \ldots, 5.99, 6\}$, $\theta \sim U[0, 2]$.

possible to build an emulator at every point in Ω and approximate the statistic of interest using design points taken from Θ. If the number of selected design points per emulator is less than S, then the total number of evaluations of $\eta(\cdot)$ is reduced. Figure 5.2 illustrates this idea. For $\mathbf{x} = 3$ fixed, the mean of 150 realizations of $\eta(3, \theta)$ is -0.32307, as shown in Figure 5.2(a). On the other hand, Figure 5.2(b) shows the same 150 realizations and their mean (the dotted straight line), this time with θ as the abscissas. Consider choosing (say) 6 design points from Θ (the diamonds). Using these training runs, a GPE could approximate the remaining 144 realizations of $\eta(3, \theta)$ and the mean of the resulting 150 values would in turn be used to approximate the mean output at $\mathbf{x} = 3$. If the same procedure is followed for all the $\mathbf{x}$'s in Ω, then $\eta(\cdot)$ would be evaluated 6×601 times. This represents 24.04% of the effort in the approach above and only 4% of the original computational effort. Additionally, if the time taken to emulate the statistic for every point in Ω is reasonably short, there might be an improvement in the overall computer processing time required.



(a) 150 realizations of $\eta(3, \theta)$ and mean output



(b) Output of $\eta(3, \theta)$ as a function of θ

Figure 5.2: 150 realizations of $\eta(3, \theta)$ and mean output as functions of both $\mathbf{x}$ and θ.

5.3 Structural Dynamics with Parametric Uncertainty

Refer to the spring-mass system shown in Chapter 3. A more credible model would incorporate uncertainty in the equation that governs the system's response. Suppose that any of the matrices in Eq. (3.3) is a random matrix, that is, it includes a random parameter $\theta \in \Theta$. Without loss of generality, if the stiffness matrix $\mathbf{K}$ is random, then the relevant simulator is

$$\eta(\omega, \theta) = \left|[-\omega^2\mathbf{M} + \mathrm{i}\omega\mathbf{C} + \mathbf{K}(\theta)]^{-1}\bar{\mathbf{f}}(\omega)\right| \tag{5.1}$$

The domain of Eq. (5.1) is $\Omega \times \Theta$, where Ω is the set of admissible frequency levels and Θ is the set of realizations of the random parameter θ. Note that for a fixed random seed, the simulator in Eq. (5.1) is deterministic, as it produces the same (scalar) output given the same stream of pseudo-random numbers. For notational convenience, the random seed does not appear explicitly as a parameter of the simulator.

Consider a dynamical system whose FRF is simulated by Eq. (5.1). Suppose that a statistic of the FRF, such as the mean or the variance, is to be estimated. Let $||\Omega|| = N$ and $\ell \in \{1, \ldots, N\}$. Also, let $\{\theta_s^\ell\}_{s=1}^S = \{\theta_1^\ell, \ldots, \theta_S^\ell\}$ be a set of realizations of θ associated to a fixed ω_ℓ in $\Omega = \{\omega_\ell\}_{\ell=1}^N$. Then, at every $\{\omega_\ell\}_{\ell=1}^N$ the mean FRF is estimated by

$$\widehat{\eta}_1(\omega_\ell, \theta_1^\ell, \ldots, \theta_S^\ell) = \frac{1}{S}\sum_{s=1}^{S} \eta(\omega_\ell, \theta_s^\ell) \tag{5.2}$$

whereas the variance of the FRF is estimated by

$$\widehat{\eta}_2(\omega_\ell, \theta_1^\ell, \ldots, \theta_S^\ell) = \frac{S}{S-1}\left[\frac{1}{S}\sum_{s=1}^{S} \eta^2(\omega_\ell, \theta_s^\ell) - [\frac{1}{S}\sum_{s=1}^{S} \eta(\omega_\ell, \theta_s^\ell)]^2\right] \tag{5.3}$$

A possible approach is Monte Carlo simulation (MCS): For $\ell = 1, \ldots, N$, simulate $\{\eta(\omega_\ell, \theta_s^\ell)\}_{s=1}^S$ and compute Eq. (5.2) or Eq. (5.3). The number of evaluations of $\eta(\cdot)$ would be $S \cdot N$. Nevertheless, if $\eta(\cdot)$ is computationally expensive, MCS can quickly become impracticable. Two strategies to cope with this problem are discussed below.

5.3.1 Strategy 1

This approach operates by sampling design points from Ω. The principle is to apply MCS to a small set of design points and use that information as the training runs necessary to emulate the statistics of interest for the remaining ω's in Ω. To do this, select n design points $\{\omega_i^*\}_{i=1}^n$, where $n \ll N$. For $i = 1, \ldots, n$, simulate $\{\eta(\omega_i^*, \theta_s^i)\}_{s=1}^S$ and compute Eq. (5.2) or Eq. (5.3). Use the resulting training runs to emulate the $N - n$ statistics corresponding to all the untried ω's in Ω. The number of evaluations of $\eta(\cdot)$ would be reduced compared to MCS, since the assumption $n \ll N \Rightarrow S \cdot n \ll S \cdot N$.

5.3.2 Strategy 2

Observe however that given the structure of $\Omega \times \Theta$, it is possible to sample design points from Θ rather than Ω. That way, the number of evaluations of $\eta(\cdot)$ can be further reduced. This approach operates by sampling design points from Θ whilst keeping the parameters in Ω fixed. For $\ell = 1, \ldots, N$, generate L_ℓ design points $\{\theta_j^\ell\}_{j=1}^{L_\ell}$. Then, compute the training runs $\{\theta_j^\ell, \eta(\omega_\ell, \theta_j^\ell)\}_{j=1}^{L_\ell}$ and emulate N_{E} runs $\{\eta(\omega_\ell, \theta_k^\ell)\}_{k=1}^{N_{\mathrm{E}}}$ at the untried points $\{\theta_k^\ell\}_{k=1}^{N_{\mathrm{E}}} = \{\theta_1^\ell, \ldots, \theta_{N_{\mathrm{E}}}^\ell\}$ in order to calculate the mean

$$\widetilde{\eta}_1(\omega_\ell, \theta_1^\ell, \ldots, \theta_{N_{\mathrm{E}}}^\ell) = \frac{1}{N_{\mathrm{E}}} \sum_{k=1}^{N_{\mathrm{E}}} \eta(\omega_\ell, \theta_k^\ell) \tag{5.4}$$

or the variance

$$\widetilde{\eta}_2(\omega_\ell, \theta_1^\ell, \ldots, \theta_{N_{\mathrm{E}}}^\ell) = \frac{N_{\mathrm{E}}}{N_{\mathrm{E}}-1} \left[\frac{1}{N_{\mathrm{E}}} \sum_{k=1}^{N_{\mathrm{E}}} \eta^2(\omega_\ell, \theta_k^\ell) - [\frac{1}{N_{\mathrm{E}}} \sum_{k=1}^{N_{\mathrm{E}}} \eta(\omega_\ell, \theta_k^\ell)]^2 \right] \tag{5.5}$$

The number of evaluations of $\eta(\cdot)$ would be $L = \sum_{\ell=1}^{N} L_\ell$. Note that if $L \ll S$, the computational burden should also be reduced compared to Monte Carlo simulation, since $L \ll S \Rightarrow L \ll S \cdot N$.

5.3.3 Computational Complexity

Both strategies above involve less evaluations of $\eta(\cdot)$ than MCS. However, the interest lies in determining the conditions under which strategy 2 is less expensive than strategy 1. Clearly, the computational effort of each strategy is divided in two: first in simulating (sampling design points either from Ω or Θ and evaluating $\eta(\cdot)$) and second in emulating. Both strategies take less time in emulating than in simulating. Then, for strategy 2 to outperform strategy 1 it has to be true that

$$L < S \cdot n \tag{5.6}$$

However, this condition is trivially met for $n > 1$, since by assumption $L \ll S$.

5.4 Numerical Investigation: A Damped Cantilever Plate

In order to compare the strategies outlined in Section 5.3, consider the finite element model of the rectangular steel plate used in Chapter 3. It is well established that, for a finite element model with n_e elements, the global stiffness matrix $\mathbf{K}$ can be obtained from the element stiffness matrices $\mathbf{K}^{(1)}, \dots, \mathbf{K}^{(n_e)}$ using the direct stiffness method (Bathe, 1995). Let E denote Young's modulus, the measure of stiffness in the material of the plate. Assume (for now) that E is the same for every element. Then

$$\mathbf{K} = E \sum_{e=1}^{n_e} \widetilde{\mathbf{K}}^{(e)} \tag{5.7}$$

Where in order to perform the summation, each element stiffness matrix $\mathbf{K}^{(e)}$ is written as a matrix $\widetilde{\mathbf{K}}^{(e)}$ of the same dimensions as $\mathbf{K}$. All the entries in $\widetilde{\mathbf{K}}^{(e)}$ are zero except those which correspond to an element degree of freedom. Analogously, for Eq. (5.1), if Young's modulus is random, but the same for every element, then

$$\mathbf{K}(\theta) = \theta \sum_{e=1}^{n_e} \widetilde{\mathbf{K}}^{(e)} \tag{5.8}$$

A number of sample FRFs of the plate were computed with the same finite element simulator described in Chapter 3. For every run of the simulator, the random Young's modulus was assumed to be of the form

$$\theta = E_0 \cdot (1 + \epsilon\, \varphi) \tag{5.9}$$

where E_0 is an initial value, $\epsilon \in \mathbb{R}^+$ with $\epsilon \ll 1$, and φ is a (pseudo) random number such that $\varphi \sim \mathcal{N}(0, 1)$. Hence, $\theta \sim \mathcal{N}(E_0, [\epsilon E_0]^2)$.

A preliminary study was carried out by running a series of 10 experiments, each with $S = 100$ Monte Carlo simulations of $\eta(\cdot)$. One of these runs is shown in Figure 5.3. The finite element model employed had 25×15 elements, $\epsilon = 0.1$, and $E_0 = 2.0 \times 10^{11}$ Pascals (Pa) (the typical value for steel). Notice that 99% of the values of θ must lie within 3 standard deviations from E_0, namely within the interval $[1.4 \times 10^{11}, 2.6 \times 10^{11}]$ and thus negative, nonphysical values of θ are unlikely. The frequency domain Ω ranges from 1 to 100 Hz. The resolution is 1 Hz, therefore $|\Omega| = N = 100$. The preliminary study confirmed that near the resonance frequencies, the system is more sensitive to parametric uncertainty. Due to this effect, the frequency $\omega = 64$ was identified to consistently exhibit the widest spread between its maximum and its minimum frequency response. This was the case for every one of the 10 repetitions of the experiment. Consequently, for $\omega = 64$ fixed, the mean and variance of the FRF was computed for sample sizes S ranging from 1 to 2000. As it can be seen in Figure 5.4, both the mean and the variance seem to stabilize around a sample size of 1000. Therefore, that was the value chosen for S.

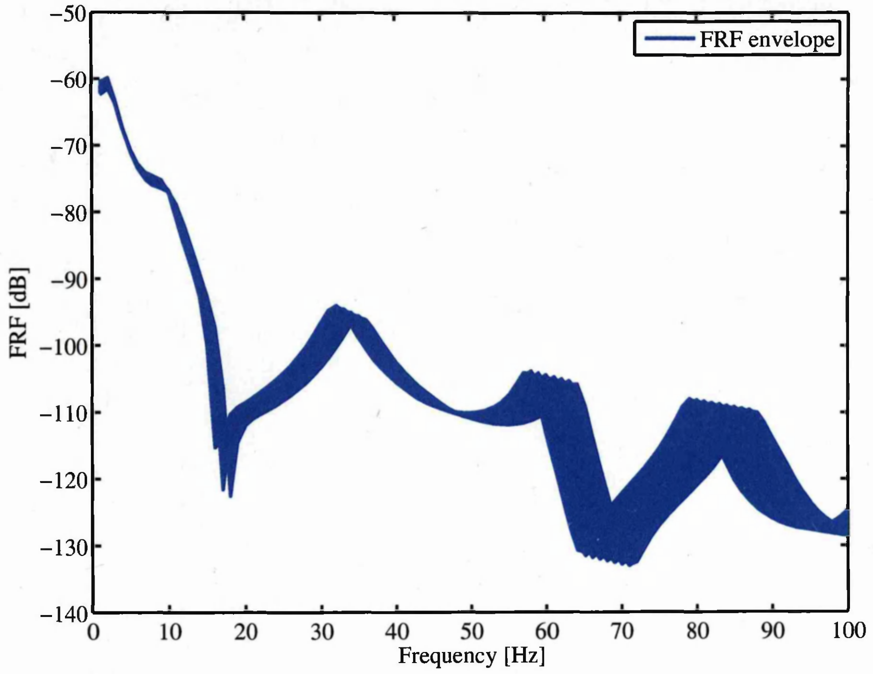
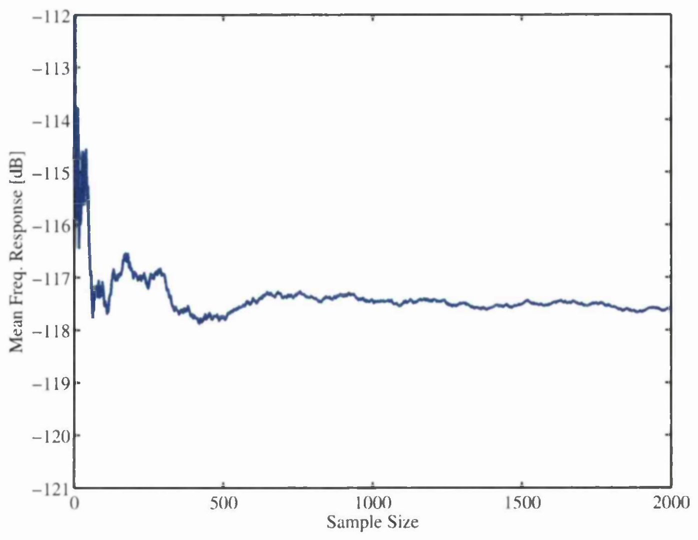


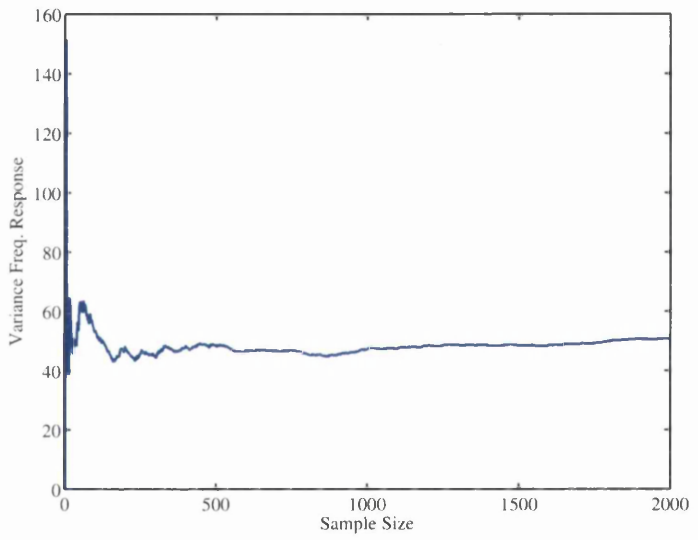
Figure 5.3: The envelope resulting from S = 100 Monte Carlo simulations of $\eta(\cdot)$, with 25 $\times$ 15 elements, $E_0 = 2.0 \times 10^{11}$ Pa, and $\epsilon = 0.1$

The Monte Carlo mean and variance of the FRF at every $\omega \in \Omega$ were taken as the benchmark against which the accuracy of strategies 1 and 2 was compared. The random seed was fixed, such that for the same design points and the same number of simulations, the statistic of the FRF remained constant. Note that it was the output of the simulator, not the value of the quantity the simulator is attempting to predict, what was taken into account to determine the accuracy of both strategies.

Strategy 1 was implemented sampling $n = N/10 = 10$ equally spaced design points from Ω. For each design point, the mean and the variance of the FRF was computed as in Eq. (5.2) and Eq. (5.3). A GPE was built upon the set of training runs. With no prior knowledge about the functional form of the mean or the variance of the FRF, $\mathbf{h}(\cdot) = 1$ was assumed in both cases. The smoothness parameters were obtained with the maximum likelihood method outlined in Chapter 2. Figure 5.5 shows the comparison between the emulated mean and the MCS benchmark. Figure 5.6 shows the comparison between the emulated variance and



(a) Mean FRF



(b) FRF Variance

Figure 5.4: Mean and variance of the frequency response function (FRF) at $\omega = 64$ for different sample sizes. Both statistics seem to stabilize for sample sizes greater than or equal to 500.

the MCS benchmark.

Previous to implementing strategy 2, the set $\{L_\ell\}_{\ell=1}^{100}$, containing the number of design points for each of the 100 emulators corresponding to every frequency level had to be determined. Since for each $\{\omega_\ell\}_{\ell=1}^{100}$ fixed the functional form of $\eta(\cdot)$ is unknown, an exploration of Θ to determine every L_ℓ would have been very costly. Instead, it was decided to take L_ℓ equal for every ω_ℓ. Ten design points for each emulator where generated using Latin hypercube sampling from the distribution of θ. As in strategy 1, $\mathbf{h}(\cdot) = 1$ was assumed for

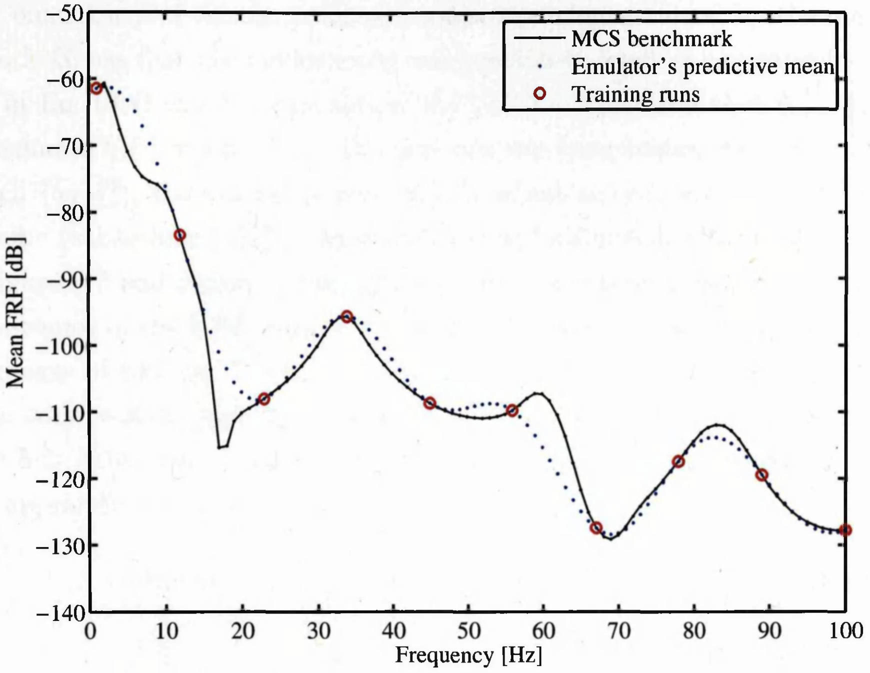


Figure 5.5: Emulation of the mean frequency response function (FRF) using strategy 1. The corresponding probability bounds have been omitted for a the sake of a clearer graph.

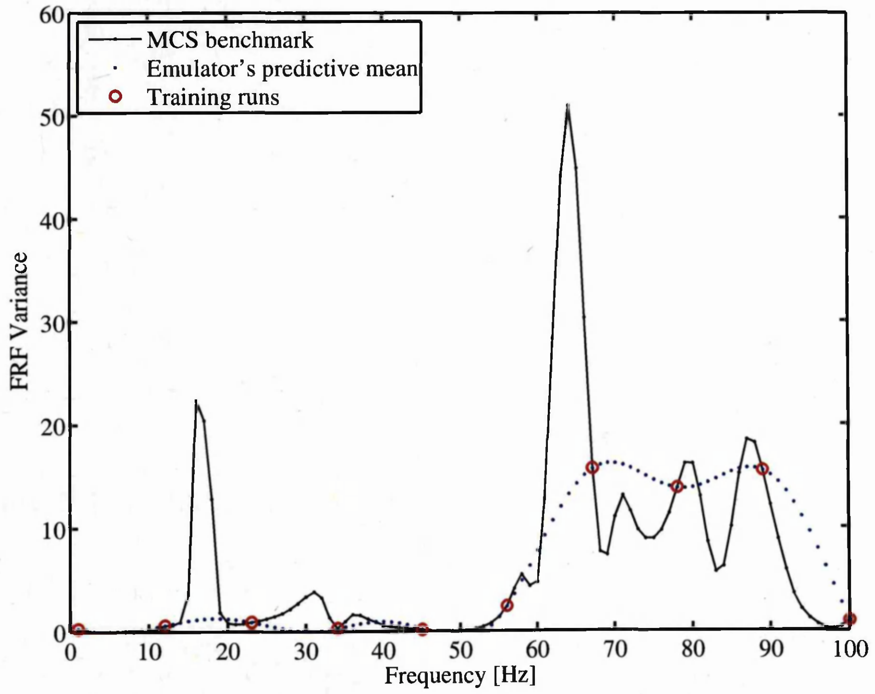


Figure 5.6: Emulation of the variance of the frequency response function (FRF) using strategy 1. The corresponding probability bounds have been omitted for a the sake of a clearer graph.

every emulator and the smoothness parameters were obtained by the maximum likelihood method. Given that the random seed was previously fixed, it was expected that for $N_E = S$, $\widetilde{\eta}_1(\cdot)$ in Eq. (5.4) should approximate $\widehat{\eta}_1(\cdot)$ in Eq. (5.2) and that $\widetilde{\eta}_2(\cdot)$ in Eq. (5.5) should approximate $\widehat{\eta}_2(\cdot)$ in Eq. (5.3). To carry out the comparison, for every emulator associated to each $\{\omega_\ell\}_{\ell=1}^{100}$, the untried points $\{\theta_k^\ell\}_{k=1}^{N_E}$ whose output was emulated were chosen to be the same realizations $\{\theta_s^\ell\}_{s=1}^{S}$ generated for the benchmark MCS runs.

Figure 5.7 and Figure 5.8 show respectively a comparison between the emulated mean and variance of the FRF against the MCS benchmark, using strategy 2. At this point, the advantages of strategy 2 over strategy 1 were confirmed. First, there was an improvement in the computation time for both the mean and the variance, as shown in Table 5.1 and Table 5.2. Note that, although strategy 2 took longer in emulating, the percentage of time both approaches took in simulating was by far larger. Thus, condition (5.6) held.

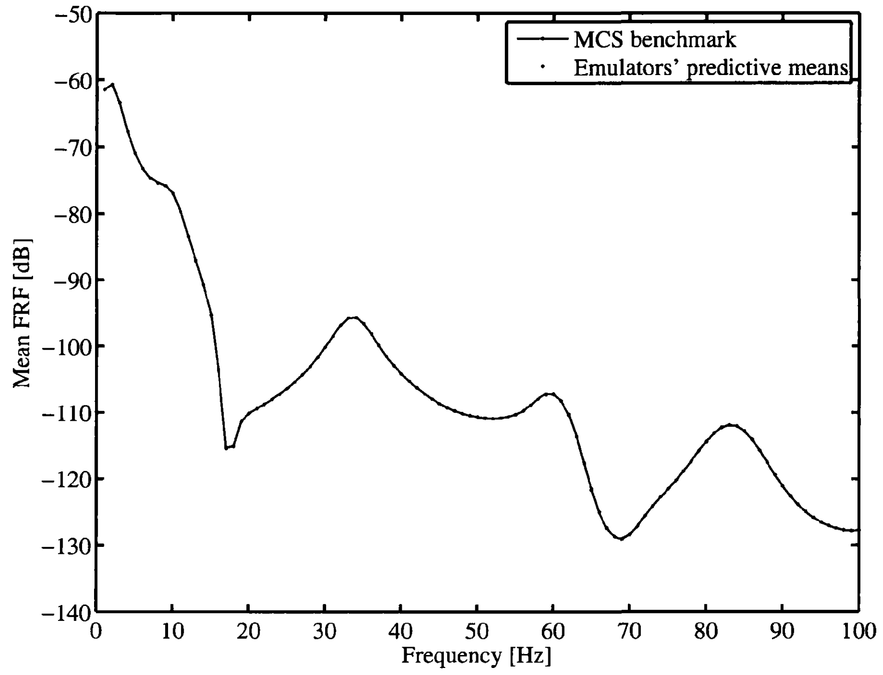


Figure 5.7: Emulation of the mean frequency response function (FRF) using strategy 2.

MEAN				
Method	Time (seconds)			Percentage
	Simulation	Emulation	Total	
MCS	73136.43	0	73136.43	100%
Strategy 1	7682.24	0.039	7682.28	10.50%
Strategy 2	782.72	24.08	806.80	1.10%

Table 5.1: Computer processing time (seconds) for the mean frequency response function.

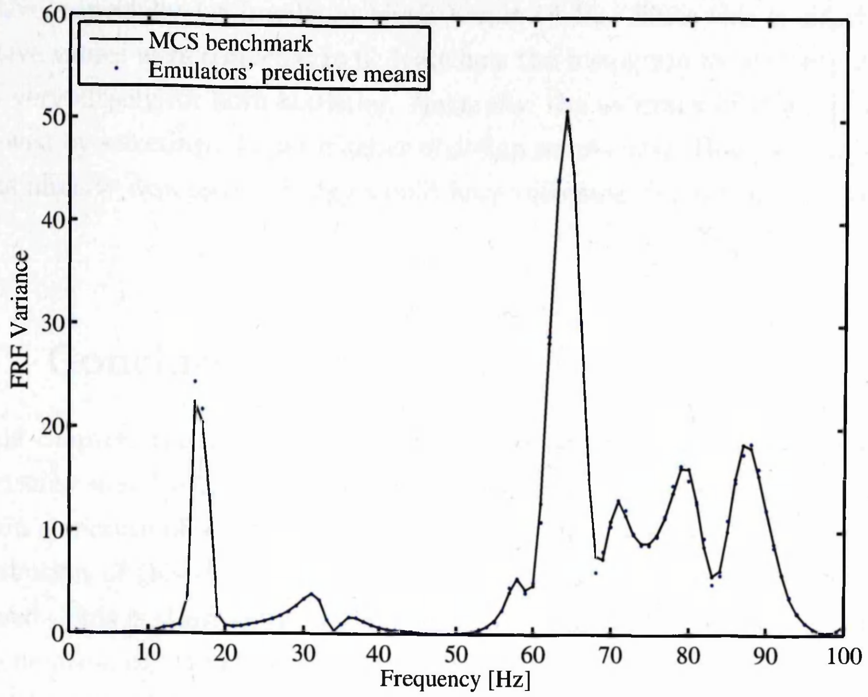


Figure 5.8: Emulation of the variance of the frequency response function (FRF) using strategy 2.

VARIANCE				
Method	Time (seconds)			Percentage
	Simulation	Emulation	Total	
MCS	73136.43	0	73136.43	100%
Strategy 1	7698.75	0.038	7698.79	10.52%
Strategy 2	756.82	24.36	781.18	1.07%

Table 5.2: Computer processing time (seconds) for the variance of the frequency response function.

The improvement in the computation time was a reflection of the dramatic reduction in the number of evaluations of $\eta(\cdot)$ when Strategy 2 was applied, as opposed to Strategy 1 and MCS. The results are summarised in Table 5.3.

Additionally, there was an improvement in accuracy. Figure 5.9 and Figure 5.10 show a comparison between the distributions of the simulated and emulated mean and variance for both strategies. In the case of strategy 1, a few emulated values of the variance were

Method	Evaluations of $\eta(\cdot)$
MCS	10^5
Strategy 1	10^4
Strategy 2	10^3

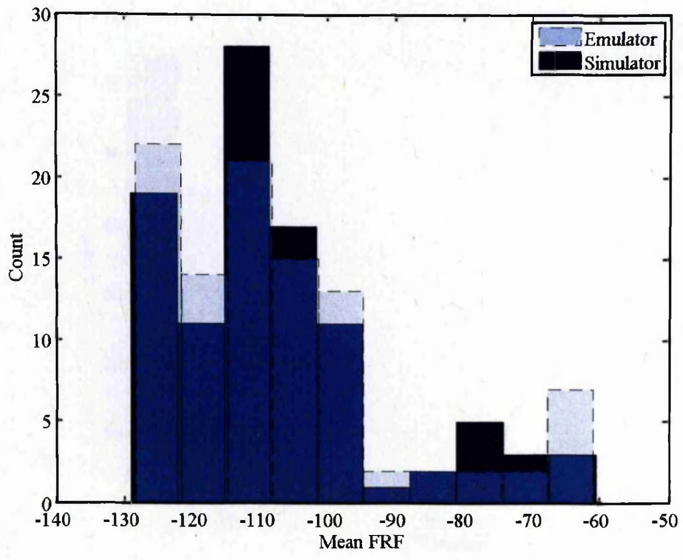
Table 5.3: Number of evaluations of the simulator $\eta(\cdot)$ by each strategy.

negative, especially for frequency levels below 10 Hz. Since this is clearly a disadvantage, negative values were corrected to 0. Note how the histogram for strategy 2 resembles that of MCS very closely for both statistics. Naturally, the accuracy of strategy 1 could have been improved by selecting a larger number of design points in Ω. However, the computation time of this already expensive strategy would have increased due to the additional evaluations of $\eta(\cdot)$.

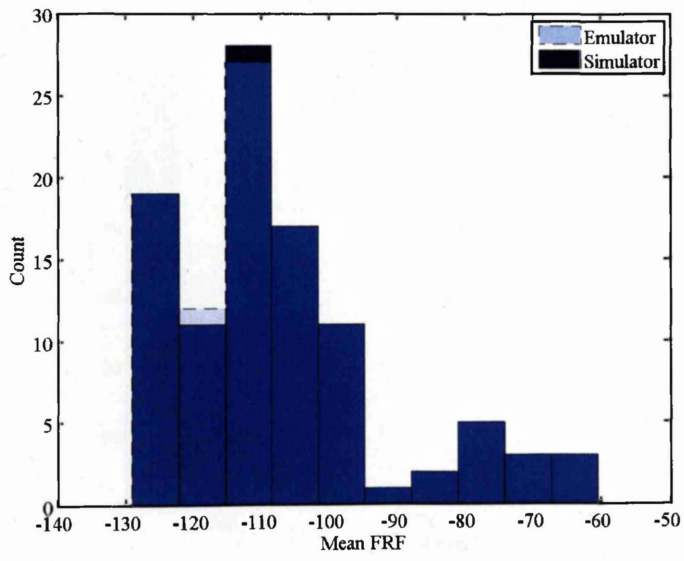
5.5 Conclusions

In this chapter, the application of GPEs to structural dynamic analysis with parametric uncertainty was discussed. A design point generation strategy which takes advantage of the domain structure of a simulator with a random parameter was proposed. This is the main contribution of this chapter, since a reduction in the number of simulator evaluations was attained. This is clearly advantageous in a context in which not only the simulator but even the generation of training runs is expensive.

In the next chapter, the representation of the random parameter as a random variable will be replaced by assuming that it is represented by a random field. This means that a novel method is needed in order to couple GPEs with one of the most frequently used stochastic finite element methods suitable for this kind of representation: the polynomial chaos expansion.

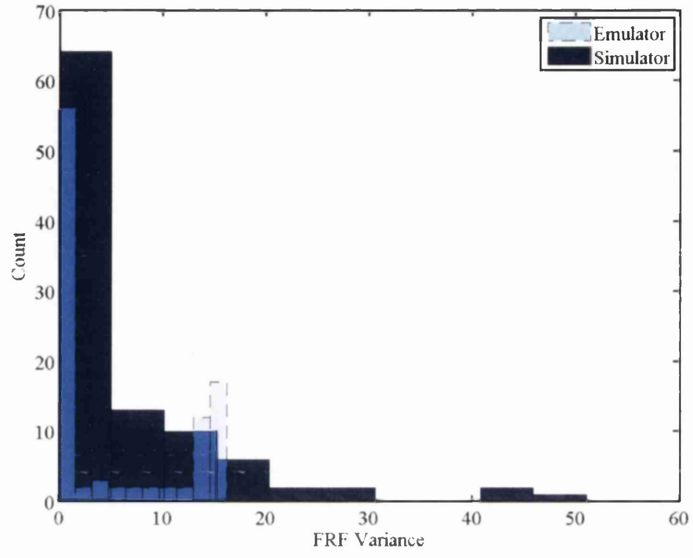


(a) Strategy 1

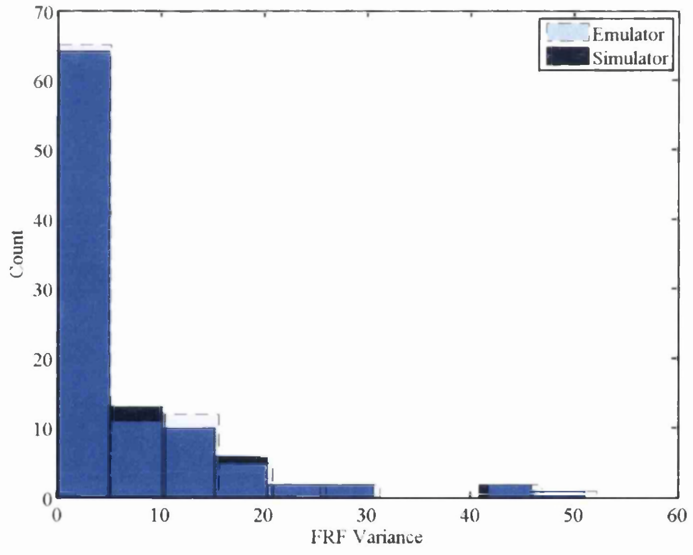


(b) Strategy 2

Figure 5.9: Histograms of simulated output mean and emulated output mean for strategy 1 and strategy 2. The degree of similarity of the distributions reflects the accuracy of each strategy.



(a) Strategy 1



(b) Strategy 2

Figure 5.10: Histograms of simulated output variance and emulated output variance for strategy 1 and strategy 2. The degree of similarity of the distributions reflects the accuracy of each strategy.

Chapter 6

Stochastic Finite Element Analysis [2]

6.1 Introduction

As mentioned in Chapter 5, the realistic modeling and prediction of engineering systems can be better achieved when their random aspects are taken into account. The associated uncertainty can be represented by stochastic coefficients in the system of stochastic partial differential equations that govern the system's response. The theoretical framework under which this problem is solved is the stochastic finite element method (SFEM). 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.

The present chapter explores the integration of GPEs into the SFEM in order to reduce the computational cost of studying engineering systems with random mechanical properties. The chapter focuses on the problem of emulating random fields based on few evaluations of a simulator. Firstly, discretised 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. Due to this, an algorithm to obtain the necessary training runs is proposed.

[2]The ideas developed in this chapter have been accepted for publication as DiazDelaO, F.A. & Adhikari, S., "Gaussian process emulators for the stochastic finite element method". *International Journal for Numerical Methods in Engineering.*

6.2 Direct Emulation of Random Fields

6.2.1 Random Field Discretisation

Extending the definition of random field given in Chapter 1, a random field $\mathcal{H}(\mathbf{x}, \theta)$ is Gaussian when for any $n \geq 1$, the vector $[\mathcal{H}(\mathbf{x}_1, \theta_0), \ldots, \mathcal{H}(\mathbf{x}_n, \theta_0)]^{\intercal}$ 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 applies for a lognormal random field. The autocorrelation function may take many distinct forms. One example is the exponential type

$$\rho(\mathbf{x}, \mathbf{x}') = e^{-\frac{|\mathbf{x}_{(1)} - \mathbf{x}'_{(1)}|}{\alpha_1} - \frac{|\mathbf{x}_{(2)} - \mathbf{x}'_{(2)}|}{\alpha_2}} \tag{6.1}$$

where $\mathbf{x}_{(i)}$ denotes the i-th 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 Chapter 2, denoted by σ.

In Chapter 1, it was mentioned that for a linear N-degree-of-freedom system, the deterministic finite element method eventually yields a system of the form

$$\mathbf{K}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \mathbf{f} \tag{6.2}$$

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 parameter, namely $\mathbf{K}(\mathbf{x}, \theta)$. It was also established in Chapter 1 that an advantageous alternative for discretising a random field $\mathcal{H}(\mathbf{x}, \theta)$ is the Karhunen-Loève expansion (KLE), for which

$$\mathcal{H}(\mathbf{x}, \theta) = \sum_{i=0}^{\infty} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) \tag{6.3}$$

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

$$\int_{\mathbb{R}^N} K(\mathbf{x}_1, \mathbf{x}_2)\phi_i(\mathbf{x}_1)d\mathbf{x}_1 = \lambda_i \phi_i(\mathbf{x}_2) \tag{6.4}$$

The truncated KLE of $\mathcal{H}(\mathbf{x}, \theta)$ up to M terms is defined as

$$\mathcal{H}(\mathbf{x}, \theta) = \mu(\mathbf{x}) + \sum_{i=1}^{M} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) \tag{6.5}$$

In particular, engineers are interested in obtaining the probability density function and the cumulative distribution function of $\mathbf{u}(\mathbf{x}, \theta)$ 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 $\mathbf{u}(\mathbf{x}, \theta)$.

Before discussing the incorporation of GPEs to the SFEM, it is worth focusing on the problem of efficiently generating realizations of the associated random field $\mathcal{H}(\mathbf{x}, \theta)$. 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 stochastic version of the linear system (6.2) with the aid of GPEs will be proposed.

6.2.2 Numerical Example: Random Field Emulation

Consider a Gaussian homogeneous two-dimensional random field $\mathcal{H}(\mathbf{x}, \theta)$ 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 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. The corresponding simulator is thus

$$\eta(\mathbf{x}, \theta) = 5 + \sum_{i=1}^{100} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(\mathbf{x}) \tag{6.6}$$

Let the spatial domain of the simulator be the square region $\mathcal{R}_L = [0, L] \times [0, L]$. 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 provided in Chapter 1, where the analytical expressions for the eigenvalues and eigenfunctions corresponding to a random field with the above characteristics were specified. Figure 6.1 shows a realization $\mathcal{H}(\mathbf{x}, \theta_0)$ of the random field with $L = 50$, which in terms of Eq. (6.6) would be $\eta(\mathbf{x}, \theta_0)$ for every $\mathbf{x} \in \mathcal{R}_{50}$. The correlation lengths were chosen to be $[\alpha_1, \alpha_2]^\intercal = [0.02L, 0.02L]^\intercal$.

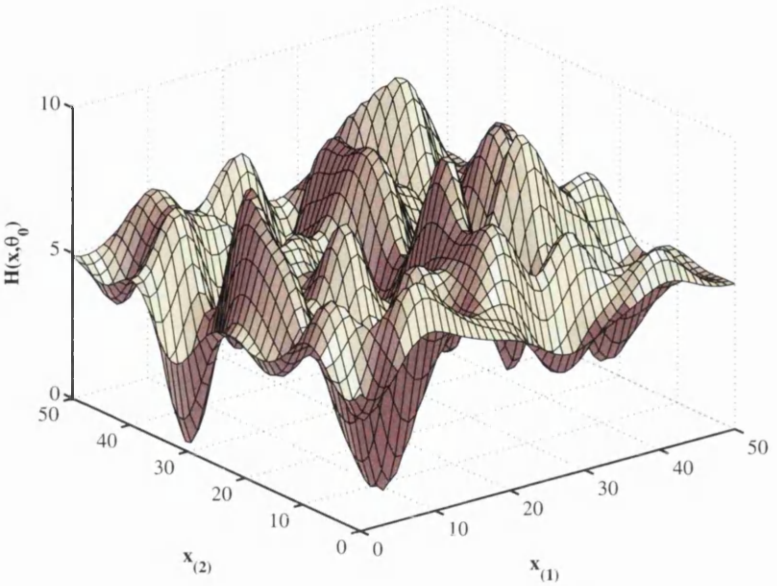
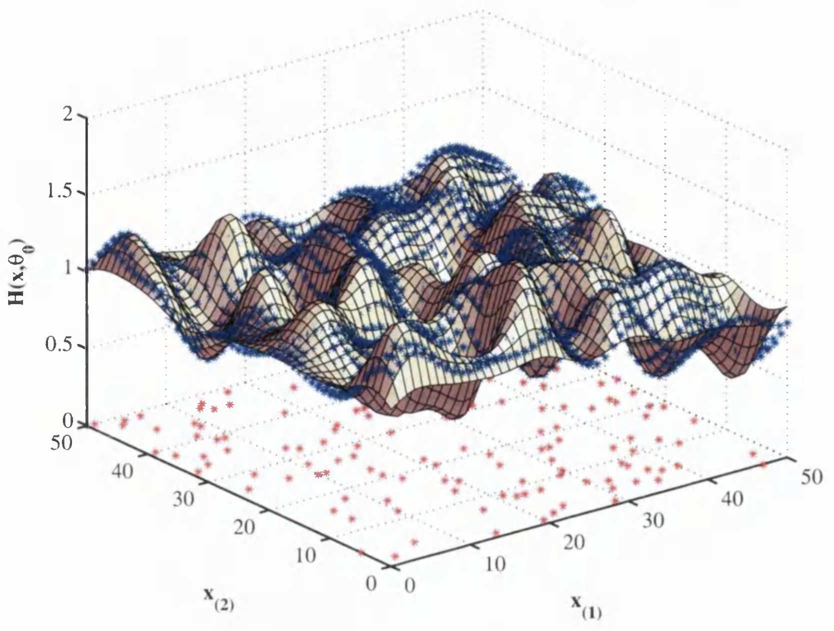
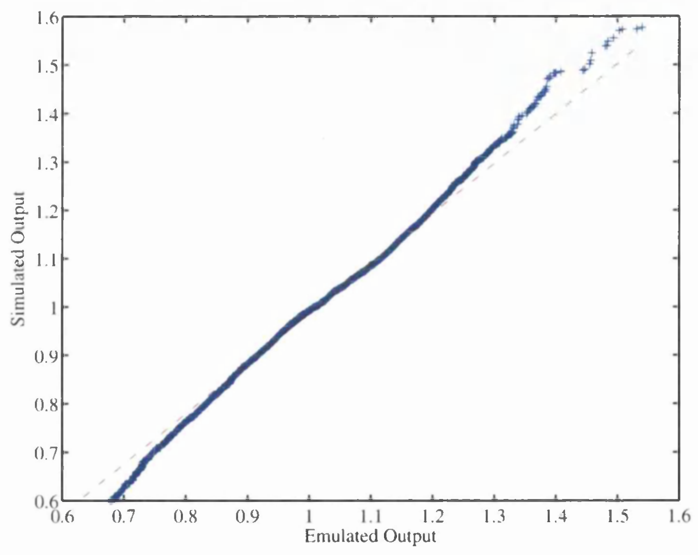


Figure 6.1: Realization of a Gaussian homogeneous random field $\mathcal{H}(\mathbf{x}, \theta)$ in $\mathcal{R}_{50} = [0, 50] \times [0, 50]$ with mean $\mu = 5$, variance $\varsigma^2 = 1$, and exponential autocorrelation function with correlation lengths $[\alpha_1, \alpha_2]^\intercal = [0.02L, 0.02L]^\intercal$. The KLE has $M = 100$ terms.

In order to implement the Gaussian process emulation algorithm to similarly generated random fields, the n design points $(\mathbf{x}, \theta_0)_1, \ldots, (\mathbf{x}, \theta_0)_n$ were selected using Latin hypercube sampling and a nearest neighbour algorithm such that every point in the Latin hypercube was translated to 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, the usual method by Haylock (1996) was employed. Figure 6.2(a) shows the values of $m^{**}(\cdot)$ compared with a realization of a random field with correlation lengths $[0.2L, 0.02L]^\intercal$, as well as the design points employed in the construction of the emulator. Figure 6.3(a) shows the emulation of a realization of another Gaussian random field, whose correlation lengths are $[0.1L, 0.02L]^\intercal$. The same design points were considered. Being $\mathcal{H}(\mathbf{x}, \theta)$ 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 better represented in Figure 6.2(b) and Figure 6.3(b), which are quantile-quantile plots (Q-Q plots) Chambers *et al.* (1983) of the emulated output against simulated output. This kind of graphical representation compares two probability distributions by plotting their quantiles against each other. In the examples below, the resulting plots are approximately linear, suggesting that the datasets come from the same distribution.



(a) Correlation lengths $= (0.02L, 0.02L)^\intercal$

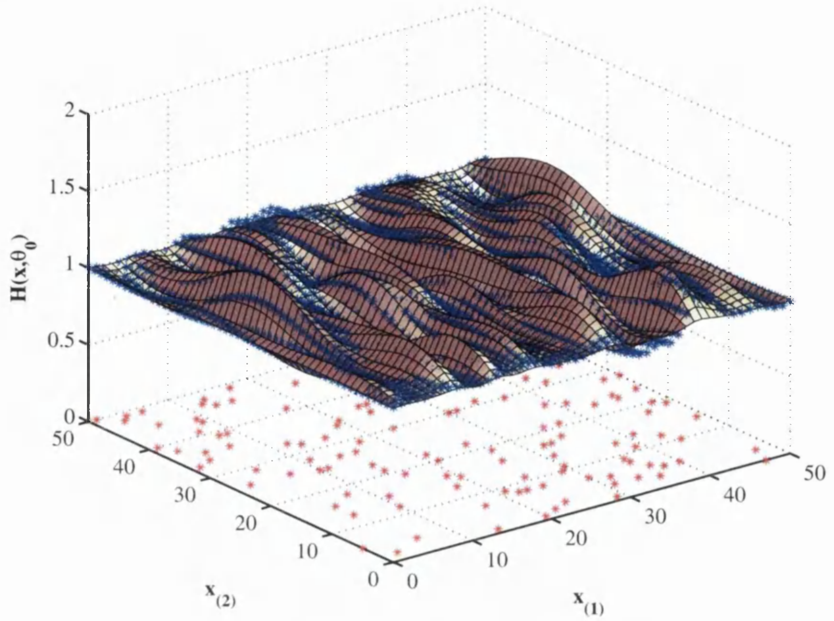


(b) Q-Q plot

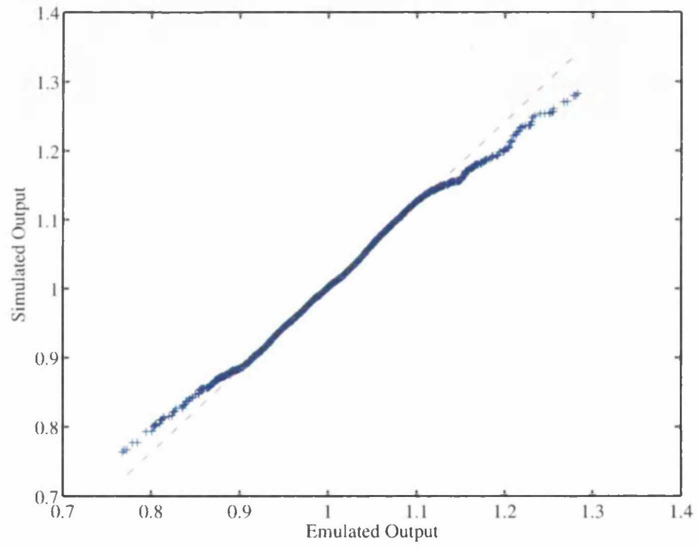
Figure 6.2: Realization $\mathcal{H}(\mathbf{x}, \theta_0)$ of a Gaussian homogeneous random field $\mathcal{H}(\mathbf{x}, \theta)$. The adequacy of the emulator is represented by the Q-Q plot below.

The exercise was repeated for two lognormal random fields with correlation lengths $[0.02L, 0.02L]^\intercal$ and $[0.02L, 0.1L]^\intercal$. The lognormal random fields were obtained using the Nataf transformation (Der Kiureghian and Li, 1986; Ditlevsen and Marsden, 1996) The results of the emulation and their respective Q-Q plots are shown in Figure 6.4 and Figure 6.5.

The exercise above was carried out for several values of L and thus an increasing number



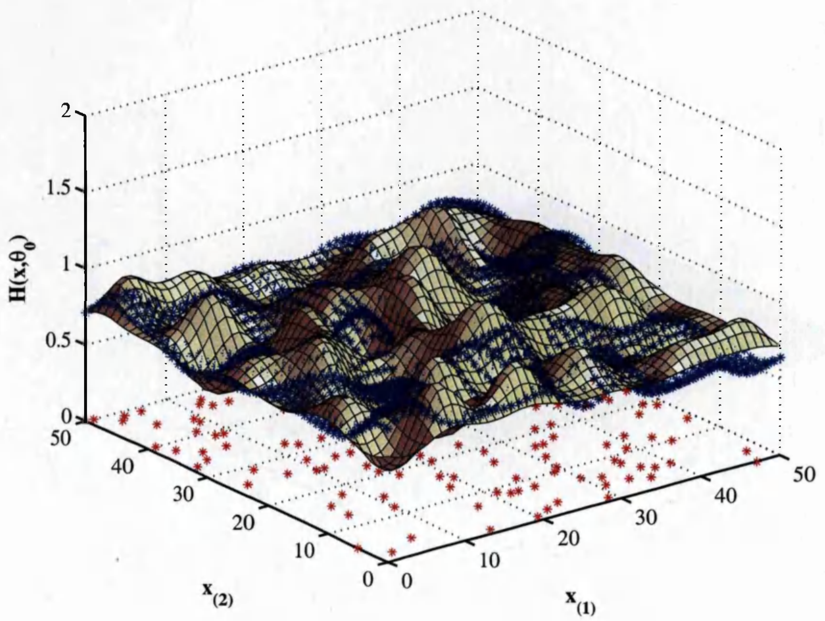
(a) Correlation lengths $= (0.1L, 0.02L)^{\mathsf{T}}$



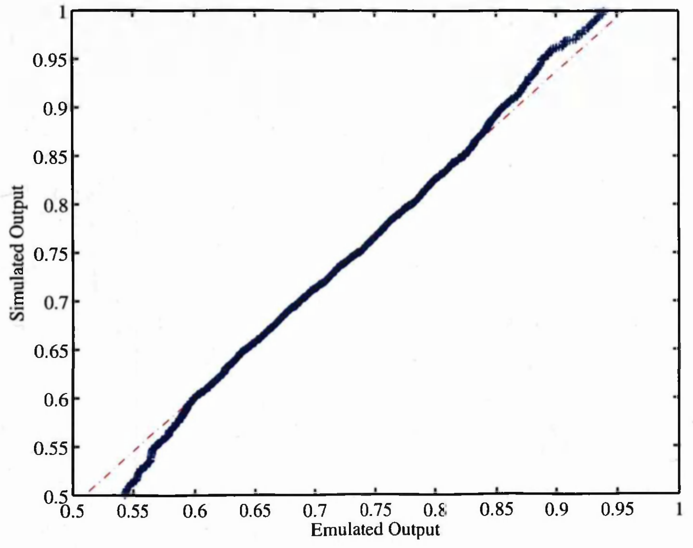
(b) Q-Q plot

Figure 6.3: Realization $\mathcal{H}(\mathbf{x}, \theta_0)$ of a Gaussian homogeneous random field $\mathcal{H}(\mathbf{x}, \theta)$. The adequacy of the emulator is represented by the Q-Q plot below.

of nodes. The percentage of design points was chosen 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 6.1. Note that the time the emulator takes includes the time of evaluating the design points.



(a) Correlation lengths = $(0.02L, 0.02L)^{\intercal}$

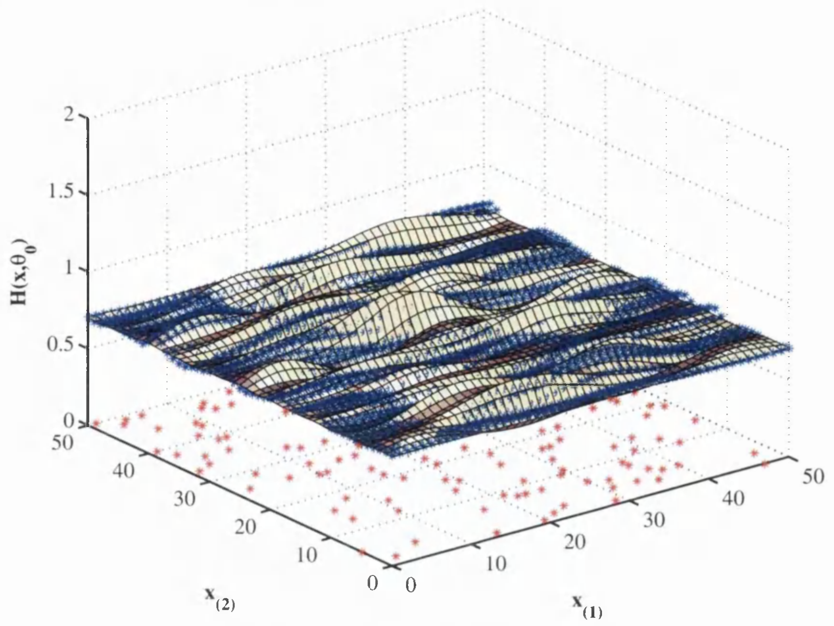


(b) Q-Q plot

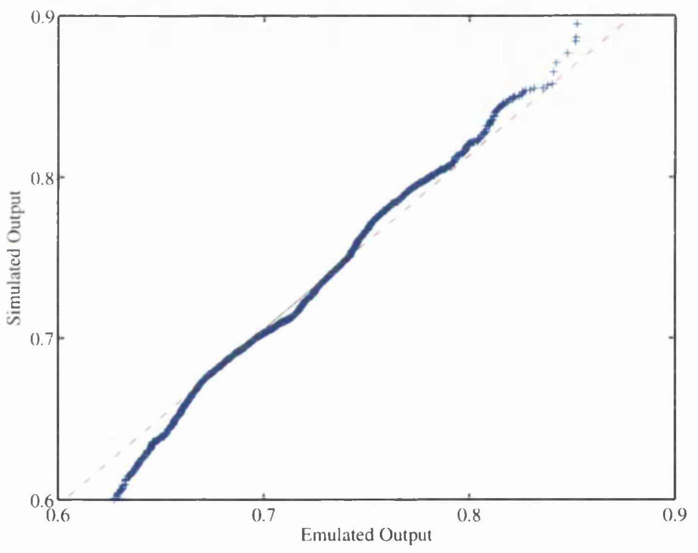
Figure 6.4: Realization $\mathcal{H}(\mathbf{x}, \theta_0)$ of a lognormal homogeneous random field $\mathcal{H}(\mathbf{x}, \theta)$. The adequacy of the emulator is represented by the Q-Q plot below.

6.3 Calculation of the Stochastic Response

The results in Section 6.2 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 Eq. (6.2) through SFEM analysis. This solution is



(a) Correlation lengths $= (0.02L, 0.1L)^{\intercal}$



(b) Q-Q plot

Figure 6.5: Realization $\mathcal{H}(\mathbf{x}, \theta_0)$ of a lognormal homogeneous random field $\mathcal{H}(\mathbf{x}, \theta)$. The adequacy of the emulator is represented by the Q-Q plot below.

important because it expresses the response of the engineering system under study. The problem of approximating it using GPEs is addressed in this section.

No. 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

Table 6.1: Number of nodes vs. CPU time employed (seconds)

6.3.1 Polynomial Chaos Expansion

As mentioned in Chapter 1, this approach consists 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

$$\mathbf{u}(\mathbf{x},\theta) = \sum_{j=0}^{P-1} \mathcal{U}_j\,(\mathbf{x})\Psi_j(\theta) \tag{6.7}$$

where each $\mathcal{U}_j$ is a deterministic vector in $\mathbb{R}^N$. Equation (6.7) is then substituted in (6.2), and $\mathbf{K}(\mathbf{x},\theta)$ is discretised using the following truncated Karhunen-Loève expansion

$$\mathbf{K}(\mathbf{x},\theta) = \sum_{i=0}^{M} \mathbf{K}_i(\mathbf{x})\xi_i(\theta) \tag{6.8}$$

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

$$\mathcal{R}(\mathbf{x},\theta) = \sum_{i=0}^{M}\sum_{j=0}^{P-1} \mathbf{K}_i(\mathbf{x})\mathcal{U}_j(\mathbf{x})\xi_i(\theta)\Psi_j(\theta) - \mathbf{f} \tag{6.9}$$

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

$$\mathcal{K}\cdot\mathcal{U} = \mathcal{F} \tag{6.10}$$

where each of the P components of $\mathcal{U} = [\mathcal{U}_0,\ldots,\mathcal{U}_{P-1}]^\intercal$ is N-dimensional and consequently the global stiffness matrix $\mathcal{K}$ is of size $NP \times NP$. Once every component in $\mathcal{U}$ is determined, the vector $\mathbf{u}(\mathbf{x},\theta)$ can be computed as in Eq. (6.7). Note that, if ν denotes the number of nodes in the finite element mesh and $\delta \geq 1$ is the number of degrees of freedom per node, then $N = \delta\nu$. 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 Karhunen-Loève expansion M, as can be seen in Table 6.2.

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

Table 6.2: Number of basis polynomials for different degrees (p) and number of terms in the Karhunen-Loève expansion (M).

In general,

$$P(p,M) = \sum_{k=0}^{p} \binom{M+k-1}{k} \tag{6.11}$$

This has an important implication, pointed out by Debusschere *et al.* (2005). 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 Karhunen-Loève expansion, the solution of the linear system (6.10) 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. In this setup, a GPE can be an inexpensive surrogate model of a polynomial chaos simulator. It can be seen from Eq. (6.7) that

$$\begin{aligned} \mathbf{u}(\mathbf{x},\theta) &= \sum_{j=0}^{P-1} \mathcal{U}_j\,(\mathbf{x})\Psi_j(\theta) \\ &= \begin{bmatrix} \mathcal{U}_0^{(1)}\,(\mathbf{x}) \\ \vdots \\ \mathcal{U}_0^{(N)}\,(\mathbf{x}) \end{bmatrix} \Psi_0(\theta) + \ldots + \begin{bmatrix} \mathcal{U}_{P-1}^{(1)}\,(\mathbf{x}) \\ \vdots \\ \mathcal{U}_{P-1}^{(N)}\,(\mathbf{x}) \end{bmatrix} \Psi_{P-1}(\theta) \end{aligned} \tag{6.12}$$

where the superscript in parenthesis numbers the row (the element) 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 $\mathcal{U}_0, \ldots, \mathcal{U}_{P-1}$, that is, $[\mathcal{U}_0^{(\gamma_1)}, \ldots, \mathcal{U}_0^{(\gamma_{n\delta})}]^\intercal$ from $\mathcal{U}_0$, $[\mathcal{U}_1^{(\gamma_1)}, \ldots, \mathcal{U}_1^{(\gamma_{n\delta})}]^\intercal$ from $\mathcal{U}_1$, until $[\mathcal{U}_{P-1}^{(\gamma_1)}, \ldots, \mathcal{U}_{P-1}^{(\gamma_{n\delta})}]^\intercal$ from $\mathcal{U}_{P-1}$. Consequently,

$n\delta$ rows (elements) of the response vector $\mathbf{u}(\mathbf{x},\theta)$ would be available, namely

$$\begin{aligned}
\mathbf{u}^{(\gamma_1)}(\mathbf{x},\theta) &= \sum_{j=0}^{P-1} \mathcal{U}_j^{(\gamma_1)}(\mathbf{x})\Psi_j(\theta) \\
\mathbf{u}^{(\gamma_2)}(\mathbf{x},\theta) &= \sum_{j=0}^{P-1} \mathcal{U}_j^{(\gamma_2)}(\mathbf{x})\Psi_j(\theta) \\
&\vdots \\
\mathbf{u}^{(\gamma_{n\delta})}(\mathbf{x},\theta) &= \sum_{j=0}^{P-1} \mathcal{U}_j^{(\gamma_{n\delta})}(\mathbf{x})\Psi_j(\theta) \qquad (6.13)
\end{aligned}$$

That way, the set $\left\{\mathcal{U}_0^{(\gamma_1)},\ldots,\mathcal{U}_{P-1}^{(\gamma_1)},\ldots,\mathcal{U}_0^{(\gamma_{n\delta})},\ldots,\mathcal{U}_{P-1}^{(\gamma_{n\delta})}\right\}$ 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 $\left\{\mathbf{u}^{(\gamma_{n\delta+1})}(\mathbf{x},\theta),\ldots,\mathbf{u}^{(\gamma_{N-n\delta})}(\mathbf{x},\theta)\right\}$, where $\{\gamma_{n\delta+1},\ldots,\gamma_{N-n\delta}\} = \{1,\ldots,N\}\backslash\Gamma$. For notational purposes, let these sets be expressed as vectors, partitioning $\mathbf{u}(\mathbf{x},\theta)$ as

$$\mathbf{u}(\mathbf{x},\theta) = \begin{pmatrix} \mathbf{u}_1(\mathbf{x},\theta) \\ \\ \mathbf{u}_2(\mathbf{x},\theta) \end{pmatrix} = \begin{pmatrix} \mathbf{u}^{(\gamma_{n\delta+1})}(\mathbf{x},\theta) \\ \vdots \\ \mathbf{u}^{(\gamma_{N-n\delta})}(\mathbf{x},\theta) \\ \mathbf{u}^{(\gamma_1)}(\mathbf{x},\theta) \\ \vdots \\ \mathbf{u}^{(\gamma_{n\delta})}(\mathbf{x},\theta) \end{pmatrix} \qquad (6.14)$$

Note however that obtaining the design points is not as simple as it was when emulating the random fields in Section 6.2: 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 (6.10) only partially, such that the corresponding training runs will be used as input for the emulation algorithm. The training runs can be expressed as $\mathbf{y} = [\mathbf{y}_1,\ldots,\mathbf{y}_\delta] \in \mathbb{R}^{n\times\delta}$, where

$$\begin{aligned}
\mathbf{y}_1 &= [\mathbf{u}^{(\gamma_1)},\ldots,\mathbf{u}^{(\gamma_\delta)}]^{\mathsf{T}} \\
\mathbf{y}_2 &= [\mathbf{u}^{(\gamma_{\delta+1})},\ldots,\mathbf{u}^{(\gamma_{2\delta})}]^{\mathsf{T}} \\
&\vdots \\
\mathbf{y}_n &= [\mathbf{u}^{(\gamma_{n\delta-\delta+1})},\ldots,\mathbf{u}^{(\gamma_{n\delta})}]^{\mathsf{T}} \qquad (6.15)
\end{aligned}$$

The following subsection elaborates on this idea and proposes an algorithm to solve the linear system (6.10) partially in order to obtain the design points and the training runs $\mathbf{y}$.

6.3.2 Partitioned Cholesky Decomposition

Let the stochastic finite element system $\mathcal{K} \cdot \mathcal{U} = \mathcal{F}$ in Eq. (6.10) be partitioned as

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^{\mathsf{T}} & \mathbf{C} \end{pmatrix} \begin{pmatrix} \mathcal{U}_1^* \\ \mathcal{U}_2^* \end{pmatrix} = \begin{pmatrix} \mathcal{F}_1 \\ \mathcal{F}_2 \end{pmatrix} \tag{6.16}$$

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 $\mathcal{U}$, $\mathcal{U}_1^*$ and $\mathcal{U}_2^*$, and $\mathcal{U}_0, \ldots, \mathcal{U}_{P-1}$ are related in the following way

$$\mathcal{U} = \begin{pmatrix} \mathcal{U}_1^* \\ \\ \mathcal{U}_2^* \end{pmatrix} = \begin{pmatrix} \mathcal{U}_0\,(\mathbf{x}) \\ \vdots \\ \mathcal{U}_{P-1}\,(\mathbf{x}) \end{pmatrix} \tag{6.17}$$

Definition. Let $\mathcal{K}$ be a positive definite matrix partitioned as in Eq. (6.16). The **partitioned Cholesky factor L** of $\mathcal{K}$ is given by

$$\mathbf{L} = \begin{pmatrix} \mathbf{L_A} & \mathbf{0} \\ \mathbf{W}^{\mathsf{T}} & \mathbf{L_D} \end{pmatrix} \tag{6.18}$$

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

$$\mathbf{W}^{\mathsf{T}}\mathbf{W} = \mathbf{B}^{\mathsf{T}}\mathbf{L}_{\mathbf{A}}^{-\mathsf{T}}\mathbf{L}_{\mathbf{A}}^{-1}\mathbf{B} = \mathbf{B}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{B} \tag{6.19}$$

The calculation of the partitioned Cholesky factor $\mathbf{L}$ of $\mathcal{K}$ is summarized in Algorithm 4.

It can be shown (George, 1981) 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 the matrix $\mathcal{K}$ is decomposed as $\mathbf{L}\mathbf{L}^{\mathrm{T}}$, the solution of the linear system (6.16) can be obtained by first solving the lower triangular linear system

$$\begin{pmatrix} \mathbf{L_A} & \mathbf{0} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{L_D} \end{pmatrix} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} \mathcal{F}_1 \\ \mathcal{F}_2 \end{pmatrix} \tag{6.20}$$

```
Alg. 4 Partitioned Cholesky factor L of $\mathcal{K}$
Input: Matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$
Output: Partitioned Cholesky factor $\mathbf{L}$
    begin
        1. Factor $\mathbf{A}$ into $\mathbf{L_A L_A^T}$
        2. Solve the (triangular) linear systems $\mathbf{L_A W} = \mathbf{B}$
        3. Compute $\mathbf{D} = \mathbf{C} - \mathbf{W^T W}$
        4. Factor $\mathbf{D}$ into $\mathbf{L_D L_D^T}$
        5. Assemble $\mathbf{L}$ as in Eq. (6.18)
    end
```

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

$$\begin{pmatrix} \mathbf{L_A^T} & \mathbf{W} \\ 0 & \mathbf{L_D^T} \end{pmatrix} \begin{pmatrix} \mathcal{U}_1^* \\ \mathcal{U}_2^* \end{pmatrix} = \begin{pmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{pmatrix} \tag{6.21}$$

The procedure is made explicit in Algorithm 5.

```
Alg. 5 Solution of the partitioned system $\mathcal{K} \cdot \mathcal{U} = \mathcal{F}$
Input: $\mathbf{L}, \mathcal{F}_1, \mathcal{F}_2$
Output: $\mathcal{U}_1^*, \mathcal{U}_2^*$
    begin
    Forward Solution:
        1. Solve $\mathbf{L_A y_1} = \mathcal{F}_1$
        2. Compute $\widetilde{\mathbf{f}}_2 = \mathcal{F}_2 - \mathbf{W^T y_1}$
        3. Solve $\mathbf{L_D y_2} = \widetilde{\mathbf{f}}_2$
    Backward Solution:
        4. Solve $\mathbf{L_D}^{\mathsf{T}} \mathcal{U}_2^* = \mathbf{y_2}$
        5. Compute $\widetilde{\mathbf{y}}_1 = \mathbf{y_1} - \mathbf{W}\mathcal{U}_2^*$
        6. Solve $\mathbf{L_A^T} \mathcal{U}_1^* = \widetilde{\mathbf{y}}_1$
    end
```

In the worst case scenario, the computational complexity of implementing Algorithm 4 together with Algorithm 5 is the following: $O(N^3P^3)$ operations are required for the Cholesky decomposition, $O(N^2P^2)$ are necessary for the forward substitution and $O(N^2P^2)$ are required for the backward substitution. The joint application of Algorithm 4 and Algorithm 5 will hereafter be referred to as the symmetric algorithm.

From Table 6.2, it can be seen that even for a system with few degrees of freedom, the number of operations $O(N^2P^2)$ can be considerably large. To potentially reduce the cost of the symmetric algorithm, an alternative approach is proposed.

6.3.3 Coupling Polynomial Chaos with GPEs

As discussed in Subsection 6.3.1, 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 Eq. (6.12) and the relationship provided by Eq. (6.17), it can be seen that in order to obtain them, it is necessary to solve the linear system (6.10) for $n\delta P$ components of $\boldsymbol{\mathcal{U}}$. Once these components are computed, the training runs would be obtained as in Eq. (6.13). Note that if these $n\delta P$ components were all contained in the partition $\boldsymbol{\mathcal{U}}_2^*$, then it would suffice to implement Algorithm 5 up to step 4. It is however unlikely that the components of $\boldsymbol{\mathcal{U}}_2^*$ will be ordered such that the nodes associated to 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 $\boldsymbol{\mathcal{K}}$ can be permuted in order to group the desired components in $\boldsymbol{\mathcal{U}}_2^*$. It is worth noting that this permutation will render a matrix that is still symmetric and positive definite, such that Algorithm 5 can be readily applied. This result is proved below.

Proposition. Let $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{d \times d}$ be a symmetric positive definite matrix. Then, any permutation $\widetilde{\boldsymbol{\mathcal{K}}}$ of the rows and columns of $\boldsymbol{\mathcal{K}}$ is symmetric positive definite.

Proof. Let $\mathbf{P} \in \mathbb{R}^{d \times d}$ be a permutation matrix, such that $\widetilde{\boldsymbol{\mathcal{K}}} = \mathbf{P}^{\mathsf{T}}\boldsymbol{\mathcal{K}}\mathbf{P}$. Then, for every $\mathbf{z} \neq \mathbf{0} \in \mathbb{R}^d$

$$\begin{aligned} \mathbf{z}^{\mathsf{T}}\widetilde{\boldsymbol{\mathcal{K}}}\mathbf{z} &= \mathbf{z}^{\mathsf{T}}\mathbf{P}^{\mathsf{T}}\boldsymbol{\mathcal{K}}\mathbf{P}\mathbf{z} \\ &= (\mathbf{P}\mathbf{z})^{\mathsf{T}}\boldsymbol{\mathcal{K}}(\mathbf{P}\mathbf{z}) \\ &= \mathbf{y}^{\mathsf{T}}\boldsymbol{\mathcal{K}}\mathbf{y} \\ &> 0 \end{aligned}$$

The symmetry follows from

$$\widetilde{\boldsymbol{\mathcal{K}}}^{\mathsf{T}} = (\mathbf{P}^{\mathsf{T}}\boldsymbol{\mathcal{K}}\mathbf{P})^{\mathsf{T}} = \mathbf{P}^{\mathsf{T}}\boldsymbol{\mathcal{K}}^{\mathsf{T}}\mathbf{P} = \mathbf{P}^{\mathsf{T}}\boldsymbol{\mathcal{K}}\mathbf{P}$$

Once the design points are obtained, the training runs in Eq. (6.15) are computed and a GPE can be built in order to approximate each of the remaining 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

$$\mathbf{u}_1^{(\ell)}(\mathbf{x}, \theta) = \mathcal{GPE}(\mathbf{u}_2(\mathbf{x}, \theta)) \tag{6.22}$$

The proposed algorithm is summarized in Algorithm 6.

Alg. 6 Solution of the partitioned system $\mathcal{K} \cdot \mathcal{U} = \mathcal{F}$ using GPEs

```
Input: $\mathcal{K}$, $\mathcal{F}_1, \mathcal{F}_2$
Output: $\mathcal{U}_1^*, \mathcal{U}_2^*$
    begin
        1. Choose a permutation $\widetilde{\mathcal{K}}$ of $\mathcal{K}$
        2. Compute the partitioned Cholesky factor $\mathbf{L}$ of $\widetilde{\mathcal{K}}$
    Forward Solution:
        3. Solve $\mathbf{L_A y_1} = \mathcal{F}_1$
        4. Compute $\widetilde{\mathbf{f}}_2 = \mathcal{F}_2 - \mathbf{W}^{\mathsf{T}}\mathbf{y}_1$
        5. Solve $\mathbf{L_D y_2} = \widetilde{\mathbf{f}}_2$
    Partial Backward Solution:
        6. Solve $\mathbf{L_D}^{\mathsf{T}}\mathcal{U}_2^* = \mathbf{y}_2$
    Design points:
        7. Compute $\mathbf{u}_2(\mathbf{x}, \theta)$ as in Eq. (6.13)
    Training runs:
        8. Compute the training runs $\mathbf{y}$ as in Eq. (6.15)
    Gaussian Process Emulation:
        for $\ell = 1, \ldots, m$
        9. $\mathbf{u}_1^{(\ell)}(\mathbf{x}, \theta) = \mathcal{GPE}(\mathbf{u}_2(\mathbf{x}, \theta))$
        end
    end
```

6.3.4 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 the prior to posterior analysis in Chapter 2, where the corresponding covariance matrix $C^{**} \in \mathbb{R}^{n \times n}$ and $\mathbf{y} \in \mathbb{R}^{n \times \delta}$. Therefore, the proposed algorithm would reduce the computational cost of solving the linear system (6.10) if and only if

$$O(N^2P^2) > O(n^2\delta^2P^2) + O(n^3\delta) \tag{6.23}$$

In Subsection 6.3.1, ν was defined as the number of nodes and $N = \delta\nu$. Let n be a small fraction of ν such that $n = \epsilon\nu$ with $0 < \epsilon \ll 1$. Thus, $n = \epsilon N/\delta$. This expression allows relating condition (6.23) to the number of nodes as follows

$$\begin{aligned} N^2P^2 &> n^2\delta^2P^2 + n^3\delta \\ \Leftrightarrow N^2P^2 &> \epsilon^2N^2P^2 + (\epsilon^3N^3)/\delta^2 \\ \Leftrightarrow P^2 &> \epsilon^2P^2 + (\epsilon^3N)/\delta^2 \\ \Leftrightarrow 1 &> \epsilon^2 + (\epsilon^3N)/(P^2\delta^2) \\ \Leftrightarrow 1 &> \epsilon^2 + (\epsilon^3\nu)/(P^2\delta) \end{aligned} \tag{6.24}$$

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 $\delta \geq 1$. On the other hand, $\epsilon^3\nu$ is a very small fraction of ν. Therefore, condition (6.23) is likely to be met, unless for example that the number of nodes ν is considerably larger than the number of basis polynomials P.

6.3.5 Efficient Memory Allocation

In order to improve the memory usage when calculating the partitioned Cholesky factor $\mathbf{L}$, George (1981) observes that the matrix $\mathbf{B}^\intercal\mathbf{A}^{-1}\mathbf{B}$ in Algorithm 4 can be calculated not only by the conventional expressions

$$\mathbf{B}^\intercal\mathbf{L}_\mathbf{A}^{-\intercal}\mathbf{L}_\mathbf{A}^{-1}\mathbf{B} = \mathbf{W}^\intercal\mathbf{W} \tag{6.25}$$

but also by defining

$$\widetilde{\mathbf{W}} = \mathbf{L}_\mathbf{A}^{-\intercal}\mathbf{W} \tag{6.26}$$

That way,

$$\mathbf{B}^\intercal\widetilde{\mathbf{W}} = \mathbf{B}^\intercal(\mathbf{L}_\mathbf{A}^{-\intercal}\mathbf{W}) = \mathbf{B}^\intercal(\mathbf{L}_\mathbf{A}^{-\intercal}(\mathbf{L}_\mathbf{A}^{-1}\mathbf{B})) \tag{6.27}$$

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 $\mathbf{W}^\intercal\mathbf{z}$ or $\mathbf{W}\mathbf{z}$ can be computed by solving a triangular system and multiplying by a sparse matrix, namely $\mathbf{B}^\intercal(\mathbf{L}_\mathbf{A}^{-\intercal}\mathbf{z})$ or $\mathbf{L}_\mathbf{A}^{-1}(\mathbf{B}\mathbf{z})$. Storage is then saved if B is sparser than $\mathbf{W}$. Moreover, the storage of $\mathbf{W}$ can be avoided by computing $\mathbf{D}$ following the asymmetric scheme: $\mathbf{B}^\intercal\widetilde{\mathbf{W}}$ can be computed one column at a time, discarding each after modifying a column of $\mathbf{D}$. That way, only a temporary vector is required.

6.3.6 Numerical Example: A Cantilever Plate

The proposed algorithm is tested in an example of a square plate model clamped along one edge presented in Ghanem and Spanos (1991). The analysis is based on a polynomial chaos simulator by Haukaas and Der Kiureghian (1999). 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 $\overline{E} = 2.0 \times 10^5$ MPa and known exponential covariance function. A finite element model of a plate with 15×15 elements (and $\nu = 256$ nodes) is shown in Figure 6.6. 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, 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%.

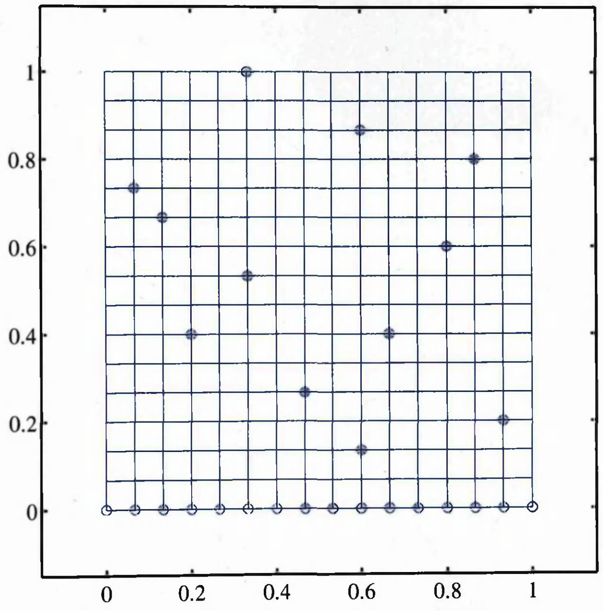
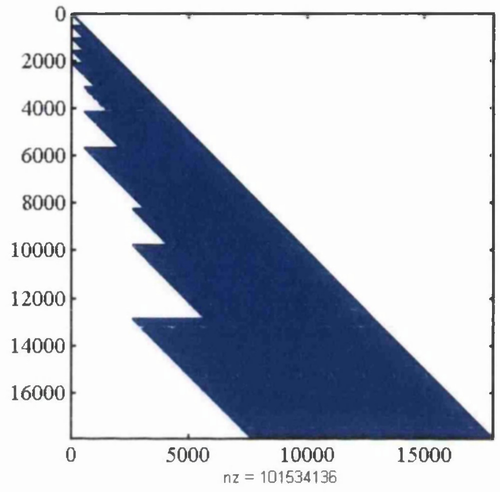
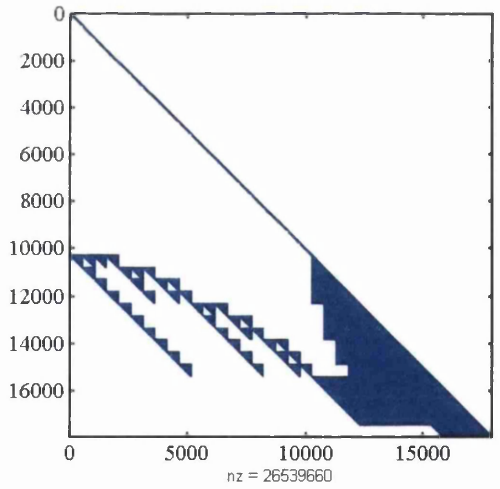


Figure 6.6: Cantilever plate model clamped along $y = 0$. The design points are shown as circles.

The polynomial chaos analysis with four basis 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 $\mathcal{K}$ had $17,920 \times 17,920$ elements. The partitioned Cholesky factor had 31.62% non-zero elements (see Figure 6.7(a)). The rows and columns of $\mathcal{K}$ corresponding to the displacement of the design points were identified and a permutation was applied in order to apply Algorithm 6. The resulting Cholesky factor of $\widetilde{\mathcal{K}}$ had 8.26% non-zero elements (see Figure 6.7(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 $\widetilde{\mathcal{K}}$ will be at least as sparse as that of $\mathcal{K}$. This is a major research topic in numerical analysis, and for now beyond the scope of this dissertation. 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.



(a) Partitioned Cholesky factor. Original matrix $\mathcal{K}$.



(b) Partitioned Cholesky factor. Permuted matrix $\widetilde{\mathcal{K}}$.

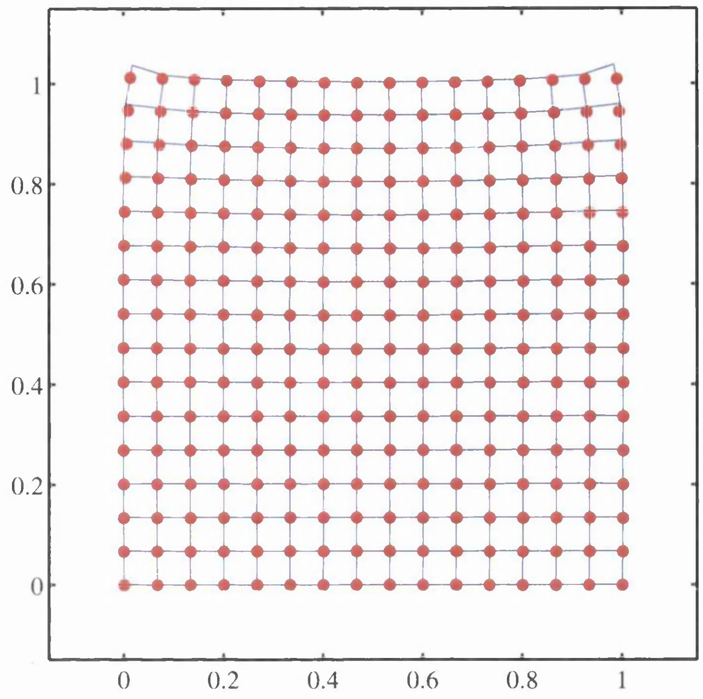
Figure 6.7: Matrix profiles showing the non-zero elements of the partitioned Cholesky factors for the original and the permuted matrix. The permuted factor of $\widetilde{\mathcal{K}}$ is much sparser in this case.

The displacements were computed for the design points, and a GPE was then built to predict the displacement of the remaining nodes. Condition (6.23) 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 of the random variables $\xi_0, \ldots, \xi_{P-1}$ where generated and each of these displacement fields was emulated. The mean simulated and emulated displacements for each node are compared in Figure 6.8(a). The relative (Euclidean) distance between the mean simulated and emulated displacements is shown in Figure 6.8(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 6.9(a) and Figure 6.9(b).

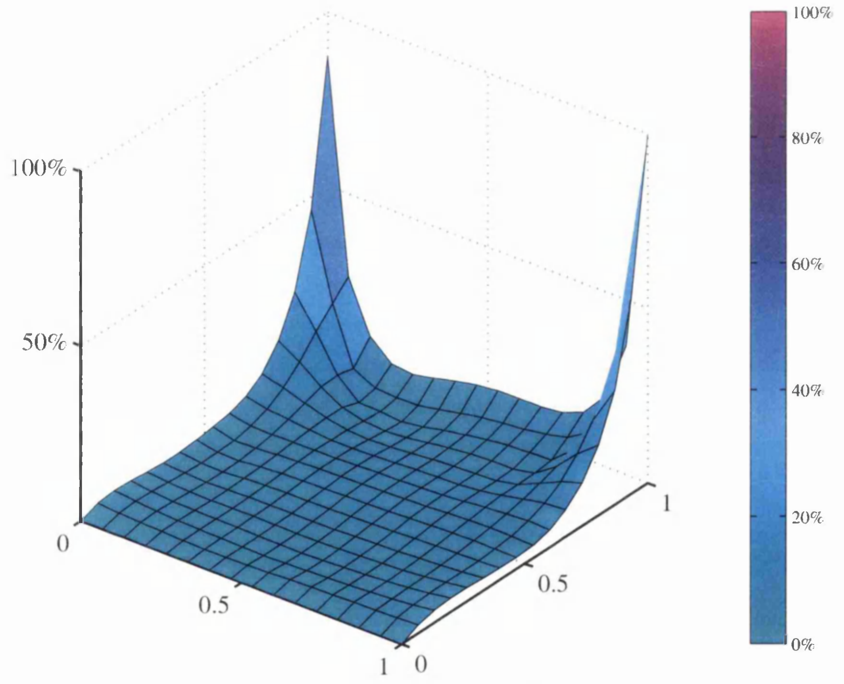
Finally, in order to treat the response $\mathbf{u}(\mathbf{x}, \theta)$ statistically, both in the vertical and horizontal directions, the probability density functions of the displacement of three nodes across the plate was generated. As Figure 6.10, Figure 6.11, and Figure 6.12 show, the uncertainty in the GPE prediction seems to increase the farther away is the node from the fixed edge, since the magnitudes of the node displacements are expected to be larger.

6.4 Conclusions

In this chapter, the capabilities of GPEs to approximate computationally expensive random fields was explored. Realizations of Gaussian and non-Gaussian random fields were emulated for an increasing number of points in the input domain. The generation of the training runs was straightforward once the simulator was defined and the output discretised with the Karhunen-Loève expansion. A more particular random field, the one induced by the solution of a stochastic finite element analysis was emulated by a coupling of the polynomial chaos expansion method with GPEs. Generating the training runs from the polynomial chaos simulator was not straightforward. Therefore, the main contribution of this chapter is the proposal of a novel algorithm that solves a partition of the main linear system and thus generates the necessary training runs. Using the proposed approach it was possible to obtain a high-resolution random field and compute the complete response of a stochastic system.

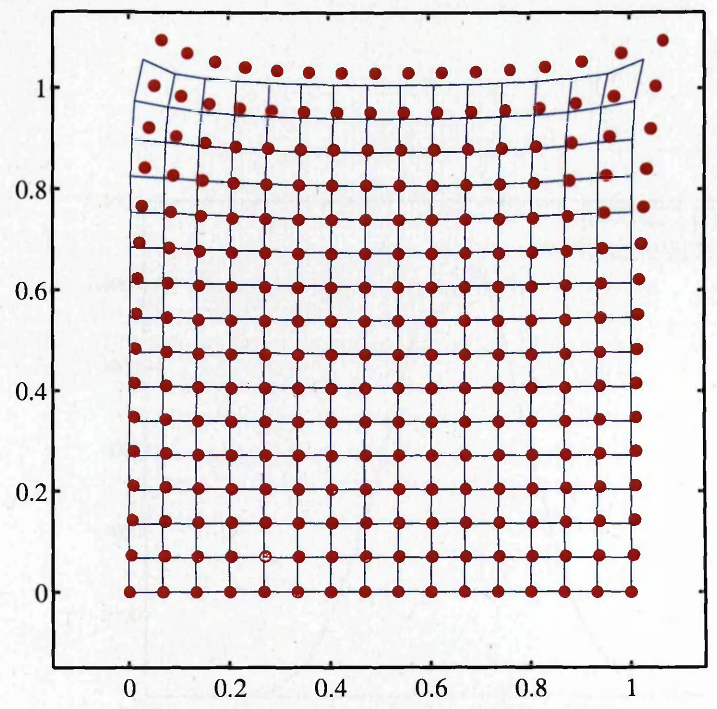


(a) Emulated and simulated displacements

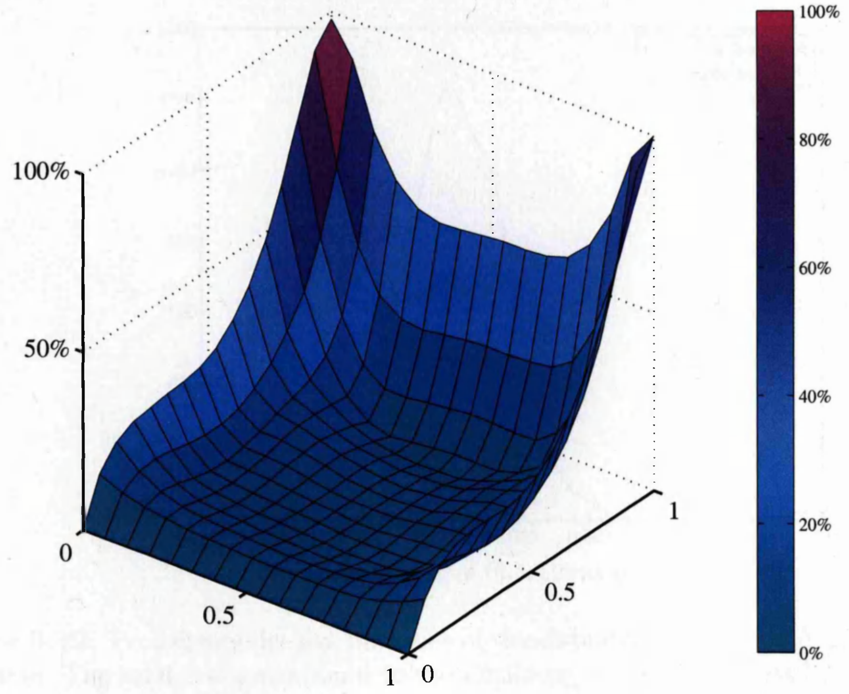


(b) Relative Euclidean distances

Figure 6.8: Comparison between the mean simulated (lines) and mean emulated (dots) displacement fields. The relative Euclidean distances are normalized by the overall maximum distance.

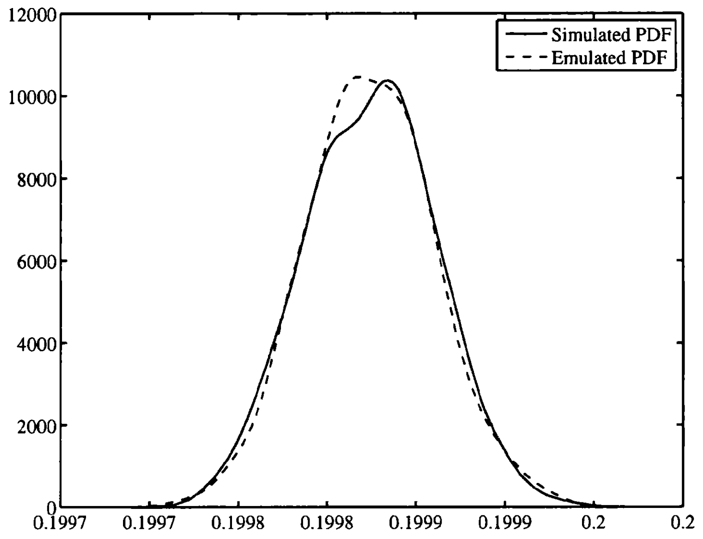


(a) Emulated and simulated displacements

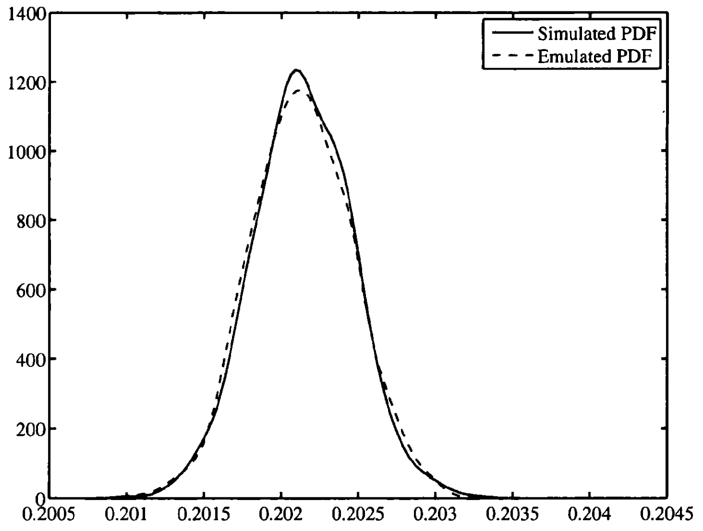


(b) Relative Euclidean distances

Figure 6.9: 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.

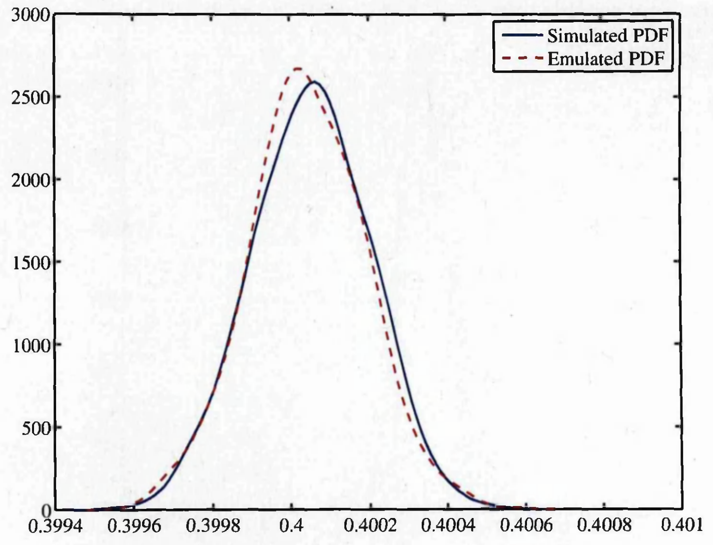


(a) Displacement of (0.2,0.2) along the x-axis

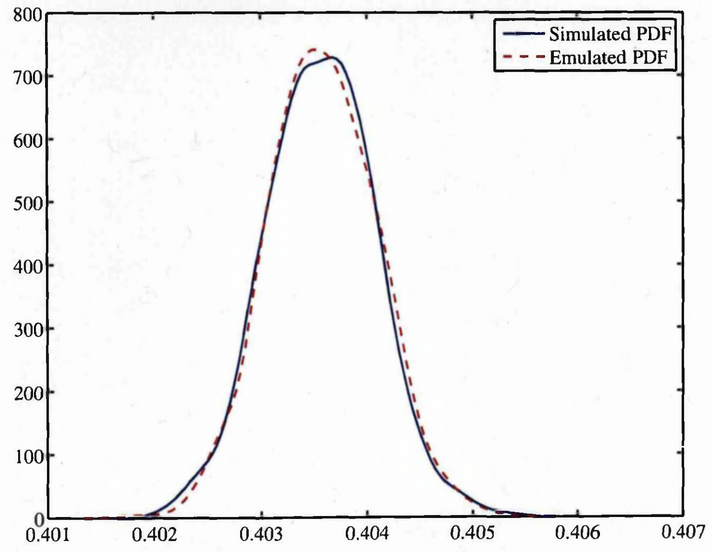


(b) Displacement of (0.2,0.2) along the y-axis

Figure 6.10: Probability density functions of the displacement of node (0.2,0.2) along the x and the y axes. The solid line corresponds to the simulator, whereas the dotted line corresponds to the emulator.

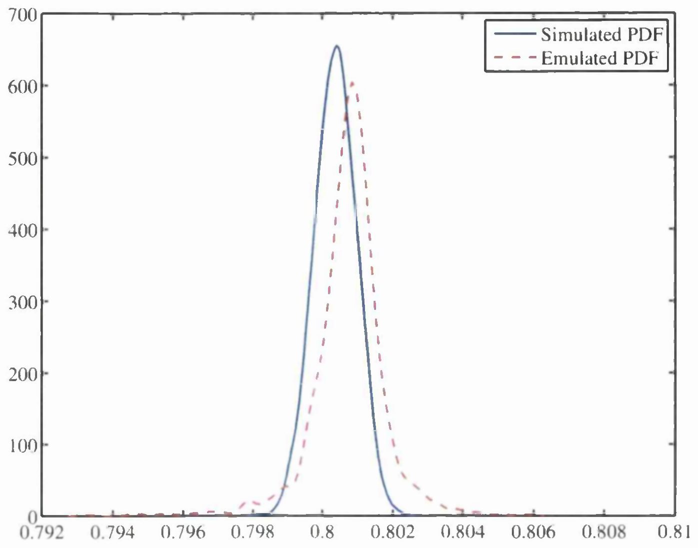


(a) Displacement of (0.4,0.4) along the x-axis

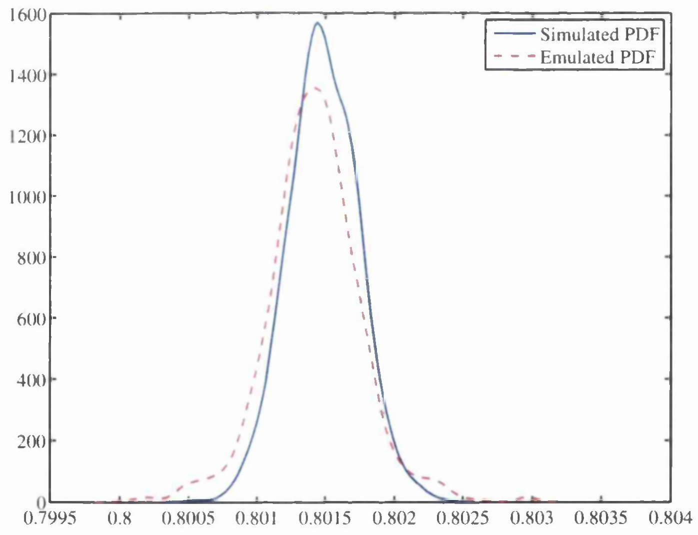


(b) Displacement of (0.4,0.4) along the y-axis

Figure 6.11: Probability density functions of the displacement of nodes (0.4,0.4) 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 (0.8,0.8) along the x-axis



(b) Displacement of (0.8,0.8) along the y-axis

Figure 6.12: Probability density functions of the displacement of node (0.8,0.8) along the x and the y axes. The solid line corresponds to the simulator, whereas the dotted line corresponds to the emulator.

Chapter 7

Summary and Conclusions

The work carried out in this dissertation aimed at introducing GPEs as an efficient predictive and uncertainty quantification tool for the analysis of complex engineering systems. Summary and contributions have been taken up at the end of the relevant chapters. This chapter recapitulates the main findings and contributions and proposes future research directions.

7.1 Summary of Completed Work

After the motivation for this dissertation was presented in Chapter 1, Chapter 2 discussed the mathematical theory behind GPEs. Basic definitions and assumptions were presented, along with the main Gaussian process emulation algorithm. Some aspects of the implementation, in particular the selection of the initial design and the estimation of the smoothness parameters were discussed, and the output of a simple simulator was emulated to illustrate the properties of the metamodel.

In Chapter 3, the application of GPEs as an efficient predictive computational tool in deterministic structural dynamics was proposed. To this end, the capabilities of GPEs were tested in both simulated and experimental contexts. The FRFs of several dynamical systems were emulated and the results were contrasted with the output of the original simulators. An FRF obtained via experimental methods was also emulated. Real data were used as the set of training runs and the experimental output was compared with the corresponding approximation. The results were particularly appealing in the medium and high frequency ranges. Since the validation of GPEs as appropriate surrogate models cannot be carried out only by visual inspection, some diagnostics of adequacy were implemented, and the agreement between a simulator's and an emulator's output was verified.

In Chapter 4, a method based on GPEs to solve boundary value problems in the context of domain decomposition was proposed. The method can assimilate a low-fidelity finite element model with a computationally more expensive high-fidelity model, given that the solution of the governing elliptic PDE is sufficiently smooth. The solution of the interface problem of the low-fidelity model was shown to provide the training runs upon which the emulators can be built in order to approximate the more expensive solution of the interface problem of a corresponding high-fidelity model. That way, the computational cost of the domain decomposition solution of the high-fidelity model was reduced. A good agreement between the proposed approximation and the direct domain decomposition solution was found for different geometries of the domain. It was shown that neither the domain, or even the partitioning subdomains, need to be convex.

In Chapter 5, the application of GPEs to systems with parametric uncertainty was discussed. A design point generation strategy which takes advantage of the domain structure of a simulator with a random parameter was proposed. Three possible advantages such a strategy might have in a situation when the generation of training runs is expensive were identified:

1. The proposed approach (strategy 2) can reduce the number of evaluations of the simulator, compared both to Monte Carlo simulation and to an approach which employs a single GPE (strategy 1). This is clearly an advantage whenever such evaluations are expensive to carry out.

2. Even if a large number of GPEs had to be built (one for each point in the frequency domain), the cost of doing so was by far less than that of evaluating the simulator repeatedly at few design points in the frequency domain.

3. Besides the decrease in computer processing time, an improvement in accuracy was observed.

The method proposed may therefore be useful for linear dynamical systems with uncertainty.

In Chapter 6, 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 points in the input domain. For these cases, the generation of the training runs was automatic once the relevant simulator was defined and implemented using the Karhunen-Loève expansion. 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 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. Due to this, a novel 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 compute the complete response of a stochastic system using the response values at few points only.

7.2 Summary of Contributions Made

The main contributions of this dissertation can be summarized as follows:

- **Structural dynamic analysis.** GPEs were shown to be an efficient and effective predictive tool for deterministic structural dynamic analysis. It is efficient as the dynamic response of a system can be approximated using only a few evaluations of the original simulator, thus dramatically reducing the CPU time required. It is effective as the approximation was shown to be satisfactory in terms of different adequacy tests (Chapter 3).

- **Domain decomposition.** A novel method for assimilating a deterministic low-fidelity finite element model with a more expensive high-fidelity finite element model was proposed. The method is based on the use of GPEs to solve the interface problem in a domain decomposition algorithm (Chapter 4).

- **Cost reduction of parametric uncertainty.** When parametric uncertainty is taken into account for the calculation of a system response, running a simulator can become much more expensive than the original deterministic simulator. Due to this, a GPE might not alleviate this increase in the computational cost, unless the generation of the training runs is done carefully. A design point generation strategy which takes advantage of the domain structure of a simulator was proposed and compared with an alternative approach and Monte Carlo simulation (Chapter 5).

- **Emulation of random fields.** The emulation of Gaussian and non-Gaussian random fields was shown to be efficient, even for an increasing number of nodes in the finite element mesh. This encouraging exploration led to the emulation of an important random field: the one defined by the stochastic response of a system whose parameters are modeled as random fields (Chapter 6).

- **Emulation of the stochastic response.** A novel method to reduce the computational cost of the polynomial chaos expansion was proposed. An important problem that had to be tackled for this novel algorithm to be implemented was the generation of training runs. The problem was solved by employing a partitioned Cholesky decomposition. The computational complexity proposed algorithm was quantified. An efficient memory management was also discussed (Chapter 6).

7.3 Outlook

There are several research directions that can be followed from the work carried out in this dissertation.

Regarding the use of GPEs in deterministic structural dynamic analysis, there are two potential advantages of using the proposed GPE-based approach. The first is that the computational cost is practically independent of the damping model. This is due to the fact that the cost of solving the governing partial differential equations at the training frequency points does not depend on whether the underlying damping model is viscous, viscoelastic or any other frequency-dependent damping model. The second advantage arises from the fact that once the training frequency points are selected, the governing partial differential equations can be solved at those points in parallel. On top of it, each solution of the linear system can be efficiently parallelized, for example using the conventional domain decomposition methods (Smith *et al.*, 2004; Quarteroni and Valli, 1999; Mathew, 2008), since the dynamic stiffness matrix is highly banded in nature.

Related to the integration of GPEs and domain decomposition methods, future work should include the extension of the proposed methodology to the three dimensional case. Additionally, the extension of the method should include stochastic partial differential equations. Some preliminary work in that direction has been already done by Sarkar *et al.* (2008).

Regarding the use of GPEs in structural dynamic analysis with random parameters, future work should include the extension to the case of more than one random parameter. The challenge for that case is to include the possible correlation between the parameters into the metamodeling, as their independence might be unlikely in certain contexts.

Finally, in the coupling between GPEs and the polynomial chaos expansion, the proposed algorithm could be refined in future research by carrying out a more profound study on how to guarantee that the partitioned Cholesky factor of the permuted matrix is at least as sparse as that of the original stiffness matrix.

7.4 Published Work

The following literature was generated from the present dissertation.

7.4.1 Journal Papers

1. DiazDelaO, F.A. & Adhikari, S. (2010), "Structural dynamic analysis using Gaussian process emulators", *Engineering Computations*, **27** (5) 580-605.

2. DiazDelaO, F.A. & Adhikari, S., "Gaussian process emulators for the stochastic finite element method". *International Journal for Numerical Methods in Engineering.* Accepted for publication.

7.4.2 Conference Papers

1. DiazDelaO, F.A. & Adhikari, S., "Coupling polynomial chaos expansions with Gaussian process emulators: An introduction", *Proceedings of the 27th Internationan Modal Analysis Conference (IMAC-XXVII)*, Orlando, FL, USA, February 9-12, 2009.

2. DiazDelaO, F.A. & Adhikari, S., "Bayesian emulators and the stochastic finite element method", *Proceedings of the Sixth International Conference on Engineering and Computational Technology*, Athens, Greece, September 2-5, 2008.

3. DiazDelaO, F.A. & Adhikari, S., "Bayesian emulator approach for complex dynamical systems", *Proceedings of the 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, Schamburg, IL, USA, April 7-10, 2008.

7.4.3 Book Chapters

1. DiazDelaO, F.A. & Adhikari, S., "Gaussian process emulators for dynamical systems with random parameters", in *Safety, Reliability and Risk of Structures, Infrastructures and Engineering Systems - Furuta, Frangopol & Shinozuka (eds.)*, Taylor & Francis Group, London, 2010.

7.5 Work Under Review

At the time of the submission of this dissertation, the following work has been submitted and is currently under review.

7.5.1 Journal Papers

1. DiazDelaO, F.A. & Adhikari, S., "Gaussian process emulation for uncertain systems". Under review.

2. DiazDelaO, F.A. & Adhikari, S., "Bayesian assimilation of multi-fidelity finite element models". Under review.

Bibliography

Adhikari, S. (2001), "Classical normal modes in non-viscously damped linear systems", *AIAA Journal*, **39** (5), pp. 978–980.

Adhikari, S. (2004), "Optimal complex modes and an index of damping non-proportionality", *Mechanical System and Signal Processing*, **18** (1), pp. 1–27.

Adhikari, S. and Friswell, M. I. (2007), "Random matrix eigenvalue problems in structural dynamics", *International Journal for Numerical Methods in Engineering*, **69** (3), pp. 562–591.

Adhikari, S., M. I. Friswell, K. L., and Sarkar, A. (2009), "Experimental uncertainty quantification in structural dynamics", *Probabilistic Engineering Mechanics*, accepted.

Adhikari, S. and Sarkar, A. (2009), "Uncertainty in structural dynamics: experimental validation of wishart random matrix model", *Journal of Sound and Vibration*, accepted.

Adhikari, S. and Wagner, N. (2004), "Direct time-domain approach for exponentially damped systems", *Computer and Structures*, **82** (29-30), pp. 2453–2461.

Adhikari, S. and Woodhouse, J. (2003), "Quantification of non-viscous damping in discrete linear systems", *Journal of Sound and Vibration*, **260** (3), pp. 499–518.

Au, S. K. and Beck, J. L. (1999), "A new adaptive importance sampling scheme for reliability calculations", *Structural Safety*, **21** (2), pp. 135 – 158.

Babuška, I. and Elman, H. C. (1989), "Some aspects of parallel implementation of the finite-element method on message passing architectures", *J. Comp. Appl. Math.*, **27**, pp. 157–187.

Babuška, I. and Suri, M. (1990), "The p- and h-p versions of the finite element method, an overview", *Computer Methods in Applied Mechanics and Engineering*, **80** (1-3), pp. 5 – 26.

Bastos, L. S. and O'Hagan, A. (2009), "Diagnostics for Gaussian process emulators", *Technometrics*, **51** (4), pp. 425–438.

Bates, R., Kennet, R., Steinberg, D., and Wynn, H. (2006), "Achieving robust design from computer simulations", *Quality Technology & Quantitative Management*, **3** (2), pp. 161–177.

Bathe, K.-J. (1995), *Finite Element Procedures*, Prentice Hall Inc., Englewood Cliffs, New Jersey, USA.

Bendali, A., Boubendir, Y., and Fares, M. (2007), "A FETI-like domain decomposition method for coupling finite elements and boundary elements in large-size problems of acoustic scattering", *Computers & Structures*, **85** (9), pp. 526 – 535.

Bezazi, A., Pierce, S. G., Worden, K., and Harkati, E. H. (2007), "Fatigue life prediction of sandwich composite materials under flexural tests using a Bayesian trained artificial neural network", *International Journal of Fatigue*, **29** (4), pp. 738 – 747.

Biot, M. A. (1958), "Linear thermodynamics and the mechanics of solids", in "Proceedings of the Third U. S. National Congress on Applied Mechanics", ASME, New York, (pp. 1–18).

Bishop, C. M. (1996), *Neural Networks for Pattern Recognition*, Oxford University Press, USA.

Bjørstad, P. E. and Widlund, O. B. (1989), "To overlap or not to overlap: A note on a domain decomposition method for elliptic problems", *SIAM J. Sci. Stat. Comput.*, **10** (5), pp. 1053–1061.

Busby, D. (2009), "Hierarchical adaptive experimental design for Gaussian process emulators", *Reliability Engineering & System Safety*, **94** (7), pp. 1183 – 1193, special Issue on Sensitivity Analysis.

Caughey, T. K. and O'Kelly, M. E. J. (1965), "Classical normal modes in damped linear dynamic systems", *Transactions of ASME, Journal of Applied Mechanics*, **32**, pp. 583–588.

Challenor, P., Hankin, R., and Marsh, R. (2006), *Avoiding Dangerous Climate Change*, Cambridge University Press, Cambridge, UK, chapter Achieving robust design from computer simulations.

Chambers, J., Cleveland, W., Kleiner, B., and Tuckey, P. (1983), *Graphical Methods for Data Analysis*, Champman & Hall.

Chowdhury, R. and Adhikari, S. (2010), "High-dimensional model representation for stochastic finite element analysis", *Applied Mathematical Modelling*, **34** (12), pp. 3917–3932.

Chowdhury, R. and Rao, B. (2009), "Assessment of high dimensional model representation techniques for reliability analysis", *Probabilistic Engineering Mechanics*, **24** (1), pp. 100 – 115.

Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J. (2001), *Concepts and Applications of Finite Element Analysis*, John Wiley & Sons, New York, USA, fourth edition.

Craig, K. J., Stander, N., Dooge, D. A., and Varadappa, S. (2005), "Automotive crashworthiness design using response surface-based variable screening and optimization", *Engineering Computations*, **22** (1), pp. 38–61.

Cressie, N. (1993), *Statistics for Spatial Data*, John Wiley & Sons.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1988), "Bayesian design and analysis of computational experiments", *Technical Report 6498*, Oak Ridge National Laboratory.

Daneshkhah, A. and Bedford, T. (2008), *Advances in Mathematical Modeling for Reliability*, Delftse University Press, The Netherlands, chapter Sensitivity analysis of a reliability system using Gaussian processes.

Debusschere, B., Najm, H., Pebay, P., Knio, O., Ghanem, R., and LeMaitre, O. (2005), "Numerical challenges in the use of polynomial chaos representations for stochastic processes", *SIAM Journal on Scientific Computing*, **22** (2), pp. 698–719.

Deodatis, G. (1991), "The weighted integral method, I : stochastic stiffness matrix", *J. Eng. Mech.*, **117** (8), pp. 1851 – 1864.

Deodatis, G. and Shinozuka, M. (1991), "The weighted integral method, II: response variability and reliablity", *J. Eng. Mech.*, **117** (8), pp. 1865 – 1877.

Der Kiureghian, A. and Li, C.-C. (1986), "Structural reliability under incomplete probability information", *J. Eng. Mech, ASCE*, **112** (1), pp. 85 – 104.

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

Ditlevsen, O. and Marsden, H. (1996), *Structural Reliability Methods*, J. Wiley and Sons, Chichester.

Elishakoff, I. and Ren, Y. J. (2003), *Large Variation Finite Element Method for Stochastic Problems*, Oxford University Press, Oxford, U.K.

Elishakoff, I., Ren, Y. J., and Shinozuka, M. (1995), "Improved finite-element method for stochastic problems", *Chaos Solitons & Fractals*, **5** (5), pp. 833–846.

Ewins, D. J. (2000), *Modal Testing: Theory and Practice*, Research Studies Press, Baldock, England, second edition.

Falsone, G. and Impollonia, N. (2002), "A new approach for the stochastic analysis of finite element modelled structures with uncertain parameters", *Computer Methods in Applied Mechanics and Engineering*, **191** (44), pp. 5067–5085.

Falsone, G. and Impollonia, N. (2003), "A new approach for the stochastic analysis of finite element modelled structures with uncertain parameters (vol 191, pg 5067, 2002)", *Computer Methods in Applied Mechanics and Engineering*, **192** (16-18), pp. 2187–2188.

Fan, H., Lampinen, J., and Levy, Y. (2006), "An easy-to-implement differential evolution approach for multi-objective optimizations", *Engineering Computations*, **23** (2), pp. 124–138.

Faravelli, L. (1989), "Response surface approach for reliability analysis", *Journal of Engineering Mechanics*, **115** (12), pp. 2763–2781.

Feng, Y., Li, C., and Owen, D. (2010), "A directed Monte Carlo solution of linear stochastic algebraic system of equations", *Finite Elements in Analysis and Design*, **46** (6), pp. 462 – 473.

Forrester, A. I. J., Sóbester, A., and Keane, A. J. (2008), *Engineering Design via Surrogate Modelling. A Practical Guide*, J. Wiley and Sons, Chichester, U. K.

Gallimard, L. and Sassi, T. (2010), "A posteriori error analysis of a domain decomposition algorithm for unilateral contact problem", *Computers & Structures*, **88** (13-14), pp. 879 – 888.

George, A. (1981), *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, New Jersey, USA.

George, P. (1991), *Automatic Mesh Generation: Application to Finite Element Methods*, Wiley, New York.

Ghanem, R. and Spanos, P. (1991), *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, USA.

Glowinski, R. and Wheeler, M. F. (1987), "Domain decomposition and mixed finite element methods for elliptic problems", *Technical Report 87-11*, Rice University.

Glowinsky, Golub, and Periaux (Eds.) (1987), *first international symposium on domain decomposition methods*, siam, Philadelphia, PA, held in Paris, January 1987.

Guibert, D. and Tromeur-Dervout, D. (2007), "Parallel adaptive time domain decomposition for stiff systems of ODEs/DAEs", *Computers & Structures*, **85** (9), pp. 553 – 562.

Haldar, A. and Mahadevan, S. (2000), *Reliability Assessment Using Stochastic Finite Element Analysis*, John Wiley and Sons, New York, USA.

Hankin, R. K. S. (2005), "Introducing BACCO, an R bundle for bayesian analysis of computer code output", *Journal of Statistical Software*, **14** (16), pp. 1–21.

Haukaas, T. and Der Kiureghian, A. (1999), "FERUM: Finite element reliability using Matlab", `http://www.ce.berkeley.edu/projects/ferum/`.

Haylock, R. (1996), *Bayesian Inference About Outputs of Computationally Expensive Algorithms with Uncertainty on the Inputs*, Ph.D. thesis, University of Nottingham, Nottingham, UK.

Haylock, R. and O'Hagan, A. (1996), *Bayesian Statistics 5*, Oxford University Press, Oxford, UK, chapter On inference for outputs of computationally expensive algorithms with uncertainty on the inputs.

Higham, N. J. (2002), "The Matrix Computation Toolbox", `http://www.maths.manchester.ac.uk/~higham/mctoolbox/`.

Hughes, T. J. R. (2000), *The Finite Element Method : Linear Static and Dynamic Finite Element Analysis*, Dover Publications, New York, USA.

Hurtado, J. E. and Barbat, A. H. (1998), "Monte carlo techniques in computational stochastic mechanics", *Archives of Computational Methods in Engineering*, **5** (1), pp. 3–29.

Impollonia, N. and Muscolino, G. (2002), "Static and dynamic analysis of non-linear uncertain structures", *Meccanica*, **37** (1-2), pp. 179–192.

Impollonia, N. and Ricciardi, G. (2006), "Explicit solutions in the stochastic dynamics of structural systems", *Probabilistic Engineering Mechanics*, **21** (2), pp. 171–181.

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990), "Minimax and maximin distance designs", *Journal of Statistical Planning and Inference*, **26** (2), pp. 131–148.

Karhunen, K. (1946), "Über lineare methoden in der wahrscheinlichkeitsrechnung", *Annales Academiae Scientiarum Fennicae*, (37).

Keane, A. and Nair, P. (2005), *Computational Approaches for Aerospace Design*, John Wiley & Sons, Chichester, UK.

Kennedy, M. C., Anderson, C. W., Conti, S., and O'Hagan, A. (2006), "Case studies in Gaussian process modelling of computer codes", *Reliability Engineering & System Safety*, **91** (10-11), pp. 1301–1309.

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

Kiureghian, A. D. and Ke, J.-B. (1988), "The stochastic finite element method in structural reliability", *Probabilistic Engineering Mechanics*, **3** (2), pp. 83 – 91.

Kleiber, M. and Hien, T. D. (1992), *The Stochastic Finite Element Method*, John Wiley, Chichester.

Kleijnen, J. P. C. (2009), "Kriging metamodeling in simulation: a review", *European Journal of Operational Research*, **192** (3), pp. 707–716.

Kolachalama, V., Bressloff, N., and Nair, P. (2007), "Mining data from hemodynamic simulations via Bayesian emulation", *BioMedical Engineering OnLine*, **6** (47).

Krige, D. G. (1951), "A statistical approach to some basic mine valuation problems on the Witwatersrand", *Journal of the Chemical, Metallurgical and Mining Engineering Society of South Africa*, **52** (6), pp. 119 – 139.

Krzanowski, W. (2000), *Principles of Multivariate Analysis*, Oxford University Press, Oxford, UK.

Li, C., Feng, Y., and Owen, D. (2006), "Explicit solution to the stochastic system of linear algebraic equations $(\alpha_1 A_1 + \alpha_2 A_2 + \ldots + \alpha_m A_m)x = b$", *Computer Methods in Applied Mechanics and Engineering*, **195** (44-47), pp. 6560 – 6576.

Li, C.-C. and Der Kiureghian, A. (1993), "Optimal discretization of random fields", *J. Eng. Mech*, **119** (6), pp. 1136 – 1154.

Liu, W. K., Belytschko, T., and Mani, A. (1986), "Probabilistic finite elements for nonlinear structural dynamics", *Computer Methods in Applied Mechanics and Engineering*, **56** (1), pp. 61 – 81.

Loève, M. (1948), *Processus Stochastic et Mouvement Brownien*, Gauthier Villars, Paris, chapter Fonctions aleatoires du second ordre.

Maia, N. M. M. and Silva, J. M. M. (Eds.) (1997), *Theoretical and Experimental Modal Analysis*, Engineering Dynamics Series, Research Studies Press, Taunton, England, series Editor, J. B. Robetrs.

Maia, N. M. M., Silva, J. M. M., and Ribeiro, A. M. R. (1998), "On a general model for damping", *Journal of Sound and Vibration*, **218** (5), pp. 749–767.

Manohar, C. S. and Keane, A. J. (1994), *Statistical Energy Analysis. An Overview with Applications in Structural Dynamics*, Cambridge University Press, Cambridge, UK, chapter Statistics of energy flows in spring-coupled one-dimensional sub-systems.

Marrel, A., Iooss, B., Laurent, B., and Roustant, O. (2009), "Calculations of Sobol indices for the Gaussian process metamodel", *Reliability Engineering & System Safety*, **94** (3), pp. 742 – 751.

Mathew, T. (2008), *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, Springer Verlag Berlin and Heidelberg GmbH & Co.v, Berlin, Germany.

Matthies, H. G., Brenner, C. E., Bucher, C. G., and Soares, C. G. (1997), "Uncertainties in probabilistic numerical analysis of structures and solids - stochastic finite elements", *Structural Safety*, **19** (3), pp. 283–336.

McFarland, J. and Mahadevan, S. (2008), "Multivariate significance testing and model calibration under uncertainty", *Computer Methods in Applied Mechanics and Engineering*, **197** (29-32), pp. 2467 – 2479.

McKay, M., Conover, W., and Beckman, R. (1979), "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", *Technometrics*, **21** (2), pp. 239–245.

Morris, M. D. and Mitchell, T. J. (1995), "Exploratory designs for computational experiments", *Journal of Statistical Planning and Inference*, **43** (3), pp. 381 – 402.

Muscolino, G., Ricciardi, G., and Impollonia, N. (2000), "Improved dynamic analysis of structures with mechanical uncertainties under deterministic input", *Probabilistic Engineering Mechanics*, **15** (2), pp. 199–212.

Nair, P. B. and Keane, A. J. (2002), "Stochastic reduced basis methods", *AIAA Journal*, **40** (8), pp. 1653–1664.

Neal, R. (1998), *Bayesian Statistics 6*, Oxford University Press, Oxford, UK, chapter Regression and Classification Using Gaussian Process Priors.

Newland, D. E. (1989), *Mechanical Vibration Analysis and Computation*, Longman, Harlow and John Wiley, New York.

Nouy, A. (2009), "Recent developments in spectral stochastic methods for the numerical solution ofstochastic partial differential equations", *Archives of Computational Methods in Engineering*, **16** (3), pp. 251–285.

Oakley, J. E. and O'Hagan, A. (2004), "Probabilistic sensitivity analysis of complex models: a Bayesian approach", *Journal of the Royal Statistical Society B*, **66** (3), pp. 751–769.

O'Hagan, A. (2006), "Bayesian analysis of computer code outputs: a tutorial", *Reliability Engineering & System Safety*, **91** (10-11), pp. 1290–1300.

Papadrakakis, M. and Papadopoulos, V. (1996), "Robust and efficient methods for stochastic finite element analysis using monte carlo simulation", *Computer Methods in Applied Mechanics and Engineering*, **134** (3-4), pp. 325–340.

Pepelyshev, A. and Oakley, J. (2009), "On the choice of correlation function and cross-validation for Gaussian processes", *Technical Report MS 3.3c.2*, The University of Sheffield.

Pérez, V. M., Renaud, J. E., and Watson, L. E. (2008), "Reduced sampling for construction of quadratic response surface approximations using adaptive experimental design", *Engineering Computations*, **25** (8), pp. 764–782.

Petyt, M. (1998), *Introduction to Finite Element Vibration Analysis*, Cambridge University Press, Cambridge, UK.

Pierce, S., Worden, K., and Manson, G. (2006), "A novel information-gap technique to assess reliability of neural network-based damage detection", *Journal of Sound and Vibration*, **293** (1-2), pp. 96 – 111.

Pierce, S. G., Worden, K., and Bezazi, A. (2008), "Uncertainty analysis of a neural network used for fatigue lifetime prediction", *Mechanical Systems and Signal Processing*, **22** (6), pp. 1395 – 1411.

Quarteroni, A. and Valli, A. (1999), *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, UK.

Rao, A. R. M., Rao, T. V. S. R. A., and Dattaguru, B. (2003), "A new parallel overlapped domain decomposition method for nonlinear dynamic finite element analysis", *Computers & Structures*, **81** (26-27), pp. 2441 – 2454.

Rao, B. and Chowdhury, R. (2009), "Enhanced high dimensional model representation for reliability analysis", *Int. J. Numer. Methods Eng.*, **77** (5), p. 719750.

Rayleigh, L. (1877), *Theory of Sound (two volumes)*, Dover Publications, New York, 1945th edition.

Rocha, H. (2009), "On the selection of the most adequate radial basis function", *Applied Mathematical Modelling*, **33** (3), pp. 1573 – 1583.

Rougier, J. (2007), "Probabilistic inference for future climate using an ensemble of climate model evaluations", *Climatic Change*, **81** (3), pp. 247–264.

Rougier, J., Sexton, D., Murphy, M., and Stainforth, D. (2007), "Emulating the sensitivity of the HadSM3 climate model using ensembles from different but related experiments", *Technical Report 07/04, MUCM*, University of Sheffield, Sheffield, UK.

Sachdeva, S. K., Nair, P. B., and Keane, A. J. (2006a), "Comparative study of projection schemes for stochastic finite element analysis", *Computer Methods in Applied Mechanics and Engineering*, **195** (19-22), pp. 2371–2392.

Sachdeva, S. K., Nair, P. B., and Keane, A. J. (2006b), "Hybridization of stochastic reduced basis methods with polynomial chaos expansions", *Probabilistic Engineering Mechanics*, **21** (2), pp. 182–192.

Sacks, J., Welch, W., Mitchell, T., and Wynn, H. (1989), "Design and analysis of computer experiments", *Statistical Science*, **4** (4), pp. 409–435.

Santner, T., Williams, B., and Notz, W. (2003), *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, London, UK.

Sarkar, A., Benabbou, N., and Ghanem, R. (2008), "Domain decomposition of stochastic pdes: parallel performance study", *International Journal of High Performance Computing Applications*, under review.

Scheidt, J. V. and Purkert, W. (1983), *Random Eigenvalue Problems*, North Holland, New York.

Schuller, G. I., Bucher, C. G., Bourgund, U., and Ouypornprasert, W. (1989), "On efficient computational schemes to calculate structural failure probabilities", *Probabilistic Engineering Mechanics*, **4** (1), pp. 10 – 18.

Schuller, G. I., Pradlwarter, H. J., and Koutsourelakis, P. S. (2004), "A critical appraisal of reliability estimation procedures for high dimensions", *Probabilistic Engineering Mechanics*, **19** (4), pp. 463 – 474.

Schwarz, H. A. (1890), *Gesammelte Mathematische Abhandlungen*, Springer, Berlin, volume 2, (pp. 133–143), first published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, volume 15, 1870, pp. 272–286.

Shewry, M. and Wynn, H. (1987), "Maximum entropy sampling", *J. Appl. Statist.*, **14**, pp. 165–170.

Shinozuka, M. and Nomoto, T. (1980), "Response variability due to spatial randomness of material properties", *Technical report*, Department of Civil Engineering, University Of Columbia, New York.

Shinozuka, M. and Yamazaki, F. (1998), "Stochastic finite element analysis: an introduction", in "Stochastic structural dynamics: Progress in theory and applications", edited by S. T. Ariaratnam, G. I. Schuëller, and I. Elishakoff, Elsevier Applied Science, London.

Silva, J. M. M. and Maia, N. M. M. (Eds.) (1998), *Modal Analysis and Testing: Proceedings of the NATO Advanced Study Institute*, NATO Science Series: E: Applied Science, Sesimbra, Portugal.

Smith, B., Bjorstad, P., and Gropp, W. (2004), *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK.

Storaasli, O. O. and Sobieszczanski-Sobieski, J. (1974), "On the accuracy of Taylor approximation for structure resizing", *American Institute of Aeronautics and Astronautics (AIAA) Journal*, (12), pp. 231–233.

Sudret, B. and Der-Kiureghian, A. (2000), "Stochastic finite element methods and reliability", *Technical Report UCB/SEMM-2000/08*, Department of Civil & Environmental Engineering, University Of California, Berkeley.

Sultan, I. A. (2007), "A surrogate model for interference prevention in the limaçon-to-limaçon machines", *Engineering Computations*, **24** (5), pp. 437–449.

Thomke, S., Holzner, M., and Gholami, T. (1999), "The crash in the machine", *Scientific American*, **280** (3), pp. 92–97.

Toal, D. J. J., Bressloff, N. W., Keane, A. J., and Holden, C. M. E. (2011), "The development of a hybridized particle swarm for kriging hyperparameter tuning", *Engineering Optimization*, to appear.

Torvik, P. J. and Bagley, R. L. (1987), "Fractional derivatives in the description of damping: materials and phenomena", in "The Role of Damping in Vibration and Noise Control", ASME DE-5, (pp. 125–135).

Toselli, A. and Windlund, O. (2005), *DomainDecomposition Methods Algorithms and Theory*, Springer Series in Computational Mathematics, Berlin.

Vanmarcke, E.-H. and Grigoriu, M. (1983), "Stochastic finite element analysis of simple beams", *J. Eng. Mech, ASCE*, **109** (5), pp. 1023 - 1214.

Wagner, N. and Adhikari, S. (2003), "Symmetric state-space formulation for a class of non-viscously damped systems", *AIAA Journal*, **41** (5), pp. 951–956.

Wan, X. L. and Karniadakis, G. E. (2006), "Beyond wiener-askey expansions: Handling arbitrary pdfs", *Journal of Scientific Computing*, **27** ((-3), pp. 455–464.

Wiener, N. (1938), "The homogeneous chaos", *American Journal of Mathematics*, **60** (4), pp. 897 - 936.

Woodhouse, J. (1998), "Linear damping models for structural vibration", *Journal of Sound and Vibration*, **215** (3), pp. 547–569.

Xiu, D. B. and Karniadakis, G. E. (2002), "The wiener-askey polynomial chaos for stochastic differential equations", *Siam Journal on Scientific Computing*, **24** (2), pp. 619–644.

Xiu, D. B. and Karniadakis, G. E. (2003), "Modeling uncertainty in flow simulations via generalized polynomial chaos", *Journal of Computational Physics*, **187** (1), pp. 137–167.

Yamazaki, F., Shinozuka, M., and Dasgupta, G. (1988), "Neumann expansion for stochastic finite element analysis", *Journal of Engineering Mechanics, ASCE*, **114** (8), pp. 1335–1354.

Zhang, J. and Ellingwood, B. (1993), "Orthogonal series expansion of random fields in reliability analysis", *J. Eng. Mech.*, **120** (12), pp. 2660 - 2677.

Zhao, M., Huang, Z., and Chen, L. (2008), "Multidisciplinary design optimization of tool head for heavy duty cnc vertical turning mill", *Engineering Computations*, **25** (7), pp. 657–676.

Zienkiewicz, O. C. and Taylor, R. L. (1991), *The Finite Element Method*, McGraw-Hill, London, fourth edition.