The University Of Sheffield.

Access to Electronic Thesis

Validating Gaussian Process Models in Computer Experiments

Leonardo Soares Bastos

Thesis submitted to the University of Sheffield
for the degree of Doctor of Philosophy

Department of Probability and Statistics

June 2010

University of Sheffield

To Thaís.

Acknowledgements

I am heartily thankful to my supervisors, Jeremy Oakley and Tony O'Hagan, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. I thank all MUCMers for the technical discussions on almost every topic associated with emulation, in particular I would like to acknowledge the help of Jonty Rougier for his support. Jo Green is thanked for her assistance with all University-related problems.

I am grateful to all my friends from Sheffield and Coventry for being my surrogate family during the years I stayed there and for their continued moral support there after. I also would like to thank some friends from Brazil that despite the distance were all the time by my side.

Finally, I am forever indebted to Thaís for her understanding, endless patience and encouragement when it was most required. I am also heartily grateful to Afzalina, Flávio, Maria and Tom for their support.

Leo Bastos, June 2010

Abstract

In this thesis we present a methodology for validating Gaussian process models: Gaussian process emulators and simulator discrepancy models. A Gaussian process emulator is a representation of our beliefs about a mathematical model implemented in a computer program known as a simulator. By "simulator discrepancy", we mean the difference between a simulator's output and the corresponding physical process. We present a set of diagnostics to validate and assess the adequacy of Gaussian process models. These diagnostics are based on comparisons between real observations and model predictions for some test data, known as validation data, defined by a sample of real observations not used to build the model. The validation data are chosen according to designs that we have developed for such purposes. The diagnostics for Gaussian process emulators and discrepancy models are useful tools during the modelling procedure. Based on the result of the diagnostics, we can identify problems such as underconfidence, overconfidence, poor estimation of some unknown parameters, which if not identified might compromise analyses using the Gaussian process models. After we identify a problem, the diagnostics may provide information on where we should collect more data in order to make the predictive model a better representation of our beliefs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Simulators

Simulators, also known as computer models, are mathematical representations of a real system implemented in a computer. The simulator, represented by $\eta(\cdot)$, is assumed to be a function of a set of inputs denoted by $\mathbf{x} = (x_1, \ldots, x_p) \in \chi \subset \mathbb{R}^p$, with output represented by $y \in \mathbb{R}$. A computer experiment is a set of simulator runs at different values of inputs, $(y_1 = \eta(\mathbf{x}_1), \ldots, y_n = \eta(\mathbf{x}_n))$. Computer experiments have been used to investigate real-world systems in almost all fields of science and technology. There are some situations where computer experiments are feasible but physical experiments are impossible. For example, the number of inputs may be too large to perform physical experiments, the cost to perform the physical experiment may be too high, or the physical experiment may be an unethical experiment.

The use of computer experiments dates back to the 1940s at Los Alamos National Laboratory in the study of the behaviour of nuclear weapons. In 1944[1], there was a quantitative investigation of the hydrodynamics of a nuclear implosion where IBM machine calculations were used to solve partial differential equations of implosion hydrodynamics.

[1] Chapter 4, Los Alamos: Technical Review to August 1944, available at `http://www.fas.org/sgp/othergov/doe/lanl/00795708.pdf`

Examples of scientific and technological developments that have been conducted using computer experiments are many and growing. In climate science, Zickfeld et al. (2004) present a low-order model of the Atlantic thermohaline circulation which is able to reproduce many features of the behaviour of coupled ocean-atmosphere circulation models such as the sensitivity of the thermohaline circulation to the amount, regional distribution and rate of climate change. Randall et al. (2007) evaluate the capabilities and limitations of global climate models used in the IPCC (Intergovernmental Panel on Climate Change). In cosmology, Benson et al. (2001) use a complex computer model to understand the processes responsible for the formation and evolution of the galaxies. In fire protection engineering, McGrattan et al. (2007) present a Fire Dynamics Simulator (FDS) which is a computational fluid dynamics model of fire-driven fluid flow. The simulator solves numerically a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires.

1.2 Emulators

Simulators can be extremely expensive to run, for example if the mathematical model is very complex, or if the required precision is very high. Therefore, the simulator is run at a limited number of inputs. The output of the simulator at untried inputs has to be predicted. Statistics plays an important role in computer modelling, providing predictions of the simulator at any given configuration of simulator inputs. Sacks et al. (1989b) provide a good description of prediction and design problems for computer experiments. A statistical representation of a simulator is known as a statistical emulator, or simply emulator. For any given configuration of input values for the simulator, the emulator provides a probabilistic prediction of the output that the simulator would produce if it were run at those inputs. Furthermore, for any set of input configurations, the emulator will provide a joint probabilistic prediction of the corresponding set of simulator outputs.

An emulator is a probability distribution that represents the simulator, where the simulator is viewed as an unknown mathematical function. Although the computer code is known,

its complexity allows $\eta(\cdot)$ to be considered an unknown function. From a Bayesian point of view Kimeldorf and Wahba (1970) and O'Hagan (1978) use Gaussian processes to describe the behaviour of an unknown mathematical function. In the 1980s, the fundamental idea of building a statistical emulator using Gaussian processes was introduced, in a non-Bayesian framework by Sacks et al. (1989b), and by Currin et al. (1988, 1991) within a Bayesian framework.

The probabilistic predictions of the simulator may take one of two forms depending on the approach used to build the emulator. In the fully Bayesian approach, the predictions are complete probability distributions (O'Hagan 2006; Kennedy et al. 2006). In the Bayes linear approach, probability distributions are not fully specified, but instead work with first and second order moments (Craig et al. 2001; Goldstein and Rougier 2006). In this thesis, we use the full Bayesian approach.

In order to build a Gaussian process emulator, the uncertainty about the simulator output is described as a Gaussian process with a particular mean function $m(\cdot)$, and a covariance function $V(\cdot,\cdot)$. If $\eta(\cdot)$ has a Gaussian process distribution then for every $n = 1, 2, \ldots$ the joint distribution of $\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)$ is multivariate normal for all $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \chi$. The mean function $m(\cdot)$ can be any function of $\mathbf{x} \in \chi$, but $V(\cdot,\cdot)$ must satisfy the property that every covariance matrix with elements $\{v(\mathbf{x}_i, \mathbf{x}_j)\}$ must be non-negative definite.

1.3 The need to validate an emulator

The Gaussian process emulators are indeed flexible models to represent our uncertainty about the simulator. However, a Gaussian process emulator can give poor predictions of the simulator outputs for at least two reasons. First, the assumption of a stationary Gaussian process with particular mean and covariance structures may be inappropriate. Second, even if these assumptions are reasonable there are various parameters to be estimated, and a bad or unfortunate choice of training dataset may suggest inappropriate values for these parameters. In the case of the correlation length parameters, where we condition on fixed

estimates rather than integrating over the full posterior distribution, we may also make a poor choice of estimate to condition on. Therefore, before using an emulator as a surrogate of a simulator, the assumptions made to build the emulator should be checked. The process of checking the Gaussian process assumptions is called the validation process.

Although Gaussian process emulators are widely used as surrogate of simulators, little is done to validate the emulators. A non-valid emulator can lead to wrong conclusions about the simulator outputs at untried inputs. For instance, it would be undesirable if a modeller uses a non-valid emulator to learn about a physical system that the simulator was intended to represent. Therefore, it is necessary to check whether or not the assumptions made to build the emulator are reasonable. In case of a failure of any assumption, the diagnostics in the validation procedure should be able to provide information in order to help the modeller to rebuild a valid emulator. The main contribution of this thesis is to present a set of diagnostics to be included in the validation procedure of a Gaussian process emulator.

1.4 Outline of the thesis

The main focus lies on developing diagnostic tools to check whether a Gaussian process emulator can represent properly our uncertainty about simulator outputs at untried inputs. In Chapter 2, we review how to build a Gaussian process emulator and discuss some analyses that can be done with an emulator such as uncertainty analysis, sensitivity analysis and calibration.

There is an extensive literature on validation of simulators. Simulators make imprecise statements about physical systems. This can be due to simplifications made in the physical theory or approximations to solutions of complex systems. Therefore, a simulator should be validated. In Chapter 3, we review validation methods for simulators in order to see if any ideas can help develop validation methods for emulators.

In Chapter 4, we present a procedure for validating Gaussian process emulators. The validation procedure contains a set of numerical and graphical diagnostics that provides

information to judge the validity of a Gaussian process emulator.

One important step to build and validate an emulator is the choice of input points where we should run the simulator. Choosing the inputs is called the design problem. In Chapter 5, we discuss the design problem. We review how to choose inputs to efficiently build an emulator. We present some designs for choosing inputs to validate an emulator. We are interested in designs that efficiently distinguish between good and bad emulators.

There are several ways in which an emulator can differ from another. We call competing emulators different emulators for the same simulator. Though we are interested in valid emulators, valid emulators are not unique, and hence it is necessary to have some criteria to rank competing emulators. In Chapter 6, we discuss some methods for comparing competing emulators. The proposed comparison methods are based on the Bayes Factor and some scoring rules.

In Chapter 7, we consider a model to predict a real system, combining a simulator with experimental data. We assume that the difference between the real system outputs and the simulator outputs is a smooth function called here the discrepancy function. We present methods of inference to predict the real system assuming a Gaussian process for the discrepancy function and different ways to deal with the unknown parameters. We extend the diagnostics for Gaussian process emulators to validate discrepancy function models.

Finally, in Chapter 8 we provide discussion and some possibilities for future work related to this thesis.

Chapter 2

Statistical analysis for simulators using emulators

2.1 Introduction

In this chapter, we consider the use of emulators as surrogates for simulators. An emulator is used to provide probabilistic predictions of the simulator at untried inputs. We present the main steps of a statistical analysis for a simulator. Our approach is based on a full Bayesian approach, where we represent our uncertainty about the outputs of a simulator using an emulator.

Simulators are usually deterministic input-output models, where running the simulator again at the same input values will always give the same outputs. However, the output value is unknown before running the simulator for a particular input set. From a Bayesian perspective, uncertainty about the output of the simulator, also called code uncertainty, can be expressed by a stochastic process. The result is a statistical representation of the simulator, known as emulator.

In Section 2.2, we formally show the emulator approach, reviewing the Gaussian process emulator with all required assumptions for building it. In Section 2.3, we demonstrate the

Gaussian process emulator with some toy examples. In Section 2.4, we discuss some analyses that can be done using an emulator as a surrogate of the simulator.

2.2 Gaussian process emulators

An emulator is a probability distribution that represents uncertainty about the simulator, where the simulator is viewed as an unknown mathematical function. Although the simulator is in principle known, we are uncertain about the actual value of the simulator output for any untried input value. From a Bayesian point of view Kimeldorf and Wahba (1970) and O'Hagan (1978) use Gaussian processes to describe the behaviour of an unknown mathematical function. In the 1980s, the fundamental idea of building a emulator using Gaussian processes was introduced, in a non-Bayesian framework by Sacks et al. (1989b), and within a Bayesian framework by Currin et al. (1988, 1991). We review the principal ideas of the Gaussian process emulator from a Bayesian point of view. For a frequentist point of view, see also Santner et al. (2003, section 3.3).

The simulator, represented by $\eta(\cdot)$, is assumed to be a function of a set of inputs denoted by $\mathbf{x} = (x_1, \ldots, x_p) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_p = \mathcal{X} \subset \mathbb{R}^p$, with output represented by $y = \eta(\mathbf{x}) \in \mathbb{R}$. In order to build a Gaussian process emulator, the uncertainty about the simulator output is described as a Gaussian process with a particular mean function $m(\cdot)$, and a covariance function $V(\cdot, \cdot)$. Formally, if $\eta(\cdot)$ is represented by a Gaussian process then for every $n = 1, 2, \ldots$ the joint distribution of $\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)$ is multivariate normal for all $\mathbf{x}_i \in \mathcal{X}$ and $i = 1, 2, \ldots, n$. The mean function $m(\cdot)$ can be any function of $\mathbf{x} \in \mathcal{X}$, but $V(\cdot, \cdot)$ must satisfy the property that every covariance matrix with elements $\{V(\mathbf{x}_i, \mathbf{x}_j)\}$ must be non-negative definite.

Our prior beliefs about the simulator $\eta(\cdot)$ are represented by a Gaussian process with mean $m_0(\cdot)$ and covariance function $V_0(\cdot, \cdot)$. Using a hierarchical formulation,

$$\eta(\cdot)|\beta, \sigma^2, \delta \sim GP\left(m_0(\cdot), V_0(\cdot, \cdot)\right), \tag{2.1}$$

where the mean function $m_0(\cdot)$ is given by

$$m_0(\mathbf{x}) \quad = \quad h(\mathbf{x})^T\beta, \tag{2.2}$$

where $h(\cdot) : \chi \subset \mathbb{R}^p \longmapsto \mathbb{R}^q$ is a known function of the inputs, where q can be different from the input space dimension p, and β is an unknown q-dimensional vector of coefficients. The function $h(\cdot)$ should be chosen to incorporate any prior knowledge about the form of $\eta(\cdot)$.

The choice for $h(\cdot)$ depends on prior beliefs about the simulator. The simplest case is when $q = 1$ and $h(\mathbf{x}) = 1$ for all x. Then the mean function is $m(\mathbf{x}) = \beta$, where now β is a scalar hyperparameter representing an unknown overall mean for the simulator output. This choice expresses no prior knowledge about how the output will respond to variation in the inputs. Another simple instance is when $h(\mathbf{x})^T = (1, \mathbf{x}^T)$, so that $q = 1 + p$, where p is the number of inputs. Then $m(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$, which expresses a prior expectation that the simulator output will show a trend in response to each of the inputs, but there is no prior information to suggest any specific non-linearity in those trends. If there is prior belief in non-linearity of response, then quadratic or higher polynomial terms might be introduced into $h(\cdot)$. In this thesis, unless it is said to the contrary, our prior beliefs for the simulator are represented by the linear mean $m(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$.

The covariance function $V_0(\cdot, \cdot)$ is given by

$$V_0(\mathbf{x}, \mathbf{x}') \quad = \quad \sigma^2 C_\delta(\mathbf{x}, \mathbf{x}'), \tag{2.3}$$

where σ^2 is an unknown scale parameter, and $C_\delta(\cdot, \cdot)$ is a known correlation function with the unknown vector of correlation parameters δ. The chosen correlation function $C_\delta(\cdot, \cdot)$ should ensure that the covariance matrix of any set of inputs is non-negative definite.

Correlation functions

A correlation function $C_\delta(\mathbf{x}, \mathbf{x}')$ represents the correlation between the simulator outputs at the input $\mathbf{x}$ and $\mathbf{x}'$ with correlation parameter δ. In statistical analysis for simulators is common to use stationary correlation functions. We say a correlation function $C(\cdot, \cdot)$ is stationary if $C(\mathbf{x}, \mathbf{x}') = R(\mathbf{x} - \mathbf{x}')$, and it is said to be isotropic if $C(\mathbf{x}, \mathbf{x}') = R(||\mathbf{x} - \mathbf{x}'||)$.

A family of stationary correlation functions which is widely used in the literature to specify a Gaussian process is the power exponential correlation function,

$$C_\delta(\mathbf{x}, \mathbf{x}') = \exp\{-|(\mathbf{x} - \mathbf{x}')/\delta_1|^{\delta_2}\}, \quad \text{for } \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \tag{2.4}$$

where $\delta_1 > 0$, and the allowable range for δ_2 to ensure positive definiteness of the corresponding correlation matrices is $\delta_2 \in (0, 2]$. If $\delta_2 = 2$ then $C_\delta(\cdot, \cdot)$ is a squared exponential correlation function, also known as Gaussian correlation function. An extension of this family is a p-dimensional separable version of the power exponential correlation function.

$$C_\delta(\mathbf{x}, \mathbf{x}') = \exp\left\{-\sum_{i=1}^{p} |(x_i - x_i')/\delta_i|^{\delta_{p+i}}\right\}, \tag{2.5}$$

where $\delta_i > 0$, and $\delta_{p+i} \in (0, 2]$ for $i = 1, \ldots, p$. As a special case, the p-dimensional separable version of the squared exponential correlation function is

$$C_\delta(\mathbf{x}, \mathbf{x}') = \exp\left\{-\sum_{i=1}^{p} |(x_i - x_i')/\delta_i|^2\right\}, \tag{2.6}$$

where the parameters $\delta = (\delta_1, \ldots, \delta_p)$ are called correlation length parameters. This formula shows the role of each correlation length parameter δ_i. The smaller its value, the closer together $\mathbf{x}_i$ and $\mathbf{x}_i'$ must be in order for the outputs at $\mathbf{x}$ and $\mathbf{x}'$ to be highly correlated. Large (small) values of δ_i mean that the output values are correlated over a wide (narrow) range of the ith input x_i. Some authors use a different parametrization for the correlation parameters, for example $\theta_i = \delta_i^{-\frac{1}{2}}$, where θ_i is called a roughness parameter.

Other correlation functions, such as linear and cubic correlation functions are given in

Currin et al. (1991). For more details of correlation functions for Gaussian processes, see Cressie (1993), Santner et al. (2003), and Rasmussen and Williams (2006). In this thesis, unless it is said to the contrary, we use the p-dimensional separable version of the squared exponential correlation function (2.6) with correlation parameters described as correlation lengths.

Training data

Suppose $\mathbf{y} = (y_1 = \eta(\mathbf{x}_1), \ldots, y_n = \eta(\mathbf{x}_n))^T$ contains n values of the simulator outputs at design points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in the input space $\chi \subset \mathbb{R}^p$. These data are called the **training dataset**. The design points are chosen attempting to cover the whole input space. Chapman et al. (1994) suggest that the sample size of the training data should be at least ten times the dimensionality of the input space, i.e. $n = 10p$. Loeppky et al. (2009) illustrate that $n = 10p$ is a reasonable rule for initial experiments.

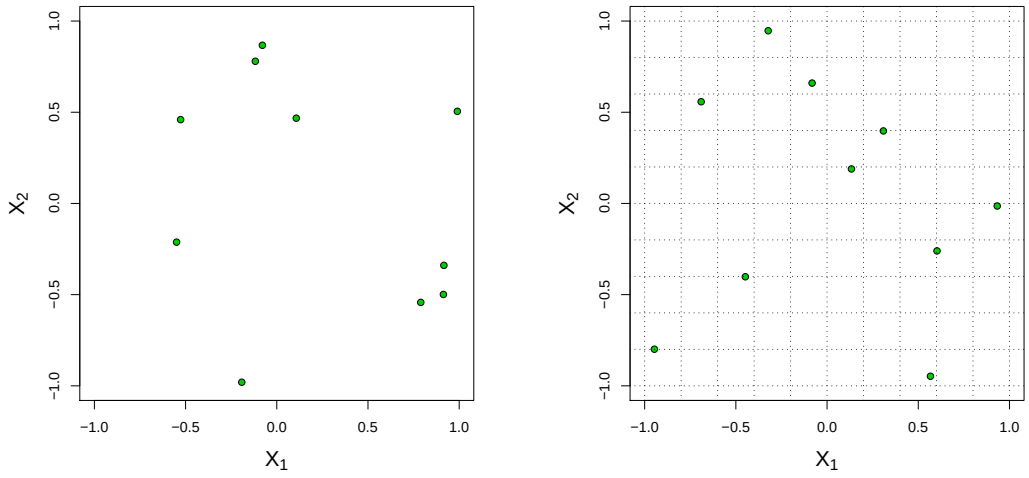
Design for building emulators

In the process of building an emulator, we need to answer the following question: *Which values in the input space should we run the simulator at in order to minimize our uncertainty with respect to the simulator?* To answer this question, several designs for computer models have been developed.

The simplest design is a random sample from a uniform distribution over the input space. Figure 2.1 (a) illustrates the simple random design of size 10 in a region $\mathcal{X} = [-1, 1]^2$. The problem with this method is that some regions of the input space can be completely uncovered. In our example, Figure 2.1 (a), we see that small values of X_1 are not covered.

McKay et al. (1979) propose Latin hypercube sampling for choosing inputs for a simulator. Latin hypercube sampling guarantees that each input is well represented in the design. Figure 2.1 (b) illustrates a Latin hypercube design of size 10 from the region $\mathcal{X} = [-1, 1]^2$. We see

that marginally, both input variable sample spaces are well covered.



(a) (b)

Figure 2.1: Sampling-based designs with 10 elements and 2 dimensions on the region $\mathcal{X} = [-1, 1]^2$: (a) Uniform random sample; (b) Latin hypercube sample.

In Chapter 5, we discuss the design problem in more detail and present other designs for building emulators such as distance-based designs, optimal designs, lattice designs, and non-random designs.

Updating the prior process

According to (2.1) the distribution of the outputs is multivariate normal as follows:

$$\mathbf{y}|\beta, \sigma^2, \delta \sim N_n\left(H\beta, \sigma^2 A\right), \tag{2.7}$$

where

$$H = [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]^T, \tag{2.8}$$

and A is the matrix with elements

$$A_{i,j} = C_\delta(\mathbf{x}_i, \mathbf{x}_j). \tag{2.9}$$

Using standard techniques for conditioning in multivariate normal distributions, it can be shown that

$$\eta(\cdot)|\beta,\sigma^2,\delta,\mathbf{y} \sim GP\left(m_0^*(\cdot), V_0^*(\cdot,\cdot)\right), \tag{2.10}$$

where

$$\begin{aligned} m_0^*(x) &= h(x)^T\beta + t_\delta(x)^T A^{-1}(\mathbf{y} - H\beta), \\ V_0^*(x,x') &= \sigma^2\left[C_\delta(x,x') - t_\delta(x)^T A^{-1} t_\delta(x')\right], \end{aligned}$$

where $t_\delta(x) = (C_\delta(x,\mathbf{x}_1),\ldots,C_\delta(x,\mathbf{x}_n))^T$.

Using a weak prior for (β,σ^2), $p(\beta,\sigma^2) \propto \sigma^{-2}$, combining with (2.7) using Bayes Theorem, the posterior for (β,σ^2) is a Normal Inverse Gamma distribution, characterised by

$$\beta|\mathbf{y},\sigma^2,\delta \sim N\left(\widehat{\beta}, \sigma^2\left(H^T A^{-1} H\right)^{-1}\right), \tag{2.11}$$

where $\widehat{\beta} = \left(H^T A^{-1} H\right)^{-1} H^T A^{-1}\mathbf{y}$, and

$$\sigma^2|\mathbf{y},\delta \sim \text{InvGam}\left(\frac{n-q}{2}, \frac{(n-q-2)\widehat{\sigma}^2}{2}\right), \tag{2.12}$$

where $\widehat{\sigma}^2 = \dfrac{\mathbf{y}^T\left(A^{-1} - A^{-1}H\left(H^T A^{-1} H\right)^{-1} H^T A^{-1}\right)\mathbf{y}}{n-q-2}$.

Integrating β out from the product of (2.10) and (2.11), it can be shown that

$$\eta(\cdot)|\mathbf{y},\sigma^2,\delta \sim GP\left(m_1(\cdot), V_1^*(\cdot,\cdot)\right), \tag{2.13}$$

where

$$m_1(x) = h(x)^T\widehat{\beta} + t_\delta(x)^T A^{-1}(\mathbf{y} - H\widehat{\beta}), \tag{2.14}$$

$$\begin{aligned} V_1^*(x,x') &= \sigma^2\left[C_\delta(x,x') - t_\delta(x)^T A^{-1} t_\delta(x')\right. \\ &+ \left.\left(h(x) - t_\delta(x)^T A^{-1} H\right)\left(H^T A^{-1} H\right)^{-1}\left(h(x') - t_\delta(x')^T A^{-1} H\right)^T\right]. \end{aligned} \tag{2.15}$$

Student-t process emulator

The Student-t process emulator is obtained by integrating σ^2 out from the product of (2.12) and (2.13), and is given by

$$\eta(\cdot)|\mathbf{y}, \delta \sim \text{Student-t Process}\left(n-q, m_1(\cdot), V_1(\cdot,\cdot)\right), \tag{2.16}$$

where

$$\begin{aligned} m_1(x) &= h(x)^T\widehat{\beta} + t_\delta(x)^T A^{-1}(\mathbf{y} - H\widehat{\beta}), && (2.17)\\ V_1(x,x') &= \widehat{\sigma}^2\left[C_\delta(x,x') - t_\delta(x)^T A^{-1} t_\delta(x')\right. \\ &+ \left.\left(h(x) - t_\delta(x)^T A^{-1}H\right)\left(H^T A^{-1}H\right)^{-1}\left(h(x') - t_\delta(x')^T A^{-1}H\right)^T\right]. && (2.18)\end{aligned}$$

where $\widehat{\beta} = \left(H^T A^{-1}H\right)^{-1} H^T A^{-1}\mathbf{y}$ and $\widehat{\sigma}^2 = \dfrac{\mathbf{y}^T\left(A^{-1} - A^{-1}H\left(H^T A^{-1}H\right)^{-1}H^T A^{-1}\right)\mathbf{y}}{n-q-2}$.

Analogously to a Gaussian process, if $\eta(\cdot)$ is represented by a Student-t process then for every $m = 1, 2, \ldots$ the joint distribution of $\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_m)$ is multivariate Student-t for all $\mathbf{x}_i \in \mathcal{X}$, $i = 1, \ldots, m$.

Inference for the correlation lengths

The correlation parameter vector δ is unknown. A prior for the correlation parameters δ should be specified. if $p(\delta)$ is the prior density function for δ, the posterior density for δ is obtained via

$$\begin{aligned} p(\delta|\mathbf{y}) &\propto p(\delta)\int\int p(y|\beta,\sigma^2,\delta)p(\beta,\sigma^2)d\beta d\sigma^2 \\ &\propto p(\delta)|A|^{-\frac{1}{2}}|H^T A^{-1}H|^{-\frac{1}{2}}(\hat{\sigma}^2)^{-\frac{n-q}{2}}, && (2.19)\end{aligned}$$

where A and $\hat{\sigma}^2$ are functions of δ. Paulo (2005) presents reference priors for the correlation lengths. Andrianakis (2009) discusses the use of different priors for the correlation lengths.

A fully Bayesian analysis would now integrate out the correlation length vector δ from the product of the densities in (2.16) and (2.19). The posterior distribution of δ is intractable, but approximate Bayesian methods can be applied, such as a Laplace approximation (Lindley 1980). Markov Chain Monte Carlo methods can be used for a fully Bayesian analysis as in Bayarri et al. (2007), but these methods require highly intensive computations. Alternatively, δ can be estimated from the posterior distribution $p(\delta|\mathbf{y})$, or from the likelihood $p(\mathbf{y}|\delta)$, and then the analysis proceeds as if these estimates were the true values. This approach is called **the plug-in approach**. Estimates for δ are most often chosen using an optimization method to find the posterior mode (Sacks et al. 1989b; Currin et al. 1991).

2.3 Illustrative examples

In this section we illustrate how to emulate a simulator using some toy examples. We begin with models having only one and two inputs.

2.3.1 One-dimensional toy example

Example 2.1 (1D toy example) *Let the simulator be the function*

$$\eta(x) \quad = \quad 5 + x + \cos x.$$

This toy example was used in Oakley (1999). Figure 2.2 presents the function we use as simulator with some training data, which we use to build the emulator.

We assume that the simulator is represented as a Gaussian process as in (2.1). The prior distribution for $(\beta, \sigma^2, \delta)$ is $p(\beta, \sigma^2, \delta) \propto \sigma^{-2} p(\delta)$, where δ follows a log Normal distribution with parameters $\mu = 0$ and $\sigma^2 = 100$. The reason for choosing this log normal prior is that it is a proper probability distribution but essentially flat over the relevant range. Therefore, conditional on the training data shown in Figure 2.2 and an estimated correlation length, our

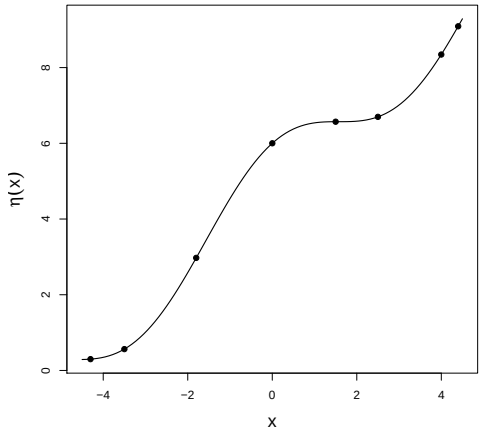
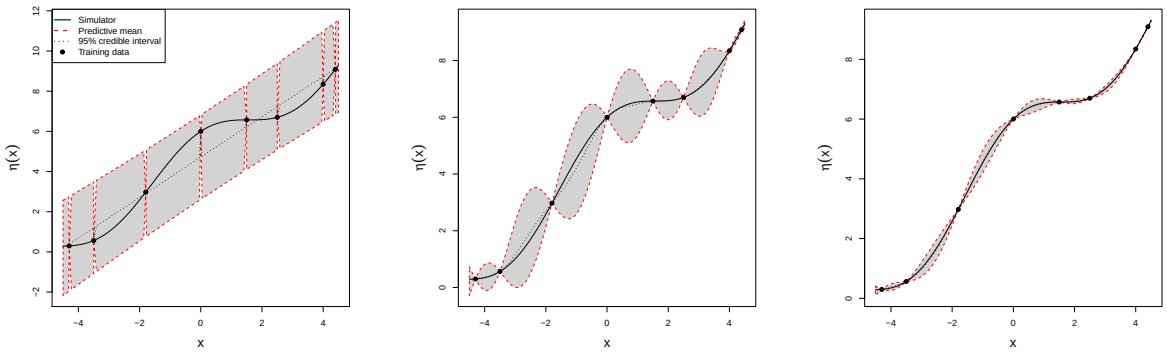


Figure 2.2: Simulator from example 2.1 with some training data given by •.

beliefs about the simulator can be described as a Student-t process emulator (2.16). Figure 2.3 presents some Student-t process emulators conditioned on the training data and different values for the correlation length, δ. Notice that, for all cases, the simulator is contained in the 95% credible intervals.

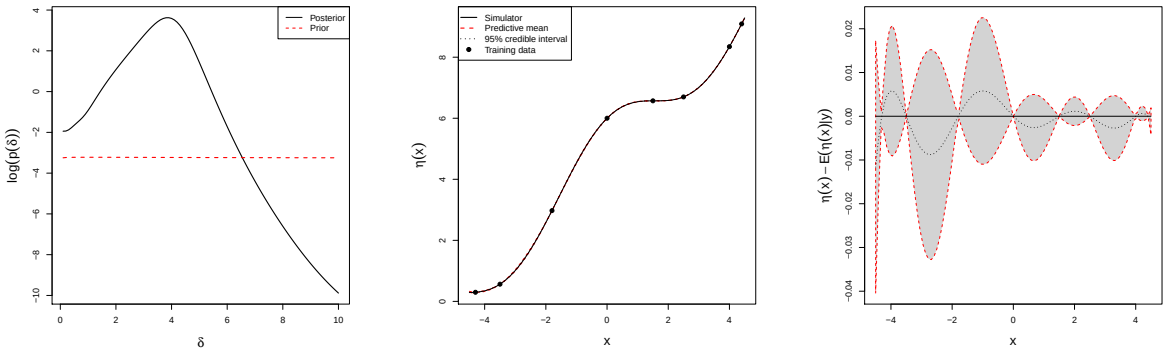
Figure 2.3 (a) presents the independent case, i.e. when $\delta = 0$. Notice that the emulator mean at untried inputs is represented by an adjusted regression line for the training data, and the widths of the 95% credible intervals for outputs at most inputs are exactly the same. This emulator is underconfident. Figure 2.3 (b) presents the case where $\delta = 1$. We assume that there is some correlation between outputs for inputs close to each other. We observe that predictions at inputs near training data have small ranges for the 95% credible intervals. The size of the 95% credible interval increases as distances between the input we want to predict at and the nearest training data increase. We are now more confident about the simulator behaviour in comparison with the independent case. Figure 2.3 (c) presents the case where $\delta = 2$. Now the correlation between outputs for inputs close to each other is significantly higher. Therefore, the size of the intervals gets smaller. We are more confident now than in the previous cases. However we need to justify why we use a correlation length with that size. We could keep increasing the correlation lengths, but we would eventually be overconfident about the simulator behaviour and make some wrong predictions.



(a) $\delta = 0$ (b) $\delta = 1$ (c) $\delta = 2$

Figure 2.3: Student-t process emulators conditioned on the training data and different values for the correlation lengths.

We estimate the correlation length δ from its posterior distribution, equation (2.19), using a log Normal prior with parameters (0,100). The posterior mode is given by $\hat{\delta} = 3.857$. Figure 2.4 (a) presents the prior and the posterior distribution for δ. We notice that the prior density is flat, as our prior beliefs about δ are vague. The Student-t emulator using $\hat{\delta}$ as the true correlation length is presented in Figure 2.4 (b). We notice that the emulator predictions are accurate with narrow 95% credible intervals. In Figure 2.4 (c), we plot the difference between the simulator and the predictive mean. Now, we can see clearly the 95% credible intervals. Even though the emulator predictions seems perfect in Figure 2.4 (b), when we look at the difference between the simulator and the predictive mean the emulator seems to be underconfident, though perhaps trivially so.



(a) (b) (c)

Figure 2.4: Example 2.1: (a) Prior and posterior distribution for the correlation length; (b) Student-t process emulator conditioned on the training and posterior mode for δ; (c) Difference between the simulator and the emulator mean with 95% credible intervals.

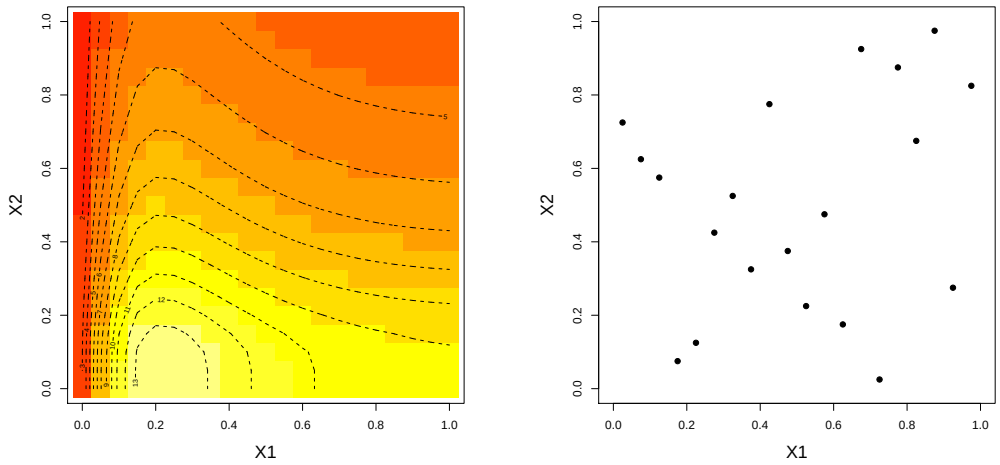
2.3.2 Two-dimensional toy example

Example 2.2 *We suppose that the simulator encodes the following mathematical function*

$$\eta(x_1, x_2) = \left(1 - e^{-\frac{1}{2x_2}}\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right), \tag{2.20}$$

where $(x_1, x_2) \in (0,1)^2$. *This example is a toy example used in the GEM-SA software*[1].

Figure 2.5 (a) presents the simulator (2.20) over the input space. The training data are composed of 20 points selected by a Latin hypercube sampling, Figure 2.5 (b). Using the training data, we estimate the correlation length parameters by maximizing the function (2.19). The estimates are $(\hat{\delta}_1, \hat{\delta}_2) = (0.2421, 0.4240)$, indicating that the simulator is more smooth with respect to the second input than the first.



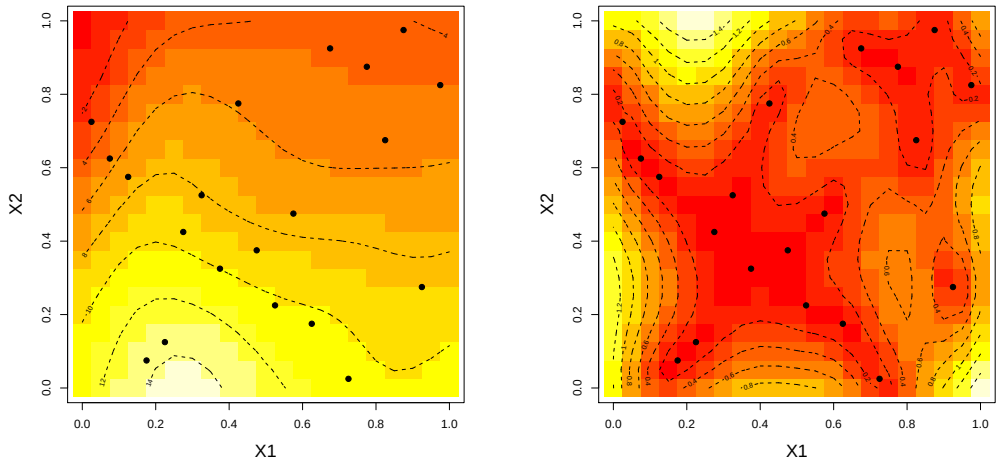
(a) (b)

Figure 2.5: Example 2.2: (a) Simulator applied on the input space; (b) The training data.

Using the training data and the estimated correlation lengths, we build the Student-t process (2.16) and we predict the simulator in a grid of 20×20 points over the input space. Figure 2.6 (a) presents the emulator posterior mean and (b) the posterior standard deviations applied over the grid of points. Visually, the emulator mean using 20 training data points is

[1]The GEM-SA software is available at `http://ctcd.group.shef.ac.uk/gem.html`.

able to capture the main features of the true simulator. Note that the colourmaps used to represent the output axes in Figures 2.5 (a) and 2.6 (a) are on the same scale. The standard deviations, as expected, are small close to the training data, and are larger far from the training data, where there is little information.



(a) (b)

Figure 2.6: Student-t process emulator using the training data: (a) emulator mean; (b) emulator standard deviation.

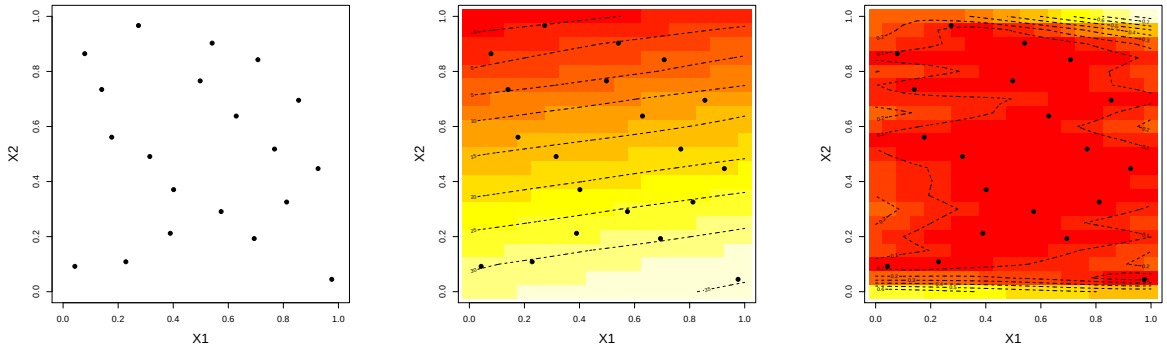
2.3.3 *Surfebm* model

The *Surfebm* model is an energy balance model of the Earth's climate used in Andrianakis (2009). The state variables are upper ocean temperatures averaged around the globe at constant latitudes. This simplified version of the model has two inputs: the solar constant and the albedo, and one output: the mean surface temperature. We normalised the inputs to the $(0, 1)^2$ space. Using a Latin hypercube design, 20 training inputs were generated and are presented in Figure 2.7 (a). Having obtained our design points, we then run the *surfebm* model at these points and get the mean surface temperature, which we denote by $\mathbf{y} = (\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_{20}))^T$.

We assume that the uncertainty on the surfebm model is well represented by a Gaussian process using a linear mean function with $h(x) = (1, x)^T$. For the covariance function we

choose $\sigma^2 C_\delta(\cdot,\cdot)$, where the correlation function $C_\delta(\cdot,\cdot)$ is the squared correlation function. Given the training data, we estimate the correlation lengths, δ, from (2.19) assuming a flat log Normal prior with parameters (0,100) for each correlation length. The posterior mode is given by $(\hat{\delta}_1, \hat{\delta}_2) = (2.2546222, 0.1443991)$, indicating that the simulator is more smooth with respect to the first input than the second.

Conditioned on the training data and on the estimated correlation lengths, we fit a Student-t process emulator (2.16) in a grid of points in the input space. Figure 2.7 (b) presents the predictive mean of the Student-t emulator. Predictions for the mean surface temperature are high values when we observe large values for the solar constant input, X_1, and small values for the albedo input, X_2. The mean surface temperature is predicted to be small when the albedo input is large and the solar constant input is small. A linear relationship between the mean surface temperature and the two inputs can be seen. Figure 2.7 (c) presents, as a measure of uncertainty, the predictive standard deviations. The variability of the predictions is small next to the training data, and the standard deviation increases in regions where there are few or no data.



(a) (b) (c)

Figure 2.7: Student-t process emulator for the surfemb model using the training data (a); Predictive mean (b); Predictive standard deviation (c).

2.3.4 Multiple output emulators

Many simulators have several outputs. In this thesis, we focus on single output simulators, but some authors consider multiple output simulators. The simplest way to emulate a multi-output simulator is to use an independent Gaussian process emulator for each output. The problem with this approach is the assumption that the outputs are all independent. Conti and O'Hagan (2007) model the multi-output simulator using a Gaussian process, taking into account the correlation between the outputs where they use a separable covariance function for the outputs. Rougier (2008) presents an alternative multivariate emulator which has not only a separable covariance function but also a separable mean function. Rougier's approach is called the outer-product emulator.

A simulator may represent a real-world process that evolves over time (or sometimes in space, or both time and space). Such simulators are multi-output with a particular structure for the outputs which are time-series (or maps, or maps varying with time). Conti et al. (2009) and Liu and West (2009) present different approaches for emulation when the output is a time series.

2.4 Analysis using emulators

The simplest analysis using an emulator is to predict the output at untried inputs. This is important when the simulator is expensive to run. Note that the emulator provides a probability distribution for that output, and not only a single value for the output prediction. Welch et al. (1992) present an application of Gaussian processes to screening (input selection) and prediction in complex simulators.

2.4.1 Uncertainty analysis

A common scenario is that one or more inputs are uncertain. Therefore, we consider the input to be a random variable $\mathbf{X}$. Consequently the output $\mathbf{Y} = \eta(\mathbf{X})$ is a random variable. We are interested in the induced distribution of $\mathbf{Y}$, known as the uncertainty distribution. If the simulator can be run at a large number of input configurations, then a simple Monte Carlo method can be used to obtain a random sample of the outputs. However even for relatively cheap simulators, those that take a few seconds to run, several thousands of simulator runs can be very expensive. Using emulators, the computational cost can be reduced significantly, where just a few hundreds of runs may be required for analyses that provide similar results. O'Hagan (2006) provides a comparison between Monte Carlo uncertainty analysis using the simulator directly and an emulator.

Uncertainty analysis for simulators using emulators was presented by Haylock and O'Hagan (1994), where they derived the posterior moments of mean and variance of the output distribution. Oakley and O'Hagan (2002) extended Haylock and O'Hagan's results deriving the posterior moments of the distribution function of the output, and made inference about the density function of the output.

2.4.2 Sensitivity analysis

Sensitivity analysis is concerned with understanding how changes in the inputs $\mathbf{x}$ affect the output y. Saltelli, Chan, and Scott (2000) discuss some different measures to quantify input sensitivity using the simulator, for example sensitivity analyses based on variance decomposition and regression modelling. Sensitivity analysis methods requiring a large number of simulator runs can be impractical for expensive simulators.

When there is uncertainty in the inputs, the input is treated as a random variable $\mathbf{X}$. Consequently the output $\mathbf{Y} = \eta(\mathbf{X})$ is also a random variable. The approach of sensitivity analysis that takes into account the uncertainty in the inputs is known as probabilistic sensitivity analysis. Learning about the distribution of the output induced by the inputs $\mathbf{X}$

is uncertainty analysis. Sensitivity analysis goes beyond uncertainty analysis, exploring how individual inputs affect the distribution of the output. Oakley and O'Hagan (2004) presented a probabilistic sensitivity analysis using the emulator where they provided Bayesian inference about sensitivity measures based on variance and regression fitting.

Kennedy, Anderson, Conti, and O'Hagan (2006) present a number of recent applications in which an emulator of a computer code is created using a Gaussian process model. They presented three case studies from the Centre for Terrestrial Carbon Dynamics (CTCD) where sensitivity analysis and uncertainty analysis are illustrated.

2.4.3 Calibration

In computer experiments, it is assumed that there is an unknown true input-output function that describes a particular real process, $\xi(\cdot)$, and the simulator is a simplified representation of this real-world function. When 'parameters' of the true real-world function are unknown, suitable physical observations may be used to learn these unknown parameters, which are specified as simulator inputs. This process is called calibration.

In calibration, as well as uncertainty in the simulator input, the simulator may be a biased version of reality, and the physical observations may include noise. These three major sources of uncertainty are considered by Kennedy and O'Hagan (2001). They presented a full Bayesian calibration analysis. Following calibration the modellers may want to predict the real process, combining the calibrated simulator with the physical observations. This is possible using the Kennedy and O'Hagan model, where the difference between the computer model and the real process is called the discrepancy function. We come back to this problem in Chapter 7. It is worth mentioning that when the simulator is expensive to run, an emulator can be used in calibration.

Bayarri et al. (2005) presented an application of the Kennedy and O'Hagan calibration model to study the effect of a collision of a vehicle. As an alternative to full Bayesian calibration, Goldstein and Rougier (2006) presented a Bayes linear approach.

2.5 Conclusion

In this chapter we have presented the emulator as a statistical representation of our knowledge about the outputs of a simulator. We have shown how to use Gaussian processes to represent our beliefs about the simulator outputs. We have shown how to update the prior process using a set of simulator runs. We illustrate the inference for emulator for some toy examples.

We have reviewed some analyses using emulators as surrogates of simulators, such as uncertainty and sensitivity analyses. The use of an emulator in these analyses is important when the simulator is expensive to run. However, before using the emulator it is necessary to check whether the emulator correctly represents our uncertainty about the simulator outputs. This process is called validation. Similarly, in computer experiments when a simulator is used to represent a real system, the simulator has to be validated. A literature review on validation for simulators is presented in Chapter 3. Diagnostics for validating Gaussian process emulators are presented in Chapter 4.

Chapter 3

Validation of models: a literature review

3.1 Introduction

In this chapter we provide a literature review on validation of models. We first consider methods for validation for deterministic models, i.e. simulators, and consider their suitability for validating emulators. A simulator, which we represent by a function $\eta(\cdot)$, is a mathematical approximation of a real system. We assume that there is an unknown true input-output function, $\xi(\cdot)$, that represents the real system, so that $\eta(\cdot)$ is an approximation of $\xi(\cdot)$. Before using a simulator to investigate a real system, it should be validated. The validation process consists of providing evidence in favour or against the simulator as a surrogate of the real system.

When a real system $\xi(\cdot)$ or a simulator $\eta(\cdot)$ is represented by a probabilistic model, the predictions are given by probability distributions. Comparisons between predictions and 'real' observations (physical observations or simulator runs) should also take into account the uncertainty associated with the predictions. In Section 3.3, we discuss some ideas for validating probabilistic predictive models.

3.2 Verification and validation of simulators

In validating a simulator there are two steps. The first step refers to the assessment of the accuracy of the numerical solutions with respect to the theoretical solutions of the complex mathematical model. The second step refers to checking if the simulator is able to represent the intended real system; this process generally involves comparison of simulator outputs to physical observations of the real system. Fishman and Kiviat (1968) have called these two steps verification and validation (V&V) respectively.

In computer science and engineering, there are guidelines for verification and validation of simulators. Published examples of guidelines include the Institute of Electrical and Electronics Engineers (IEEE 1991, 1998), the US Department of Defence (USDoD 1996), the American Institute of Aeronautics and Astronautics (AIAA 1998), and the U.S. Food and Drug Administration (FDA 2002).

Some formal definitions are given in Trucano et al. (2006).

Verification is the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solutions to the model.

Validation is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

The literature on V&V for simulators is quite extensive and represents different perspectives and approaches, such as philosophical theories about validation, statistical techniques and software practices. Some reviews of the literature about V&V are given in Balci and Sargent (1984), Kleijnen (1995b), Roache (1998) and Oberkampf and Trucano (2000).

The statistical validation techniques depend on the availability of the physical observations. Kleijnen (1999) described three levels of data availability: (i) when real data is scarce; (ii) when only the outputs from the real system are available; (iii) when both outputs and inputs from the real system are available.

3.2.1 Validating with scarce real data

In some applications, physical data may be either scarce or completely missing, for example, when modelling hypothetical nuclear accidents. In studies where there are no real data, then strong validation claims are impossible. However, analysts can perform simulation studies to find out whether the simulator contradicts qualitative expert knowledge. The process of asking experts whether the simulation outputs are reasonable is called Face Validity (Sargent 1979). Another validation technique is sensitivity analysis (Welch et al. 1992; Kleijnen 1995b). This is performed to show whether the effect of changes to inputs agrees with an expert's prior qualitative knowledge.

3.2.2 Validating using outputs of the real system only

In this situation the inputs of the real system cannot be measured, and only the outputs are available. Therefore, we have a set of runs of the simulator $(y_1 = \eta(\mathbf{x}_1), \ldots, y_n = \eta(\mathbf{x}_n))$ and a set of observations of the real process $(z_1, \ldots, z_m)$ at unknown input values. For example, Kleijnen (1995a) presents a case study involving the search for mines by means of sonar, where a simulator is validated whose output is the probability that in a certain position there is a mine. In this case study, the US Navy had one team that deposited some mines on the sea bottom. However, it was impossible to measure environmental variables such as temperature and salinity of the sea water, which were simulator inputs. To validate the simulator, in the first stage a sensitivity analysis was performed where the results were compared with expert intuition. In a second stage there is a comparison of binary (success/failure) outcome of n runs of the simulator and m field trials.

The two-stage validation can be applied in situations where only the outputs of the real system are available. The first stage is a face validation, where sensitivity analysis is performed in order to show whether the effect of changes on inputs agree with expert prior qualitative knowledge. In the second stage the empirical distribution of the simulator output obtained using the outputs $(y_1, y_2, \ldots, y_n)$ is compared with the empirical distribution

of the real output obtained using the observations $(z_1, z_2, \ldots, z_m)$, where these distributions would be the same for an ideal simulator assuming that both the simulator outputs and the real observations represent the same input space. The most popular statistical techniques for comparing two distributions are the χ^2 test and the Komolgorov-Smirnov test.

3.2.3 Validating using inputs and outputs of the real system

We have seen that it is possible to compare expert knowledge with the result of sensitivity analysis. But in the situation that both output and inputs of the real process are available, a more thorough validation analysis can be performed. Let z_i be a measure of the real system at location $\mathbf{x}_i$, for $i = \{1, 2, \ldots, n\}$. For the real inputs, we also obtain the respective simulator runs $(y_1 = \eta(\mathbf{x}_1), \ldots, y_n = \eta(\mathbf{x}_n))$. Kleijnen et al. (1998) call this process "trace-driven simulation".

The simplest comparison method is a graphical comparison between simulator outputs and real observations. Based on expert knowledge and with a degree of confidence given by the expert, the simulator can be judged as a good or bad approximation to the real system. This method is very subjective, but it is often used in practice. Kozempel et al. (1995) proposed to fit a regression line and test whether the fitted line has unit slope and intercept zero.

Quantifying the validity of a simulator can also be seen as a hypothesis testing problem when we calculate the difference between simulator outputs and physical observations and test a null hypothesis that the difference has zero mean. Hills and Trucano (1999, 2001) propose the statistic

$$X^2 = (\mathbf{z} - \mathbf{y})^T (\Sigma_d)^{-1} (\mathbf{z} - \mathbf{y}),$$

where $\mathbf{z} = (z_1, \ldots, z_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$, and Σ_d is the estimated covariance matrix for the difference between the simulator outputs and the experimental observations. Under the assumption that errors between the simulator outputs and that of physical observations are independent normal random variables, the statistic X^2 has a chi-square distribution with n

degrees of freedom. Hills and Trucano also mentioned that non-parametric methods can be used to compare the simulation and the real outputs, such as the Wilcoxon test for comparing the mean of the simulator output with the real observations, and the Kolmogorov-Smirnov test to establish whether the distribution of difference between simulator output and real observations is normally distributed.

Oberkampf and Barone (2006) assumed that the physical observations follow a normal distribution with mean μ, and proposed a validation metric based on confidence intervals for the true error, $D = \bar{\mathbf{y}} - \mu$, where $\bar{\mathbf{y}}$ is mean of the simulator outputs, and μ is the true mean of the real process. A confidence interval for D is given by

$$\left(\tilde{D} - t_{\frac{\alpha}{2},m-1}\frac{s_d}{\sqrt{m}}; \tilde{D} + t_{\frac{\alpha}{2},m-1}\frac{s_z}{\sqrt{m}}\right), \tag{3.1}$$

where $\tilde{D} = \bar{\mathbf{y}} - \bar{z}$, $\bar{z}$ is the sample mean of m observed values of the real system, s_d is the estimated standard deviation of $d_i = (y_i - z_i)$ for $i = 1, \ldots, n$, and $t_{\frac{\alpha}{2},v}$ is the $1 - \frac{\alpha}{2}$ quantile of Student-t distribution for v degrees of freedom.

Oberkampf and Barone (2006) also extended the metric (3.1) where r experimental replications at each location $\mathbf{x}$ are available $(z_1(\mathbf{x}), \ldots, z_r(\mathbf{x}))$. A confidence interval for the true error for a specific input value $\mathbf{x}$ is given by

$$\left(\tilde{D}(\mathbf{x}) - t_{\frac{\alpha}{2},r-1}\frac{s_d(\mathbf{x})}{\sqrt{r}}; \tilde{D}(\mathbf{x}) + t_{\frac{\alpha}{2},r-1}\frac{s_d(\mathbf{x})}{\sqrt{r}}\right), \tag{3.2}$$

where $\tilde{D}(\mathbf{x}) = \eta(\mathbf{x}) - \bar{z}(\mathbf{x})$, $\bar{z}(\mathbf{x})$ is the sample mean of observed values of the real system at input $\mathbf{x}$ based on r replications of the experiment, and $s_d(\mathbf{x})$ is the estimated standard deviation of $(\eta(\mathbf{x}) - z_k(\mathbf{x}))$ for $k = 1, \ldots, r$.

The authors also presented some global metrics in order to provide a more compact statement of a validation metric result. The *average relative error metric* and the *average relative confidence indicator* are defined by

$$\left|\frac{\tilde{D}}{\bar{z}}\right|_{avg} = \int_{x\in\chi}\left|\frac{\eta(x) - \bar{z}(x)}{\bar{z}(x)}\right| dx, \quad \text{and} \quad \left|\frac{CI}{\bar{z}}\right|_{avg} = \frac{t_{\frac{\alpha}{2},r-1}}{\sqrt{r}}\int_{x\in\chi}\left|\frac{s_d(x)}{\bar{z}(x)}\right| dx$$

where CI is half-width of the confidence interval (3.2).

Other proposed metrics are the *maximum relative error metric* and the *maximum relative confidence indicator*. These metrics allow identifying some particular point over the range of the data that should be noted. They are respectively defined by

$$\left|\frac{\tilde{D}}{\bar{z}}\right|_{max} = \max_{x\in\chi}\left|\frac{\eta(x)-\bar{z}(x)}{\bar{z}(x)}\right|, \quad \text{and} \quad \left|\frac{CI}{\bar{z}}\right|_{max} = \frac{t_{\frac{\alpha}{2},r-1}}{\sqrt{r}}\max_{x\in\chi}\left|\frac{s_d(x)}{\bar{z}(x)}\right|.$$

3.2.4 Validating the simulator using the calibration model

A common assumption is that physical data are observed with noise (Santner et al. 2003). We observe

$$z_i = \xi(\mathbf{x}_i) + \epsilon_i, \tag{3.3}$$

where ϵ_i is the observation error for the ith observation.

Another assumption is that the simulator is a biased version of reality. This assumption is reasonable since a simulator is in fact an approximation of a real system.

$$\xi(\cdot) = \eta(\cdot) + d(\cdot), \tag{3.4}$$

where $d(\cdot)$ is an unknown discrepancy function, also known as the bias function.

Combining (3.3) and (3.4), Kennedy and O'Hagan (2001) present the calibration model for linking observations to simulator outputs:

$$z_i = \xi(\mathbf{x}_i) + \epsilon_i = \rho\eta(\mathbf{x}_i, \theta) + d(\mathbf{x}_i) + \epsilon_i, \tag{3.5}$$

where ϵ_i is the observation error for the ith observation, ρ is an unknown regression coefficient, θ is a calibration parameter for the simulator, and $d(\cdot)$ is a discrepancy function. The discrepancy function, the simulator and the observational errors are all independent from each other. Note that some authors only consider $\rho = 1$.

Bayarri et al. (2002, 2005) proposed a Bayesian validation framework with two main steps. The first step is a prediction step based upon the statistical methodology proposed by Kennedy and O'Hagan (2001), where a Bayesian calibration model is built and used for making predictions. The second step is a validation step based on Bayesian hypothesis testing methodology. It is tested whether the simulator can represent the real process. Let M_0 denote the model without the discrepancy function and M_1 the model including the discrepancy function, i.e

$$M_0 : z_i = \eta(x_i, \theta) + \epsilon_i,$$
$$M_1 : z_i = \eta(x_i, \theta) + d(x_i) + \epsilon_i.$$

Given a prior probability for each model, π_0 and π_1 respectively, Bayes theorem gives that the posterior probability of M_0 is given by

$$p(M_0|\mathbf{y}, \mathbf{z}) = \frac{\pi_0 l_0(\mathbf{y}, \mathbf{z})}{\pi_0 l_0(\mathbf{y}, \mathbf{z}) + \pi_1 l_1(\mathbf{y}, \mathbf{z})} \tag{3.6}$$

where $l_i(\mathbf{y}, \mathbf{z})$ is the marginal likelihood for the model M_i, $i = 0, 1$.

Rebba, Mahadevan, and Huang (2006) presented a validation metric based on a Bayes Factor where they compare the simulator output with experimental data when both are stochastic. Rebba et al. also presented a model error estimation methodology and sensitivity analysis of the validation metric.

Wang et al. (2009) criticised the validation metric proposed by Oberkampf and Barone (2006), (3.1) and (3.2), and based on Kennedy and O'Hagan (2001) Wang et al. developed a Bayesian procedure for simulator validation. They proposed a validation metric based on prediction intervals. Let $l_d(x)$ and $u_d(x)$ be the lower and upper bounds of the $100(1-\alpha)\%$ prediction interval for $d(\cdot)$, where $d(\cdot)$ is the discrepancy function defined in (3.5). Define

$$\Delta_{min}(x) = \begin{cases} 0, & \text{if } \eta(x) \in (l_d(x), u_d(x)), \\ \min\left\{|\eta(x) - l_d(x)|, |\eta(x) - u_d(x)|\right\}, & \text{otherwise,} \end{cases}$$

and

$$\Delta_{max}(x) = \max\left\{|\eta(x) - l_d(x)|, |\eta(x) - u_d(x)|\right\}.$$

Let Δ_0 be the maximum allowable deviation between the simulator output and the real experiment that the user will tolerate. Then the validation rules are as follows: reject the simulator at x if $\Delta_{min}(x) > \Delta_0$, accept the simulator at x if $\Delta_{max}(x) < \Delta_0$, and there is no conclusion and more physical observations are needed if $\Delta_{min}(x) \leq \Delta_0 \leq \Delta_{max}(x)$.

3.3 Validating emulators

We now consider the problem of validating an emulator rather than validate the simulator itself. The emulator is a probabilistic predictive model for the simulator, and validating an emulator should provide information in order to check whether the emulator represents well our uncertainty about the simulator $\eta(\cdot)$.

The validation definition presented in Section 3.2 needs to be adapted as follows.

Validation is the process of determining that a probabilistic model represents our uncertainty about the true model (a simulator or a real system) appropriately.

An example of an invalid model is a model in which the uncertainty on the predictions is too large. Such model is called underconfident model. Analogously, if the uncertainty on the predictions is too small, then we have an invalid model which is overconfident.

In elicitation, it is possible to determine the appropriateness of a set of probability judgements, where probability judgements for a series of events are compared with the observed relative frequency of these same events (O'Hagan et al. 2006). Here, we want to check if the probability judgements for an unknown function appropriately represent uncertainty about the function which can be either a simulator $\eta(\cdot)$ or the real system $\xi(\cdot)$. The problem with using elicitation-based methods for evaluating probability judgements about an unknown function is that even if we observe the function at many points, we only have a single 'sample' of the random process. Instead, we must check the assumptions made to build the probabilistic model.

3.3.1 Existing methods for validating emulators

The use of emulators has become very popular. However, there is little literature on validating such models. Sacks et al. (1989b) use quantile-quantile plots of the standardised residuals to check if their emulator represents well a simulator. Rougier et al. (2007) presented a cross-validation diagnostic called the "leave-one-out" diagnostic, where they left one element out of the training data and predicted it, repeating this procedure for all elements and plotting credibility intervals for each element. They also presented a diagnostic where they leave out more than one element.

Validating an emulator can be compared with validating linear models with dependent errors. However, the Gaussian process emulator is modelling a deterministic function, and hence the 'predictions' for observed values used to build the model are perfect as the Gaussian process interpolates the training data exactly, with no uncertainty. Residuals can only be obtained using cross validation methods, or, more appropriately, a new dataset. Therefore, the diagnostics used in linear models need to be adapted to be applied in the computer model framework.

In the context of linear models, Haslett and Hayes (1998) presented a general theory for residuals, where they define the marginal and the conditional residuals. The marginal residuals are the errors between the observed values and the fitted values, whereas the conditional residuals are the errors between the predictive values for observed values not used to build the model. Fraccaro et al. (2000) presented graphical diagnostics for marginal and conditional residuals in a time series regression context. The authors decompose the estimated variance matrix of the residuals using a Cholesky decomposition, and rotate them in order to have uncorrelated residuals with unit variance. Houseman et al. (2004) used the Cholesky decomposition of the inverted covariance matrix to rotate the residuals of linear mixed models and time series. Houseman at al. provided a quantile-quantile plot of these uncorrelated errors providing asymptotic properties of the empirical cumulative distribution and pointwise standard errors.

The Gaussian process emulators that we consider in this thesis model deterministic func-

tions, so that there are no marginal errors as defined by Haslett and Hayes (1998). Specifically, emulator predictions for outputs used to build the emulator will exactly equal the respective outputs, with zero variances. We can only obtain prediction errors for simulator runs not used to build the emulator, the conditional errors in Haslett and Hayes (1998). Another important feature that should be considered in the diagnostics for validating emulators is the error correlation due to a spatial correlation structure. Therefore our uncertainty about a simulator output at inputs near a training data point is smaller than our uncertainty about a simulator output at inputs far from any training data point.

Graphical diagnostics presented by Houseman et al. (2004), who use a quantile-quantile plot for the uncorrelated conditional errors, and by Fraccaro et al. (2000) who plot uncorrelated conditional errors against the data order, will be used in this work. However, the Cholesky decomposition depends on the data order. In Fraccaro et al. (2000) the data are indexed by time, but in computer modelling the data order is typically arbitrary. Therefore, it is necessary to present some decomposition methods that are invariant to the data order. In this work, we propose to use the eigen and the pivoted Cholesky decompositions to build the uncorrelated errors, as will be discussed in Section 4.3.3.

3.3.2 Validating calibration models

In Kennedy and O'Hagan (2001), the simulator is approximated by a Gaussian process emulator where the authors propose statistical methodology to make predictions of the real process. Kennedy and O'Hagan propose a probabilistic predictive model to represent the real system using the model (3.5). O'Hagan (2006) refers to the predictive model for reality as a **predictor**.

In order to validate a predictor, we need to check whether the predictor represents well our uncertainty about the reality, taking into account our uncertainty for the simulator, the discrepancy function and some unknown parameters. Therefore, we need to check all assumptions made and compare the probabilistic predictions with real observations. Kennedy and O'Hagan (2001) use quantile-quantile plots of the standardised residuals of their cal-

ibration model to check statistical assumptions. Goldstein and Rougier (2006) present a diagnostic based on the deviation between real observations and Bayes linear predictions of the simulator for the same inputs, where they presented a standardised deviation and compared the values with a distribution with zero mean and unit variance. In Chapter 7, we present a set of diagnostics for the full Bayesian approach for the discrepancy function model, which is a particular case of the calibration model.

3.4 Conclusion

The validation and verification methods for simulators are essential for developing simulators. They determine whether a simulator can be used as a surrogate of the real system with accuracy defined by the simulator user. There are several V&V guides and a very extensive literature for validation and verification for simulators. However, there is little attention on validation when the simulator or the real process is represented by a stochastic process.

In this thesis, we focus on validation of probabilistic predictive models as surrogates of simulators and real systems. The V&V methods provide information about the accuracy of the model, and here in this thesis we are interested in describing how the probabilistic predictive model represents our uncertainty about the object that the model was developed to imitate.

The validation process for probabilistic model predictions is based on diagnostics, checking the assumptions made to build the predictive model and comparing the predictions with 'real' observations of the model (simulator runs or physical observations), taking into account the uncertainty provided by the predictive model. In Chapter 4, we present a set of diagnostics for the validation of Gaussian process emulators used as surrogates of simulators. In Chapter 7, we show how to validate the discrepancy function model used to model a real system which is a particular case of the Kennedy and O'Hagan (2001) calibration model.

Chapter 4

Diagnostics for Gaussian process emulators

4.1 Introduction

Emulators have been used as stochastic approximations of expensive simulators in several areas of science, but in order to build emulators some assumptions and approximations are made. Unless the emulator correctly represents the simulator, inferences made using that emulator will be invalid. Hence, emulators need to be subjected to validation testing. There is a large literature on using emulators to represent expensive simulators, but there has been little research into validating emulators before using them. In this chapter, we propose some numerical and graphical diagnostics for Gaussian process emulators that take into account simulator uncertainty.

In Section 4.2, we briefly review the ideas of Gaussian process emulation presented in Chapter 2, and describe some possible problems with using Gaussian process emulators. In Section 4.3, we present some methods that have been proposed for validating computer models, and then we present some numerical and graphical diagnostics for Gaussian process emulators. The proposed diagnostics depend on new runs of the simulator where we compare

the observed outputs with their respective emulator predictions. In Section 4.5, we demonstrate the diagnostic tools in synthetic and real examples. Where the emulator is failing to represent the simulator adequately, the diagnostics give warnings that allow the source of the validation problems to be identified.

4.2 Emulation

An emulator is a stochastic process that represents the simulator, where the simulator is viewed as an unknown mathematical function. We assume that the uncertainty about the simulator output is described as a Gaussian process with a mean function $h(\cdot)^T\beta$, and a homoscedastic covariance function $\sigma^2 C_\delta(\cdot,\cdot)$ with squared exponential correlation function (2.6), in which δ are correlation lengths.

Therefore, conditional on some training data, $\mathbf{y} = (y_1 = \eta(\mathbf{x}_1), \ldots, y_n = \eta(\mathbf{x}_n))^T$ and integrating out (β, σ^2) our beliefs about the simulator output at non observed inputs is described by a Student-t process:

$$\eta(\cdot)|\mathbf{y}, \delta \sim \text{Student Process}\,(n-q, m_1(\cdot), V_1(\cdot,\cdot)) \tag{4.1}$$

where

$$\begin{aligned} m_1(x) &= h(x)^T\widehat{\beta} + t_\delta(x)^T A^{-1}(\mathbf{y} - H\widehat{\beta}), \\ V_1(x,x') &= \widehat{\sigma}^2\left[C_\delta(x,x') - t_\delta(x)^T A^{-1} t_\delta(x')\right. \\ &+ \left.\left(h(x) - t_\delta(x)^T A^{-1}H\right)\left(H^T A^{-1}H\right)^{-1}\left(h(x') - t_\delta(x')^T A^{-1}H\right)^T\right]. \end{aligned}$$

where $\widehat{\beta} = \left(H^T A^{-1}H\right)^{-1} H^T A^{-1}\mathbf{y}$, and $\widehat{\sigma}^2 = \dfrac{\mathbf{y}^T\left(A^{-1} - A^{-1}H\left(H^T A^{-1}H\right)^{-1}H^T A^{-1}\right)\mathbf{y}}{n-q-2}$.

The emulator distribution depends on the correlation lengths. Here we use the plug-in method, where an estimate for the correlation lengths, $\hat{\delta}$, is used as the true value δ ignoring uncertainty. Estimates for the correlation lengths are obtained from their posterior

distribution (2.19).

4.2.1 Possible problems with Gaussian process emulators

Although the Gaussian process is a very flexible class of distributions for representing prior knowledge about the computer model, the Gaussian process emulator (4.1) can give poor predictions of simulator outputs for at least two reasons. First, the assumption of a stationary Gaussian process with particular mean and covariance structures may be inappropriate. Second, even if these assumptions are reasonable there are various parameters to be estimated, and a bad or unfortunate choice of training dataset may suggest inappropriate values for these parameters. In the case of the correlation length parameters, where we condition on fixed estimates rather than integrating over the full posterior distribution, we may also make a poor choice of estimate to condition on.

The uncertainty about the simulator outputs is represented by a Gaussian process, so joint normality of the simulator outputs has to be a reasonable judgement. In particular, the emulator asserts that it is very unlikely for the true simulator output to be more than two or three predictive standard deviations from the predictive mean, and that it is no more likely to be above the predictive mean than below it. In this context transformations may be useful.

In addition to the assumption of normality, specific forms are assumed for the mean and the covariance functions. If the assumed form of the mean in (2.2) is wrong, because inappropriate regressors have been used in $h(\cdot)$, or if the coefficients β have been poorly estimated, then the emulator predictions may be systematically too low or too high in some regions of the input space.

In (2.3), stationarity is assumed for the covariance function, implying that we expect the simulator output to respond with similar degrees of smoothness at all points in the input space. It is assumed that there is a common variance σ^2 and the correlation function depends only on $(\mathbf{x} - \mathbf{x}')$. Hence, either unequal variance or a correlation structure that depends on

spatial position $(\mathbf{x}, \mathbf{x}')$ instead of the difference $(\mathbf{x} - \mathbf{x}')$ only are failures of the stationarity assumption. In practice, simulators may respond much more rapidly to changes in the inputs at some parts of the space than others. In the case of such non-stationarity, credible intervals of emulator predictions can be too wide in regions of low responsiveness or too narrow in regions where the response is more dynamic.

Finally, although the form of the covariance function may be appropriate, we may estimate the correlation lengths (δ) poorly. Poor estimation of the correlation parameters leads to credible intervals that are too wide or too narrow in the neighbourhood of the training data points.

In the next section, we present some diagnostics that can be useful for identifying problems in emulator predictions. These diagnostics are based on statistical comparisons between new (validation) runs of the simulator and their respective predictions.

4.3 Diagnostics for validating Gaussian process emulators

In order to validate an emulator, our diagnostics will be based on comparisons between emulator predictions and simulator runs for a new dataset. Let $\mathbf{X}^{(v)} = (\mathbf{x}_1^{(v)}, \mathbf{x}_2^{(v)}, \ldots, \mathbf{x}_m^{(v)})^T$ denote a non-observed set of inputs, called validation input data. The simulator outputs for the validation input data are given by $\mathbf{y}^{(v)} = \eta(\mathbf{X}^{(v)})$ where $\mathbf{y}^{(v)} = (y_1^{(v)}, \ldots, y_m^{(v)})^T$, and $\eta(\mathbf{X}^{(v)}) = \left(\eta(\mathbf{x}_1^{(v)}), \ldots, \eta(\mathbf{x}_m^{(v)})\right)^T$. The validation input data should be selected to cover the whole of that part of the input space over which we wish to use the emulator. Otherwise, we might validate an emulator which does not represent the simulator for a particular non-observed subset of the input space.

A general diagnostic $D(\cdot)$ is a function of the validation data output, and we propose to compare the observed $D(\mathbf{y}^{(v)})$ with the reference distribution of $D(\eta(\mathbf{X}^{(v)}))$ conditioned on the training data. If $D(\mathbf{y}^{(v)})$ lies in an appropriately chosen region with a small probability,

then this suggests that there is a conflict between the emulator and the simulator. The test region(s) for $D(\cdot)$ should be chosen so that $D(\mathbf{y}^{(v)})$ falling in the region is associated with a particular failure in the construction of the emulator. If across a range of such diagnostics there are no indications of conflict, then we can suppose that the emulator is representing uncertainty about the simulator appropriately.

4.3.1 Individual prediction errors

Individual prediction errors for the validation data are given by the differences between the observed simulator outputs and the predictive mean output at the same inputs, i.e. $(\mathbf{y}_i^{(v)} - E[\eta(\mathbf{x}_i^{(v)})|\mathbf{y}])$, for $i = 1, 2, \cdots, m$.

We can consider each standardised prediction error

$$D_i^I(\mathbf{y}^{(v)}) = \frac{y_i^{(v)} - E[\eta(\mathbf{x}_i^{(v)})|\mathbf{y}]}{\sqrt{V[\eta(\mathbf{x}_i^{(v)})|\mathbf{y}]}} \tag{4.2}$$

as a diagnostic. If the emulator represents uncertainty about the simulator properly, the standardised prediction errors have standard Student-t distributions, conditional on the training data and the estimated correlation parameters δ. In practice, the number of training data is generally large enough so that the degrees of freedom is large and we can consider these to be standard normally distributed. Hence, individual large errors, with absolute values larger than 2, say, indicate a conflict between the simulator and the emulator. An isolated outlier of this kind might be ignored, or might indicate a local problem just around the inputs for that validation data point. This can be investigated further by obtaining a few more validation data runs in that vicinity.

If there are a larger number of large standardised errors, this would indicate a more systematic problem. If large errors of the same sign arise in some part of the input space, this suggests an inappropriate choice of mean function. It may also be indicative of a failure of the stationarity assumption.

If large errors arise primarily in validation points that are close to training data points, this indicates that one of more on the correlation parameters have been over-estimated, so that the emulator predictions are too strongly influenced by nearby training data points. If there are no such obvious patterns in the occurrence of large errors, then the problem may occur due to lack of homoscedasticity.

Patterns of unexpectedly small standardised errors may indicate conflicts complementary to those discussed above. For instance, if validation points close to training data points give unexpectedly small standardised errors, this suggests under-estimation of correlation parameters. Graphical displays can be powerful ways of spotting patterns of large or small errors, and are discussed in Section 4.3.4.

4.3.2 Mahalanobis distance

Although the collection of individual standardised errors $D^I(\mathbf{y}^{(v)})$ provide a range of useful diagnostics, it is also important to be able to summarise them in a single diagnostic.

Hills and Trucano (1999, 2001) use a χ^2-test to compare the simulator output with the real process in a V&V approach. The same idea can be used as a diagnostic to compare emulator predictions with the simulator outputs under the same inputs. Their diagnostic is given by

$$D_{\chi^2}(\mathbf{y}^{(v)}) = \sum_{i=1}^{m} D_i^I(\mathbf{y}^{(v)})^2. \tag{4.3}$$

For a large training dataset, i.e. when $n \to \infty$, and with independence among the output values, the distribution of $D_{\chi^2}\left(\eta(\mathbf{X}^{(v)})\right)$ converges to a chi-squared distribution with m degrees of freedom. However, the independence assumption is too strong. For example, if the simulator is a smooth function, then similar outputs are expected when the inputs are close to each other in the input space. This correlation is captured by the emulator in the covariance function.

A natural extension of (4.3) allowing the correlation among the outputs is the Mahalanobis distance between the emulator and the simulator outputs at the validation inputs set, given

by

$$D_{MD}(\mathbf{y}^{(v)}) = \left(\mathbf{y}^{(v)} - E\left[\eta(\mathbf{X}^{(v)})|\mathbf{y}\right]\right)^T \left(V\left[\eta(\mathbf{X}^{(v)})|\mathbf{y}\right]\right)^{-1} \left(\mathbf{y}^{(v)} - E\left[\eta(\mathbf{X}^{(v)})|\mathbf{y}\right]\right), \quad (4.4)$$

where the elements of the predictive mean vector $E\left[\eta(\mathbf{X}^{(v)})|\mathbf{y}\right]$ and the predictive covariance matrix $V\left[\eta(\mathbf{X}^{(v)})|\mathbf{y}\right]$ for the Gaussian process emulator are given respectively by (2.17) and (2.18). Extreme values (large or small) for the observed Mahalanobis distance ($D_{MD}(\mathbf{y}^{(v)})$) indicate a conflict between the emulator and simulator.

Theorem 1 *Under Gaussian process emulator assumptions, the distribution of $D_{MD}(\eta(\mathbf{X}^{(v)}))$ conditional on the training data and an estimate of the correlation parameter δ is a scaled F-Snedecor distribution with m and $n-q$ degrees of freedom,*

$$\frac{(n-q)}{m(n-q-2)} D_{MD}(\eta(\mathbf{X}^{(v)}))|\mathbf{y}, \delta \sim F_{m,n-q}. \quad (4.5)$$

Proof: Define $W = \frac{(n-q)}{m(n-q-2)} D_{MD}(\eta(\mathbf{X}^{(v)}))$. Conditional on the training data, σ^2 and δ, we have that

$$W = \frac{\sigma^2(n-q)}{\hat{\sigma}^2 m(n-q-2)} Z, \quad (4.6)$$

where by (2.13), $Z = (\eta(\mathbf{X}^{(v)}) - m_1(\mathbf{X}^{(v)}))^T \left(V_1^*(\mathbf{X}^{(v)})\right)^{-1} (\eta(\mathbf{X}^{(v)}) - m_1(\mathbf{X}^{(v)}))$ follows a chi-squared distribution with m degrees of freedom. As a result,

$$W|\sigma^2, \mathbf{y}, \delta \sim \text{Gamma}\left(\frac{m}{2}, \frac{\hat{\sigma}^2 m(n-q-2)}{2\sigma^2(n-q)}\right).$$

The distribution of σ^2 conditional on $\mathbf{y}$ and δ is given by (2.12). The distribution of W after integrating σ^2 out is given by

$$\begin{aligned} p(w|\mathbf{y}, \delta) &= \int_0^\infty p(w|\sigma^2, \mathbf{y}, \delta) p(\sigma^2|\mathbf{y}, \delta) \mathrm{d}\sigma^2 \\ &= \frac{\left(\hat{\sigma}^2(n-q-2)\right)^{\frac{m+n-q}{2}}}{2^{\frac{m+n-q}{2}} \Gamma\left(\frac{m}{2}\right) \Gamma\left(\frac{n-q}{2}\right)} \left(\frac{mw}{n-q}\right)^{\frac{m}{2}} \frac{1}{w} \\ &\times \int_0^\infty (\sigma^2)^{-\frac{m+n-q}{2}-1} \exp\left\{-\frac{1}{\sigma^2} \frac{(n-q-2)\hat{\sigma}^2}{2} \left(1 + \frac{mw}{n-q}\right)\right\} \mathrm{d}\sigma^2. \quad (4.7) \end{aligned}$$

The integrand is the kernel of an inverse Gamma distribution with parameters $\frac{m+n-q}{2}$ and $\frac{(n-q-2)\hat{\sigma}^2}{2}\left(1+\frac{mw}{n-q}\right)$. Therefore

$$p(w|\mathbf{y},\delta) = \frac{\Gamma\left(\frac{m+n-q}{2}\right)}{\Gamma\left(\frac{m}{2}\right)\Gamma\left(\frac{n-q}{2}\right)}\frac{(mw)^{\frac{m}{2}}}{w}\frac{(n-q)^{\frac{n-q}{2}}}{[(n-q)+mw]^{\frac{m+n-q}{2}}}, \quad w>0, \tag{4.8}$$

which is the density of a F-Snedecor distribution with m and $n-q$ degrees of freedom. □

The expected value and the variance of the Mahalanobis distance are respectively given by

$$\begin{aligned} E[D_{MD}(\eta(\mathbf{X}^{(v)}))|\mathbf{y},\delta] &= m, \\ \mathrm{Var}[D_{MD}(\eta(\mathbf{X}^{(v)}))|\mathbf{y},\delta] &= \frac{2m(m+n-q-2)}{n-q-4}. \end{aligned}$$

As previously mentioned, an unexpectedly large or small value of $D_{MD}(\mathbf{y}^{(v)})$ indicates a conflict between the emulator and the simulator. If such a problem arises, it is important to explore individual errors, to look for patterns of large or small values, so as to identify the most likely cause of the problem. We now consider alternative ways to decompose the Mahalanobis distance into individual diagnostics for this purpose.

4.3.3 Variance decompositions

Individual prediction errors (4.2) are correlated, which introduces some risks in interpreting them. Also, looking at individual errors may not effectively identify some conflicts between the emulator and simulator. For instance, two errors may not individually be large, but if they have opposite signs when they are strongly positively correlated, then this suggests a conflict. Let $\mathbf{G}$ be a standard deviation matrix such that $V[\eta(\mathbf{X}^{(v)})|\mathbf{y}] = \mathbf{G}\mathbf{G}^T$. Then the vector of transformed errors

$$D_{\mathbf{G}}(\mathbf{y}^{(v)}) = \mathbf{G}^{-1}(\mathbf{y}^{(v)} - E[\eta(\mathbf{X}^{(v)})|\mathbf{y}]) \tag{4.9}$$

are uncorrelated and have unit variances. If the normality assumption made for the outputs is reasonable, the distribution of each of these errors is a standard Student-t with $(n-q)$ degrees of freedom. We can consider these as an alternative set of diagnostics. As with the errors $D^I(\mathbf{y}^{(v)})$, we look for individual large transformed errors and patterns of large and small values. However, the structure of $\mathbf{G}$ will give different interpretations to such patterns. This approach is similar to that of Houseman et al. (2004), who use rotated residuals for linear models with correlated errors.

Another property of this diagnostic is that $D_{MD}(\mathbf{y}^{(v)}) = D_{\mathbf{G}}(\mathbf{y}^{(v)})^T D_{\mathbf{G}}(\mathbf{y}^{(v)})$. That is, the sum of squares of the elements of $D_{\mathbf{G}}(\mathbf{y}^{(v)})$ is the Mahalanobis distance, and hence we can interpret these diagnostics as decomposing $D_{MD}(\mathbf{y}^{(v)})$.

There are many ways to decompose a positive definite matrix into the product of a square root matrix and its transpose. The natural choices are the Cholesky decomposition and the eigen decomposition (Golub and van Loan 1996). The eigen decomposition is very popular, but the Cholesky decomposition is computationally cheaper and we shall see that it is more intuitive to interpret than the eigen decomposition.

Eigen decomposition. When $\mathbf{G}$ is the eigen decomposition matrix, we denote the elements of the vector $D_{\mathbf{G}}(\mathbf{y}^{(v)})$ by $D^E(\mathbf{y}^{(v)})$, and call them eigen errors. When a large $D_i^E(\mathbf{y}^{(v)})$ is identified, further information may be gained by studying which individual errors are given the largest weights in the linear combination of the individual predictive errors. If the weights single out as important a particular individual error, then this error should be studied as suggested in Section 4.3.1. If the weights emphasize a subset of the individual prediction errors, then it might indicate a problem in the region of the input space around the inputs of the validation data points associated, indicating a possible non-stationarity problem.

Cholesky decomposition. The Cholesky decomposition is the special case where $\mathbf{G}^T$ is the unique upper triangular matrix $\mathbf{R}$ such that $V[\eta(\mathbf{X}^{(v)})|\mathbf{y}] = \mathbf{R}^T\mathbf{R}$, and we denote the elements of the vector $D_{\mathbf{G}}(\mathbf{y}^{(v)})$ by $D^C(\mathbf{y}^{(v)})$, and call them Cholesky errors. Then $\mathbf{G}^{-1}$ is also a triangular matrix, and $D_i^C(\mathbf{y}^{(v)})$ is the unique linear combination of the first

i validation errors such that its predictive variance is the conditional variance of the i-th validation error given the preceding $i-1$ errors. Although this has the benefit of producing a set of uncorrelated transformed errors that are still linked to the individual validation points (in contrast to the eigen decomposition), the decomposition is not invariant to how we order the validation points, and patterns of high or low values have no obvious interpretation.

Pivoted Cholesky decomposition. By permuting the validation dataset, we obtain different Cholesky decompositions. Any such permutation may detect different anomalies. However, in order to have the benefits of both the eigen and Cholesky decompositions, we have found the most effective diagnostics are achieved by permuting the data so that the first element is the one with the largest variance, the second element is the one with the largest predictive variance conditioned on the first element, and so on. We then denote the elements of the vector $D_{\mathbf{G}}(\mathbf{y}^{(v)})$ by $D^{PC}(\mathbf{y}^{(v)})$, and call them pivoted Cholesky errors. This permutation can be obtained by applying the pivoted Cholesky decomposition, which returns a permutation matrix $\mathbf{P}$ and the unique upper triangular matrix $\mathbf{R}$ such that $\mathbf{P}^T V[\eta(\mathbf{X}^{(v)})|\mathbf{y}]\mathbf{P} = \mathbf{R}^T\mathbf{R}$. So $\mathbf{G} = \mathbf{P}\mathbf{R}^T$. More details about the pivoted Cholesky decomposition including the algorithm are presented in Section 4.4.

A group of unusually large or small pivoted Cholesky errors in the first part of the sequence suggest non-homogeneity, while a number of unusually large or small errors in the latter part of the sequence indicate poor estimation of δ or an inappropriate correlation structure. In addition, we have the benefit that each of the $D^{PC}(\mathbf{y}^{(v)})$s is associated with a particular validation data point, which makes it easy to investigate individual large errors. The pivoted Cholesky decomposition will therefore be our choice in the examples of Section 4.5.

4.3.4 Graphical methods

Graphical displays are efficient ways to investigate the adequacy of the emulator predictions and to check some assumptions made to build the emulator (2.16). We propose some graphical methods using both the individual standardised errors (4.2) and the uncorrelated standardised errors (4.9).

Plot of the individual errors against the emulator's predictions. In this graphical diagnostic, we search for patterns suggesting a problem in the mean function. For example, if for some particular ranges of the output the errors are systematically positive (or negative) this indicates a misspecification of the mean function. Heteroscedasticity of the individual errors suggests that the simulator should be studied as a non-stationary process. Large absolute individual errors might suggest that the predictive variance is too small, and individual errors very close to zero might suggest too large a variance.

In addition to plotting individual errors $D^I(\mathbf{y}^{(v)})$ in this way, we can plot the uncorrelated standardised errors obtained by the Cholesky or pivoted Cholesky decomposition, because each error can be mapped to one emulator prediction. However, we are then less likely to see groups of systematic deviations indicating problems with the mean function. Consider, for instance, a group of positive individual errors in some part of the plot. These may arise from validation points that are relatively close together in the input space. The first of these points to be plotted in the Cholesky or pivoted Cholesky sequences $D^C(\mathbf{y}^{(v)})$ or $D^{PC}(\mathbf{y}^{(v)})$ may show up in the plot as a large error, but the subsequent ones are conditioned on the first and may appear normal.

We cannot plot the uncorrelated standardised errors obtained by eigen decomposition in this way.

Plot of the errors against the index. The meaning of the index depends on which error we are plotting. For individual errors $D^I(\mathbf{y}^{(v)})$ and Cholesky errors $D^C(\mathbf{y}^{(v)})$, the index is the validation data order. For eigen errors, the index gives the order of the $D^E(\mathbf{y}^{(v)})$s with the largest predictive variance. For pivoted Cholesky errors, the index is the pivoting order, which gives the order of the $D^{PC}(\mathbf{y}^{(v)})$s with the largest conditional predictive variance. For all these graphics, it is expected that the errors should be fluctuating around zero with a constant variance and no special patterns. Too many large errors indicates an underestimation of the variance. On the other hand, too many small errors indicates an overestimation of the variance. In both cases, it can also suggest that the simulator is a non-stationary process.

The pivoted Cholesky and the eigen decomposition provide an extra interpretation that we can associate with the correlation structure. In both cases, if we observe either large or very small errors at the beginning of the plot, i.e., on the left-hand side, it indicates a failure of estimation of predictive variance, or non-stationarity. If we observe large (or very small) errors at the end of the plot, however, i.e., on the right-hand side, it indicates that the correlation length parameters were over (under) estimated or the chosen correlation structure is unsuitable.

Quantile-quantile plots. Under the normality assumption, the uncorrelated standardised errors $D_G(\mathbf{y}^{(v)})$ have standard Student-t distributions with $(n-q)$ degrees of freedom. So, the quantile-quantile plot (QQ-plot) using this distribution becomes a natural graphical diagnostic. In a QQ-plot, if the points lie close to the 45-degree line through the origin, the normality assumption for the simulator outputs is a reasonable assumption. If the points cluster around a line with slope less (or greater) than one, the implication is that the predictive variability was over-estimated (or under-estimated).

Curvature in the plot indicates non-normality, while outliers at either end of the plot suggest local fitting problems or non-stationarity.

The interpretation of the QQ-plot using uncorrelated standardised errors is independent of the decomposition method, and is generally informative for both eigen and pivoted Cholesky decompositions. Although the distribution of each individual standardised error, $D_i^I(\mathbf{y}^{(v)})$, is also a standard student-t distribution, the fact that the errors are correlated makes the QQ-plot more difficult to interpret.

Plots of errors against inputs. Plotting the standardised errors against the corresponding values of each input is also helpful. Again, we expect to see a horizontal band containing the errors. These plots are used to identify different behaviour of the errors in some parts of the input space, indicating possible failure of the stationarity assumption. This graphic can also indicate that the relationship between the input and the prediction is not fully represented in the mean function. For example, we can identify a pattern that was not included in the mean function.

Notice that we cannot plot the eigen errors in this way. Whilst it is possible to plot the Cholesky or pivoted Cholesky errors against input values, because of the linking of each error to a validation data point, interpretation is complicated by the conditioning in the same way as when plotting against the emulator mean.

4.3.5 Other diagnostics

Credible interval diagnostic. Another diagnostic is the proportion of validation outputs which lie in their marginal credible intervals. For each validation element using (2.16) a $100\alpha\%$ credible interval for the simulator output can be built, denoted by $CI_i(\alpha)$ for $i = 1, \ldots, m$. This diagnostic is given by

$$D_{CI}(\mathbf{y}^{(v)}) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(\mathbf{y}_i^{(v)} \in CI_i(\alpha)) \tag{4.10}$$

where $\mathbf{1}(\cdot)$ is an indicator function. We expect that the observed value for $D_{CI}(\mathbf{y}^{(v)})$ should be close to α. However, because the outputs are not independent the reference distribution of $D_{CI}(\cdot)$ is not binomial. The only practical way to compute the reference distribution is by simulation.

The distribution of $D_{CI}(\cdot)$ can be obtained by the following Monte Carlo simulation. We sample a large number of samples from the multivariate Student-t distribution with $n - q$ degrees of freedom, mean vetor $E[\eta(\mathbf{X}^{(v)})|\mathbf{y}]$ and covariance matrix $V[\eta(\mathbf{X}^{(v)})|\mathbf{y}]$, and then for each sample we calculate $D_{CI}(\cdot)$. The empirical distribution of the calculated $D_{CI}(\cdot)$s is a good estimate of the distribution of $D_{CI}(\eta(\mathbf{X}^{(v)})$. In particular, the mean and the square of standard deviation are respectively estimates of the expectation and the variance of $D_{CI}(\eta(\mathbf{X}^{(v)})$.

This diagnostic is a supplement to the Mahalanobis distance. For example, we can have many unusually large and unusually small errors and yet still have an acceptable value for $D_{MD}(\mathbf{y}^{(v)})$. The $D_{CI}(\mathbf{y}^{(v)})$ diagnostic offers a way to identify this kind of heterogeneity.

Predictive density diagnostic. It is worth mentioning here another interpretation of the Mahalanobis distance. Since the distribution of validation outputs is a multivariate Student-t distribution with mean $m_1(\mathbf{X}^{(v)})$, covariance matrix $V_1(\mathbf{X}^{(v)})$ and $n-q$ degrees of freedom, the density function itself can be a diagnostic,

$$D_{PD}(\mathbf{y}^{(v)}) = K\left[1+\frac{1}{n-q}D_{MD}(\mathbf{y}^{(v)})\right]^{-\frac{m+n-q}{2}} \tag{4.11}$$

where $K = \frac{\Gamma\left(\frac{m+n-q}{2}\right)}{\Gamma\left(\frac{n-q}{2}\right)}((n-q)\pi)^{-\frac{m}{2}}|V_1(\mathbf{X}^{(v)})|^{-\frac{1}{2}}$. A small value for this diagnostic would indicate a conflict between the emulator and simulator. Note, however, that $D_{PD}(\mathbf{y}^{(v)})$ is just a decreasing function of the Mahalanobis distance $D_{MD}(\mathbf{y}^{(v)})$, and so small values of the predictive density correspond directly with large values of the Mahalanobis distance.

4.4 Pivoted Cholesky decomposition

The Cholesky decomposition was developed by the French mathematician André-Louis Cholesky, and published after his death by Benoît (1924). The method received little attention after its publication. However, the Cholesky decomposition was analysed by Fox et al. (1948) and, in the same year, Turing (1948) presented a result on stability of the method. After that, the Cholesky decomposition became very popular. See more details about historical context in Taussky and Todd (2006) and Brezinski (2006).

Matrices are denoted in this work by bold capital letters, $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$, and submatrices are specified with the colon notation, as used in Golub and van Loan (1996). $\mathbf{A}(p{:}q, r{:}s)$ denotes the submatrix of $\mathbf{A}$ formed by the intersection of rows p to q and columns r to s. Particular cases: $\mathbf{A}(i,j)$ denotes the element in row i and column j, $\mathbf{A}(i,:)$ denotes row i, and $\mathbf{A}(:,j)$ column j.

Formally, the Cholesky method decomposes a symmetric positive definite matrix $\mathbf{A}$ uniquely into a product of an upper triangular matrix $\mathbf{R}$ and its transpose, i.e. $\mathbf{A} = \mathbf{R}^T\mathbf{R}$, or equivalently $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is a lower triangular matrix. This method is the iterative

process described in Algorithm 4.1, where basically at each step k the row k of matrix $\mathbf{R}$ is calculated and the matrix $\mathbf{A}(k:n, k:n)$ is updated and used in an iterative process.

Algorithm 4.1 This algorithm computes the Cholesky decomposition $\mathbf{A} = \mathbf{R}^T\mathbf{R}$ of a symmetric positive definite matrix $\mathbf{A} \in \Re^{n \times n}$

$\mathbf{R} = \mathbf{0}$ {define a $n \times n$ zero matrix}
for $k = 1$ to n **do**
  $\mathbf{R}(k,k) = \sqrt{\mathbf{A}(k,k)}$
  $\mathbf{R}(k, k+1:n) = \mathbf{R}(k,k)^{-1}\mathbf{A}(k, k+1:n)$
  $\mathbf{A}(k+1:n, k+1:n) = \mathbf{A}(k+1:n, k+1:n) - \mathbf{R}(k, k+1:n)^T\mathbf{R}(k, k+1:n)$
end for

Using the Cholesky decomposition the determinant of $\mathbf{A}$ is given by the square of the product of the diagonal values of matrix $\mathbf{R}$, $det(\mathbf{A}) = \prod_{i=1}^{n} \mathbf{R}(i,i)^2$, and the inverse is obtained by $(\mathbf{A})^{-1} = (\mathbf{R}^T\mathbf{R})^{-1} = \mathbf{R}^{-1}(\mathbf{R}^{-1})^T$, where $\mathbf{R}^{-1}$ is calculated by the backward substitution method (Gentle 1998). The Cholesky decomposition is often used to solve the linear system $\mathbf{A}x = b$, when $\mathbf{A}$ is symmetric positive definite. For example, the equations of the linear least squares problem are of this form (Gentle 1998).

An arbitrary permutation of rows and columns of matrix $\mathbf{A}$ can be decomposed by the Cholesky algorithm, $\mathbf{P}^T\mathbf{A}\mathbf{P} = \mathbf{R}^T\mathbf{R}$, where $\mathbf{P}$ is a permutation matrix and $\mathbf{R}$ is an upper triangular matrix. The permutation matrix is an orthogonal matrix, so the matrix $\mathbf{A}$ can be rewritten as $\mathbf{A} = (\mathbf{R}\mathbf{P}^T)^T\mathbf{R}\mathbf{P}^T$, so that $\mathbf{R}\mathbf{P}^T$ is a decomposition of $\mathbf{A}$. Although $\mathbf{R}\mathbf{P}^T$ is not the Cholesky decomposition of $\mathbf{A}$, because it is not triangular, $\mathbf{R}\mathbf{P}^T$ can be computed quickly using a simple modification of Algorithm 4.1.

The process of finding $\mathbf{R}$ and $\mathbf{P}$ for symmetric positive definite matrices is called the pivoted Cholesky decomposition (PCD). The PCD is an extension of Cholesky decomposition where a pivoting step is included. The pivoting step consists of finding, at each iteration k, the largest possible element of the diagonal of the updated matrix $\mathbf{A}(k{:}n, k{:}n)$. The index of the largest element is saved and called the pivot. The PCD algorithm, which is an extension of the Cholesky decomposition Algorithm 4.1, is described in Algorithm 4.2.

Notice that the determinant of $\mathbf{A}$ is the same as the determinant of its symmetric permutation $\mathbf{P^T}\mathbf{A}\mathbf{P}$, i.e. $det(\mathbf{A}) = det(\mathbf{P^T}\mathbf{A}\mathbf{P})$, but, in order to find the inverse of $\mathbf{A}$,

Algorithm 4.2 This algorithm computes the pivoted Cholesky decomposition $\mathbf{P^T A P} = \mathbf{R}^T\mathbf{R}$ of a symmetric positive semidefinite matrix $\mathbf{A} \in \Re^{n\times n}$. The nonzero elements of the permutation matrix $\mathbf{P}$ are given by $\mathbf{P}(piv(k), k) = 1,\ k = 1, \ldots, n$

$\mathbf{R} = \mathbf{0}$ {define a $n \times n$ zero matrix}
$piv = 1 : n$
for $k = 1$ to n **do**
  $\mathbf{B} = \mathbf{A}(k : n, k : n)$
  $q = \{i : \mathbf{A}(i, i) = \max(diag(\mathbf{B}))\}$ {Finding the pivot}
  $\mathbf{A}(:, k) \leftrightarrows \mathbf{A}(:, q)$ {Swap columns}
  $\mathbf{R}(:, k) \leftrightarrows \mathbf{R}(:, q)$ {Swap columns}
  $\mathbf{A}(k, :) \leftrightarrows \mathbf{A}(q, :)$ {Swap rows}
  $piv(k) \leftrightarrows piv(q)$ {Swap pivoting position}
  $\mathbf{R}(k, k) = \sqrt{\mathbf{A}(k, k)}$
  $\mathbf{R}(k, k + 1 : n) = \mathbf{R}(k, k)^{-1}\mathbf{A}(k, k + 1 : n)$
  $\mathbf{A}(k + 1 : n, k + 1 : n) = \mathbf{A}(k + 1 : n, k + 1 : n) - \mathbf{R}(k, k + 1 : n)^T\mathbf{R}(k, k + 1 : n)$
end for

the following relationship using the inverse of its symmetric permutation can be used: $(\mathbf{A})^{-1} = \mathbf{P}(\mathbf{P^T A P})^{-\mathbf{1}}\mathbf{P^T}$, where $(\mathbf{P^T A P})^{-\mathbf{1}} = \mathbf{R}^{-1}(\mathbf{R}^{-1})^T$.

More information about the numerical analysis of the pivoted Cholesky decomposition can be found in Higham (2002).

4.5 Examples

In this section, we use two datasets to illustrate the proposed diagnostics for Gaussian process emulators. The first example is an artificial model with two inputs, while the second example is a model of reflectance for a homogeneous plant canopy. In both examples, the prior relationship between the output and the inputs is represented by the mean function (2.2) with $h(\mathbf{x})^T = (1, \mathbf{x}^T)$. This envisages a linear trend in response to each input, which is widely used for Gaussian process emulation. Although computer models are in practice almost certainly nonlinear, we often have little prior knowledge of what form the nonlinearity will take.

4.5.1 Two-input toy model

We perform our diagnostics in example 2.2, where the training data are composed of 20 points selected by Latin hypercube sampling. The validation data are composed of 25 points independently selected using another Latin hypercube sample. Figure 4.1 presents the training and the validation datasets. Using the training data, the estimated correlation length parameters are $(\hat{\delta}_1, \hat{\delta}_2) = (0.2421, 0.4240)$.

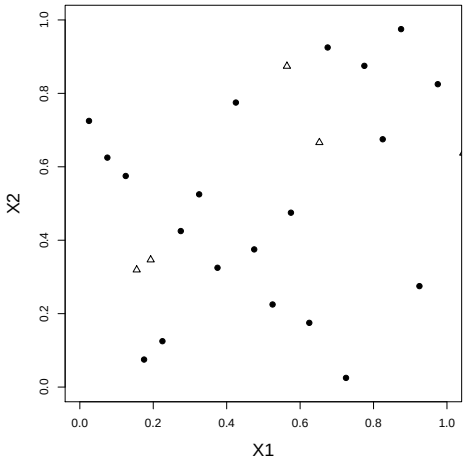


Figure 4.1: Training ($\bullet$) and validation ($\triangle$) datasets sampled from independent Latin hypercube sampling scheme.

Then, we predict the output for the validation points using a Student-t process (2.16) conditional on the training data and the estimated correlation lengths. The observed chi-square diagnostic, $D_{\chi^2}(\mathbf{y}^{(v)}) = 24.411$, is very close to its expected value $E[D_{\chi^2}(\eta(\mathbf{X}^{(v)}))] = 25$, suggesting that the emulator is a good approximation of the simulator. However, this diagnostic ignores the fact that the outputs are correlated. Table 4.1 presents the observed Mahalanobis distance and credible interval diagnostics, and some statistics of their predictive distributions.

The observed Mahalanobis distance, $D_{MD}(\mathbf{y}^{(v)}) = 70.36$, is an extreme value of its theoretical distribution, a scaled F distribution with parameters (25,18). This indicates a conflict between the emulator and the simulator. The observed credible interval diagnostic is less dramatic. We see that 92% (23 out of 25) of the simulator outputs lie in their respective

95% marginal credible intervals built by the emulator. The lower quartile of the distribution of $D_{CI}(\eta(\mathbf{X}^{(v)}))$ obtained via simulation, is 0.92, and indeed it is not surprising to have 2 values out of 25 lying outside their 95% intervals.

The chi-square and credible interval diagnostics suggest that emulator predictions are marginally satisfactory but the Mahalanobis distance makes it clear that jointly they are far from valid. This may be related to poor estimation of the correlation length parameters or to a non-homogeneity in the input space.

Table 4.1: The observed Mahalanobis distance and credible interval diagnostics, with some summaries of their predictive distributions, for the toy example.

	Obs.	Expected	Std. dev.	1stQ	Median	3rdQ
$D_{MD}(\cdot)$	70.360	25.000	12.404	16.016	21.778	29.438
$D_{CI}(\cdot)$	0.920	0.951	0.072	0.920	1.000	1.000

Figure 4.2 presents some graphical diagnostics using the individual standardised errors. Figure 4.2 (a) presents the individual standardised errors against the emulator predictions given by the expected value. There is no obvious pattern, although the two largest individual errors are associated with small values of the predictions, which might indicate a problem in the mean function or an stationarity problem. Figure 4.2 (b) is the QQ-plot of the individual errors, and supports the finding of the chi-square and credible interval diagnostics that the predictions appear valid marginally.

In order to check a possible stationarity problem, the individual standardised errors are plotted against each input, Figure 4.2 (c) and (d). The two large errors are associated with small values of the input 1, and large values of the input 2. This might be connected to a sub-region of the input space not represented in the training data. If possible, new runs of the simulator in this sub-region may improve the emulator. Also, it can be seen in Figure 4.2 (c) that the larger the value of input 1, the smaller is the variability of the individual errors. This can indicate non-stationarity, or perhaps an over estimation of the correlation length δ_1. For the second input, Figure 4.2 (d), there is no particular pattern to the errors.

The diagnostics presented in Figure 4.2 ignore the correlation structure of the errors, and

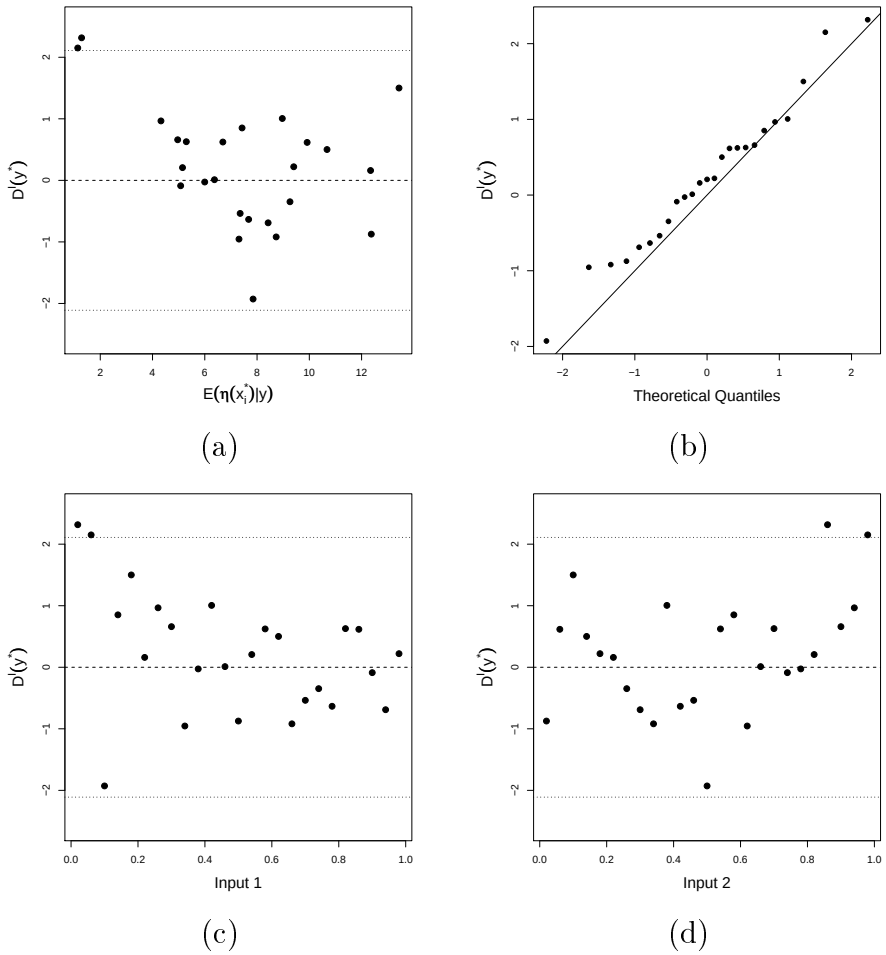
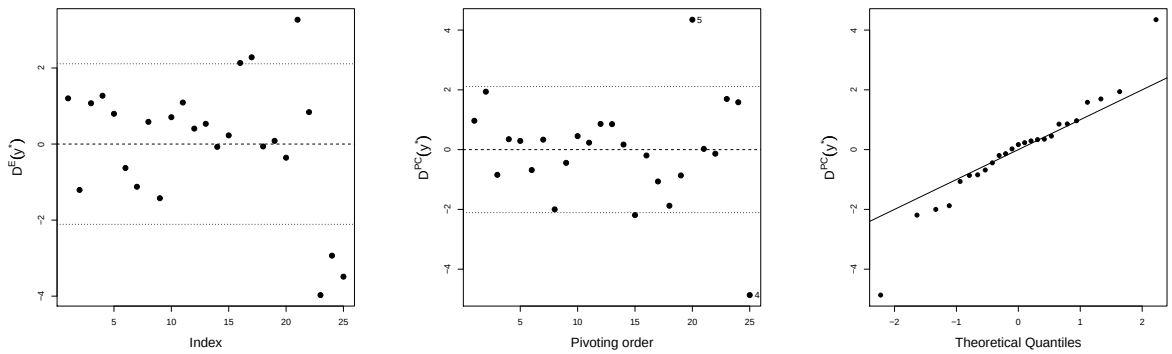


Figure 4.2: Graphical diagnostics for the toy example using the individual standardised errors: (a) $D^I(\mathbf{y}^{(v)})$ against the emulator predictions; (b) quantile-quantile plot; (c) $D_i(\mathbf{y}^{(v)})$ against input 1; (d) $D_i(\mathbf{y}^{(v)})$ against input 2.

so do not address the problem observed using the Mahalanobis distance diagnostic. Figure 4.3 presents graphical diagnostics using the uncorrelated standardised errors. The eigen errors are presented in Figure 4.3 (a), where we can observe large errors at the end of the plot. This clearly indicates a problem with the correlation structure, which could be either a misspecification of the correlation function, or an over-estimation of the correlation length parameters. When we examine the weights given by the eigen vector for the large errors, we cannot identify a pattern for the components of the 16th, 17th and 23rd eigen errors. However, the components of the 21st eigen value indicate the 5th validation point as a very important point for the size of this eigen error. And all validation points related to the largest weights of the 24th and 25th eigen errors present values for input 1 smaller than 0.5.

The pivoted Cholesky errors are presented in Figure 4.3 (b). Large errors are again observed at the end of the plot in this diagnostic, suggesting a problem with the correlation structure. However, with this plot it is easier to explore the nature of the problem, because there are only two large values and we can link each one to a single validation data point. Their input vectors are (0.14, 0.58) and (0.18, 0.10). These two points are different from the two large individual standardised errors, but they also are characterized by small values of input 1, suggesting that the emulator predictions are not valid over this part of the input space.

Figure 4.3 (c) presents the quantile-quantile plot of the pivoted Cholesky errors, where it can be seen that the points cluster around a line with slope slightly greater than one, so that there may be a small under-estimation of the predictive variability. However, the two outliers, which were the two large values in Figure 4.3 (b), are the most striking feature of this plot.



(a) (b) (c)

Figure 4.3: Graphical diagnostics for the toy example using the uncorrelated standardised errors: (a) the eigen errors $D^E(\mathbf{y}^{(v)})$, against the eigenvalue number; (b) the pivoted Cholesky errors $D^{PC}(\mathbf{y}^{(v)})$, against the pivoting order; (c) quantile-quantile plot of the pivoted Cholesky errors.

In summary, the numerical and graphical diagnostics indicate some conflicts between the emulator and the simulator. The conflict seems to be related to poor estimation of the correlation parameters, and perhaps to non-stationarity and the training data not adequately covering a sub-region of input space where X_1 takes values below 0.2.

Since the simulator (2.20) is a simple function we can quickly run it more times. So, we combine the training data with the validation data, plus another 5 observations randomly selected in a subregion of the input space where both inputs are smaller than 0.5. Using the updated training data, the estimated correlation length parameters are $(\hat{\delta}_1, \hat{\delta}_2) = (0.1764, 0.4116)$, which are smaller than the previous estimates, confirming the suspicion of over-estimation, especially δ_1.

A new validation dataset using a Latin hypercube sampling was selected, and the Mahalanobis distance and credible interval diagnostics are presented in Table 4.2. The Mahalanobis distance is still higher than expected, but now suggests much less conflict with the simulator. The credible interval diagnostic again indicates that the emulator predictions for the validation data are individually reasonable.

Table 4.2: The observed Mahalanobis distance and credible interval diagnostics, with some summaries of their predictive distributions, for the toy example after new training data.

	Obs.	Expected	Std. dev.	1stQ	Median	3rdQ
$D_{MD}(\cdot)$	51.129	30	10.230	23.172	28.958	36.324
$D_{CI}(\cdot)$	0.933	0.950	0.058	0.933	0.967	1.000

Figure 4.4 (a) presents the individual standardised errors. There is no obvious pattern, and although there are some errors outside the credibility bounds, they are not large enough to suggest a serious conflict. Figure 4.4 (b) suggests that the emulator is underestimating the highest output values, and so is perhaps not adequately capturing the peak of the output surface.

The pivoted Cholesky errors are presented in Figure 4.4 (c). Although there are no very large errors, there are too many outside the bounds. The QQ-plot of the pivoted Cholesky errors, Figure 4.4 (d), confirms that the emulator's predicted variability is a bit smaller than the observed variability.

Although the diagnostics indicate that there may still be some validation problems, the emulator built with the updated training data does improve the predictions. At this point we may make a final build of the emulator using all the simulator runs (the original training

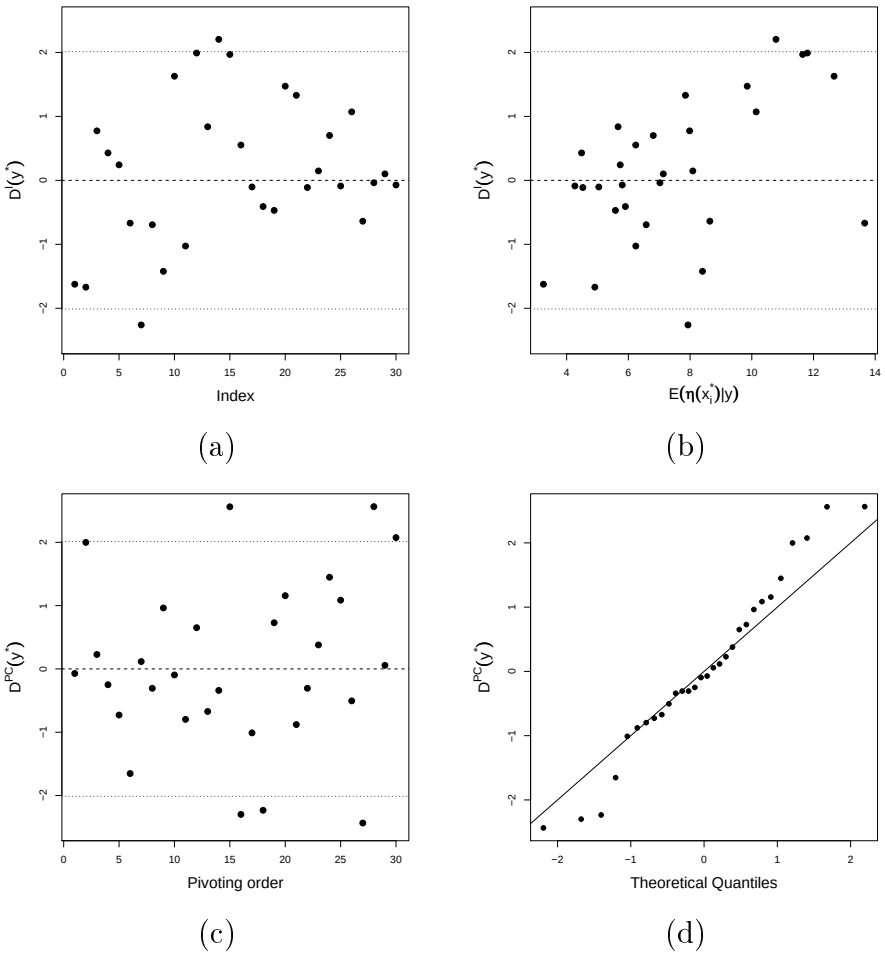


Figure 4.4: Graphical diagnostics for the toy example after new training data: individual standardised errors $D^I(\mathbf{y}^{(v)})$ against (a) the validation data order, and (b) the emulator predictions; (c) pivoted Cholesky errors $D^{PC}(\mathbf{y}^{(v)})$ against the pivoting order; (d) quantile-quantile plot of pivoted Cholesky errors.

data, the original validation data, the extra 5 points and the final validation data).

Figure 4.5 shows the steady improvement of the emulator. Figure 4.5 (a) shows the emulator mean plotted against the two inputs, based on just the original training data. Figures 4.5 (b) and (c) show how the emulator evolves as we add the original validation data and the extra 5 points, and then when we add in the additional validation data. The training data for each emulator and also the validation data used to build the diagnostics are illustrated in each graphic. Figure 4.5 (d) presents the true value of the simulator (a plot that will not generally be available to us with a real simulator). Figure 4.5 (d) shows that this is a difficult function to emulate, which is very sensitive to input 1 when its value is small. The original emulator does not capture this behaviour, but after adding the original validation data and 5 more points the correct shape is emerging. The final emulator in Figure 4.5 (c) does a very good job of representing the simulator.

4.5.2 Nilson-Kuusk model

In this section, a real dataset is used as an example for the proposed diagnostics. The simulator was built based on the Nilson-Kuusk model, which is a reflectance model for a homogeneous plant canopy. The simulator has 5 inputs, the solar zenith angle, the leaf area index, relative leaf size, the Markov clumping parameter and one parameter called λ. For more details of this model, and for the real meanings of these inputs and the outputs, see Nilson and Kuusk (1989) and Kuusk (1996). For our analysis, the inputs were rescaled to make all the input values lie between 0 and 1, and the inputs are referenced as inputs 1 to 5.

The training data and the validation data contain 150 and 100 points, respectively, and are supplied as example data with the GEM-SA software (`http://ctcd.group.shef.ac.uk/gem.html`). The training and the validation data were selected using independent Latin hypercube designs. The estimated correlation length parameters are

$$\hat{\delta} = (0.4607, 1.1183, 2.7996, 2.2590, 0.1470).$$

We see that the correlation decays fastest for input 5, indicating that the model output

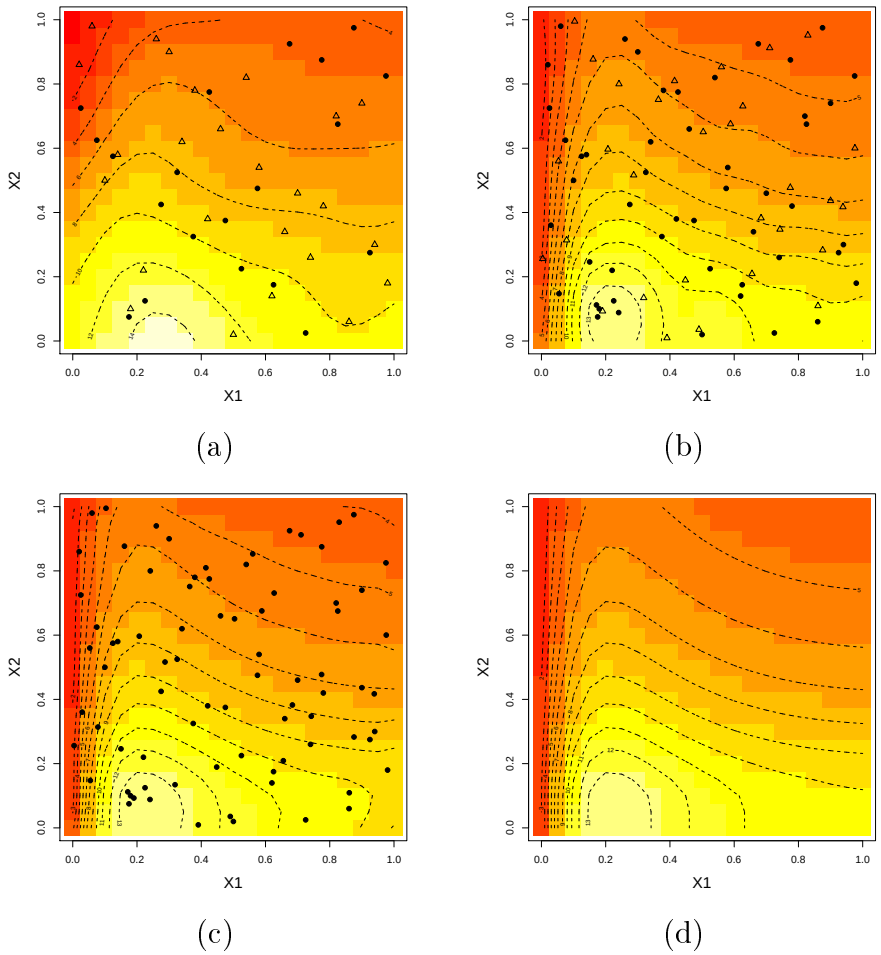


Figure 4.5: Predictive mean of the Gaussian process emulator built with (a) the original training data; (b) the updated training data; (c) all observations as the training data; (d) The two dimensional toy model evaluated over the input space. Training data (•) and validation data (△).

responds most strongly (and most non-linearly) to this input. Input 3, in contrast, has a high correlation length, suggesting that the output responds very smoothly to changes in this input.

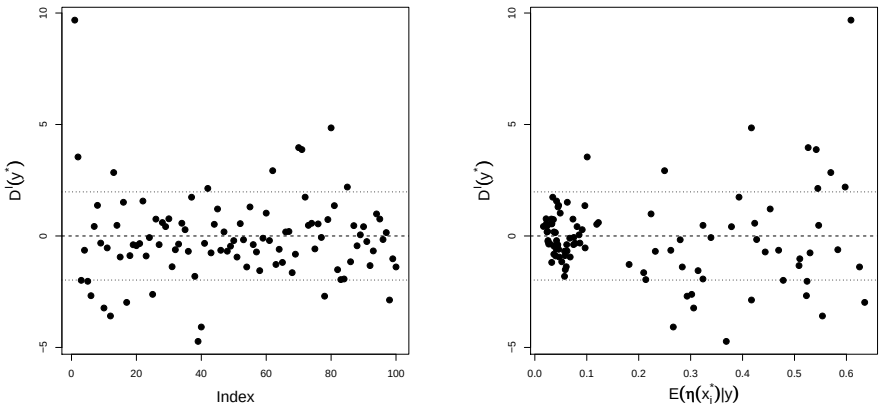
The Mahalanobis distance and credible interval diagnostics are presented in Table 4.3. Both diagnostics point to a major discrepancy between the emulator and the simulator. They also suggest that the variability of the process is greater than was estimated, or the stationarity assumption is too strong.

Figure 4.6 (a) presents the individual standardised errors against the order of the vali-

Table 4.3: The observed Mahalanobis distance and credible interval diagnostics and some summaries of their predictive distributions, Nilson-Kuusk model.

	Obs.	Expected	Std. dev.	1stQ	Median	3rdQ
$D_{MD}(\cdot)$	750.237	100.000	18.593	87.288	98.813	111.964
$D_{CI}(\cdot)$	0.80	0.95	0.0275	0.93	0.95	0.97

dation data. Many large errors can be observed, but there is no particular pattern. The only very large error suggests a local fitting problem around that validation point, while the remaining large errors indicate either an under-estimation of the variability, or a non-stationarity problem. Figure 4.6 (b) plots the individual standardised errors against the emulator predictions given by the expected value. The variability of the errors for small values of the predictions is smaller than the error variability for the large values of the predictions, indicating heteroscedasticity.



(a) (b)

Figure 4.6: Graphical diagnostics for the Nilson-Kuusk model using the individual standardised errors: (a) $D^I(\mathbf{y}^{(v)})$ against the validation data order; (b) $D^I(\mathbf{y}^{(v)})$ against the emulator predictions.

In order to find whether there is a particular subspace in the input space where the behaviour is different, the individual errors are plotted against each input, Figure 4.7. There is no clear systematic pattern for the inputs 1 to 4. However, the variability of the errors seems to depend on whether the input 5 is greater than 0.5, or 700 on the original scale. The last panel in Figure 4.7 plots the model output against input 5 for the combined training and validation data, and shows a clear nonlinearity and change of behaviour in the model

for values of input 5 above 700.

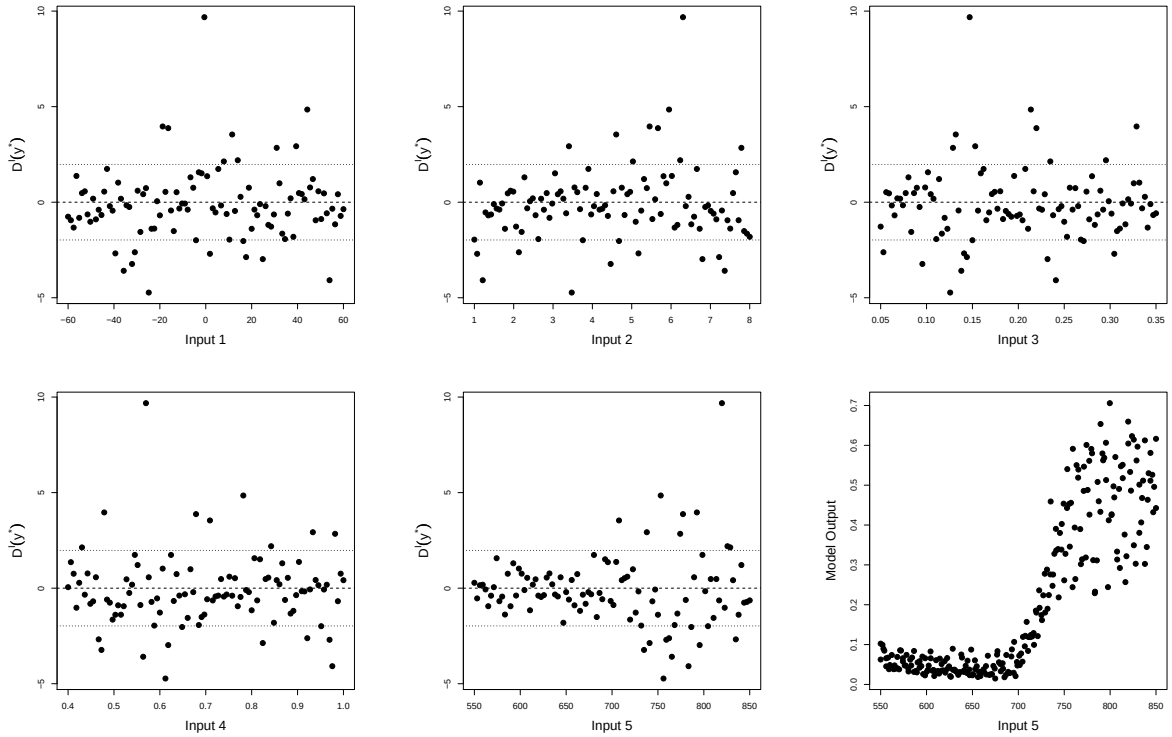
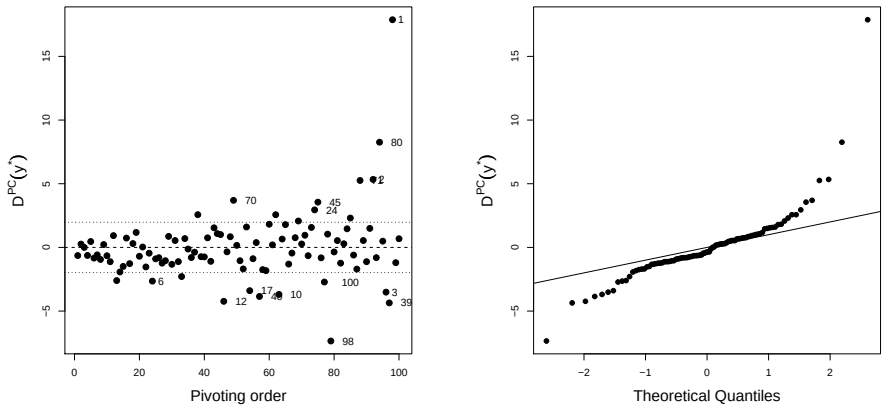


Figure 4.7: Individual standardised errors for the Nilson-Kuusk model, $D^I(\mathbf{y}^{(v)})$, against the five input variables. Also the combined training and validation model outputs against input 5.

The pivoted Cholesky errors are presented in Figure 4.8 (a). Large errors at the end of the plot indicates a possible over-estimation of the correlation length parameters, although the suggested non-stationarity of the model may be causing these large conditional errors. The QQ-plot of the pivoted errors, Figure 4.8 (b), indicates that the observed variability is bigger than the estimated, with many large values supporting the suggested non-stationarity problem.

According to the diagnostics, the emulator built with the training data is not a good and valid representation of the simulator. The diagnostics consistently point to the presence of non-stationarity and/or heteroscedasticity, with the model output for values of input 5 above 700 being shifted and more variable than when this input is below 700. Actions to improve the emulator might include adapting the mean function to the apparent shape of the response



(a) (b)

Figure 4.8: Graphical diagnostics for the Nilson-Kuusk model using the the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{y}^{(v)})$ against the pivoting order; (b) Quantile-quantile plot.

to input 5, allowing for a different variance when input 5 is above 700, or transforming the output variable to induce more homoscedasticity. Such changes, together with rebuilding the emulator using all 250 data points, should improve the fit substantially, but this should be checked against new validation data. However, we have no access to the simulator code, so such an analysis is not feasible

Instead, 50 validation data points randomly chosen were added to the training data. The points were randomly chosen because, if we run the emulator adding extra points in the area where the input space is between 650 and 800, it would have very few data points in that region for validation. The other 50 validation data points were used for the diagnostics. Attempting to improve the emulator, we change the mean function allowing a 4th-order polynomial for input 5, i.e. $h(\mathbf{x})^T = (1, \mathbf{x}^T, x_5^2, x_5^3, x_5^4)$. To stabilize the variability over the input space, a log transformation of the output was used. Alternatively, more complex analyses could be used since the diagnostics suggested the presence of non-stationarity in the input space. For instance, a non-stationary Treed Gaussian process model can be used (Gramacy and Lee 2008). Another approach would be using a linear spline with knots at 700 and 775 or 800 for input 5 as the mean function instead of a 4th-order polynomial, but for this example we have opted for the polynomial mean function because of its simplicity.

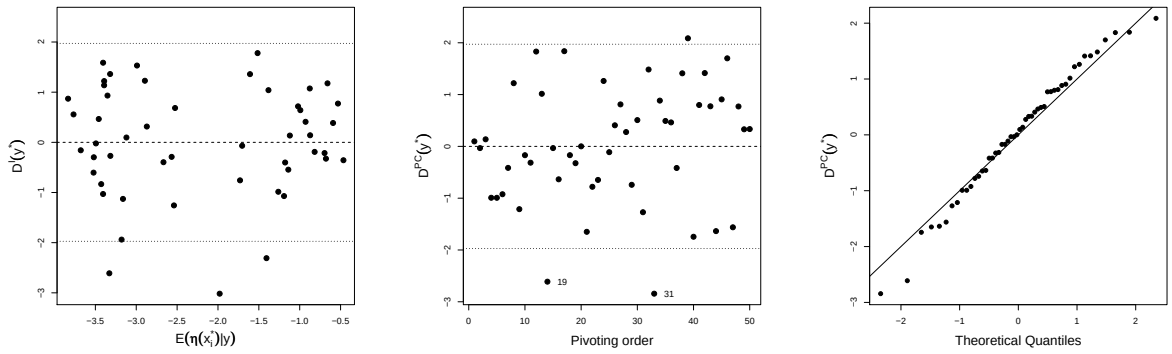
The Mahalanobis distance and credible interval diagnostics are presented in Table 4.4.

Both diagnostics suggest no conflict between the rebuilt emulator and the Nilson-Kuusk model.

Table 4.4: The observed Mahalanobis distance and credible interval diagnostics and some summaries of their predictive distributions of the updated emulator for the Nilson-Kuusk model.

	Obs.	Expected	Std. dev.	1stQ	Median	3rdQ
$D_{MD}(\cdot)$	63.873	50.000	11.305	43.007	49.968	58.320
$D_{CI}(\cdot)$	0.94	0.951	0.032	0.92	0.96	0.98

Figure 4.9 shows as expected that the rebuilt emulator can be a surrogate for the Nilson-Kuusk model. Figure 4.9 (a) presents the individual standardised errors against the emulator predictions, and Figure 4.9 (b) presents the pivoted Cholesky errors against the pivoting order. Both figures indicate no obvious pattern. Figure 4.9 (c) presents the QQ-plot of the pivoted Cholesky errors. As the points lie close to the 45-degree line, the normality assumption for the log transformed simulator outputs appears to be reasonable.



(a) (b) (c)

Figure 4.9: Graphical diagnostics of the updated emulator for the Nilson-Kuusk model: (a) Individual standardised errors, $D^I(\mathbf{y}^{(v)})$, against the emulator predictions; (b) Pivoted Cholesky errors, $D^{PC}(\mathbf{y}^{(v)})$, against the pivoting order; (c) Quantile-quantile plot of $D^{PC}(\mathbf{y}^{(v)})$.

We have seen in this example that the natural response to the diagnostics in the original emulator fit would have been to include all the validation data in the training sample, to add extra validation points with values of input 5 in the neighbourhood of 700, and then to validate the new emulator with extra validation points. This was not practical because

we did not have access to the simulator to make more runs, but we were instead able to produce a valid emulator with only 50 extra training data, randomly selected from the original validation sample. This result is encouraging because additional simulator runs can often be costly to obtain.

4.6 Concluding remarks

In this chapter, we have presented a set of diagnostics for validating Gaussian process emulators. We believe that this is a very important step before using the emulator as surrogate for the simulator, because a non-valid emulator can induce wrong conclusions. Our diagnostics focus on comparing emulator outputs with new runs of the computer model, referred to here as validation data, in a way that takes account of the uncertainties and correlations in the Gaussian process emulator predictions.

We proposed two kinds of diagnostics, numerical and graphical. The numerical diagnostics are functions of the validation data outputs, where we compare the observed value of each diagnostic with its induced distribution by the predictive distribution of the emulator outputs. The graphical diagnostics are visualisations of the prediction errors, where we consider the individual standardised errors and the uncorrelated standardised errors. The diagnostics are able to indicate whether the emulator and its uncertainty can represent the simulator. If the emulator fails, the diagnostics can give information about where the problem might be.

However, the interpretations we have suggested for the various diagnostics should be used with care, since in a complex system such as a Gaussian process emulator all the elements will interact in determining the diagnostic values. Better understanding of how to read different combinations of diagnostics will come with more experience of their use.

The diagnostics are equally effective when the training dataset is large. In this case, the mean function (2.17) will typically reproduce the simulator almost perfectly even for a non-stationary simulator, and the predictive variances will be very small throughout the region of input space covered by the training data. Even though the validation data may appear to

follow the emulator mean function very closely, the diagnostics may nevertheless find that they are not close enough relative to the small predictive variances, so that standardised errors are not acceptable.

In the case of a large training dataset, computational problems are often encountered due to near-singularity of the predictive variance matrix. The pivoted Cholesky decomposition is then particularly valuable because the ill-conditioned nature of the matrix will emerge with pivoting variances eventually becoming negative due to rounding errors. Such components should be ignored, and the diagnostics based only on the uncorrelated errors produced up to that point.

In practice, "all models are wrong"[1] and no emulator will represent its simulator with perfect validity. The emulator will still be useful, and 'good enough', if any remaining conflicts are small. In particular, an emulator built on a large training dataset (and hence with uniformly small predictive variances) may be deemed adequate even if there are substantive validation failures. It would be useful to have some measures for whether the emulator is 'good enough', but they are likely to depend on the uses to which the emulator will be put. We explore some measures of goodness of fit in Chapter 6.

[1]The full quotation is "Essentially, all models are wrong, but some are useful." from Box, George E. P.; Norman R. Draper (1987). *Empirical Model-Building and Response Surfaces.* Wiley. pp. p. 424.

Chapter 5

Designs for building and validating emulators

5.1 Introduction

In this chapter, we investigate how to select the emulator training and validation inputs. In order to build an emulator, we want a design that makes the emulator predict accurately the simulator output at non observed inputs. To validate an emulator we want a design that efficiently distinguishes between good and bad emulators. The judgement about the emulator's validity is based on the diagnostics presented in Chapter 4.

Since the behaviour of the simulator is unknown, designs that provide information about the simulator output throughout the input space are needed. Predictions for the simulator are made using emulators that represent our probability judgements about simulators. Design and prediction for simulators were first addressed in the literature by Sacks et al. (1989a,b). In Section 5.2, we describe designs that we use to build emulators. Our focus is on designs that cover the input space, which we call a space-filling property. We review Latin hypercube sampling, distance-based designs and non-random designs.

An emulator can replace a simulator in analyses where runs for the simulator are com-

putationally expensive. However, the emulator needs to be subjected to validation testing. Otherwise, inferences made using the emulator will be invalid. In order to validate an emulator, new simulator runs are needed where the simulator and the emulator outputs are compared. The design for validation problem is to select inputs minimizing the possibility of making wrong conclusions, i.e. judging a 'bad' emulator to be valid, or a good emulator to be not valid. In Section 5.3, we propose some designs for validating Gaussian process emulators. In Section 5.4, we use a simulation study to test the performance of the proposed validation designs.

5.2 Designs for building Gaussian process emulators

In order to build an emulator, we need to run the simulator at different points in the input space. The process of choosing these input points is the design problem considered here. The simplest design uses uniform random sampling, where we sample from a uniform distribution over the input space. A problem with this method is that some regions of the input space may not be covered. One way to solve this problem is using stratified sampling, where the input space is partitioned into J disjoint strata, $\mathcal{X}_j$, and from each stratum a uniform random sample with size n_j is taken. Using the stratified sampling, all regions of the input space $\mathcal{X}$ are represented by the $\sum_j n_j$ input values.

The simulator output may only depend on few of the inputs, in which case we want to be sure that points are well spread across the space of these 'important' inputs. A design that spreads points evenly throughout the input space does not necessarily have this property. The Latin hypercube design is a design such that a projection of the set of points into each dimension has the points evenly spread over that dimension. In the next section, we review Latin hypercube sampling. We also review other designs that are evenly spread throughout the input space which are called distance-based designs. Finally, we review some non-random designs used in numerical integration.

5.2.1 Latin hypercube sampling

Latin hypercube designs are popular in the computer experiments literature (Santner et al. 2003). This popularity is attributed to two reasons. Firstly, Latin hypercube designs are simple to generate. Secondly, in a Latin hypercube sample, the observations in each input dimension are evenly spread.

McKay et al. (1979) proposed Latin hypercube sampling as an alternative to simple random sampling in a Monte Carlo study. Latin hypercube sampling ensures that, marginally, the sample space of each input is well covered. Stein (1987) shows that the asymptotic variance of the expectation of the simulator output using Latin hypercube sampling is less than that obtained using simple random sampling. Stein also presents a method for producing Latin hypercube samples when the inputs are dependent. Here, we assume that all input variables are independent.

The following procedure describes how to obtain a Latin hypercube design, $\mathbf{X}$.

Let $\mathbf{B}$ be an $n \times p$ matrix, where each column of $\mathbf{B}$ is an independent random permutation of $\{1, 2, \ldots, n\}$. Let U_{ij} $(i = 1, 2 \ldots, n,\ j = 1, 2 \ldots, p)$ be np independent uniformly distributed random variables on (0, 1) which are also independent of $\mathbf{B}$. Then the j-th element of the i-th input is given by

$$\mathbf{X}_{ij} = \frac{B_{ij} + U_{ij}}{n}.$$

Note that if the input space is $[a, b]^p$, a straightforward transformation can be used.

Even though a Latin hypercube design represents well every input marginally, it does not necessarily have good space-filling properties.

5.2.2 Distance-based designs

Distance-based designs are designs based on measures of distance between points that quantifies how evenly spread the points are. Let $\mathbf{X} \subset \mathcal{X}$ be an arbitrary design consisting of input points $\{x_1, x_2, \ldots, x_n\}$. Let *dist* be a metric on $\mathcal{X}$, for instance, the Euclidean distance.

For any design, the minimum distance between two points can be obtained. A design that maximizes this measure is called a *maximin* design. A maximin design is denoted by $\mathbf{X}_{Mm}$, and is formally given by

$$\mathbf{X}_{Mm} = \max_{\mathbf{X} \subset \mathcal{X}} \min_{\{x,x'\} \in \mathbf{X}} dist(x, x'), \tag{5.1}$$

where $dist(x, x')$ is the Euclidean distance between x and x'. Maximin designs tend to place points near the boundary of the input space region, which may not provide sufficient information about the middle of the input space.

Another distance-based design is a design that every point in the input space $\mathcal{X}$ is close to some point in $\mathbf{X}$. This design is called a *minimax* design. A minimax design is denoted by $\mathbf{X}_{mM}$, and is formally given by

$$\mathbf{X}_{mM} = \min_{\mathbf{X} \subset \mathcal{X}} \max_{x \in \mathcal{X}} \min_{x' \in \mathbf{X}} dist(x, x'). \tag{5.2}$$

One problem with minimax designs is that they are difficult to generate. Consequently, they are rarely used.

Maximin Latin hypercube designs

Within the class of Latin hypercube designs, we can choose one that satisfies a distance-based criterion, for example the maximum minimal distance. Such a design is called a *maximin Latin hypercube* design. This design was examined by Morris and Mitchell (1997). Figure 5.1 (a) presents 50 points from a two dimensional maximin Latin hypercube design, where 10,000 Latin hypercube designs were generated and the design with the largest minimum distance is chosen.

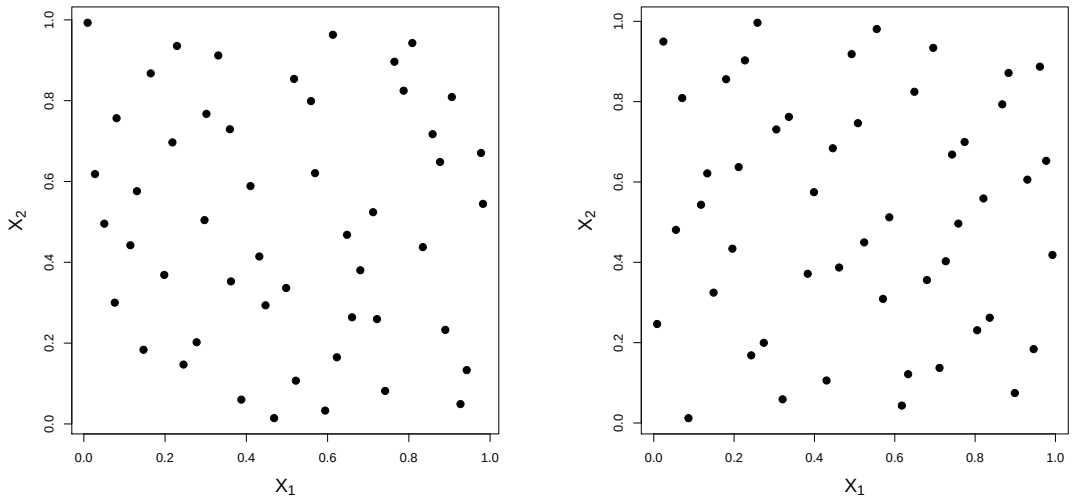
5.2.3 Non-random designs

Lattice designs. A lattice design is one of a number of non-random space-filling designs suitable for defining a set of points in the simulator input space for creating a training sample. The points are chosen on a regularly space grid superimposed on the input space. In computer experiments, lattice designs were considered by Bates et al. (1996, 1998).

A number of different sequences of numbers have been proposed that have space-filling properties. The sequences use different algorithms to generate them, but all have the property that they are potentially infinite in length, as a new sequence can be added to an old one. A design of points is obtained simply by taking the first points in the sequence. Examples are the Weyl sequence, the Halton sequence, and the Sobol' sequence (Neiderreiter 1992). A Weyl sequence is similar to a lattice design in the way it is generated, but with generators that are irrational numbers. A Halton sequence also has a prime integer "generator" for each dimension, and each prime generates a sequence of fractions.

Sobol' sequence. The Sobol' sequence uses the same set of coordinates as a Halton sequence with generator 2 for each dimension, but then reorders them according to a complicated rule. Galanti and Jung (1997) illustrate the Sobol' sequence including simple numerical examples. There is a function in R for producing Sobol' sequences. Sobol' sequences in R are produced with the function *runif.sobol* from the package 'fOptions'. Figure 5.1 (b) presents 50 points from a two dimensional Sobol' sequence design.

As advantage of Sobol' sequences is that a Sobol' sequence is cheaper to generate than other sequences. In contrast to Latin hypercube designs, longer Sobol sequences can be constructed from a shorter Sobol sequence by adding points to the shorter sequence. Latin hypercube designs must be reconstructed if more points are needed. Santner et al. (2003, page 160) suggests that if more information about the correlation lengths is needed, then, due to a greater variety of inter-point distances, designs based on Sobol' sequence may be preferable to Latin hypercube designs.



(a) (b)

Figure 5.1: Two-dimensional design of size 50 using: (a) Maximin Latin hypercube design, where 10000 Latin hypercubes were generated and the one with maximum minimal distance is chosen; (b) Sobol' sequence design.

5.3 Design for validating Gaussian process emulators

Now we consider how to select inputs to perform a validation analysis of a Gaussian process emulator. We want to choose a validation design $\mathbf{X}^{(\mathbf{v})}$ that minimizes the possibility of making wrong validation conclusions, i.e. validating a bad emulator or not validating a good emulator. The validation analysis of an emulator is based on numerical and graphical diagnostics presented in Chapter 4.

If our uncertainty about the simulator output at untried inputs is well represented by an emulator, we say that this emulator is valid. If we run the simulator in regions of the input space where there is significant information, i.e. we are fairly certain about a value for the simulator output, then a valid emulator should provide a probability distribution for the output concentrated on the actual output value. On the other hand, if we run the simulator in regions of the input space with weak information or no information, i.e. our uncertainty about the simulator output is high, the emulator should provide a flat probability distribution for the output. Therefore, a validation design should contain elements in different regions of emulation uncertainty varying from low to high. Then we can understand how the emulator

behaves under different sources of information and check whether the emulator represents our uncertainty about the simulator outputs.

The designs described in Section 5.2 can be used as validation designs. However, it is not necessarily guaranteed that regions with small and large uncertainty are covered, because the training data are not used to select the validation design. For instance, if we use a Sobol' sequence design as validation design, by definition the validation inputs would fill the gaps among the training inputs. Therefore, validation points in small uncertainty regions may not be observed. An independent design such as an independent Latin hypercube design may have points in regions of with different levels of emulator uncertainty.

We are interested in checking how the emulator behaves under different uncertainty conditions. Therefore, we propose methods for choosing the validation inputs under two situations. In the first situation, we choose validation inputs without using the simulator outputs at the training inputs. We consider a criterion based on distance between the validation inputs and the training inputs. In the second situation, we propose design criteria where the validation inputs are chosen conditioned on the training data. We use some criteria based on the predictive variance of the emulator to generate the validation designs.

5.3.1 Distance-based validation design

Emulation uncertainty at a validation input will depend on the distance of the validation input from the nearest training input. If the nearest training input is close to a validation input, then we expect to have low uncertainty for the output at the validation input. Analogously, if the nearest training input is far from a validation input, then high uncertainty is expected. Therefore, let $dist_{\mathbf{X}}(\mathbf{x}^{(v)})$ be the minimum distance between a validation input point $\mathbf{x}^{(v)}$ and the training points $\mathbf{X}$, i.e.

$$dist_{\mathbf{X}}(\mathbf{x}^{(v)}) = \min_{x \in \mathbf{X}} dist(x, \mathbf{x}^{(v)}).$$

To choose a design that contains validation points with different levels of uncertainty,

we should choose a design that maximises the variability of the inter-point distances of the validation inputs and the training inputs. Therefore, our proposed criterion is the variance of the inter-point distances between the validation inputs and the training inputs, i.e.

$$\Upsilon_{VarDist}(\mathbf{X}^{(v)}) = Var\left(dist_{\mathbf{X}}(\mathbf{x}_1^{(v)}), \ldots, dist_{\mathbf{X}}(\mathbf{x}_m^{(v)})\right). \tag{5.3}$$

We are interested in a design that maximises this criterion, because we can have validation input points with minimum distance to training data points varying from small to large, providing input points with emulator uncertainty varying from low and large. The proposed validation design is

$$\mathbf{X}_{VarDist}^{(v)} = \max_{\mathbf{X}^{(v)} \subset \mathcal{X}} \Upsilon_{VarDist}(\mathbf{X}^{(v)}). \tag{5.4}$$

The optimisation method to solve (5.4) requires high-dimensional optimisation algorithms increasing considerably the computational cost. Alternatively, we could choose distance-based designs for validation in a class of Latin hypercube designs.

In practice, we generate a large number of Latin hypercube designs, which is computationally cheap, and select the design the optimises the criterion (5.3). This procedure gives an approximation for the optimal validation design in the class of the Latin hypercube designs. Hence, a **distance-based Latin hypercube design for validation** is generated after sampling many Latin hypercube designs and choosing the design with largest value of the criterion (5.3). Notice that this procedure may not arbitrarily get validation points close to training points.

5.3.2 Combined design for validation

The distance-based validation design selects points in different uncertainty regions according to the optimisation method in (5.4). As an alternative, we propose a two-part sampling procedure. In the first part, the data is chosen according to any independent design described in Section 5.2. In the second part, the input points are randomly chosen from a low emulation uncertainty region. These low emulation uncertainty regions are defined by regions nearby

the training input points. Such a design is called a **combined validation design**.

Let m be the sample size for the validation data. According to this combined design idea, m_1 validation points are chosen for the first part of the sampling procedure, and $m_2 = m - m_1$ validation points are chosen in a such way that the correlation between each validation point and the closest training data point is high. The combined design for validation of size m is defined by

$$\mathbf{X}_C^{(v)} = \begin{pmatrix} \mathbf{X}_{C_1}^{(v)} \\ \mathbf{X}_{C_2}^{(v)} \end{pmatrix},$$

where the design $\mathbf{X}_{C_1}^{(v)}$ are the inputs from an independent design, and $\mathbf{X}_{C_2}^{(v)}$ are the inputs from a region with points highly correlated with some training data.

For sampling $\mathbf{X}_{C_1}^{(v)}$, we can use any space-filling design described in section 5.2. However, we must guarantee that all validation inputs are different from the training inputs, otherwise it would be a waste of effort since the simulator is a deterministic function. An independent design for validation can have input points in regions with uncertainty varying from low to high. But, the training data size is generally small, since the simulator is generally computationally expensive. Therefore, it is likely that the validation inputs from a independent design lie in regions with medium to high uncertainty.

To sample $\mathbf{X}_{C_2}^{(v)}$, we only use the training inputs and not the outputs. We need to fix a maximum distance, d_0, considered close in the input space. This is to guarantee that the validation inputs are chosen from regions with low uncertainty. Centred on a training input, $\mathbf{x}$, a low uncertainty region would be the ellipsoid

$$R_{\mathbf{x}} = \left\{ (z_1, \ldots, z_p) \in \mathcal{X} : \sum_k (z_k - \mathbf{x}_k)^2 < d_0 \right\}. \tag{5.5}$$

The validation input points, $\mathbf{X}_{C_2}^{(v)}$, are chosen using the following strategy:

1. Fix a maximum distance, d_0, considered close;

2. Randomly select one training data point, $\mathbf{x}$;

3. Define the ellipsoid region $R_{\mathbf{x}}$, equation (5.5);

4. Randomly choose a validation point from a uniform distribution in region $R_{\mathbf{x}}$;

5. Repeat 2-4 to generate m_2 validation points.

It is not simple to fix a distance considered close in the input space, particularly in high dimensions. As an alternative, we could set a value, say ρ, for the minimum correlation between two points in the input space still considered high. The strategy for choosing the validation points $\mathbf{X}_{C_2}^{(v)}$ remains the same, but the ellipsoid region is now given by

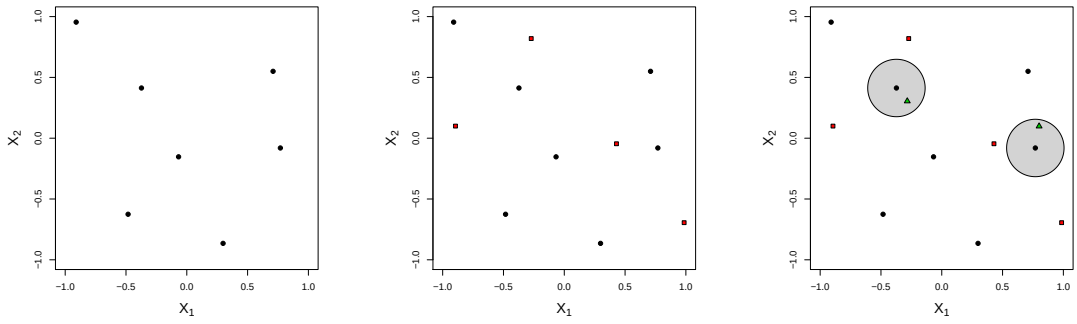
$$R_{\mathbf{x}} = \{\mathbf{z} = (z_1, \ldots, z_p) \in \mathcal{X} : C_\delta(\mathbf{z}, \mathbf{x}) > \rho\}, \tag{5.6}$$

where $C_\delta(\cdot, \cdot)$ is a correlation function with correlation parameter δ. It is necessary to choose a correlation function and also fix a value for the correlation parameter δ. If the training outputs are available, δ can be estimated from its posterior distribution (2.19). Note that the ellipsoid region (5.5) is just a particular case of the ellipsoid region (5.6).

Figure 5.2 illustrates the proposed combined design for validation in a two-dimension example. In Figure 5.2 (a), $n = 7$ training inputs were selected from a Latin hypercube design. In Figure 5.2 (b), $m_1 = 4$ validation inputs, $\mathbf{X}_{C_1}^{(v)}$, were chosen from an independent Latin hypercube design. Finally, in Figure 5.2 (c), $m_2 = 2$ training inputs were randomly chosen. For each chosen input a region $R_{\mathbf{x}}$, equation (5.6), is defined. The validation points are randomly sampled from a uniform distribution in each region. The correlation lengths used are $\psi_1 = \psi_2 = 0.5$, and the minimum correlation considered high was $\rho = 0.80$.

5.3.3 Predictive variance-based validation design

The distance-based and the combined validation designs do not require simulator runs at the training inputs, except for the combined validation design when we want to estimate the correlation parameters from the data. But, if we have access to the training data before



(a) (b) (c)

Figure 5.2: Illustrating the combined design for validation in a two-dimensional case. (a) Training inputs; (b) choosing $\mathbf{X}_{C_1}^{(v)}$; (c) choosing $\mathbf{X}_{C_2}^{(v)}$.

choosing the validation inputs, we can then build an emulator and sample the validation inputs using the uncertainty region defined by the emulator.

Conditional on the training data and an estimate for the correlation lengths, a Student-t process emulator is given by equation (2.16), with predictive mean, $m_1(\cdot)$, and predictive variance matrix, $V_1(\cdot,\cdot)$, given by (2.17) and (2.18), respectively. Let $V_1(\mathbf{X^{(v)}}) = \{V_1(x_i^{(v)}, x_j^{(v)})\}$ be the predictive variance matrix for a validation set of inputs, $\mathbf{X^{(v)}}$.

In order to choose designs covering regions with different levels of uncertainty, we present designs based on some criteria applied to the predictive variance matrix $V_1(\mathbf{X^{(v)}})$. We want a criterion that chooses input points from regions with high and low uncertainty. In simple terms, we want input points where the predictive variance is large and also inputs points where the predictive variance is small, taking into account the correlation structure of the emulator.

Our first criterion is based on the eigenvalues of the predictive variance matrix. The proposed design is based on the variance of the eigenvalues, i.e.

$$\Upsilon_{VarEig}(\mathbf{X}^{(v)}) = Var\left(\lambda_1(V_1(\mathbf{X}^{(v)})), \ldots, \lambda_m(V_1(\mathbf{X}^{(v)}))\right), \tag{5.7}$$

where $\lambda_i(A)$ is the i-th eigenvalue of matrix A. A large eigenvalue is associated with a linear combination of the observed inputs that has a large variance, whereas a small eigenvalue is

associated with a linear combination that has a small variance. A design that maximizes this criterion is more likely to have validation input points in different levels of emulator uncertainty, because a large variance of the eigenvalues indicates the presence of large and small eigenvalues. Therefore, the design for validation based on the criterion (5.7) is given by

$$\mathbf{X}^{(v)}_{VarEig} = \max_{\mathbf{X}^{(v)} \subset \mathcal{X}} \Upsilon_{VarEig}(\mathbf{X}^{(v)}). \tag{5.8}$$

Alternatively, we propose a criterion based on the pivoted Cholesky decomposition of the predictive variance matrix. The criterion is the variance of the predictive conditional variances given by the diagonal of the pivoted Cholesky decomposition matrix:

$$\Upsilon_{VarChol}(\mathbf{X}^{(v)}) = Var\left(R_1(V_1(\mathbf{X}^{(v)})), \ldots, R_m(V_1(\mathbf{X}^{(v)}))\right), \tag{5.9}$$

where $R_i(A)$ is the i-element of diagonal matrix of the pivoted Cholesky decomposition of A. $R_i(V_1(\mathbf{X}^{(v)}))$ is the predictive conditional variance for the observation i. An observation with a large (small) predictive conditional variance is associated with a large (small) emulator uncertainty. Therefore, the design for validation based on the criterion (5.9) is given by

$$\mathbf{X}^{(v)}_{VarChol} = \max_{\mathbf{X}^{(v)} \subset \mathcal{X}} \Upsilon_{VarChol}(\mathbf{X}^{(v)}). \tag{5.10}$$

The optimization methods required on (5.8) and (5.10) can be very computationally demanding, particularly in high-dimensional problems. So, we propose to choose predictive variance-based validation designs in a class of Latin hypercube designs.

5.4 Simulation examples

In this section, we present some examples where we generate the presented validation designs and run some diagnostics. The simulator examples are realisations of Gaussian processes where we can control the associated parameters. We show the effect on the diagnostics for

different validation designs in different scenarios. We vary the validation sample size and the simulator dimension, and use different estimates of the correlation lengths.

For the training data, we use the $n = 10p$ rule of thumb for the number of inputs points to be chosen (Chapman et al. 1994). This rule of thumb is considered a good rule for initial experiments by Loeppky et al. (2009). Conditional on the training data, we generate six different designs for validation. The first design is an independent Latin hypercube design referred to as **Ind**. The second design is a Latin hypercube distance-based validation design, **VarDist**, that depends only on the training inputs. The next two designs depend on the training data and its emulator. They are Latin hypercube predictive variance-based validation designs maximising the criteria (5.7) and (5.9), respectively referred to as **VarEig** and **VarChol**. A combined validation design using the squared exponential correlation function and the plug-in estimates for the correlation lengths is also generated, **Comb**. The last design is a Sobol' sequence design generated after the training data, here referred to as **Sobol**.

In order to generate the Latin hypercube designs for validation (**VarDist**, **VarEig**, and **VarChol**), we sampled 500 Latin hypercube designs and chose those designs that maximise the criteria (5.3), (5.7) and (5.9) respectively.

Example 5.1 (Gaussian process with linear mean) ***Simulator:*** *The simulator is a random realisation of a p-dimensional Gaussian process with linear mean, constant variance and squared exponential correlation functions. The mean, $m(\cdot)$, and covariance functions, $V(\cdot,\cdot)$, are given by*

$$\begin{aligned} m(\mathbf{x}) &= \sum_{k=1}^{p}(-1)^{k+1}x_k, \quad \mathbf{x} \in (0,1)^p, \\ V(\mathbf{x},\mathbf{x}') &= 0.5\exp\left\{-\sum_{k=1}^{p}(x_i - x_i')^2/\delta_i^*\right\}, \end{aligned}$$

where $\delta^* = (\delta_1^* = 0.5, \ldots, \delta_p^* = 3.0)$ is a vector of size p with $\delta_i^* = 0.5 + 2.5\frac{(i-1)}{(p-1)}$ for $i = 1, 2, \ldots, p$.

The training inputs are generated from a p-dimensional Sobol' sequence using the $n = 10p$ rule, and the outputs are realisations of the generated function from the Gaussian process.

Emulator: *The emulator is a Student-t process emulator, (2.16), with a linear mean prior, squared exponential correlation function and correlation lengths given by the true value δ^*.*

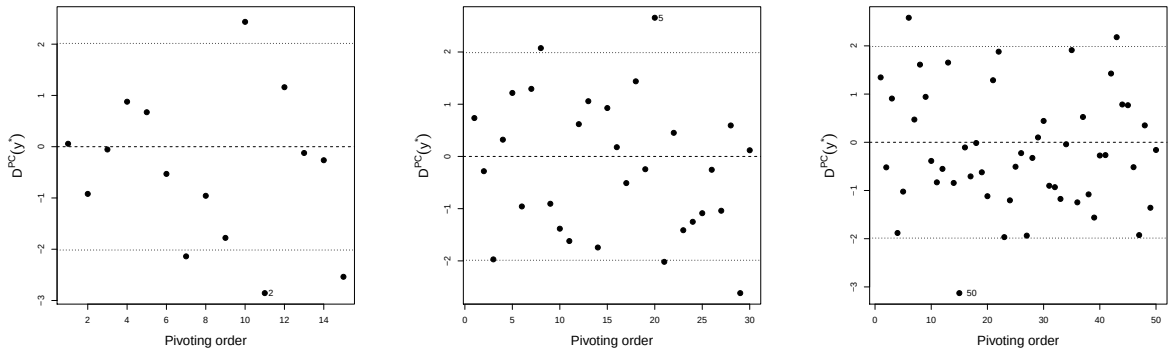
We generate the validation designs for the following simulator dimensions: $p = \{3, 4, 5, 10\}$. For each value of p, we also vary the validation data size: $m = \{3p, 5p, 8p\}$. Since the simulator is generally an expensive model, we choose these values representing respectively a small validation data size, a validation data size equal to half of the training data size, and a validation data size close to the training data size.

Table 5.1 presents the Mahalanobis distance for different validation designs, different values of the dimension, p, and different rules for choosing the validation sample size. In each row of the table, the expected value for the Mahalanobis distance is given by the validation sample size m, and the theoretical distribution is a scaled-F distribution (4.5). Since the simulator is a Gaussian process, the Student-t process is a valid emulator. For most of the scenarios, the Mahalanobis distance correctly identifies a valid model.

p	n	m	**Ind**	**VarDist**	**VarEig**	**VarChol**	**Comb**	**Sobol**
3	30	$3p = 9$	5.547	6.9111	6.9462	6.7767	7.6492	6.6811
		$5p = 15$	4.4368	17.6107	16.3944	18.9399	10.8218	14.2104
		$8p = 24$	17.4516	9.0955	18.5808	16.0421	14.6797	9.7183
4	40	$3p = 12$	5.9192	13.5215	13.8821	21.1716	17.0041	17.4977
		$5p = 20$	14.8215	18.3512	8.7402	20.4286	19.6991	26.5775
		$8p = 32$	26.2084	27.7727	31.6892	21.6513	31.7014	19.6572
5	50	$3p = 15$	32.9686*	21.4718	10.7046	10.022	12.1325	30.2155
		$5p = 25$	30.7485	29.9447	30.5001	37.1104	21.6913	36.3786
		$8p = 40$	38.3397	59.7747	57.9237	32.5198	35.9539	50.7044
10	100	$3p = 30$	25.9753	48.1343	50.5976*	27.5991	31.0009	39.612
		$5p = 50$	40.2536	34.5041	52.5942	50.6586	66.1407	75.5288*
		$8p = 80$	71.6175	84.2976	66.6316	84.0993	85.1438	97.6646

Table 5.1: Mahalanobis distance of the emulator predictions of example 5.1 for different validation designs, simulator dimension and validation data size. Values marked with '$*$' are those that are outside the 95% credible interval.

For the scenarios in Table 5.1 where valid emulators were not identified by the Mahalanobis distance, the observed values of the Mahalanobis distance are near the limit of their credible intervals. Figure 5.3 presents the pivoted Cholesky errors against the pivoting order for these emulators. The graphical diagnostics suggest that there is no evidence against the assumption of valid emulators.



(a) **Ind**, $m = 15$ (b) **VarEig**, $m = 30$ (c) **Sobol**, $m = 50$

Figure 5.3: Pivoted Cholesky errors for the emulators with large Mahalanobis distance in Table 5.1.

Large uncorrelated errors can have a strong impact on the Mahalanobis distance, as we observed in Figure 5.3. Hence, we should be careful when we have small validation data sizes. To illustrate this, we use the following example:

Example 5.2 (Gaussian process with linear mean) ***Simulator:*** *The simulator is a random realisation of a p-dimensional Gaussian process with linear mean, constant variance and squared exponential correlation functions. The mean, $m(\cdot)$, and covariance functions, $V(\cdot,\cdot)$ are given by*

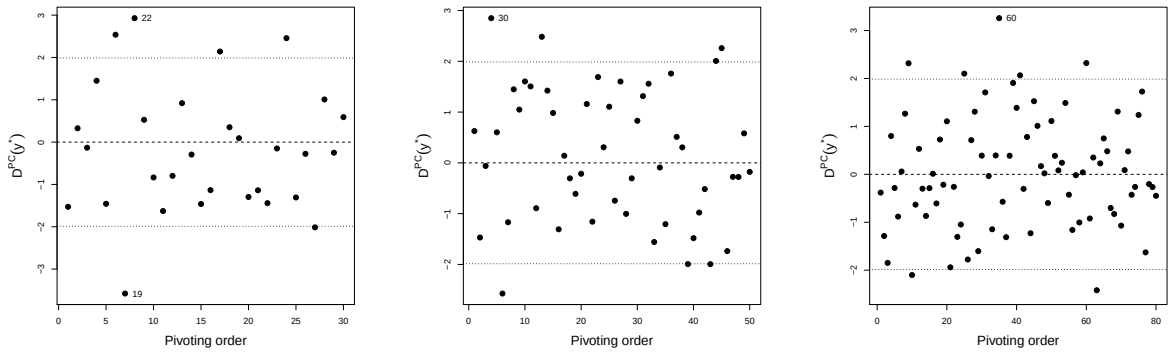
$$\begin{aligned} m(\mathbf{x}) &= \sum_{k=1}^{p}(-1)^{k+1}x_k, \quad \mathbf{x} \in (0,1)^p, \\ V(\mathbf{x},\mathbf{x}') &= 0.5\exp\left\{-\sum_{k=1}^{p}(x_i - x_i')^2/\delta_i^*\right\}, \end{aligned}$$

where $\delta^* = (\delta_1^* = 0.5, \ldots, \delta_p^* = 3.0)$, $\delta_i^* = 0.5 + 2.5\frac{(i-1)}{(p-1)}$ for $i = 1, 2, \ldots, p$.

The training inputs are $10p$ *points generated from a* p*-dimensional Sobol' sequence, and the outputs are realisations of the generated function from the Gaussian process.*

Emulator: *The emulator is a Student-t process emulator, (2.16), with a linear mean prior, squared exponential correlation function and correlation lengths given by the maximum likelihood estimates for correlation lengths.*

A random function was generated from a 10-dimension Gaussian process, 100 training inputs were chosen and the simulator outputs are evaluated as described in example 5.2. Combined validation designs with size $m = \{30, 50, 80\}$ were generated, the outputs were observed from the simulator, and the emulator was evaluated at each validation design. The observed Mahalanobis distances, with 95% credible intervals were 66.22 (16.92, 55.50), 86.54 (30.33, 81.54), 105.2751 (51.00, 120.51), respectively. For the first two validation samples the emulator would be considered non valid. Figures 5.4 a-c show the pivoted Cholesky errors for each validation dataset. The graphical diagnostics suggest that there is no obvious pattern in the uncorrelated errors, and therefore all three emulators can be considered as valid emulators. Hence, the large observed Mahalanobis distances have been influenced by few uncorrelated errors. The effect of the large uncorrelated errors on the Mahalanobis distance is stronger when the validation sample size is smaller.



(a) $m = 30$ (b) $m = 50$ (c) $m = 80$

Figure 5.4: Pivoted Cholesky errors for the emulators of the simulator given in example 5.2 with $p = 10$. using the combined validation design.

The Student-t process depends on the plug-in estimate for the correlation lengths. In our

next example, we vary the emulator correlation lengths to test the effect on the diagnostics of different validation designs.

Example 5.3 (Gaussian process with linear mean) ***Simulator:*** *The simulator is a random realisation of a p-dimensional Gaussian process with linear mean, constant variance and squared exponential correlation functions. The mean, $m(\cdot)$, and covariance functions, $V(\cdot,\cdot)$, are given by*

$$\begin{aligned} m(\mathbf{x}) &= \sum_{k=1}^{5}(-1)^{k+1}x_k, \quad \mathbf{x}\in(0,1)^5, \\ V(\mathbf{x},\mathbf{x}') &= 0.5\exp\left\{-\sum_{k=1}^{5}(x_i-x_i')^2/\delta_i^*\right\} \end{aligned}$$

where $\delta^* = (0.500, 1.125, 1.750, 2.375, 3.000)$. The training inputs are 50 points generated from a 5-dimensional Sobol' sequence, and the outputs are realisations of the generated function from the Gaussian process.

Emulator: *The emulator is a Student-t process emulator, (2.16), with a linear mean prior and squared exponential correlation function.*

We build an emulator using the true value δ^*, the maximum likelihood estimate $\hat{\delta}$, $\delta_i = 0, \forall i$, and $\delta_i = 6, \forall i$. For the last two correlation lengths, we have an emulator that assumes independence between any two outputs and an overconfident emulator. For each different plug-in estimate, we select six validation designs with size $m = 25$, evaluate the emulator and the simulator and compare the results.

Tables 5.2 and 5.3 present the observed Mahalanobis distances and the mean squared error. The expected value for the theoretical Mahalanobis distance is 25 and the 95% credible interval is (12.18, 50.12). Using the true value, the emulators applied in the validation designs were correctly identified as valid emulators. The same happens with the emulators using the maximum likelihood estimates, suggesting that the correlation lengths were well estimated. The independent emulator was also judged as a valid emulator. They are valid in the sense that the probability distribution derived from the emulator represents well our uncertainty

about the simulator output. Note that the accuracy for the predictive mean is very poor when we compare the mean squared errors of the emulators using the true δ and the independent emulators. Finally, when the correlation length is too high, the Mahalanobis distances are very high indicating non valid emulators. This is reasonable because when the correlation is too high it leads to overconfident emulators.

δ	**Ind**	**VarDist**	**VarEig**	**VarChol**	**Comb**	**Sobol**
δ^*	30.75	29.94	30.50	37.11	21.69	36.38
$\hat{\delta}$	43.66	48.20	44.05	39.89	32.22	23.27
$\{\delta_i = 0\}$	16.11	21.31	22.53	24.07	21.10	20.61
$\{\delta_i = 6\}$	1031.16	1385.12	1459.46	952.27	1061.02	1235.05

Table 5.2: Mahalanobis distance of the emulator predictions of example 5.3 with $p = 5$, validation data size $m = 25$ and different values of the correlation length.

δ	**Ind**	**VarDist**	**VarEig**	**VarChol**	**Comb**	**Sobol**
δ^*	0.0032	0.0075	0.0013	0.004	0.0009	0.0021
$\hat{\delta}$	0.0012	0.0025	0.0034	0.0022	0.0029	0.0014
$\{\delta_i = 0\}$	0.1844	0.2358	0.2698	0.2708	0.2494	0.23
$\{\delta_i = 6\}$	0.3047	0.6647	0.5226	0.3879	0.2792	0.8508

Table 5.3: Mean squared error of the emulator predictions of example 5.3 with $p = 5$, validation data size $m = 25$ and different values of the correlation length.

Example 5.4 (Gaussian process with cubic polynomial mean) *The simulator is a random realisation of a p-dimensional Gaussian process with cubic polynomial mean, constant variance and squared exponential correlation functions. The mean, $m(\cdot)$, and covariance functions, $V(\cdot,\cdot)$, are given by*

$$\begin{aligned} m(\mathbf{x}) &= \sum_{k=1}^{p}(-1)^{k+1}(x_k - x_k^3), \quad \mathbf{x} \in (0,1)^p, \\ V(\mathbf{x},\mathbf{x}') &= 0.5\exp\left\{-\sum_{k=1}^{p}(x_i - x_i')^2/\delta_i^*\right\}, \end{aligned}$$

where $\delta^* = (\delta_1^* = 0.5, \ldots, \delta_p^* = 3.0)$ is a vector of size p with $\delta_i^* = 0.5 + 2.5\frac{(i-1)}{(p-1)}$ for $i = 1, 2, \ldots, p$.

In this example, we generate the simulators from a Gaussian process with a cubic polynomial mean. Our emulator is Student-t process emulator with a linear mean prior and squared exponential correlation function. The training inputs are $10p$ generated from a p-dimensional Sobol' sequence with $p = \{3, 4, 5, 10\}$. We use the likelihood estimates for δ as the plug-in estimates in the emulator. Also for each value of p, we vary the validation data size $m = \{3p, 5p, 8p\}$.

Table 5.4 presents the Mahalanobis distance for different validation designs, different values of the dimension p, and different rules for choosing the validation sample size. In most of the scenarios, the observed Mahalanobis distance suggests a valid model. For those emulators for which the observed Mahalanobis distances were outside the 95% credible intervals, they are nevertheless all close to the upper limits of their credible intervals. For each extreme Mahalanobis distance, a graphical diagnostic check using the pivoted Cholesky errors was done. The graphical diagnostics suggest that there is no evidence against the assumption of valid emulators.

p	n	m	**Ind**	**VarDist**	**VarEig**	**VarChol**	**Comb**	**Sobol**
3	30	$3p = 9$	8.9327	8.0877	9.2482	7.7687	7.5084	8.4024
		$5p = 15$	11.7315	28.0005	23.4686	11.508	14.4801	22.8294
		$8p = 24$	40.6416	37.9715	28.5239	66.8603*	34.0021	42.1951
4	40	$3p = 12$	20.3136	28.5577	23.5867	19.8706	36.1634*	31.6188
		$5p = 20$	36.7652	34.7823	38.0688	29.994	32.467	32.0816
		$8p = 32$	61.6603*	34.0332	51.135	61.6753*	68.6478*	64.5197*
5	50	$3p = 15$	7.469	12.7537	11.7713	9.7113	9.753	27.7157
		$5p = 25$	26.3703	30.7192	21.8193	28.3528	27.8541	17.648
		$8p = 40$	34.3706	72.5109*	61.5465	48.788	50.0809	39.2298
10	100	$3p = 30$	46.0342	88.1531*	64.1226*	48.8914	40.8658	44.6645
		$5p = 50$	72.4895	59.9118	50.9156	59.0834	99.4419*	58.6468
		$8p = 80$	140.2513*	92.9186	112.9737	127.4352*	161.6534*	104.1254

Table 5.4: Mahalanobis distance of the emulator predictions of example 5.4 for different validation designs, simulator dimension and validation data size. Values marked with '$*$' are those that are outside the 95% credible interval.

5.4.1 Monte Carlo study

In the previous examples, we have randomness when the validation designs were chosen, except for the Sobol' sequence which is a non-random design. We also have some randomness when outputs of the simulator are generated; this is due to the simulator in the examples being a random function sampled from a Gaussian process. In order to take this randomness into account, we perform a Monte Carlo study.

The simulator is a 3-dimensional function generated from the Gaussian process of example 5.2. We generate 30 training data points, estimate the correlation lengths, and fit the Student-t process emulator. We repeat 250 times the process of generating all the previous six validation designs, evaluating the outputs and obtaining the Mahalanobis distance for each validation design. The sample size of each validation design is $m = 8p = 24$ points.

In order to verify the performance of the validation designs, we have done the study for different estimates for the correlation length. We use the true value $\delta^* = (0.50, 1.75, 3.00)$, the maximum likelihood estimate $\hat{\delta} = (0.5276, 1.2762, 2.6049)$, a deliberately reduced value for the correlation length $\delta = 0.1\delta^*$, and an inflated value for the correlation length $\delta = 1.5\delta^*$. The last two values for correlation lengths were included to show the effect of the designs on the diagnostics when the emulators are respectively under and overconfident.

If the assumptions for the emulator are correct, the distribution for the Mahalanobis distance is a scaled F distribution given in equation (4.5). Table 5.5 presents the proportion of observed Mahalanobis distances that lie in the 95% credible interval for the Monte Carlo simulation. For each correlation length scenario, the validation designs have similar proportions of valid models, except for the reduced correlation length where the combined design has a slightly lower proportion of valid models than other designs. We notice that the proportion of observed Mahalanobis distances inside the interval is very small for the inflated correlation lengths. When we reduce the correlation lengths, we have a high proportion of valid models.

Figure 5.5 presents the box-plots of the Mahalanobis distances with the theoretical 95%

δ	Ind	VarDist	VarEig	VarChol	Comb	Sobol
δ^*	0.940	0.936	0.948	0.932	0.936	0.952
$\hat{\delta}$	0.996	0.992	0.980	0.988	0.988	0.984
$0.10\delta^*$	0.996	1.000	0.988	0.988	0.936	1.000
$1.50\delta^*$	0.044	0.036	0.036	0.040	0.040	0.036

Table 5.5: Proportion of valid models in the Monte Carlo simulation of the 3D Gaussian process simulator for each validation design and different values for correlation length estimate.

credible interval. We notice that the observed Mahalanobis distances have a similar behaviour for all validation designs when we use the nominated true value, maximum likelihood estimate and the inflated value for the correlation lengths (Figures 5.5 a-c).

When we use the small value for the correlation lengths ($\delta = 0.10\delta^*$) the combined validation design and Sobol' sequence behave differently (Figure 5.5 d). This difference is associated with observations from regions of small uncertainty. For the Sobol' sequence there are no values very close to each other; this is due to the space filling properties of the Sobol' sequences. Therefore, there are no or very few observations where the emulator is overconfident. For the validation designs based on optimising Latin hypercube using various criteria, the proportion of observations where the emulator is overconfident is similar in each case and larger than the Sobol' sequence, and therefore the boxplots for the Mahalanobis distance present smaller values. For the combined validation design, a third of the observations are deliberately chosen from small uncertainty regions, it therefore presents more underconfident values and consequently smaller Mahalanobis distances.

In this simple Monte Carlo experiment, we conclude that all proposed validation designs provide points where the diagnostics will correctly provide evidence in favour of or against the validity of the emulator when our emulators are underconfident or represent well our uncertainty about the simulator outputs. This is because all designs are able to choose points from regions with medium and high uncertainty levels. However, when we have an overconfident emulator we need inputs from low uncertainty regions. The Sobol' sequence is a space-filling sequential design, and therefore there are no input points close to training inputs. The optimal Latin hypercube validation designs choose some points from small regions,

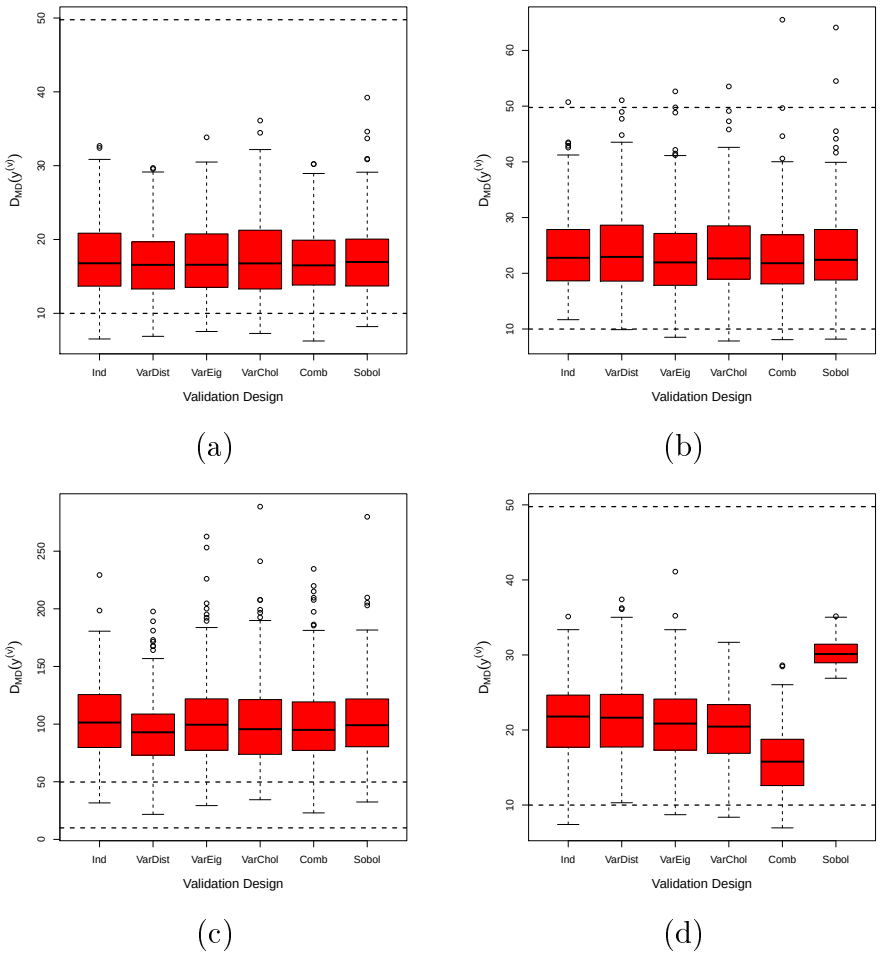


Figure 5.5: Box-plot of the observed Mahalanobis distances obtained in the Monte Carlo simulation of the 3-input Gaussian process simulator using the following correlation lengths: (a) the nominal true value δ^*; (b) the Maximum Likelihood estimate $\hat{\delta}$; (c) the inflated value $1.5\delta^*$; (d) the reduced valued $0.1\delta^*$.

but not many points. Also, the combined design, that deliberately chooses inputs close to training data, generate validation designs that provide information against the validity of the emulator.

5.5 Conclusions

The design for validation is an important issue that should be considered when the emulator needs to be validated. We have presented some designs for building and validating Gaussian

process emulators. Designs for building Gaussian process emulators are desired to have space-filling properties, and we reviewed distance-based designs, maximin Latin hypercube designs and some non-random designs. Designs for validating Gaussian process emulators should cover regions with different levels of uncertainty.

We propose validation designs that depend on the availability of the training data. If only the training inputs are available, then we should use distance-based validation designs or a combined design with a fixed value for the largest distance between points where the inputs are still considered close. If the training outputs are also available, we build the emulator using the training data and select the validation inputs from regions with different levels of uncertainty based on the predictive variance matrix. In our simulation study, the designs behave similarly to each other, but the combined validation design seems to be better at identifying underconfident emulators.

Chapter 6

Comparing competing emulators

6.1 Introduction

In this chapter, we present some methods for comparing competing emulators. Different model choices can lead to different emulators for the same simulator. A set of emulators for the same simulator are referred as competing emulators. The diagnostic methods presented in Chapter 4 identify valid emulators, but if we have two or more valid emulators, the proposed diagnostics are not designed to indicate the best emulator. Hence, it is necessary to consider some tools that allow us to compare competing emulators.

Gaussian process emulators can differ from each other for a number of reasons. For example, they can have different $h(\cdot)$ functions, different correlation functions $C_\delta(\cdot,\cdot)$, different ways to deal with the unknown parameters (plug-in or numerical integration), different designs for training data, etc.

The comparison methods are based on a new dataset where we compare the observed value of the simulator output with the predictions made by each emulator. The emulator with 'the best' predictions is considered the best emulator. However, there are several ways to measure how good the predictions of a particular emulator are. Here, we intend to describe some of these methods providing rules for comparing competing emulators.

One simple comparison criterion for two competing emulators is the mean squared error, which compares the predictions for a simulator, $\tilde{\eta}(\mathbf{x}^{(v)})$, with the observed value, $y^{(v)}$, as follows:

$$MSE(\mathbf{y}) = \frac{1}{m}\sum_{i=1}^{m}\left(y_i^{(v)} - \tilde{\eta}(\mathbf{x}_i^{(v)})\right)^2, \tag{6.1}$$

where the prediction function, $\tilde{\eta}(\mathbf{x}^{(v)})$, depends on the loss function adopted for predicting the simulator. If the squared error loss function is used, the prediction function, $\tilde{\eta}(\mathbf{x}^{(v)})$, is the predictive mean function of the emulator. The smaller the $MSE(\mathbf{y})$ the better the accuracy of the emulator.

The mean squared error only provides information about the emulator accuracy. It ignores the prediction uncertainty. A statistic that takes into account uncertainty in the emulator predictions is the Mahalanobis distance, which was presented in (4.4). The expected value for the Mahalanobis distance is the rank of the predictive covariance matrix, so emulators that present values for the Mahalanobis distance near the rank of the predictive covariance matrix are considered valid emulators. However, the observed Mahalanobis distance cannot be used to rank different emulators. One option for choosing the best emulator is to select from among the valid emulators the one with smaller mean squared error (6.1). But as we stated earlier, the mean squared error only takes into account the emulator's accuracy.

Therefore, we need statistics for ranking competing emulators that take into account the prediction accuracy and the prediction uncertainty. In Bayesian statistics, the natural way for comparing competing models is the Bayes factor, which we review in section 6.2. Alternative comparison methods known as proper scoring rules are reviewed in section 6.3. In section 6.4, we illustrate the comparison methods with some examples.

6.2 Bayes factors

The Bayes factor is a practical tool for comparing predictions made by two competing scientific theories. It was developed by Jeffreys (1935, 1961), and it is very popular in Bayesian

$2\log_e B_{12}$	B_{12}	Evidence against M_2
0-2	1-3	Not worth more than a bare mention
2-6	3-20	Positive
6-10	20-150	Strong
>10	>150	Very strong

Table 6.1: Reference table for Bayes factor comparisons proposed by Kass and Raftery (1995)

analysis. The Bayes factor is mainly used to compare competing models. Suppose we have two competing models, M_1 and M_2, with predictive density for data D given by $p(D|M_1)$ and $p(D|M_2)$ respectively. The Bayes factor for M_1 against M_2 is

$$B_{12} = \frac{p(D|M_1)}{p(D|M_2)}. \tag{6.2}$$

A large value for the Bayes factor suggest that M_1 is more appropriate than M_2. Kass and Raftery (1995) proposed a guideline, shown in Table 6.1 for interpreting the Bayes factor. Their table is a slight modification of the guideline proposed by Jeffreys (1961, app. B). Kass and Raftery considered the transformation of twice the natural logarithm of the Bayes factor because it is on the same scale as the likelihood ratio test statistic.

If a model contains unknown parameters, say θ, the predictive density is obtained by integrating over the parameter space, so

$$p(D|M_i) = \int p(D|M_i, \theta_i)\pi(\theta_i|M_i)d\theta_i,$$

where θ_i is a vector of parameters under M_i, $\pi(\theta_i|M_i)$ is its prior distribution, and $p(D|M_i, \theta_i)$ is the probability density conditional on θ_i or the likelihood of θ_i. For a review of Bayes factors see Kass and Raftery (1995) and references therein.

In the emulation context, for the most of the situations we use weak priors to represent the unknown parameters associated with the Gaussian process emulator. If we use improper priors then the Bayes factor would depend on non-specified constants and could not be evaluated. In order to deal with this problem, we could use some data to learn about the unknown parameters and another dataset to evaluate the Bayes factor. This method is

called **the partial Bayes factor** (Lempers 1971; Aitkin 1991). It is straightforward to use the partial Bayes factor to compare competing emulators. The emulators are built with some training data, $(\mathbf{y}, \mathbf{X})$, and then we use some testing data, or validation data, $\mathbf{X}^{(v)}$ to evaluate the predictive distribution of the simulator at the same testing inputs for all competing emulators. The Bayes factor for emulator E_1 against emulator E_2 is

$$B_{12} = \frac{p(\eta(\mathbf{X}^{(v)})|E_1, \mathbf{y}, \mathbf{X})}{p(\eta(\mathbf{X}^{(v)})|E_2, \mathbf{y}, \mathbf{X})}, \tag{6.3}$$

where E_i refers to the information associated with emulator i, $i = \{1, 2\}$. Analogously, if we have several emulators, we simply choose one as baseline, and compare the baseline emulator to every other emulator using table 6.1 as reference.

6.3 Scoring rules

In this section, we present alternative summary measures for evaluating emulators. Scoring rules assign a numerical score based on the predictive distribution and on the observed real process. They have been used in the field of expert elicitation to encourage the expert to make careful assessments and to be honest (O'Hagan 2006). Scoring rules measure the quality of probabilistic forecasts and rank competing models (Gneiting and Raftery 2007).

Let $s(F, y)$ be the score assigned for the stated predictive distribution F and the observed value y, where y has distribution G. We assume that a scoring rule is a penalty measure that the forecaster wishes to minimize. A scoring rule is proper if the expected value of $s(F, y)$ for an observation y drawn from G is maximized if $F = G$. It is strictly proper if the maximum is unique. See Gneiting et al. (2007) and Gneiting and Raftery (2007) for more information about strictly proper scoring rules.

6.3.1 Logarithmic score

The logarithmic score proposed by Good (1952) is the logarithm of the predictive density, f, evaluated at the observation y, and is a proper scoring rule. We have

$$\log S(F, y) = \log f(y), \tag{6.4}$$

with the larger the logarithmic score the better. If we take the difference between the logarithmic score of two different models at the same observations, we have the log Bayes factor of the first model against the second.

In the Gaussian process emulator (2.10) using plug-in estimates for the correlation lengths, the logarithmic score is $\log p(\eta(\mathbf{X}^{(v)})|\mathbf{y}, \mathbf{X}, \delta)$. This is either a multivariate normal density, or, in the case when (σ^2, β) are integrated out, a multivariate student-t density. In the case where the correlation lengths are integrated out numerically, it is necessary to obtain a numerical approximation for the joint density function. Banerjee et al. (2003, page 132) propose a method called composition sampling which consists of approximating the joint predictive density using a sample of the unknown parameters. The composition method of finding a numerical approximation of the joint predictive density is the following:

$$p(\eta(\mathbf{X}^{(v)})|\mathbf{y}, \mathbf{X}) \approx \frac{1}{M} \sum_{k=1}^{M} p(\eta(\mathbf{X}^{(v)})|\mathbf{y}, \mathbf{X}, \delta^{(k)}), \tag{6.5}$$

where, for instance, $p(\cdot|\mathbf{y}, \mathbf{X}, \delta)$ is the multivariate Student-t density, and $(\delta^{(1)}, \ldots, \delta^{(M)})$ is a sample from the posterior distribution of the correlation lengths. Conditional on a sample of the correlation lengths, the k-th sample from the predictive distribution of $\eta(\mathbf{X}^{(v)})$ is obtained by sampling from (2.16) conditional on the training data and $(\delta^{(k)})$.

6.3.2 Energy score

Gneiting and Raftery (2007) claim that the logarithmic score lacks robustness, so they propose the use of the **energy score**. The energy score, es, is a generalization of the continuous

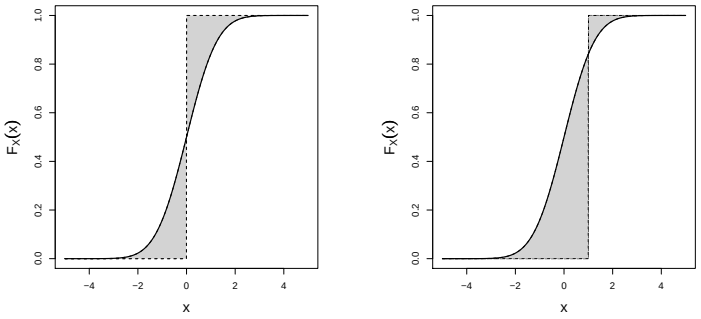
ranked probability score, *crps*, which compares, for each individual prediction, the empirical distribution and the theoretical distribution. If F is a predictive cumulative probability function and y is an observation from the process, then the continuous ranked probability score is defined as

$$crps(F, y) = -\int_{-\infty}^{\infty} (F(z) - \mathbb{I}(y \leq z))^2 \, dz. \tag{6.6}$$

If F has finite first moments, the *crps* can be rewritten as

$$crps(F, y) = \frac{1}{2} E|Z - Z'| - E|Z - y|, \tag{6.7}$$

where Z and Z' are independent copies of a random variable with distribution F and finite first moments. Figure 6.1 illustrates two different values of the *crps* when F is a standard normal distribution. The shaded areas illustrate the region over which we integrate to obtain the *crps*.



$crps(N(0,1), 0) = 0.2337$ $\quad$ $crps(N(0,1), 1) = 0.6024$

Figure 6.1: Illustrating the continuous ranked probability score of a standard normal distribution.

The energy score is a multivariate version of the continuous ranked probability score. The energy score is given by

$$es(F, \mathbf{y}) = \frac{1}{2} E||\mathbf{Z} - \mathbf{Z}'|| - E||\mathbf{Z} - \mathbf{y}|| \tag{6.8}$$

where $\mathbf{Z}$ and $\mathbf{Z}'$ are independent copies of a random vector with distribution F, and $||\cdot||$ denotes the Euclidean norm. Székely (2000) shows that the energy score is strictly proper

when the expectation $E||\mathbf{Z}||$ exists.

The energy score can be calculated using the following computationally efficient Monte Carlo approximation

$$es(F, \mathbf{y}) \approx \frac{1}{2(M-1)} \sum_{i=1}^{M-1} ||\mathbf{z}_i - \mathbf{z}_{i+1}|| - \frac{1}{M} \sum_{i=1}^{M} ||\mathbf{z}_i - \mathbf{y}|| \tag{6.9}$$

where $\mathbf{z}_1, \ldots, \mathbf{z}_M$ is a simple random sample of size M from the predictive distribution F. For emulators, the predictive density F is given by the emulator itself, and the observations of the process $\mathbf{y}$ are the simulator outputs at the testing inputs.

In order to compare several emulators, for each emulator we make predictions for the same testing inputs and calculate the Energy score using the testing outputs. The best emulator according to the energy score is the emulator with the largest observed energy score.

6.3.3 Dawid score

The Dawid score is a proper scoring rule that depends on the first and second moments only (Dawid 1998; Dawid and Sebastiani 1999). The Dawid score is proportional to a multivariate normal log density:

$$ds(F, \mathbf{y}) = -\log(det(\Sigma)) - (\mathbf{y} - \mu)^T \Sigma^{-1} (\mathbf{y} - \mu). \tag{6.10}$$

If the predictive process F is a multivariate normal distribution with mean μ and variance Σ, the Dawid score is equivalent to the logarithmic score. The difference between the Dawid scores of two different models can be seen as a numerical approximation to the log Bayes Factor. The Dawid score for an emulator is given by

$$DS = -\log(det(V[\mathbf{X}^{(v)}|\mathbf{y}])) - D_{MD}(\mathbf{y}^{(v)}), \tag{6.11}$$

where $D_{MD}(\cdot)$ is the Mahalanobis distance (4.4).

The Dawid score is useful in situations where the predictive density is not available, or perhaps too complex or numerically expensive to evaluate. For example, the Dawid score can be used to compare emulators using the Bayes linear approach (Goldstein and Wooff 2007).

6.4 Comparing emulators: examples

In this section, we consider how to compare competing emulators using the Bayes factor and the presented scoring rules. We compare the prediction performance of the competing emulators on a common dataset. This dataset should be a representative sample of the simulator throughout the input space. This is because an emulator could be better in a particular region of the input space, but worse in others. So, the predictive dataset should cover the input space. We use the methods described in Chapter 5 for choosing the dataset used for comparing competing emulators.

Comparing two different emulators using partial Bayes factor is described in equation (6.3), and the result can be interpreted according to table 6.1. The scoring rules are also used, the emulator with the largest score being considered the best according to the respective scoring rule.

Two-input examples

Two-input example 1

We compare the emulators built for the artificial model (Example 2.2) presented in section 2.3.2. The simulator is given by

$$\eta(x_1, x_2) = \left(1 - e^{-\frac{1}{2x_2}}\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right), \tag{6.12}$$

where $(x_1, x_2) \in (0,1)^2$.

An emulator was built using 20 training data points. A second emulator was updated using 30 more training data, with 50 training data points in total. Figures 4.5 (a) and (b) present respectively the training data and the predictive mean for both emulators.

To compare these two emulators, we need a new set of simulator runs. Using a Latin hypercube design, 30 new inputs were sampled (Figure 6.2) and their respective simulator outputs were obtained using equation (6.12).

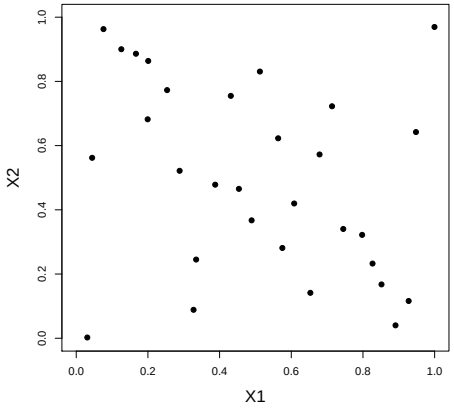


Figure 6.2: 30 inputs for comparison generated using a Latin hypercube design.

Table 6.2 presents some comparison statistics for each emulator. For the first emulator, using the 20 training data, the plug-in estimates for the correlation lengths are $(\hat{\delta}_1, \hat{\delta}_2) = (0.2421, 0.4240)$. The Mahalanobis distance suggests the emulator is non-valid since its observed value is outside a 95% credible interval (15.23, 56.83). On the other hand, the second emulator, using 50 training data, is considered valid as the observed D_{MD} is inside the 95% credible interval. The plug-in estimates for the correlation lengths are $(\hat{\delta}_1, \hat{\delta}_2) = (0.1763, 0.4116)$. The mean square error, as expected when we increase the training data, is smaller for the second emulator. The energy score, the logarithmic score and the Dawid score are larger for the second emulator, which also favours the second emulator. From the logarithmic score, we can calculate the log Bayes factor for the second emulator against the first emulator: $\log BF = 44.1766 - 7.2562 = 36.9204$. According to table 6.1, there is very strong evidence against the first emulator.

Model	MSE	$D_{MD}(\mathbf{y}^{(v)})$	ES	$logS$	DS
1st Emulator	1.7088	97.2659	-5.5661	7.2562	45.1661
2nd Emulator	0.0388	46.2913	-0.7514	44.1766	142.4725

Table 6.2: Comparison statistics for the two input example.

This simple example illustrates a situation where we expect a better performance from the emulator with more data, i.e., the second emulator. The comparison measures were consistent suggesting that the second emulator is the best.

Two-input example 2

Example 6.1 *Let the simulator be the following 2-dimensional function*

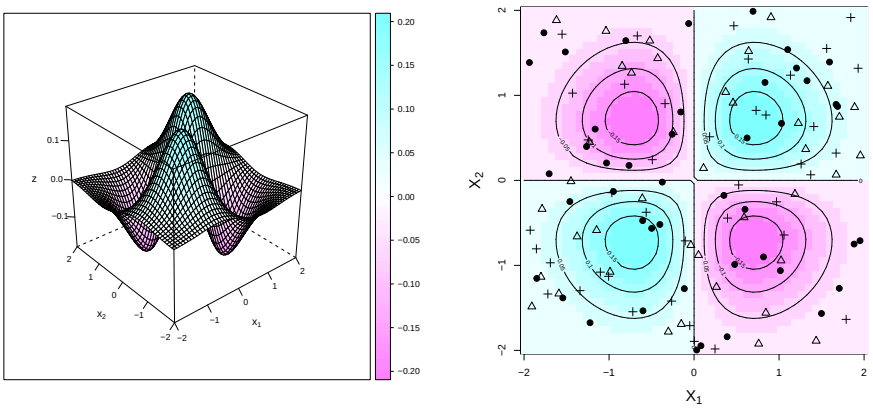
$$\eta(x_1, x_2) = x_1 x_2 e^{-x_1^2 - x_2^2}, \qquad (x_1, x_2) \in (-2, 2)^2. \tag{6.13}$$

In this example, we test different prior assumptions for the Gaussian process. We use different mean functions by changing the $h(\cdot)$, and we also change the estimates of the correlation lengths. The simulator is illustrated in Figure 6.3 (a), where we can see that the simulator is a smooth function.

We sampled 45 training data points using two independent Latin hypercube designs, with 20 and 25 observations respectively. We also sampled 40 testing data points using an independent Latin hypercube. Figure 6.3 (b) presents the contour plot of the simulator with training and testing data denoted by ($\bullet$) and ($\triangle$) respectively.

Our prior assumptions about the simulator are described by Gaussian process emulators with different mean functions $h(\cdot)$. The correlation function used is the squared exponential correlation function. Conditional on the training data and on estimates of the correlation lengths, the emulators are given by Student-t processes given by equation (2.16).

Table 6.3 presents the $h(\cdot)$ function used, the plug-in estimates for the correlation lengths, $\hat{\delta}$, the mean squared error and the Mahalanobis distance for each emulator built conditional



(a) (b)

Figure 6.3: Example 6.1: (a) Perspective plot of the simulator; (b) Contour plot of the simulator with the 45 training inputs (•), 40 testing inputs (△) and another 40 testing data (+) generated from Latin hypercube designs.

on the training data. The correlation lengths were estimated maximizing by the function (2.19) using its respective $h(\cdot)$ function. The estimates for the correlation lengths are very similar among the emulators, and the same behaviour is observed for the mean squared errors which are very small. The Mahalanobis distances are significantly small suggesting that all emulators are underconfident.

$h(x)$	$\hat{\delta}$	MSE	$D_{MD}(\mathbf{y}^{(v)})$
(1)	$\hat{\delta}_{(0)} = (1.0463, 1.0504)$	0.000006	1.2983
$(1, x))$	$\hat{\delta}_{(1)} = (1.0591, 1.0639)$	0.000006	1.4904
$(1, x, x^2)$	$\hat{\delta}_{(2)} = (1.0698, 1.0755)$	0.000007	1.6482
$(1, x, x^2, x^3)$	$\hat{\delta}_{(3)} = (1.0784, 1.0854)$	0.000007	1.7781

Table 6.3: Comparison statistics for example 6.1 for different values of the $h(\cdot)$ function.

Figure 6.4 shows the pivoted Cholesky errors using the different prior mean functions. The errors confirm that the emulators are similar to each other and underconfident. They all show smaller variability at the end of the pivoting order suggesting underestimation of the correlation lengths.

We now use a trial-and-error procedure to find a valid emulator for each $h(\cdot)$ function.

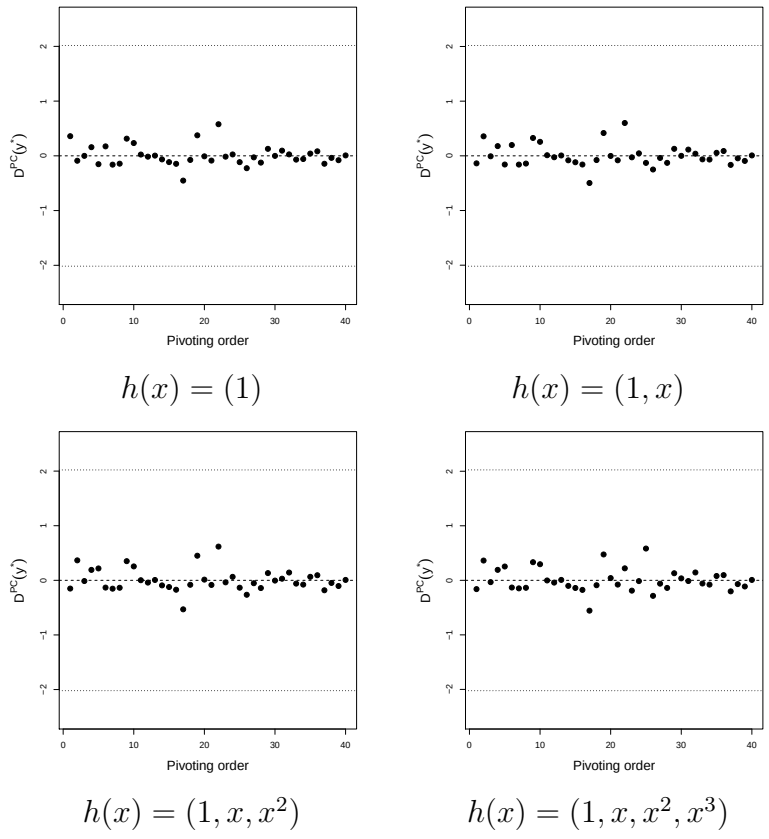


Figure 6.4: Example 6.1: (a) Pivoted Cholesky errors for emulators using different $h(\cdot)$ functions.

Since the Mahalanobis distance and the pivoted Cholesky errors suggested that the correlation lengths were underestimated, we multiply the estimated correlation lengths by a factor α, for $\alpha = \{1.00, 1.05, 1.10, \ldots, 1.50\}$ and calculate the Mahalanobis distance for each emulator. The results are presented in Table 6.4. The expected value for the Mahalanobis distance is 40, and its credible interval depends on $h(\cdot)$. For the four $h(\cdot)$ functions used, the emulator is valid when the respective estimated correlation lengths are increased by 25%. When we increase the estimated correlation lengths by more than 30% the emulators become overconfident. However, in order to validate the emulators properly we need new testing data, so we have a second testing dataset for this example given in Figure 6.3 (b) denoted by the symbol '+'. For the second testing dataset 40 inputs were generated using a Latin hypercube and the outputs were obtained running the simulator (6.1) at all testing inputs.

	$h(\cdot)$			
α	(1)	$(1, x)$	$(1, x, x^2)$	$(1, x, x^2, x^3)$
1.00	1.2983	1.4904	1.6482	1.7781
1.05	2.6425	3.0633	3.4107	3.6912
1.10	5.3735	6.2368	6.9404	7.4982
1.15	10.6137	12.2625	13.5667	14.5789
1.20	20.0081	22.9266	25.1245	26.8074
1.25	**35.7233**	**40.4923**	**43.9384**	**46.3692**
1.30	60.1442	67.1736	71.8689	76.3921
1.35	98.3387	107.3004	110.2110	115.1054
1.40	134.0551	161.8854	146.9015	140.2221
1.45	188.4756	212.9052	189.8107	216.3926
1.50	199.9750	260.9421	275.0082	281.7863

Table 6.4: Mahalanobis distance for example 6.1 for different values of the $h(\cdot)$ function and different correlation lengths. Highlighted values refer to the values near to the expected value 40.

Using the original training data, we build emulators using the four specifications of the $h(\cdot)$ function, and for each $h(\cdot)$ function, we use three different plug-in estimates for the correlation length. Firstly, we use the maximum likelihood estimate (MLE) for δ. Secondly, we use an inflated estimate suggested by Table 6.4, $1.25\hat{\delta}$. Finally, we use, as reference, an "independence "emulator which is an emulator for which any two ouputs are independent, i.e. $\delta = (0, 0)^T$.

Each emulator was used to predict the outputs for the second testing data. Table 6.5 presents the mean squared error, the Mahalanobis distance, the energy score, the logarithmic score and the Dawid score for each emulator using the second testing dataset. The emulators using the MLE for the correlation lengths result in smaller values for the mean squared error. However, they are all underconfident emulators. On the other hand, the independent emulators give considerably larger values for the mean squared errors while the Mahalanobis distance also suggest underconfident emulators. When we use the inflated correlation lengths, we confirm the analysis from Table 6.4 that suggests that the emulators using the inflated correlation length would be valid.

According to the scoring rules presented in Table 6.5, the emulators using the constant

$h(x)$	$\hat{\delta}$	MSE	$D_{MD}(\mathbf{y}^{(v)})$	es	$logS$	Ds
(1)	$\hat{\delta}_{(0)}$	0.000006	1.2983	-0.0181	228.1067	517.2064
	$1.25\hat{\delta}_{(0)}$	0.000032	35.7233	-0.0258	229.7439	533.5404
	(0,0)	0.005263	29.8460	-0.3269	47.1021	167.7014
$(1, x)$	$\hat{\delta}_{(1)}$	0.000006	1.4904	-0.0183	228.8452	518.3852
	$1.25\hat{\delta}_{(1)}$	0.000036	40.4923	-0.0267	226.6803	527.5471
	(0,0)	0.005352	29.0846	-0.3254	46.2119	165.8137
$(1, x, x^2)$	$\hat{\delta}_{(2)}$	0.000007	1.6482	-0.0185	229.0388	518.4204
	$1.25\hat{\delta}_{(2)}$	0.000039	43.9384	-0.0291	223.8017	521.7219
	(0,0)	0.005426	27.3132	-0.3318	45.6088	164.3032
$(1, x, x^2, x^3)$	$\hat{\delta}_{(3)}$	0.000007	1.7781	-0.0190	228.7165	517.3691
	$1.25\hat{\delta}_{(3)}$	0.000044	46.3692	-0.0302	221.2542	516.5003
	(0,0)	0.005470	26.0902	-0.3344	44.8420	162.5118

Table 6.5: Comparison statistics for example 6.1 for different values of the $h(\cdot)$ function and different estimates for the correlation lengths.

mean prior, $h(x) = (1)$, perform better than the others. Among the valid emulators, those whose Mahalanobis distances are near its expected value, 40, the best emulator is the one with prior mean given by a constant function, i.e. $m(x) = h(x)^T\beta = \beta_0$, and inflated correlation length. This emulator presents the overall largest value for the logarithmic score and the Dawid score, and the largest value for the energy score among the valid emulators.

6.5 Discussion

In this chapter, we have presented the Bayes factor and some scoring rules to compare competing emulators. We believe that it is necessary to provide comparison measures for emulators since the diagnostics for validation presented in the previous chapter do not rank models. The proposed comparison methods depend on new runs of the simulator, referred to here as testing data, where different emulators should make predictions at the same inputs to make them comparable. We suggest comparing valid models only, although the comparison methods do not require a valid model.

For comparing competing emulators, we suggest the use of the partial Bayes factor, as

we believe it is the best way to compare models in a Bayesian framework. However, in some situations it is not possible to evaluate the predictive density function of the emulator analytically. Therefore, we should use numerical approximations for the density function or try alternative comparison methods such as the energy score and the Dawid score. The energy score is numerically obtained when the samples from the emulator are available. The Dawid score depends only on the first two moments of the emulator, and is an alternative measure when the previous measures cannot be evaluated.

We have focused on the case in which the simulator gives a single output. It would be useful to extend the comparison methods to multiple-output emulators. The extension would be straightforward, since the emulator is still a Gaussian process and the predictive density will be known. However, further research is need before implementation.

Chapter 7

Analysis and diagnostics for discrepancy models

7.1 Introduction

In this Chapter, we consider a model to predict a real system, $\xi(\cdot)$, combining a fast simulator, $\eta(\cdot)$, with experimental data. Experimental data are observations of the real system that the simulator was design to represent. Such a model is a simplified version of the calibration model proposed by Kennedy and O'Hagan (2001). The difference between the real system and the simulator, i.e., $d(\cdot) = \xi(\cdot) - \eta(\cdot)$, is called the discrepancy function. A model that uses a fast simulator, fast in the sense that there is no need of emulation, and experimental data is called a **discrepancy function model**. This model is described in in Section 7.2.

In Section 7.3, we review some different methods for dealing with some unknown parameters in the discrepancy function model. The method most commonly used is the plug-in method, in which we obtain estimates for the parameters and condition on the parameter estimates as the true values. Other methods attempt to take into account the uncertainty associated with the unknown parameters. Nagy et al. (2007) propose a computationally cheap integration method for emulators that we adapt here. We also briefly describe the

MCMC and the Kennedy and O'Hagan approaches.

Our aim is to provide a set of diagnostics to validate a discrepancy model. In Section 7.4, we present some graphical and numerical diagnostics that provide some information about the validity of the model. The proposed diagnostics are extensions of the diagnostics proposed in Chapter 4.

7.2 Discrepancy function model

The aim is to learn about a real system, represented by $\xi(\cdot)$, which is simulated by $\eta(\cdot,\cdot)$. We consider two different types of inputs: the control inputs which are the same inputs as in the real system function, and the calibration inputs or parameters which are inputs for the simulator only. The calibration parameters can, for example, be physical constants associated with the real system, or numerical parameters associated with the numerical precision required. The real system is represented as the simulator plus an unknown function $d(\cdot)$ called the discrepancy function:

$$\xi(\cdot) = \eta(\cdot, \theta) + d(\cdot), \tag{7.1}$$

where the calibration parameters, denoted by θ, are unknown inputs for the simulator only. The domain for the real system function, ξ, is the input space, $\mathcal{X} \in \mathbb{R}^p$. Note that the discrepancy function, d, has the same domain as ξ, and the simulator domain depends on the control inputs $\mathbf{x} \in \mathcal{X}$ and the calibration parameters.

For a particular input $\mathbf{x} \in \mathcal{X}$, the observed real process includes measurement errors, or observational errors. For a given input $\mathbf{x}$, the observational error is added to (7.1). Therefore, observations of the real process $z_1, \ldots, z_n$ at location $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are subject to observational errors $e_1, \ldots, e_n$. Therefore, the model is given by

$$z_i = \xi(\mathbf{x}_i) + e_i = \eta(\mathbf{x}_i, \theta) + d(\mathbf{x}_i) + e_i, i = 1, 2, \ldots, n. \tag{7.2}$$

Our prior uncertainty about the real process ξ should incorporate our prior uncertainty about the associated processes and parameters. One important assumption is that the prior distributions for the discrepancy function, the observational errors and the simulator are independent. We assume that the simulator is fast to run, so there is no need to emulate the simulator. The calibration parameters θ are still unknown, and therefore any information about θ is included in its prior distribution. Each observational error, e_i, is assumed to be normally distributed with zero mean, and constant variance σ_e^2, i.e.

$$e_i|\sigma_e^2 \sim N\left(0, \sigma_e^2\right), \quad \forall i = 1, \ldots, n. \tag{7.3}$$

We represent the uncertainty about the discrepancy function by a Gaussian process with mean function $m_d(\cdot)$ and covariance function $V_d(\cdot, \cdot)$:

$$d(\cdot) \sim GP(m_d(\cdot), V_d(\cdot, \cdot)). \tag{7.4}$$

The mean function, $m_d(\cdot)$, is generally represented by a linear function $h_d(\cdot)^T\beta_d$, where $h_d(\cdot)$ is a known function and β_d a vector of unknown regression parameters. The covariance function is a stationary covariance function $V_d(\cdot, \cdot)$. Here, we use a homoscedastic covariance function in the form $V_d(\cdot, \cdot) = \sigma_d^2 C_{\delta_d}(\cdot, \cdot)$. We use the squared exponential correlation function (2.6) with correlation lengths given by the vector $\delta_d = (\delta_{d,1}, \ldots, \delta_{d,p})$.

Therefore, our prior uncertainty about the real system $\xi(\cdot)$ is described by a Gaussian process with mean and covariance functions given by $m_\xi(\cdot)$ and $V_\xi(\cdot, \cdot)$ respectively, i.e.

$$\xi(\cdot)|\theta, \sigma_d^2, \delta_d \sim GP\left(m_\xi(\cdot), V_\xi(\cdot, \cdot)\right), \tag{7.5}$$

where

$$m_\xi(\mathbf{x}) = E[\xi(\mathbf{x})|\theta] = \eta(\mathbf{x}, \theta) + m_d(\mathbf{x}), \tag{7.6}$$

$$V_\xi(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}[\xi(\mathbf{x}), \xi(\mathbf{x}')|\sigma_d^2, \delta_d] = V_d(\mathbf{x}, \mathbf{x}'). \tag{7.7}$$

The training data is a set of experimental data from the real system, $(\mathbf{z}, \mathbf{X})$, where, for instance, we have n different input values, $\mathbf{X} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T)^T$ and for each input $\mathbf{x}_i$, the observed real process is denoted by z_i. The training data is used to update our beliefs about the real process and the parameters θ and $\Psi = (\sigma_d^2, \delta_d, \sigma_e^2)$.

It is straightforward to update our beliefs about $\xi(\cdot)$ conditional on the training data, θ and Ψ. Using properties of the multivariate normal distribution, it can be shown that the posterior distribution of the real process is

$$\xi(\cdot)|\mathbf{z}, \mathbf{X}, \theta, \Psi \sim GP\left(m_\xi^*(\cdot; \theta, \Psi), V_\xi^*(\cdot, \cdot; \Psi)\right). \tag{7.8}$$

The posterior mean function is given by

$$\begin{aligned} m_\xi^*(x; \theta, \Psi) &= E[\xi(x)|\mathbf{z}, \mathbf{X}, \theta, \Psi] \\ &= m_\xi(x; \theta) + V_\xi(x, \mathbf{X})^T \left(V_\xi(\mathbf{X}) + \sigma_e^2 \mathcal{I}_n\right)^{-1} (\mathbf{z} - m_\xi(\mathbf{X}; \theta)), \end{aligned} \tag{7.9}$$

where the notation $m_\xi(\mathbf{X}; \theta)$ corresponds to the vector $(m_\xi(\mathbf{x}_1; \theta), \ldots, m_\xi(\mathbf{x}_n; \theta))^T$, $V_\xi(x, \mathbf{X})$ is a vector such that $V_\xi(\mathbf{x}, \mathbf{X})[i] = V_\xi(\mathbf{x}, \mathbf{x}_i)$ for $i = 1, \ldots, n$, and $V_\xi(\mathbf{X})$ is a matrix such that $V_\xi(\mathbf{X})[i, j] = V_\xi(\mathbf{x}_i, \mathbf{x}_j)$.

The posterior covariance function is

$$\begin{aligned} V_\xi^*(x, x'; \Psi) &= \mathrm{Cov}[\xi(x), \xi(x')|\mathbf{z}, \mathbf{X}, \Psi] \\ &= V_\xi(x, x'|\Psi) - V_\xi(x, \mathbf{X})^T \left(V_\xi(\mathbf{X}) + \sigma_e^2 \mathcal{I}_n\right)^{-1} V_\xi(x, \mathbf{X}). \end{aligned} \tag{7.10}$$

The posterior distribution (7.8) depends on the true values of parameters (θ, Ψ) which are unknown quantities. The likelihood for (θ, Ψ) is built from (7.5), and hence the posterior distribution for (θ, Ψ) is

$$p(\theta, \Psi|\mathbf{z}, \mathbf{X}) \propto p(\theta, \Psi) MN(\mathbf{z}, m_z(\mathbf{X}; \theta), \left(V_\xi(\mathbf{X}) + \sigma_e^2 \mathcal{I}_n\right)) \tag{7.11}$$

where $p(\theta, \Psi)$ is a prior distribution, $MN(x, m, V)$ denotes the density of the multivariate

normal at vector x, with mean vector m and covariance matrix V.

7.3 Inference for discrepancy function models

We are interested in making predictions about the real system, $\xi(\cdot)$, conditioned on a set of experimental data using the predictive process (7.8). However, we should deal with the unknown quantities (θ, Ψ). The simplest way is to estimate (θ, Ψ) and condition on them as true values. This method is called the plug-in method, and estimates for (θ, Ψ) can be found by maximizing the posterior (7.11). The plug-in method fails to take into account uncertainty in the unknown quantities (θ, Ψ). An MCMC algorithm could be used, although it is computationally expensive. Kennedy and O'Hagan (2001) proposed a method where the calibration parameter is numerically integrated out, and the remaining parameters are estimated and used as true values. Nagy et al. (2007) proposed a Monte Carlo method that approximates the posterior (2.19) by a multivariate log normal and integrates over the correlation parameters in a Gaussian process emulator. We adapt Nagy et al.'s algorithm for the discrepancy function model.

In order to illustrate the inference methods, we use the following toy example.

Example 7.1 (Toy example) *We assume that the real process is one realization of the model (7.5), where*

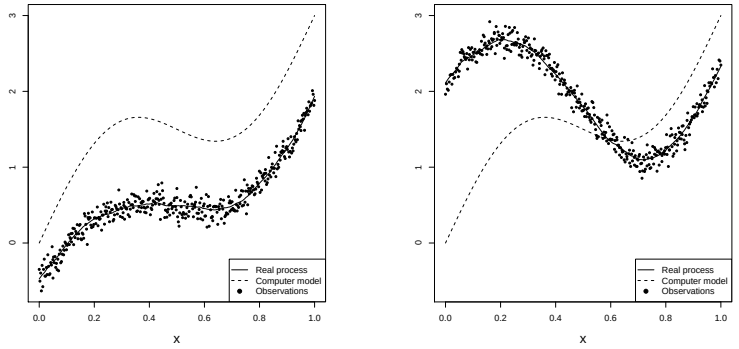
$$\begin{aligned} \eta(x,\theta) &= x\theta + \sin(3x\theta) \\ m_d(x) &= 0, \forall x \\ V_d(x,x') &= \sigma_d^2 \exp\left\{-\frac{(x-x')^2}{\delta_d^2}\right\} \end{aligned}$$

We generated two different samples from this process:

(a) *400 noisy observations illustrated in Figure 7.1 (a) where 15 are randomly chosen as training data.*

(b) *400 noisy observations illustrated in Figure 7.1 (b) where 100 are randomly chosen as training data.*

The parameters used to generate the processes were $\theta = 3$, $\sigma_d^2 = 2$, $\delta_d = 0.5$, *and* $\sigma_e^2 = 0.01$.



(a) (b)

Figure 7.1: Two different processes sampled from the Gaussian process described in example 7.1 and the associated simulator.

For example 7.1, we use independent vague priors for θ and Ψ.

$$\begin{aligned}
\theta &\sim N(0, 100), \\
\sigma_d^2 &\sim InvGamma(4.01/2, 2.01/2), \\
\delta_d &\sim Gamma(0.01, 0.01), \\
\sigma_e^2 &\sim InvGamma(4.01/2, 2.01/2).
\end{aligned}$$

However, for most situations there may be at least some information about the likely size of measurement errors, i.e., information about σ_e^2, that should be included in the prior.

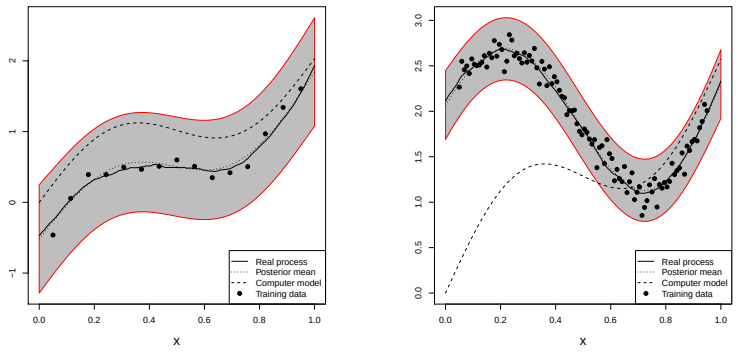
7.3.1 Plug-in method

Predictions for the real process are based on the process (7.8) using plug-in estimates $(\hat{\theta}, \hat{\Psi})$ for (θ, Ψ). For instance, $(\hat{\theta}, \hat{\Psi})$ could be the joint posterior mode of (7.11) obtained by an

optimization method, such as the Downhill simplex method (Nelder and Mead 1965) or the simulated annealing algorithm (Belisle 1992).

For example 7.1 (a), the plug-in method is applied to the training data with size 15 randomly chosen from the 400 observations. The parameter estimates are $\hat{\theta} = 2.0335$, $\hat{\sigma}_d^2 = 0.2845$, $\hat{\delta}_d = 0.5060$ and $\hat{\sigma}_e^2 = 0.1156$. The calibration parameter and the variances were poorly estimated, especially the estimated observation variance which is quite big. Figure 7.2 (a) presents the training data and the estimated real process using plug-in estimates of (θ, Ψ). The mean estimated process represents the real process well, but the associated uncertainty is very high. This is due to the poor estimation of some parameters.

For example 7.1 (b), a larger training dataset containing 100 observations was chosen and (θ, Ψ) were estimated. The parameter estimates are $\hat{\theta} = 2.5721$, $\hat{\sigma}_d^2 = 0.7808$, $\hat{\delta}_d = 0.7492$ and $\hat{\sigma}_e^2 = 0.0295$. Figure 7.2 (b) presents the training data and the estimated real process. The parameter estimates are closer to the true values, but the uncertainty is apparently greater than it should be.

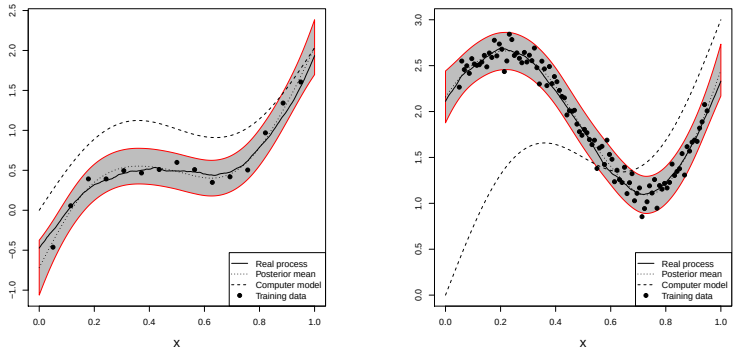


(a) (b)

Figure 7.2: Estimated processes using the plug-in method: (a) using 15 training data chosen from example 7.1 (a); (b) using 100 training data chosen from example 7.1 (b).

The large uncertainty for the predictive intervals of example 7.1 (a) and (b) is due to poor estimation of the unknown parameters (θ, Ψ). In order to illustrate this, Figure 7.3 presents predictions for the real system using the true value for each parameter $\theta = 3$, $\sigma_d^2 = 2$, $\delta_d = 0.5$ and $\sigma_e^2 = 0.01$. The widths of the predictive intervals given the true values are

visably lower than those of the predictive intervals given the estimated values presented in Figure 7.2.



Example 7.1 (a) Example 7.1 (b)

Figure 7.3: Estimated processes using the plug-in method at the true value for each parameter.

7.3.2 Nagy et al.'s approach

Nagy et al. (2007) propose a method for inference about the correlation parameters in a Gaussian process emulator. The authors present a Monte Carlo method where the distribution of the unknown parameters, on the log scale, is approximated by a normal distribution centred on the posterior mode and covariance matrix given by the negative of the inverse of the Hessian matrix of the posterior density for the parameters of interest. The authors also compared their approach with the plug-in and MCMC approaches, where they found that their approach is better in the sense that it combines the computational efficiency of the plug-in method and the Bayesian approach of dealing with the uncertainty in the unknown parameters.

We adapt Nagy et al.'s method for the discrepancy model (7.1). The normal approximation for the posterior distribution of $(\theta, \log \Psi)$ is derived from (7.11). This approach is summarized as follows:

1. Obtain the posterior mode of the posterior distribution of $\tau = (\theta, \log \Psi)$, denoted by

$\hat{\tau}$.

2. Compute the Hessian matrix (the matrix of second derivatives) at $\hat{\tau}$, denoted by $\mathcal{H}_\tau$.

3. Sample from the multivariate normal distribution $N(\hat{\tau}, -\mathcal{H}_\tau^{-1})$ to obtain M Monte Carlo samples $\tau^{(1)}, \ldots, \tau^{(M)}$.

4. The approximated predictive mean is given by the average

$$\widehat{E}[\xi(x)|\mathbf{z}, \mathbf{X}] = \frac{1}{M}\sum_{k=1}^{M} m_1^*(x; \theta^{(k)}, \Psi^{(k)}), \tag{7.12}$$

and the approximated predictive covariance is given by

$$\widehat{\mathrm{Cov}}[\xi(x), \xi(x')|\mathbf{z}, \mathbf{X}] = \widehat{E}[\xi(x)\xi(x')|\mathbf{z}, \mathbf{X}] - \widehat{E}[\xi(x)|\mathbf{z}, \mathbf{X}]\widehat{E}[\xi(x')|\mathbf{z}, \mathbf{X}]^T, \tag{7.13}$$

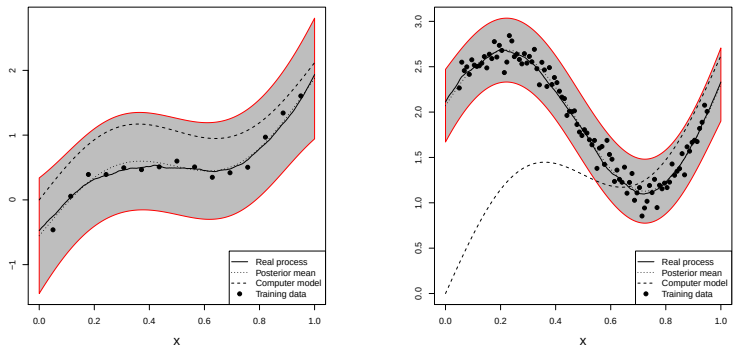
where

$$\widehat{E}[\xi(x)\xi(x')|\mathbf{z}, \mathbf{X}] = \frac{1}{M}\sum_{k=1}^{M} V_\xi^*(x, x'; \Psi^{(k)}) + m_\xi^*(x; \theta^{(k)}, \Psi^{(k)}) m_\xi^*(x'; \theta^{(k)}, \Psi^{(k)})^T, \tag{7.14}$$

$m_\xi^*(x; \theta, \Psi)$ and $V_\xi^*(x, x'; \theta, \Psi)$ are given in equations (7.9) and (7.10) respectively.

Nagy et al.'s method was applied to the same dataset used to illustrate the plug-in method. The posterior mode for $(\theta, \log(\Psi))$ and the Hessian matrix were obtained using the optimization method '*optim*' implemented in R. The estimated process for example 7.1 (a) and (b) are presented in Figure 7.4 (a) and (b) respectively. In both cases, the real process is contained in the 95% credible intervals, but the uncertainty is visually large, especially for example 7.1 (a).

In both cases in example 7.1, the predictive intervals for the real system $\xi(\cdot)$ using Nagy et al.'s method are very similar to those predictive intervals using the plug-in method (Figure 7.2). The reason for that is the poor estimation of the parameters (θ, Ψ). The approximated density functions for each parameter of both examples 7.1 (a) and (b) are presented in Figure 7.5. Both variances were poorly estimated, the true values being both far out in the tail of



(a) (b)

Figure 7.4: Estimated processes using Nagy et al.'s method: (a) using 15 training data chosen from example 7.1 (a); (b) using 100 training data chosen from example 7.1 (b).

the approximated distributions. This is the reason for the large uncertainty in the predictions in Figure 7.4.

7.3.3 Kennedy and O'Hagan's approach

This approach was proposed by Kennedy and O'Hagan (2000, 2001). Predictions for the real process take into account uncertainty about the calibration parameter. In order to predict the real process it is necessary to find plug-in estimates for Ψ and to integrate θ out from the joint posterior. We assume that the calibration parameter θ and the parameter Ψ are independent, i.e $p(\theta, \Psi) = p(\theta)p(\Psi)$. However this assumption is not strictly necessary.

The predictive distribution for the real process cannot be analytically derived, however, conditional on the parameter Ψ, we can derive the predictive mean as

$$\begin{aligned} E[\xi(x)|\mathbf{z}, \mathbf{X}, \Psi] &= \int E[\xi(x)|\mathbf{y}, \mathbf{X}, \Psi, \theta]\, p(\theta|\mathbf{z}, \mathbf{X}, \Psi) d\theta \\ &= \int m^*_\xi(x; \theta, \Psi) p(\theta|\mathbf{z}, \mathbf{X}, \Psi) d\theta, \end{aligned} \tag{7.15}$$

and the predictive covariance function is

$$\mathrm{Cov}\left[\xi(x), \xi(x')|\mathbf{z}, \mathbf{X}, \Psi\right] = E[\xi(x)\xi(x')|\mathbf{z}, \mathbf{X}, \Psi] - E[\xi(x)|\mathbf{z}, \mathbf{X}, \Psi]\, E[\xi(x')|\mathbf{z}, \mathbf{X}, \Psi], \tag{7.16}$$

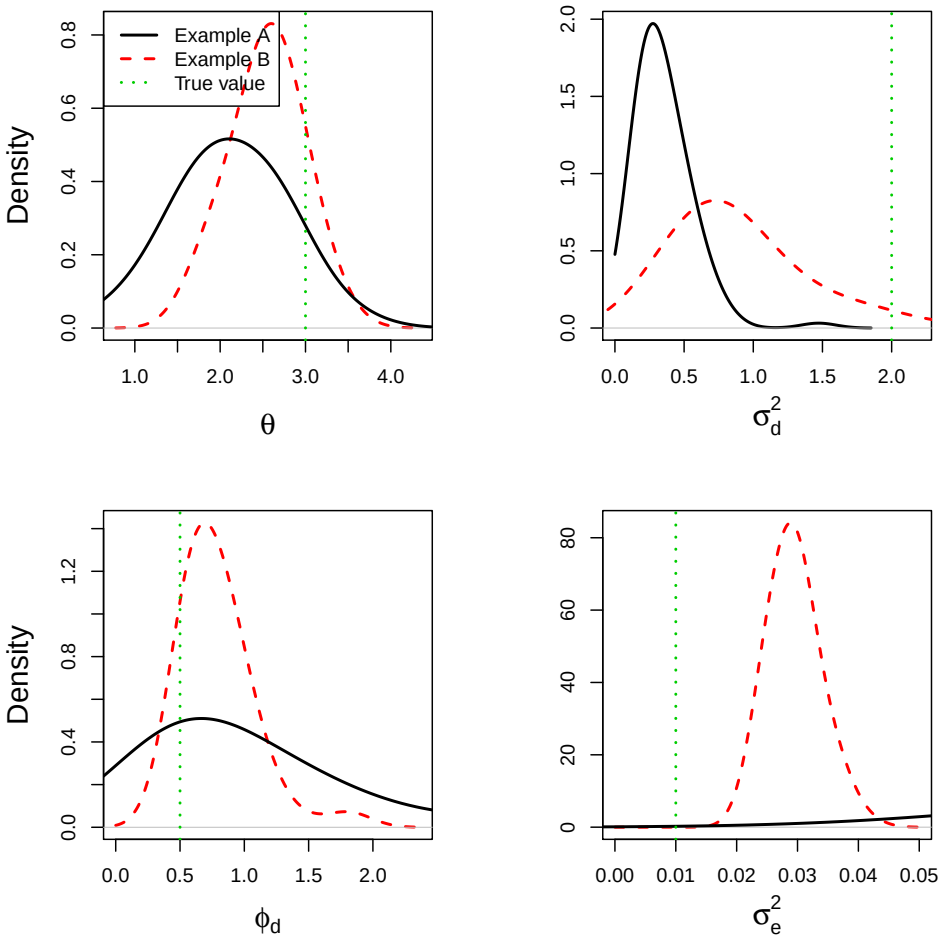


Figure 7.5: Approximated marginal posterior distribution of each unknown parameter of example 7.1 (a) and (b) using Nagy et al.'s method. The true value for each parameter is presented as a vertical dotted line on each Figure.

where $E[\xi(x)\xi(x')|\mathbf{z},\mathbf{X},\Psi] = \int \left(V_\xi^*(x,x';\Psi) + m_\xi^*(x;\theta,\Psi)m_\xi^*(x';\theta,\Psi)\right) p(\theta|\mathbf{z},\mathbf{X},\Psi)d\theta$, and $m_\xi^*(\cdot;\theta,\Psi)$ and $V_\xi^*(\cdot,\cdot;\Psi)$ are presented in equation (7.8).

The integrals in (7.15) and (7.16) are recommended to be solved using the iterative Gauss-Hermite quadrature method (Kennedy and O'Hagan 2001).

Plug-in estimates for Ψ are obtained as

$$\hat{\Psi} = arg\max_{\Psi} p(\Psi|\mathbf{z},\mathbf{X}), \tag{7.17}$$

where

$$p(\Psi|\mathbf{z},\mathbf{X}) = \int_{\theta} p(\theta,\Psi|\mathbf{z},\mathbf{X})d\theta, \tag{7.18}$$

and $p(\theta,\Psi|\mathbf{z},\mathbf{X})$ is defined in equation (7.11).

7.3.4 MCMC approach

In this approach, to make predictions of the real system, the calibration parameter θ and the parameters Ψ are integrated out from the joint posterior. The predictive distribution for the real process cannot be analytically derived. However, we can approximate the predictive mean and the predictive covariance as (7.12) and (7.13). The samples $((\theta^{(1)},\Psi^{(1)}),\ldots,(\theta^{(M)},\Psi^{(M)}))$ are generated from the posterior distribution (7.11) using an MCMC sampler. Assuming that θ and Ψ are independent *a priori*, the full conditional distributions are

$$p(\theta|\mathbf{z},\mathbf{X},\Psi) \quad \propto \quad p(\theta)\exp\left\{-\frac{1}{2}(\mathbf{z}-m_\xi)^T V_\xi^{-1}(\mathbf{z}-m_\xi)\right\}, \tag{7.19}$$

and

$$p(\Psi|\mathbf{z},\mathbf{X},\theta) \quad \propto \quad p(\Psi)|V_\xi|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(\mathbf{z}-m_\xi)^T V_\xi^{-1}(\mathbf{z}-m_\xi)\right\}. \tag{7.20}$$

The MCMC algorithm consists of the following steps.

1. Initialize the iteration process, i.e. attribute values for $\theta^{(0)}$ and $\Psi^{(0)}$.
2. Sample a value for $\theta^{(k)}$ from the conditional distribution $f(\theta|\mathbf{z},\mathbf{X},\Psi^{(k-1)})$ given in (7.19).
   (a) Sample $\theta^\star$ from a proposal distribution, $q(\cdot|\theta^{(k-1)})$.
   (b) Accept $\theta^\star$, i.e. $\theta^{(k)}=\theta^\star$, with probability

$$\alpha(\theta^\star,\theta^{(k-1)}) = \min\left(1, \frac{f(\theta^\star|\mathbf{z},\mathbf{X},\Psi^{(k-1)})q(\theta^{(k-1)}|\theta^\star)}{f(\theta^{(k-1)}|\mathbf{z},\mathbf{X},\Psi^{(k-1)})q(\theta^\star|\theta^{(k-1)})}\right). \tag{7.21}$$

(c) If $\theta^\star$ is rejected, then $\theta^{(k)} = \theta^{(k-1)}$.

3. Sample a value for $\Psi^{(k)}$ from the conditional distribution $f(\Psi|\mathbf{z}, \mathbf{X}, \theta^{(k)})$ given in (7.20).

(a) Sample $\Psi^\star$ from a proposal distribution, $q(\cdot|\Psi^{(k-1)})$

(b) Accept $\Psi^\star$, i.e. $\Psi^{(k)} = \Psi^\star$, with probability

$$\alpha(\Psi^\star, \Psi^{(k-1)}) = \min\left(1, \frac{f(\Psi^\star|\mathbf{z}, \mathbf{X}, \theta^{(k)})q(\Psi^{(k-1)}|\Psi^\star)}{f(\Psi^{(k-1)}|\mathbf{z}, \mathbf{X}, \theta^{(k)})q(\Psi^\star|\Psi^{(k-1)})}\right). \tag{7.22}$$

(c) If $\theta^\star$ is rejected, then $\theta^{(k)} = \theta^{(k-1)}$.

4. Repeat steps 2 and 3 a large number of times and monitor convergence of the chain.

Although the Kennedy and O'Hagan's method and the MCMC approach are more appropriate methods for dealing with uncertainty in the unknown parameters, both of these methods are computationally expensive. In the next section, we present diagnostics for validating the discrepancy function model using the plug-in and Nagy et al.'s methods. The diagnostics for the discrepancy function models using Kennedy and O'Hagan and the MCMC approaches would be the same as those diagnostics using Nagy et al.'s approach.

7.4 Diagnostics for validating the discrepancy function model

Discrepancy function models should be validated. Otherwise, predictions from an invalid model may simply lead to wrong conclusions about the real process. In order to validate a model, it is necessary to use a set of diagnostics. In Chapter 4, we proposed a set of numerical and graphical diagnostics to validate Gaussian process emulators. The proposed diagnostics are based on the performance of the predictive model on a new dataset, called the validation data. For each diagnostic there is an observed value from the validation data, and a distribution induced by the predictive distribution. Extreme values may indicate non

valid emulators. The same procedure can be used for validating the discrepancy function model.

Let $(\mathbf{z}^{(v)}, \mathbf{X}^{(v)})$ be the validation data, where $z_i^{(v)}$ is an noisy observation of the real system under conditions given by the input vector $\mathbf{x}_i^{(v)}$, and $\mathbf{X}^{(v)} = ({\mathbf{x}_1^{(v)}}^T, \ldots, {\mathbf{x}_m^{(v)}}^T)^T$. The size of the validation data is defined as m. There are no studies on the literature about the ideal size of the validation data, but we recommend that the validation data cover the input space. A diagnostic for validation could be any statistic $D^G(\cdot)$ that can be evaluated at the observed validation data, $D^G(\mathbf{z}^{(v)})$, and compared with the induced distribution of the random variable $D^G(\xi(\mathbf{X}^{(v)}))$. A good diagnostic is the one that provides information on where the problem could be in case of a model that is not valid.

7.4.1 Diagnostics when the plug-in method is used

The set of diagnostics when the plug-in method is used are essentially the same as those diagnostics presented in Chapter 4. The reason for that is because in Chapter 4 we developed diagnostics for Gaussian process emulators, and here the predictive process for the real system is also a Gaussian process, (7.8).

The simplest diagnostic is the standardised individual predictive error, $D^I(\cdot)$, described in (4.2).

$$D^I(z_i^{(v)}; \mathbf{x}_i^{(v)}) = \frac{z_i^{(v)} - E[\xi(\mathbf{x}_i^{(v)})]}{\sqrt{\mathrm{Var}[\xi(\mathbf{x}_i^{(v)})]}}, \qquad \text{for } i = 1, \ldots, m. \tag{7.23}$$

For a valid predictive model we expect to observe values between -2 and 2, which corresponds to the 95% credible interval for each individual error, since each prediction is normally distributed as (7.8) with plug-in estimates for (θ, Ψ). The individual errors can be plotted against the predictive mean, or against any of the inputs to see if there is a pattern. The problem, however, is that such plot does not take into account the correlation among the predictions.

The next diagnostic takes into account the correlation among the predictions. The Maha-

lanobis distance, $D_{MD}(\cdot)$, is a generalisation of the sum of squares of the individual errors. It requires the predictive mean and predictive covariance matrix:

$$D_{MD}(\mathbf{z}^{(v)};\mathbf{X}^{(v)}) = (\mathbf{z}^{(v)} - E[\xi(\mathbf{X}^{(v)})])^T \operatorname{Cov}[\xi(\mathbf{X}^{(v)})]^{-1}(\mathbf{z}^{(v)} - E[\xi(\mathbf{X}^{(v)})]) \tag{7.24}$$

where the i-th element of vector $E[\xi(\mathbf{X}^{(v)})]$ is $E[\xi(\mathbf{x}_i^{(v)})]$, and the element (i,j) of the matrix $\operatorname{Cov}[\xi(\mathbf{X}^{(v)})]$ is $\operatorname{Cov}[\xi(\mathbf{x}_i^{(v)}), \xi(\mathbf{x}_j^{(v)})]$. For a valid model, the Mahalanobis distance follows a chi squared distribution with degrees of freedom given by $rank(\operatorname{Cov}[\xi(\mathbf{X}^{(v)})])$. A large value for this diagnostic may suggest overconfidence. On the other hand, a small value may suggest underconfidence.

In Section 4.3.3, we presented the uncorrelated errors of the form $\mathbf{G}^{-1}(\mathbf{z}^{(v)} - E[\xi(\mathbf{X}^{(v)})])$, where the predictive variance matrix is decomposed into $\operatorname{Cov}[\xi(\mathbf{X}^{(v)})] = \mathbf{G}\mathbf{G}^T$. There are several decomposition methods and we used the pivoted Cholesky decomposition. The pivoted Cholesky decomposition of the variance matrix is given by $\mathbf{P}^T \operatorname{Cov}[\xi(\mathbf{X}^{(v)})]\mathbf{P} = \mathbf{U}^T\mathbf{U}$, where $\mathbf{U}$ is an upper triangular matrix and $\mathbf{P}$ is a permutation matrix. Therefore, the pivoted Cholesky errors are given by

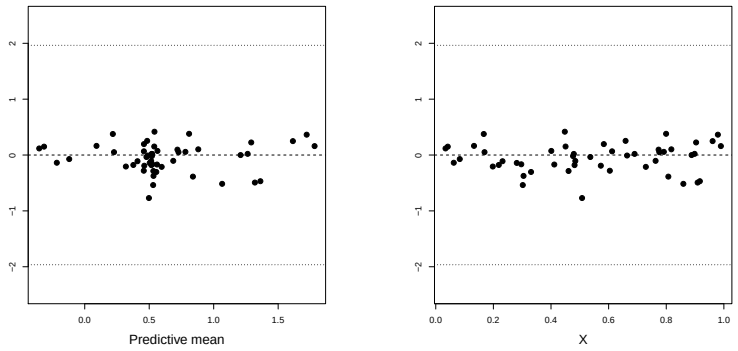
$$D^{PC}(\mathbf{z}^{(v)};\mathbf{X}^{(v)}) = \mathbf{P}\mathbf{U}^{-1}(\mathbf{z}^{(v)} - E[\xi(\mathbf{X}^{(v)})]). \tag{7.25}$$

One advantage of the pivoted Cholesky error is that each single pivoted Cholesky error is associated with one validation data point. Therefore, if an extreme value for a $D^{PC}(\mathbf{z}^{(v)};\mathbf{X}^{(v)})$ is observed, the associated validation data can be tracked where the cause of the problem may be identified. If the predictive distribution is known, one very useful graphical display is the QQ-plot with credible intervals (Houseman et al. 2004).

Toy Examples

Example 7.1 (a). The training data consisted of a random sample of 15 observations and the estimated model is presented in Figure 7.2 (a). For this example, we use 50 observations randomly chosen as validation data points. The individual errors (7.23) are presented in

Figure 7.6. Figure 7.6 (a) presents the individual errors against the predictive mean. There is no pattern associated with the mean function, however, the variability of the errors is smaller than expected. Figure 7.6 (b) presents the individual errors versus the input. There is no pattern suggesting a non-stationary problem.

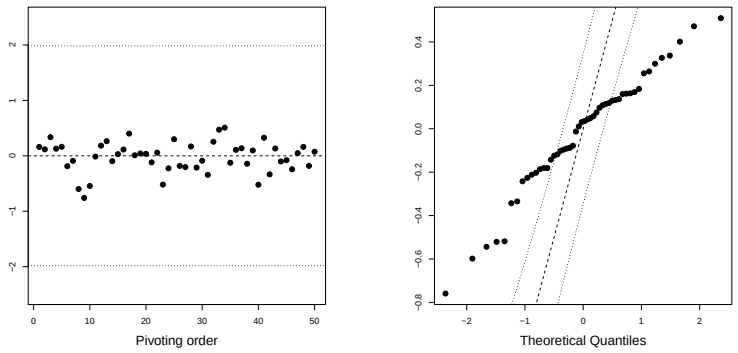


(a) (b)

Figure 7.6: Validation diagnostics for the toy example 7.1 (a), using 15 training data points and 50 validation data points, based on the individual errors: (a) $D^I(\mathbf{z}^{(v)})$ against the predictive mean; (b) $D^I(\mathbf{z}^{(v)})$ against the validation inputs.

Figure 7.7 (a) presents the pivoted Cholesky errors (7.25), where the variability of the errors is considerably smaller than expected. The Mahalanobis distance (7.24), $D_{MD}(\mathbf{z}^{(v)}) = 3.7317$, confirms the underconfidence, where the expected value is 50, and the credible interval is (32.36, 71.42). Figure 7.7 (b) presents the Quantile-Quantile plot that emphasizes that the error variability is small. We now consider increasing the training data sample size.

Using the selected training data, uncertainty in the predictions is very large. The parameters were poorly estimated, especially $\hat{\sigma}_e^2$ which is far too big. To reduce this uncertainty, we can use more data, in particular more data at the same training inputs to obtain more information about the observational variance error. So, we observe again the real process at the original training inputs, and add the validation data to the training data. Therefore, the new training data contain 80 observations: 15 for the previous training data, 50 for the validation and 15 more observations evaluated at the same inputs as the initial training data.

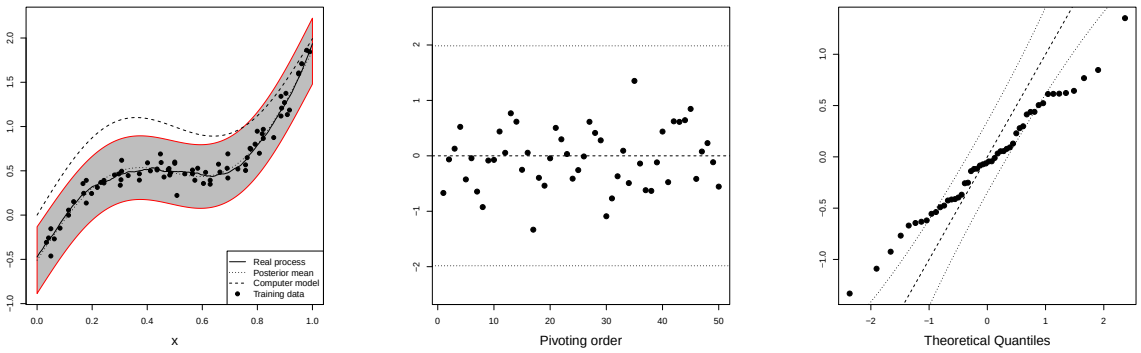


(a) (b)

Figure 7.7: Validation diagnostics for the toy example 7.1 (a) , using 15 training data points and 50 validation data points, based on the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{z}^{(v)})$ against the pivoting order; (b) QQ plot of $D^{PC}(\mathbf{z}^{(v)})$.

A new validation dataset containing 50 observations was randomly chosen and used for diagnostics. Figure 7.8 (a) presents the estimated process using the new training data. The range of the predictive intervals are smaller than before, but they still seem to be larger than they should be. The Mahalanobis distance is 14.54, which is smaller than the expected 50 and the 95% credible interval is (32.36, 71.42). Figure 7.8 (b) does not suggest poor estimation of the correlation lengths because the variability of the errors is constant throughout the pivoting order, but the variance is too small which suggests either the estimated discrepancy variance, $\hat{\sigma}_d^2$, or the observational error variance, $\hat{\sigma}_e^2$, is too large. According to Figure 7.8 (c) the normality assumption seems reasonable, but the variance of the pivoted Cholesky errors is too small.

We now suppose that we know the observational variance, i.e., $\sigma_e^2 = 0.01$. This assumption is not too strong; for instance this information could be obtained throughout the specifications of the measurement instrument used to collect the data. The remaining parameters $(\theta, \sigma_d^2, \delta_d)$ still need to be estimated. We use the 80 training data points to estimate them. Figure 7.9 (a) presents the predictive real process. 50 new validation data elements were randomly chosen, and the Mahalanobis distance is 48.02, which is very close to its expected value. Figure 7.9 (b) and (c) present diagnostics with the pivoted Cholesky errors, and both figures suggests that the estimated process is a valid model.



(a) (b) (c)

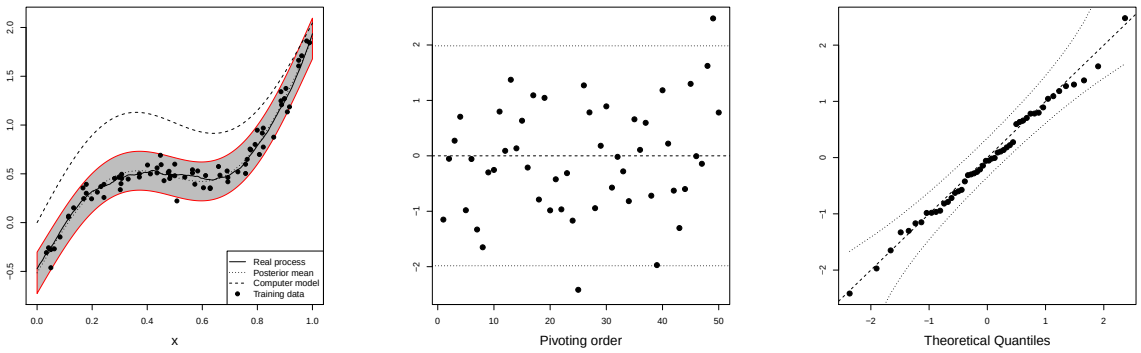
Figure 7.8: Validation diagnostics for the toy example 7.1 (a), using 80 training data points and 50 new validation data points, based on the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{z}^{(v)})$ against the pivoting order; (b) QQ plot of $D^{PC}(\mathbf{z}^{(v)})$.

Example 7.1 (b). The training data consist of a random sample of 100 observations used to build the predictive model presented in Figure 7.2 (b). The validation data consist of 50 observations randomly selected from the remaining 300 observations sampled from the real process. Figure 7.10 (a) presents the individual errors against the predictive mean and Figure 7.10 (b) presents the individual errors against the input. In both cases, there is no noticeable pattern associated with the mean function, but the variability of the errors is smaller than expected.

The observed Mahalanobis distance $D_{MD}(\mathbf{z}^{(v)}) = 12.3824$ suggests underconfidence. The expected value is 50, and the credible interval is (32.36, 71.42). Figure 7.11 (a) presents the pivoted Cholesky errors (7.25), where the variability is very small, but there is no patterns indicating problems on the correlation structure. Figure 7.11 (b) presents the Quantile-Quantile plot where we observe light tails indicating small variability.

7.4.2 Diagnostics when numerical approximations are used

Here, we present some diagnostics for the discrepancy function model when the parameters (θ, Ψ) are numerically integrated out via sampling methods. This includes methods such as MCMC and Nagy et al.'s method. The diagnostics depend on the predictive distribution



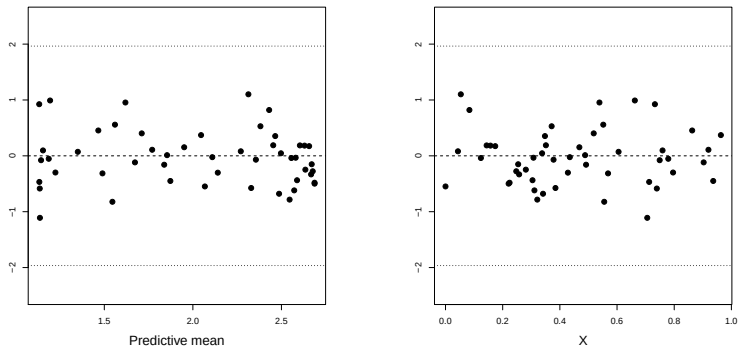
(a) (b) (c)

Figure 7.9: Validation diagnostics for the toy example 7.1 (a), using 80 training data points, 50 validation data points and a fixed value for the observational variance ($\sigma_e^2 = 0.01$), based on the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{z}^{(v)})$ against the pivoting order; (b) QQ plot of $D^{PC}(\mathbf{z}^{(v)})$.

of $\xi(\mathbf{X}^{(v)})$ which cannot be derived analytically. Two ways to deal with this are (i) using a Gaussian approximation of the predictive distribution of $\xi(\mathbf{X}^{(v)})$ with vector mean and covariance matrix derived numerically via (7.12) and (7.13); (ii) using *composition* sampling, (Banerjee et al. 2003, page 132), where we sample from the predictive distribution of $\xi(\mathbf{X}^{(v)})$. The k-th sample from the predictive distribution of $\xi(\mathbf{X}^{(v)})$ is obtained by sampling from (7.8) conditional on the training data and $(\theta^{(k)}, \Psi^{(k)})$.

The individual errors (7.23) for a valid model are expected to be values randomly distributed around zero, and the distribution for each individual error is, for case (i), approximated by a standard Gaussian distribution. For case (ii), the distribution of the individual errors is derived numerically for each error. Analogously, the pivoted Cholesky errors (7.25) should be independent and approximated by standard Gaussian for case (i) and numerically derived for case (ii). In the numerical approximation (ii), we keep the pivoting order of the validation data, to guarantee that the distribution of each numerical pivoted Cholesky error correspond to one observed pivoted Cholesky error.

The Mahalanobis distance, regardless of the induced distribution, has an expected value given by the rank of the predictive covariance matrix, m (Goldstein and Wooff 2007). Under the Gaussian approximation, case (i), the approximated distribution of the Mahalanobis



(a) (b)

Figure 7.10: Validation diagnostics for the toy example 7.1 (b), using 100 training data points and 50 validation data points, based on the individual errors: (a) $D^I(\mathbf{z}^{(v)})$ against the predictive mean; (b) $D^I(\mathbf{z}^{(v)})$ against the validation inputs.

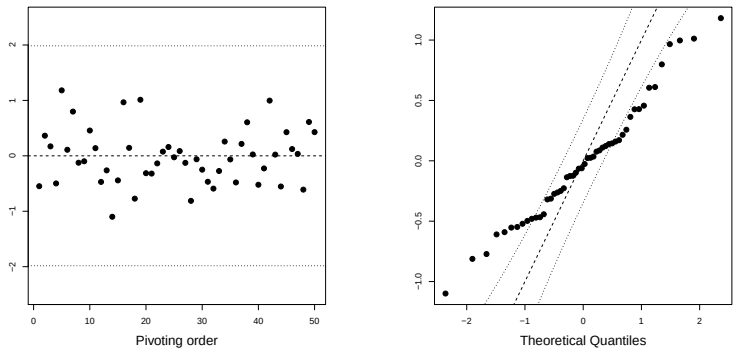
distance is a chi-squared distribution with m degrees of freedom. For case (ii) the distribution is numerically approximated.

We use example 7.1 to illustrate both methods for deriving the predictive distribution of $\xi(\mathbf{X}^{(v)})$.

Toy example using the Gaussian approximation

In example 7.1 (a) and (b), the predictive distribution is approximated by a Gaussian distribution, as described previously in case (i). A sample of the unknown parameters (θ, Ψ) is obtained using Nagy et al.'s approach.

Example 7.1 (a). The same 15 training data and 50 validation data points used for the plug-in method are used here. The estimated process using Nagy et al.'s method is presented in Figure 7.4 (a). The predictive distribution for the real process at the validation inputs is approximated by a Gaussian distribution. Figure 7.12 (a) presents the individual errors against the predictive mean. The variability of the errors is very small suggesting underconfidence. Figure 7.12 (b) presents the individual errors versus the inputs which also underconfidence.



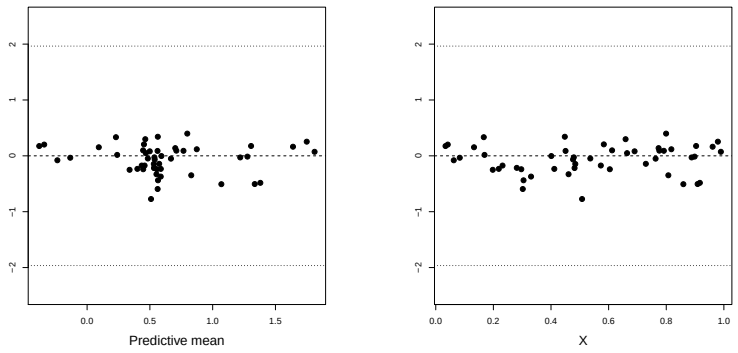
(a) (b)

Figure 7.11: Validation diagnostics for the toy example 7.1 (a), using 100 training data points and 50 validation data points, based on the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{z}^{(v)})$ against the pivoting order; (b) QQ plot of $D^{PC}(\mathbf{z}^{(v)})$.

The pivoted Cholesky errors (7.25) are shown in Figure 7.13 (a). It can be seen that the variability of the errors is small indicating underconfidence. The Mahalanobis distance (7.24) confirms this, with $D_{MD}(\mathbf{z}^{(v)}) = 3.6933$, where under the Gaussian approximation, the expected value for the Mahalanobis distance should be 50, with approximate 95% credible interval given by (32.36, 71.42). Figure 7.13 (b) presents the Quantile-Quantile plot that emphasizes that the error variability is small. In summary, all diagnostics are indicating underconfidence. We suggest increasing the size of the training dataset to reduce the uncertainty in the unknown parameters.

Comparing the prediction using the plug-in method with those using the Nagy et al.'s method, we notice the predictions are very similar. This can be done visually comparing Figure 7.2 (a) with Figure 7.4 (a), and numerically the Mahalanobis distance using the same validation data are respectively 3.7317 and 3.6933.

Example 7.1 (b). The same 100 training data and 50 validation data points used on the plug-in method are used here. The estimated process using Nagy et al.'s method is presented in Figure 7.4 (a). Figure 7.14 (a) presents the individual errors against the predictive mean. The variability of the errors is small suggesting underconfidence. Figure 7.14 (b) presents the individual errors versus the inputs which also underconfidence.



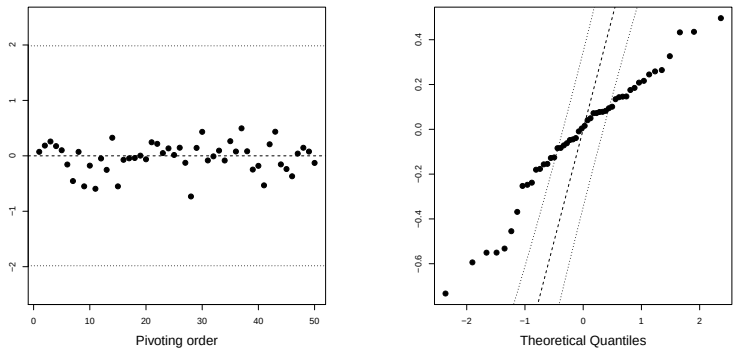
(a) (b)

Figure 7.12: Validation diagnostics for the Gaussian approximation of the toy example 7.1 (a) using 15 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach: Individual errors (a) against the predictive mean; (b) against the validation inputs.

The Mahalanobis distance is $D_{MD}(\mathbf{z}^{(v)}) = 11.9662$, where the expected value is 50, and the numerical approximation of the 95% credible interval is (27.1929, 89.6213) suggesting underconfidence. The pivoted Cholesky errors (7.25) are shown in Figure 7.15 (a). It can be seen that the variability of the errors is small indicating underconfidence. Figure 7.15 (b) presents the Quantile-Quantile plot that emphasizes that the error variability is small. In summary, all diagnostics are indicating underconfidence. We suggest increasing the size of the training dataset to reduce the uncertainty in the unknown parameters.

In order to tackle the underconfidence problem, a new training data was created adding the validation data and some new observation at 30 randomly selected inputs from the original training data. Hence, the new training data contains 180 observations. A new validation dataset with 50 observations was selected.

Figure 7.16 (a) presents the individual errors against the predictive mean. The variability of the errors is small suggesting underconfidence. Figure 7.16 (b) presents the individual errors versus the inputs which also underconfidence. The Mahalanobis distance is $D_{MD}(\mathbf{z}^{(v)}) = 24.9740$, which is small considering that the expected value is 50, and its numerical approximation for the 95% credible interval is (28.5281, 74.3808) suggesting underconfidence. The pivoted Cholesky errors (7.25) are shown in Figure 7.16 (c). It can be seen that the variability of the errors is small indicating underconfidence.



(a) (b)

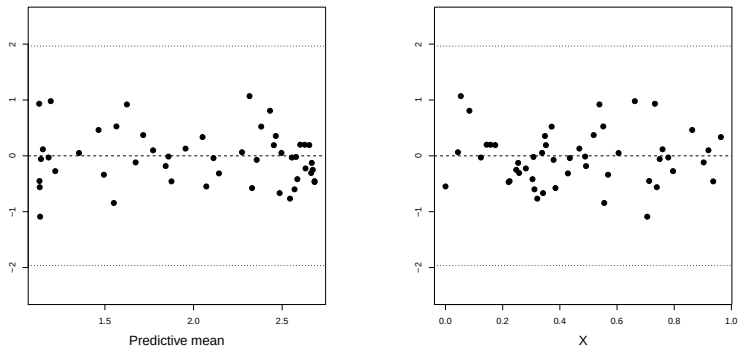
Figure 7.13: Validation diagnostics for the Gaussian approximation of the toy example 7.1 (a) using 15 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach based on the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{z}^{(v)})$ against the pivoting order; (b) QQ plot of $D^{PC}(\mathbf{z}^{(v)})$.

After we increased the training data, all diagnostics are still indicating underconfidence. However, we can notice an improvement on the diagnostics towards validating the model for example 7.1 since the variability of the errors is larger when the training data was increased.

Toy example using composition sampling

For examples 7.1 (a) and (b), the same training and validation data used to illustrate the Gaussian approximation are used here to illustrate the composition sampling. Predictions for the real process are numerically obtained using the composition method and samples from the posterior distribution of the unknown parameters (θ, Ψ) are obtained using Nagy et al.'s approach.

Example 7.1 (a). The diagnostics presented in Figure 7.17 suggests underconfidence. The individual errors are presented on Figures 7.17 (a) and (b), we notice a large variability for the errors suggesting underconfidence. Figure 7.17 (c) shows the pivoted Cholesky errors versus the pivoting order. It confirms that the predictive model has large uncertainty. The intervals for pivoted Cholesky errors were made by fixing the pivoting order of for the predictions of the validation data, otherwise, for each prediction sampled using the composition



(a) (b)

Figure 7.14: Validation diagnostics for the Gaussian approximation of the toy example 7.1 (b) using 100 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach: Individual errors (a) against the predictive mean; (b) against the validation inputs.

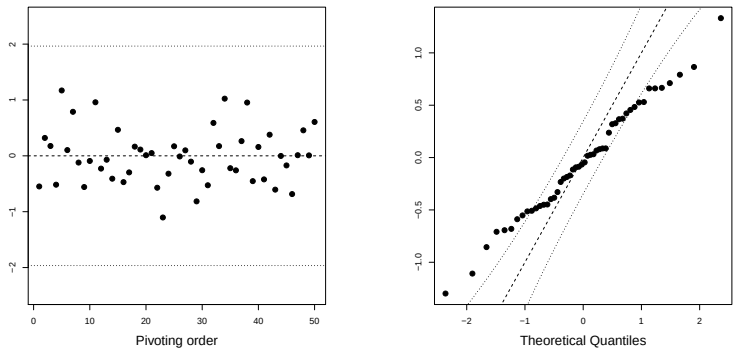
method a new pivoting order could be generated.

Comparing the diagnostics using the Gaussian approximation with the diagnostics when the composition sampling is used, we notice that the intervals using the composition sampling are smaller for this example (Figures 7.12, 7.13 (c), and 7.17).

Example 7.1 (b). Graphical diagnostics are presented in Figure 7.18, and the credible intervals are based on 300 samples of the predictive real process at the validation inputs. The graphical diagnostics suggest underconfidence, and for this example the training data with 180 observation and a new validation dataset are used.

The individual errors are presented in Figures 7.19 (a) and (b), where there are no noticeable patterns. We observe few observations outside the intervals. The uncorrelated errors are presented in Figure 7.19 (c) where again there are no noticeable patterns suggesting a valid model.

We compare the diagnostics of the model built for example 7.1 (b) using the Gaussian approximation with the model using the composition sampling in Figures 7.16 and 7.19. We have two different conclusions, the second model is a valid model, while the first one is still an underconfident model. This is because for that training data sample the Gaussian approximation for the predictive distribution of the real process is not a good approximation,



(a) (b)

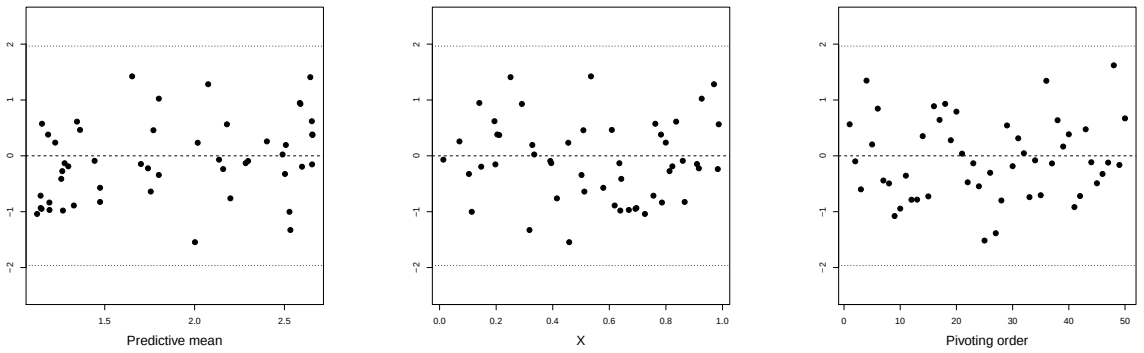
Figure 7.15: Validation diagnostics for the Gaussian approximation of the toy example 7.1 (b) using 100 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach based on the pivoted Cholesky errors: (a) $D^{PC}(\mathbf{z}^{(v)})$ against the pivoting order; (b) QQ plot of $D^{PC}(\mathbf{z}^{(v)})$.

whereas the numerical approximation provided by the composition method leads to a model that represents better the uncertainty about the real process.

In these examples, we show that graphical diagnostics can be obtained when the unknown parameters are numerically integrated over their joint posterior distribution. We show some approximation for the distribution of the diagnostics using the composition sampling method. These diagnostics can also be applied to the Student-t process emulator when the correlation lengths are numerically integrated out using a Monte Carlo method.

7.5 Discussion

In this Chapter, we presented the discrepancy function model when the simulator is cheap to run. We presented some inference methods to predict the real system when our uncertainty about the discrepancy function can be represented by a Gaussian process. The inference methods depend on the way we deal with the unknown quantities (θ, Ψ). We described the plug-in, the Nagy et al., the Kennedy and O'Hagan, and the MCMC approaches, where we gave emphasis to the plug-in and the Nagy et al.'s approaches. Some diagnostics for validating the predictive methods were presented and illustrated for the plug-in and Nagy



(a) (b) (c)

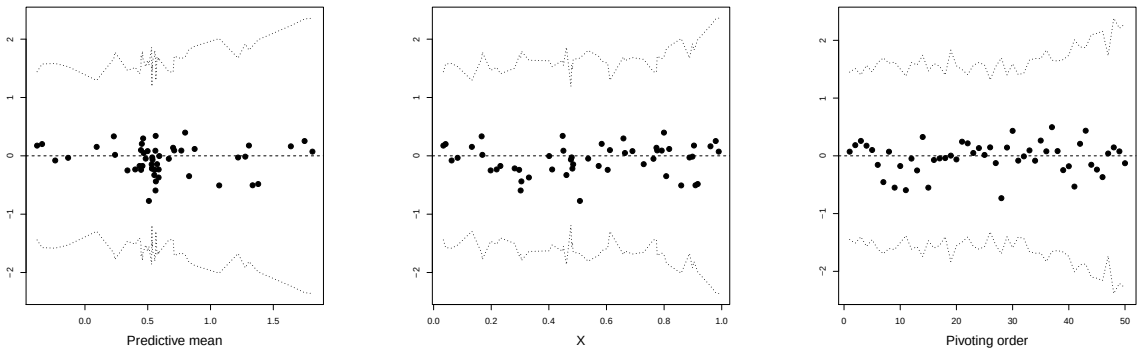
Figure 7.16: Validation diagnostics for the Gaussian approximation of the toy example 7.1 (b) using 180 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach: (a) $D^I(\mathbf{z}^{(v)})$ against the predictive mean; (b) $D^I(\mathbf{z}^{(v)})$ against the validation inputs; (c) pivoted Cholesky errors versus the pivoting order.

et al.'s approaches.

Nagy et al.'s approach depends on the maximum likelihood estimates (MLEs) and an approximation for the Hessian matrix. For large training data sizes, the difference between the predictions using this approach and using plug-in methods will be very small, because the approximated distribution for the parameters will be centred on the MLEs with a little uncertainty. However, when the sample size is small the uncertainty in the parameters will be larger and Nagy et al.'s approach would be more appropriate for dealing with the uncertainty in the predictions.

The proposed diagnostics are very useful tools during the modelling procedure. Based on the results from the diagnostics, we can identify problems such as underconfidence, overconfidence, poor estimation of some unknown parameters. After we identify a problem, the diagnostics may provide information on where we should collect more data in order to make the predictive model a better representation of our beliefs about the real system.

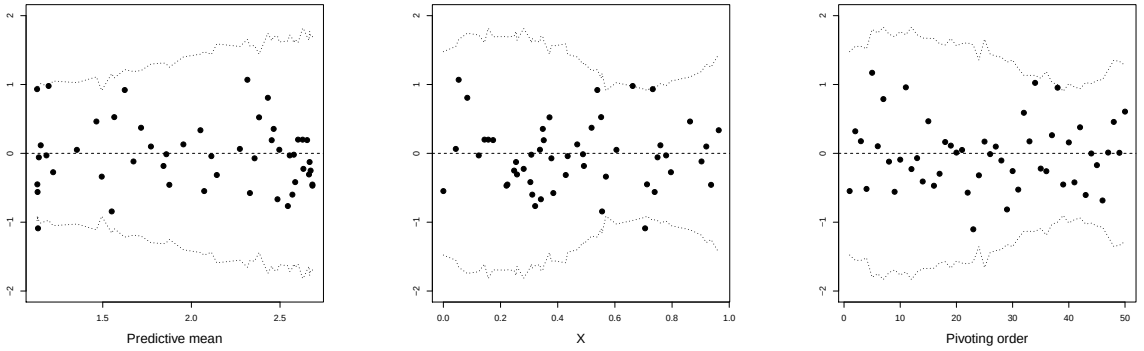
The diagnostics proposed to validate discrepancy function models are similar to those diagnostics proposed to validate Gaussian process emulators (Chapter 4). The innovations here are the diagnostics for the discrepancy function model when numerical approximations are used. The induced distribution of the diagnostics is obtained by either analytical or nu-



(a) (b) (c)

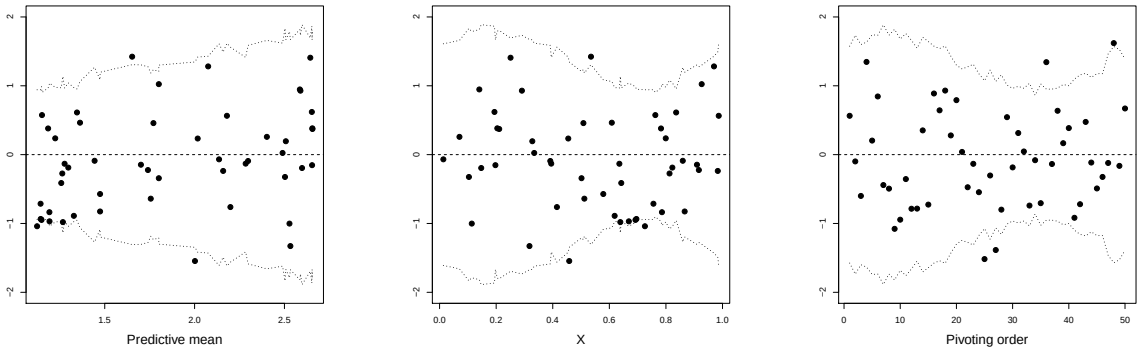
Figure 7.17: Validation diagnostics for the toy example 7.1 (a) using the composition sampling to derive the predictive distribution, 15 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach: (a) $D^I(\mathbf{z}^{(v)})$ against the predictive mean; (b) $D^I(\mathbf{z}^{(v)})$ against the validation inputs; (c) pivoted Cholesky errors versus the pivoting order.

merical approximations. These diagnostics can also be used for validating Gaussian process emulators when the correlation lengths are numerically integrated out.



(a) (b) (c)

Figure 7.18: Validation diagnostics for the toy example 7.1 (b) using sampling procedures to derive the predictive distribution using, 100 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach : (a) $D^I(\mathbf{z}^{(v)})$ against the predictive mean; (b) $D^I(\mathbf{z}^{(v)})$ against the validation inputs; (c) pivoted Cholesky errors versus the pivoting order.



(a) (b) (c)

Figure 7.19: Validation diagnostics for the toy example 7.1 (b) using sampling procedures to derive the predictive distribution using, 180 training data, 50 validation data and (θ, Ψ) sampled via Nagy et al.'s approach : (a) $D^I(\mathbf{z}^{(v)})$ against the predictive mean; (b) $D^I(\mathbf{z}^{(v)})$ against the validation inputs; (c) pivoted Cholesky errors versus the pivoting order.

Chapter 8

Conclusions

8.1 Summary and contributions

The first main contribution of this thesis is to provide a detailed diagnostic analysis to validate scalar output Gaussian process emulators. This is a very important step before using the emulator as surrogate for the simulator, because a non-valid emulator can result in wrong conclusions. The second contribution is to present some designs for validation. Another important contribution of this thesis is to present some statistics that can be used to compare competing emulators. This is due to the proposed validation diagnostics being unable to rank different valid emulators. The final contribution is to extend the diagnostics for validating Gaussian process emulators to validate discrepancy function models, where we presented some diagnostics for validating the discrepancy function model when plug-in estimates are used and when the unknown parameters are integrated out.

The demand for emulators has been increasing considerably, but there has been very little research on validating emulators. We presented a set of numerical and graphical diagnostics to compare the emulator predictions with the simulator outputs at some inputs chosen for validation purposes. Our proposed diagnostics not only provide information about the validity of an emulator, but also guide the modeller in fixing problems when the emulator

is not valid. One important numerical diagnostic is the Mahalanobis distance, because it is a single measure that takes into account the correlation structure of the emulator. Uncorrelated errors are also presented in order to reduce the risk of misinterpretation. The innovation is the use of the pivoted Cholesky errors, where the pivoting order provides an extra interpretation that we can associate with the correlation structure. In section 4.3.4, we presented a set of graphical diagnostics that provide information about the validity of the emulator.

The design for validation problem is defined by choosing the inputs to run the simulator whose outputs are compared with the emulator predictions. Designs for validation were discussed in Section 5.3. The innovation is to develop designs where we choose input points from regions with different uncertainty levels. The proposed designs depend on the availability of the training data. If only the training inputs are available, we suggest distance-based validation designs. When all training data is available, we presented validation designs based on the emulator uncertainty using the predictive variance matrix.

The numerical and graphical diagnostics presented in Chapter 4 provide information about the validity of an emulator. However, they do not provide a measure that can be used to rank competing emulators. In Chapter 6, we discussed some comparison criteria used to rank competing emulators, where we suggest using Bayes factors. Some alternatives are provided based on scoring rules, in case the density function is not available, such as emulators derived using Bayes linear methods or emulators where the correlation lengths were numerically integrated out.

In Chapter 7, we presented diagnostics for discrepancy function models. We modelled the discrepancy function as the difference between the computer model and the real system, where we have calibration parameters associated with the simulator and a variance parameter associated with observation errors. We have shown that using plug-in methods, the diagnostics are the same as those proposed in Chapter 4, but when we numerically integrate the unknown parameters out the diagnostics can be adapted. We proposed, in Section 7.4.2, a set of diagnostics using a sample of the predictive distribution for the emulator.

8.2 Future work

In Chapter 4, we provided diagnostics for single-output Gaussian process emulators. However complex models can have two or more outputs. Hence, a multiple output emulator can be a more appropriate surrogate of a simulator. The proposed diagnostics for single-output emulators can still be applied on the multiple output case if the multi-output emulator is a Student-t process as in Conti and O'Hagan (2007). However, the current diagnostics would consider the vector of outputs evaluated at one particular validation input as a set of different simulation runs. For some numerical diagnostics, such as the Mahalanobis distance, this is not a problem. But for some diagnostics such as the pivoted Cholesky errors, the pivoting order would not have a one-to-one relationship to the validation data, because there is more than one output for the same set of inputs. Thus, we should investigate diagnostic tools for multiple output emulators.

The discrepancy function model is a simplified version of the calibration model proposed by Kennedy and O'Hagan (2001). In the Kennedy and O'Hagan calibration model, the uncertainty about the simulator is represented by a Gaussian process. Therefore, we should have a two-step validation procedure where in the first step we validate the emulator, and in the second step we validate the calibration model. Diagnostics for the Kennedy and O'Hagan model is an area of future research.

The main results about diagnostics for Gaussian process emulators presented in Chapter 4 are published in Bastos and O'Hagan (2009).

Bibliography

AIAA (1998), *Guide for the verification and validation of computational fluid dynamics simulations*, American Institute of Aeronautics and Astronautics, AIAA-G-077-1998.

Aitkin, M. (1991), "Posterior Bayes factors (with discussion)," *Journal of the Royal Statistical Society B*, 53, 111–142.

Andrianakis, Y. (2009), "Parameter estimation and prediction using Gaussian Processes," Tech. Rep. MUCM Technical report 09/05, University of Southampton.

Balci, O. and Sargent, R. G. (1984), "A bibliography on the credibility, assessment and validation of simulation and mathematical models," *Simuletter*, 15, 15–27.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2003), *Hierarchical Modeling and Analysis for Spatial Data*, Monographs on Statistics and Applied Probability, Chapman & Hall/CRC, 1st ed.

Bastos, L. S. and O'Hagan, A. (2009), "Diagnostics for Gaussian process emulators," *Technometrics*, 51, 439–451.

Bates, R., Buck, R., Riccomagno, E., and Wynn, H. (1996), "Experimental design and observation for large systems," *Journal of the Royal Statistical Society B*, 58, 77–94.

Bates, R., Riccomagno, E., Schwabe, R., and Wynn, H. (1998), "The use of lattices in the design of high-dimensional experiments," *IMS Lecture Notes*, 34, 26–35.

Bayarri, M., Berger, J., Kennedy, M., Kottas, T., Paulo, R., Cafeo, J., Lin, C., and Tu,

J. (2005), "Bayesian Validation of a Computer Model for Vehicle Crashworthiness," Tech. Rep. 163, National Institute of Statistical Sciences.

Bayarri, M. J., Berger, J., Paulo, R., Sacks, J., Cafeo, J. A., Cavendish, J., Lin, C. H., and Tu, J. (2007), "A Framework for validation of computer models," *Technometrics*, 49, 138–154.

Bayarri, M. J., Berger, J. O., Higdon, D., Kennedy, M. C., Kottas, A., Paulo, R., Sacks, J., Cafeo, J. A., Cavendish, J., Lin, C. H., , and Tu, J. (2002), "A framework for validation of computer models," Tech. Rep. 128, National Institute of Statistical Sciences.

Belisle, C. J. P. (1992), "Convergence theorems for a class of simulated annealing algorithms on Rd," *Journal of Applied Probability*, 29, 885–895.

Benoît, C. (1924), "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un systàme d'équations linéaires en nombre inférieur à celui des inconnues – Application de la méthode à la résolution d'un systéme défini d'équations linéaires (Procédé du Commandant Cholesky)," *Bulletin Géodésique*, 2, 67–77.

Benson, A. J., Frenk, P. C. S., Baugh, C. M., Cole, S., and Lacey, C. G. (2001), "The clustering evolution of the galaxy distribution," *Monthly Notices of the Royal Astronomical Society*.

Brezinski, C. (2006), "The life and work of André Cholesky," *Numerical Algorithms*, 43, 279–288.

Chapman, W. L., Welch, W. J., Bowman, K. P., Sacks, J., and Walsh, J. E. (1994), "Arctic Sea Ice Variability: Model Sensitivities and a Multidecadal Simulation," *Journal of Geophysical Research*, 99, 919–935.

Conti, S., Gosling, J. P., Oakley, J., and O'Hagan, A. (2009), "Gaussian process emulation of dynamic computer codes," *Biometrika*, 96, 663–676.

Conti, S. and O'Hagan, A. (2007), "Bayesian Emulation of Complex Multi-Output and Dynamic Computer Models," Tech. Rep. Technical report 569/07, The University of Sheffield.

Craig, P. S., Goldstein, M., Rougier, J. C., and Seheult, A. H. (2001), "Bayesian Forecasting for Complex Systems Using Computer Simulators," *Journal of the American Statistical Association*, 96, 717–729.

Cressie, N. (1993), *Statistics for Spatial Data*, New York: J. Wiley.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1988), "A Bayesian Approach to the Design and Analysis of Computer Experiments," Tech. Rep. ORNL-6498, Oak Ridge National Laboratory.

— (1991), "Bayesian Prediction of Deterministic Functions, with Applications to the Design and Analysis of Computer Experiments," *Journal of the American Statistical Association*, 86, 953–963.

Dawid, A. P. (1998), "Coherent Measures of Discrepancy, Uncertainty and Dependence, With Applications to Bayesian Predictive Experimental Design," Tech. Rep. Research Report 139, University College London, Dept. of Statistical Science.

Dawid, A. P. and Sebastiani, P. (1999), "Coherent Dispersion Criteria for Optimal Experimental Design," *The Annals of Statistics*, 27, 65–81.

FDA (2002), *General principles of software validation; Final guidance for Industry and FDA staff*, U.S. Food and Drug Administration.

Fishman, G. S. and Kiviat, P. J. (1968), "The statistics of discrete-event simulation," *Simulation*, 10, 185–195.

Fox, L., Huskey, H. D., and Wilkinson, J. H. (1948), "Notes on the solution of linear algebraic equations," *The Quarterly Journal of Mechanics and Applied Mathematics*, 1, 149–173.

Fraccaro, R., Hyndman, R. J., and Veevers, A. (2000), "Residual diagnostic plots for checking for model mis-specification in time series regression," *Australia and New Zealand Journal of Statistics*, 42, 463–477.

Galanti, S. and Jung, A. (1997), "Low-Discrepancy Sequences: Monte Carlo Simulation of Option Prices," *Journal of Derivatives*, 63–83.

Gentle, J. E. (1998), *Numerical Linear Algebra for Applications in Statistics*, Springer.

Gneiting, T., Baladdaoui, F., and Raftery, A. E. (2007), "Probabilistic Forecasts, Calibration, and Sharpness," *Journal of the Royal Statistical Society B*, 69, 243–268.

Gneiting, T. and Raftery, A. E. (2007), "Stricly Proper Scoring Rules, Prediction, and Estimation," *Journal of the American Statistical Association*, 102, 359–378.

Goldstein, M. and Rougier, J. (2006), "Bayes Linear Calibrated Prediction for Complex Systems," *Journal of the American Statistical Association*, 101, 1132–1143.

Goldstein, M. and Wooff, D. (2007), *Bayes linear statistics: theory and methods*, vol. Volume 635 of Wiley series in probability and statistics, John Wiley and Sons.

Golub, G. H. and van Loan, C. F. (1996), *Matrix computations*, Baltimore: Johns Hopkins University Press, 3rd ed.

Good, I. J. (1952), "Rational Decisions," *Journal of the Royal Statistical Society B*, 14, 107–114.

Gramacy, R. B. and Lee, H. K. H. (2008), "Bayesian Treed Gaussian Process Models With an Application to Computer Modeling," *Journal of the American Statistical Association*, 103, 1119–1130.

Haslett, J. and Hayes, K. (1998), "Residuals for the linear model with general covariance structure," *Journal of the Royal Statistical Society B*, 60, 201–215.

Haylock, R. G. and O'Hagan, A. (1994), "On inference for outputs of computationally expensive algorithms with uncertainty on the inputs," Tech. Rep. 94-18, Department of Mathematics, University of Nottihgham.

Higham, N. J. (2002), *Accuracy and Stability of Numerical Algorithms*, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, second Edition. First edition 1996.

Hills, R. G. and Trucano, T. G. (1999), "Statistical Validation of Engineering and Scientific Models: Background," Tech. Rep. SAND99-1256, Sandia National Laboratory.

— (2001), "Statistical Validation of Engineering and Scientific Models: A Maximum Likelihood Based Metric," Tech. Rep. SAND2001-1783, Sandia National Laboratory.

Houseman, E. A., Ryan, L. M., and Coull, B. A. (2004), "Cholesky Residuals for Assessing Normal Errors in a Linear Model With Correlated Outcomes," *Journal of the American Statistical Association*, 99, 383–394.

IEEE (1991), *IEEE Standard Glossary of Software Engineering Terminology*, The Institute of Electrical and Electronics Engineers, IEEE Std 610.12-1990.

— (1998), *IEEE Standard for Software Verification and Validation*, The Institute of Electrical and Electronics Engineers, IEEE Std 1012-1998.

Jeffreys, H. (1935), "Some tests of significance, treated by the theory of probability," *Proceedings of the Cambridge philosophy society*, 31, 203–222.

— (1961), *Theory of Probability*, Oxford, UK: Oxford University Press, 3rd ed.

Kass, R. E. and Raftery, A. E. (1995), "Bayes Factors," *Journal of the American Statistical Association*, 90, 773–795.

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, 1301–1309.

Kennedy, M. C. and O'Hagan, A. (2000), "Supplementary details on Bayesian calibration of computer codes." Tech. rep., University of Sheffield.

— (2001), "A Bayesian calibration of computer models (with discussion)," *Journal of the Royal Statistical Society B*, 63, 425–464.

Kimeldorf, G. S. and Wahba, G. (1970), "A correspondence between Bayesian estimation on stochastic processes and smoothing by splines," *The Annals of Mathematical Statistics*, 41, 495–502.

Kleijnen, J. P. C. (1995a), "Statistical validation of simulation models," *European Journal of Operational Reasearch*, 82, 21–34.

— (1995b), "Verification and Validation of simulation models," *European Journal of Operational Reasearch*, 82, 145–162.

— (1999), "Validation of Models: Statistical Techniques and Data Availability," in *Winter Simulation Conference*, eds. Farrington, P. A., Nembhard, H. B., Sturrock, D. T., and Evans, G. W., pp. 647–654.

Kleijnen, J. P. C., Bettonvil, B., and Van Groenendaal, W. (1998), "Validation of Trace-Driven simulation models: A novel regression test," *Management Science*, 44, 812–819.

Kozempel, M. F., Tomasula, P., and Craig, J. C. (1995), "The development of the ERRC food process simulator," *Simulation Practice and Theory*, 2, 221–236.

Kuusk, A. (1996), "A computer-efficient plant canopy reflectance model," *Computers and Geosciences*, 22, 149–163.

Lempers, F. (1971), *Posterior Probabilities of Alternative Linear Models*, Rotterdam: University press.

Lindley, D. V. (1980), "Approximate Bayesian Methods," in *Bayesian Statistics*, eds. Bernardo, J. M., DeGroot, M. H., Lindley, D. V., and Smith, A. F. M., Valencia: Univ. Press, pp. 223–237.

Liu, F. and West, M. (2009), "A Dynamic Modelling Strategy for Bayesian Computer Model Emulation," *Bayesian Analysis*, 4, 393–412.

Loeppky, J. L., Sacks, J., and Welch, W. J. (2009), "Choosing the Sample Size of a Computer Experiment: A Practical Guide," *Technometrics*, 51, 366–376.

McGrattan, K. B., Hostikka, S., and Floyd, J. E. (2007), "Fire Dynamics Simulator (Version 5), User's Guide," Nist special publication 1019-5, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239–245.

Morris, M. D. and Mitchell, T. J. (1997), "Exploratory designs for computer experiments," *Journal of Statistical Planinning and Inference*, 43.

Nagy, B., Loeppky, J. L., and Welch, W. J. (2007), "Fast Bayesian Inference for Gaussian Process Models," Tech. Rep. Technical Report #230, The University of British Columbia, Department of Statistics.

Neiderreiter, H. (1992), *Random Number Generation and Quasi-Monte Carlo Methods*, Philadelphia: SAIM.

Nelder, J. A. and Mead, R. (1965), "A simplex algorithm for function minimization," *Computer Journal*, 7, 308–313.

Nilson, T. and Kuusk, A. (1989), "A reflectance model for the homogeneous plant canopy and its inversion," *Remote Sensing of Environment*, 27, 157–167.

Oakley, J. (1999), "Bayesian Uncertainty Analysis for Complex Computer Codes," Ph.D. thesis, The University of Sheffield, Sheffield, UK.

Oakley, J. E. and O'Hagan, A. (2002), "Bayesian inference for the uncertainty distribution of computer model outputs," *Biometrika*, 89, 769–784.

— (2004), "Probabilistic sensitivity analysis of complex models: a Bayesian approach," *Journal of the Royal Statistical Society B*, 66, 751–769.

Oberkampf, W. and Trucano, T. (2000), "Validation Methodology in computational fluid dynamics," Tech. Rep. 2000-2549, American Institute of Aeronautics and Astronautics.

Oberkampf, W. L. and Barone, M. F. (2006), "Measures of agreement between computation and experiment: validation metrics," *Journal of Computational Physics*, 217, 5–36.

O'Hagan, A. (1978), "Curve fitting and optimal design for predictions," *Journal of the Royal Statistical Society B*, 40, 1–42.

— (2006), "Bayesian analysis of computer code outputs: A tutorial," *Reliability Engineering & System Safety*, 91, 1290–1300.

O'Hagan, A., Buck, C. E., Daneshkhah, A., Eiser, J. R., Garthwaite, P. H., Jenkinson, D. J., Oakley, J. E., and Rakow, T. (2006), *Uncertain Judgements: Eliciting Experts' Probabilities*, Wiley.

Paulo, R. (2005), "Default priors for Gaussian processes," *The Annals of Statistics*, 33, 556–582.

Randall, D. A., Wood, R. A., Bony, S., Colman, R., Fichefet, T., Fyfe, J., Kattsov, V., Pitman, A., Shukla, J., Srinivasan, J., Stouffer, R. J., Sumi, A., and Taylor, K. E. (2007), "Climate Models and Their Evaluation," in *Climate Change 2007: the Physical Science Basis*, eds. Solomon, S., Qin, D., Manning, M., Marquis, M., Averyt, K., Tignor, M., Miller, H., and Zhenlin, C., Cambridge, United Kingdom and New York, NY, USA.: Cambridge University Press.

Rasmussen, C. E. and Williams, C. K. I. (2006), *Gaussian Processes for Machine Learning*, The MIT Press.

Rebba, R., Mahadevan, S., and Huang, S. (2006), "Validation and error estimation of computational models," *Reliability Engineering & System Safety*, 91, 1390–1397.

Roache, P. J. (1998), *Verification and Validation in computer science and engineering*, Albuquerque: Hermosa.

Rougier, J. (2008), "Efficient Emulators for Multivariate Deterministic Functions," *Journal of Computational and Graphical Statistics*, 17, 827–843.

Rougier, J., Sexton, D. M. H., Murphy, J. M., and Stainforth, D. (2007), "Emulating the sensitivity of the HadAM3 climate model using ensembles from different but related experiments," Tech. Rep. 07/04, MUCM, The University of Sheffield, being revised for the Journal of Climate.

Sacks, J., Schiller, S. B., and Welch, W. J. (1989a), "Designs for Computer Experiments," *Technometrics*, 31, 41–47.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989b), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409–423.

Saltelli, A., Chan, K., and Scott, M. (eds.) (2000), *Sensitivity Analysis*, New York, USA: Wiley.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003), *The Design and Analysis of Computer Experiments*, New York: Springer-Verlag.

Sargent, R. G. (1979), "Validation of Simulation Models," in *Winter Simulation Conference*, pp. 497–503.

Stein, M. (1987), "Large Sample Properties of Simulations Using Latin Hypercube Sampling," *Technometrics*, 29, 143–151.

Székely, G. J. (2000), "E-Statistics: The Energy of Statistical Samples," Tech. Rep. Technical Report 03-05, Bowling Green State University, Dept. of Mathematics and Statistics.

Taussky, O. and Todd, J. (2006), "Cholesky, Toeplitz and the triangular factorization of symmetric matrices," *Numerical Algorithms*, 41, 197–202.

Trucano, T. G., Swiler, L. P., Igusa, T., Oberkampf, W. L., and Pilch, M. (2006), "Calibration, validation, and sensitivity analysis: What's what," *Reliability Engineering & System Safety*, 91, 1331–1357.

Turing, A. M. (1948), "Rounding-off errors in matrix processes," *The Quarterly Journal of Mechanics and Applied Mathematics*, 1, 287–308.

USDoD (1996), *Verification, Validation, and Accreditation (VV&A) Recommended Practices Guide*, United States Department of Defense, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering.

Wang, S., Chen, W., and Tsui, K.-L. (2009), "Bayesian Validation of Computer Models," *Technometrics*, 51, 439–451.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992), "Screening, predicting, and computer experiments." *Technometrics*, 34, 15–25.

Zickfeld, K., Slawig, T., and Rahmstorf, S. (2004), "A low-order model for the response of the Atlantic thermohaline circulation to climate change," *Ocean Dynamics*, 54, 8–26.