Diagnostics and Simulation-Based Methods for Validating Gaussian Process Emulators

Younus Al-Taweel

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

University of Sheffield

School of Mathematics and Statistics

Supervisor: Jeremy Oakley

March 2018

To my parents and my family.

Acknowledgements

I would like to sincerely thank my supervisor Professor Jeremy Oakley for the support and the encouragement that he has given me during my PhD studies. I am thankful for the help and knowledge he has provided me with to enable me to complete my PhD thesis. This thesis could not have been completed without his generous and professional assistance. I also wish to express my sincere thanks to my advisor Dr. Kevin Walters for his guidance and support. I would like also to thank the staff and postgraduate students of the Mathematics and Statistics department for being helpful and friendly at all times. I especially thank my mother, brothers and sisters for their support, prayers and for their entire love. Many thanks to my wife and my family who support me at all times. I would like to thank my uncles, Aqeel and Abdulrahman, and my cousin Naktal. Finally, I would like to also thank all my friends in particular Arqam for being supportive, and providing me with the friendship that I needed during my studies.

I also acknowledge The Higher Committee For Education Development in Iraq for funding of my studentship research and thank all its members for their support. I would also like to thank the department of Mathematics, the College of Education for Pure Science, the Mosul University, the Ministry of Higher Education and Scientific Research in Iraq for giving me the opportunity to obtain the PhD degree.

Abstract

Emulation is a statistical technique that can be utilised for estimating model simulations when the computer models are too computationally expensive to run. Emulators need to be subjected to a validation process since various assumptions have to be made. One assumption is that the computer model output is thought of as a realization of a Gaussian process with a mean and a covariance function. The computer model, however, is not a random sample from the Gaussian process distribution. In this thesis, we develop a graphical diagnostic that can be used to investigate whether the Gaussian process assumption is suitable for building emulators.

Diagnostic methods can be used to assess the validity of the statistical model in order to investigate the best probability model for describing the computer model. However, it is not always possible to derive the required reference distribution for some diagnostics analytically. In this thesis, a simulation-based method is developed based on simulating samples from the posterior distribution of the output function. This simulation-based method can be used to obtain the reference distribution of diagnostics that cannot be obtained analytically. The observed diagnostic values will be 'consistent' with the simulated diagnostic values if the Gaussian process emulator is valid.

Contents

Chapter 1

Introduction

1.1 Computer models

In recent years, computer experiments have increasingly been used as replacements for physical experiments which are considered impractical, impossible or too costly. In order to perform a computer experiment, it is necessary to construct a computer model, called a simulator, which is a mathematical representation of the real system and is usually implemented on a computer. We consider a deterministic simulator, i.e. one that will produce the same outputs if it is run at the same inputs. The process of running the simulator at different input values is known as a computer experiment. Mathematically, the simulator is referred to as a function, $y = f(\mathbf{x})$ for $\mathbf{x} \in \chi \subset \mathbb{R}^p$, where $\mathbf{x} = (x_1, \ldots, x_p)$ is a vector of inputs and the output is a scalar, $y \in \mathbb{R}$. Typically, simulators will have multiple outputs but in this thesis we focus on single output simulators. Some simulators may also be stochastic, but we do not consider this here.

1.2 Applications of computer models

Computer models have widely been used to investigate the systems of the real world in most science and technology fields. For example, in engineering, Amiri (2012) used the CMG STARS reservoir simulator (Computer Modelling Grup) to simulate low-frequency electrical heating for different models. This simulator models multicomponent thermal flow under electrical heating. Vitsas (2016) used a commercial flight simulator, called *X-Plane* Flight Simulator 10. This simulator gives the user the opportunity for flying different military, commercial and unconventional experimental aircraft over global scenery which covers most geographical areas on the Earth.

In climate and environment, the pCNEM (probabilistic Canadian NAAQS (National Ambient Air Quality Standards) Exposure Model) simulator was developed by Zidek et al. (2005). The pCNEM simulator generates a pollutant concentrations sequence to which a randomly chosen person is exposed over time. Crookston et al. (2010) modified a Climate Forest Vegetation Simulator (Climate-FVS) that provides a useful tool for forest managers. The Climate-FVS model incorporates the potential impacts of climate change in forest plans. The Climate-FVS model also simulates the potential impacts of climate change on various climatically diverse landscapes.

Spracklen et al. (2005) developed the Global Model of Aerosol Processes (GLOMAP) to be an extension of a chemical transport model. The GLOMAP generates the evolution of the global aerosol size distribution. This model also involves the processes of aerosol nucleation, condensation, wet and dry deposition and cloud processing. Emanuel (2002) constructed a relatively simple climate simulator consisting of a single ocean layer and a two-layer atmosphere. This simple climate simulator is able to produce multiple overlapping stable equilibrium states based on a few feedback processes.

Weaver et al. (2001) developed the University of Victoria (UVic) Earth System Climate Model consisting of an energy-moisture balance atmospheric model, a dynamic-thermodynamic sea-ice model and an ocean general circulation model. This model can capture the pathways of bottom, deep and intermediate waters as revealed via simulations which involve the passive tracers release. The model also generates routes of the warm and cold water by returning the upper layer water to the Atlantic Ocean to compensate for North Atlantic Deep Water production and export.

In health, Jandarov et al. (2014) developed an epidemic model that involves a formulation for the spatial transmission among various host cities. This model describes spatiotemporal patterns of epidemics and can accommodate small population sizes and disease recolonization. In population, Baggaley et al. (2012b) considered a wavefront model for the spread of Neolithic culture across Europe. The wavefront model allows both an isotropic background spread that incorporates the impacts of local geography, and a localized anisotropic spread connected with major waterways.

1.3 The need for surrogate models

Computer models can be computationally very expensive to run. This means that it can take many hours or even several days to return a value of y at a single of $\mathbf{x}$. Therefore, the simulator can only be run at a limited number of inputs. The computationally expensive problem can be caused, for example, by the simulator being very complex or a high degree of precision being required.

Gaussian process emulators have been widely used as surrogates of computer models in many fields of science and technology. The first use of Gaussian process emulators as surrogates of computer models was by Sacks et al. (1989b). They present a description of how statistical inference can be used in computer modelling for estimating simulators. Currin et al. (1991) developed the concept of emulators under a Bayesian framework. Gaussian process emulators can be used not only to provide approximations for computationally expensive computer models, but also to provide a probability distribution for the computer models. This probability distribution can be then used to run any subsequent analysis of the computer models.

In certain situations, the statistical assumptions that are made in building Gaussian process emulators may not be precisely satisfied. If the assumptions do not hold, the results that depend on emulators will not be accurate. Hence, Gaussian process emulators are required to be subjected to a validation process using appropriate diagnostics. Some diagnostic methods are just based on the differences between emulator predictions and the simulator outputs. Other diagnostic methods consider the uncertainty in the emulator predictions. Bastos and O'Hagan (2009) propose a number of numerical and graphical diagnostic methods that take into account uncertainty in the emulator predictions. Their diagnostics are based on comparisons between the validation outputs of the simulator and the emulator predictions. Bastos and O'Hagan (2009) propose comparing the observed value of the diagnostic with its distribution.

In this thesis, we aim to extend the work that is given by Bastos and O'Hagan (2009) for diagnostic methods that consider the uncertainty in the emulator predictions. We evaluate the performance of current diagnostic methods for examining assumptions that are made in constructing Gaussian process emulators. We apply these diagnostic methods on emulators that are built on different simulators that have different behaviours. We also present a modification of an existing graphical diagnostic method which makes the diagnostic more informative. We develop a graphical diagnostic method that tests coverage properties of Gaussian process emulators. Another contribution is developing a simulation-based method that can be used to obtain the distribution of any diagnostic and it may be applied when the diagnostic distribution cannot be found analytically.

1.4 Outline of the thesis

The focus of this thesis lies in diagnostic methods for examining assumptions that are made in building Gaussian process emulators. The thesis consists of six chapters:

- In Chapter 2, we review literature on applications of Gaussian process emulators in several areas of science. The aim is to see what choices authors have made for the mean and the covariance functions when building Gaussian process emulators. Furthermore, we want to find the popular designs for generating the design points, the most popular methods that authors used for estimating the correlation length parameters, and the most popular method that authors prefer to use for validating their emulators: the cross-validation methods or separate validation sets. In addition, we aim to investigate what diagnostic methods have been used for validating Gaussian process emulators. The processes of constructing emulators, as surrogates of simulators, under a Bayesian framework are reviewed. We also review several methods used for estimating the correlation parameters in the correlation function. Then, several designs for training and validation inputs that have been used in building Gaussian process emulators are reviewed.

- In Chapter 3, the concept of uncertainty calibration in Gaussian process emulators is described in terms of overconfidence and underconfidence of emulators. We also present the concept of diagnostics and develop a modification of an existing diagnostic that makes the diagnostic more informative. We investigate the performance of a number of current diagnostic methods for examining assumptions that are made in constructing Gaussian process emulators.

- In Chapter 4, we focus on the development of a graphical diagnostic method that examines coverage properties of Gaussian process emulators. In addition, since the distribution of some diagnostic methods cannot be derived analytically, we develop a method, called simulation-based method, that can be used to obtain the distribution of any diagnostic. We also investigate the performance of our diagnostics with data that have different properties to that of a Gaussian process.

- Stationary Gaussian process emulators have been used widely in the literature. However, the stationary assumption for building Gaussian process emulators may not be suitable for functions that show discontinuity or nonstationary in their behaviour. In Chapter 5, two types of ‘advanced’ Gaussian process emulators that were used for dealing with nonstationary functions are reviewed. These advanced Gaussian process emulators are built under a Bayesian framework. It is necessary to test whether the Gaussian process assumption is suitable for building these advanced Gaussian process emulators or not. We also investigate the performance of diagnostic methods for these advanced Gaussian process emulators.

- Finally, the conclusion and recommendations of this study as well as ideas for future research relevant to this thesis are discussed in Chapter 6.

Chapter 2

Statistical inference for complex simulators using emulators

2.1 Introduction

In this chapter, we review a well-known statistical method, called emulation, for tackling the problem of computationally expensive simulators and predicting outputs of simulators. We explain the concept of emulators and present the processes of building emulators under a Bayesian framework. We present various methods for estimating the correlation parameters in the correlation function. We also review some designs for choosing the training and validation inputs. We also present literature search for Gaussian process emulators. Finally, we consider an example as an illustration of Gaussian process emulators.

2.2 Emulators

Statistical inference can be used in computer modelling to provide approximations of simulators. The simulator is a function, $f(\cdot)$, where for an input $\mathbf{x} = (x_1, \ldots, x_p) \in \chi \subset \mathbb{R}^p$, the output is $y = f(\mathbf{x})$. Suppose we have a set of inputs, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and evaluations $\mathbf{y} = \{y_1 = f(\mathbf{x}_1), \ldots, y_n = f(\mathbf{x}_n)\}$ of the simulator outputs at these inputs. Now, suppose we wish to run the simulator at more values of inputs, but cannot do so because the simulator can be computational very expense to run. Thus, we wish to make joint inferences about a set of simulator outputs $y_{n+1} = f(\mathbf{x}_{n+1}), y_{n+2} = f(\mathbf{x}_{n+2}), \ldots$ given the available evaluations of simulator outputs, $\mathbf{y}$.

We consider $f(\cdot)$ to be an uncertain function, in that the simulator output $f(\mathbf{x})$ is unknown before running the simulator at the untried input value $\mathbf{x}$. Then, we can use a Bayesian perspective to construct a probability distribution for $f(\cdot)$ based on $\mathbf{y}$, and quantify the uncertainty about $f(\cdot)$ due to running the simulator at a limited number of inputs. We call this probability distribution an emulator which can be then used to run any subsequent analysis of the simulator. In fact, this problem can be entirely perceived as a statistical regression problem and any regression model can be used for constructing emulators.

The idea of emulators was proposed by Sacks et al. (1989b) and Currin et al. (1991) developed the concept of emulators using a Bayesian framework. Emulators have been used widely in various applications in most science and technology fields. In order to find applications of Gaussian process emulators, we searched the Web of Science (formerly Web of Knowledge) under the terms ‘Gaussian process emulator’, ‘Gaussian process metamodel’, ‘Gaussian process model’ and ‘Gaussian process regression’ according to the following strategy:

1. We used the quotation marks to make our search more accurate.

2. As of 17/10/2017, we found about 1230 studies under the term 'Gaussian process regression' and 299 studies under the term 'Gaussian process model'. We read the titles and abstracts of some papers that were thought to be relevant to our topic.

3. We found about 71 studies under the terms 'Gaussian process metamodel' and 'Gaussian process emulator'. We focused on papers with mostly different authors in order to see difference between methods and applications, because some authors may use the same methodology in their papers. For example, some authors used the same forms of the mean and the covariance functions, the same design and the same method for estimating the correlation parameters in their publications.

4. We also added some other papers that were referred to in some of these studies because they contain some detail and some of them used different methodology.

The aim of this search is to find what choices authors have made when building Gaussian process emulators. For example, what they have used for the mean and the covariance functions [1] and what are the most popular forms of these functions. Furthermore, we want to find the designs that have been used for generating the design points, for example, the Latin hypercube design (LHD) and the sliced Latin hypercube design (SLHD). Also, we want to find the most popular methods that authors used for estimating the correlation length parameters such as the maximum likelihood estimate (MLE) and a Markov Chain Monte Carlo (MCMC) algorithm. In addition, we want to see which method that authors prefer to use

[1]We will define these and the subsequent terms in Section 2.3 and the subsequent sections.

for validating their emulators: the cross-validation methods or separate validation sets. Finally, we aim to find diagnostic methods that have been used for validating their emulators. Some authors use diagnostics that depend only on the difference between the simulator outputs and the emulator predictions such as the mean squared error (MSE), the root mean squared error (RMSE), the standardised root mean squared error (SRMSE) and the predictivity coefficient (Q^2). Some other authors use diagnostic methods that consider the uncertainty in the emulator predictions such as the Mahalanobis distance and the individual standardised errors. Table 2.1 presents some of these applications with some detail.

Table 2.1: Applications of Gaussian process emulators.

No.	**Authors**	**Simulator**	**No. of inputs**	**No. of design points**	**Mean function**	**Correlation function**	**Design**	**Correlation parameter estimate**	**Validation Method**	**Validation**
1-	Kennedy and O'Hagan (2001)	nuclear release model	2	25	linear in inputs	squared exponential	LHD	MLE	None	None
2-	Oakley and O'Hagan (2004)	oil-field model	40	101	linear in inputs	squared exponential	LHD	MLE	None	None
3-	Bayarri et al. (2007)	spot welding model	4	26	constant	squared exponential	maximin LHD	MLE	None	None
4-	Kolachalama et al. (2007)	maximal wall shear stress model	14	100	constant	squared exponential	LHD	MLE	cross-validation	individual standardised errors
5-	Rojnik and Naveršnik (2008)	health economic model	28 and 30	400	GEM-SA default	GEM-SA default	maximin LHD	GEM-SA default	10000 validation points	plot of predictions against true values
6-	Petropoulos et al. (2009)	soil vegetation atmospheric transfer model	30	400	GEM-SA default	GEM-SA default	LP-τ	GEM-SA default	cross-validation	RMSE
7-	Johnson et al. (2011)	milk contamination model	3	20	linear in inputs	squared exponential	maximin LHD and Sobol sequence	MLE	10 validation points	pivoted Cholesky errors
8-	Lee et al. (2011)	complex global aerosol model	8	80	constant and linear in inputs	squared exponential	maximin LHD	GEM-SA default	24 validation points	plot of predictions against true values with 95% credible

Continued on Next Page. . .

Table 2.1 – Continued

No.	Authors	Simulator	No. of inputs	No. of design points	Mean function	Correlation function	Design	Correlation parameter estimate	Validation Method	Validation
										intervals
9-	Ba and Joseph (2012)	heat exchanger model	4	64	constant	squared exponential	LHD	MLE	14 validation points	RMSE
10-	Baggaley et al. (2012a)	wavefront model	3	200	linear in inputs	squared exponential	maximin LHD	MCMC	100 validation points	Mahalanobis distance
11-	Gramacy and Lee (2012)	rocket booster model	3	3041	linear in inputs	squared exponential	Cartesian product	MCMC	multifold cross-validation	plot of predictions with 95% credible intervals
12-	Miller et al. (2012)	potential surface model	3	349	constant	squared exponential	maximin LHD	MLE	None	None
13-	Olson et al. (2012)	University of Victoria Earth system climate model	3	250	linear in inputs	squared exponential	uniform	fixed value	cross-validation	plot of predictions against true values
14-	Sham Bhat et al. (2012)	ocean tracer model	6	988	linear in inputs	squared exponential	space-filling	MLE	cross-validation	plot of predictions against true values
15-	Tokmakian et al. (2012)	C-GOLDSTEIN climate model	16	96	linear in inputs	squared exponential	LHD	MLE	100 validation points	individual standardised errors
16-	Bijak et al. (2013)	semi-	3	343	linear in	squared	Cartesian	GEM-SA	cross-	SRMSE

Continued on Next Page…

Table 2.1 – Continued

No.	**Authors**	**Simulator**	**No. of inputs**	**No. of design points**	**Mean function**	**Correlation function**	**Design**	**Correlation parameter estimate**	**Validation Method**	**Validation**
		artificial population model			inputs	exponential	product	default	validation	
17-	Sergienko et al. (2013)	CO_2 storage reservoir model	3	30	constant	squared exponential	maximin LHD	MLE	cross-validation	Q^2 and RMSE
18-	Bounceur et al. (2014)	ocean atmosphere vegetation model	3	27	linear in inputs	squared exponential	algorithm based on LHD	MLE	cross-validation	plot of 66%, 95% and 99% credible intervals for predictions
19-	Jandarov et al. (2014)	gravity model	4	20	linear in inputs	squared exponential	uniform	MLE	None	None
20-	Novak et al. (2014)	heavy ion collisions model	6	729	constant	power exponential	LHD	MLE	32 validation points	emulator error
21-	Sexton and Everingham (2014)	sugarcane growth model	14	320	linear in inputs	squared exponential	Maximin LHD	GEM-SA default	80 validation points	SRMSE
22-	Wan et al. (2014)	bridge model	6	300	constant	squared exponential	Sobol sequence	MLE	cross-validation	individual standardised errors and QQ-plot
23-	Andrianakis et al. (2015)	infectious disease model	22	220	linear in inputs	Matérn	maximin LHD	MLE	20 validation points	Mahalanobis distance

Continued on Next Page…

Table 2.1 – Continued

No.	Authors	Simulator	No. of inputs	No. of design points	Mean function	Correlation function	Design	Correlation parameter estimate	Validation Method	Validation
24-	Barbillon et al. (2015)	Michaelis-Menten pharmacokinetic model	3	25,50 and 100	linear in inputs	squared exponential	minimax space-filling	MLE	None	None
25-	Chang et al. (2015)	cardiac cell model	6	200	linear in inputs	squared exponential	LHD	MLE	20 validation points	Mahalanobis distance and individual standardised errors
26-	Katurji et al. (2015)	fire emissions production model	3 and 4	432	package	package	package	package	100 validation points	"relative bias"
27-	Lan et al. (2015)	oil reservoir model	9	1000	linear in inputs	squared exponential	MICE	MLE	None	None
28 -	Marrel et al. (2015)	total instantaneous blockage model	27	750	linear in inputs	Matérn	LHD	MLE	cross-validation	Q^2
29-	McDonnell et al. (2015)	nuclear masses model	12	200	constant	squared exponential	LHD	MLE	17 validation points	plot of predictions with 90% credible intervals
30-	Xing et al. (2015)	subsurface flow in a porous medium model	2	80	constant	squared exponential	Sobol sequence	MLE	145 and 227 validation points	relative squared error

Continued on Next Page…

Table 2.1 – Continued

No.	Authors	Simulator	No. of inputs	No. of design points	Mean function	Correlation function	Design	Correlation parameter estimate	Validation Method	Validation
31-	Bowman and Woods (2016)	atmospheric dispersion model	2	80	constant	squared exponential	maximin LHD	minimise RMSE	35 validation points	RMSE
32-	Chang et al. (2016)	Ice Sheet model	10	499	constant	squared exponential	LHD	MLE	multifold cross-validation	plot of predictions against true values
33-	Gómez-Dans et al. (2016)	Soil-Leaf-Canopy radiative transfer model	10	300	constant	squared exponential	LHD	MLE	1000 validation points	RMSE and maximum absolute error
34-	Gu et al. (2016)	TITAN2D model	4	2048 and 50	linear in inputs	power exponential	maximin LHD	posterior mode	633 validation points	MSE
35-	Han and Yong Tan (2016)	Design of a chemical cyclone model	7	100	constant	squared exponential	LHD	MLE	None	None
36-	Le Gratiet et al. (2016)	elastic truss structure model	10	100	constant	Matérn	LHD	cross-validation	10000 cross-validation	Q^2
37-	Liu and Guillas (2016)	tsunami model	5	360	linear in inputs	squared exponential	LHD	MLE	multifold cross-validation	SRMSE
38-	Li et al. (2016)	dynamic building energy model	11	1200	linear in inputs	squared exponential	LHD	MCMC	1200 validation points	RMSE

Continued on Next Page…

Table 2.1 – Continued

No.	Authors	Simulator	No. of inputs	No. of design points	Mean function	Correlation function	Design	Correlation parameter estimate	Validation Method	Validation
39-	Machac et al. (2016)	urban hydrodynamic drainage model	7	40	linear in inputs	squared exponential	LHD	MLE	1000 validation points	RMSE
40-	Montagna and Tokdar (2016)	rocket booster model	3	20	linear in inputs	squared exponential	LHD	MCMC	900 validation points	RMSE
41-	Overstall and Woods (2016)	humanitarian relief mission model	13	120	constant and linear in inputs	squared exponential	SLHD	posterior mode	120 validation points	root relative mean squared error, RMSE and QQ-plots
42-	Sarkar et al. (2016)	wave energy converter model	4	800	constant	squared exponential	LHD	MLE	cross-validation	plot of predictions against true values
43-	Zhang et al. (2016)	two-delay blowfly model	11	20000	constant	squared exponential	maximin LHD	posterior mode	20000 validation points	SRMSE
44-	Kim and Park (2017)	double glazing system model	8	500	constant	squared exponential	LHD	MLE and MCMC	multifold cross-validation	RMSE and mean absolute error
45-	Oakley and Youngman (2017)	natural history model	25	2000	constant	squared exponential	maximin LHD	MLE	None	None

Note that Katurji et al. (2015) used a package to construct a Gaussian process emulator but they did not provide details about building their Gaussian process emulator and they referred to Oakley and O'Hagan (2004). Rojnik and Naveršnik (2008), Petropoulos et al. (2009), Lee et al. (2011), Bijak et al. (2013) and Sexton and Everingham (2014) used GEM-SA software and they did not provide all the detail about their Gaussian process emulators. The GEM-SA software, written by Marc Kennedy but no longer available, is Gaussian Emulation Machine for Sensitivity Analysis. This software allows the user to build an emulator of a computer model and performs uncertainty and sensitivity analyses of the model.

We noticed from this search that there was an increase of the use of Gaussian process emulators for computer model calibrations in the last few years. Calibration involves inferring a set of values of unknown inputs such that the observed data fit to the corresponding outputs of the computer model as closely as possible. The inferred set of values are treated estimates of the unknown inputs and can be used to run any subsequent analysis.

Calibration methods require running the simulator at different input values which may become impractical for expensive simulators. Gaussian process emulators can be used as surrogates of simulators. For example, Chang et al. (2016) constructed a calibration of ice-sheet model depends on a reduced dimensional emulator. They approximated their model by a Gaussian process emulator. Then, they inferred the model parameters using the data and based on the emulator. Oakley and Youngman (2017) constructed a Gaussian process emulator for the likelihood function for calibrating a moderately computationally expensive simulator of a physical process to data from the process to find values of the model inputs.

2.3 Gaussian process emulators

This section reviews Gaussian process regression, which has become very common for building emulators. We first describe a Gaussian process and then explain how emulators can be constructed from a Bayesian perspective.

2.3.1 Gaussian process

A Gaussian process is defined as an infinite collection of random variables, any finite number of which have a joint Gaussian distribution. Mathematically, if the joint distribution of $\mathbf{y} = \{y_1 = f(\mathbf{x}_1), \dots, y_n = f(\mathbf{x}_n)\}$, for every $(n = 1, 2, \dots)$, has a multivariate normal distribution for all $\mathbf{x}_1, \dots, \mathbf{x}_n$, then $f(\cdot)$ has a Gaussian process distribution. This property makes the Gaussian process popular in modelling computer models due to its flexible structure which can adapt to complex functions between inputs and outputs as well as being mathematically tractable. The Gaussian process is specified by its mean function, $m(\cdot)$, and covariance function, $V(\cdot,\cdot)$. Thus, the Gaussian process can be written as

$$f(\cdot) \sim GP(m(\cdot), V(\cdot,\cdot)).$$

For example, if we have two input values, $\mathbf{x}_1$ and $\mathbf{x}_2$, we write

$$\begin{pmatrix} \mathrm{E}[f(\mathbf{x}_1)] \\ \mathrm{E}[f(\mathbf{x}_2)] \end{pmatrix} = \begin{pmatrix} m(\mathbf{x}_1) \\ m(\mathbf{x}_2) \end{pmatrix},$$

$$\begin{pmatrix} \mathrm{Var}[f(\mathbf{x}_1)] & \mathrm{Cov}[f(\mathbf{x}_1), f(\mathbf{x}_2)] \\ \mathrm{Cov}[f(\mathbf{x}_2), f(\mathbf{x}_1)] & \mathrm{Var}[f(\mathbf{x}_2)] \end{pmatrix} = \begin{pmatrix} V(\mathbf{x}_1, \mathbf{x}_1) & V(\mathbf{x}_1, \mathbf{x}_2) \\ V(\mathbf{x}_2, \mathbf{x}_1) & V(\mathbf{x}_2, \mathbf{x}_2) \end{pmatrix}.$$

The mean function can be specified by various forms. The linear form, however, is more usual and convenient in terms of simplifying the subsequent steps in building the emulator. We review here the linear mean function in some detail.

The linear mean function: The linear form of the mean function is given by

$$m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}, \tag{2.3.1}$$

where $\boldsymbol{\beta}$ is a vector of unknown coefficients parameters that will be inferred from the data and $\mathbf{h}(\cdot) : \chi \subset \mathbb{R}^p \longrightarrow \mathbb{R}^q$ is a vector of q known regression functions of $\mathbf{x}$, which describe the global trend of the simulator's behaviour. The linear form of the mean function was suggested by O'Hagan (1978), where he described how the inference about $\boldsymbol{\beta}$ is made analytically. The choice of $\mathbf{h}(\cdot)$ is arbitrary, although it should be selected to incorporate prior belief about the form of the simulator. The regression functions, $\mathbf{h}(\cdot)$, can be chosen in different forms:

1. One of the simplest cases of the regression function is when $q = 1$ and $\mathbf{h}(\mathbf{x}) = 1$ for all $\mathbf{x}$. In this case, the mean function is equal to the coefficient parameter, $\boldsymbol{\beta}$, and represents an unknown overall mean for the output. This means that there is a prior expectation that there is no trend in how the output will respond to the variation in the inputs.

2. Another simple case of the regression function is when $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$. In this case, q is equal to $1 + p$ where p is the number of input variables. The mean function in this case is $m(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$, which represents a prior expectation that the output of the simulator will exhibit a linear trend in response to each input variable.

3. The quadratic form or higher polynomial terms could be a choice for the regression function if nonlinearity is our prior expectation about the response.

Table 2.1 in Section 2.2 shows 24 authors used the linear form in inputs of the mean function and 20 of authors used a constant mean function. Lee et al. (2011) used the constant mean function, $m(\mathbf{x}) = \beta$, and the linear mean function,

$m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ to a construct Gaussian process emulator for global aerosol model and they found a minor difference in the results.

In order to choose the most convenient form of $\mathbf{h}(\cdot)$, Vernon et al. (2010) used backwards stepwise elimination. They considered a polynomial in the mean function terms. Before discarding an input variable, they fitted a third order polynomial to see the set of active variables, which is the most influence set of inputs on the outputs, based on the adjusted R^2.

2.3.2 The covariance functions

The covariance function between the simulator outputs is written as

$$V(\mathbf{x}, \mathbf{x}') = \sigma^2 C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}), \tag{2.3.2}$$

where σ^2 is an unknown parameter that represents the overall variance and $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$ is a known correlation function with a vector of unknown correlation parameters $\boldsymbol{\delta}$. The correlation function must satisfy the property that, $C(\mathbf{x}, \mathbf{x}; \boldsymbol{\delta}) = 1$. Furthermore, the correlation function must be chosen in such that the covariance matrix of any set of outputs is positive semi-definite.

In many studies, an assumption of a separable covariance function is made, where the correlation function is the product of p one-dimensional correlation functions given by

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \prod_{i=1}^{p} C_i(x_i, x_i'; \delta_i). \tag{2.3.3}$$

The correlation function is a crucial part of the Gaussian process and has an essential role in building Gaussian process emulators, where its choice can have a great effect on the emulator predictions, especially with small sample sizes. In this section, we present some popular forms of the correlation function that have been used in computer experiments.

The squared exponential correlation function, also called the Gaussian correlation function, is the most popular and convenient choice in terms of building subsequent steps in Gaussian process emulators. Its form is given by

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \exp\left\{ -\sum_{i=1}^{p} \left(\frac{x_i - x_i'}{\delta_i} \right)^2 \right\}, \tag{2.3.4}$$

where $\delta_i > 0$ are unknown correlation length parameters. Large values of δ_i suggest that the output is a smooth function of the i-th input while small values indicate highly nonlinearity.

The power exponential correlation function is a generalization of the squared exponential correlation function and it has been used in the literature in building Gaussian process emulators. The form of the power exponential correlation function is given by

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \exp\left\{ -\sum_{i=1}^{p} \left| \frac{(x_i - x_i')}{\delta_i} \right|^{\gamma} \right\}, \tag{2.3.5}$$

where $\delta_i > 0$ are the correlation length parameters and $\gamma \in (0, 2]$ is the power parameter. The power exponential correlation function will be the squared exponential correlation function if $\gamma = 2$. In this case it will be infinitely differentiable with respect to x_i. The power exponential correlation function will be differentiable only once if the power parameter, γ, lies in the interval $(1, 2)$. In contrast, if $\gamma \leq 1$, then it will not be differentiable at all.

The Matérn correlation function is another choice of the correlation function which is given by

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) = \prod_{i=1}^{p} \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x_i - x_i'|}{\delta_i} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}|x_i - x_i'|}{\delta_i} \right), \tag{2.3.6}$$

where ν and δ_i are positive parameters, K_ν is a modified Bessel function. The parameters δ_i are the correlation length parameter and ν is the smoothness parameter. The smoothness parameter, ν, controls the existence of the simulator derivatives and it behaves like the power parameter in the power exponential correlation function.

The most popular cases of the Matérn correlation function in computer experiments are when the smoothness parameter is $\nu = 3/2$ and $\nu = 5/2$:

$$\begin{aligned}
C_{\nu=3/2}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) &= \prod_{i=1}^{p} \left(1 + \frac{\sqrt{3}|x_i - x_i'|}{\delta_i}\right) \exp\left(-\frac{\sqrt{3}|x_i - x_i'|}{\delta_i}\right), \\
C_{\nu=5/2}(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) &= \prod_{i=1}^{p} \left(1 + \frac{\sqrt{5}|x_i - x_i'|}{\delta_i} + \frac{5|x_i - x_i'|^2}{3\delta_i^2}\right) \exp\left(-\frac{\sqrt{5}|x_i - x_i'|}{\delta_i}\right).
\end{aligned}$$

If $\nu \to \infty$, the Matérn correlation function will be the squared exponential correlation function (Rasmussen and Williams, 2006).

Table 2.1 in Section 2.2 shows that the squared exponential correlation function was used by 37 authors. This may be because it has been extensively used in literature as well as it expresses the smoothness of Gaussian process as a function of the inputs. Some authors also used the squared exponential correlation function because it is infinitely differentiable which makes it convenient for using Gaussian process to model $f(\mathbf{x})$ and its derivatives. Table 2.1 also shows the power exponential correlation function was used twice by , Novak et al. (2014) and Gu et al. (2016) with power $\gamma \in [1, 2]$. Table 2.1 also shows the Matérn correlation function was used in three of the selected papers.

Nonstationary covariance function: In building Gaussian process emulators, it is very common to use a stationary correlation function, which means that the correlation function $C(\mathbf{x}, \mathbf{x}')$ satisfies the condition

$$C(\mathbf{x}, \mathbf{x}') = C(\mathbf{x} + \mathbf{a}, \mathbf{x}' + \mathbf{a}), \tag{2.3.7}$$

for any $\mathbf{a} \in \mathbb{R}^p$.

The covariance structure can be extended to be a nonstationary function, which allows the model to adapt to functions whose smoothness varies with the inputs. Paciorek and Schervish (2003) introduced a nonstationary correlation function

$$C(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\delta}) = \int_{\mathbb{R}^2} K_{\mathbf{x}_i}(u) K_{\mathbf{x}_j}(u) du, \tag{2.3.8}$$

where $\mathbf{x}_i, \mathbf{x}_j$ and u are locations in $\mathbb{R}^2$ and $K_{\mathbf{x}}(\cdot)$ is a kernel function. They proposed a particular example of a nonstationary correlation function given by

$$C(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\delta}) = |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} |(\Sigma_i + \Sigma_j)/2|^{\frac{-1}{2}} \exp(-Q_{ij}) \tag{2.3.9}$$

with

$$Q_{ij} = (\mathbf{x}_i - \mathbf{x}_i)^T ((\Sigma_i + \Sigma_j)/2)^{-1} (\mathbf{x}_i - \mathbf{x}_i), \tag{2.3.10}$$

where Σ_i and Σ_j are the correlation matrices at $\mathbf{x}_i$ and $\mathbf{x}_j$.

Nonstationary variance: The assumption of the constant variance, σ^2, in the covariance function can be relaxed, where we can incorporate a variance model, $\sigma^2(\mathbf{x})$, in the covariance matrix rather than σ^2. The variance model, $\sigma^2(\mathbf{x})$, quantifies the change of local variability. Ba and Joseph (2012) used a nonstationary variance model, $\sigma^2(\mathbf{x})$, where they separated it as $\sigma^2(\mathbf{x}) = \sigma^2 \nu(\mathbf{x})$, where σ^2 is an unknown constant variance and $\nu(\mathbf{x})$ is a function of $\mathbf{x}$.

2.3.3 Constructing Gaussian process emulators

In this section, we follow the presentation in Bastos (2010) for building Gaussian process emulators. The simulator, $f(\cdot)$, is considered as an uncertain function, so to construct a Gaussian process emulator, we represent the uncertainty about the simulator outputs by a Gaussian process. This means that our prior knowledge about $f(\cdot)$ is represented by a Gaussian process with a prior mean function $m(\cdot)$

and a prior covariance function $V(\cdot,\cdot)$ using a hierarchical model

$$f(\cdot)|\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta} \sim GP(m(\cdot),V(\cdot,\cdot)), \tag{2.3.11}$$

where the prior mean function is given by

$$\mathrm{E}[f(\mathbf{x})|\boldsymbol{\beta}] = m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}, \tag{2.3.12}$$

and the prior covariance function is given by

$$\mathrm{Cov}[f(\mathbf{x}),f(\mathbf{x}')|\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta}] = V(\mathbf{x},\mathbf{x}') = \sigma^2 C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta}). \tag{2.3.13}$$

Updating the prior

Now, suppose we have n evaluated outputs, $\mathbf{y} = (y_1 = f(\mathbf{x}_1),\ldots,y_n = f(\mathbf{x}_n))^T$, of the simulator at training inputs, $\mathbf{x}_1,\ldots,\mathbf{x}_n$. According to equation (2.3.11), we describe our uncertainty about $\mathbf{y}$ with a multivariate normal distribution.

$$\mathbf{y}|\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta} \sim N_n(H\boldsymbol{\beta},\sigma^2 A), \tag{2.3.14}$$

where

$$H = [\mathbf{h}(\mathbf{x}_1),\ldots,\mathbf{h}(\mathbf{x}_n)]^T, \tag{2.3.15}$$

$$A = \begin{pmatrix} C(\mathbf{x}_1,\mathbf{x}_1) & C(\mathbf{x}_1,\mathbf{x}_2) & \cdots & C(\mathbf{x}_1,\mathbf{x}_n) \\ C(\mathbf{x}_2,\mathbf{x}_1) & C(\mathbf{x}_2,\mathbf{x}_2) & \cdots & C(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ C(\mathbf{x}_n,\mathbf{x}_1) & C(\mathbf{x}_n,\mathbf{x}_2) & \cdots & C(\mathbf{x}_n,\mathbf{x}_n) \end{pmatrix}. \tag{2.3.16}$$

Given the available outputs of the simulator, $\mathbf{y}$, the distribution of $f(\cdot)$ will be another Gaussian process given by

$$f(\cdot)|\mathbf{y},\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta} \sim GP(m_0(\cdot),V_0(\cdot,\cdot)), \tag{2.3.17}$$

where

$$\begin{aligned} m_0(\mathbf{x}) &= \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{y} - H\boldsymbol{\beta}) \\ V_0(\mathbf{x}, \mathbf{x}') &= \sigma^2[C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) - \mathbf{t}(\mathbf{x})^T A^{-1}\mathbf{t}(\mathbf{x}')], \end{aligned}$$

where $\mathbf{t}(\mathbf{x}) = (C(\mathbf{x}, \mathbf{x}_1; \boldsymbol{\delta}), \ldots, C(\mathbf{x}, \mathbf{x}_n; \boldsymbol{\delta}))^T$.

Removing the conditioning on $\boldsymbol{\beta}$ and σ^2

To find the posterior distribution of $f(\cdot)$ given $\mathbf{y}$ and $\boldsymbol{\delta}$ only, we first write

$$p(f(\cdot), \boldsymbol{\beta}, \sigma^2|\mathbf{y}, \boldsymbol{\delta}) = p(f(\cdot)|\mathbf{y}, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta})p(\boldsymbol{\beta}, \sigma^2|\mathbf{y}, \boldsymbol{\delta}). \tag{2.3.18}$$

The distribution of $p(f(\cdot)|\mathbf{y}, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta})$ is already known by equation (2.3.17), so we only need to find $p(\boldsymbol{\beta}, \sigma^2|\mathbf{y}, \boldsymbol{\delta})$. Bayes' Theorem gives

$$p(\boldsymbol{\beta}, \sigma^2|\mathbf{y}, \boldsymbol{\delta}) \propto p(\boldsymbol{\beta}, \sigma^2|\boldsymbol{\delta})p(\mathbf{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}). \tag{2.3.19}$$

Oakley and O'Hagan (2002) assumed, for convenience, a non-informative prior for $(\boldsymbol{\beta}, \sigma^2)$ which is $p(\boldsymbol{\beta}, \sigma^2|\boldsymbol{\delta}) \propto \sigma^{-2}$. Combining this prior with equation (2.3.14), the posterior distribution for $(\boldsymbol{\beta}, \sigma^2)$ has a Normal-inverse-gamma distribution, characterised by

$$\boldsymbol{\beta}|\mathbf{y}, \sigma^2, \boldsymbol{\delta} \sim N(\hat{\boldsymbol{\beta}}, \sigma^2(H^T A^{-1} H)^{-1}) \tag{2.3.20}$$

and

$$\sigma^2|\mathbf{y}, \boldsymbol{\delta} \sim \text{Inv-gamma}\left(\frac{n-q}{2}, \frac{(n-q-2)\hat{\sigma}^2}{2}\right), \tag{2.3.21}$$

where for any $\theta \sim \text{Inv-gamma}(a, b)$, the density function will be given by $f(\theta) = \frac{b^a}{\Gamma(a)}\theta^{-(a+1)}\exp(-\frac{b}{\theta})$, where $\theta, a, b > 0$. The $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ are defined by

$$\hat{\boldsymbol{\beta}} = (H^T A^{-1} H)^{-1} H^T A^{-1}\mathbf{y} \tag{2.3.22}$$

$$\hat{\sigma}^2 = \frac{\mathbf{y}^T(A^{-1} - A^{-1}H(H^T A^{-1} H)^{-1} H^T A^{-1})\mathbf{y}}{n-q-2}. \tag{2.3.23}$$

By multiplying equations (2.3.17) and (2.3.20) and integrating $\boldsymbol{\beta}$ out, it can be shown that:

$$f(\cdot)|\mathbf{y},\sigma^2,\boldsymbol{\delta} \sim GP(m_1(\cdot),V_1^*(\cdot,\cdot)), \tag{2.3.24}$$

where the posterior mean, $m_1(\cdot)$, is

$$m_1(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{y} - H\hat{\boldsymbol{\beta}}) \tag{2.3.25}$$

and the posterior variance, $V_1^*(\cdot,\cdot)$, is

$$\begin{aligned} V_1^*(\mathbf{x},\mathbf{x}') &= \sigma^2\Big[C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta}) - \mathbf{t}(\mathbf{x})^T A^{-1}\mathbf{t}(\mathbf{x}') \qquad (2.3.26)\\ &+ (\mathbf{h}(\mathbf{x}) - \mathbf{t}(\mathbf{x})^T A^{-1}H)(H^TA^{-1}H)^{-1}(\mathbf{h}(\mathbf{x}') - \mathbf{t}(\mathbf{x}')^TA^{-1}H)^T\Big]. \end{aligned}$$

By multiplying equations (2.3.21) and (2.3.24) and integrating σ^2 out, we can obtain the posterior emulator which is given by:

$$f(\cdot)|\mathbf{y},\boldsymbol{\delta} \sim \text{Student-}t \text{ Process}(n-q, m_1(\cdot), V_1(\cdot,\cdot)), \tag{2.3.27}$$

where

$$\begin{aligned} m_1(\mathbf{x}) &= \mathbf{h}(\mathbf{x})^T\hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{y} - H\hat{\boldsymbol{\beta}}), \qquad (2.3.28)\\ V_1(\mathbf{x},\mathbf{x}') &= \hat{\sigma}^2\Big[C(\mathbf{x},\mathbf{x}';\boldsymbol{\delta}) - \mathbf{t}(\mathbf{x})^T A^{-1}\mathbf{t}(\mathbf{x}') \qquad (2.3.29)\\ &+ (\mathbf{h}(\mathbf{x}) - \mathbf{t}(\mathbf{x})^T A^{-1}H)(H^TA^{-1}H)^{-1}(\mathbf{h}(\mathbf{x}') - \mathbf{t}(\mathbf{x}')^TA^{-1}H)^T\Big]. \end{aligned}$$

Although the posterior emulator is a Student-t process, we refer to the emulator as a Gaussian process throughout the thesis because the prior is a Gaussian process and in practice we use a large number of design points, so with large degrees of freedom the Student-t distribution is similar to the Gaussian distribution.

2.4 Inference for Gaussian process parameters

The parameters $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$ are assumed as unknown in building Gaussian process emulators. Posterior distributions for $\boldsymbol{\beta}$ and σ^2 can be found analytically conditional on $\boldsymbol{\delta}$, but there is no analytical method for obtaining the posterior distribution of the correlation length parameters, $\boldsymbol{\delta}$. The simplest way is to give them fixed values that can be suggested according to the prior knowledge about the simulator smoothness.

An alternative method is to obtain an estimate $\hat{\boldsymbol{\delta}}$ of $\boldsymbol{\delta}$ from the likelihood function $f(\mathbf{y}|\boldsymbol{\delta})$ or from the posterior distribution $f(\boldsymbol{\delta}|\mathbf{y})$ and then proceed with the analysis conditioning on $\boldsymbol{\delta} = \hat{\boldsymbol{\delta}}$. This method is called a plug-in approach. Several methods have been used for estimating the correlation length parameters. We present in this section a number of these methods for estimating these unknown parameters.

2.4.1 Estimating correlation length parameters using the posterior mode

In this section, we consider estimating the correlation length parameters using the posterior mode. Some authors refer to estimating $\boldsymbol{\delta}$ using the posterior mode and other refer to estimating $\boldsymbol{\delta}$ using the maximum likelihood estimate (MLE). However, if we use a flat prior for estimating $\boldsymbol{\delta}$ using the posterior mode, the two methods will be equivalent. The density function of $\mathbf{y}$ conditional on $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\delta}$ is given by

$$f(\mathbf{y}|\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta}) = \frac{|A|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{n}{2}}(2\pi)^{\frac{n}{2}}} \exp\left\{-(\mathbf{y}-H\boldsymbol{\beta})^T\frac{A^{-1}}{2\sigma^2}(\mathbf{y}-H\boldsymbol{\beta})\right\}. \qquad (2.4.1)$$

Using an improper uniform prior for each element of $\boldsymbol{\delta}$ and a non-informative priors for $\boldsymbol{\beta}$ and σ^2, $f(\boldsymbol{\beta},\sigma^2) \propto \sigma^{-2}$, the posterior density of the parameters $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\delta}$ is

$$f(\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta}|\mathbf{y}) = \frac{|A|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{n}{2}}(2\pi)^{\frac{q}{2}}} \exp\left\{-(\mathbf{y}-H\boldsymbol{\beta})^T\frac{A^{-1}}{2\sigma^2}(\mathbf{y}-H\boldsymbol{\beta})\right\}.\sigma^{-2}. \qquad (2.4.2)$$

Hence, we can obtain the posterior $f(\boldsymbol{\delta}|\mathbf{y})$ as :

$$\begin{aligned} f(\boldsymbol{\delta}|\mathbf{y}) &= \int\int \frac{|A|^{\frac{-1}{2}}}{(\sigma^2)^{\frac{n}{2}}(2\pi)^{\frac{q}{2}}} \exp\left\{-(\mathbf{y}-H\boldsymbol{\beta})^T \frac{A^{-1}}{2\sigma^2}(\mathbf{y}-H\boldsymbol{\beta})\right\} .\sigma^{-2} d\sigma^2 d\boldsymbol{\beta} \\ &= \int\int \frac{|A|^{\frac{-1}{2}}}{(\sigma^2)^{\frac{(n+2)}{2}}(2\pi)^{\frac{q}{2}}} \exp\{-[(\mathbf{y}-H\hat{\boldsymbol{\beta}})^T \frac{A^{-1}}{2\sigma^2}(\mathbf{y}-H\hat{\boldsymbol{\beta}}) \\ &+ (\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^T \frac{H^T A^{-1} H}{2\sigma^2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})]\} d\sigma^2 d\boldsymbol{\beta}. \end{aligned} \quad (2.4.3)$$

Since $\boldsymbol{\beta}|\sigma^2, \boldsymbol{\delta} \sim N(\hat{\boldsymbol{\beta}}, \sigma^2(H^T A^{-1} H)^{-1})$, then it can be seen that

$$\int \exp\left\{-(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^T \frac{H^T A^{-1} H}{2\sigma^2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\right\} d\boldsymbol{\beta} = \frac{(\sigma^2)^{\frac{q}{2}}(2\pi)^{\frac{q}{2}}}{|H^T A^{-1} H|^{1/2}}$$

Hence, by integrating $\boldsymbol{\beta}$ out from equation (2.4.3) we obtain

$$\begin{aligned} f(\boldsymbol{\delta}, \sigma^2|\mathbf{y}) &\propto \int \frac{|A|^{-\frac{1}{2}}|H^T A^{-1} H|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{1}{2}(n+2-q)}} \exp\left\{-(\mathbf{y}-H\hat{\boldsymbol{\beta}})^T \frac{A^{-1}}{2\sigma^2}(\mathbf{y}-H\hat{\boldsymbol{\beta}})\right\} d\sigma^2 \\ &= |A|^{-\frac{1}{2}}|H^T A^{-1} H|^{-\frac{1}{2}} \\ &\times \int (\sigma^2)^{-\frac{(n-q)}{2}-1} \exp\left\{-(\mathbf{y}-H\hat{\boldsymbol{\beta}})^T \frac{A^{-1}}{2\sigma^2}(\mathbf{y}-H\hat{\boldsymbol{\beta}})\right\} d\sigma^2. \end{aligned}$$

We can notice that the integrand is proportional to the inverse-gamma density with parameters $(\frac{n-q}{2})$ and $\left(\frac{(n-q-2)\hat{\sigma}^2}{2}\right)$, where $\hat{\sigma}^2 = \frac{(\mathbf{y}-H\hat{\boldsymbol{\beta}})^T A^{-1}(\mathbf{y}-H\hat{\boldsymbol{\beta}})}{n-q-2}$. Therefore, we can integrate σ^2 out to obtain

$$f(\boldsymbol{\delta}|\mathbf{y}) \propto |A|^{-\frac{1}{2}}|H^T A^{-1} H|^{-\frac{1}{2}}(\hat{\sigma}^2)^{-\frac{(n-q)}{2}} = g(\boldsymbol{\delta}). \quad (2.4.4)$$

An estimate for $\boldsymbol{\delta}$ can be obtained by maximizing (2.4.4)

$$\hat{\boldsymbol{\delta}} = \arg\max_{\boldsymbol{\delta}}(f(\boldsymbol{\delta}|\mathbf{y})). \quad (2.4.5)$$

2.4.2 Markov Chain Monte Carlo algorithm

The posterior distribution of $\boldsymbol{\delta}$ can be obtained by multiplying equation (2.4.4) with a prior distribution for $\boldsymbol{\delta}$. Some authors, for example Andrianakis and

Challenor (2011), used a Markov Chain Monte Carlo (MCMC) algorithm with different priors of the correlation length parameters to obtain samples from the posterior distribution of the correlation length parameters, $g(\boldsymbol{\delta})$. After obtaining samples from $g(\boldsymbol{\delta})$, denoted by $\boldsymbol{\delta}^i$, $i = 1, \ldots, M$, inferences can be done using the predictive distribution of $f(\mathbf{x})$, equation (2.3.24). The mean and the variance can be calculated as follows:

$$\hat{m}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} m_1^i(\mathbf{x}) \tag{2.4.6}$$

$$\hat{V}(\mathbf{x}, \mathbf{x}') = \frac{1}{M}\sum_{i=1}^{M} V_1^i(\mathbf{x}, \mathbf{x}') + \frac{1}{M}\sum_{i=1}^{M}[m_1^i(\mathbf{x}) - \hat{m}(\mathbf{x})][m_1^i(\mathbf{x}') - \hat{m}(\mathbf{x}')] \tag{2.4.7}$$

where $m^i(\mathbf{x})$ and $V^i(\mathbf{x}, \mathbf{x}')$ are the posterior mean and the posterior variance, given by (2.3.28) and (2.3.29), calculated at $\boldsymbol{\delta}^i$. The algorithm that was proposed by Andrianakis and Challenor (2011) can be described as follows: suppose $\hat{\boldsymbol{\delta}}$ is an estimate of the $\boldsymbol{\delta}$ value that maximizes the posterior $g(\boldsymbol{\delta})$. Then, the Hessian matrix, $H_{\hat{\boldsymbol{\delta}}}$, (the matrix of second derivatives) is calculated at $\hat{\boldsymbol{\delta}}$. Let $V_{\hat{\boldsymbol{\delta}}} = -c^2 H_{\hat{\boldsymbol{\delta}}}^{-1}$ where c is a scaler that controls the convergence rate of the algorithm. They set $c = 2.4/\sqrt{p}$ and then samples of $\boldsymbol{\delta}$ can be obtain as follows:

1. Initialize $\boldsymbol{\delta}^{(1)}$ at $\hat{\boldsymbol{\delta}}$.

2. Calculate $N(0, V_{\hat{\boldsymbol{\delta}}})$ at $\boldsymbol{\delta}^{(i)}$ and add it to $\boldsymbol{\delta}^{(i)}$ and call the result $\boldsymbol{\delta}^*$.

3. Calculate $\alpha = \frac{g(\boldsymbol{\delta}^*)}{g(\boldsymbol{\delta}^{(i)})}$

4. Set

$$\boldsymbol{\delta}^{(i+1)} = \begin{cases} \boldsymbol{\delta}^* & \text{with probability } \alpha \\ \boldsymbol{\delta}^{(i)} & \text{with probability } 1-\alpha \end{cases}$$

5. Repeat the 2-4 steps until an adequate number of drawn samples.

2.4.3 Cross-validation method

The cross-validation method is another choice for estimating the correlation parameters, but more computations are required in this method. Estimating $\boldsymbol{\delta}$ using the cross-validation method can be achieved using the following strategy: Suppose $\mathbf{y}_{-i}$ are all the outputs except the observation $y_i = f(\mathbf{x}_i)$. For a given value of $\boldsymbol{\delta}$, the posterior distribution of $f(\cdot)$ can be derived conditional on $\mathbf{y}_{-i}$. Then, we calculate the difference, d_i, between the posterior mean of $y_i = f(\mathbf{x}_i)$, $\mathrm{E}[f(\mathbf{x}_i)|\mathbf{y}_{-i}, \boldsymbol{\delta}]$, and the observed value, $y_i = f(\mathbf{x}_i)$. We repeat this procedure for $i = 1, \ldots, n$. The estimated values of the correlation length parameters, $\boldsymbol{\delta}$, minimize $\sum_{i=1}^{n} d_i^2$.

Table 2.1 in Section 2.2 shows 29 authors estimated the correlation length parameters using the maximum likelihood. This is because this method is less computationally expensive than the Markov Chain Monte Carlo algorithm and the cross-validation method. Table 2.1 also shows that 3 authors used the posterior mode, 5 authors used an MCMC algorithm and 1 author used the cross-validation method for estimating the correlation length parameters.

2.5 Designs for building emulators

In this section, we discuss the choice of design points for Gaussian process emulators. The design points should cover all the input space and spread evenly over the input space with the smallest number of points. A design that achieves these properties is called a space-filling design which will be focused on in this section.

The design points that are used for building Gaussian process emulators are called training points, where the simulator is evaluated at these training points.

In this section, we review some space-filling designs, first we describe the Latin hypercube design, and then describe the sliced Latin hypercube design.

2.5.1 Latin hypercube design

The Latin hypercube design (LHD) was introduced by McKay et al. (1979) as an alternative to simple random sampling design. The LHD is considered as a type of stratified sample because it partitions the region of each variable into n subregions of equal probability.

The LHD is very common in computer experiments literature as it is simple to generate and the observations are spread evenly through the input space for each variable. Table 2.1 in Section 2.2 shows 33 authors used the LHD to generate the design points for Gaussian process emulators.

Suppose we have a simulator, $y = f(\mathbf{x})$, with $\mathbf{x} = (x_1, \ldots, x_p)$. We want to generate n design points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$, and we write the design points in a matrix

$$\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}$$

with element j, k denoted by $\mathbf{X}_{jk}$.

Let Π be an $n \times p$ matrix, in which each of its columns is an independent random permutation of $\{1, 2, \ldots, n\}$. Let U_{jk} be independent, identical and uniformly distributed random variables on $(0, 1)$ and independent of Π. Then, an $n \times p$ Latin hypercube design, $\mathbf{X}$, is generated by

$$\mathbf{X}_{jk} = \frac{\Pi_{jk} - U_{jk}}{n}. \tag{2.5.1}$$

Figure 2.1 shows a LHD of size 10 in the region $\chi = [0, 1]^2$. It can be seen that the marginal spaces of both input variables are well covered.

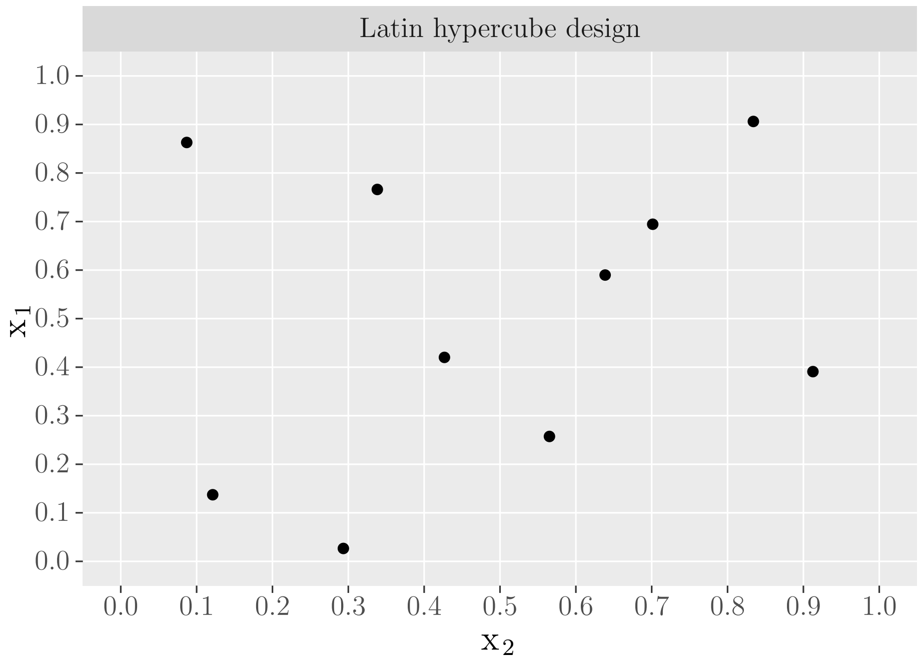

Figure 2.1: Latin hypercube design with 10 points and 2 dimensions on the region $\chi = [0, 1]^2$. The points spread evenly over the input space.

2.5.2 Sliced Latin hypercube design

The sliced Latin hypercube design (SLHD) was proposed by Qian (2012) and it is a type of a space-filling design. The SLHD is a special Latin hypercube design, which can be split into slices of smaller Latin hypercube designs. The SLHD has an attractive feature which is that maximum uniformity in any one-dimensional projection is achieved in each slice of the design.

Constructing a sliced Latin hypercube design

We describe here an easy-to-implement method, introduced by Qian (2012), for constructing a SLHD. This method is also flexible in run size and can be used with any number of factors. We illustrate this method with a simple example.

Suppose we want to construct a SLHD with n points with q slices, where each slice is a Latin hypercube design. We write these q LHDs as $\mathbf{X}_1, \ldots, \mathbf{X}_q$. To do that, let $n = mt$, where m and t are positive integers. We define $Z_n = \{1, \ldots, n\}$ and divide Z_n into m blocks, $\mathbf{b}_1, \ldots, \mathbf{b}_m$, where

$$\mathbf{b}_i = \{a \in Z_n | \lceil a/t \rceil = i\}, \qquad (2.5.2)$$

where for any $a \in R$, $\lceil a \rceil$ is defined as the smallest integer no less than a.

As an example, suppose the input is 3-dimensional with $n = 12$ design points in total and the aim is to construct a SLHD made up of 3 slices. Suppose $m = 4$ and $t = 3$, so we divide Z_{12} into four blocks, each of which contains 3 elements, $\mathbf{b}_1 = \{1, 2, 3\}$, $\mathbf{b}_2 = \{4, 5, 6\}$, $\mathbf{b}_3 = \{7, 8, 9\}$ and $\mathbf{b}_4 = \{10, 11, 12\}$, where $\mathbf{b}_i = \{a \in Z_{12} | \lceil a/3 \rceil = i\}$, for $i = 1, \ldots, 4$.

Then, we need to generate a matrix, called a sliced permutation matrix, M_n, by two steps:

Step 1: For $i = 1, \ldots, m$, permute independently the elements in the blocks $\mathbf{b}_1, \ldots, \mathbf{b}_m$. In the example, this step will be, for $i = 1, \ldots, 4$, we permute independently the elements in the blocks $\mathbf{b}_1, \ldots, \mathbf{b}_4$ to obtain, say, $\mathbf{b}_1 = \{3, 2, 1\}$, $\mathbf{b}_2 = \{5, 6, 4\}$, $\mathbf{b}_3 = \{9, 8, 7\}$ and $\mathbf{b}_4 = \{11, 12, 10\}$.

Step 2: Let M_n be an $m \times t$ matrix with rows $\mathbf{b}_1, \ldots, \mathbf{b}_m$. For $j = 1, \ldots, t$, randomly shuffle the entries in the $j - th$ column of M_n with a permutation carried out from one column to another independently. In the example, we put

$\mathbf{b}_1, \ldots, \mathbf{b}_4$ in a matrix

$$\begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{pmatrix} = \begin{pmatrix} 3 & 2 & 1 \\ 5 & 6 & 4 \\ 9 & 8 & 7 \\ 11 & 12 & 10 \end{pmatrix}$$

and then we randomly shuffle the elements in each column to obtain a sliced permutation matrix M_{12}

$$M_{12} = \begin{pmatrix} 9 & 8 & 10 \\ 5 & 2 & 1 \\ 11 & 12 & 4 \\ 3 & 6 & 7 \end{pmatrix}.$$

This permutation matrix can be used to provide the values of a single input where each column will correspond to one slice of Latin hypercube, so the first slice will have 9, 5, 11 and 3, the second slice will have 8, 2, 12 and 6 and so forth. This means that we rearrange the permutation matrix of each input and organise them in 3 matrices to obtain the final design.

To construct a SLHD, we first generate independently q sliced permutation matrices, $M_n^1, \ldots, M_n^q$, using the two steps above. In our example, we generate $q = 3$ different independent sliced permutation matrices

$$M_{12}^1 = M_{12} = \begin{pmatrix} 9 & 8 & 10 \\ 5 & 2 & 1 \\ 11 & 12 & 4 \\ 3 & 6 & 7 \end{pmatrix}, M_{12}^2 = \begin{pmatrix} 6 & 12 & 4 \\ 2 & 5 & 3 \\ 11 & 8 & 10 \\ 7 & 1 & 9 \end{pmatrix}, M_{12}^3 = \begin{pmatrix} 7 & 6 & 11 \\ 3 & 10 & 4 \\ 5 & 9 & 1 \\ 12 & 2 & 8 \end{pmatrix}.$$

Then, for $c = 1, \ldots, t$, we construct an $m \times q$ matrix $M^{(c)}$ by putting its $j-th$ column to be the $c-th$ column of M_n^j, for $j = 1, \ldots, q$. For example, the first column of M_n^1 will be the first column of $M^{(1)}$, the first column of M_n^2 will be the second column of $M^{(1)}$, $\ldots$, the first column of M_n^j will be the last column of $M^{(1)}$.

This is obtained as follows, for $c = 1, \ldots, 3$, we obtain a 4×3 matrix, $M^{(c)}$, by letting its $j - th$ column to be the $c - th$ column of M_{12}^j, for $j = 1, \ldots, 3$. For example, the first column of M_{12}^1 will be the first column of $M^{(1)}$, the first column of M_{12}^2 will be the second column of $M^{(1)}$, the first column of M_{12}^3 will be the last column of $M^{(1)}$ and so forth.

$$M^{(1)} = \begin{pmatrix} 9 & 6 & 7 \\ 5 & 2 & 3 \\ 11 & 11 & 5 \\ 3 & 7 & 12 \end{pmatrix}, M^{(2)} = \begin{pmatrix} 8 & 12 & 6 \\ 2 & 5 & 10 \\ 12 & 8 & 9 \\ 6 & 1 & 2 \end{pmatrix}, M^{(3)} = \begin{pmatrix} 10 & 4 & 11 \\ 1 & 3 & 4 \\ 4 & 10 & 1 \\ 7 & 9 & 8 \end{pmatrix}.$$

Finally, we combine the matrices $M^{(1)}, \ldots, M^{(t)}$, row by row to obtain the matrix M, given by

$$M = \cup_{c=1}^{t} M^{(c)}. \tag{2.5.3}$$

In our example, the matrices $M^{(1)}, M^{(2)}$ and $M^{(3)}$ are combined row by row to obtain the matrix M. For example, the first column of $M^{(1)}$, the first column of $M^{(2)}$ and the first column of $M^{(3)}$ will be the first column of M, the second column of $M^{(1)}$, the second column of $M^{(2)}$ and the second column of $M^{(3)}$ will be the second column of M and so forth.

$$M^T = \left(\begin{array}{cccc|cccc|cccc} 9 & 5 & 11 & 3 & 8 & 2 & 12 & 6 & 10 & 1 & 4 & 7 \\ 6 & 2 & 11 & 7 & 12 & 5 & 8 & 1 & 4 & 3 & 10 & 9 \\ 7 & 3 & 5 & 12 & 6 & 10 & 9 & 2 & 11 & 4 & 1 & 8 \end{array}\right),$$

where $M^{(1)}, M^{(2)}$ and $M^{(3)}$ in M are divided by the solid lines. As seen, M is a special LHD where each column is a permutation on Z_{12}, and each of

$\lceil M^{(1)}/3 \rceil$, $\lceil M^{(2)}/3 \rceil$ and $\lceil M^{(3)}/3 \rceil$ is a smaller LHD of 4 runs, where each column is a permutation on Z_4.

Hence, an $n \times q$ sliced Latin hypercube design, $\mathbf{X} = x_{ik}$, with q slices and $M = m_{ik}$, can be generated as follows:

$$x_{ik} = \frac{m_{ik} - u_{ik}}{n}, \text{ for } i = 1, \ldots, n, \ k = 1, \ldots, q, \tag{2.5.4}$$

where u_{ik} are independent uniform random variables on $[0, 1)$ and the u_{ik} and the m_{ik} are mutually independent. In our example, using equation (2.5.4), we obtain a SLHD, $\mathbf{X}$, of 12 runs in three factors with three slices, $\mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$. Figure 2.2 shows a SLHD of size 12 in a region $\chi = [0, 1]$ in the example.

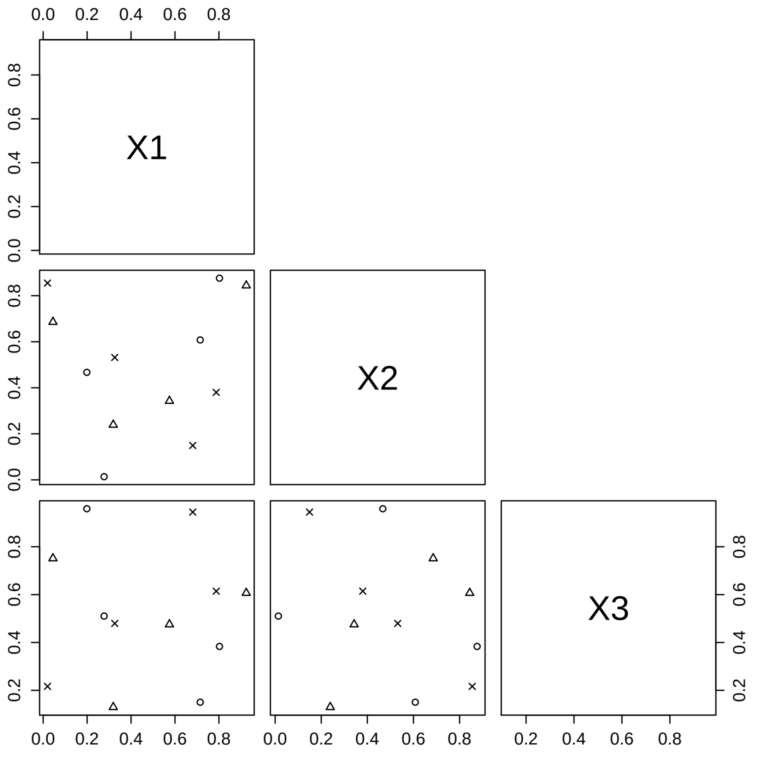

Figure 2.2: Bivariate projections of sliced Latin hypercube design with three slices $\mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$ that are represented by $\circ, +$ and $\triangle$ respectively. The points spread evenly over the input space for each slice.

2.5.3 Distance-Based designs

A design that is based on measures of distance between points and quantifies how the points are spread evenly, is called a distance-based design. Let $\mathbf{X} \subset \chi$ be an arbitrary design, which contains $\{x_1, \ldots, x_n\}$ points and let d be a metric on χ, for example, the Euclidean distance. We can measure the minimum distance between two points for any design. A design that maximizes the minimum distance is called a maximin distance design and we denote it by $\mathbf{X}_{Mm}$,

$$\mathbf{X}_{Mm} = \max_{\mathbf{X} \subset \chi} \min_{\{x, x'\} \in \mathbf{X}} d(x, x'). \tag{2.5.5}$$

A distance-based design will be called a minimax design if every point $x \in \chi$ is close to some points in $\mathbf{X}$ and it is given by

$$\mathbf{X}_{mM} = \min_{\mathbf{X}} \max_{x \in \chi} d(x, \mathbf{X}). \tag{2.5.6}$$

Minimax and maximin distance criteria measure how uniformly the points are scattered through the region. Therefore, $\mathbf{X}_{Mm}$ ensures that no two points in the input space are too close to each other (Fang et al., 2010).

Maximin Latin Hypercube Designs: A maximum Latin hypercube design, which was described by Morris and Mitchell (1995), can be obtained when we choose a design that satisfies a distance-based criterion, that is, maximizes the minimum distance between points from a set of Latin hypercube design.

2.5.4 Other space-filling designs

Lattice Designs: The points in this design are selected on a grid such that, they appear equally spaced in the input space. The lattice design was considered in computer experiments by Bates et al. (1996). To generate n lattice points in p-

dimensions, we need a positive integer set $g = \{g_1, \ldots, g_p\}$, so for $j = 0, \ldots, n-1$, we can generate lattice points as follows:

$$\mathbf{X}_{j+1} = \left(h(\frac{j}{n}g_1), \ldots, h(\frac{j}{n}g_p) \right), \tag{2.5.7}$$

where $h(x)$ returns the non integer component of x, e.g $h(2.4) = 0.4$. The problem with lattice design is that it is difficult to find suitable generator, g, because g and n are required to form a set of prime numbers and this may not fill the input space very well.

The space-filling designs have been achieved by many different sequences of numbers, where different algorithms are used to generate these designs. For instance, Weyl sequence has a generator, g, of irrational numbers on a regular space grid. A Halton sequence has a prime integers generator for each dimension, and a sequence of fractions is generated for each prime.

Sobol' Sequence: In the Sobol' design, also called LP-τ, a sequence as a set of coordinates is generated for each dimension in the same way of a Halton sequence, but the points are recorded according to a complicated rule. Galanti and Jung (1997) demonstrate the Sobol' sequence by a simple numerical example. The Sobol' sequence provides points that have a useful property that, the sequence can be expanded and it will still be a Sobol' sequence. For example, we can construct longer Sobol' sequences from a shorter Sobol' sequence by adding points to the shorter sequence. In contrast, the LHD must be recomputed if more points are needed (Santner et al., 2003).

The Sobol' sequence can be used in a computer experiments for almost all applications and it may be more convenient than the LHD if we need more information about the correlation length because it can generate a variety of inter-point distances (Santner et al., 2003).

2.6 Example

In this section, we consider the Borehole function as an example of a simulator, and for illustration, we suppose it is computationally expensive. The Borehole model is commonly used for testing methods in computer experiments. Worley (1987) used the Borehole model to describe the flow rate of water through a borehole. The Borehole model is given by:

$$f(\mathbf{x}) = \frac{2\pi T_u(H_u - H_l)}{\ln(\frac{r}{r_w})\left(1 + \frac{2LT_u}{\ln(\frac{r}{r_w})r_w^2 K_w} + \frac{T_u}{T_l}\right)}, \tag{2.6.1}$$

where $\mathbf{x} = (r_w, r, T_u, H_u, T_l, H_l, L, K_w)$. The response variable $f(\mathbf{x})$ represents the flow rate of water through the borehole in $m^3/year$. The eight input variables and their ranges and units are as follows:

- $r_w \in [0.05, 0.15](m)$ is the radius of borehole.
- $r \in [100, 50000](m)$ is the radius of influence.
- $T_u \in [63070, 115600](m^2/year)$ is the transmissivity of upper aquifer.
- $H_u \in [990, 1110](m)$ is the potentiometric head of upper aquifer.
- $T_l \in [63.1, 116](m^2/year)$ is the transmissivity of lower aquifer.
- $H_l \in [700, 820](m)$ is the potentiometric head of lower aquifer.
- $L \in [1120, 1680](m)$ is the length of borehole.
- $K_w \in [9855, 12045](m/year)$ is the hydraulic conductivity of borehole.

The design points were obtained by generating 80 training inputs using a SLHD and they are denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_{80}$. Then, we evaluated the output of the

Borehole model at these training inputs, $\mathbf{y} = (y_1 = f(\mathbf{x}_1), \ldots, y_{80} = f(\mathbf{x}_{80}))$. Before fitting a Gaussian process emulator, we transformed each input variables to be in the interval $[-1, 1]^8$. To construct the Gaussian process emulator with Bayesian framework, we assume that the prior uncertainty on the Borehole model is represented by a Gaussian process, equation (2.3.11). We used a linear mean function with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and a covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4).

According to the training data, the parameters $\boldsymbol{\delta}$ were estimated by the maximum likelihood and the results are $(0.64, 10.34, 5.69, 0.64, 3.72, 0.28, 5.73, 0.43)$. It can be noticed that some inputs have large correlation length parameters, indicating that the simulator is more smooth with these inputs. After obtaining the estimated values of the parameters, the Gaussian process emulator was validated with 20 validation inputs, also generated by a SLHD, conditioned on the training points and the estimated correlation length parameters. Figure 2.3 shows the posterior emulator for the Borehole function with narrow 95% credible intervals.

The predicted points seem to be reasonable since most of them lie on the $y = x$ line. Moreover, the uncertainty, which is represented by the error bars, seems to be very small, indicating that the emulator predictions are good approximations of the simulator outputs with 95% credible intervals.

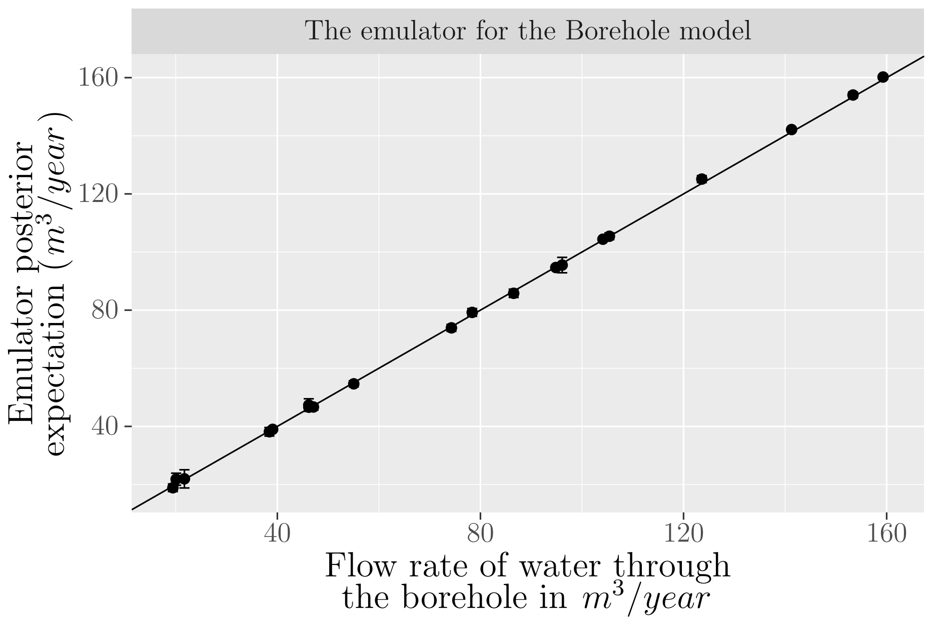

Figure 2.3: The posterior emulator approximations against observed values for the Borehole model with 95% credible intervals. The emulator predictions are good approximations of the simulator outputs with small 95% credible intervals.

2.7 Summary

In this chapter, we have reviewed the concept of emulators as surrogates of simulators. We have presented the processes of building emulators based on the Bayesian perspective and we have presented some methods for estimating the Gaussian process parameters. We have shown how design points can be chosen using the Latin hypercube design and the sliced Latin hypercube design.

Since emulators have been used in various areas of science, it is necessary to check the validation of emulators before using them. In the next chapter, we review a number of diagnostic methods as well as develop a modification of an existing diagnostic method for validating Gaussian process emulators.

Chapter 3

Validating Gaussian process emulators

3.1 Introduction

In this chapter, we explain the concept of uncertainty calibration in Gaussian process emulators. We also discuss situations where Gaussian process emulators may not perform well. We discuss the concept of diagnostics and review a number of published diagnostic methods for examining assumptions that are made in constructing Gaussian process emulators. We also develop a modification of an existing diagnostic method. Finally, we examine the performance of diagnostic methods with some different Gaussian process emulators in order to investigate whether these diagnostic methods can detect potential problems in some of these emulators.

3.2 Uncertainty calibration in emulators

In this section, we explain the concept of uncertainty calibration in Gaussian process emulators. When Gaussian process emulators do not perform well, we may obtain poor predictions of the simulator outputs or poor quantification of uncertainty in the emulator predictions. By poor predictions, we mean that the predictive mean is relatively far from the simulator outputs. By poor quantification of uncertainty, we mean the predictive variance is too large or too small resulting in the emulator being underconfident or overconfident respectively.

Let $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$ be a set of inputs, called validation inputs, and evaluations $\mathbf{y}^* = (y_1^* = f(\mathbf{x}_1^*), \ldots, y_m^* = f(\mathbf{x}_m^*))$ of the simulator outputs at these inputs, called validation outputs. Given a Gaussian process emulator for $f(\cdot)$, we can obtain, for example, the 95% credible interval for each validation output. We aim to investigate the proportion of the 95% credible intervals that contain the validation outputs. By 'good' uncertainty calibration we mean that the frequency in which the validation outputs lie inside the 95% credible intervals is 'close' to what we expect it to be (95%). However, the validation outputs are not independent and so the procedure is more complex because the distribution of the number of the 95% credible intervals that would contain the validation outputs is not known.

An underconfident emulator can occur when we obtain very wide 95% credible intervals for the emulator predictions and the validation outputs always lie inside these 95% credible intervals. Therefore, the 95% credible intervals show too much uncertainty. An overconfident emulator can occur when we obtain small 95% credible intervals for the emulator predictions, but the validation outputs always lie outside the 95% credible intervals, and hence there is an overconfidence in the emulator predictions. In order to illustrate the concept of uncertainty calibration graphically, we constructed different Gaussian process emulators on one-dimensional simulators, Figure 3.1.

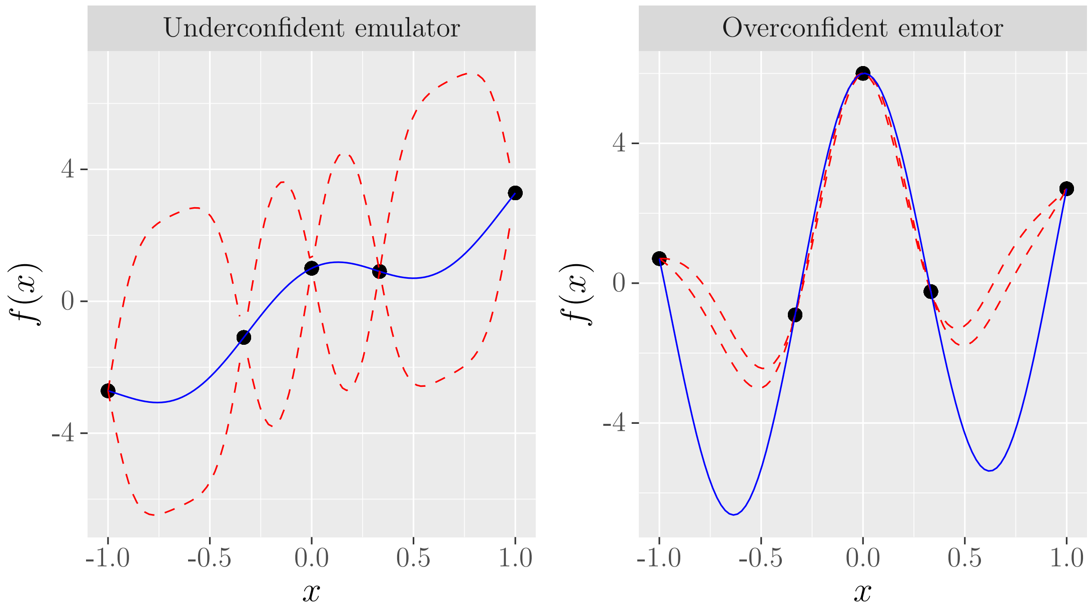

Figure 3.1: Gaussian process emulators conditioned on the training data and different values of the correlation length parameter. The black dots represent the training outputs; the solid line is the simulator and the dotted lines are the point-wise 95% credible intervals. The left panel shows an underconfident emulator while the right panel shows an overconfident emulator.

The left panel in Figure 3.1 shows an underconfident Gaussian process emulator. This is because the widths of the 95% credible intervals for the outputs are very wide and the simulator always lies inside the 95% credible intervals for any input. The right panel in Figure 3.1 presents an overconfident Gaussian process emulator. It is seen that the 95% credible intervals are small and the simulator lies well outside the 95% credible intervals over much of the input space.

3.3 Situations of inappropriate assumptions in building Gaussian process emulators

In computer modelling, emulators can be constructed under a Bayesian perspective using any joint probability distribution. The Gaussian distribution, however, is the simplest one that is mathematically tractable and leads to simple forms of the posterior mean and the posterior variance. Using other types of probability distribution could make the subsequent analysis very complex with no analytic expressions for the posterior mean and the posterior variance.

However, the simulator is not a random sample from the Gaussian process distribution. This means that when using the Gaussian process as a model of the simulator, we may not obtain accurate predictions and the uncertainty in the emulator predictions may not be quantified very well. We review a number of situations in which Gaussian process emulators may not perform well.

1. The stationary Gaussian process may fail to adapt to the smoothness of a variable in the function when the function varies more quickly in some parts than in others in the input region. In this case, the covariance will be large in some parts in the input space than in other parts and a stationary covariance function may not adapt to this behaviour.

2. Gaussian process emulators contain many parameters that need to be inferred from the data and inappropriate estimates for the Gaussian process emulators parameters may be obtained.

3. Gaussian process emulators may produce poor predictions of the simulator outputs if inappropriate forms of the mean function and the covariance function are chosen in building the emulators.

Therefore, Gaussian process emulators need to be subjected to the validation process. The validation process is the process of checking the assumptions that are used in building the Gaussian process emulator. The distribution of the emulator depends on the correlation parameters. In this chapter, for some emulators, the plug-in method is used where an estimate $\hat{\boldsymbol{\delta}}$, equation (2.4.5), for the correlation parameters is used as the true value $\boldsymbol{\delta}$ without considering uncertainty. Moreover, for some other emulators, we use the given true values of the correlation parameters, so we can investigate the consequence of estimating the correlation parameters.

3.4 Diagnostic methods for Gaussian process emulators

Diagnostic methods have been used to validate Gaussian process emulators and test the assumptions that are used in building the emulators. In general, the diagnostic can be perceived as a set of processes that can be used to assess the validity of the statistical model. For example, diagnostics can be useful tools for assessing the assumptions of the underling model. They can examine the model structure or studying subgroups of observations that have a relatively significant effect on the model predictions. Diagnostics can be graphical or quantitative results, each of which can provide a useful guide for analyses. They can provide directions to improve the assumptions of the model. Thus, by using diagnostic methods, we aim to investigate the best probability model for describing the simulator.

To explain the concept of diagnostics, suppose we have a set of validation inputs, $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$, which must be distinct from the training inputs. Thus, $\mathbf{y}^* = (y_1^* = f(\mathbf{x}_1^*), \ldots, y_m^* = f(\mathbf{x}_m^*))$ are evaluations of the simulator outputs at these validation inputs. Before running the simulator at these inputs, $\mathbf{y}^*$ is uncertain and has a multivariate Student-t distribution conditional on the training data and $\boldsymbol{\delta}$. We also define $\mathbf{y}_{obs}^*$ to be the observed values of $\mathbf{y}^*$ obtained after running the simulator.

A general diagnostic, $K(\cdot)$, can be perceived as a function of the validation outputs, $\mathbf{y}^*$, and the emulator for $f(\cdot)$: it is a function given by $K(\mathbf{y}^*, p(f(\mathbf{x}^*)|\mathbf{y}))$ and for simplicity, we denote it by $K(\mathbf{y}^*)$. Before observing the validation outputs, $K(\mathbf{y}^*)$ can be thought of as a random variable with a distribution induced by the distribution of $\mathbf{y}^*$. Therefore, it is necessary to make inference about the reference distribution of $K(\cdot)$.

We define $K(\mathbf{y}_{obs}^*)$ to be the observed value of $K(\mathbf{y}^*)$ which is calculated after observing the validation outputs. Bastos and O'Hagan (2009) propose comparing the observed value of the diagnostic, $K(\mathbf{y}_{obs}^*)$, with the distribution of $K(\mathbf{y}^*)$. This means that comparing the observed value of the diagnostic with its induced distribution by the posterior distribution of the outputs. If the observed value of the diagnostic has a small probability and lies in an appropriately selected region, a conflict will then be indicated between the simulator and the emulator. The emulator will be an accurate representation of the simulator if there are no conflicts over a wide range of such diagnostics.

We can notice that the concept of diagnostics is similar to the concept of scoring rules. A scoring rule, $S(p(\mathbf{y}^*), \mathbf{y}^*)$, can be seen as a function of the true values and a predictive distribution, $p(\mathbf{y}^*)$ (Reich and Cotter, 2015). Scoring rules evaluate the predictions after the events or variables of interest are observed. They provide a numerical score based on the predictive distribution and on the true values. Thus, scoring rules can be used to measure the accuracy of the predictions. Therefore, there is a connection between the concept of diagnostics and the concept of scoring rules. The diagnostic $K(\mathbf{y}^*)$ can be seen as a scoring rule where it is a function of the emulator of $f(\cdot)$ and the validation outputs.

However, the focus can be deferent. Scoring rules may wish to measure how well data are predicted. Diagnostics are used to investigate whether the emulator is a valid model for the simulator and not overconfident or underconfident.

3.4.1 Cross-validation method

Gaussian process emulators are usually validated using new inputs called validation inputs. In practice, however, we may have a small number of available simulator runs and we wish to use all of them as training data. Therefore, the cross-validation technique is being used as an alternative choice for validating Gaussian process emulators.

To illustrate the procedure of the cross-validation method, suppose that simulator outputs are evaluated at training inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$, to obtain training outputs $\mathbf{y} = \{y_1 = f(\mathbf{x}_1), \ldots, y_n = f(\mathbf{x}_n)\}$. We construct an emulator for $f(\cdot)$ via

$$f(\cdot)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim GP(m(\cdot), V(\cdot,\cdot)), \tag{3.4.1}$$

where

$$m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} \tag{3.4.2}$$

and

$$V(\mathbf{x}, \mathbf{x}') = \sigma^2 C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}). \tag{3.4.3}$$

For $i = 1, \ldots, n$, suppose that $\mathbf{y}_{-i}$ are the outputs of all training runs except the observation $y_i = f(\mathbf{x}_i)$. For given values of the parameters $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$, the posterior distribution of each output value, $y_i = f(\mathbf{x}_i)$, can be derived using the other $n-1$ observations, $\mathbf{y}_{-i}$. The posterior distribution of y_i is given by

$$y_i|\mathbf{y}_{-i}, \boldsymbol{\delta} \sim \text{Student-}t(n-q-1, m_i(\cdot), V_i(\cdot,\cdot)), \tag{3.4.4}$$

where $m_i(\cdot)$ and $V_i(\cdot,\cdot)$ are the posterior mean and the posterior variance given by equations (2.3.28) and (2.3.29) based on $\mathbf{y}_{-i}$. In this case, the general diagnostic, $K(\cdot)$, is a function of the simulator outputs, $\mathbf{y}$, and the posterior distribution of y_i, equation (3.4.4), and for simplicity, we denote it by $K(\mathbf{y})$.

The cross-validation can be used to predict more than one point. In this case, it will be called multifold cross-validation where $\mathbf{y}$ is split into k folds, and each fold contains t points, $t < n$, so we can remove one fold and predict its points jointly.

3.4.2 Simple diagnostic methods

Simple diagnostic methods have been used widely for validating Gaussian process emulators. They are based on calculating the differences between the validation outputs and the emulator predictions. We use the term 'simple diagnostic methods' to mean methods that do not consider uncertainty in the emulator predictions. They only depend on the posterior mean and do not take into account the posterior variance.

Table 2.1 in Section 2.2 shows that several authors used similar simple measures for validating their Gaussian process emulators. For example, the predictivity coefficient Q^2 diagnostic, also called Nash-Sutcliffe model efficiency coefficient, has been used by Marrel et al. (2015) as a diagnostic for Gaussian process emulators. The predictivity coefficient diagnostic is proposed by Nash and Sutcliffe (1970) and it is given by

$$Q^2 = 1 - \frac{\sum_{j=1}^{m}\left[\mathrm{E}[y_j^*|\mathbf{y},\boldsymbol{\delta}] - y_j^*\right]^2}{\sum_{j=1}^{m}\left[y_j^* - \frac{1}{n}\sum_{i=1}^{n} y_i\right]^2}, \tag{3.4.5}$$

where $\mathrm{E}[y_j^*|\mathbf{y},\boldsymbol{\delta}]$ are elements of the predictive mean of the Gaussian process emulator given by equation (2.3.28). The emulator predictions will be 'accurate' when the Q^2 value is close to 1.

Simple diagnostic methods using the cross-validation method

Table 2.1 in Section 2.2 also shows that several authors used simple measures for validating their Gaussian process emulators using the cross-validation method. For example, Petropoulos et al. (2009) used the root mean squared error to validate their emulators. The mean squared error is the mean of the squared errors between the emulator predictions and the simulator outputs.

$$\mathrm{MSE} = \frac{\sum_{i=1}^{n}(\mathrm{E}[y_i|\mathbf{y}_{-i},\boldsymbol{\delta}] - y_i)^2}{n},$$

where $\mathrm{E}[y_i|\mathbf{y}_{-i},\boldsymbol{\delta}]$ are elements of the predictive mean of the Gaussian process emulator given by equation (2.3.28). Thus, the root mean squared error is the square root of the mean squared error, $\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$. The MSE and RMSE are scoring rules that measure the mean squared difference between the emulator predictions and the simulator outputs. Thus, they are negatively oriented scores which means that the lower value is the better.

Also, as seen in Table 2.1, other authors considered different ways for standardising the differences between the simulator outputs and the emulator predictions. For example, Bijak et al. (2013) and Liu and Guillas (2016) used the standardised root mean squared error to validate their emulator. The root mean squared error can be standardised using the range

$$\text{SRMSE} = \frac{\text{RMSE}}{R},$$

where $R = \max(\mathbf{y}) - \min(\mathbf{y})$.

We can notice that the predictivity coefficient diagnostic, equation (3.4.5), can be seen as the skill score of the MSE. The scoring rule, S, can be converted to a skill score with respect to a reference forecast using the form

$$SS_{ref} = \frac{S - S_{ref}}{S_{perf} - S_{ref}} \times 100\%, \tag{3.4.6}$$

where S_{perf} is the accuracy measure that would be achieved by the perfect forecast and S_{ref} is the accuracy of a reference forecast which is an estimate of the marginal distribution of the predictions. A skill score is a generic term that refers to the accuracy of a prediction or an estimate of the true value with respect to a reference forecast. The skill score can be interpreted as the percentage of the improvement over the reference forecast.

For the MSE, a reference forecast is given by

$$\text{MSE}_{ref} = \frac{1}{m}\sum_{i=1}^{m}\Big(y_i^* - \frac{1}{n}\sum_{i=1}^{n} y_i\Big)^2. \tag{3.4.7}$$

The skill score for the MSE can be obtained using equation (3.4.6).

$$\begin{aligned} SS_{MSE} &= \frac{\text{MSE} - \text{MSE}_{ref}}{0 - \text{MSE}_{ref}} && (3.4.8)\\ &= 1 - \frac{\text{MSE}}{\text{MSE}_{ref}}, && (3.4.9)\end{aligned}$$

where the perfect forecast have $\text{MSE} = 0$. Perfect predictions result in a MSE of zero, and skill score value of 1. The skill score for the MSE will be zero when $\text{MSE} = \text{MSE}_{ref}$ whereas the skill score for the MSE will have a negative value when the MSE is larger than MSE_{ref} (Warner, 2010).

3.4.3 Diagnostic methods that measure uncertainty

There has been limited work on diagnostic methods that take into account uncertainty in the emulator predictions. Bastos and O'Hagan (2009) propose a number of numerical and graphical diagnostic methods based on comparisons between the validation outputs of the simulator and the emulator predictions. The diagnostic methods proposed by Bastos and O'Hagan (2009) include the individual standardised errors, the Mahalanobis distance, the credible interval diagnostic, the eigen and the pivoted Cholesky decompositions of the covariance matrix and some other diagnostics. We discuss here some of these diagnostic methods in some detail.

Individual standardised errors

The individual standardised errors for the validation outputs are given by

$$K_i^I(\mathbf{y}^*) = \frac{y_i^* - \mathrm{E}[y_i^*|\mathbf{y},\boldsymbol{\delta}]}{\sqrt{V[y_i^*|\mathbf{y},\boldsymbol{\delta}]}}, \tag{3.4.10}$$

where $V[y_i^*|\mathbf{y},\boldsymbol{\delta}]$, for $i = 1,\ldots,m$, are the elements of the predictive variance of the Gaussian process emulator given by equation (2.3.29). Each individual standardised error can be perceived as a diagnostic. The individual standardised errors will have a standard Student-t distribution if uncertainty about the simulator is properly represented by the emulator. Using a large number of training points, however, the individual standardised errors can be considered to have a standard normal distribution due to having large degrees of freedom.

A conflict will be indicated between the emulator and the simulator if large individual standardised errors are found, with an absolute value larger than, say, 3. Isolated extreme individual standardised errors may suggest a local problem only for those validation points. If large individual standardised errors tend to correspond to validation points that are close to training points, then overestimation of some correlation length parameters may be indicated. This means that the nearby training data points may influence the emulator predictions too strongly. In contrast, the underestimation of correlation length parameters may be suggested if there are small individual standardised errors.

A systematic problem will be indicated if many large individual standardised errors are found. An unsuitable choice of the mean function will be suggested or the stationary assumption is not appropriate if large errors with the same sign appear in parts of the input space (Bastos and O'Hagan, 2009). To illustrate the idea of diagnostics that compare the observed value of the diagnostic, $K(\mathbf{y}^*_{obs})$, with the distribution of $K(\mathbf{y}^*)$, we consider the following simple example.

Example

Suppose we have only one validation input, $\mathbf{x}^*$, and its evaluation $y^* = f(\mathbf{x}^*)$ of the simulator output, so the diagnostic could be

$$K^I(y^*) = \frac{f(\mathbf{x}^*) - \mathrm{E}[f(\mathbf{x}^*)|\mathbf{y}, \boldsymbol{\delta}]}{\sqrt{V[f(\mathbf{x}^*)|\mathbf{y}, \boldsymbol{\delta}]}}, \tag{3.4.11}$$

where $\mathrm{E}[f(\mathbf{x}^*)|\mathbf{y}, \boldsymbol{\delta}]$ and $V[f(\mathbf{x}^*)|\mathbf{y}, \boldsymbol{\delta}]$ are the predictive mean and the predictive variance for the Gaussian process emulator. Now for this particular choice of a diagnostic, if the emulator is valid, then before we observe $f(\mathbf{x}^*)$, the diagnostic $K^I(y^*)$ has a Student-t distribution with $n - q$ degrees of freedom. This is a simple example for a simple diagnostic whose distribution is known. Therefore, after observing the validation output, $y^*_{obs} = f_{obs}(\mathbf{x}^*)$, we calculate the observed diagnostic

$$K^I(y^*_{obs}) = \frac{f_{obs}(\mathbf{x}^*) - \mathrm{E}[f(\mathbf{x}^*)|\mathbf{y}, \boldsymbol{\delta}]}{\sqrt{V[f(\mathbf{x}^*)|\mathbf{y}, \boldsymbol{\delta}]}}. \tag{3.4.12}$$

We then compare the observed diagnostic, $K^I(y^*_{obs})$, with the distribution of $K^I(y^*)$, which is a Student-t distribution with $\mathrm{E}[f(\mathbf{x}^*)|\mathbf{y},\boldsymbol{\delta}]$, $V[f(\mathbf{x}^*)|\mathbf{y},\boldsymbol{\delta}]$ and $n-q$ degrees of freedom. If the observed value of the diagnostic, $K^I(y^*_{obs})$ is close to its expected value from this distribution, the diagnostic may suggest that the emulator is an accurate representation of the simulator. Otherwise, a conflict will be suggested between the emulator and the simulator.

Mahalanobis distance

The individual standardised errors provide a set of useful diagnostics. However, summarising them in a single value is also valuable. The Mahalanobis distance can be used as a diagnostic to measure the overall fit. The Mahalanobis distance considers the correlation among the simulator outputs and it is given by

$$K_{MD}(\mathbf{y}^*) = (\mathbf{y}^* - \mathrm{E}[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}])^T(V[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}])^{-1}(\mathbf{y}^* - \mathrm{E}[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}]). \tag{3.4.13}$$

A conflict will be indicated between the emulator and the simulator if the value of the Mahalanobis distance is extreme. Under Gaussian process emulator assumptions, Bastos and O'Hagan (2009) proved that the reference distribution of the Mahalanobis diagnostic, conditional on the training data and a correlation length parameter estimate, is a scaled F-Snedecor distribution with m and $n-q$ degrees of freedom:

$$\frac{(n-q)}{m(n-q-2)}K_{MD}(\mathbf{y}^*)|\mathbf{y},\boldsymbol{\delta} \sim F_{m,n-q}. \tag{3.4.14}$$

Variance decompositions

The individual standardised errors may be difficult to interpret due to the correlation among them. In order to tackle this problem, suppose $\mathbf{G}$ is a standard deviation matrix such that $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}] = \mathbf{G}\mathbf{G}^T$. The following transformed errors

$$K_{\mathbf{G}}(\mathbf{y}^*) = \mathbf{G}^{-1}(\mathbf{y}^* - \mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]) \tag{3.4.15}$$

will then have uncorrelated values. Under the normality assumption, each error has a standard Student-t distribution with $(n-q)$ degrees of freedom. In the $K_{\mathbf{G}}(\mathbf{y}^*)$ diagnostics, we look for large and small errors. A useful property of this diagnostic is that the sum of squares of these transformed errors, $K_{\mathbf{G}}(\mathbf{y}^*)$, is the Mahalanobis distance

$$K_{MD}(\mathbf{y}^*) = K_{\mathbf{G}}(\mathbf{y}^*)^T K_{\mathbf{G}}(\mathbf{y}^*)$$

and hence these diagnostics can be interpreted as a decomposition of $K_{MD}(\mathbf{y}^*)$. A positive definite matrix can be decomposed into the product of a square root matrix and its transpose by many various methods and Bastos and O'Hagan (2009) suggest the Cholesky decomposition.

Cholesky decomposition

We can obtain the Cholesky decomposition, $\mathbf{G}_C$, when $\mathbf{G}^T$ is the unique upper triangular matrix, $\mathbf{R}$ such that $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}] = \mathbf{R}^T\mathbf{R}$, and the elements of $K_{\mathbf{G}}(\mathbf{y}^*)$, denoted by $K^C(\mathbf{y}^*)$, will be called Cholesky errors. Each of the Cholesky errors, $K_i^C(\mathbf{y}^*)$, is the unique linear combination of the first i validation errors such that the predictive variance of the first i validation error is the conditional variance of the i-th validation error given the $i-1$ errors. The Cholesky errors produce uncorrelated transformed errors associated with the individual validation points. However, the decomposition is not invariant to the order of the validation points (Bastos and O'Hagan, 2009).

Pivoted Cholesky decomposition

Bastos and O'Hagan (2009) proposed the pivoted Cholesky decomposition to construct the uncorrelated errors, equation (3.4.15). The pivoted Cholesky decomposition can be obtained by a permutation of validation points, such that the first error corresponds to the largest predictive variance. The second error corresponds to the largest predictive variance given the first error and so forth. The pivoted Cholesky errors can be calculated by

$$K^{PC}(\mathbf{y}^*) = \mathbf{G}_{PC}^{-1}(\mathbf{y}^* - \mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]), \tag{3.4.16}$$

where $\mathbf{G}_{PC} = \mathbf{PR}$ and $\mathbf{P}$ is a permutation matrix. Thus, by plotting the pivoted Cholesky errors against the index, which is the pivoting order, we expect the errors to be around 0 with a constant variance. The pivoted Cholesky decomposition can easily be obtained in R with the function `chol(`$V[\mathtt{y}^*|\mathtt{y}, \boldsymbol{\delta}]$` ,pivot=TRUE)`.

An overconfident or underconfident emulator, is suggested if many large errors or many small errors are found in the plot. Both these cases can also indicate a nonstationarity. Moreover, further interpretation is provided by the pivoted Cholesky decomposition which they can be linked with the correlation structure. Large or small pivoted Cholesky errors on the left side of the plot suggest a nonstationary process or poor estimation of the predictive variance. In contrast, unsuitable correlation structure or poor estimation of the correlation length parameters can be indicated if extreme errors (large or small) are seen on the right side of the plot.

To show how the pivoted Cholesky decomposition works, we consider a simple one-dimensional simulator given by

$$f(x) = 3x + \sin(3x)$$

with two training inputs $\mathbf{x} = (x_1 = 0.2, x_2 = 0.8)$. After obtaining the simulator outputs at these training inputs, we derived the Gaussian process emulator assuming values of the parameters $\boldsymbol{\beta}$, σ^2 and δ. We chose the validation inputs to be $\mathbf{x}^* = (x_1^* = 0.5, x_2^* = 1.4, x_3^* = 1.5)$, where the second and the third validation points were chosen to be close to each other and far from the first validation point. Figure 3.2 shows the locations of the training points and the validation points.

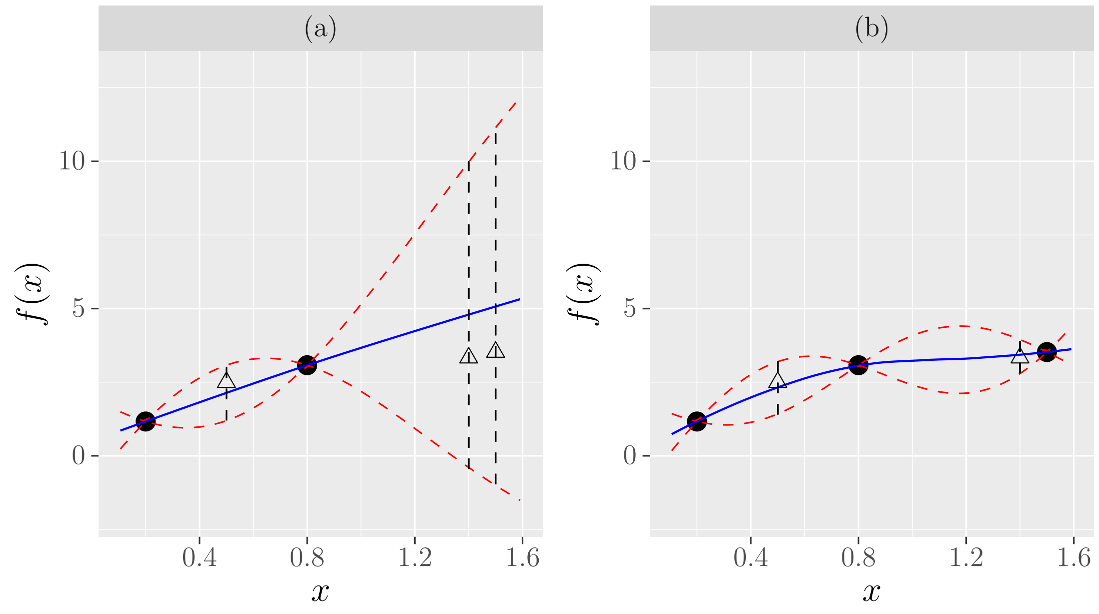

Figure 3.2: The locations of the training points (●), the validation points (△) and the posterior emulator with 95% credible intervals. In plot (a), the emulator was built using two training inputs and three validation inputs. In plot (b) the emulator was built using three training inputs and two validation inputs, where the third validation input in plot (a) was treated as a training input in plot (b).

The posterior variance matrix is

$$\mathrm{Var}[f(\mathbf{x}^*)|\mathbf{y}] \quad = \begin{pmatrix} 0.222 & -0.499 & -0.485 \\ -0.499 & 6.718 & 7.823 \\ -0.485 & 7.823 & 9.223 \end{pmatrix}$$

and as we can see that the third validation point has the largest variance, $V[f(x_3^*)|\mathbf{y}] = 9.223$; the second validation point has the second largest variance, $V[f(x_2^*)|\mathbf{y}] = 6.718$, and the first validation point has the smallest variance, $V[f(x_1^*)|\mathbf{y}] = 0.222$.

Using the `chol(`$V[\mathtt{y}^*|\mathtt{y}, \boldsymbol{\delta}]$ `,pivot=TRUE)` function in R, we then calculated the pivoted Cholesky decomposition, $\mathbf{G}_{PC} = \mathbf{PR}$, where $\mathbf{R}^T\mathbf{R} = V[\mathbf{y}^*|\mathbf{y}]$ and $\mathbf{P}$ is a permutation matrix, and it is as follows

$$\mathbf{G}_{PC} \quad = \begin{pmatrix} 3.037 & -0.160 & 2.576 \\ 0 & 0.443 & -0.197 \\ 0 & 0 & 0.211 \end{pmatrix},$$

where the pivoting order is $(3, 1, 2)$. The first index in the pivoting order is 3, which corresponds to the third validation output because it has the largest variance, $\sqrt{V[f(x_3^*)|\mathbf{y}]} = 3.037$. Then, conditional on $\mathbf{y}$ and $f(x_3^*)$, the first validation point has the second largest variance, $\sqrt{V[f(x_1^*)|\mathbf{y}, f(x_3^*)]} = 0.443$, where the second index in the pivoting order is 1. Finally, the third index in the pivoting order corresponds to the second validation point which has the smallest variance conditioned on $\mathbf{y}, f(x_3^*)$ and $f(x_1^*)$ with $\sqrt{V[f(x_2^*)|\mathbf{y}, f(x_1^*), f(x_3^*)]} = 0.211$. Note that these three (conditional) standard deviations are given in the diagonal of $\mathbf{G}_{PC}$.

Plot of the pivoted Cholesky errors against the conditional standard deviations

In this section, we suggest a modification of the pivoted Cholesky diagnostic. We propose a plot of pivoted Cholesky errors against the conditional standard deviations. In the plot of pivoted Cholesky errors against the pivoting order, the errors will be spread uniformly in the x-axis because they are plotted by index. Hence, the x-axis only represents an ordinal scale, which refers to the order of the pivoted Cholesky errors.

We propose instead to plot pivoted Cholesky errors against conditional standard deviations, so the x-axis now represents a ratio scale. The ratio scale is more informative because it allows us to make comparisons between the points. For example, we can see which points are close to each other and which points are far from the others. This allows us to determine how close the validation points are to each other and to the training points.

Mathematically, let $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$ be validation inputs, so we order these validation inputs according to conditional variances of validation outputs. The conditional standard deviations can be obtained as follows:

- Calculate $v_1 = V(f(\mathbf{x}_1^*)|\mathbf{y}), \ldots, v_m = V(f(\mathbf{x}_m^*)|\mathbf{y})$, and then choose $\mathbf{x}_{(1)}^* = \mathbf{x}_{i_1}^*$ where
$$i_1 = \arg\max_i v_i.$$

- Calculate $v_{i|(1)} = V(f(\mathbf{x}_i^*)|\mathbf{y}, f(\mathbf{x}_{(1)}^*))$ for $i = 1, \ldots, m$ and $(i \neq i_1)$, then choose $\mathbf{x}_{(2)}^* = \mathbf{x}_{i_2}^*$ where
$$i_2 = \arg\max_i v_{i|(1)}.$$

- Calculate $v_{i|(1),(2)} = V(f(\mathbf{x}_i^*)|\mathbf{y}, f(\mathbf{x}_{(1)}^*), f(\mathbf{x}_{(2)}^*))$ for $(i \neq i_1 \neq i_2)$, then choose $\mathbf{x}_{(3)}^* = \mathbf{x}_{i_3}^*$ where

$$i_3 = \arg\max_i v_{i|(1),(2)}$$

and so forth. Therefore, we plot the pivoted Cholesky errors, $K^{PC}(\mathbf{y}^*)$, against $\mathbf{S} = \{S_1 = \sqrt{v_{i_1}}, S_2 = \sqrt{v_{i_2|(1)}}, S_3 = \sqrt{v_{i_3|(1),(2)}}, \dots, S_m = \sqrt{v_{i_m|(1),(2),\dots,(m-1)}}\}$, where $v_{i_1} = V[f(\mathbf{x}_{(1)}^*)|\mathbf{y}], v_{i_2|(1)} = V[f(\mathbf{x}_{(2)}^*)|\mathbf{y}, f(\mathbf{x}_{(1)}^*)], \dots, v_{i_m|(1),(2),\dots,(m-1)} = V[f(\mathbf{x}_{(m)}^*)|\mathbf{y}, f(\mathbf{x}_{(1)}^*) \dots, f(\mathbf{x}_{(m-1)}^*)]$.

In order to scale the conditional standard deviations, the conditional standard deviations can be standardised by

$$K^{CSD}(\mathbf{y}^*) = \frac{4 \times \mathbf{S}}{R}, \tag{3.4.17}$$

where $R = \max(\mathbf{Y}) - \min(\mathbf{Y})$ and $\mathbf{Y} = (\mathbf{y}, \mathbf{y}^*)$. This scale shows, approximately, how wide the 95% credible intervals are relative to the range of the data.

3.5 Illustrative examples

In this section, we examine the performance of some diagnostic methods on different Gaussian process emulators constructed on different simulators. First, we present the mathematical expression and a brief description of each example. We then explain how we constructed a Gaussian process emulator for each example.

3.5.1 Borehole model

The description and the mathematical expression of the Borehole model have been given in Section 2.6.

3.5.2 OTL Circuit function

The output transformerless (OTL) Circuit function was used by Ben-Ari and Steinberg (2007) to show the performance of kriging methods for non-parametric smoothing of high-dimensional data. The output variable V_m is the midpoint voltage and it is given by

$$V_m(\mathbf{x}) = \frac{(V_{b1}+0.74)\beta(R_{c2}+9)}{\beta(R_{c2}+9)+R_f} + \frac{11.35R_f}{\beta(R_{c2}+9)+R_f} + \frac{0.74R_f\beta(R_{c2}+9)}{(\beta(R_{c2}+9)+R_f)R_{c1}},$$

where $V_{b1} = \frac{12R_{b2}}{R_{b1}+R_{b2}}$. The OTL Circuit function contains six input variables:

- $R_{b1} \in [50, 150]$ is the resistance $b1$ (K-Ohms).
- $R_{b2} \in [25, 70]$ is the resistance $b2$ (K-Ohms).
- $R_f \in [0.5, 3]$ is the resistance f (K-Ohms).
- $R_{c1} \in [1.2, 2.5]$ is the resistance $c1$ (K-Ohms).
- $R_{c2} \in [0.25, 1.2]$ is the resistance $c2$ (K-Ohms).
- $\beta \in [50, 300]$ is the current gain (Amperes).

3.5.3 Piston Simulation function

The Piston Simulation function was developed by Kenett and Zacks (1998) to simulate a piston motion within a cylinder. The output C is measured to be the cycle time that the piston takes to complete one cycle in seconds. It is given by

$$\begin{aligned} C(x) &= 2\pi\sqrt{\frac{M}{k+S^2\frac{P_0V_0}{T_0}\frac{T_a}{V^2}}}, \text{where} \\ V &= \frac{S}{2k}\left(\sqrt{A^2+4k\frac{P_0V_0}{T_0}T_a}-A\right), \\ A &= P_0S+19.62M-\frac{kV_0}{S}. \end{aligned}$$

The input variables that affect the performance of the piston are as follows:

- $M \in [30, 60]$ is the piston weight (kg).
- $S \in [0.005, 0.020]$ is the piston surface area (m^2).
- $V_0 \in [0.002, 0.010]$ is the initial gas volume (m^3).
- $k \in [1000, 5000]$ is the spring coefficient (N/m).
- $P_0 \in [90000, 110000]$ is the atmospheric pressure (N/m^2).
- $T_a \in [290, 296]$ is the ambient temperature (K).
- $T_0 \in [340, 360]$ is the filling gas temperature (K).

3.5.4 Multivariate Student-t simulator (Mt)

The purpose of this example is to examine the performance of diagnostics with data that have a different properties to that of a Gaussian process. The multivariate Student-t distribution has heavy tails, so we aim to investigate whether the diagnostics can detect this behaviour. We generated n design points in the $[-1, 1]^8$ space using a Latin hypercube design (LHD), denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where $\mathbf{x}_i$ is a vector of eight input variables.

Assuming values of the parameters $\boldsymbol{\beta} = (5, 10.5, 21.2, 9.4, 1.2, 15, 8, 0.2, 1.8)$, $\sigma^2 = 150$ and $\boldsymbol{\delta} = (1.8, 31.23, 2.26, 10.09, 2.04, 0.21, 1.05, 1.4)$, we first calculated the mean vector, $M = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ with $\mathbf{h}(\mathbf{x}) = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function, $C(\mathbf{x}, \mathbf{x}')$, given in equation (2.3.4). Then, an $n \times 1$ vector of 'simulator outputs', $\mathbf{y} =$

$(y_1, \ldots, y_n)$, was generated from the multivariate Student-t distribution with 3 degrees of freedom, M and V

$$\mathbf{y} \sim \text{multivariate Student-}t\ (3, M, V). \tag{3.5.1}$$

Thus, we have $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, and we selected some of them to be training points and the rest to be the validation points.

3.5.5 Nonstationary variance simulator (NSV)

The purpose of this example is to examine the performance of diagnostic methods with data that have a nonstationary variance. The aim therefore is to investigate whether or not the diagnostics can detect that the Gaussian process emulator with the assumption of a stationary variance is not suitable for these data. We generated n design points in the space $[-1, 1]^8$ using a LHD, denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where $\mathbf{x}_i$ is a vector of eight input variables.

Assuming values of the parameters $\boldsymbol{\beta} = (0.3, 0.5, 2.2, 2.4, 0.2, 0.1, 0.8, 0.2, 0.8)$ and $\boldsymbol{\delta} = (0.803, 3.239, 2.2634, 0.0945, 0.047, 0.216, 1.05, 1.4)$, the Gaussian process was specified with a linear mean function, equation (2.3.1), with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4). First, we calculated the mean vector $M = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ and the correlation matrix, $C = C(\mathbf{x}, \mathbf{x}')$. Then, the variance was chosen to be a nonstationary where we first chose a function of the inputs defined by

$$g(\mathbf{x}_i) = b_0 + b_1 x_1 + \ldots + b_8 x_8, \tag{3.5.2}$$

where $b_0 = 0.3, b_1 = 0.4, b_2 = 0.2, b_3 = 0.01, b_4 = 0.13, b_5 = 0.01, b_6 = 0.03, b_7 =$

$0.8, b_8 = 0.12$. Then, a matrix Σ was obtained by

$$\Sigma = \begin{pmatrix} g^2(\mathbf{x}_1) & g(\mathbf{x}_1)g(\mathbf{x}_2) & \cdots & g(\mathbf{x}_1)g(\mathbf{x}_n) \\ g(\mathbf{x}_2)g(\mathbf{x}_1) & g^2(\mathbf{x}_2) & \cdots & g(\mathbf{x}_2)g(\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathbf{x}_n)g(\mathbf{x}_1) & g(\mathbf{x}_n)g(\mathbf{x}_2) & \cdots & g^2(\mathbf{x}_n) \end{pmatrix}. \tag{3.5.3}$$

The choice of equation (3.5.2) results in standard deviations that vary by a factor of 2 over the input space. To obtain the variance matrix, we multiplied the matrix Σ by the matrix C element by element, $V_{ij} = \Sigma_{ij} \times C_{ij}$.

Then, an $n \times 1$ vector of 'simulator outputs', $\mathbf{y} = (y_1, \ldots, y_n)$, was generated from the multivariate normal distribution with mean M and covariance matrix V

$$\mathbf{y} \sim N_n(M, V). \tag{3.5.4}$$

Thus, we have $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, and we selected some of them to be training points and the rest to be the validation points.

3.5.6 A Gaussian process simulator (GP)

The purpose of this example is to investigate the performance of diagnostics when the Gaussian assumption known to be correct. If the data come from the Gaussian distribution, Gaussian process emulators should pass such diagnostics. We generated n design points in the space $[-1, 1]^8$ using a LHD from the Gaussian process.

Assuming values of the parameters $\boldsymbol{\beta} = (5, 10.5, 21.2, 9.4, 1.2, 15, 8, 0.2, 1.8)$, $\sigma^2 = 3$ and $\boldsymbol{\delta} = (1.8, 31.23, 2.26, 10.09, 2.04, 0.21, 1.05, 1.4)$, we first calculated the mean vector, $M = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ with $\mathbf{h}(\mathbf{x}) = (1, \mathbf{x}^T)$ and the covariance matrix,

$V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4), where $\mathbf{x}_i$ is a vector of eight input variables. Then, an $n \times 1$ vector of 'simulator outputs', $\mathbf{y}$, was generated from the Gaussian distribution with M and V

$$\mathbf{y} \sim N_n(M, V). \tag{3.5.5}$$

Thus, we have $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, and we selected some of them to be training points and the rest to be the validation points.

For the Multivariate t simulator, nonstationary variance simulator and the Gaussian process simulator, we will examine the performance of diagnostics for emulators that are derived based on the case when the values of the correlation length parameters, $\boldsymbol{\delta}$ are known and also for emulators that are derived based on the estimated values of the correlation length parameters, $\hat{\boldsymbol{\delta}}$.

Building Gaussian process emulators

Before fitting a Gaussian process, we transformed each input variable for the Borehole model, the OTL Circuit function and the Piston Simulation function to be in the interval $[-1, 1]^8$. The design points for the examples were obtained using a LHD. We built the emulators using different set sizes of the training points, five times the input dimension p of each model, $5p$, ten times the input dimension of each model, $10p$, and twenty times the input dimension of each model, $20p$. The number of the validation points was chosen to be three times the input dimension, $3p$. Then, we evaluated outputs of each model at its training inputs to obtain the training outputs.

In order to construct Gaussian process emulators under a Bayesian framework, we assumed that the prior uncertainty of each model is represented by a Gaussian process, equation (2.3.11). We used a linear mean function with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4). The correlation length parameters were estimated by the maximum likelihood. We then derived the Gaussian process emulator for each model.

The performance of the Gaussian process emulators

In order to show the typical behaviour of our emulators, we plotted them based on a single choice of training and validation points. Figure 3.3 shows the Gaussian process emulators with 95% credible intervals, where the emulators are denoted by (B) for the Borehole model, (OTLC) for the OTL Circuit model and (PS) for the Piston Simulation model.

The emulator plot of the B simulator with $5p$ training points shows that most of the predicted points do not lie on the $y = x$ line. Moreover, its uncertainty, which is represented by the error bars, is large. It can also be shown that one point seems to be far from the others and lies far from the line $y = x$. This suggests that the emulator performance is not accurate in some parts of the input space. By increasing the training points to $10p$, the emulator performance improved as most of the predicted points now lie on the $y = x$ line. Moreover, its uncertainty seems to be small, but some points still do not lie on the $y = x$ line. Using $20p$ training points, all the predicted points lie close to the $y = x$ line with small uncertainty, indicating that the emulator predictions are good approximations of the simulator outputs.

Using $5p$ training points, the predicted points of the OTLC simulator seem to be reasonable since most of them lie on the $y = x$ line. Moreover, the uncertainty

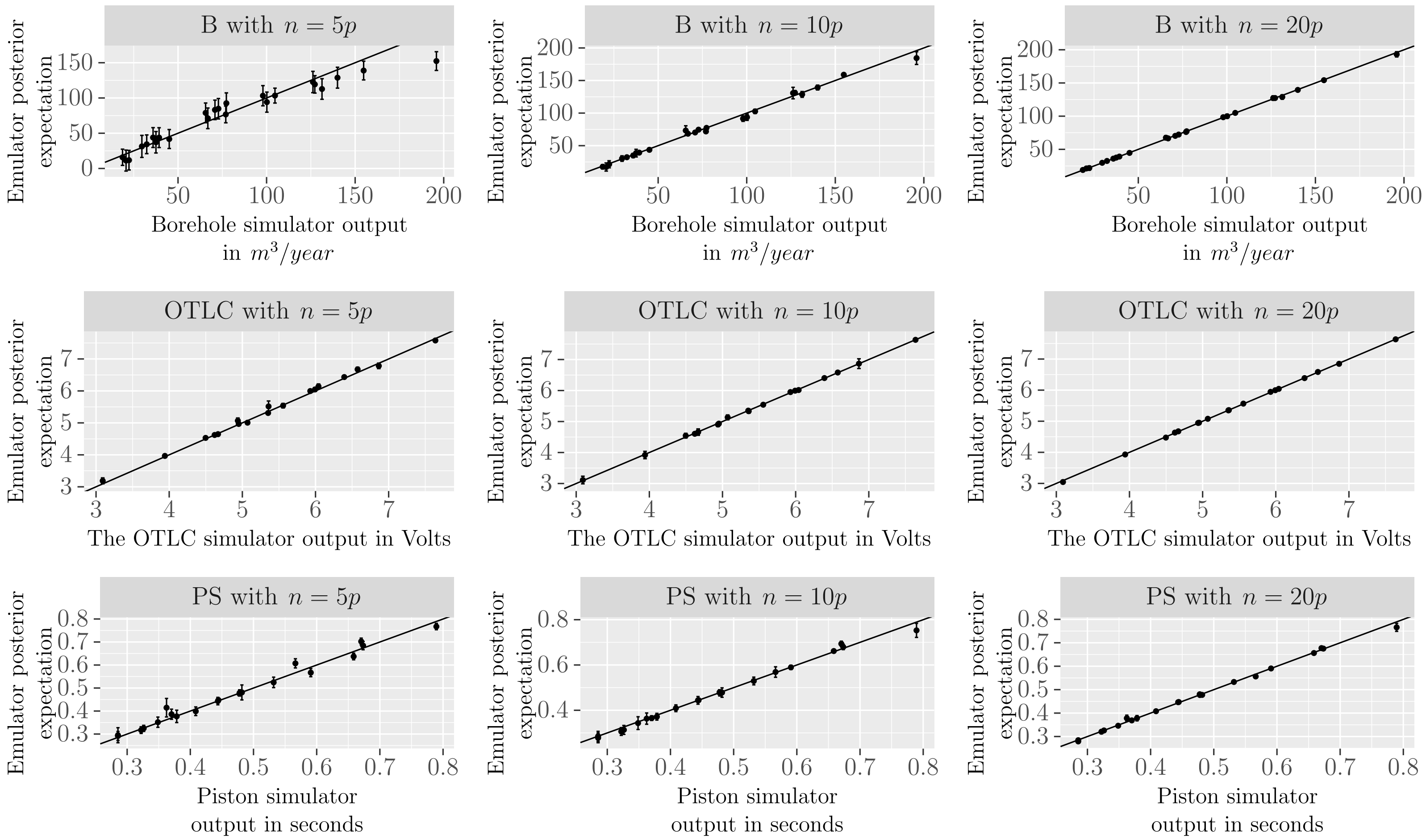

Figure 3.3: Emulator posterior mean with 95% credible intervals against true simulator output values, at separate set of validation points, $3p$, using different sample sizes of training points. The 95% credible intervals are very narrow, so that they are not visible in some plots. Most of the plots show the predictions of emulators are good approximations of the simulator outputs.

seems to be small. With $10p$ and $20p$ training points, all the predicted points lie on the $y = x$ line with very small uncertainty, indicating that the emulator predictions are good approximations of the simulator outputs.

With $5p$ training points, the predicted points of the PS simulator also seem to be reasonable and have relatively small 95% credible intervals. Several points, however, do not lie on the $y = x$ line. The predicted points were improved with $10p$ training points and the uncertainty seems to be small, but some points still do not lie on the $y = x$ line. With $20p$ training points, the emulator plot shows that most of the predicted points lie on the $y = x$ line with small uncertainty.

Figure 3.4 shows the Gaussian process emulators with 95% credible intervals, where the emulators are denoted by (Mt) for the multivariate Student-t simulator and (NSV) for the nonstationary variance simulator. It can be shown that with $5p$ training points, most of the predicted points do not lie on the $y = x$ line. Moreover, their uncertainties are very large. With $10p$ and $20p$ training points, it can be seen that most of the predicted points still do not lie on the $y = x$ line. Moreover, their uncertainties are still very large. This may suggest that the predictions are not good approximations of the true values (the simulated values). We can also note that the predicted points of the NSV simulator have the same level of error bars for all the validation outputs.

We can conclude that increasing the training points have improved the emulator predictions for the Borehole model, the OTL Circuit model and the Piston Simulation model. However, the emulator predictions for the multivariate Student-t simulator and the nonstationary variance simulator were not improved by increasing the training points.

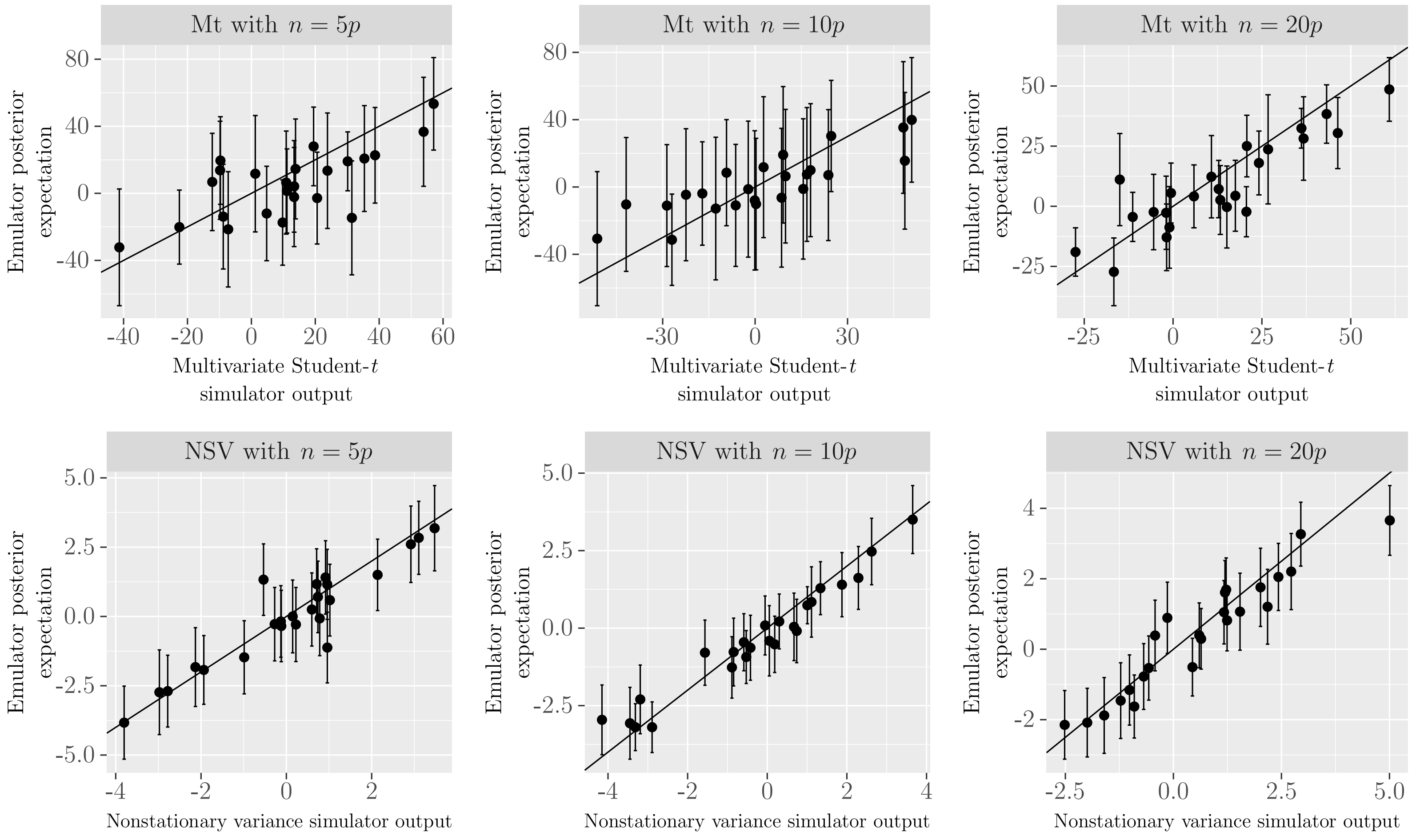

Figure 3.4: Emulator posterior mean with 95% credible intervals against true simulator output values, at separate set of validation points, $3p$, using different sample sizes of training points. The plots show large 95% credible intervals.

3.5.7 Evaluating some diagnostics for the emulators

We performed a number of numerical and graphical diagnostic methods for the emulators in order to check whether these diagnostics can detect the behaviour of the data. These diagnostic methods are based on comparisons between the emulator predictions and the simulator outputs.

Diagnostics measure the overall fit of emulators

We calculated the standardised root mean squared error, SRMSE, the predictivity coefficient, Q^2, and the Mahalanobis distance, $K_{MD}(\cdot)$, that measure the overall fit of Gaussian process emulators. We considered these diagnostics with different combinations of sample sizes for training and validation inputs. Moreover, we considered these diagnostics with different methods for validating emulators, the cross-validation method and using separate sets of validation points, Table 3.1.

Table 3.1: The notations and numbers of training and validation points with different methods for building Gaussian process emulators, the cross-validation method and separate set of validation points.

Number of training points	Cross-validation method	Number of validation points		
		$2p$	$3p$	$1000p$
$5p$	$5pcv$	$5p2p$	$5p3p$	$5p1000p$
$10p$	$10pcv$	$10p2p$	$10p3p$	$10p1000p$
$20p$	$20pcv$	$20p2p$	$20p3p$	$20p1000p$

To simplify the notations, we used, for example, '$10p2p$' to refer to $10p$ training points and $2p$ validation points, and '$10pcv$' to refer to $10p$ training points with cross-validation method. Each of the 12 combinations for training and validation inputs was replicated 100 times with different design points for each replication. We selected $2p$ and $3p$ as 'plausible' numbers of validation points and $1000p$ is selected to provide an indication of 'true' predictive performance over the input space.

Having 100 values of each diagnostic, we used the box-plot to show the performance of each diagnostic. The box-plots of the SRMSE for the emulators with different combinations of sample sizes for training and validation inputs for both the cross-validation method and separate set of validation points for validating emulators are shown in Figure 3.5.

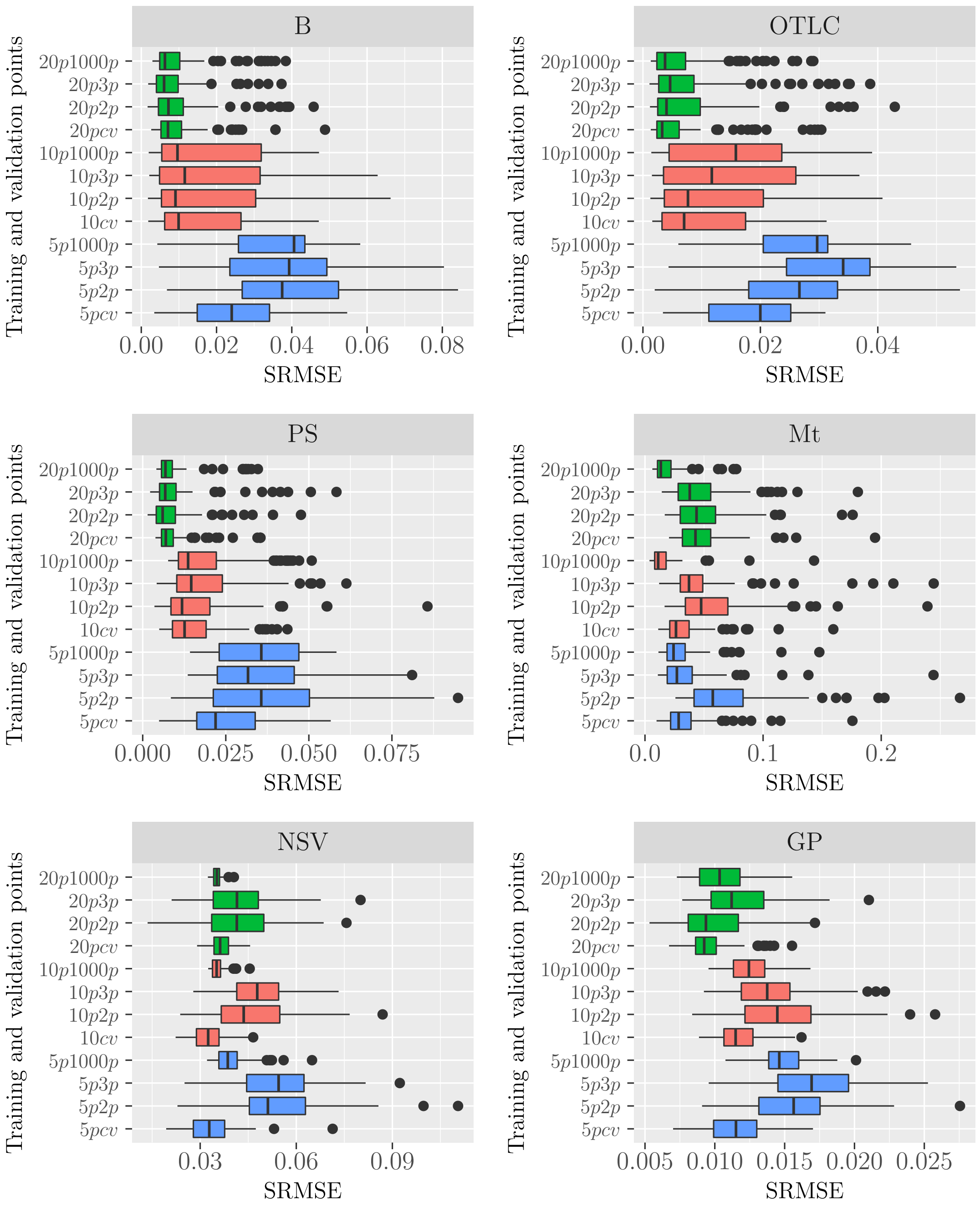

Figure 3.5: Box-plots of the SRMSE obtained using different numbers of training and validation points and different methods for building emulators, the cross-validation method and separate set of validation points. The box-plots for the emulators present small values of the SRMSE, especially with cross-validation method. The box-plots show that increasing the number of the validation points is insignificant. There is no obvious pattern in the box-plots for the emulator of the Mt simulator.

According to the box-plots of the SRMSE for the emulators, we can obtain the following results:

- **Result 1**: The box-plots for the emulators present small SRMSE values.

  The box-plots show small values of the SRMSE. The values of the SRMSE for the emulators of the B, OTLC and PS simulators approach to zero as the number of training points increases. However, the values of the SRMSE in the box-plots for the emulators of NSV and GP simulators were slightly reduced by increasing the training points.

- **Result 2**: The SRMSE values with cross-validation method are less representative.

  As seen the box-plots provide smaller values than those using separate validation points for almost all the different combinations of training and validation points. Using $20p$ training points, the box-plots of the SRMSE with cross-validation method have become almost equivalent to those with separate validation points for all the different numbers of validation points.

- **Result 3**: The set size of the validation points is less important.

  As we can see that the box-plots for almost all the different combinations of training and validation points, increasing the number of the validation points from $2p$ to $3p$ and then to $1000p$ has a minor impact on the SRMSE performance if the number of training points is fixed.

- **Result 4**: There is no obvious pattern in the box-plots of the SRMSE for the emulator of the Mt simulator.

  The box-plots do not show any obvious pattern for deciding how the emulator performs with cross-validation or using separate validation points.

Also, there is no obvious feature for deciding how the emulator performs by increasing the training or the validation points.

We also applied the predictivity coefficient, Q^2 to the emulators. However, the Q^2 is the skill score of the MSE and negative values of the Q^2 can be obtained which makes it difficult to interpret. This can occur when we have large values of $\sum_{j=1}^{m}\left[\mathrm{E}[y_j^*|\mathbf{y},\boldsymbol{\delta}] - y_j^*\right]^2$ over small values of $\sum_{j=1}^{m}\left[y_j^* - \frac{1}{n}\sum_{i=1}^{n} y_i\right]^2$ in equation (3.4.5). Furthermore, the standardised root mean squared error is the square root of the mean squared error that is scaled by $R = \max(\mathbf{y}) - \min(\mathbf{y})$. The predictivity coefficient diagnostic is one minus the mean squared error that is scaled by $\frac{\sum_{j=1}^{m}\left[y_j^* - \frac{1}{n}\sum_{i=1}^{n} y_i\right]^2}{m}$. Thus, the SRMSE diagnostic and the Q^2 diagnostic measure the same quantity but with different scales.

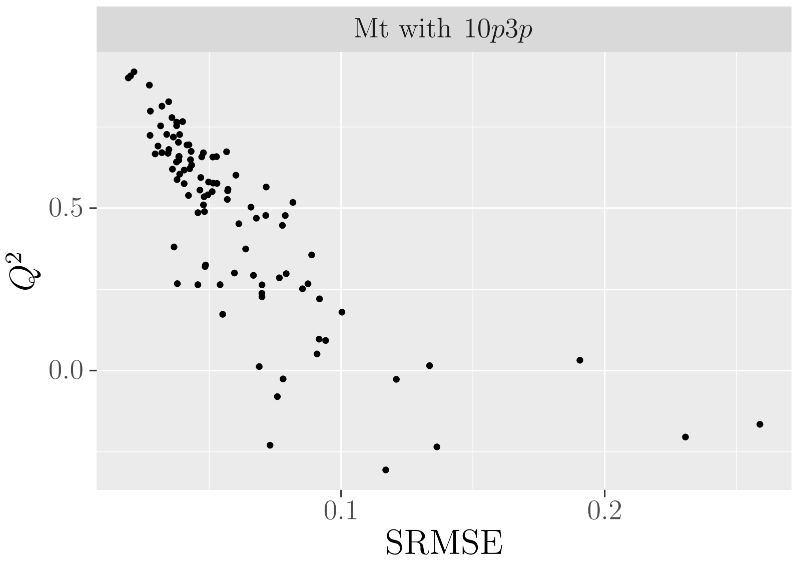

Figure 3.6: Plot of the SRMSE against the Q^2 for the emulator of the multivariate t simulator. The plot shows a negative linear relationship between these two diagnostics.

Figure 3.6 shows the relationship between these two diagnostics for the emulator of the multivariate t simulator. As we can see that the values of the Q^2 decrease as the values of the SRMSE increase, suggesting a strong negative linear relationship between these two diagnostics. This means that we can expect the values of Q^2 if we know the values of the SRMSE. Thus, these two diagnostics provide almost the same information.

The Mahalanobis distance is another diagnostic that we applied to our emulators. Figure 3.7 presents the box-plots of the Mahalanobis distance for the examples. For the emulators of the Mt, NSV and GP simulators, we examine the performance of the Mahalanobis distance for emulators that are derived based on the estimated values of the correlation length parameters, $\hat{\boldsymbol{\delta}}$ and also for emulators that are derived based on the original values of $\boldsymbol{\delta}$. The MtO, NSVO and GPO refer to the emulators of the Mt, NSV and GP simulators that are built based on the original values of $\boldsymbol{\delta}$. As we can see that the box-plots of the Mahalanobis distance are very tight due to the outliers. This makes the box-plots of the Mahalanobis distance very difficult to interpret.

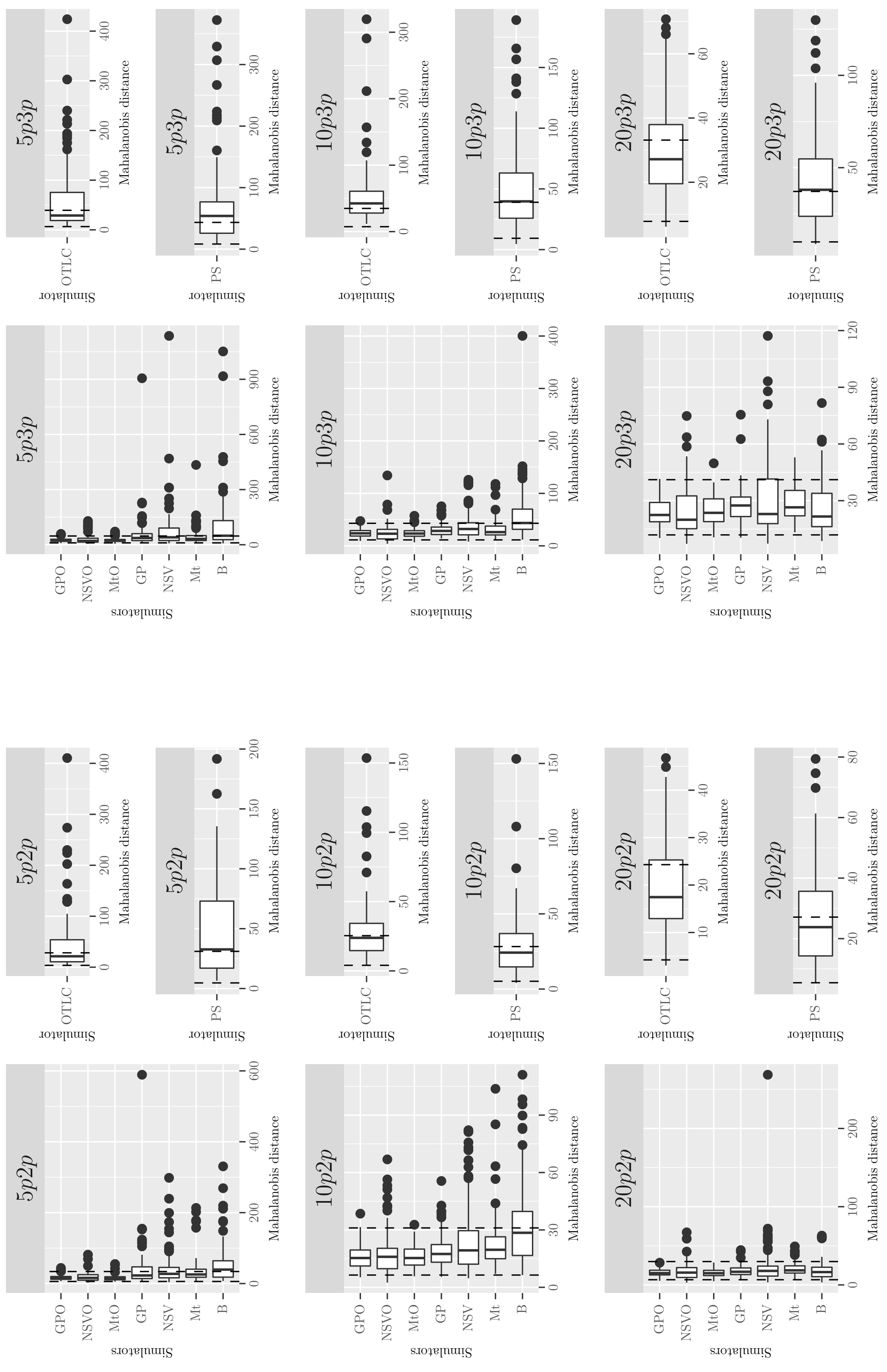

Figure 3.7: Box-plots of the Mahalanobis distance and the 95% credible intervals, the dotted lines, obtained using different combinations of training and validation points for the emulators. The MtO, NSVO and GPO refer to the emulators of the Mt, NSV and GP simulators that are built based on the original values of $\boldsymbol{\delta}$. The box-plots are very difficult to interpret due to the outliers. The plots of the OTLC and PS simulators are separated as they have different dimensions from the others.

Therefore, we excluded the outliers from the plots in order to show the performance of diagnostics clearly. Figure 3.8 presents the box-plots of the Mahalanobis distance for the examples after excluding the outliers.

According to Figure 3.8, we can obtain the following results:

- **Result 1**: The Mahalanobis distance values are more often outside the bounds of the 95% credible intervals for the emulators of the B, OTLC and PS simulators than expected, when derived based on the estimated values of the correlation parameters.

  Assuming the Gaussian process assumption is valid, but using an estimate $\hat{\boldsymbol{\delta}}$ of the correlation parameters, the box-plots of the Mahalanobis distance provide relatively many large values with $5p2p$ that lie outside the bounds. The Mahalanobis distance values that lie outside the bounds decrease by increasing the number of the training points to $10p2p$ and $20p2p$. However, many large values of the Mahalanobis distance still do not lie in the 95% credible interval. This may indicate that a problem in the process of estimating the correlation parameters. Moreover, this may also suggest that based on $\hat{\boldsymbol{\delta}}$, the emulator may not necessarily behave well.

- **Result 2**: The Mahalanobis distance did not detect the potential problems in the emulators of the Mt and NSV simulators.

  Using the true $\boldsymbol{\delta}$, the box-plots present smaller values of the Mahalanobis distance for all the different combinations of the training and validation points. Furthermore, using an estimate $\hat{\boldsymbol{\delta}}$ of the correlation parameters, the Mahalanobis distance values more often lie inside the bounds for the emulators of the Mt and NSV simulators. The box-plots present small values of the Mahalanobis distance with $10p2p$ and $20p2p$.

- **Result 3**: The Mahalanobis distance does not behave very consistently over replications for the emulators of the B, OTLC and PS simulators.

  It can be seen that the box-plots of the Mahalanobis distance show a variability in the performance of the Mahalanobis distance over replications for the emulators of these real simulators. Although, the values of the Mahalanobis distance that lie outside the bounds decrease by increasing the number of the training points to $10p2p$ and $20p2p$, many large values of the Mahalanobis distance still do not lie in the 95% credible interval.

- **Result 4**: The effect of increasing the validation points is negligible.

  It can be seen in the all the box-plots that increasing the validation points from $2p$ to $3p$ has a minor impact on the performance of the Mahalanobis distance. This is because there is no a big change in the distribution of the Mahalanobis distance as we change the number of the validation points.

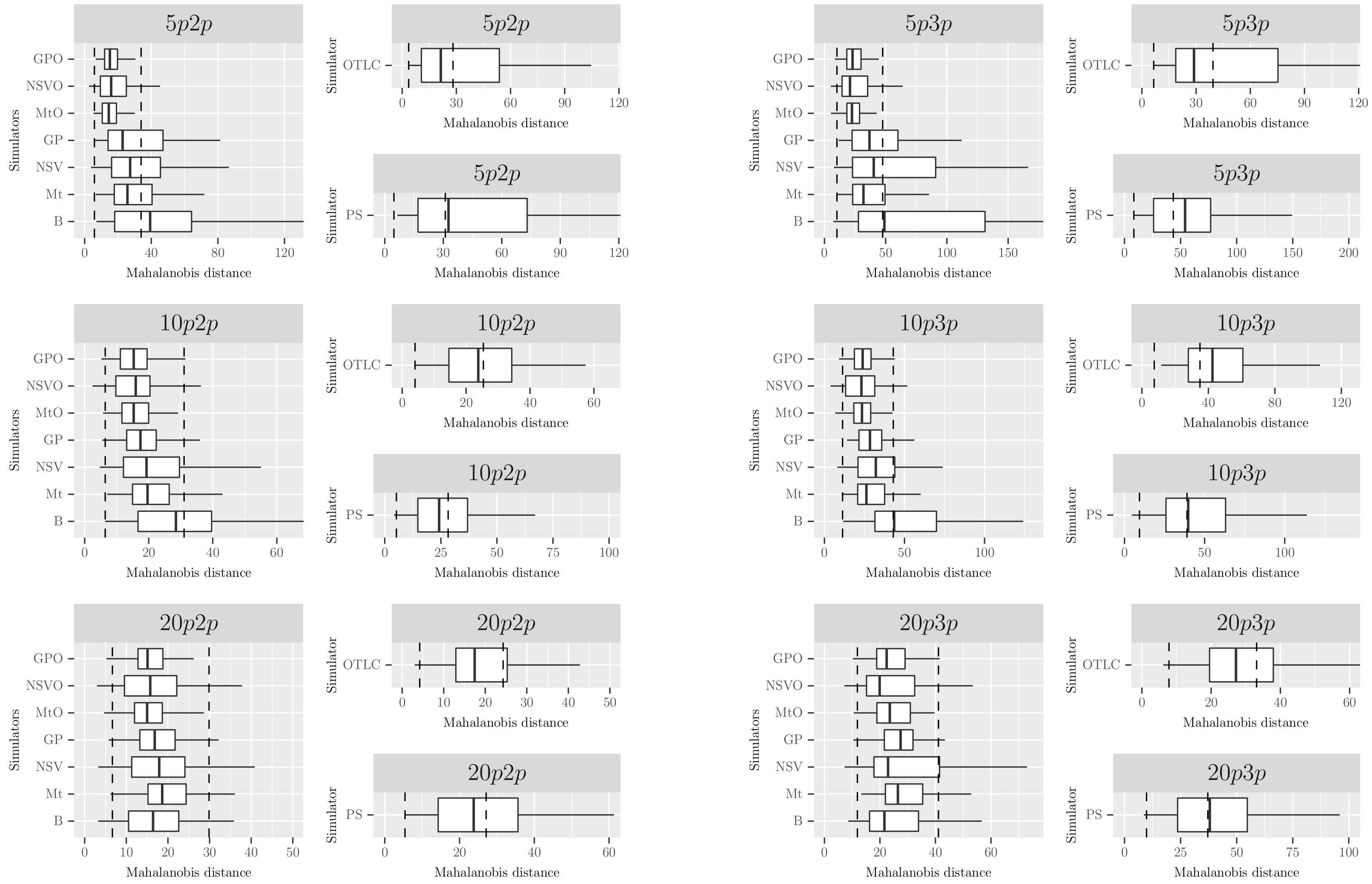

Figure 3.8: Box-plots of the Mahalanobis distance and the 95% credible intervals, the dotted lines, obtained using different combinations of training and validation points for the emulators. The MtO, NSVO and GPO refer to the emulators of the Mt, NSV and GP simulators that are built based on the original values of $\boldsymbol{\delta}$. The plots of the OTLC and PS simulators are separated as they have different dimensions from the others. The plots that are based on $\hat{\boldsymbol{\delta}}$ shows large Mahalanobis distance values. The plots of the Mt and NSV simulators shows small values of the Mahalanobis distance.

As we have seen that the performance of the Mahalanobis distance is not consistent over replications for the emulators of some simulators. According to equation (3.4.13), it is possible to obtain large value of the Mahalanobis distance even when the posterior mean is very close to true value of the simulator. This happens when the difference between the posterior mean and the true value is small and the posterior variance is very small. Hence, we plotted the Mahalanobis distance against the SRMSE for the emulator of the Multivariate t simulator to investigate the relationship between these two diagnostics, Figure 3.9.

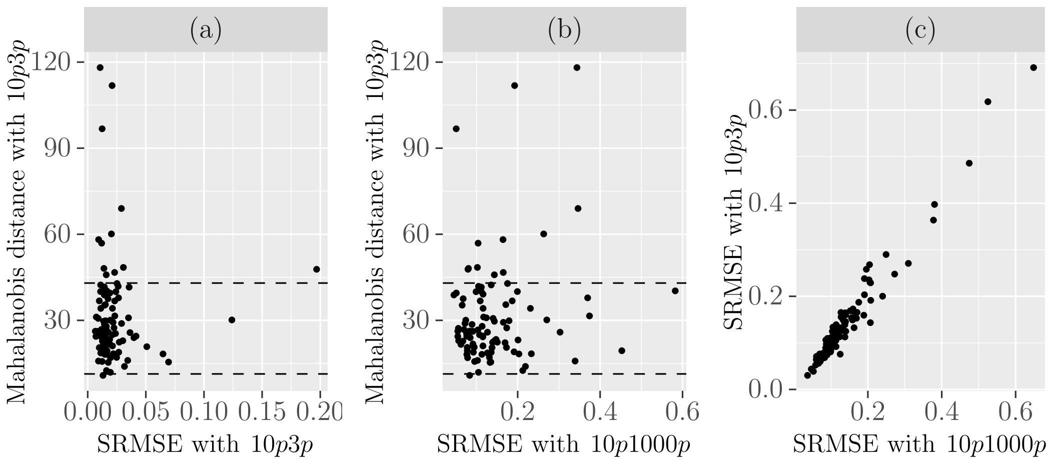

Figure 3.9: Plots of the SRMSE against the Mahalanobis distance with 95% credible interval for the emulator of the Multivariate t simulator: (a) SRMSE with $10p3p$ against the Mahalanobis distance with $10p3p$;(b) SRMSE with $10p1000p$ against the Mahalanobis distance with $10p3p$; (c) SRMSE with $10p1000p$ against SRMSE with $10p3p$. Plots (a) and (b) show many large Mahalanobis distance values while plot (c) demonstrates a positive relationship between SRMSE with $10p1000p$ and SRMSE with $10p3p$.

Figure 3.9 (a) shows the plot of the SRMSE against the Mahalanobis distance using $10p3p$. As we can see that many large values of the Mahalanobis distance lie outside the 95% credible interval and correspond to small values of SRMSE. This indicates that the difference between the posterior mean and the true values is small and the posterior variance is very small which leads to large values of the Mahalanobis distance. Thus, when we obtain large Mahalanobis distance value, further investigation is needed because the prediction may be very close to the true value and the variance is very small.

In Figure 3.9 (b) we plotted the SRMSE with $10p1000p$ against the Mahalanobis distance with $10p3p$. There are also many large values of the Mahalanobis distance lie outside the 95% credible interval with small values of SRMSE. This confirms that large values of the Mahalanobis distance can be obtained when the difference between the posterior mean and the true values is small and the posterior variance is very small.

Figure 3.9 (c) presents the plot of SRMSE with $10p1000p$ against the SRMSE with $10p3p$. We can see that there is a strong positive linear relationship between the values of the SRMSE with $10p1000p$ and the values of the SRMSE with $10p3p$. This means that increasing the number of the validation points to $1000p$ has a minor impact on the SRMSE performance. Thus, a small number of validation points, $3p$, is sufficient for obtaining a reliable result of the SRMSE diagnostic.

Graphical diagnostics for Gaussian process emulators

We performed graphical diagnostics that consider a comparison between each validation point and its predicted value and take into account the uncertainty in the emulator predictions. We considered the plot of pivoted Cholesky errors against the pivoting order and the plot of pivoted Cholesky errors against the scaled conditional standard deviations, $K^{CSD}(\mathbf{y}^*)$. Figure 3.10 shows the plots of the pivoted Cholesky errors against the pivoting order and the plots of the pivoted Cholesky errors against the scaled conditional standard deviations for the emulator of the Borehole simulator using $10p3p$.

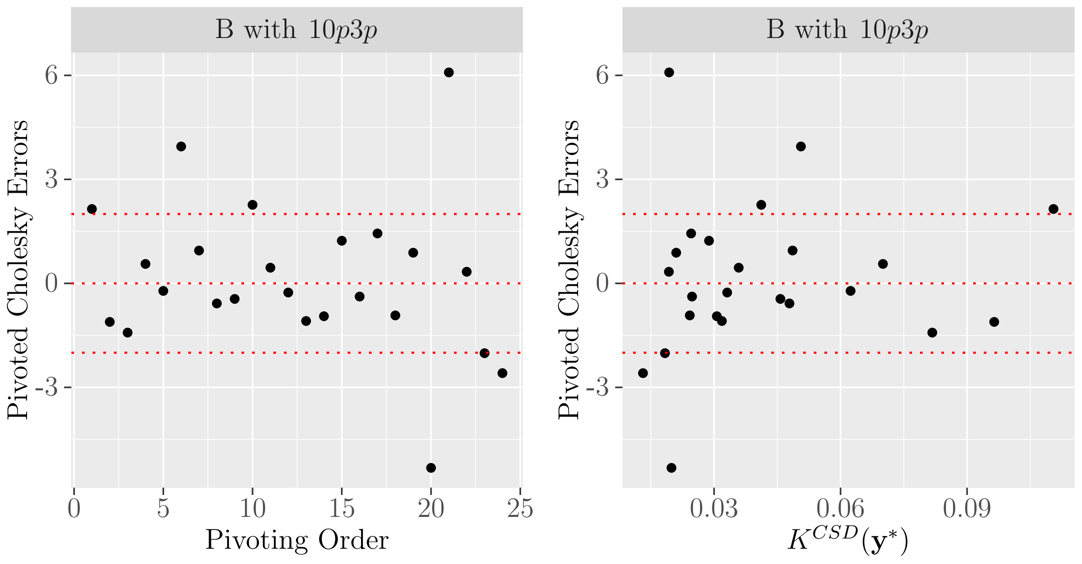

Figure 3.10: Plots of the pivoted Cholesky errors against the pivoting order and the pivoted Cholesky errors against the scaled conditional standard deviations for the emulator of the Borehole simulator with $10p3p$. Note that an increase in pivoting order corresponds to a decrease in conditional standard deviations.

It can be shown that most of the pivoted Cholesky errors are not very large and most of them lie inside the bounds, but several large pivoted Cholesky errors are seen in the plot. The scaled conditional standard deviations, however, can be considered small since most of them are close to zero. We can also notice that the large pivoted Cholesky errors correspond to very small scaled conditional standard deviations. In addition, the pivoted Cholesky errors decrease as the scaled conditional standard deviations increase.

Figure 3.11 shows the plots of the pivoted Cholesky errors against the pivoting order and the pivoted Cholesky errors against the scaled conditional standard deviations, $K^{CSD}(\mathbf{y}^*)$, for the emulator of the Mt simulator using $10p3p$.

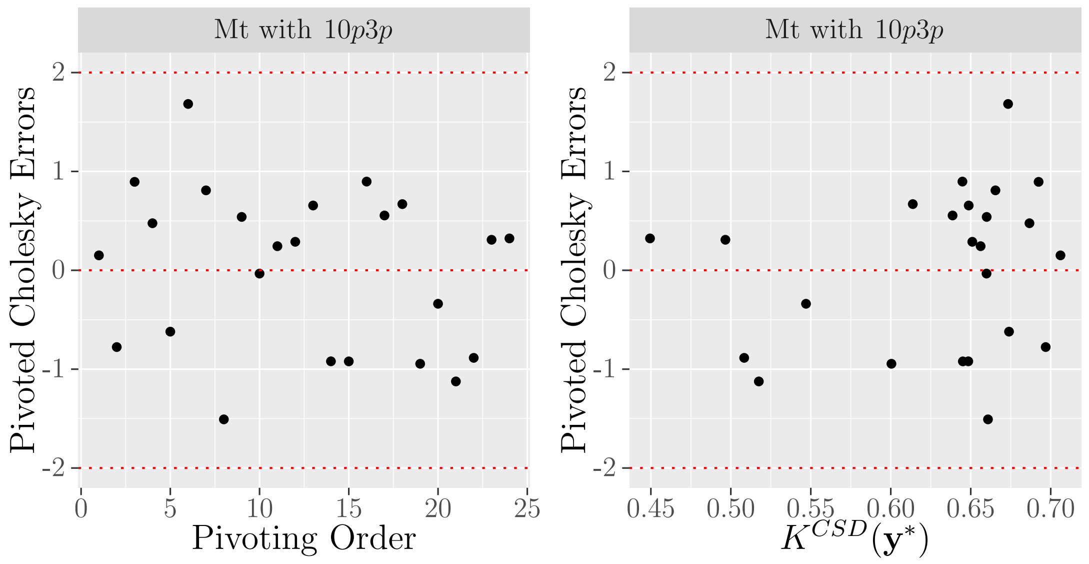

Figure 3.11: Plot of the pivoted Cholesky errors against the pivoting order and the pivoted Cholesky errors against the scaled conditional standard deviations for the emulator of the multivariate t simulator with $10p3p$. The plots show small pivoted Cholesky errors and large scaled conditional standard deviations.

It can be shown that the pivoted Cholesky errors are very small and all of the lie inside the bounds. Although the pivoted Cholesky errors for the Mt simulator are very small, the scaled conditional standard deviations are very large as they are not close to zero.

We can conclude that the plot of the pivoted Cholesky errors against the scaled conditional standard deviations is more informative than the plot of the pivoted Cholesky errors against the pivoting order. This is because the scaled conditional standard deviations show the large widths of the intervals of some emulators.

3.6 Conclusion and recommendations

In this chapter, we have explained the concept of uncertainty calibration. Diagnostic methods have been reviewed for checking assumptions of Gaussian process emulators. We have developed a modification of an existing diagnostic for validating Gaussian process emulators. We have investigated the performance of some current diagnostic methods for different Gaussian process emulators. According to the diagnostics that were applied to our examples, our recommendations are as follows:

- The cross-validation method can be unreliable with a small number of training inputs, in that they may not agree with results from separate validation sets as seen with box-plots of the standardised root mean squared error.
- It is important to use more than one diagnostic method for validating Gaussian process emulators, but not simply two related methods such as the standardised root mean squared error and the predictivity coefficient. This is because these diagnostics measure the same quantity but on different scales.
- For the validation points, we suggest to use two times the number of the input variables, $2p$. We have seen that increasing the number of validation points has a minor impact on improving the performance of diagnostics.
- For the number of the training points, we have noticed that five times the number of input variables was not adequate for good predictions in many

cases. We recommend to use at least ten times the number of the input variables.

- We recommend using scaled conditional standard deviations rather than pivoted order as it is more informative in the plot of the pivoted Cholesky errors.

Chapter 4

A simulation-based method and the coverage interval diagnostic

4.1 Introduction

Diagnostic methods can be used for checking and validating Gaussian process emulators. For some diagnostics, however, it is not possible to derive analytically the required reference distribution, which is the distribution of the observed diagnostic value if the simulator was a realisation from a Gaussian process. In this chapter, we develop a simulation-based method based on simulating samples from the posterior distribution of the output function. This simulation-based method can be used to obtain the reference distribution of diagnostics that cannot be obtained analytically. If the Gaussian process emulator is valid, the observed values of the diagnostic for the validation data will be 'consistent' with the simulated diagnostic values.

In this chapter, we also extend and develop a diagnostic proposed by Bastos and O'Hagan (2009). We develop a graphical presentation that measures the coverage properties of $(1-\alpha)100\%$ posterior credible intervals for the validation outputs. Thus, we can investigate whether the proportion of the credible intervals that contain the validation outputs is as we expect it to be or not. This will help to assess whether the Gaussian process assumption is suitable for building emulators.

In Section 4.2, we develop a coverage interval diagnostic for Gaussian process emulators using separate validation sets. In Section 4.3, we present the coverage interval diagnostic using the cross-validation method. The quantile-quantile plot for testing the normality assumption is reviewed in Section 4.4. In Section 4.5, we present the simulation-based procedure to obtain samples from the distribution of diagnostics. An illustrative example for explaining the coverage interval diagnostic is shown in Section 4.6. An example of data from multivariate Student-t distribution is given in Section 4.7. An example of the coverage interval diagnostic for nonstationary variance simulator is shown in Section 4.8. The conclusion is provided in Section 4.9.

4.2 The coverage interval diagnostic using separate validation sets

In this section, we develop Bastos and O'Hagan's coverage interval diagnostic. Suppose we have a Gaussian process emulator for $f(\cdot)$ based on n training points. Let $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$ be a set of validation inputs. Before evaluating $f(\cdot)$ at these validation inputs, we can obtain a credible interval for each validation output, $y_i^* = f(\mathbf{x}_i^*)$. We aim to investigate the proportion of the credible intervals that contain the validation outputs. Once the true validation outputs are obtained, if the credible intervals for the validation outputs are very wide and the validation outputs always lie inside these credible intervals, we describe the emulator as underconfident. The emulator will be described as overconfident if the credible intervals for the validation outputs are small and the validation outputs always lie outside the credible intervals.

In order to explain the procedure of the coverage interval diagnostic, suppose we have m credible intervals

$$\begin{pmatrix} (L_\alpha(\mathbf{x}_1^*), U_\alpha(\mathbf{x}_1^*)) \\ . \\ . \\ (L_\alpha(\mathbf{x}_m^*), U_\alpha(\mathbf{x}_m^*)) \end{pmatrix}, \tag{4.2.1}$$

where $L_\alpha(\cdot)$ and $U_\alpha(\cdot)$ are the lower and the upper bounds of the $(1-\alpha)100\%$ credible intervals for the validation outputs. Each of these credible intervals should have a probability of $1-\alpha$ of containing the validation outputs, $f(\mathbf{x}_i^*)$. This means that, for a valid emulator, we expect to have

$$p(L_\alpha(\mathbf{x}_i^*) < f(\mathbf{x}_i^*) < U_\alpha(\mathbf{x}_i^*)) = 1-\alpha \tag{4.2.2}$$

for all $i = 1, \ldots, m$. Bastos and O'Hagan (2009) define the credible interval diagnostic, $K_\alpha^{CI}(\cdot)$, to be

$$K_\alpha^{CI}(\mathbf{y}^*) = \frac{1}{m}\sum_{i=1}^{m} I\left\{L_\alpha(\mathbf{x}_i^*) < f(\mathbf{x}_i^*) < U_\alpha(\mathbf{x}_i^*)\right\}, \tag{4.2.3}$$

where $I\{\cdot\}$ is the indicator function. This formula determines the proportion of the validation outputs lying inside the $(1-\alpha)100\%$ credible intervals for the validation outputs. Thus, it helps us to investigate whether or not these credible intervals are meaningful. Bastos and O'Hagan (2009) considered this diagnostic for $(1-\alpha) = 0.95$ and compared this numerical value with its expected value and the lower and upper quartiles.

We consider this credible interval diagnostic for multiple values of $(1-\alpha)$ and develop a graphical presentation to measure the coverage properties of $(1-\alpha)100\%$ posterior credible intervals for the validation outputs. We call this diagnostic the coverage interval diagnostic and it can be used to investigate whether the Gaussian process assumption is suitable for building Gaussian process emulators or not. We also consider how this diagnostic can be used in a cross-validation framework.

The expected value of $K^{CI}_{\alpha}(\cdot)$ is $(1-\alpha)$. If the emulator is valid and the values of the terms in the sum, in equation (4.2.3), are independent, the diagnostic $K^{CI}_{\alpha}(\cdot)$ will have a binomial distribution, $\text{Bin}(m, 1-\alpha)$. However, the terms are not independent, and we cannot analytically derive the distribution of $K^{CI}_{\alpha}(\cdot)$. To illustrate this, suppose we have a one-dimensional simulator with 5 training points as shown in Figure 4.1. Suppose also we have two predicted points, the red points, that are close to each other with their 95% credible intervals as shown in the plot.

Figure 4.1 shows that the simulator does not pass through the 95% credible intervals for the two predicted points. For two predicted points that are close to each other, if the simulator does not pass through the 95% credible interval for the first predicted point, the simulator is not likely to pass through the 95% credible interval for the second predicted point as well, hence the events are not independent.

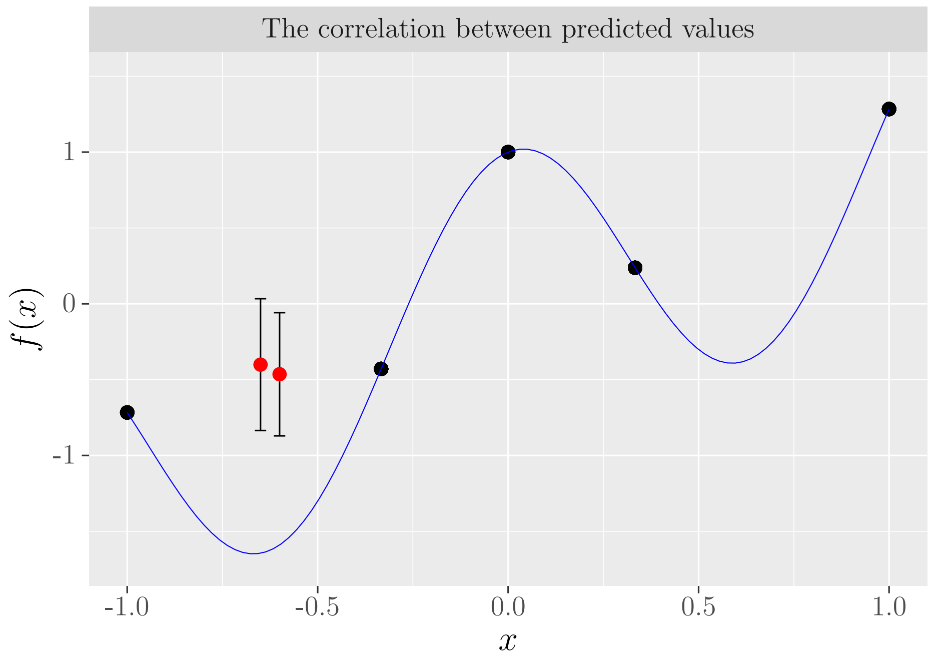

Figure 4.1: A simulator with 5 training points, the black points, and two predicted points, the red points, by a Gaussian process emulator with 95% credible intervals. The plot demonstrates the correlation between the predicted errors. For two predicted points close to each other, if one interval missed the true point, it is likely the other interval to miss the true point as well.

4.3 The coverage interval diagnostic using the cross-validation method

The coverage interval diagnostic can also be used to examine the Gaussian process assumption in Gaussian process emulators using the cross-validation method. The cross-validation method is desirable when we have a small number of available simulator runs and we wish to use all of them as training data. Suppose that simulator outputs are evaluated at training inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$, to obtain training outputs $\mathbf{y} = \{y_1 = f(\mathbf{x}_1), \dots, y_n = f(\mathbf{x}_n)\}$. Therefore, we construct an emulator for $f(\cdot)$ via

$$f(\cdot)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim GP(m(\cdot), V(\cdot, \cdot)), \tag{4.3.1}$$

where $m(\mathbf{x})$ and $V(\mathbf{x}, \mathbf{x}')$ are a prior mean and a prior covariance function given by (2.3.12) and (2.3.13) respectively. For $i = 1, \dots, n$, suppose that $\mathbf{y}_{-i}$ are the outputs of all data except the observation $y_i = f(\mathbf{x}_i)$. For given values of the parameters $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$, the posterior distribution of each output value, $y_i = f(\mathbf{x}_i)$, is a Student-t given by equation (3.4.4).

In the previous section, the emulator is validated using a separate validation sets, so that we have an emulator based on n training points and the distribution of any validation set is a Student-t given by equation (2.3.27). For the cross-validation method, the procedure of validating the emulator is based on removing one point and then predicting it using the other $n-1$ points. Therefore, the emulator validated after removing one point is not the emulator obtained based on the whole set of training points.

In this case, diagnostics for the emulator for $y_i = f(\mathbf{x}_i)$ are based on $\mathbf{y}_{-i}$ and this does not validate the emulator based on the whole training set. However, the simulator outputs are thought to be as samples from the Gaussian process. Thus, it is possible to sample from a multivariate normal distribution with a prior mean and a prior covariance. Then, we test the performance of diagnostics using cross-validation method. If this procedure is repeated with a large number of simulated outputs for each training output, the performance of diagnostics using the cross-validation procedure is expected to be similar of that using new validation sets. According to this procedure for the cross-validation method, we can derive the distribution of diagnostics using these samples and investigate how the data behave if they are drawn from a multivariate Gaussian distribution.

The observed values of the coverage interval diagnostic for the $(1-\alpha)100\%$ credible intervals can be calculated via the cross-validation procedure. Suppose we can obtain n credible intervals for the n output values, $\mathbf{y}$. The coverage interval diagnostic is then given by

$$K_\alpha^{CICV}(\mathbf{y}) = \sum_{i=1}^{n} I\left\{L_{\alpha,-i}(\mathbf{x}_i) < f(\mathbf{x}_i) < U_{\alpha,-i}(\mathbf{x}_i)\right\}, \tag{4.3.2}$$

where $L_{\alpha,-i}(\mathbf{x}_i)$ and $U_{\alpha,-i}(\mathbf{x}_i)$ are the lower and the upper bounds of the $(1-\alpha)100\%$ credible intervals for each output value, $y_i = f(\mathbf{x}_i)$.

4.3.1 Validating the prior assumptions rather than the posterior emulator

In the previous section, we calculate the coverage interval diagnostic using a separate validation set, $K_\alpha^{CI}(\mathbf{y}^*)$. Therefore, we can investigate the distribution of $p(K_\alpha^{CI}(\mathbf{y}^*)|\mathbf{y},\hat{\boldsymbol{\delta}})$. For the cross-validation method, however, we cannot consider the distribution of $p(K_\alpha^{CICV}(\mathbf{y})|\mathbf{y},\hat{\boldsymbol{\delta}})$ as it is determined by $\mathbf{y}$. Thus, we can instead consider what the distribution of the coverage interval diagnostic should be given $\hat{\boldsymbol{\delta}}$ only, $p(K_\alpha^{CICV}(\mathbf{y})|\hat{\boldsymbol{\delta}})$. In this case, we can test the prior normality assumption which means that we test whether the data come from the multivariate normal distribution or not, $\mathbf{y} \sim N(h(\mathbf{x})^T\beta, \sigma^2 A)$.

The performance of the coverage interval diagnostic for the $(1-\alpha)100\%$ credible intervals using the cross-validation method will be comparable with that using a separate validation set. We aim to understand how the coverage interval diagnostic, $K_\alpha^{CICV}(\mathbf{y})$, should behave if the emulator is valid. For a valid emulator, the performance of the coverage interval diagnostic using the cross-validation method will be similar to that using a separate validation set. This helps us to investigate whether the data come from the multivariate normal distribution or not.

In order to investigate whether the data come from the multivariate normal distribution or not, $\mathbf{y} \sim N(h(\mathbf{x})^T\beta, \sigma^2 A)$, values of $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$ need to be chosen. The choice of $\boldsymbol{\beta}$ and σ^2 is not important where any given values of these parameters can be chosen. This is because the distribution of $K_\alpha^{CICV}(\cdot)$ will not be affected if different values of $\boldsymbol{\beta}$ and σ^2 are chosen as will be shown in Appendix 4.A. The choice of the correlation length parameters, however, can make changes in the distribution of $K_\alpha^{CICV}(\cdot)$ as it will affect the correlations between the points. Thus, we used the posterior distributions of the parameters where the values of $\hat{\boldsymbol{\beta}}$ are used in the prior mean and the value of $\hat{\sigma}^2$ with the estimated values of the correlation parameters, $\hat{\boldsymbol{\delta}}$, are used in the prior covariance.

4.3.2 Investigating the distribution of $p(K_\alpha^{CICV}(\mathbf{y})|\hat{\boldsymbol{\delta}})$

The values of the terms in the sum, in equation (4.3.2), are not independent. To illustrate this, suppose we have a one-dimensional simulator with 7 training points as shown in Figure 4.2. Note that there is one outlying training point. Figure 4.2 (a) shows a Gaussian process emulator after removing the outlying point and predicting it conditional on the other six points. It can be seen that the 95% credible intervals are very small and the outlying training point is unlikely to be contained in the 95% credible intervals when it is excluded from the training points.

In Figure 4.2 (b), we returned the outlying point and built a Gaussian process emulator after removing the fourth point. It is shown that there is a significant change in the 95% credible intervals for the outputs. The 95% credible intervals for the neighbouring outputs are now large and are shifted away from the true outputs at the neighbouring inputs. This indicates that the credible intervals depend on the locations of the outputs, so that if an omitted point is outlier, then reintroducing it in the training set is likely to shift away the 95% credible intervals from the true outputs at neighbouring true outputs. This illustrates why the terms in the sum, in equation (4.3.2), are correlated and so the distribution of $K_{\alpha}^{CICV}(\cdot)$ cannot be derived analytically.

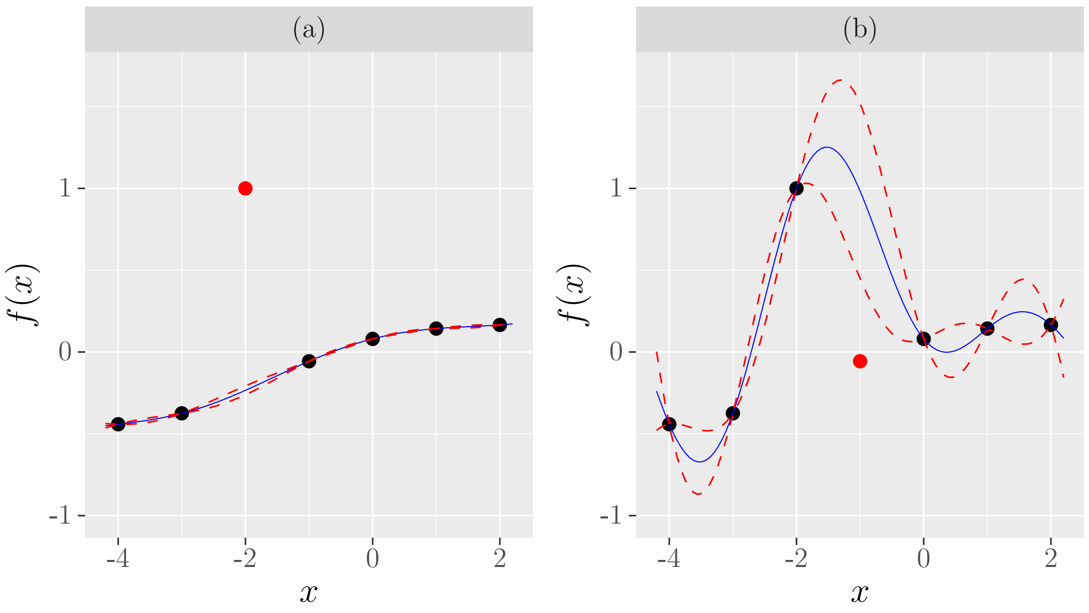

Figure 4.2: Gaussian process emulators with 95% credible intervals using the cross-validation method. The black dots represent the training outputs for emulators; the red dots are the removed points; the solid line is the posterior mean and the dotted lines are the point-wise 95% credible intervals: (a) removing the third training point; (b) removing the fourth training point. The plots demonstrate that including the outlier as a training point, the credible intervals of the neighbouring outputs will be shifted away from the true outputs.

4.4 Quantile-quantile (QQ) plots

The QQ-plot can also be used to assess coverage, but in a different way. The coverage interval diagnostic considers the correlation between the validation outputs and it provides the proportion of the $(1-\alpha)100\%$ credible intervals that contain the validation outputs. Thus, the coverage interval diagnostic provides a direct assessment of the coverage properties of the credible intervals and so we can investigate whether these $(1-\alpha)100\%$ credible intervals are meaningful or not.

In contrast, the QQ-plot is based on the distribution of uncorrelated standardised errors, equation (3.4.15), which is a Student-t distribution with $(n-q)$ degrees of freedom under the Gaussian process assumption. In the QQ-plot, the Gaussian process assumption for the simulator outputs will be reasonable if the points lie close to the $y = x$ line through the origin. An overconfident (or an underconfident) emulator may be indicated if the points cluster around a line with slope greater (or less) than one, but it is harder to assess the extent of the overconfidence (underconfidence).

4.4.1 Example

In this example, we illustrate the difference between the coverage interval diagnostic and the QQ-plot. Suppose we have 100 independent and identically distributed random variables from a standard normal distribution denoted by $\mathbf{y} = \{y_1, \ldots, y_{100}\}$. We can calculate prediction (credible) intervals for these data with different α between 0.1 and 0.95. For example, 90% of the data are expected to be in $(-1.645, 1.645)$. The number of observations that lie inside the interval can be then calculated by the coverage interval diagnostic, equation (4.2.3), with different values of α. Since the observations are independent, the distribution of the coverage interval diagnostic is $\text{Bin}(100, 1-\alpha)$ and hence the intervals of the observed coverage interval diagnostic values can be calculated. Figure 4.3 shows the plot of the coverage interval diagnostic and the QQ-plot with 95% credible intervals for the normal data.

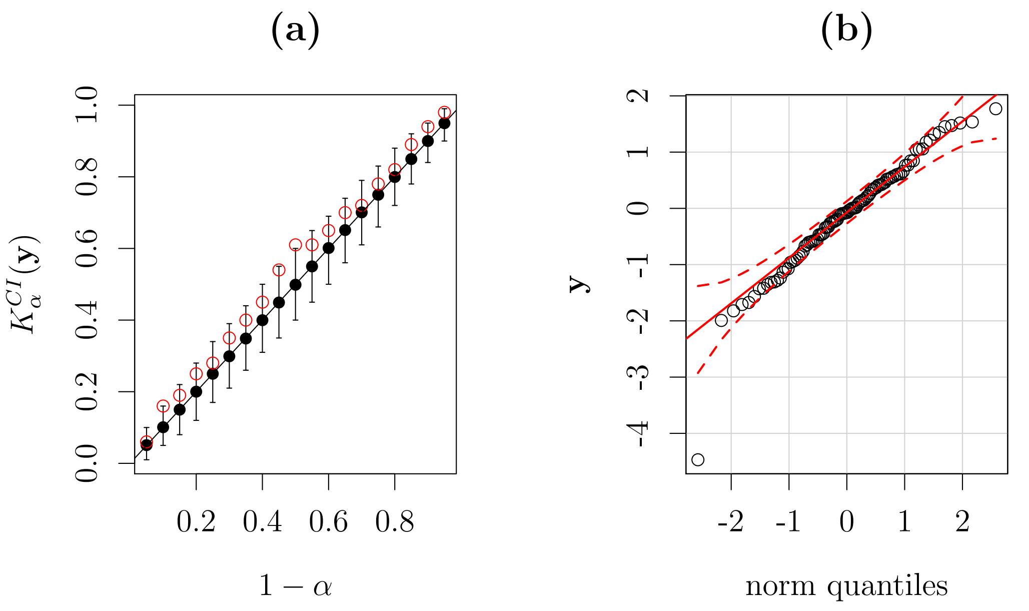

Figure 4.3: The coverage interval diagnostic plot and the QQ-plot for the normal data. Plot (a) shows the observed coverage interval diagnostic values as red points and the mean values of the corresponding simulated values as black points against different values of $(1-\alpha)$ with 95% credible intervals. Plot (b) shows the QQ-plots with 95% credible intervals, obtained using the `car` package, by Fox et al. (2016), in R. The plots suggest that the data come from normal distribution.

Figure 4.3 (a) shows the observed values (as proportions) of the coverage interval diagnostic and their simulated values against different values of $(1-\alpha)$ with 95% credible intervals. It can be shown that only one observed value of the coverage interval diagnostic lies outside the 95% credible intervals, suggesting that a normal distribution assumption is appropriate. Figure 4.3 (b) presents the QQ-plot with 95% credible interval for the standard normal data. It can be seen that most of the points lie close to the $y = x$ line. This also indicates that the data come from normal distribution. However, there is an outlier point that lies outside the 95% credible interval, but this is itself does not imply poor coverage.

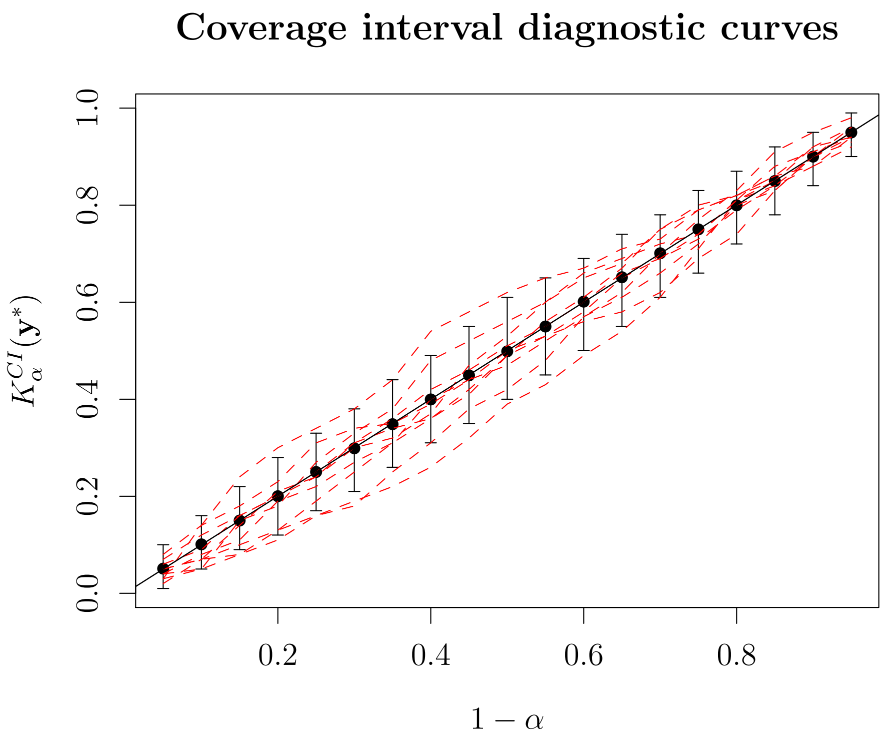

Figure 4.4: Observed coverage interval diagnostic curves for different datasets.

The observed coverage interval diagnostic values for different values of $(1-\alpha)$ are highly correlated as they monotonically non-decreasing with $(1-\alpha)$. Figure 4.4 shows the observed coverage interval diagnostic curves for different datasets, where for each dataset, the observed coverage interval diagnostic values were calculated for different values of $(1-\alpha)$. From Figure 4.4, we can see that the coverage properties are likely to be similar for two values of α that are close to each other. Figure 4.4 also shows that it is possible to have good coverage properties for one value of α, but poor at α' if α and α' further apart.

Now, suppose also we have independent and identically distributed random variables from a Student-t distribution with 3 degree of freedom, denoted by $Y_1, \ldots, Y_{100}$. Assuming normality for these data, we calculated the coverage interval diagnostic, equation (4.2.3), with different values of α. Figure 4.5 shows the plot of the coverage interval diagnostic and the QQ-plot with 95% credible intervals for the Student-t data.

Figure 4.5 (a) shows the observed values (as proportions) of the coverage interval diagnostic and their simulated values against different values of $(1-\alpha)$ with 95% credible intervals. It can be shown that four observed values of coverage interval diagnostic lie outside the 95% credible intervals. This suggests that there is an overconfidence and that the data do not come from a normal distribution.

Figure 4.5 (b) presents the QQ-plot with 95% credible interval for the Student-t data. It can be seen that most of the points cluster around the $y = x$ line, but there are many points which lie outside the 95% credible interval. Whilst suggesting overconfidence, the resulting coverage properties are less clear.

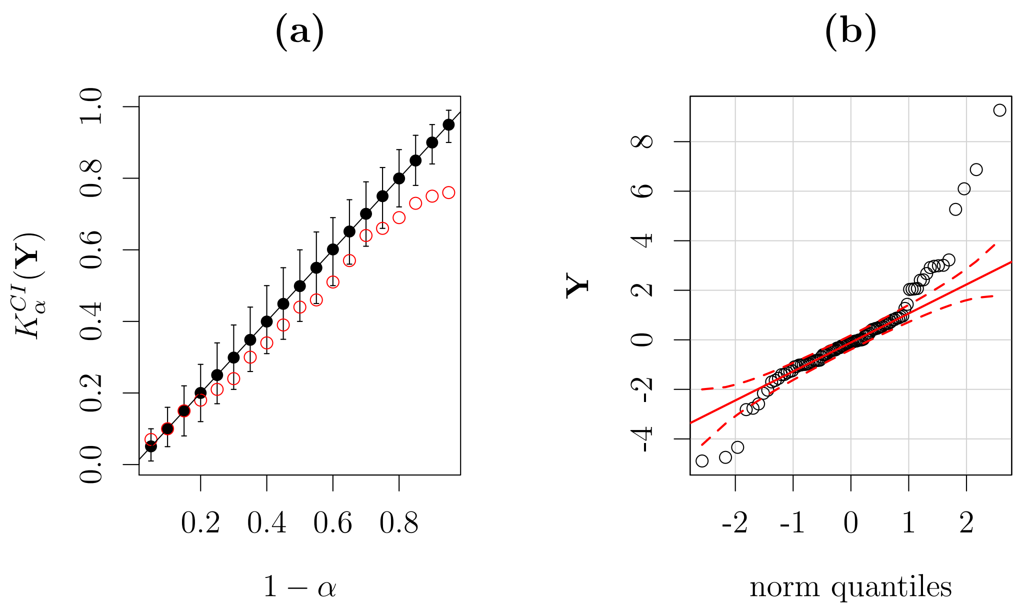

Figure 4.5: The coverage interval diagnostic plot and the QQ-plot for the Student-t data. Plot (a) shows the observed coverage interval diagnostic values as red points and the mean values of the corresponding simulated values as black points against different values of $(1-\alpha)$ with 95% credible intervals. Plot (b) shows the QQ-plots with 95% credible intervals, obtained using the `car` package in R. Plot (a) indicates an overconfidence. Plot (b) shows many points lying outside the intervals.

4.5 Simulation-based method

In this section, we develop a simulation-based method that can be used to determine the required reference distribution of diagnostics. It may be applied when analytical methods are intractable. Bastos and O'Hagan (2009) proposed obtaining the distribution of the coverage interval diagnostic by simulation. We develop a simulation-based method to obtain the distribution of any diagnostic. We also consider how the distribution of the coverage interval diagnostic can be used in a cross-validation framework.

4.5.1 The simulation-based method for diagnostics

The diagnostic, $K(\cdot)$, is a function of the validation outputs, $\mathbf{y}^*$, and the emulator for $f(\cdot)$, and can be thought of as a random variable with a distribution induced by the distribution of $\mathbf{y}^*$. Therefore, it is necessary to investigate the consistency of the observed diagnostic value, $K(\mathbf{y}^*_{obs})$, with its distribution.

For some diagnostics, it is possible to obtain their distribution analytically. For example, the Mahalanobis distance diagnostic, equation (3.4.13), has a scaled F-Snedecor distribution with m and $n-q$ degrees of freedom. For some diagnostics, however, their distribution may not be found analytically. For example, as seen in Sections 4.2 and 4.3, the distribution of the coverage interval diagnostic cannot be found analytically. Hence, we develop a simulation-based method that can be used to obtain the required distribution of such diagnostics.

Suppose we have a Gaussian process emulator for $f(\cdot)$ based on n training inputs, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and their evaluations of the simulator outputs, $\mathbf{y} = \{y_1 = f(\mathbf{x}_1), \ldots, y_n = f(\mathbf{x}_n)\}$. Let $\mathbf{x}^*_1, \ldots, \mathbf{x}^*_m$ be a set of m validation inputs. The procedure of using the simulation-based method to obtain the reference distribution of the diagnostic $K(\cdot)$ can be achieved through the following steps:

- Sample $\mathbf{y}^{*(i)} = \{f_{(i)}(\mathbf{x}^*_1), \ldots, f_{(i)}(\mathbf{x}^*_m)\}$ from the multivariate Student-t distribution with $n-q$ degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$.

- Evaluate $K_{(i)}(\mathbf{y}^{*(i)}) = K(\mathbf{y}^{*(i)}, p(f(\cdot)|\mathbf{y}))$ to obtain one sample $K_{(i)}(\mathbf{y}^{*(i)})$ from the distribution of $K(\cdot)$.

We repeat these steps to obtain N draws, $K_{(1)}(\mathbf{y}^{*(i)}), \ldots, K_{(N)}(\mathbf{y}^{*(N)})$, from the distribution of $K(\cdot)$. According to the procedure above, we can then approximate the required reference distribution of any diagnostic of interest.

4.5.2 The simulation-based method for the coverage interval diagnostic using separate validation sets

In this section, we consider the simulation-based method for the coverage interval diagnostic since its distribution cannot be found analytically. Bastos and O'Hagan (2009) considered the mean and the square of standard deviation of a large number of samples of the coverage interval diagnostic as estimates of the expectation and the variance of the coverage interval diagnostic. They considered the observed value of the coverage interval diagnostic with summaries of its predictive distribution at $(1-\alpha) = 0.95$. We develop a simulation-based method to obtain a graphical presentation of the coverage interval diagnostic distribution. We describe how to obtain the expected value and the 95% credible intervals of the coverage interval diagnostic.

Suppose we have a Gaussian process emulator for $f(\cdot)$ based on n training points. Suppose also we have a set of m validation inputs, $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$. In order to obtain the reference distribution of $K_{\alpha}^{CI}(\cdot)$, the simulation-based method can be used. We first simulate outputs from the posterior distribution of $f(\cdot)$. Then, we calculate the coverage interval diagnostic with different values of α between

0.1 and 0.95. The procedure of using simulation-based method to obtain samples of the diagnostic $K_{\alpha}^{CI}(\cdot)$ can be achieved according to the following algorithm.

Algorithm 1 This algorithm generates samples of the coverage interval diagnostic from its reference distribution

```
Inputs: An emulator, $p\Big(f(\cdot)|f(\mathbf{x}_1),\ldots,f(\mathbf{x}_n)\Big)$.
        $m$ validation inputs, $\mathbf{x}_1^*,\ldots,\mathbf{x}_m^*$.
1: For $i=1$ to $m$, calculate the $(1-\alpha)100\%$ credible intervals for each validation
   output based on the posterior mean, $\mathrm{E}[f(\mathbf{x}_i^*)|\mathbf{y},\boldsymbol{\delta}]$, and the posterior variance,
   $V[f(\mathbf{x}_i^*)|\mathbf{y},\boldsymbol{\delta}]$, that are given in equations (2.3.28) and (2.3.29), where the
   lower and upper bounds of the $(1-\alpha)100\%$ credible intervals are given by
      $$L_\alpha(\mathbf{x}_i^*) = \mathrm{E}[f(\mathbf{x}_i^*)|\mathbf{y},\boldsymbol{\delta}] - t_{n-q;\frac{\alpha}{2}}\sqrt{V[f(\mathbf{x}_i^*)|\mathbf{y},\boldsymbol{\delta}]} \quad (4.5.1)$$
      $$U_\alpha(\mathbf{x}_i^*) = \mathrm{E}[f(\mathbf{x}_i^*)|\mathbf{y},\boldsymbol{\delta}] + t_{n-q;\frac{\alpha}{2}}\sqrt{V[f(\mathbf{x}_i^*)|\mathbf{y},\boldsymbol{\delta}]} \quad (4.5.2)$$
   end for.
2: For $j=1$ to $J$, simulate outputs $\mathbf{y}^{*(j)}=\{f_{(j)}(\mathbf{x}_1^*),\ldots,f_{(j)}(\mathbf{x}_m^*)\}$ from the
   multivariate Student-$t$ distribution with $n-q$ degrees of freedom, the poste-
   rior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}]$, of the emulator.
3: Calculate the coverage interval diagnostic
      $$K_\alpha^{CI}(\mathbf{y}^{*(j)}) = \sum_{i=1}^{m} I\left\{L_\alpha(\mathbf{x}_i^*) < f_{(j)}(\mathbf{x}_i^*) < U_\alpha(\mathbf{x}_i^*)\right\}. \quad (4.5.3)$$
   end for.
Output: $K_\alpha^{CI}(\mathbf{y}^{*(1)}),\ldots,K_\alpha^{CI}(\mathbf{y}^{*(J)})$, a sample from the distribution of
        $K_\alpha^{CI}(\cdot)$ assuming a valid emulator.
```

4.5.3 The simulation-based method for the coverage interval diagnostic using the cross-validation method

In order to obtain the distribution of the coverage interval diagnostic using the cross-validation method, the simulation-based method can be used. First, we sample from the multivariate normal distribution with a prior mean, $H\boldsymbol{\beta}$, and a prior variance, $\sigma^2 A$. Hence, values of $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$ need to be chosen. The choice of $\boldsymbol{\beta}$ and σ^2 is not important where any given values of these parameters can be chosen. This is because the distribution of $K_\alpha^{CICV}(\cdot)$ will not be affected if different values of $\boldsymbol{\beta}$ and σ^2 are chosen as will be shown in Appendix 4.A. The choice of the correlation length parameters, however, can make changes in the distribution of $K_\alpha^{CICV}(\cdot)$ as it will affect the correlations between the points. Thus, we used the posterior distributions of the parameters where the values of $\hat{\boldsymbol{\beta}}$ are used in the prior mean and the value of $\hat{\sigma}^2$ with the estimated values of the correlation parameters, $\hat{\boldsymbol{\delta}}$, are used in the prior covariance.

Then, we obtain n credible intervals for the n simulated output values. We calculate the coverage interval diagnostic for the n simulated output values with different values of α between 0.1 and 0.95. The procedure of using the simulation-based method to obtain the reference distribution of the coverage interval diagnostic can be achieved according to the following algorithm.

Algorithm 2 This algorithm generates samples of the coverage interval diagnostic from its reference distribution using the cross-validation method

```
Inputs: $n$ training inputs, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and $\hat{\boldsymbol{\beta}}, \widehat{\sigma^2}$ and $\hat{\boldsymbol{\delta}}$ from the posterior emulator.
1: For $j = 1$ to $J$, simulate outputs $\mathbf{y}^{(j)} = \{y_1^{(j)} = f_{(j)}(\mathbf{x}_1), \ldots, y_n^{(j)} = f_{(j)}(\mathbf{x}_n)\}$ from the multivariate normal distribution with a prior mean, $H\hat{\boldsymbol{\beta}}$, and a prior variance, $\hat{\sigma}^2 A$.
       $$\mathbf{y}^{(j)} | \boldsymbol{\delta} = \hat{\boldsymbol{\delta}} \sim N_n(H\hat{\boldsymbol{\beta}}, \hat{\sigma}^2 A). \quad (4.5.4)$$
   where $H$ and $A$ are given by equations (2.3.15) and (2.3.16) respectively.
2:   For $i = 1$ to $n$, derive the posterior distribution for the simulated output $y_i = f(\mathbf{x}_i)$ conditional on the other $\mathbf{y}_{-i}^{(j)}$ simulated outputs
       $$f(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \boldsymbol{\delta} = \hat{\boldsymbol{\delta}} \sim \text{Student-}t(n-q-1, m_1(\cdot), V_1(\cdot,\cdot)). \quad (4.5.5)$$
     where $m_1(\cdot)$ and $V_1(\cdot,\cdot)$ are the posterior mean and the posterior variance given by equations (2.3.28) and (2.3.29) respectively based on $\mathbf{y}_{-i}^{(j)}$.
3:   Calculate the $(1-\alpha)100\%$ credible intervals for the simulated output based on the posterior mean, $\mathrm{E}[f_{(j)}(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \hat{\boldsymbol{\delta}}]$, and the posterior variance, $V[f_{(j)}(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \hat{\boldsymbol{\delta}}]$, where the lower and upper bounds of the $(1-\alpha)100\%$ credible intervals are given by
       $$L_{\alpha,-i}(\mathbf{x}_i) = \mathrm{E}[f(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \hat{\boldsymbol{\delta}}] - t_{n-q-1;\frac{\alpha}{2}} \sqrt{V[f(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \hat{\boldsymbol{\delta}}]} \quad (4.5.6)$$
       $$U_{\alpha,-i}(\mathbf{x}_i) = \mathrm{E}[f(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \hat{\boldsymbol{\delta}}] + t_{n-q-1;\frac{\alpha}{2}} \sqrt{V[f(\mathbf{x}_i) | \mathbf{y}_{-i}^{(j)}, \hat{\boldsymbol{\delta}}]} \quad (4.5.7)$$
4:   Calculate the coverage interval diagnostic
       $$K_\alpha^{CICV}(\mathbf{y}^{(j)}) = \sum_{i=1}^{n} I\left\{L_{\alpha,-i}(\mathbf{x}_i) < f_{(j)}(\mathbf{x}_i) < U_{\alpha,-i}(\mathbf{x}_i)\right\}. \quad (4.5.8)$$
   end for.
end for.
Output: $K_\alpha^{CICV}(\mathbf{y}^{(1)}), \ldots, K_\alpha^{CICV}(\mathbf{y}^{(J)})$, a sample from the distribution of $K_\alpha^{CICV}(\cdot)$ assuming a valid emulator.
```

4.6 Illustrative example

In order to illustrate the procedure of the coverage interval diagnostic, we applied the coverage interval diagnostic on emulators built on the Borehole model described in Section 2.6. The aim is to determine whether the Gaussian process assumption is suitable for building emulators. We also applied the QQ-plot on the emulator of the Borehole model in order to investigate whether we can obtain the same information that is obtained by the coverage interval diagnostic.

The design points were generated by a sliced Latin hypercube design (SLHD), proposed by Qian (2012), where we generated 80 design inputs with two slices, each of which has 40 inputs. Before fitting the Gaussian process, each input variable was transformed to be in the interval $[-1, 1]^8$. First, we evaluated simulator outputs of the Borehole model at the 40 inputs of the first slice, $\mathbf{y} = (y_1 = f(\mathbf{x}_1), \dots, y_{40} = f(\mathbf{x}_{40}))$. We then fitted the Gaussian process based on the 40 inputs of the first slice using a linear mean with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4). Then, after obtaining the estimated values of the correlation length parameters, $\hat{\boldsymbol{\delta}}$, we used two methods for validating Gaussian process emulators

Method 1: Using the first slice of the design points as training inputs and the second slice as validation inputs.

Method 2: Performing the cross-validation procedure with the first slice to derive the posterior distribution of each output value, $y_i = f(\mathbf{x}_i)$ given the other 39 output values, $\mathbf{y}_{-i}$.

After obtaining Gaussian process emulators, we calculated the $(1-\alpha)100\%$ credible intervals for each validation output of these two methods. Then, we calculated the observed values of the coverage interval diagnostic for the two methods, equations (4.2.3) and (4.3.2), with different values of α.

In order to obtain the reference distribution of the coverage interval diagnostic, $K_{\alpha}^{CI}(\cdot)$, for the Method 1, the simulation-based method was used. We sampled 1000 vectors of 40 validation outputs from the multivariate Student-t distribution with 31 degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$. Then, we obtained simulated values of coverage interval diagnostic, $K_{\alpha}^{CI}(\mathbf{y}^*_{sim})$, according to Algorithm 1.

The reference distribution of $K_{\alpha}^{CICV}(\cdot)$ for the Method 2, can also be obtained using the simulation-based method. First, we sampled 1000 vectors of 40 'training' outputs from the multivariate normal distribution with a prior mean and a prior covariance. We used $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}, \sigma^2 = \hat{\sigma}^2$ and $\boldsymbol{\delta} = \hat{\boldsymbol{\delta}}$ from the posterior emulator in the prior mean and in the prior covariance. Then, we obtained simulated values of coverage interval diagnostic, $K_{\alpha}^{CICV}(\mathbf{y}_{sim})$, according to Algorithm 2.

We calculated the simulated values of coverage interval diagnostic for the two methods with different values of α. Figure 4.6 shows the observed coverage interval diagnostic values (as proportions) and the mean values of their simulated values against different values of $(1-\alpha)$ with 95% credible intervals for the two methods. We used the 2.5% sample quantile as the lower bound of the 95% credible intervals and the 97.5% sample quantile as the upper bound of the 95% credible intervals.

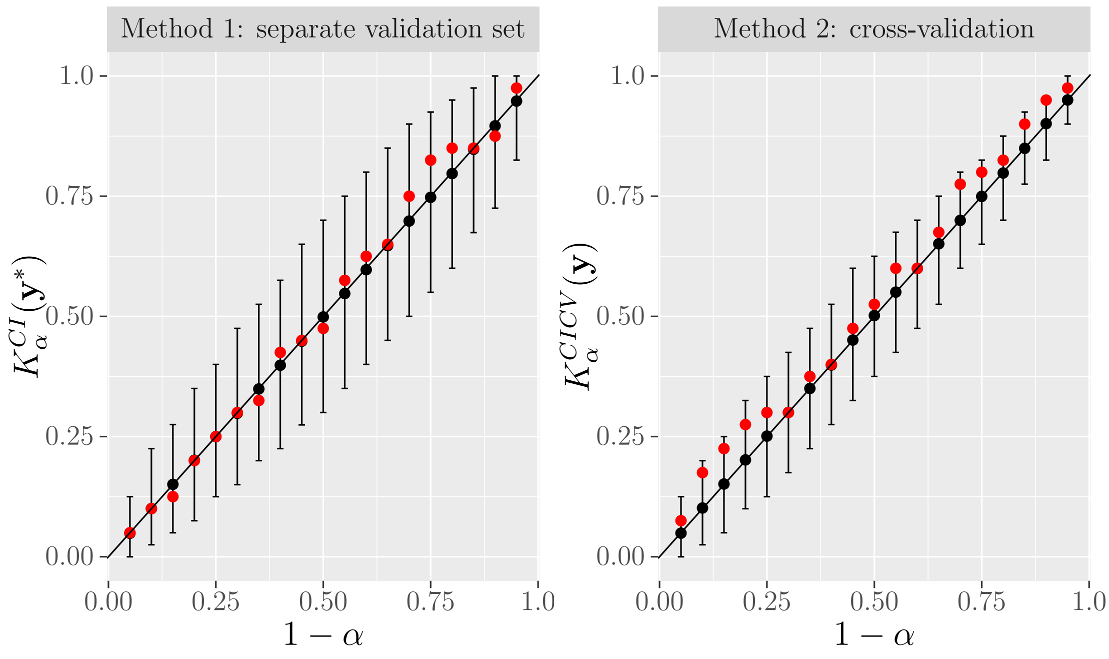

Figure 4.6: The observed coverage interval diagnostic values as red points and the mean values of the corresponding simulated values as black points for the emulator of the Borehole model against different values of $(1-\alpha)$ with 95% credible intervals based on 40 training points and 40 validation points (for Method 1). The plots suggest that the Gaussian process assumption is valid.

The observed values of the coverage interval diagnostic, $K_\alpha^{CI}(\mathbf{y}^*_{obs})$, for the Method 1 are very close to their expectations. Moreover, all of them lie inside the 95% credible intervals, suggesting that the assumption of Gaussian process is suitable for building emulators. For the cross-validation method (Method 2), most of the observed diagnostic values, $K_\alpha^{CICV}(\mathbf{y}_{obs})$, are very close to their expectations. In addition, all the observed values of the coverage interval diagnostic, $K_\alpha^{CICV}(\mathbf{y}_{obs})$, lie inside the 95% credible intervals. This suggests that the Gaussian process assumption is suitable for building emulators and the cross-validation method is also appropriate for validating Gaussian process emulators.

As we can see that the 95% credible intervals of the coverage interval diagnostic, $K_{\alpha}^{CI}(\mathbf{y}^*_{obs})$, for Method 1 are wider than those for Method 2. The coverage interval diagnostic for Method 1 is given by

$$\begin{aligned} K_{\alpha}^{CI}(\mathbf{y}^*) &= \sum_{i=1}^{40} I\left\{L_{\alpha}(\mathbf{x}_i^*) < f(\mathbf{x}_i^*) < U_{\alpha}(\mathbf{x}_i^*)\right\} \\ &= \sum_{i=1}^{40} I\left\{E_i\right\}, \end{aligned}$$

where $E_i = L_{\alpha}(\mathbf{x}_i^*) < f(\mathbf{x}_i^*) < U_{\alpha}(\mathbf{x}_i^*)$ for $i = 1, \dots, 40$. The variance of the coverage interval diagnostic is

$$\mathrm{Var}(K_{\alpha}^{CI}(\mathbf{y}^*)) = \sum_{i=1}^{40} \mathrm{Var}(I\left\{E_i\right\}) + \sum_{i=1}^{40}\sum_{i\neq j}^{40} Cov\Big(I\{E_i\}, I\{E_j\}\Big).$$

Hence, we calculated the sum of covariances between the events $I\{E_i\}$ and $I\{E_j\}$ for $i = 1, \dots, 40$ and $i \neq j$. We chose here, for example, $1-\alpha = 0.95$.

$$\begin{aligned} \mathrm{Var}(\frac{1}{40}\sum_{i=1}^{40} I\{E_i\}) &= \frac{1}{40^2}\sum_{i=1}^{40} \mathrm{Var}(I\left\{E_i\right\}) + \frac{1}{40^2}\sum_{i=1}^{40}\sum_{i\neq j}^{40} Cov\Big(I\{E_i\}, I\{E_j\}\Big) \\ &= \frac{1}{40^2}\sum_{i=1}^{40} \mathrm{Var}(I\left\{E_i\right\}) \\ &+ \frac{1}{40^2}\sum_{i=1}^{40}\sum_{i\neq j}^{40} \Big(\mathrm{E}(I\{E_i\}I\{E_j\}) - \mathrm{E}(I\{E_i\})\mathrm{E}(I\{E_j\})\Big) \\ &= \frac{\alpha(1-\alpha)}{40} + \frac{1}{40^2}\sum_{i=1}^{40}\sum_{i\neq j}^{40} \Big(p(E_i, E_j) - p(E_i)p(E_j)\Big) \\ &= \frac{0.95(1-0.95)}{40} + \frac{1}{40^2}\sum_{i=1}^{40}\sum_{i\neq j}^{40} \Big(p(E_i, E_j) - (0.95)(0.95)\Big) \\ &= 0.0011875 + 0.003991611 \end{aligned}$$

The probability $p\Big(E_i, E_j\Big)$ means that the joint probability for the events E_i and

E_i

$$p\Big(E_i, E_j\Big) \;=\; p\Big(L_\alpha(\mathbf{x}_i^*) < f(\mathbf{x}_i^*) < U_\alpha(\mathbf{x}_i^*) \text{ and } L_\alpha(\mathbf{x}_j^*) < f(\mathbf{x}_j^*) < U_\alpha(\mathbf{x}_j^*)\Big).$$

As we can see the sum of covariances between the events $I\{E_i\}$ and $I\{E_j\}$ for $i = 1, \ldots, 40$ and $i \neq j$ are approximately three times of the variances. Hence, this makes the 95% credible intervals of the coverage interval diagnostic for Method 1 are wider than those for Method 2.

However, it can be shown from Figure 4.6 that the 95% credible intervals for the observed values of the coverage interval diagnostic are wide. This indicates that the number of the validation points is small. Thus, we repeated the process above with 200 design points with two slices, each of which has 100 inputs. Hence, for the Method 1, we have now 100 credible intervals for the $m = 100$ validation outputs of the second slice. For the Method 2, we have 100 credible intervals for the $n = 100$ training outputs of the first slice. Figure 4.7 shows the observed values (as proportions) of the coverage interval diagnostic and the mean values of the corresponding simulated values against different values of $(1 - \alpha)$ with 95% credible intervals based on 100 outputs.

It can be seen that most of the observed values of the diagnostic, $K_\alpha^{CI}(\mathbf{y}_{obs}^*)$, for the Method 1 are very close to their expectations. Moreover, all of the observed values of the diagnostic, $K_\alpha^{CI}(\mathbf{y}_{obs}^*)$, lie inside the 95% credible intervals, but now the 95% credible intervals are less wide than before. For the cross-validation method, the 95% credible intervals now are very small. The observed values of the coverage interval diagnostic, $K_\alpha^{CICV}(\mathbf{y}_{obs})$, are also close to their expectations. There are two observed values of the diagnostic, $K_\alpha^{CICV}(\mathbf{y}_{obs})$, lie outside the 95% credible intervals.

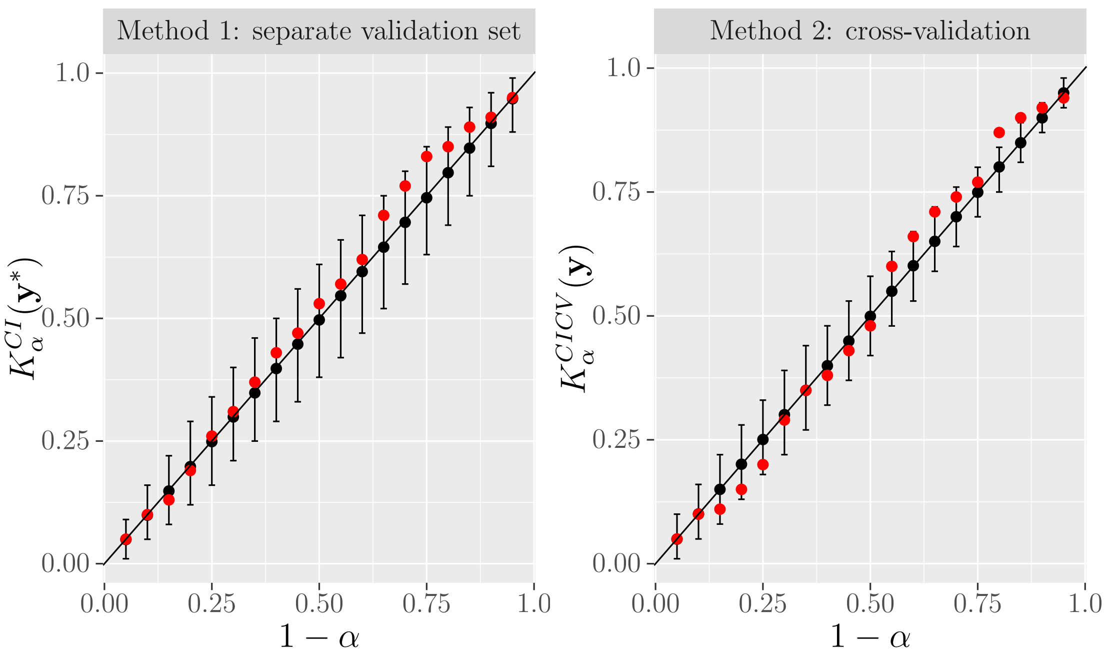

Figure 4.7: The observed values of the coverage interval diagnostic as red points and the mean values of the corresponding simulated values as black points for the emulator of the Borehole model against different values of $(1 - \alpha)$ with 95% credible intervals based on 100 training points and 100 validation points (for Method 1). The plots suggest that the Gaussian process assumption is valid.

In this example, the two methods, the cross-validation method and using separate set of validation points, give broadly the same conclusion, in that the Gaussian process assumption is suitable for emulating the Borehole model.

We also applied the QQ-plots on the emulator of the Borehole model. Figure 4.8 presents the QQ-plot with 95% credible interval of the pivoted Cholesky errors for the emulator of the Borehole model using a separate set of validation points (Method 1).

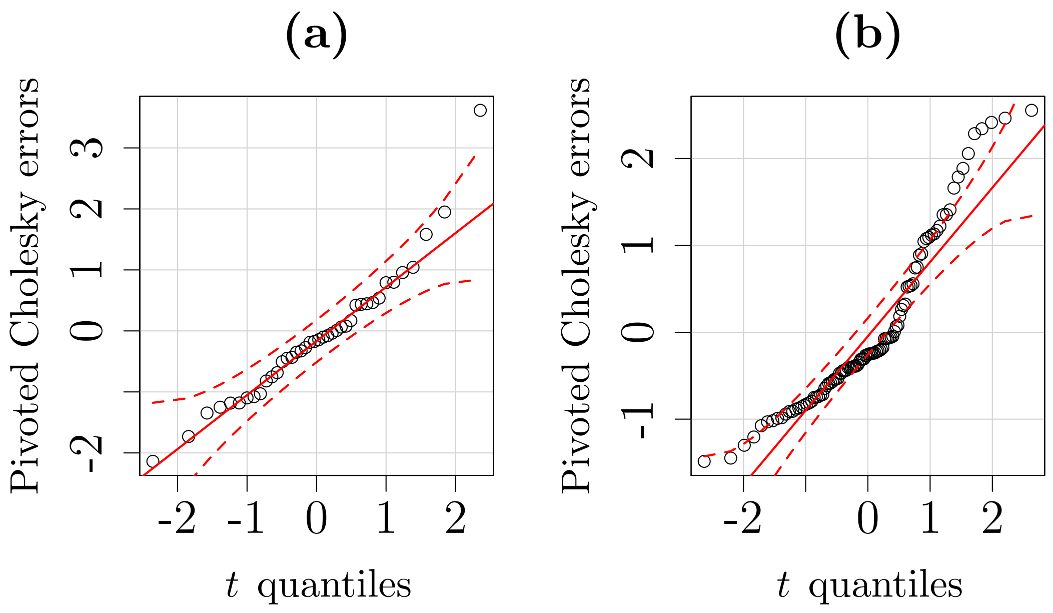

Figure 4.8: The QQ-plots with 95% credible intervals, obtained using the `car` package in R, of the pivoted Cholesky errors for the emulator of the Borehole model using separate validation sets: (a) based on 40 training points and 40 validation points; (b) based on 100 training points and 100 validation points. Figure (a) may indicate that the emulator is underconfident while Figure (a) shows many points lie outside the 95% credible interval.

Figure 4.8 (a) shows the QQ-plot of pivoted Cholesky errors with 40 validation points. It can be shown that most of the points lie close to the $y = x$ line. Moreover, only one point lies outside the 95% credible interval. However, the QQ-plot may indicate that the emulator is underconfident since the points are around a line with slope less than one. Figure 4.8 (b), shows the QQ-plot of the pivoted Cholesky errors with 100 validation points. It can be shown that there are many points which do not lie close to the $y = x$ line and some of them lying outside the 95% credible interval.

It can be concluded that the coverage interval diagnostic plot and the QQ-plot can both be used for examining the Gaussian process assumption for building emulators. However, each plot may provide different information from the other. The coverage interval diagnostic considers the correlation between the validation outputs and it provides a direct assessment of the coverage properties of the credible intervals. The coverage interval diagnostic provides the proportion of the $(1-\alpha)100\%$ credible intervals that contain the validation outputs. Hence, we can investigate whether these $(1-\alpha)100\%$ credible intervals are meaningful.

For the QQ-plot, it is based on the standardised errors distribution which is a Student-t distribution with $(n-q)$ degrees of freedom. The QQ-plot is less informative than the coverage interval diagnostic plot. For example, the point that lies outside the 95 credible intervals in Figure 4.8 (a) and the points that lie outside the 95% credible interval in Figure 4.8 (b) do not provide any clear information.

4.7 The coverage interval diagnostic with data from a multivariate Student-t distribution

The purpose of this example is to examine the performance of the coverage interval diagnostic with data which exhibit a different properties to that of a Gaussian process. The multivariate Student-t distribution has heavy tails, so we aim to investigate whether the coverage interval diagnostic can detect this behaviour. We generated 40 training inputs in the space $[-1,1]^8$ using a LHD, denoted by $\mathbf{X}_1 = (\mathbf{x}_1, \ldots, \mathbf{x}_{40})$, where $\mathbf{x}_i$ is a vector of eight inputs. We then generated 1000 inputs in the space $[-1,1]^8$ using a LHD, denoted by $\mathbf{X}_2 = (\mathbf{x}_{41}, \ldots, \mathbf{x}_{1040})$.

For selected values of the parameters $\boldsymbol{\beta} = (1, -0.5, 2.2, -3.4, -2.2, 1, 3, 0.6, -6.4)$, $\sigma^2 = 3$ and $\boldsymbol{\delta} = (0.8, 3.23, 2.26, 0.09, 0.04, 0.21, 1.05, 1.40)$, we calculated the mean vector, a linear mean function with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$, and the covariance matrix $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4)

$$M = \begin{pmatrix} \mathbf{h}(\mathbf{X}_1)^T\boldsymbol{\beta} \\ \mathbf{h}(\mathbf{X}_2)^T\boldsymbol{\beta} \end{pmatrix}, \tag{4.7.1}$$

$$V = \sigma^2 \begin{pmatrix} C(\mathbf{X}_1, \mathbf{X}_1) & C(\mathbf{X}_1, \mathbf{X}_2) \\ C(\mathbf{X}_2, \mathbf{X}_1) & C(\mathbf{X}_2, \mathbf{X}_2) \end{pmatrix}. \tag{4.7.2}$$

Then, a 1040×1 vector of ‘simulator outputs’, $\mathbf{Y} = (y_1, \ldots, y_{1040})$, was generated from the multivariate Student-t distribution with 3 degrees of freedom, M and V

$$\mathbf{Y} \sim \text{multivariate Student-}t\ (3, M, V). \tag{4.7.3}$$

Thus, $\mathbf{y} = (y_1, \ldots, y_{40})$ are the simulated outputs at the training inputs and $y_{41}, \ldots, y_{1040}$ are the simulated outputs at the 1000 inputs.

4.7.1 Building a Gaussian process emulator

Using the training data, we fitted the Gaussian process using a linear mean, equation (2.3.1), with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4). Then, we derived the Gaussian process emulator.

In order to show the “true validation result”, we applied the coverage interval diagnostic on the emulator of the Multivariate t data using all the 1000 of the

$\mathbf{X}_2$ inputs as validation points. We first calculated the $(1-\alpha)100\%$ credible intervals for each 'simulated output'. Then, we calculated the observed values of the coverage interval diagnostic, equation (4.2.3), with different values of α.

The distribution of the diagnostic $K_{\alpha}^{CI}(\cdot)$ cannot be found analytically, and so the simulation-based method was used. We sampled 1000 vectors of 1000 validation outputs from the multivariate Student-t distribution with 31 degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, of the emulator. Then, simulated coverage interval diagnostic values were obtained according to Algorithm 1 with different values of α. Figure 4.9 shows the observed values (as proportions) of the diagnostic $K_{\alpha}^{CI}(\mathbf{y}^*_{obs})$ with the mean values of the corresponding simulated values against different values of $(1-\alpha)$ with 95% credible intervals using 40 training points and 1000 validation points.

It can be seen that sixteen from nineteen of the observed coverage interval diagnostic values lie under their error bars of the 95% credible intervals. This indicates that the emulator is overconfident. Different samples of size 1040 did not always produce overconfident emulators. The aim of this example, however, is to obtain a dataset where the Gaussian process emulator is overconfident and Figure 4.9 represents a dataset with the desired properties.

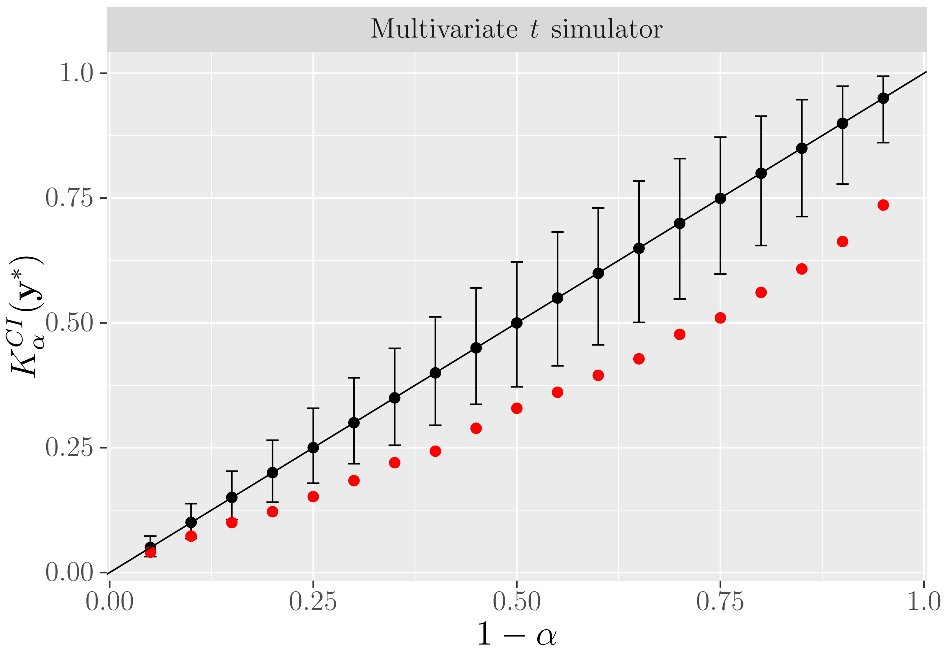

Figure 4.9: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the Multivariate t data against different values of $(1-\alpha)$ with 95% credible intervals based on 40 training points and 1000 validation points. The plot shows an overconfident emulator.

Now we aim to investigate weather or not we can detect the "true validation result" using a small set of validation points or using the cross-validation method. Hence, we consider the two following methods

Method 1: Choosing a random sample of size 40 from the 1000 inputs as validation inputs. We will denote to the validation inputs by $\mathbf{x}_1^*, \ldots, \mathbf{x}_{40}^*$ and so their simulated outputs will be denoted by $\mathbf{y}^* = (y_1^*, \ldots, y_{40}^*)$.

Method 2: Performing the cross-validation procedure with the training points to derive the posterior distribution of each output value, $y_i = f(\mathbf{x}_i)$ given the other 39 output values, $\mathbf{y}_{-i}$.

We then applied the coverage interval diagnostic on the emulator of the Multivariate t data. We first calculated the $(1-\alpha)100\%$ credible intervals for each 'simulated output'. Then, we calculated the observed coverage interval diagnostic values for the two methods, equations (4.2.3) and (4.3.2), with different values of α.

Because the distributions of the diagnostics $K_{\alpha}^{CI}(\cdot)$ and $K_{\alpha}^{CICV}(\cdot)$ cannot be found analytically, we used the simulation-based method to obtain the reference distributions of $K_{\alpha}^{CI}(\cdot)$ and $K_{\alpha}^{CICV}(\cdot)$. For Method 1, we sampled 1000 vectors of 40 validation outputs from the multivariate Student-t distribution with 31 degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y},\boldsymbol{\delta}]$, of the emulator. Then, we obtained simulated values of the coverage interval diagnostic according to Algorithm 1 with different values of α.

For Method 2, we first sampled 1000 vectors of 40 'training' outputs from the multivariate normal distribution with a prior mean and a prior covariance. We used $\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}, \sigma^2=\hat{\sigma}^2$ and $\boldsymbol{\delta}=\hat{\boldsymbol{\delta}}$ from the posterior emulator in the prior mean and in the prior covariance. Then, we obtained simulated values of coverage interval diagnostic, $K_{\alpha}^{CICV}(\mathbf{y}_{sim})$, according to Algorithm 2 with different values of α. Figure 4.10 shows the observed values (as proportions) of the diagnostics $K_{\alpha}^{CI}(\mathbf{y}^*_{obs})$ and $K_{\alpha}^{CICV}(\mathbf{y}_{obs})$ with the mean values of the corresponding simulated values against different values of $(1-\alpha)$ with 95% credible intervals. We used the 2.5% sample quantile as the lower bound of the 95% credible intervals and the 97.5% sample quantile as the upper bound of the 95% credible intervals.

The observed coverage interval diagnostic values for Method 1, $K_{\alpha}^{CI}(\mathbf{y}^*_{obs})$, are far from their expectations with 13 from 19 observed values lying outside the 95% credible intervals (under their error bars). We repeated these process with many different samples of size 40 and we almost always obtained the same result.

This indicates that the emulator is overconfident and the result with Method 1 is robust to different subsets of validation points. For Method 2, it can be seen that all the observed diagnostic values, $K_{\alpha}^{CICV}(\mathbf{y}_{obs})$, are close to their expectations and all of them lie inside their 95% credible intervals. This suggests that the Gaussian process assumption is valid and the cross-validation method did not detect the overconfidence.

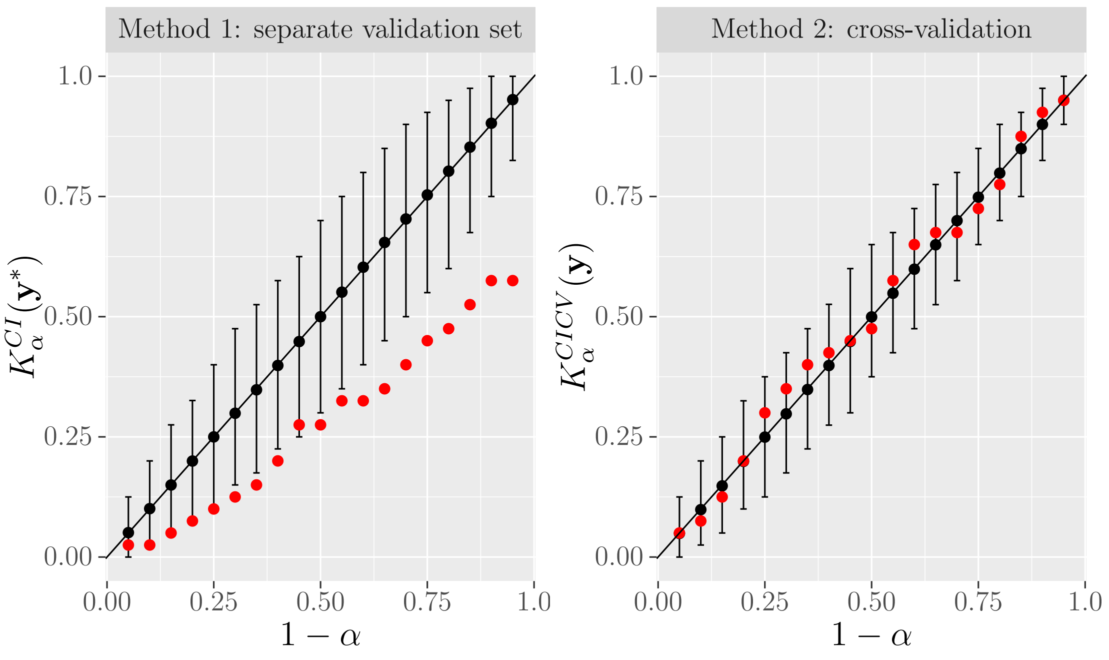

Figure 4.10: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the Multivariate t data against different values of $(1-\alpha)$ with 95% credible intervals based on 40 training points and 40 validation points (for Method 1). The Method 1 plot indicates an overconfident emulator whereas the Method 2 plot indicates the Gaussian process assumption is valid.

We increased the number of the training points to be 80. We also used a

1000 points as validation points in order to show the "true validation result" of the coverage interval diagnostic with 80 training points. Figure 4.11 shows the observed values (as proportions) of the diagnostic $K^{CI}_{\alpha}(\mathbf{y}^*_{obs})$ with the mean values of the corresponding simulated values against different values of $(1-\alpha)$ with 95% credible intervals using 80 training points and 1000 validation points.

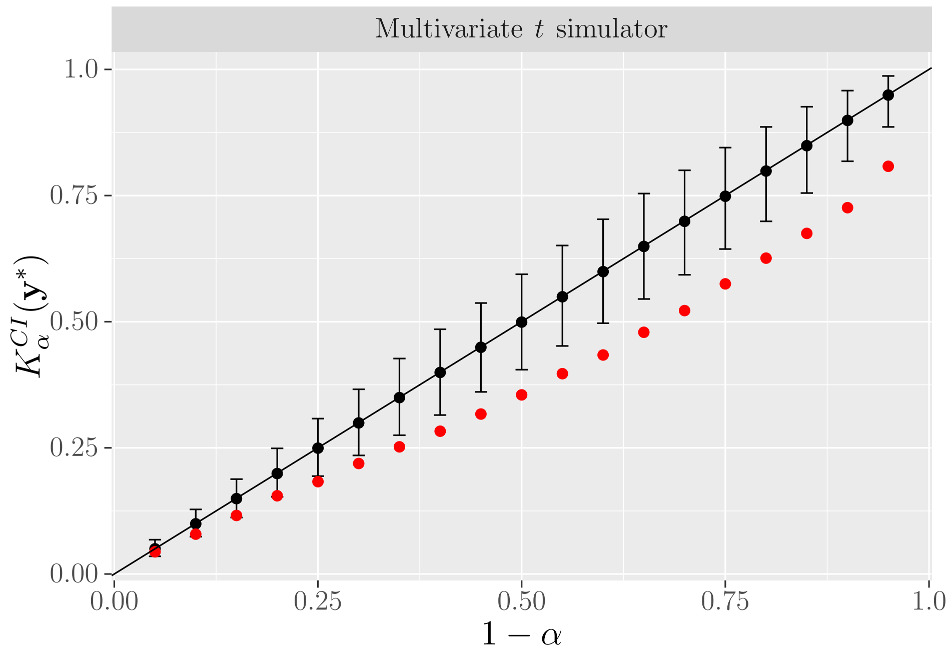

Figure 4.11: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the Multivariate t data against different values of $(1-\alpha)$ with 95% credible intervals based on 80 training points and 1000 validation points. The plot indicates an overconfident emulator.

It can be seen that most of the observed values of the coverage interval diagnostic are far from their expectations and fifteen of them lie under their error bars of the 95% credible intervals, indicating the overconfidence of the emulator.

Now, for the Method 1, we used a sample of 80 points to be validation points and so we have 80 credible intervals for the $m = 80$ validation outputs. For the Method 2, we have 80 credible intervals for the $n = 80$ training outputs. Figure 4.12 shows the observed values (as proportions) of the coverage interval diagnostic and the mean values of the corresponding simulated values against different values of $(1 - \alpha)$ with 95% credible intervals based on 80 training outputs.

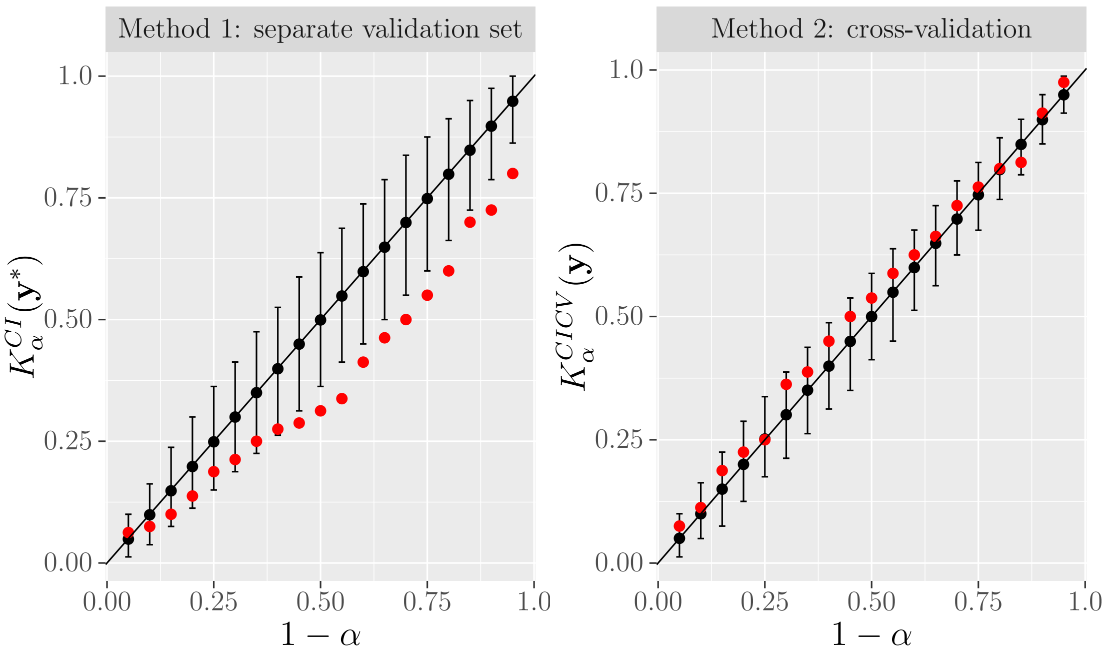

Figure 4.12: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the Multivariate t data against different values of $(1 - \alpha)$ with 95% credible intervals based on 80 training points and 80 validation points (for Method 1). The Method 1 plot indicates an overconfident emulator whereas the Method 2 plot indicates the Gaussian process assumption is valid.

For Method 1, there are 11 from 19 of the observed coverage interval diagnostic values lying outside the 95% credible intervals. We repeated these process with many different samples of size 80 and always obtained the same result, indicating that the emulator is overconfident. The observed values of the coverage interval diagnostic for Method 2 are close to their expectations and all of them lie inside their corresponding 95% credible intervals. This suggests that the Gaussian process assumption is valid.

It can be concluded that the coverage interval diagnostic using a separate validation set was successful in detecting the behaviour of the data whose distribution has heavy tails and has a different properties to that of a Gaussian process. The coverage interval diagnostic showed that the emulator for the multivariate Student-t data is overconfident using different size of validation sets. Hence, the Gaussian process emulator is not appropriate for data generated from a multivariate Student-t distribution. For the cross-validation method, however, the coverage interval diagnostic failed in detecting that the data have a different properties to that of a Gaussian process. This suggests that the cross-validation method for the coverage interval diagnostic is unreliable as it is not sensitive to data that have a different properties to that of a Gaussian process.

We also applied the QQ-plot on the emulator of the Multivariate t data. Figure 4.13 presents the QQ-plot of the pivoted Cholesky errors with 95% credible intervals.

Figure 4.13 (a) shows the QQ-plots with 95% credible interval of the pivoted Cholesky errors with 40 validation points. It can be shown that most of the points cluster around the $y = x$ line and all the points lie inside the 95% credible interval. The QQ-plots may suggest that the Gaussian process assumption is suitable for building the emulator for the Multivariate t data. Figure 4.13 (b) shows the QQ-plots with 95% credible interval of the pivoted Cholesky errors with 80 validation points. It can be shown that most of the points cluster around the $y = x$ line and lie inside the 95% credible interval. However, several points lie outside the 95% credible interval that are not informative. The QQ-plots may suggest that the Gaussian process assumption is suitable for building the emulator for the Multivariate t data and they did not detect that the data have heavy tails and comes from a multivariate Student-t distribution.

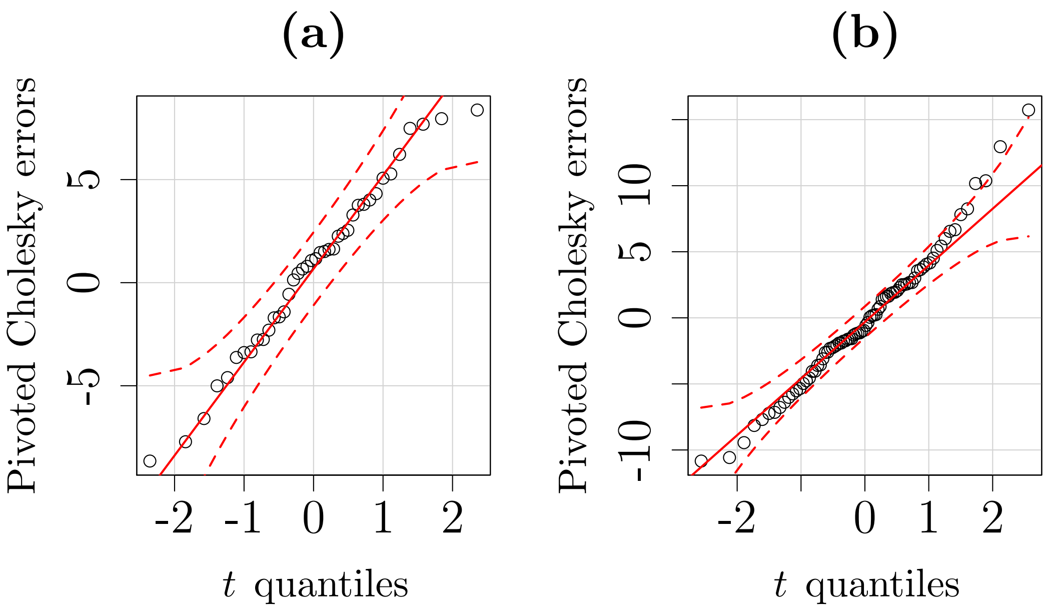

Figure 4.13: The QQ-plot with 95% credible interval, obtained using the `car` package in R, of the pivoted Cholesky errors for the Multivariate t data (a) with 40 training points and 40 validation points (b) with 80 training points and 80 validation points. Plot (a) shows that all the points lie inside the 95% credible intervals whereas plot (b) shows that several points lie outside the 95% credible intervals.

Figure 4.14 presents the QQ-plot of the pivoted Cholesky errors with 95% credible intervals using all the 1000 of the $\mathbf{X}_2$ inputs as validation points. Figure 4.14 (a) shows the QQ-plots with 95% credible interval of the pivoted Cholesky errors based on 40 training points and 1000 validation points. It can be shown that most of the points cluster around the $y = x$ line, but many points lie outside the 95% credible interval. Figure 4.14 (b) shows the QQ-plots with 95% credible interval of the pivoted Cholesky errors based on 80 training points and 1000 validation points. The points also cluster around the $y = x$ line with many points lie outside the 95% credible interval. The points that lie outside the 95% credible interval are not informative and some of the are very extreme.

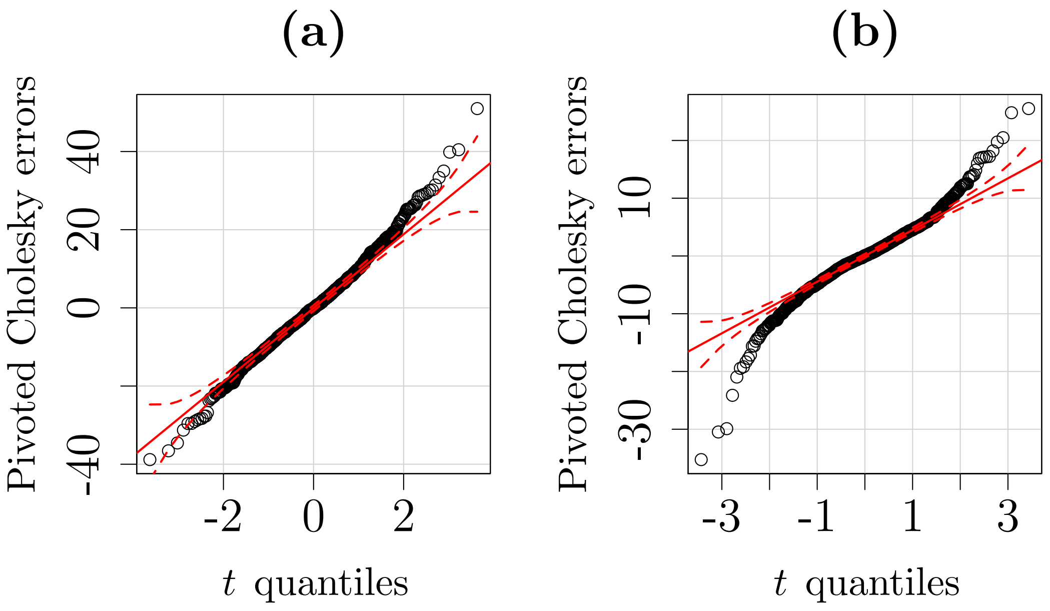

Figure 4.14: The QQ-plot with 95% credible interval, obtained using the `car` package in R, of the pivoted Cholesky errors for the Multivariate t data (a) with 40 training points and 1000 validation points (b) with 80 training points and 1000 validation points. The plots show that many points lie outside the 95% credible intervals.

4.8 Nonstationary variance simulator (NSV)

The aim of this example is to investigate the performance of the coverage interval diagnostic with data that have a nonstationary variance. We generated 40 training inputs in the space $[-1,1]^8$ using a LHD, denoted by $\mathbf{X}_1 = (\mathbf{x}_1, \ldots, \mathbf{x}_{40})$, where $\mathbf{x}_i$ is a vector of eight inputs. We also generated 1000 inputs in the space $[-1,1]^8$ using a LHD, denoted by $\mathbf{X}_2 = (\mathbf{x}_{41}, \ldots, \mathbf{x}_{1040})$.

Assuming values of the parameters $\boldsymbol{\beta} = (5, 10.5, 21.2, 9.4, 0.2, 15, 8, 0.2, 1.8)$ and $\boldsymbol{\delta} = (1.8, 31.23, 2.26, 10.09, 2.04, 0.21, 1.05, 1.4)$, the Gaussian process was specified with a linear mean function, equation (2.3.1), with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4). First, we calculated the mean vector $M = \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta}$ and the correlation matrix, $C = C(\mathbf{x}, \mathbf{x}')$.

$$M = \begin{pmatrix} \mathbf{h}(\mathbf{X}_1)^T \boldsymbol{\beta} \\ \mathbf{h}(\mathbf{X}_2)^T \boldsymbol{\beta} \end{pmatrix}, \tag{4.8.1}$$

$$C = \begin{pmatrix} C(\mathbf{X}_1, \mathbf{X}_1) & C(\mathbf{X}_1, \mathbf{X}_2) \\ C(\mathbf{X}_2, \mathbf{X}_1) & C(\mathbf{X}_2, \mathbf{X}_2) \end{pmatrix}. \tag{4.8.2}$$

Then, the variance was chosen to be a nonstationary where we first chose a function of the inputs defined by

$$g(\mathbf{x}_i) = b_0 + b_1 x_1 + \ldots + b_8 x_8, \tag{4.8.3}$$

where $b_0 = 0.3, b_1 = 0.4, b_2 = 0.2, b_3 = 0.01, b_4 = 0.13, b_5 = 0.01, b_6 = 0.03, b_7 = 0.8, b_8 = 0.12$. Then, a matrix Σ was obtained by

$$\Sigma = \begin{pmatrix} g^2(\mathbf{x}_1) & g(\mathbf{x}_1)g(\mathbf{x}_2) & \cdots & g(\mathbf{x}_1)g(\mathbf{x}_{80}) \\ g(\mathbf{x}_2)g(\mathbf{x}_1) & g^2(\mathbf{x}_2) & \cdots & g(\mathbf{x}_2)g(\mathbf{x}_{80}) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathbf{x}_{80})g(\mathbf{x}_1) & g(\mathbf{x}_{80})g(\mathbf{x}_2) & \cdots & g^2(\mathbf{x}_{80}) \end{pmatrix}. \tag{4.8.4}$$

The choice of equation (4.8.3) results in standard deviations that vary by a factor of 2 over the input space. To obtain the variance matrix, we multiplied the matrix Σ by the matrix C element by element, $V_{ij} = \Sigma_{ij} \times C_{ij}$. Then, a 1040×1 vector of 'simulator outputs', $\mathbf{Y} = (y_1, \dots, y_{1040})$, was generated from the multivariate normal distribution with mean M and covariance matrix V

$$\mathbf{y} \sim N_{80}(M, V). \tag{4.8.5}$$

Thus, $\mathbf{y} = (y_1, \dots, y_{40})$ are the simulated outputs at the training inputs and $y_{41}, \dots, y_{1040}$ are the simulated outputs at the 1000 inputs.

4.8.1 Building a Gaussian process emulator

After obtaining the 40 training outputs from the multivariate normal distribution, a Gaussian process emulator was derived using a linear mean, equation (2.3.1), with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4).

In order to show the "true validation result" for the emulator of the NSV simulator, the coverage interval diagnostic was applied using all the 1000 of the $\mathbf{X}_2$ inputs as validation points. First, the $(1-\alpha)100\%$ credible intervals were calculated for the 'simulated output'. Then, we calculated the observed values of the coverage interval diagnostic, equation (4.2.3), with different values of α.

The simulation-based method was used to obtain the distribution of $K_\alpha^{CI}(\cdot)$. We sampled 1000 vectors of 1000 validation outputs from the multivariate Student-t distribution with 31 degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, of the emulator. Then, simulated values, $K_\alpha^{CI}(\mathbf{y}^*_{sim})$, were obtained according to Algorithm 1 with different values of α. Figure 4.15 shows the observed values (as proportions) of the diagnostics $K_{\alpha}^{CI}(\mathbf{y}^*_{obs})$ with the mean values of the corresponding simulated values against different values of $(1-\alpha)$ with 95% credible intervals using 40 training points and 1000 validation points.

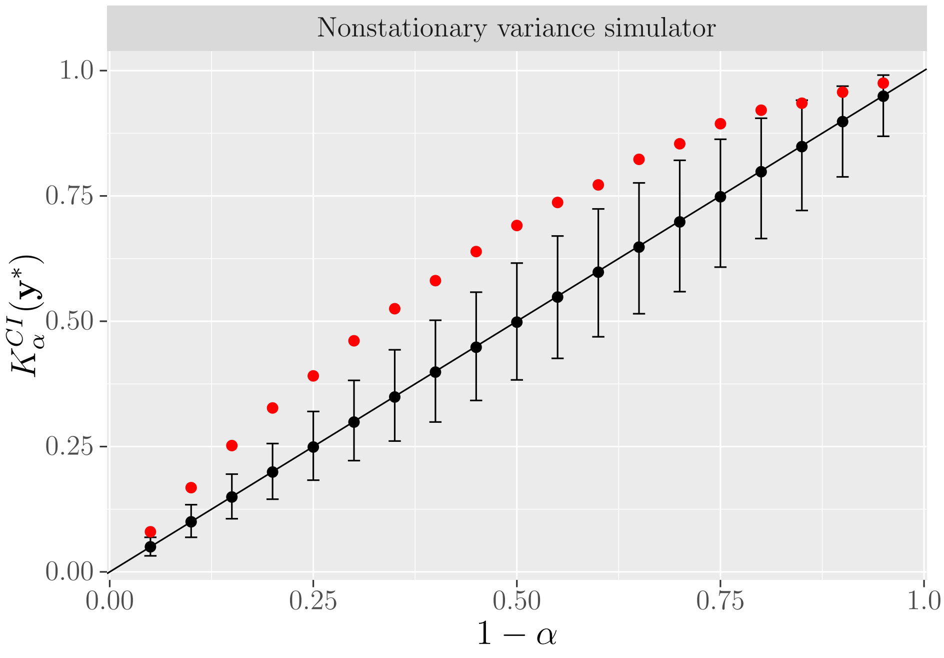

Figure 4.15: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the NSV simulator against different values of $(1-\alpha)$ with 95% credible intervals based on 40 training points and 1000 validation points. The plot shows an underconfident emulator.

It can be seen that 16 observed values from 19 of the coverage interval diagnostic lie above their error bars of the 95% credible intervals, indicating that the emulator is underconfident. Different samples of size 1040 did not always indicate underconfident emulators. However, we aim in this example to obtain a dataset where the Gaussian process emulator is underconfident and Figure 4.15

represents a dataset with the desired properties.

A small set of validation points and the cross-validation method were used then to investigate weather we can detect the “true validation result”. Hence, we considered the two following methods as in the previous section

Method 1: Choosing a random sample of size 40 from the 1000 inputs as validation inputs. We will denote to the validation inputs by $\mathbf{x}_1^*, \ldots, \mathbf{x}_{40}^*$ to be distinct from the training inputs and so their simulated outputs will be denoted by $\mathbf{y}^* = (y_1^*, \ldots, y_{40}^*)$.

Method 2: Performing the cross-validation procedure with the first slice to derive the posterior distribution of each output value, $y_i = f(\mathbf{x}_i)$ given the other 39 output values, $\mathbf{y}_{-i}$.

Then, we applied the coverage interval diagnostic on the emulator of the NSV simulator for these two methods. We first calculated the $(1-\alpha)100\%$ credible intervals for each ‘simulated output’. We then calculated the observed coverage interval diagnostic values for the two methods, equations (4.2.3) and (4.3.2), with different values of α. The simulation-based method was used to obtain the reference distributions of $K_\alpha^{CI}(\cdot)$ and $K_\alpha^{CICV}(\cdot)$. For Method 1, we sampled 1000 vectors of 40 validation outputs from the multivariate Student-t distribution with 31 degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, of the emulator. Then, simulated values, $K_\alpha^{CI}(\mathbf{y}_{sim}^*)$, were obtained according to Algorithm 1 with different values of α.

For Method 2, we first sampled 1000 vectors of 40 ‘training’ outputs from the multivariate normal distribution with a prior mean and a prior covariance. We used $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}, \sigma^2 = \hat{\sigma}^2$ and $\boldsymbol{\delta} = \hat{\boldsymbol{\delta}}$ from the posterior emulator in the prior mean and in the prior covariance. Then, we obtained simulated values of the coverage interval diagnostic, $K_\alpha^{CICV}(\mathbf{y}_{sim})$, according to Algorithm 2. Figure

4.16 shows the observed values (as proportions) of the diagnostics $K_\alpha^{CI}(\mathbf{y}^*_{obs})$ and $K_\alpha^{CICV}(\mathbf{y}_{obs})$ with the mean values of the corresponding simulated values against different values of $(1-\alpha)$ with 95% credible intervals. We used the 2.5% sample quantile as the lower bound of the 95% credible intervals and the 97.5% sample quantile as the upper bound of the 95% credible intervals.

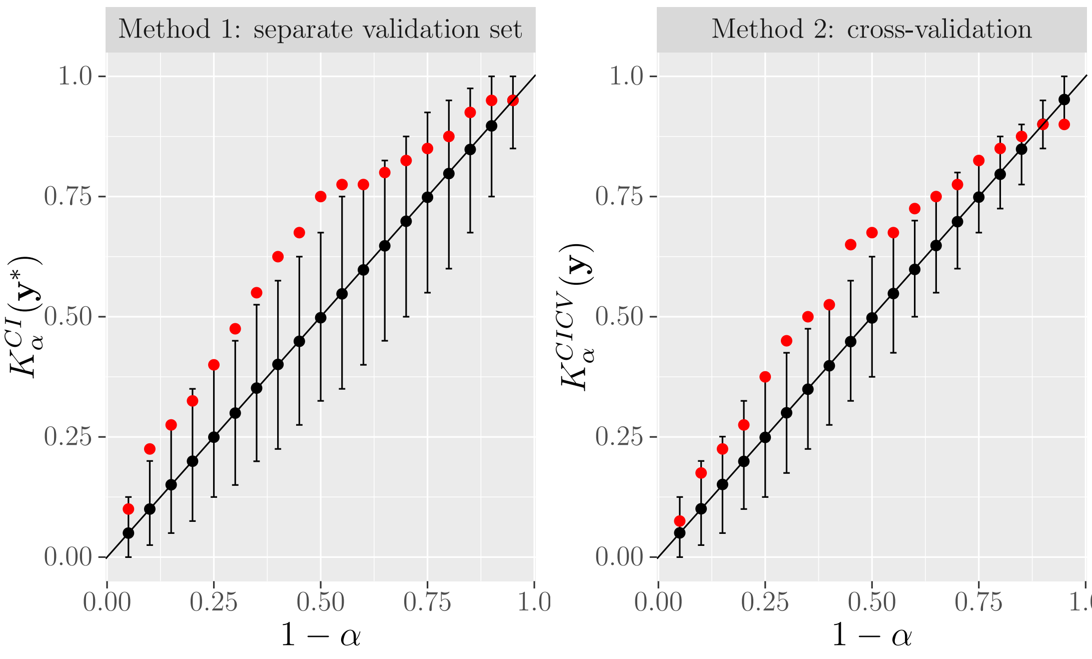

Figure 4.16: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the NSV simulator against different values of $(1-\alpha)$ with 95% credible intervals based on 40 training points and 40 validation points (for Method 1). The plots indicates underconfident emulators.

It can be seen that 7 from 19 observed coverage interval diagnostic values with separate validation points lie outside the 95% credible intervals (above the error bars). This indicates that the emulator is underconfident. For Method 2, it can be seen that 5 observed diagnostic values, $K_\alpha^{CICV}(\mathbf{y}_{obs})$, lie above the error bars of the 95% credible intervals. This also suggests that the cross-validation method detected the underconfidence of the Gaussian process emulator.

We then increased the number of the training inputs to be 80, $\mathbf{X}_1 = (\mathbf{x}_1, \ldots, \mathbf{x}_{80})$. We also used a 1000 points as validation points in order to show the "true validation result" of the coverage interval diagnostic with 80 training points. Figure 4.17 shows the observed values (as proportions) of the diagnostic $K_\alpha^{CI}(\mathbf{y}^*_{obs})$ with the mean values of the corresponding simulated values against different values of $(1-\alpha)$ with 95% credible intervals.

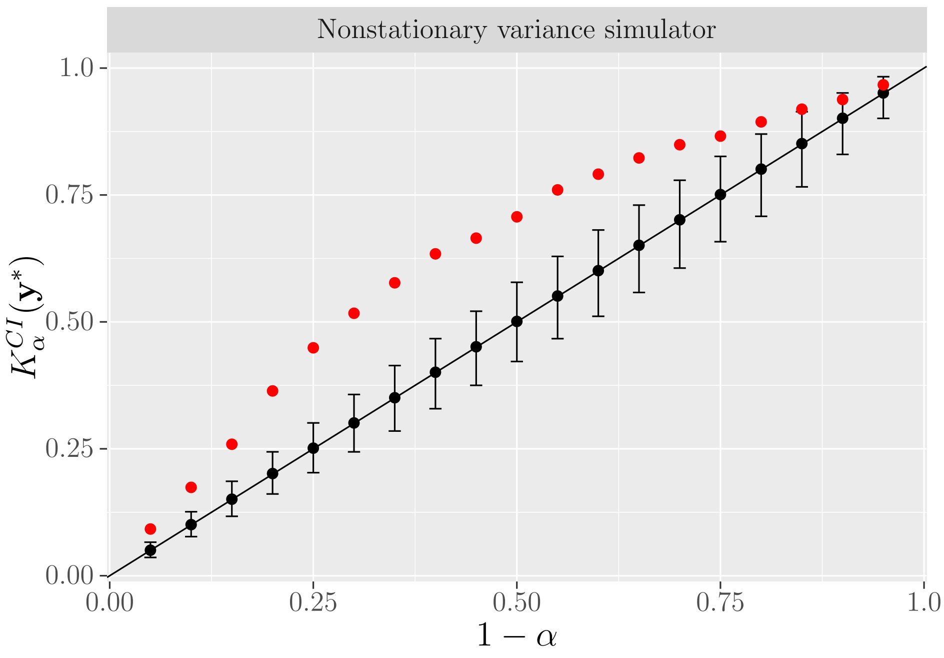

Figure 4.17: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the NSV simulator against different values of $(1-\alpha)$ with 95% credible intervals based on 80 training points and 1000 validation points. The plot indicates an underconfident emulator.

It can be seen that most of the observed coverage interval diagnostic values are far from their expectations and sixteen from nineteen lie above their error bars of the 95% credible intervals, indicating the underconfidence of the emulator.

Now, for the Method 1, we used a sample of 80 points to be validation points and so we have 80 credible intervals for these validation outputs. For the Method 2, we have 80 credible intervals for the $n = 80$ training outputs. Figure 4.18 shows the observed values (as proportions) of the coverage interval diagnostic and the mean values of the corresponding simulated values against different values of $(1 - \alpha)$ with 95% credible intervals based on 80 training outputs.

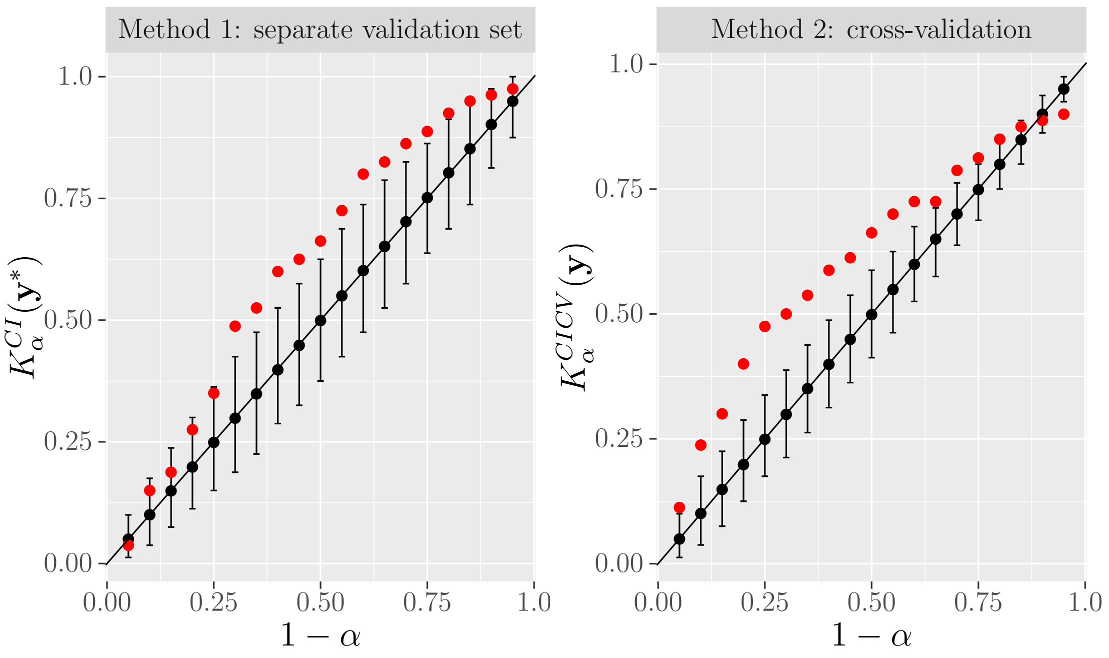

Figure 4.18: The observed values of the coverage interval diagnostic, as red points, and the mean values of the corresponding simulated values, as black points, for the emulator of the NSV simulator against different values of $(1 - \alpha)$ with 95% credible intervals based on 80 training outputs and 80 validation points. The plots indicate underconfident emulators.

For Method 1, there are 11 from 19 of the observed coverage interval diagnostic values lying outside the 95% credible intervals. There are 15 observed values of the coverage interval diagnostic for Method 2 lie above the 95% credible intervals. The two plots suggests that the Gaussian process emulator is underconfident.

It can be concluded that the coverage interval diagnostic using separate validation set and the cross-validation method were successful in detecting the behaviour of the data. The coverage interval diagnostic using the two methods showed that the emulator for the NSV simulator is underconfident.

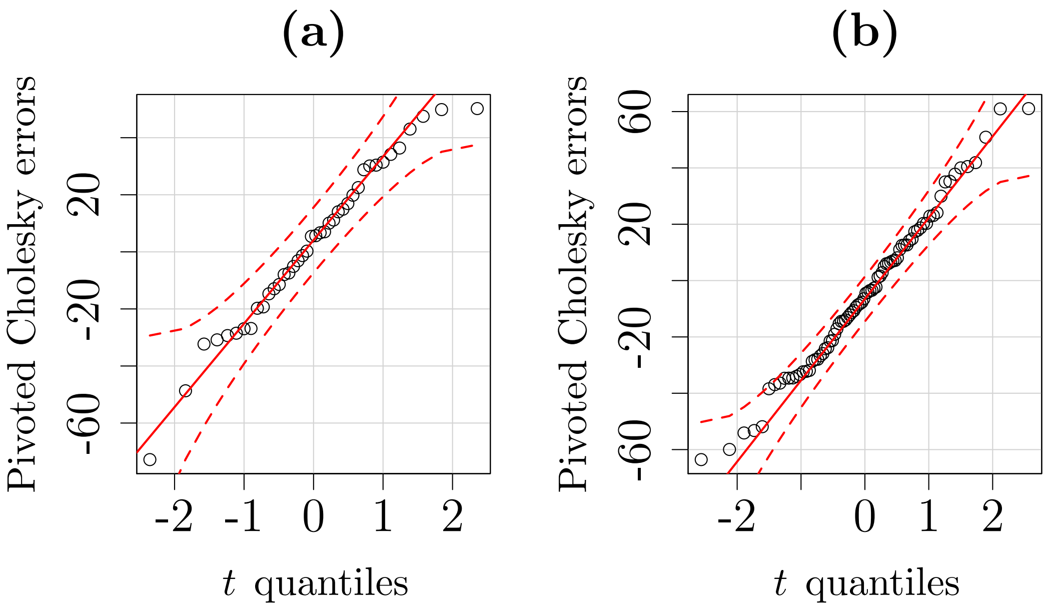

Figure 4.19: The QQ-plot with 95% credible interval, obtained using the `car` package in R, of the pivoted Cholesky errors for the emulator of the NSV simulator (a) with 40 training points and 40 validation points (b) with 80 training points and 80 validation points. The plots show that all the points lie inside the 95% credible intervals.

We also applied the QQ-plot on the emulator of the NSV simulator. Figure 4.19 (a) and (b) present the QQ-plots of the pivoted Cholesky errors with 95% credible intervals using 40 and 80 validation points. It can be shown that all the points cluster around the $y = x$ line and all of them lie inside the 95% credible interval. The QQ-plots may suggest that the Gaussian process emulator is overconfident since the points are around a line with slope greater than one.

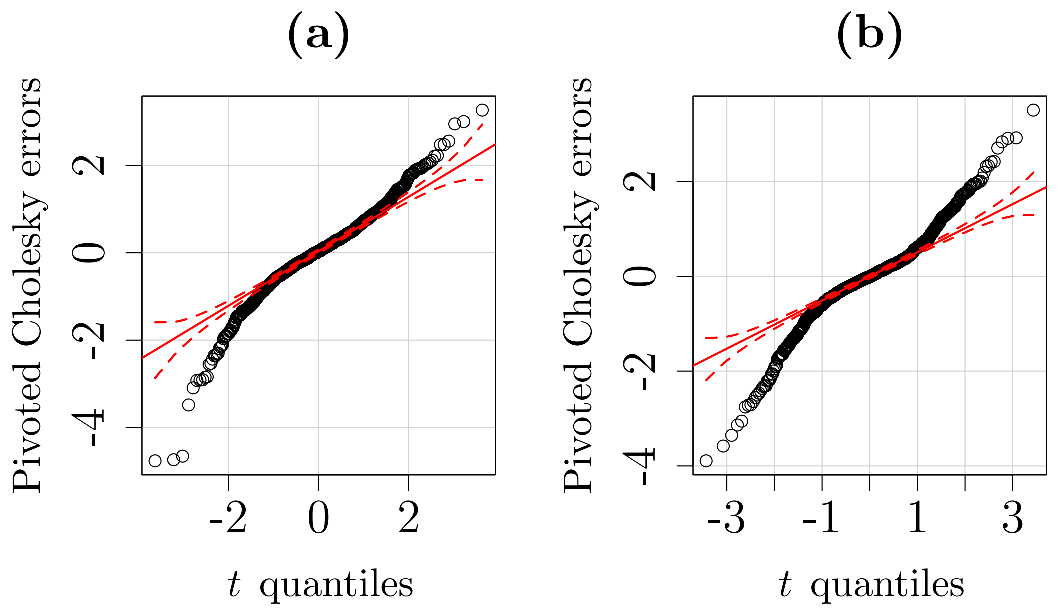

Figure 4.20: The QQ-plot with 95% credible interval, obtained using the `car` package in R, of the pivoted Cholesky errors for the emulator of the NSV simulator (a) based on 40 training points and 1000 validation points (b) based on 80 training points and 1000 validation points. The plots show that many points lie outside the 95% credible intervals.

Using 1000 validation points, Figure 4.20 (a) and (b) present the QQ-plot of the pivoted Cholesky errors with 95% credible intervals based on 40 and 80 training points. It can be shown that many points do not cluster around the $y = x$ line and lie outside the 95% credible interval. The points that lie outside the 95% credible interval are not informative and some of them are extreme.

4.9 Conclusion

In this chapter, we have developed the coverage interval diagnostic for checking the Gaussian process assumption of Gaussian process emulators. We have presented the procedure of simulation-based approach for generating samples from the distribution of diagnostics using separate validation sets and the cross-validation method. We have applied the coverage interval diagnostic on different examples using both separate validation sets and the cross-validation method. It has been shown how the coverage interval diagnostic using separate validation sets can detect the data that have a different properties to that of a Gaussian process.

However, the coverage interval diagnostic with the cross-validation method can be less reliable. This is because it was not successful in detecting that the Gaussian process emulator is not appropriate for data from a multivariate Student-t distribution as it have heavy tails and have a different properties to that of a Gaussian process. Hence, we recommend to use the coverage interval diagnostic to investigate the Gaussian process assumption in building Gaussian process emulators using separate validation sets as it is more reliable than with the cross-validation method.

4.A Appendix A

4.A.1 Using simulation to estimate the distribution of $K_\alpha^{CICV}(\cdot)$

In this appendix, we use simulation to estimate the distribution of the coverage interval diagnostic, $K_\alpha^{CICV}(\cdot)$, using the cross-validation method when $f(\cdot)$ is a sample from a Gaussian process. We want to show that $K_\alpha^{CICV}(\cdot)$ has the same distribution for any values of $\boldsymbol{\beta}$ and σ^2. Suppose we a have a vector, $\mathbf{y} \sim N_n(H\boldsymbol{\beta}, \sigma^2 A)$. For $i = 1, \ldots, n$, suppose that $\mathbf{y}_{-i}$ are all the simulated outputs except the observation $y_i = f(\mathbf{x}_i)$. We derive the posterior distribution for the simulated output $y_i = f(\mathbf{x}_i)$ conditional on the other $\mathbf{y}_{-i}$ simulated outputs which is a Student-t given by equation (3.4.4).

Then, we can obtain n credible intervals for the n output values, $\mathbf{y}$

$$\begin{pmatrix} (L_{\alpha,-i}(\mathbf{x}_1), U_{\alpha,-i}(\mathbf{x}_1)) \\ . \\ . \\ (L_{\alpha,-i}(\mathbf{x}_n), U_{\alpha,-i}(\mathbf{x}_n)) \end{pmatrix},$$

where

$$\begin{aligned} L_{\alpha,-i}(\mathbf{x}_i) &= \mathrm{E}[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] - t_{n-q-1;\frac{\alpha}{2}} \sqrt{V[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}]} \\ U_{\alpha,-i}(\mathbf{x}_i) &= \mathrm{E}[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] + t_{n-q-1;\frac{\alpha}{2}} \sqrt{V[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}]} \end{aligned}$$

with

$$\begin{aligned} \mathrm{E}[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] &= \mathbf{h}(\mathbf{x}_i)^T\hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}(\mathbf{y}_{-i} - H_{-i}\hat{\boldsymbol{\beta}}), \\ V[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] &= \hat{\sigma}^2\Big[C(\mathbf{x}_i, \mathbf{x}'_i; \boldsymbol{\delta}) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}\mathbf{t}(\mathbf{x}'_i) \\ &+ (\mathbf{h}(\mathbf{x}_i) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1} H_{-i})(H_{-i}^T A_{-i}^{-1} H_{-i})^{-1}(\mathbf{h}(\mathbf{x}'_i) - \mathbf{t}(\mathbf{x}'_i)^T A_{-i}^{-1} H_{-i})^T\Big]. \end{aligned}$$

Now suppose we have a different distribution $\tilde{\mathbf{y}} \sim N_n(H(\boldsymbol{\beta}+\boldsymbol{\alpha}), \tau^2 A)$, i.e. we will use $K_\alpha^{CICV}(\cdot)$ with different mean and variance (but not correlation) parameters. This can be achieved by a transformation

$$\tilde{\mathbf{y}} = a\mathbf{y} + \mathbf{b}$$

with $a = \frac{\tau}{\sigma}$ and $\mathbf{b} = -\frac{\tau}{\sigma}H\boldsymbol{\beta} + H(\boldsymbol{\beta}+\boldsymbol{\alpha})$, so that $\mathrm{E}[\tilde{\mathbf{y}}] = H(\boldsymbol{\beta}+\boldsymbol{\alpha})$ and $\mathrm{Var}[\tilde{\mathbf{y}}] = \tau^2 A$. We now obtain the posterior distribution for the simulated output $\tilde{y}_i$ conditional on the other $\tilde{\mathbf{y}}_{-i}$ which is

$$\tilde{y}_i|\tilde{\mathbf{y}}_{-i} \quad \sim \quad \text{Student-}t(n-q-1, \tilde{m}_i(\mathbf{x}), \tilde{V}_i(\mathbf{x},\mathbf{x}))$$

with

$$\begin{aligned}
\mathrm{E}[\tilde{y}_i|\tilde{\mathbf{y}}_{-i},\boldsymbol{\delta}] &= \mathbf{h}(\mathbf{x}_i)^T\hat{\tilde{\boldsymbol{\beta}}} + \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}(\tilde{\mathbf{y}}_{-i} - H_{-i}\hat{\tilde{\boldsymbol{\beta}}}),\\
V[\tilde{y}_i|\tilde{\mathbf{y}}_{-i},\boldsymbol{\delta}] &= \hat{\tilde{\sigma}}^2\Big[C(\mathbf{x}_i,\mathbf{x}_i';\boldsymbol{\delta}) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}\mathbf{t}(\mathbf{x}_i')\\
&+ (\mathbf{h}(\mathbf{x}_i) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}H_{-i})(H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}(\mathbf{h}(\mathbf{x}_i') - \mathbf{t}(\mathbf{x}_i')^T A_{-i}^{-1}H_{-i})^T\Big].
\end{aligned}$$

where

$$\begin{aligned}
\hat{\tilde{\boldsymbol{\beta}}} &= (H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}H_{-i}^T A_{-i}^{-1}\tilde{\mathbf{y}}_{-i}\\
&= (H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}H_{-i}^T A_{-i}^{-1}(\frac{\tau}{\sigma}\mathbf{y}_{-i} - \frac{\tau}{\sigma}H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\alpha})\\
&= \frac{\tau}{\sigma}\hat{\boldsymbol{\beta}} - \frac{\tau}{\sigma}\boldsymbol{\beta} + \boldsymbol{\beta} + \boldsymbol{\alpha}.
\end{aligned}$$

and

$$\begin{aligned}
\hat{\tilde{\sigma}}^2 &= \frac{\tilde{\mathbf{y}}_{-i}^T(A_{-i}^{-1} - A_{-i}^{-1}H_{-i}(H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}H_{-i}^T A_{-i}^{-1})\tilde{\mathbf{y}}_{-i}}{n-q-2}\\
(n-q-2)\hat{\tilde{\sigma}}^2 &= \tilde{\mathbf{y}}_{-i}^T(A_{-i}^{-1} - A_{-i}^{-1}H(H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}H_{-i}^T A_{-i}^{-1})\tilde{\mathbf{y}}_{-i}\\
&= (\frac{\tau}{\sigma}\mathbf{y}_{-i} - \frac{\tau}{\sigma}H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\alpha})^T(A_{-i}^{-1} - A_{-i}^{-1}H_{-i}(H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}H_{-i}^T A_{-i}^{-1})\\
&\times (\frac{\tau}{\sigma}\mathbf{y}_{-i} - \frac{\tau}{\sigma}H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\alpha}).
\end{aligned}$$

After some algebra, we can obtain

$$\begin{aligned}\hat{\tilde{\sigma}}^2 &= (\frac{\tau}{\sigma})^2 \frac{\mathbf{y}_{-i}^T(A_{-i}^{-1} - A_{-i}^{-1}H_{-i}(H_{-i}^T A_{-i}^{-1} H_{-i})^{-1} H_{-i}^T A_{-i}^{-1})\mathbf{y}_{-i}}{n-q-2} \\ &= (\frac{\tau}{\sigma})^2 \hat{\sigma}^2.\end{aligned}$$

Thus, the posterior mean, $\mathrm{E}[\tilde{y}_i|\tilde{\mathbf{y}}_{-i}, \boldsymbol{\delta}]$, and the posterior variance, $V[\tilde{y}_i|\tilde{\mathbf{y}}_{-i}, \boldsymbol{\delta}]$, can be written as follows

$$\begin{aligned}E[\tilde{y}_i|\tilde{\mathbf{y}}_{-i}, \boldsymbol{\delta}] &= \mathbf{h}(\mathbf{x}_i)^T \hat{\tilde{\boldsymbol{\beta}}} + \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}(\tilde{\mathbf{y}}_{-i} - H_{-i}\hat{\tilde{\boldsymbol{\beta}}}), \\ &= \mathbf{h}(\mathbf{x}_i)^T (\frac{\tau}{\sigma}\hat{\boldsymbol{\beta}} - \frac{\tau}{\sigma}\boldsymbol{\beta} + \boldsymbol{\beta} + \boldsymbol{\alpha}) \\ &+ \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}\Big(\frac{\tau}{\sigma}\mathbf{y}_{-i} - \frac{\tau}{\sigma}H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\beta} + H_{-i}\boldsymbol{\alpha} \\ &- \frac{\tau}{\sigma}H_{-i}\hat{\boldsymbol{\beta}} + \frac{\tau}{\sigma}H_{-i}\boldsymbol{\beta} - H_{-i}\boldsymbol{\beta} - H_{-i}\boldsymbol{\alpha}\Big) \\ &= \mathbf{h}(\mathbf{x}_i)^T (\frac{\tau}{\sigma}\hat{\boldsymbol{\beta}} - \frac{\tau}{\sigma}\boldsymbol{\beta} + \boldsymbol{\beta} + \boldsymbol{\alpha}) + \frac{\tau}{\sigma}\mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}(\mathbf{y}_{-i} - H_{-i}\hat{\boldsymbol{\beta}}) \\ &= \frac{\tau}{\sigma}\mathbf{h}(\mathbf{x}_i)^T\hat{\boldsymbol{\beta}} + \mathbf{h}(\mathbf{x}_i)^T((1-\frac{\tau}{\sigma})\boldsymbol{\beta} + \boldsymbol{\alpha}) + \frac{\tau}{\sigma}\mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}(\mathbf{y}_{-i} - H_{-i}\hat{\boldsymbol{\beta}}) \\ &= \frac{\tau}{\sigma}E[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] + \mathbf{h}(\mathbf{x}_i)^T((1-\frac{\tau}{\sigma})\boldsymbol{\beta} + \boldsymbol{\alpha}). \\ &= \frac{\tau}{\sigma}(E[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] - \mathbf{h}(\mathbf{x}_i)^T\boldsymbol{\beta}) + \mathbf{h}(\mathbf{x}_i)^T(\boldsymbol{\beta} + \boldsymbol{\alpha}). \\ &= aE[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}] + \mathbf{b}.\end{aligned}$$

and

$$\begin{aligned}V[\tilde{y}_i|\tilde{\mathbf{y}}_{-i}, \boldsymbol{\delta}] &= \hat{\tilde{\sigma}}^2\Big[C(\mathbf{x}_i, \mathbf{x}_i'; \boldsymbol{\delta}) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}\mathbf{t}(\mathbf{x}_i') \\ &+ (\mathbf{h}(\mathbf{x}_i) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}H_{-i})(H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}(\mathbf{h}(\mathbf{x}_i') - \mathbf{t}(\mathbf{x}_i')^T A_{-i}^{-1}H_{-i})^T\Big]. \\ &= (\frac{\tau}{\sigma})^2\hat{\sigma}^2\Big[C(\mathbf{x}_i, \mathbf{x}_i'; \boldsymbol{\delta}) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}\mathbf{t}(\mathbf{x}_i') \\ &+ (\mathbf{h}(\mathbf{x}_i) - \mathbf{t}(\mathbf{x}_i)^T A_{-i}^{-1}H_{-i})(H_{-i}^T A_{-i}^{-1}H_{-i})^{-1}(\mathbf{h}(\mathbf{x}_i') - \mathbf{t}(\mathbf{x}_{-i}')^T A_{-i}^{-1}H_{-i})^T\Big]. \\ &= a^2V[y_i|\mathbf{y}_{-i}, \boldsymbol{\delta}].\end{aligned}$$

Hence, $\tilde{y}_i$ lies inside its credible interval if and only if y_i lies inside its credible interval

$$L_{\alpha,-i}(\mathbf{x}_i) < y_i < U_{\alpha,-i}(\mathbf{x}_i) \iff \tilde{L}_{\alpha,-i}(\mathbf{x}_i) < \tilde{y}_i < \tilde{U}_{\alpha,-i}(\mathbf{x}_i).$$

This means that, for the cross-validation method, the distribution of the coverage interval diagnostic, $K_{\alpha}^{CICV}(\cdot)$ does not change if different values of $\boldsymbol{\beta}$ and σ^2 are used.

Chapter 5

Diagnostics for advanced Gaussian process emulators

5.1 Introduction

In this chapter, we consider diagnostics for more complex emulators than we have considered so far. In particular, we consider emulators that can handle nonstationary covariance. The stationary assumption, equation (2.3.7), of the covariance function for building Gaussian process emulators has been assumed for a theoretical convenience rather than for representing reality. In practice, however, this assumption can be easily challenged. Poor predictions or poor quantification of uncertainty may be obtained if the stationary assumption deviates from the truth. On the other hand, fitting Gaussian process emulators with nonstationary correlation functions can be difficult for high-dimensional functions and may not be computationally tractable.

We consider two extensions of Gaussian process emulators for estimating nonstationary functions. The first one is given by Gramacy and Lee (2012) who present treed Gaussian process models that are based on the idea of partitioning the input space into regions. Then, a stationary Gaussian process emulator is fitted within each region independently. The second extension is given by Ba and Joseph (2012) who develop composite Gaussian process models. The composite Gaussian process models can be constructed as sum of two Gaussian processes with a variance being a function of inputs. We selected these two models as they avoid using complex nonstationary covariance functions and avoid including any extra 'input' variables. Ba and Joseph (2012) validated their composite Gaussian process models using the root mean squared error. Hence, we investigate the development of diagnostic methods that consider uncertainty in the emulator predictions for these two complex emulators.

There are other works for nonstationary modeling but not considered in this chapter. Montagna and Tokdar (2016) propose a nonstationary Gaussian process emulator that is based on two stationary Gaussian processes, one nested into the other. The nonstationarity is achieved by adding an extra latent input, inferred from the input variables, into the input space. This latent input can indicates regions of the input space that have abrupt changes of the simulator outputs and improves inadequacies in the fit. Xiong et al. (2007) propose a nonlinear mapping approach based on density functions to represent the structure of a nonstationary covariance function.

In Section 5.2, we review treed Gaussian process (TGP) emulators for estimating discontinuous and nonstationary functions. In Section 5.3 we review composite Gaussian process (CGP) emulators for dealing with nonstationary functions. In Section 5.4, we present the procedure of calculating the coverage interval diagnostic for TGP and CGP emulators. In Section 5.5 we examine the performance of diagnostic methods for TGP and CGP emulators with an illustrative example. The conclusion is provided in Section 5.6.

5.2 Treed Gaussian process emulators

In this section, we review treed Gaussian process (TGP) models, introduced by Gramacy and Lee (2012), which can be used for emulating discontinuous and nonstationary models. TGP emulators are based on the idea of partitioning the space of the inputs into R regions, $\{r_\nu\}_{\nu=1}^{R}$. Then, a stationary Gaussian process emulator is fitted for data within each region independently. The aim of this approach is to split the input space into small partitions that lead to a simple overall model. This partitioning can help with dealing with nonstationary models as well as reducing the computational demands by fitting models to fewer points.

The TGP emulator is flexible in that it can be smooth in a part of the input space and nonsmooth in another part. The TGP emulator makes binary splits in the input space on the value of a single input, for example, $x_1 > 0.6$. These partitions are recursive in that each new region is a subregion of a previous region. For example, the input space may be divided into two halves by the midpoint. Then, the space above (or below) the midpoint is divided by the second partition and so forth. An independent stationary Gaussian process emulator is then applied in each of the R regions.

5.2.1 Constructing TGP emulators

In order to construct the TGP emulator, a tree $\mathcal{T}$ divides the space of an input into R partitions, $\{r_\nu\}_{\nu=1}^{R}$. Each of these R partitions includes training inputs,

$\mathbf{X}_\nu$, and evaluations $\mathbf{y}_\nu$ of simulator outputs at these inputs. Following the same procedure for representing uncertainty about the simulator by equation (2.3.14), uncertainty about $\mathbf{y}_\nu$ is described by a stationary Gaussian process with a linear mean function for each region, r_ν, separately. However, the distributions of $\boldsymbol{\beta}_\nu$ and σ^2_ν are different from those in Section 2.3. The distributions of $\boldsymbol{\beta}_\nu$ and σ^2_ν depend on other hyperparameters as follows

$$\begin{aligned}
\mathbf{y}_\nu|\boldsymbol{\beta}_\nu, \sigma^2_\nu, \boldsymbol{\delta}_\nu &\sim N_{n_\nu}(H_\nu\boldsymbol{\beta}_\nu, \sigma^2_\nu A_\nu),\\
\boldsymbol{\beta}_0 &\sim N(\boldsymbol{\mu}, \mathbf{B}),\\
\boldsymbol{\beta}_\nu|\sigma^2_\nu, \tau^2_\nu, \mathbf{W}, \boldsymbol{\beta}_0 &\sim N(\boldsymbol{\beta}_0, \sigma^2_\nu\tau^2_\nu\mathbf{W}),\\
\tau^2_\nu &\sim \text{Inv-gamma}\Big(\frac{\alpha_\tau}{2}, \frac{q_\tau}{2}\Big),\\
\sigma^2_\nu &\sim \text{Inv-gamma}\Big(\frac{\alpha_\sigma}{2}, \frac{q_\sigma}{2}\Big),\\
\mathbf{W}^{-1} &\sim \text{Wishart}((\rho\mathbf{V})^{-1}, \rho),
\end{aligned} \tag{5.2.1}$$

where $H_\nu = (\mathbf{1}, \mathbf{X}_\nu)$ and $A_\nu = \mathbf{C}_\nu(\mathbf{X}, \mathbf{X}', \boldsymbol{\delta}_\nu)$ is the correlation between the simulator training outputs in the r_ν region. The unknown coefficients parameters, $\boldsymbol{\beta}_\nu$, are believed to have a normal distribution with a common mean $\boldsymbol{\beta}_0$, common correlation function, $\mathbf{W}$ and a specific variance $\sigma^2_\nu\tau^2_\nu$. Each of these hyperparameters follows a distribution, as shown in (5.2.1), where σ^2_ν is the overall variance in the r_ν region, $\sigma^2_\nu\tau^2_\nu$ is the variance for $\boldsymbol{\beta}_\nu$ and $\mathbf{W}$ is a $p \times p$ matrix. The hyperparameters $\boldsymbol{\mu}, \mathbf{B}, \mathbf{V}, \rho, \alpha_\tau, q_\tau, \alpha_\sigma$ and q_σ are assumed to be known. We can see that the hyperparameters σ^2_ν and τ^2_ν do not depend on the correlation length parameters.

The correlation function structure is slightly different from that in Section 2.3. The correlation function for the TGP emulator is considered with a nugget parameter, $\mathbf{C}_\nu(\mathbf{x}, \mathbf{x}') = C_\nu(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}) + g_\nu d_{j,k}$, where $d_{\cdot,\cdot}$ represents the Kronecker delta function. When the computer model has a numerical error, the nugget parameter, $g_\nu > 0$, can be included to introduce a measurement error in the stochastic process. Including a nugget parameter to the correlation function makes the correlation function numerically more stable and prevents the correlation matrix from becoming singular. Moreover, without including a nugget parameter, Gramacy and Lee (2012) argue that the Gaussian process emulator may not provide accurate results if the model assumptions are not satisfied. However, including a positive nugget leads to extra variance around the design points.

Gramacy and Lee (2012) consider the separable power exponential correlation function given by equation (2.3.5) or the isotropic power exponential correlation function

$$C_\nu(\mathbf{x}, \mathbf{x}'; \delta) = \exp\left\{-\frac{\|\mathbf{x}_i - \mathbf{x}_i'\|^\gamma}{\delta}\right\}, \tag{5.2.2}$$

where $\delta > 0$ and $\gamma \in (0, 2]$.

A prior for g_ν is proposed to be $\text{Exp}(\lambda)$, whereas a prior for δ is chosen to be

$$\begin{aligned} p(\delta, g) &= p(\delta) \times p(g) && (5.2.3) \\ &= p(g) \times \frac{1}{2}[\text{Gamma}(\delta | a = 1, b = 20) && (5.2.4) \\ &+ \text{Gamma}(\delta | a = 10, b = 10)]. \end{aligned}$$

This prior function provides equal mass for δ that represents a population for Gaussian process parameterizations for nonsmooth surfaces and separate surfaces for smooth or approximately linear.

A prior distribution of a tree is specified through a tree-generating process. The process starts with a null tree and assumes all data in one region. Then, a leaf node, $\eta \in T$, that represents an input space region, splits with probability $a(1 + q_\eta)^b$, where q_η is the depth of the leaf node. The parameters a and b are chosen to give a suitable size of trees. Gramacy and Lee (2012) used $a = 0.5$ and $b = 2$ and they found that they work well in practice. The prior distribution for the splitting process includes selecting the splitting dimension, u, from a discrete uniform distribution. The split location, s, is then selected uniformly from a subset of the locations in the $u-th$ dimension.

5.2.2 Estimating the parameters

The values of the parameters $\boldsymbol{\theta}_\nu = \{\boldsymbol{\beta}, \sigma^2, \tau^2, \boldsymbol{\delta}\}_\nu$ are obtained using the data $\{\mathbf{X}_\nu, \mathbf{y}_\nu\}$, for $\nu = 1, \ldots, R$. The full set of parameters, $\boldsymbol{\theta} = \boldsymbol{\theta}_0 \bigcup_\nu^R \boldsymbol{\theta}_\nu$, will be conditional on the tree $\mathcal{T}$, where $\boldsymbol{\theta}_0 = \{\mathbf{W}, \boldsymbol{\beta}_0, \boldsymbol{\xi}\}$, and $\boldsymbol{\xi}$ is a known parameter that can be given a prior distribution similar to how a prior is specified for $\boldsymbol{\beta}_0$. Gramacy and Lee (2012) used an MCMC algorithm to obtain samples from the posterior distribution of $\boldsymbol{\theta}$ by first drawing from $\boldsymbol{\theta}_\nu|\boldsymbol{\theta}_0$ for $\nu_1, \ldots, \nu_R$. Then, $\boldsymbol{\theta}_0$ is drawn as $\boldsymbol{\theta}_0| \bigcup_\nu^R \boldsymbol{\theta}_\nu$.

5.2.3 Predictions of TGP emulators

The emulator predictions are conditional on the structure of the tree and they are averaged over the posterior distribution of $(\mathcal{T}, \boldsymbol{\theta})$ to obtain full accounting of uncertainty. The posterior distribution of $f(\cdot)$ in the r_ν region conditional on $\boldsymbol{\theta}$ and $\mathcal{T}$ follows a normal distribution with a posterior mean and a posterior variance given by

$$\begin{aligned} m_\nu(\mathbf{x}) &= \mathbf{h}(\mathbf{x})^T \tilde{\boldsymbol{\beta}}_\nu + \mathbf{t}_\nu(\mathbf{x})^T A_\nu^{-1}(\mathbf{y}_\nu - H_\nu \tilde{\boldsymbol{\beta}}_\nu), && (5.2.5) \\ V_\nu(\mathbf{x}, \mathbf{x}') &= \sigma_\nu^2 \left[\mathbf{K}(\mathbf{x}, \mathbf{x}') - \mathbf{q}_\nu(\mathbf{x})^T \mathbf{Q}_\nu^{-1} \mathbf{q}_\nu(\mathbf{x})\right], && (5.2.6) \end{aligned}$$

where

$$\begin{aligned}
\mathbf{h}(\mathbf{x}) &= (\mathbf{1}, \mathbf{x}), \qquad (5.2.7)\\
\mathbf{t}_\nu(\mathbf{x}) &= \Big(\mathbf{C}_\nu(\mathbf{x}, \mathbf{x}_1), \ldots, \mathbf{C}_\nu(\mathbf{x}, \mathbf{x}_{n_\nu})\Big)^T,\\
\tilde{\boldsymbol{\beta}}_\nu &= \mathbf{V}_{\tilde{\boldsymbol{\beta}}_\nu}\Big(H_\nu^T A_\nu^{-1}\mathbf{y}_\nu + \mathbf{W}^{-1}\boldsymbol{\beta}_{\mathbf{0}}/\tau_\nu^2\Big),\\
\mathbf{V}_{\tilde{\boldsymbol{\beta}}_\nu} &= \Big(H_\nu^T A_\nu^{-1} H_\nu + \mathbf{W}^{-1}/\tau_\nu^2\Big)^{-1},\\
\mathbf{Q}_\nu^{-1} &= \Big(A_\nu + \tau_\nu^2 H_\nu \mathbf{W} H_\nu^T\Big)^{-1},\\
\mathbf{q}_\nu(\mathbf{x}) &= \mathbf{t}_\nu(\mathbf{x}) + \tau_\nu^2 H_\nu \mathbf{W}\mathbf{h}(\mathbf{x}),\\
\mathbf{K}(\mathbf{x}, \mathbf{x}') &= \mathbf{C}_\nu(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}_\nu) + \tau_\nu^2 \mathbf{h}(\mathbf{x})^T \mathbf{W}\mathbf{h}(\mathbf{x}),
\end{aligned}$$

where $\mathbf{h}(\mathbf{x})$ is the regression functions in the ν partition just like the regression functions in equation (2.3.1). The $\mathbf{t}_\nu(\mathbf{x})$ in the ν partition is the same as $\mathbf{t}(\mathbf{x})$ in Section 2.3 but with a nugget parameter. The parameter $\tilde{\boldsymbol{\beta}}_\nu$ depends also on $\boldsymbol{\delta}$ and $\mathbf{y}_\nu$ but now it also depends on some other parameters, $\boldsymbol{\beta}_0$ and τ_ν^2, and it has different structure from $\hat{\boldsymbol{\beta}}$ in (2.3.22). Moreover, unlike $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}), \mathbf{t}(\mathbf{x})$ and A^{-1} in posterior variance, equation (2.3.29), the terms $\mathbf{K}(\mathbf{x}, \mathbf{x}'), \mathbf{q}_\nu(\mathbf{x})$ and $\mathbf{Q}_\nu^{-1}$ have an extra term as shown in (5.2.7).

Conditional on a tree, $\mathcal{T}$, the posterior distribution of $f(\cdot)$ with the posterior mean and the posterior variance, equations (5.2.5) and (5.2.6), is discontinuous across the boundaries of the tree partitions. Also, in the posterior for the tree, $\mathcal{T}$, uncertainty is higher near boundaries than in other parts. However, samples from the joint posterior distribution of $(\mathcal{T}, \boldsymbol{\theta})$, can be obtained. The average of these samples may smooth out near the boundaries of the partition. The TGP model has the ability to retain the flexibility that is necessary to model discontinuities when the data indicate a nonsmooth process.

Integrating out the dependence on the tree can be achieved using MCMC. The TGP emulators are implemented using an R package, called `tgp`. A description of the `tgp` package with examples is provided by Gramacy (2007). Estimates of the parameters of the TGP emulators and integrating out the dependence on the tree require using MCMC algorithms. This makes the TGP emulators computational more expensive to run than the stationary Gaussian process emulator that is described in Section 2.3.

5.2.4 One-dimensional synthetic example

We illustrate the TGP emulator with a one-dimensional example. Consider a simulator given by

$$f(x) = \begin{cases} \sin(\frac{x}{10}) + 0.2\cos(x), & \text{if } x > -2 \\ 0.1 + \frac{\cos(x)}{20}, & \text{otherwise .} \end{cases} \tag{5.2.8}$$

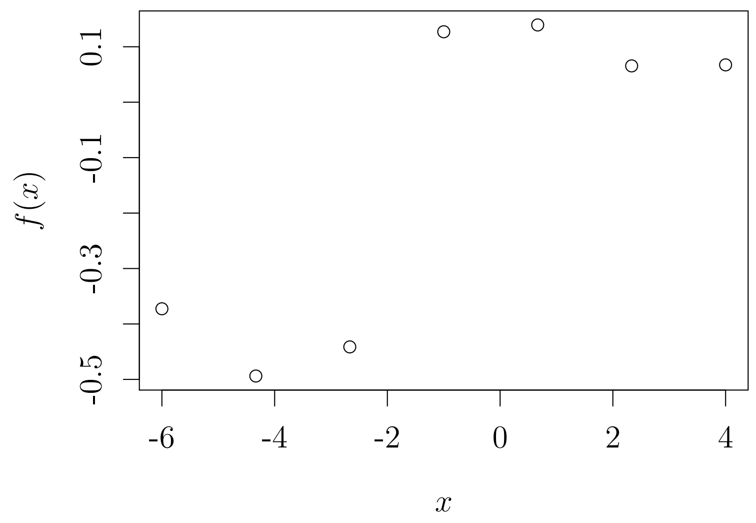

Figure 5.1: The simulator outputs of model (5.2.8) at some training inputs. The function is discontinuous at $x = -2$.

The inputs were chosen to be in the space [-6, 2], where we generated a sequence of 7 points to be the training inputs. Figure 5.1 shows the inputs against the model outputs. The function is discontinuous at $x = -2$ and shows different behaviours in different parts of the input space.

We fitted a stationary Gaussian process emulator and a TGP emulator for the simulator where we used the `tgp` package to fit the TGP emulator. Figure 5.2 shows the stationary Gaussian process emulator and the TGP emulator with the 95% credible interval.

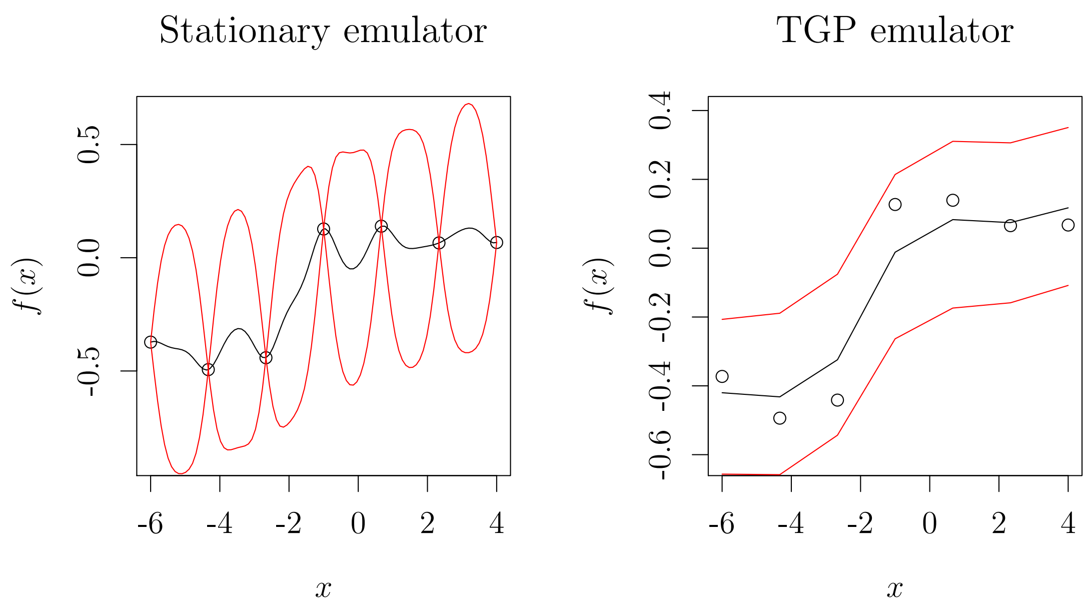

Figure 5.2: Emulator posterior mean with 95% credible intervals against true simulator outputs for the stationary Gaussian process emulator and the TGP emulator based on 7 points. The uncertainty is large for both emulators.

The left panel is from the stationary Gaussian process emulator. It is shown that the uncertainty is large and the emulator is underconfident. The right panel is from the TGP emulator. It is shown that the 95% credible interval is also large and the TGP emulator does not show any partition in the input space. The posterior mean of the TGP emulator does not pass through the training data and the uncertainty is not zero at the training data due to including a nugget term in the correlation function.

Hence, we increased the number of the points in the sequence and we found that the number of the points in the sequence should be at least 26 in order to obtain a partition in the input space. Figure 5.3 shows the stationary Gaussian process emulator and the TGP emulator with the 95% credible interval.

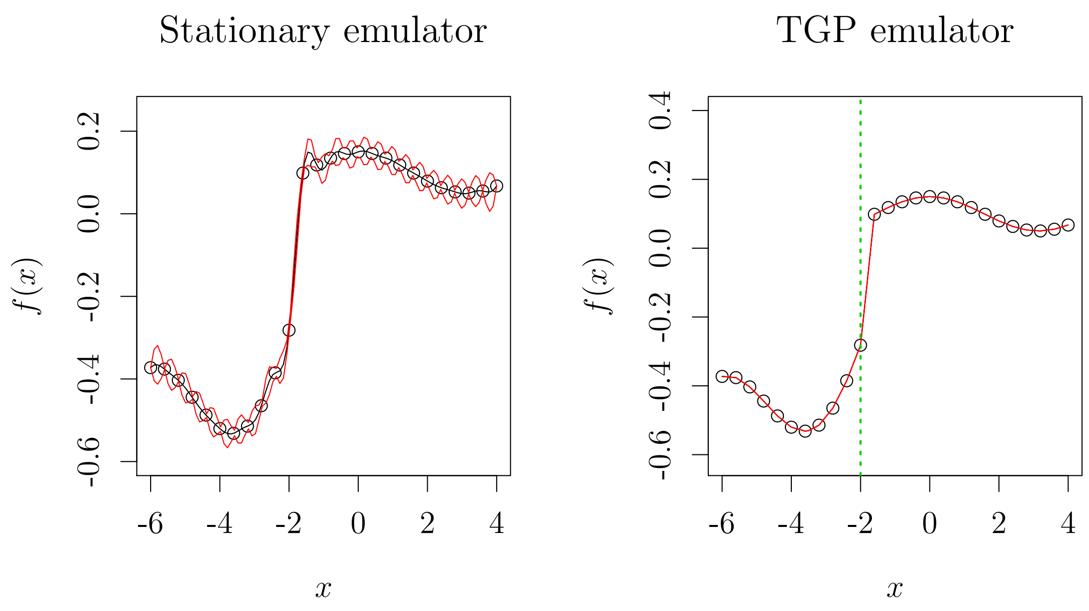

Figure 5.3: Emulator posterior mean with 95% credible intervals against true simulator outputs for the stationary Gaussian process emulator and the TGP emulator based on 26 points. The uncertainty in the TGP emulator is smaller than that in the stationary Gaussian process emulator.

The left panel shows that the stationary Gaussian process emulator is a little underconfident. The right panel shows that the 95% credible interval of the TGP emulator is smaller than that of the stationary Gaussian process emulator. The TGP emulator is able to adapt to the behaviour of the data. This is because the TGP emulator fits a Gaussian process emulator to the points that are less than $x = -2$ and fits another Gaussian process emulator to the points that are greater than $x = -2$.

The vertical line on the right panel shows a typical treed partition $\hat{\mathcal{T}}$. Figure 5.4 shows the maximum a posteriori (MAP) tree, $\hat{\mathcal{T}}$, for the TGP emulator. The number of the training points in the first partition is 11 with $\hat{\sigma}_1^2 = 0.0664$ whereas the number of the training points in the second partition is 15 with $\hat{\sigma}_2^2 = 0.0112$.

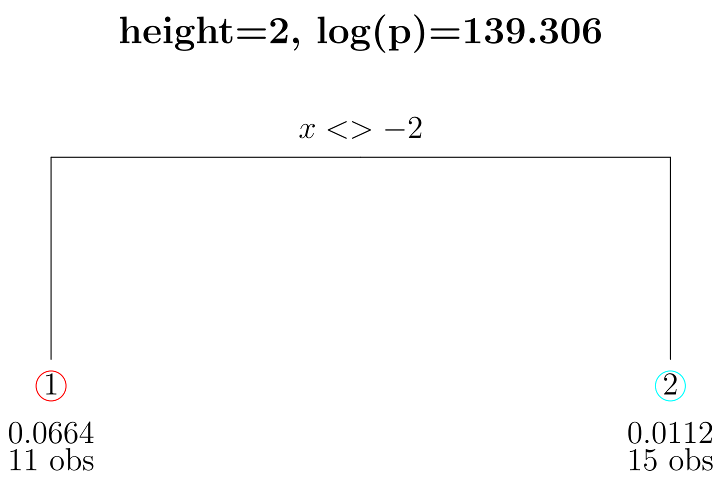

Figure 5.4: The MAP tree of height 2 with the number of training points, n_ν, and σ_ν^2 in each partition of the tree. The partition is at $x = -2$ and the variance in partition 1 is larger than that in partition 2.

5.3 Composite Gaussian process emulators

In the nonstationary Gaussian process model literature, most of the studies, such as Paciorek and Schervish (2003) and Anderes and Stein (2008), concentrate on deriving complex nonstationary correlation functions, which may become computationally intractable to fit in high dimensional inputs. Ba and Joseph (2012) developed a nonstationary model, called a composite Gaussian process (CGP) model, that has the ability to approximate expensive nonstationary simulators. The CGP model can be developed by two steps. First, the CGP is constructed as sum of two Gaussian processes, with one thought of as replacing the mean $h(\cdot)^T\boldsymbol{\beta}$. Second, the variance is defined to be a function of inputs, $\sigma^2(\mathbf{x})$, in order to remedy the change of variability in the output. In this section, we review the process of constructing CGP models in some detail.

Suppose we have n different training inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and evaluations $\mathbf{y} = \{y_1 = f(\mathbf{x}_1), \ldots, y_n = f(\mathbf{x}_n)\}$ of the simulator outputs at these inputs. A stationary Gaussian process as given by (2.3.11) can be represented by an equivalent way as follows

$$\mathbf{y} = m(\mathbf{x}) + Z(\mathbf{x}), \tag{5.3.1}$$

where $m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ and $Z(\mathbf{x}) \sim GP(0, \sigma^2 C(\cdot,\cdot))$. The posterior mean is given by equation (2.3.28). If a constant mean, μ, is used, the stationary Gaussian process model will be defined as

$$\mathbf{y} = \mu + Z(\mathbf{x}). \tag{5.3.2}$$

The posterior mean in this case is

$$m_1(\mathbf{x}) = \hat{\mu} + \mathbf{t}^T(\mathbf{x})A^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}), \tag{5.3.3}$$

where $\mathbf{1}$ is an n-dimensional vector with all elements 1, and $\hat{\mu} = (\mathbf{1}^T A^{-1}\mathbf{1})^{-1}(\mathbf{1}^T A^{-1}\mathbf{y})$.

The posterior mean in equation (2.3.28) with a linear mean function will give better predictions than that in (5.3.3) if $m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$ is close to the true global trend. However, the correct function $\mathbf{h}(\mathbf{x})$ is rarely known in practice and the predictions will be worse if the trend is specified wrongly. This is because with expensive simulators, we only have a limited number of simulator outputs and so the design points may not cover the input space well. Hence, using inappropriate mean function, the emulator will capture the variation for points that are close to training points and will revert to the inappropriate mean function for points that are far from training points. Moreover, the points that are far from the training points will have large variances and the points that are close to training points will have small variances. Hence, assuming a constant overall variance, σ^2, will not be appropriate.

5.3.1 Improving the mean model

The Gaussian process consists of a linear mean function, $m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta}$, as the global trend and a Gaussian process model, $Z(\mathbf{x})$, for local adjustments. Instead of choosing a parametric form for $h(\cdot)$, Ba and Joseph (2012) propose adding another Gaussian process as follows

$$\begin{aligned} \mathbf{y} &= Z_{global}(\mathbf{x}) + Z_{local}(\mathbf{x}), \\ Z_{global}(\cdot) &\sim GP(\mu, \tau^2 g(\cdot,\cdot)), \\ Z_{local}(\cdot) &\sim GP(0, \sigma^2 l(\cdot,\cdot)), \end{aligned} \tag{5.3.4}$$

where $Z_{global}(\mathbf{x})$ and $Z_{local}(\mathbf{x})$ are stationary and independent of each other such that $\tau^2 > \sigma^2$ and $g(\mathbf{x}, \mathbf{x}') > l(\mathbf{x}, \mathbf{x}')$ to ensure that the $Z_{global}(\cdot)$ process is smoother. Hence, the $Z_{global}(\cdot)$ process, with constant mean μ, variance τ^2 and correlation function $g(\cdot,\cdot)$, is smooth and is intended to describe the global trend.

The $Z_{local}(\cdot)$ process, with mean 0, variance σ^2 and correlation function $l(\cdot,\cdot)$, is for local adjustments. This model can be seen as an extension of the Gaussian process in equation (5.3.2) that can adapt to more complex models.

Model (5.3.4) can be written as $\mathbf{y} \sim GP(\mu, \tau^2 g(\cdot,\cdot) + \sigma^2 l(\cdot,\cdot))$, and it is still a Gaussian process as it is the sum of two independent Gaussian processes. Using the same procedure given in Section 2.3, the posterior mean can be written as

$$m_1(\mathbf{x}) = \hat{\mu} + (\mathbf{g}(\mathbf{x}) + \lambda \mathbf{l}(\mathbf{x}))^T (\mathbf{G} + \lambda \mathbf{L})^{-1} (\mathbf{y} - \hat{\mu}\mathbf{1}), \tag{5.3.5}$$

where

$$\begin{aligned}
\hat{\mu} &= (\mathbf{1}^T(\mathbf{G}+\lambda\mathbf{L})^{-1}\mathbf{1})^{-1}\mathbf{1}^T(\mathbf{G}+\lambda\mathbf{L})^{-1}\mathbf{y}, \\
\mathbf{g}(\mathbf{x}) &= (g(\mathbf{x},\mathbf{x}_1;\boldsymbol{\theta}),\ldots,g(\mathbf{x},\mathbf{x}_n;\boldsymbol{\theta}))^T, \\
\mathbf{l}(\mathbf{x}) &= (l(\mathbf{x},\mathbf{x}_1;\boldsymbol{\alpha}),\ldots,l(\mathbf{x},\mathbf{x}_n;\boldsymbol{\alpha}))^T, \\
\mathbf{G} &= \begin{pmatrix} g(\mathbf{x}_1,\mathbf{x}_1;\boldsymbol{\theta}) & \cdots & g(\mathbf{x}_1,\mathbf{x}_n;\boldsymbol{\theta}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{x}_n,\mathbf{x}_1;\boldsymbol{\theta}) & \cdots & g(\mathbf{x}_n,\mathbf{x}_n;\boldsymbol{\theta}) \end{pmatrix}, \\
\mathbf{L} &= \begin{pmatrix} l(\mathbf{x}_1,\mathbf{x}_1;\boldsymbol{\alpha}) & \cdots & l(\mathbf{x}_1,\mathbf{x}_n;\boldsymbol{\alpha}) \\ \vdots & \ddots & \vdots \\ l(\mathbf{x}_n,\mathbf{x}_1;\boldsymbol{\alpha}) & \cdots & l(\mathbf{x}_n,\mathbf{x}_n;\boldsymbol{\alpha}) \end{pmatrix}
\end{aligned}$$

and $\lambda = \sigma^2/\tau^2$ represents the ratio of variances. Although the structure of correlation functions is different, $\mathbf{G}$ and $\mathbf{L}$ are similar to A in equation (2.3.14) as well as $g(\mathbf{x}_1,\mathbf{x}_1;\boldsymbol{\theta})$ and $l(\mathbf{x}_1,\mathbf{x}_1;\boldsymbol{\alpha})$ are similar as $\mathbf{t}(\mathbf{x})$. Here, the correlation functions is chosen to be squared exponential correlation functions with different correlation length parameters

$$g(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}) = \exp\left\{-\sum_{i=1}^{p}\theta_i\,(x_i - x_i')^2\right\}, \tag{5.3.6}$$

$$l(\mathbf{x},\mathbf{x}';\boldsymbol{\alpha}) = \exp\left\{-\sum_{i=1}^{p}\alpha_i\,(x_i - x_i')^2\right\}, \tag{5.3.7}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_p)$ are unknown correlation length parameters such that, $0 \leq \boldsymbol{\theta} \leq \boldsymbol{\alpha}^l$ and $\boldsymbol{\alpha}^l \leq \boldsymbol{\alpha}$. Usually, the bounds $\boldsymbol{\alpha}^l$ are chosen to be moderately large to ensure that $Z_{global}(\mathbf{x})$ is smoother than $Z_{local}(\mathbf{x})$.

The global trend in model (5.3.4) is expected to capture most of the variation in the output than the local process, and so λ is restricted to be in the $[0,1]$ interval. Model (5.3.5) will be reduced to model (5.3.2) if $\lambda = 0$.

5.3.2 Improving both the mean and variance models

Model (5.3.4) can be further relaxed by incorporating a variance model $\sigma^2(\mathbf{x})$ rather than σ^2 as follows:

$$\begin{aligned} \mathbf{y} &= Z_{global}(\mathbf{x}) + \sigma(\mathbf{x}) Z_{local}(\mathbf{x}), \\ Z_{global}(\mathbf{x}) &\sim GP(\mu, \tau^2 g(\cdot,\cdot)), \\ Z_{local}(\mathbf{x}) &\sim GP(0, l(\cdot,\cdot)). \end{aligned} \tag{5.3.8}$$

The $Z_{local}(\mathbf{x})$, with a variance model $\sigma^2(\mathbf{x})$, quantifies the change of local variability such that $\sigma(\mathbf{x}) Z_{local}(\mathbf{x}) \sim GP(0, \sigma^2(\mathbf{x}) l(\cdot,\cdot))$. Model (5.3.8) can be written as $\mathbf{y} \sim GP(\mu, \tau^2 g(\cdot,\cdot) + \sigma^2(\mathbf{x}) l(\cdot,\cdot))$ which can be seen as an extension of the Gaussian process in equation (5.3.2) with a nonstationary covariance function $\tau^2 g(\cdot,\cdot) + \sigma^2(\mathbf{x}) l(\cdot,\cdot)$. This model is a Gaussian process with a constant mean function and the covariance function is a sum of two covariance functions. The first covariance function is with constant variance, τ^2 and correlation function, $g(\cdot,\cdot)$. The second covariance function is with nonstationary variance, $\sigma^2(\mathbf{x})$ and correlation function, $g(\cdot,\cdot)$.

Ba and Joseph (2012) propose to separate the variance $\sigma^2(\mathbf{x})$ as $\sigma^2(\mathbf{x}) = \sigma^2 \upsilon(\mathbf{x})$, where σ^2 is an unknown variance and $\upsilon(\mathbf{x})$ is a positive function that fluctuates around the unit value. In order to obtain values of $\upsilon(\mathbf{x})$, let $\Sigma = \text{diag}\{\upsilon(\mathbf{x}_1), \ldots, \upsilon(\mathbf{x}_n)\}$ be the standardised local variances at each training point $\mathbf{x}_1, \ldots, \mathbf{x}_n$. The Σ can be set as $\Sigma = I$ initially and then values of $\upsilon(\mathbf{x})$ can be obtain as follows:

- Calculate the squared residuals of the global trend, $\mathbf{s}^2 = (s_1^2, \ldots, s_n^2)^T$ where $s_i = y_i - m_{global}(x_i)$, for $i = (1, \ldots, n)$, and $m_{global}(x_i)$ is given later by equation (5.3.16).
- A function for $\upsilon(\mathbf{x})$ is proposed as

$$\upsilon(\mathbf{x}) = \frac{\mathbf{g}_b^T(\mathbf{x})\mathbf{s}^2}{\mathbf{g}_b^T(\mathbf{x})\mathbf{1}}, \tag{5.3.9}$$

  where $\mathbf{g}_b(\mathbf{x}) = (g_b(\mathbf{x}, \mathbf{x}_1; \boldsymbol{\theta}), \ldots, g_b(\mathbf{x}, \mathbf{x}_n; \boldsymbol{\theta}))^T$ with

$$g_b(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \exp\left\{-b\sum_{i=1}^{p} \theta_i \left(x_i - x_i'\right)^2\right\}, \tag{5.3.10}$$

  where $b \in [0, 1]$ is an extra bandwidth parameter.
- Evaluate $\hat{\upsilon}_i = \upsilon(\mathbf{x}_i)$ for $i = 1, \ldots, n$ and update Σ to $\Sigma = \text{diag}\{\hat{\upsilon}_1, \ldots, \hat{\upsilon}_n\}$.
- Rescaled the standardised local volatilities $\upsilon(\mathbf{x})$ and Σ

$$\Sigma \longleftarrow \Sigma/\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\upsilon}_i\right) \qquad \text{and} \qquad \upsilon(\mathbf{x}) \longleftarrow \upsilon(\mathbf{x})/\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\upsilon}_i\right). \tag{5.3.11}$$

The process above is repeated within a loop for few times to stabilize the estimates of $\upsilon(\mathbf{x})$ and Σ. The bandwidth parameter b in equation (5.3.10) adds additional flexility in controlling the smoothness of the variance function such that $\mathbf{g}_b(\mathbf{x}) \longrightarrow \mathbf{1}$ as $b \longrightarrow 0$, and $\mathbf{g}_b(\mathbf{x}) = \mathbf{g}(\mathbf{x})$ if $b = 1$. When $b = 0$, the $\upsilon(\mathbf{x})$ will be smoothed out to a constant function even the global trend is not flat. The benefit of the standardisation in (5.3.11) is to make the diagonal elements of Σ have unit mean, that is essential for keeping the ratio λ consistent in the global trend.

Equation (5.3.10) can be seen as a weighted average of $\mathbf{s}^2$ values. This function is smooth since it is linear in the observations s_i^2, and so the variances of any two points in the global trend will be more related if these points are strongly correlated.

The posterior CGP emulator

Suppose now we want to know $f(\cdot)$ at a validation input $\mathbf{x}^*$. According to the assumptions in model (5.3.8), the $f(\mathbf{x}^*)$, and the training outputs, $\mathbf{y} = \{y_1, \ldots, y_n\}$, have a multivariate normal distribution

$$\begin{pmatrix} f(\mathbf{x}^*) \\ \mathbf{y} \end{pmatrix} \sim N_{n+1}\left[\begin{pmatrix} \mu \\ \mu\mathbf{1} \end{pmatrix}, \begin{pmatrix} \tau^2 + \sigma^2 \upsilon(\mathbf{x}) & (\tau^2\mathbf{g}(\mathbf{x}) + \sigma^2\upsilon^{1/2}(\mathbf{x})\Sigma^{1/2}\mathbf{l}(\mathbf{x}))^T \\ \tau^2\mathbf{g}(\mathbf{x}) + \sigma^2\upsilon^{1/2}(\mathbf{x})\Sigma^{1/2}\mathbf{l}(\mathbf{x}) & \tau^2\mathbf{G} + \sigma^2\Sigma^{1/2}\mathbf{L}\Sigma^{1/2} \end{pmatrix}\right]. \tag{5.3.12}$$

In order to obtain the posterior distribution of $f(\cdot)$, we can follow the same procedure that is given in Section 2.3 for deriving the posterior emulator. Assuming a noninformative prior for μ, $p(\mu) \propto 1$, and integrating μ out, the posterior distribution of $f(\cdot)$, conditional on $\tau^2, \sigma^2, \boldsymbol{\theta}$ and $\boldsymbol{\alpha}$, is a normal distribution with a posterior mean

$$\begin{aligned} m_1(\mathbf{x}) &= \hat{\mu} + (\tau^2\mathbf{g}(\mathbf{x}) + \sigma^2\upsilon^{1/2}(\mathbf{x})\Sigma^{1/2}\mathbf{l}(\mathbf{x}))^T(\tau^2\mathbf{G} + \sigma^2\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})^{-1} \\ &\times (\mathbf{y} - \hat{\mu}\mathbf{1}) \qquad (5.3.13) \\ &= \hat{\mu} + \mathbf{q}(\mathbf{x})^T\mathbf{Q}^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}) \end{aligned}$$

and a posterior variance

$$V_1(\mathbf{x}, \mathbf{x}) = \tau^2\left\{1 + \lambda\upsilon(\mathbf{x}) - \mathbf{q}(\mathbf{x})^T\mathbf{Q}^{-1}\mathbf{q}(\mathbf{x}) + \frac{(1 - \mathbf{q}(\mathbf{x})^T\mathbf{Q}^{-1}\mathbf{1})^2}{\mathbf{1}^T\mathbf{Q}^{-1}\mathbf{1}}\right\}, \tag{5.3.14}$$

where

$$\begin{aligned}
\hat{\mu} &= (\mathbf{1}^T\mathbf{Q}^{-1}\mathbf{1})^{-1}\mathbf{1}^T\mathbf{Q}^{-1}\mathbf{y},\\
\mathbf{q}(\mathbf{x}) &= \mathbf{g}(\mathbf{x}) + \lambda v^{1/2}(\mathbf{x})\Sigma^{1/2}\mathbf{l}(\mathbf{x}),\\
\mathbf{Q} &= \mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2}.
\end{aligned}$$

If the function $v(\mathbf{x})$ degenerates to a constant function, the posterior mean (5.3.13) will be reduced to the posterior mean in (5.3.5). Unlike the posterior variance of the stationary Gaussian process emulator, the posterior variance of the CGP model, equation (5.3.14), depends on the local variability of the $Z_{local}(\mathbf{x})$ model. This can improve the prediction intervals for the CGP model.

The posterior mean, equation (5.3.13), and the posterior variance, equation (5.3.14), is for one predicted output. However, they can be generalized to be for a vector of m of predicted outputs and they will be the similar to those in equations (2.3.28) and (2.3.29) for the stationary Gaussian process emulator in Chapter 2, but with $\hat{\mu}, \mathbf{q}(\mathbf{x})$ and $\mathbf{Q}$ instead of $\mathbf{h}(\mathbf{x})^T\hat{\boldsymbol{\beta}}, \mathbf{t}(\mathbf{x})$ and A.

The posterior mean (5.3.13) can be decomposed into two parts:

$$\begin{aligned}
m_1(\mathbf{x}) &= m_{global}(\mathbf{x}) + m_{local}(\mathbf{x}), && (5.3.15)\\
m_{global}(\mathbf{x}) &= \hat{\mu} + \mathbf{g}^T(\mathbf{x})(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}), && (5.3.16)\\
m_{local}(\mathbf{x}) &= \lambda v^{1/2}(\mathbf{x})\mathbf{l}^T(\mathbf{x})\Sigma^{1/2}(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}). && (5.3.17)
\end{aligned}$$

5.3.3 Estimating the parameters

Ba and Joseph (2012) derive maximum likelihood estimates for μ and τ^2:

$$\begin{aligned}
\hat{\mu}(\lambda, \boldsymbol{\theta}, \boldsymbol{\alpha}, b) &= (\mathbf{1}^T(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})^{-1}\mathbf{1})^{-1}(\mathbf{1}^T(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})^{-1}\mathbf{y}),\\
\hat{\tau}^2(\lambda, \boldsymbol{\theta}, \boldsymbol{\alpha}, b) &= \frac{1}{n}(\mathbf{y} - \hat{\mu}\mathbf{1})^T(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})^{-1}(\mathbf{y} - \hat{\mu}\mathbf{1}).
\end{aligned}$$

We can see that $\hat{\mu}$ and $\hat{\tau}^2$ are similar to $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ in equations (2.3.22) and (2.3.23) but here we have $H = \mathbf{1}$ and with correlation $\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2}$.

The maximum likelihood estimates for $(\lambda, \boldsymbol{\theta}, \boldsymbol{\alpha}, b)$ can be obtained by minimizing the log-likelihood:

$$\phi(\lambda, \boldsymbol{\theta}, \boldsymbol{\alpha}, b) = n\log(\hat{\tau}^2(\lambda, \boldsymbol{\theta}, \boldsymbol{\alpha}, b)) + \log(\det(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})). \qquad (5.3.18)$$

This likelihood function has $2p + 2$ unknown parameters, whereas the likelihood of the stationary Gaussian process model in equation (5.3.2) contains only p unknown parameters. Therefore, estimating the CGP model parameters becomes more difficult with large input dimension p. Thus, Ba and Joseph (2012) suppose

$$\alpha_j = \theta_j + \kappa, \qquad j = 1, \ldots, p, \qquad (5.3.19)$$

which makes the CGP model contains only $p+3$ unknown parameters $(\lambda, \boldsymbol{\theta}, \kappa, b)$. Thus, we can obtain the MLEs by minimizing

$$\phi(\lambda, \boldsymbol{\theta}, \kappa, b) = n\log(\hat{\tau}^2(\lambda, \boldsymbol{\theta}, \kappa, b)) + \log(\det(\mathbf{G} + \lambda\Sigma^{1/2}\mathbf{L}\Sigma^{1/2})), \qquad (5.3.20)$$

under the constraints $\lambda \in [0, 1], b \in [0, 1], \theta_j \in [0, \alpha^l]$ and $\kappa \in [\alpha^l, \infty]$ for $j = 1, \ldots, p$.

The CGP model with the $p + 3$ unknown parameters is still computationally more expensive than the stationary Gaussian process emulator given by equation (5.3.2). This is due to adding another Gaussian process model to the Gaussian process and making the variance to be a function of the inputs.

5.3.4 Example

In this example, we illustrate the difference between the process of building stationary Gaussian process emulators and CGP emulators. Consider a one-dimensional simulator given by

$$f(x) = \exp(-\sin(15(x - 0.7)^3)) + x.$$

We chose 8 training inputs in the interval $[0, 1]$, and then we evaluated the simulator output at these training inputs. In order to construct the stationary Gaussian process emulator, we assumed that the prior uncertainty on the simulator is represented by equation (5.3.1). We used $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and $Z(\mathbf{x}) \sim GP(0, \sigma^2 C(\cdot, \cdot))$, with the squared exponential correlation function $C(x, x', \delta)$ given in equation (2.3.4). The correlation length parameters were estimated by the maximum likelihood and then the stationary Gaussian process emulator was derived. Figure 5.5 shows the posterior distribution of $f(\cdot)$.

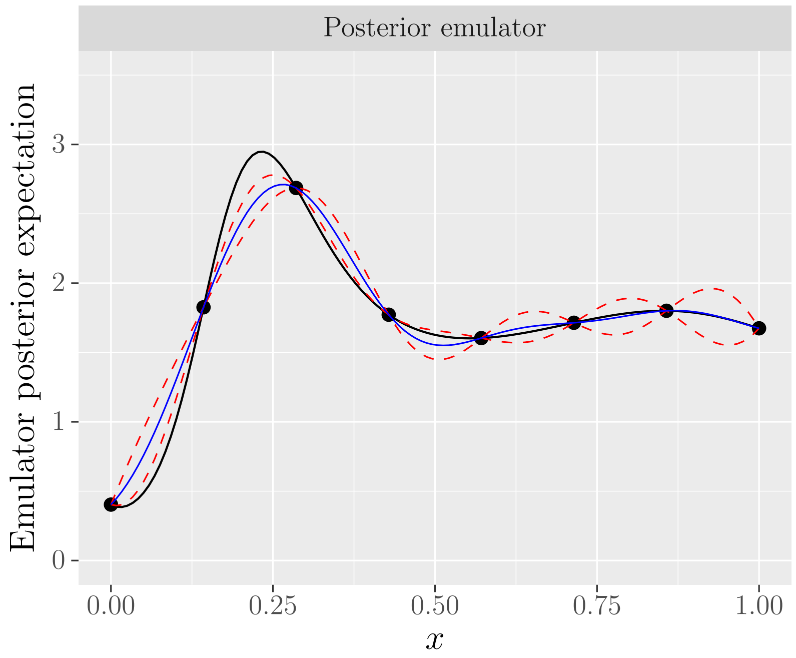

Figure 5.5: The posterior distribution of $f(\cdot)$ for the stationary Gaussian process emulator. The posterior mean is the blue line and the true simulator is the black line with the point-wise 95% credible intervals as the dotted lines. The emulator is overconfident in a part of the input space and underconfident in another part.

The plot shows the posterior mean, the blue line, and the true simulator, the black line, with the 95% credible intervals, the dotted lines, for the stationary Gaussian process emulator. It can be shown that the emulator is overconfident between around 0 and 0.4 in the nonsmooth part of the input space where the 95% credible intervals are small and the simulator lies well outside the 95% credible intervals. Moreover, the emulator is a little underconfident between around 0.4 and 1 in the smooth part of the input space as the 95% credible intervals are wide and the simulator always lies inside the 95% credible intervals for any input between around 0.6 and 1.

For the CGP emulator, we assumed that the prior uncertainty on the simulator is represented by two independent Gaussian processes. The first one is $Z_{global}(\mathbf{x}) \sim GP(\mu, \tau^2 g(\cdot,\cdot))$, with constant mean μ and covariance function $\tau^2 g(\cdot,\cdot)$. The second one is $Z_{local}(\mathbf{x}) \sim GP(0, \sigma^2 l(\cdot,\cdot))$, with mean 0 and correlation function $\sigma^2 l(\cdot,\cdot)$. The correlation functions $g(x, x', \theta)$ and $l(x, x', \alpha)$ were chosen to be the squared exponential correlation function, equations (5.3.6) and (5.3.7). We used the maximum likelihood for estimating the correlation length parameters and then we derived the CGP emulator. Figure 5.6 shows the components of the CGP emulator.

The first and the second panels in the first row show prior samples from the $Z_{global}(\mathbf{x})$ and the $Z_{local}(\mathbf{x})$ where the samples are smooth. In order to represent our prior uncertainty about the simulator, we combine the $Z_{global}(\mathbf{x})$ and the $Z_{local}(\mathbf{x})$. The last panel in the first row shows the plot of $Z_{global}(\mathbf{x}) + Z_{local}(\mathbf{x})$ where the function is also smooth.

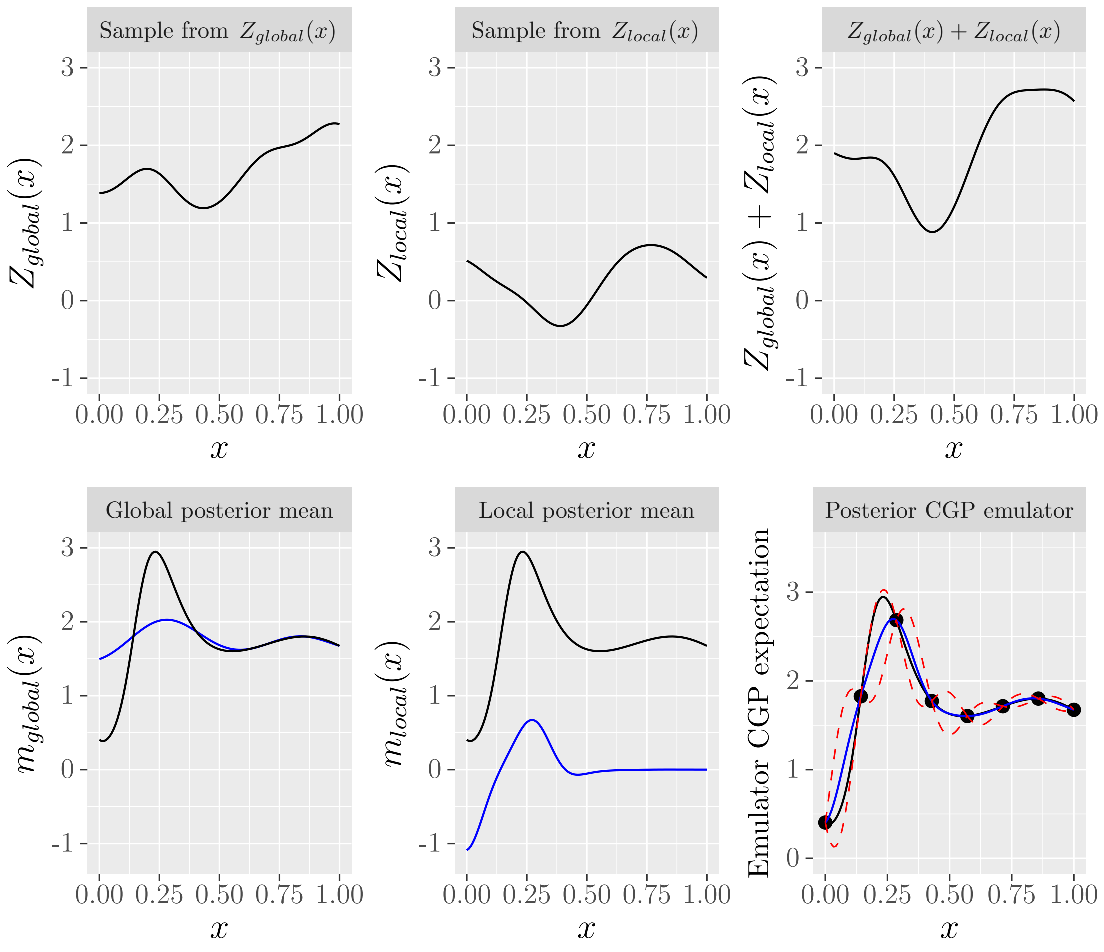

Figure 5.6: Plots of the components of the CGP emulator. The first row shows plots of $Z_{global}(\mathbf{x})$, $Z_{local}(\mathbf{x})$ and $Z_{global}(\mathbf{x}) + Z_{local}(\mathbf{x})$. The first and the second panels in the second row show the blue line as the global posterior mean and the local posterior mean whereas the black line is the true simulator. The global posterior mean is close to true simulator in the smooth part, between around 0.4 and 1, of the input space. The shape of the local posterior mean mimics the shape of the true simulator. The last panel in the second row shows the true simulator, the black line, and the posterior mean, the blue line, with the point-wise 95% credible intervals, the dotted lines. The coverage of the 95% credible intervals is good.

The first panel in the second row shows that the global posterior mean, the blue line, is close to the true simulator, the black line, in the smooth part of the input space, between around 0.4 and 1. The second panel in the second row shows that the shape of the local posterior mean mimics the shape of the true simulator in the nonsmooth part and the smooth part. The last panel in the second row shows that the posterior mean is more closer to the true simulator than that for stationary Gaussian process emulator in most of the input space. The 95% credible intervals are now smaller in the smooth part of the input space. Moreover, the coverage of the 95% credible intervals is good in the nonsmooth part of the input space. Thus, the performance of the CGP emulator is better than that of the stationary emulator in that the CGP emulator is able to deal with behaviour of this nonstationary function.

5.4 The coverage interval diagnostic for TGP and CGP models

In this section, we consider the coverage interval diagnostic, $K_\alpha^{CI}(\cdot)$, to investigate whether the Gaussian process assumption is suitable for building TGP and CGP emulators. In order to calculate the observed values of the coverage interval diagnostic, we can use the same procedure that is described in Section 4.2.

In order to obtain the reference distribution of $K_\alpha^{CI}(\cdot)$, the simulation-based method can be used. Suppose we have a TGP emulator or a CGP emulator for $f(\cdot)$ based on n training points. Suppose also we have a set of m validation inputs, $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$. We first simulate outputs from the posterior distribution of $f(\cdot)$. Then, we calculate the coverage interval diagnostic with different values of α between 0.1 and 0.95. The procedure of using simulation-based method to obtain samples of the diagnostic $K_\alpha^{CI}(\cdot)$ for the TGP model and the CGP model can be achieved according to the following algorithm.

Algorithm 3 This algorithm generates samples of the coverage interval diagnostic from its reference distribution

```
Inputs: A TGP or a CGP emulator, $p(f(\cdot)|f(\mathbf{x}_1),\ldots,f(\mathbf{x}_n))$.
        $m$ validation inputs, $\mathbf{x}_1^*,\ldots,\mathbf{x}_m^*$.
1: For $i=1$ to $m$, calculate the $(1-\alpha)100\%$ credible intervals for each validation
   output based on the posterior mean, $\mathrm{E}[f(\mathbf{x}_i^*)|\mathbf{y}]$, and the posterior variance,
   $V[f(\mathbf{x}_i^*)|\mathbf{y}]$, where the lower and upper bounds of the $(1-\alpha)100\%$ credible
   intervals are given by
     $L_\alpha(\mathbf{x}_i^*) = \mathrm{E}[f(\mathbf{x}_i^*)|\mathbf{y}] - z_{\frac{\alpha}{2}}\sqrt{V[f(\mathbf{x}_i^*)|\mathbf{y}]}$   (5.4.1)
     $U_\alpha(\mathbf{x}_i^*) = \mathrm{E}[f(\mathbf{x}_i^*)|\mathbf{y}] + z_{\frac{\alpha}{2}}\sqrt{V[f(\mathbf{x}_i^*)|\mathbf{y}]}$   (5.4.2)
   end for.
2: For $j=1$ to $J$, simulate outputs $\mathbf{y}^{*(j)} = \{f_{(j)}(\mathbf{x}_1^*),\ldots,f_{(j)}(\mathbf{x}_m^*)\}$ from the
   multivariate normal distribution with the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}]$, and the
   posterior variance, $V[\mathbf{y}^*|\mathbf{y}]$.
3: Calculate the coverage interval diagnostic
     $K_\alpha^{CI}(\mathbf{y}^{*(j)}) = \sum_{i=1}^{m} I\left\{L_\alpha(\mathbf{x}_i^*) < f_{(j)}(\mathbf{x}_i^*) < U_\alpha(\mathbf{x}_i^*)\right\}.$   (5.4.3)
   end for.
Output: $K_\alpha^{CI}(\mathbf{y}^{*(i)}),\ldots,K_\alpha^{CI}(\mathbf{y}^{*(J)})$, a sample from the distribution of $K_\alpha^{CI}(\cdot)$
        assuming valid emulator.
```

For the TGP emulator, the location of the tree is uncertain where at each iteration, we may have different tree. Thus, it may not be appropriate to sample from the multivariate normal distribution with the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}]$, and then calculate the coverage interval diagnostic based on these samples.

However, we can start with the multivariate normal distribution to be the approximate distribution of the outputs of the TGP emulator. So, in order to obtain the distribution of the coverage interval diagnostic for the TGP emulator, we will approximate the posterior TGP emulator by the multivariate normal distribution and we will suppose that this approximate distribution is good for the outputs of the TGP emulator.

If the location of the tree does not change significantly at each iteration, the normality assumption may be then appropriate for these samples. Thus, we can calculate the coverage interval diagnostic for these simulated outputs. If the location of the tree change significantly at each iteration, the multivariate normal distribution may not be appropriate to sample from it.

Moreover, for the TGP emulator, we can also obtain the observed values of the coverage interval diagnostic and their reference distribution in each of the tree partitions. Thus, we can investigate whether the Gaussian process assumption is suitable for building the stationary Gaussian process emulator in each partition of the tree, $\mathcal{T}$. Suppose we have m_ν validation inputs in the r_ν region, $\mathbf{X}^*_\nu$, and evaluations $\mathbf{y}^*_\nu$ of the simulator outputs at these validation inputs. Therefore, we can obtain m_ν credible intervals for the validation outputs in each partition, each of which should have a probability of $1-\alpha$ of containing the validation outputs, $\mathbf{y}^*_\nu$.

In order to obtain the reference distribution of $K_{\alpha}^{CI}(\cdot)$ for the Gaussian process emulator in each of the tree partitions, we can use the simulation-based method. Suppose we have a stationary Gaussian process emulator in each partition of the tree based on n_{ν} training points. We first simulate outputs from the posterior distribution of $f(\cdot)$ for each partition. Then, we calculate the coverage interval diagnostic with different values of α. The procedure of using simulation-based method to obtain samples of the coverage interval diagnostic, $K_{\alpha}^{CI}(\cdot)$, for each partition in the tree can be achieved by using Algorithm 3 but for the Gaussian process emulator in each of the tree partitions.

5.5 Modified borehole model illustrative example

In this section, we consider the stationary Gaussian process, TGP and CGP emulators for an illustrative simulator: a modification of the borehole model. Suppose the simulator is given by

$$f(\mathbf{x}) = \frac{2\pi T_u(H_u - H_l)}{\ln(\frac{r}{r_w})\left(1 + \frac{2LT_u}{\ln(\frac{r}{r_w})r_w^2 K_w} + \frac{T_u}{T_l}\right)} + g(x_9), \tag{5.5.1}$$

where

$$g(x_9) = \begin{cases} \sin(\frac{x_9}{10}) + 0.3\sin(x_9), & \text{if } x_9 > -2 \\ 80 + \frac{\cos(x_9)}{20}, & \text{otherwise} \end{cases},$$

where $\mathbf{x} = (r_w, r, T_u, H_u, T_l, H_l, L, K_w, x_9)$. The nine input variables with their ranges and units are as follows:

- $r_w \in [0.05, 0.15](m)$ is the radius of borehole.
- $r \in [100, 50000](m)$ is the radius of influence.

- $T_u \in [63070, 115600](m^2/year)$ is the transmissivity of upper aquifer.
- $H_u \in [990, 1110](m)$ is the potentiometric head of upper aquifer.
- $T_l \in [63.1, 116](m^2/year)$ is the transmissivity of lower aquifer.
- $H_l \in [700, 820](m)$ is the potentiometric head of lower aquifer.
- $L \in [1120, 1680](m)$ is the length of borehole.
- $K_w \in [9855, 12045](m/year)$ is the hydraulic conductivity of borehole.
- $x_9 \in [-6, 2]$ is a dummy input to introduce discontinuity.

We generated $n = 100$ training inputs by a LHD and each input variable was transformed to be in the interval $[-1, 1]^9$. Then, we evaluated simulator outputs of model (5.5.1) at the $n = 100$ inputs, $\mathbf{y} = (y_1 = f(\mathbf{x}_1), \dots, y_{100} = f(\mathbf{x}_{100}))$. We then derived the stationary Gaussian process emulator based on the $n = 100$ inputs using a linear mean with $\mathbf{h}(\mathbf{x})^T = (1, \mathbf{x}^T)$ and the covariance matrix, $V = \sigma^2 C(\mathbf{x}, \mathbf{x}')$, with the squared exponential correlation function $C(\mathbf{x}, \mathbf{x}')$ given in equation (2.3.4). We also derived the TGP emulator and the CGP emulator based on the $n = 100$ inputs, where we used the `tgp` package in R for the TGP emulator. Then, we validated the stationary Gaussian process emulator, the TGP emulator and the CGP emulator with $m = 24$ validation inputs, also generated by a LHD.

For the TGP emulator, Figure 5.7 shows that the MAP tree partitions the input space at $x_9 = -0.212121$ with height 2. The number of the training points in the first partition is 40 with $\hat{\sigma}_1^2 = 0.0013$ whereas the number of the training points in the second partition is 60 with $\hat{\sigma}_2^2 = 0.001$.

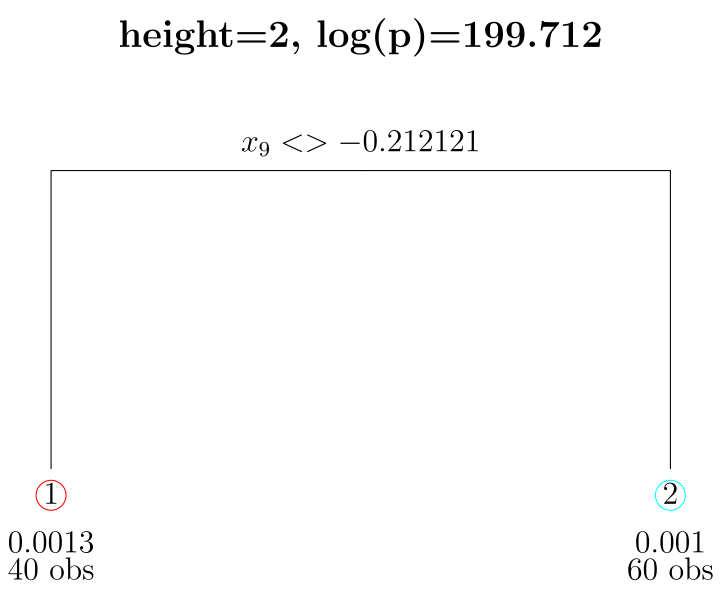

Figure 5.7: The MAP tree of height 2 with the number of training points, n_ν, and $\hat{\sigma}^2_\nu$ at each leaf using 100 training points. The partition is at $x_9 = -0.212121$.

5.5.1 The coverage interval diagnostic for the Modified borehole model

We applied the coverage interval diagnostic on the stationary Gaussian process emulator, the TGP emulator and the CGP emulator that are built on the modified borehole model in (5.5.1). We calculated the $(1-\alpha)100\%$ credible intervals for each validation output. Then, we calculated the observed values of the coverage interval diagnostic, equation (4.2.3), with different values of, α.

The reference distribution of the coverage interval diagnostic, $K^{CI}_\alpha(\cdot)$, is unknown because the values of the terms in the sum, in equation (4.2.3), are not independent. Hence, to obtain the reference distribution of the coverage interval diagnostic, $K_{\alpha}^{CI}(\cdot)$, the simulation-based method was used. For the stationary Gaussian process emulator, we sampled 1000 vectors of 100 validation outputs from the multivariate Student-t distribution with 90 degrees of freedom, the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$ equation (2.3.28), and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$ equation (2.3.29). Then, we obtained simulated values of the coverage interval diagnostic, $K_{\alpha}^{CI}(\mathbf{y}^*_{sim})$, according to Algorithm 1.

Figure 5.8 shows the observed coverage interval diagnostic values (as proportions) and the mean values of their simulated values against different values of $(1-\alpha)$ with 95% credible intervals for the stationary Gaussian process emulator. We used the 2.5% sample quantile as the lower bound of the 95% credible intervals and the 97.5% sample quantile as the upper bound of the 95% credible intervals.

It can be seen that all the observed values of the coverage interval diagnostic lie under their expectations with 5 observed values of the coverage interval diagnostic lying outside the 95% credible intervals. This suggests that using stationary Gaussian process emulator for the whole input space may not suitable for the modified borehole model.

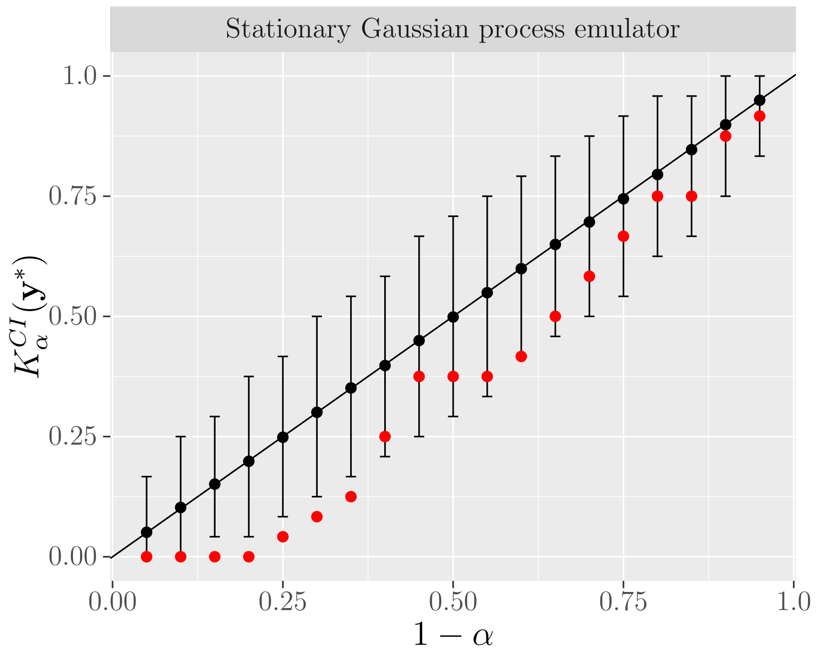

Figure 5.8: The observed coverage interval diagnostic values as red points and the mean values of the corresponding simulated values as black points against different values of $(1-\alpha)$ with 95% credible intervals for the stationary Gaussian process emulator of the modified borehole model based on 100 training points and 24 validation points. The plot shows that the emulator is overconfident for smaller credible intervals.

In order to obtain the reference distribution of the coverage interval diagnostic for the TGP and the CGP emulators, we sampled 1000 simulated outputs for each validation input from the multivariate normal distribution with the posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}]$, of the emulators. Then, we obtained simulated values of coverage interval diagnostic, $K_{\alpha}^{CI}(\mathbf{y}_{sim}^*)$, according to Algorithm 3. We calculated the simulated values of coverage interval diagnostic with different values of α. Figure 5.9 shows the observed values (as proportions) of the coverage interval diagnostic and the mean values of their simulated values against different values of $(1-\alpha)$ with 95% credible intervals for the TGP emulator and the CGP emulator based on 100 training points.

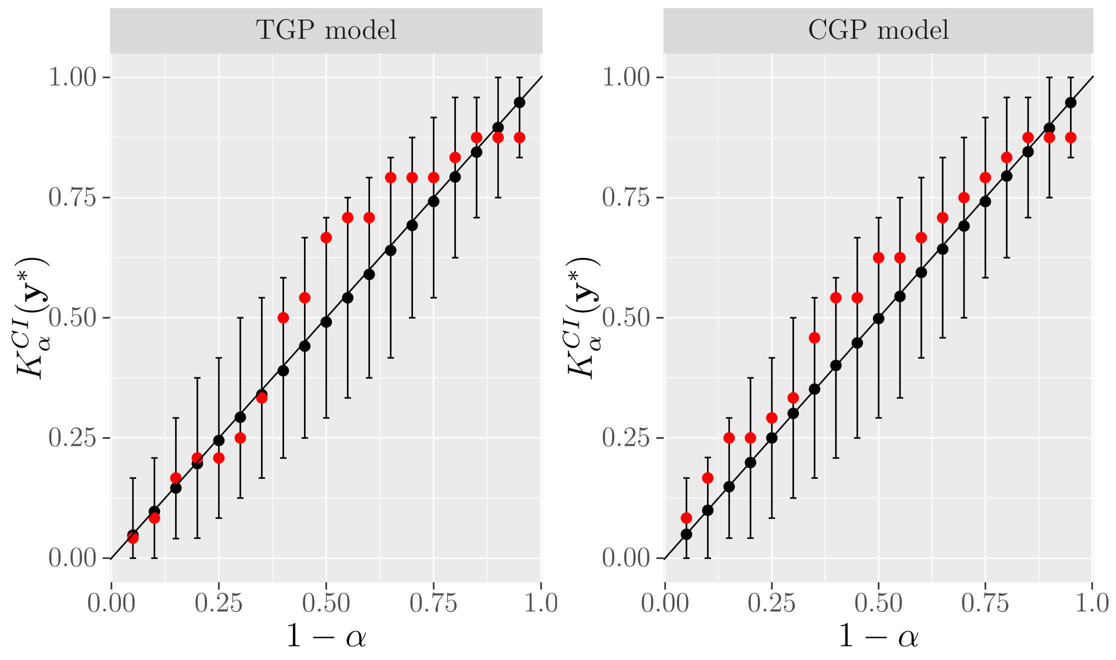

Figure 5.9: The observed values of the coverage interval diagnostic as red points and the mean values of the corresponding simulated values as black points against different values of $(1-\alpha)$ with 95% credible intervals for the TGP and the CGP emulators of the modified borehole model based on 100 training points and 24 validation points. The plots show that all the observed values lie inside the 95% credible intervals for both TGP and CGP emulators.

We can see that, for both the TGP emulator and the CGP emulator, most of the observed values of the coverage interval diagnostic are close to their expectations. Also, all the observed values of the coverage interval diagnostic, $K_{\alpha}^{CI}(\mathbf{y}_{obs})$, lie inside the 95% credible intervals. This indicates that the coverage is reasonable and the Gaussian process could be a reasonable choice for this data.

The overall coverage of the input space for the TGP emulator is acceptable. However, it is not necessary that the coverage is also acceptable in the partitions. For example, it is possible to obtain an underconfident emulator in one partition and an overconfident emulator in another partition but the overall coverage is sensible. Hence, we now investigate coverage properties in each partition of the input space. We calculated the $(1-\alpha)100\%$ credible intervals for each validation output in each of the tree partitions. In partition 1, there are $n_1 = 40$ training inputs, $\mathbf{X}_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_{40}\}$, and $m_1 = 10$ validation inputs, $\mathbf{X}_1^* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_{10}^*\}$. Thus, the training outputs in the first partition are $\mathbf{y}_1 = \{y_1, \ldots, y_{40}\}$ and the validation outputs are $\mathbf{y}_1^* = \{y_1^*, \ldots, y_{10}^*\}$. In partition 2, there are $n_2 = 60$ training inputs, $\mathbf{X}_2 = \{\mathbf{x}_{41}, \ldots, \mathbf{x}_{100}\}$, and $m_2 = 14$ validation inputs, $\mathbf{X}_2^* = \{\mathbf{x}_{11}^*, \ldots, \mathbf{x}_{24}^*\}$. Hence, the training outputs are $\mathbf{y}_2 = \{y_{41}, \ldots, y_{100}\}$ and the validation outputs are $\mathbf{y}_2^* = \{y_{11}^*, \ldots, y_{24}^*\}$.

According to the TGP emulator, we split the posterior mean into two vectors. The first vector is $m_1(\mathbf{X}_1^*) = \{m_1(\mathbf{x}_1^*), \ldots, m_1(\mathbf{x}_{10}^*)\}$ as the posterior mean of the partition 1 emulator. The second vector is $m_2(\mathbf{X}_2^*) = \{m_2(\mathbf{x}_{11}^*), \ldots, m_2(\mathbf{x}_{24}^*)\}$ as the posterior mean of the partition 2 emulator where $m_\nu(\mathbf{X}_\nu^*)$, for $\nu = 1, 2$, is the posterior mean of the emulators in each partition, equation (5.2.5). Also, we split the posterior covariance matrix of the TGP emulator into two matrices. The first matrix is

$$V_1(\mathbf{X}_1^*, \mathbf{X}_1^{*'}) = \begin{pmatrix} V_1(\mathbf{x}_1^*, \mathbf{x}_1^*) & \cdots & V_1(\mathbf{x}_1^*, \mathbf{x}_{10}^*) \\ \vdots & \ddots & \vdots \\ V_1(\mathbf{x}_{10}^*, \mathbf{x}_1^*) & \cdots & V_1(\mathbf{x}_{10}^*, \mathbf{x}_{10}^*) \end{pmatrix}$$

as the posterior covariance matrix of the partition 1 emulator and the second matrix is

$$V_2(\mathbf{X}_2^*, \mathbf{X}_2^{*'}) \;=\; \begin{pmatrix} V_2(\mathbf{x}_{11}^*, \mathbf{x}_{11}^*) & \cdots & V_2(\mathbf{x}_{11}^*, \mathbf{x}_{24}^*) \\ \vdots & \ddots & \vdots \\ V_2(\mathbf{x}_{24}^*, \mathbf{x}_{11}^*) & \cdots & V_2(\mathbf{x}_{24}^*, \mathbf{x}_{24}^*) \end{pmatrix}$$

as the posterior covariance matrix of the partition 2 emulator where $V_\nu(\mathbf{X}_\nu^*, \mathbf{X}_\nu^{*'})$, for $\nu = 1, 2$, is the posterior covariance of the emulators in each partition, equation (5.2.6).

Then, we calculated the observed values of the coverage interval diagnostic with different values of α for the emulator in each partition

$$K_\alpha^{CI}(\mathbf{y}_\nu^*) = \sum_{i=1}^{m_\nu} I\left\{L_\alpha(\mathbf{x}_i^*) < f(\mathbf{x}_i^*) < U_\alpha(\mathbf{x}_i^*)\right\} \tag{5.5.2}$$

for $\nu = 1, 2$. In order to obtain the distribution of the coverage interval diagnostic, we sampled 1000 simulated outputs for each validation input from the multivariate normal distribution with $\mathrm{E}[\mathbf{y}_\nu^*|\mathbf{y}_\nu]$ and $V[\mathbf{y}_\nu^*|\mathbf{y}_\nu]$, where $\mathrm{E}[\mathbf{y}_\nu^*|\mathbf{y}_\nu]$ and $V[\mathbf{y}_\nu^*|\mathbf{y}_\nu]$, for $\nu = 1, 2$, are the posterior mean and the posterior variance for each emulator in the partitions. Then, we obtained simulated values of coverage interval diagnostic for each partition according to Algorithm 3 for the emulator in each of the tree partitions with different values of α.

Figure 5.10 shows the observed values (as proportions) of the coverage interval diagnostic and the mean values of their simulated values against different values of $(1 - \alpha)$ with 95% credible intervals for the validation output in each partition.

It can be seen that some observed values of the coverage interval diagnostic, the red points, for partition 1 are close to their lower bounds of the 95% credible intervals and not close to their expectations, but all of the observed values lie inside the 95% credible intervals. For partition 2, although some observed values of the coverage interval diagnostic are close to their upper bounds of the 95%

credible intervals and not close to their expectations, all the observed values of the coverage interval diagnostic lie inside the 95% credible intervals. This suggests that the Gaussian process assumption is valid for the emulator in partition 1 and the emulator in partition 2. Also, we can see that the 95% credible intervals for the observed values of the coverage interval diagnostic for partition 1 are wider than those for partition 2.

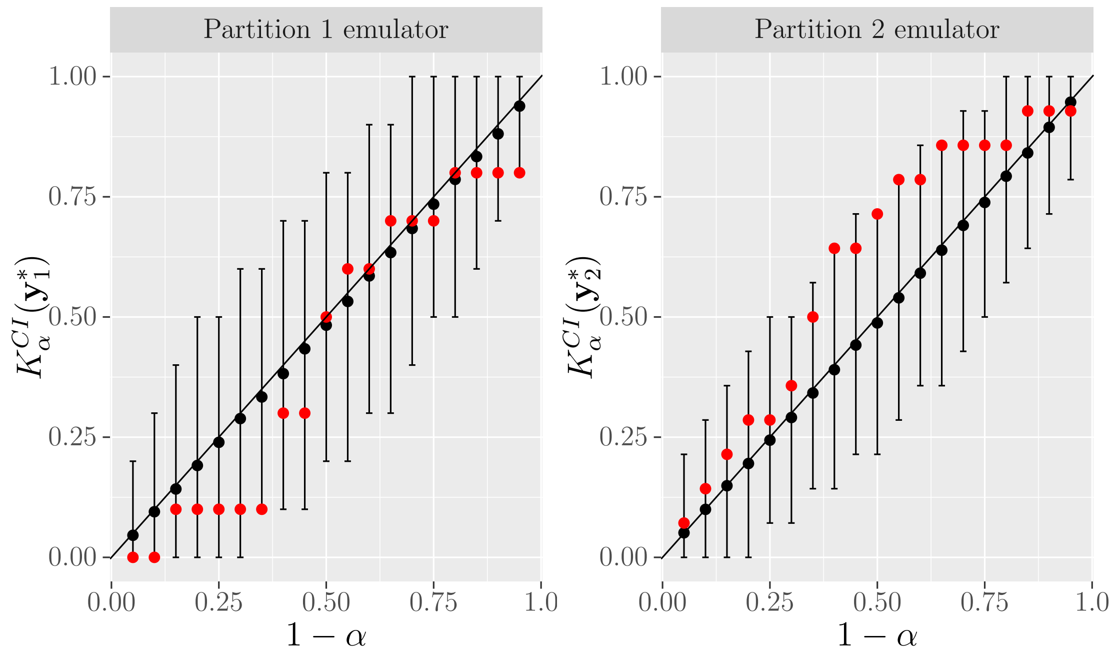

Figure 5.10: The observed values of the coverage interval diagnostic as red points and the mean values of the corresponding simulated values as black points against different values of $(1-\alpha)$ with 95% credible intervals for the emulators in each of the tree partitions. The emulator in partition 1 is based on 40 training points and 10 validation points whereas the emulator in partition 2 is based on 60 training points and 14 validation points. The plots show that some point close to their lower bounds of the 95% credible intervals for partition 1 and some point close to their upper bounds of the 95% credible intervals for partition 2.

5.5.2 The plot of pivoted Cholesky errors and the scaled conditional standard deviations

In this section, we consider the plots of pivoted Cholesky errors against the pivoting order and the pivoted Cholesky errors against the scaled conditional standard deviations for the stationary Gaussian process, TGP and the CGP emulators. We calculated pivoted Cholesky errors by equation (3.4.16) where $\mathbf{P}^T V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]\mathbf{P} = \mathbf{R}^T\mathbf{R}$. So $\mathbf{G}_{PC} = \mathbf{PR}$ and $\mathbf{P}$ is a permutation matrix.

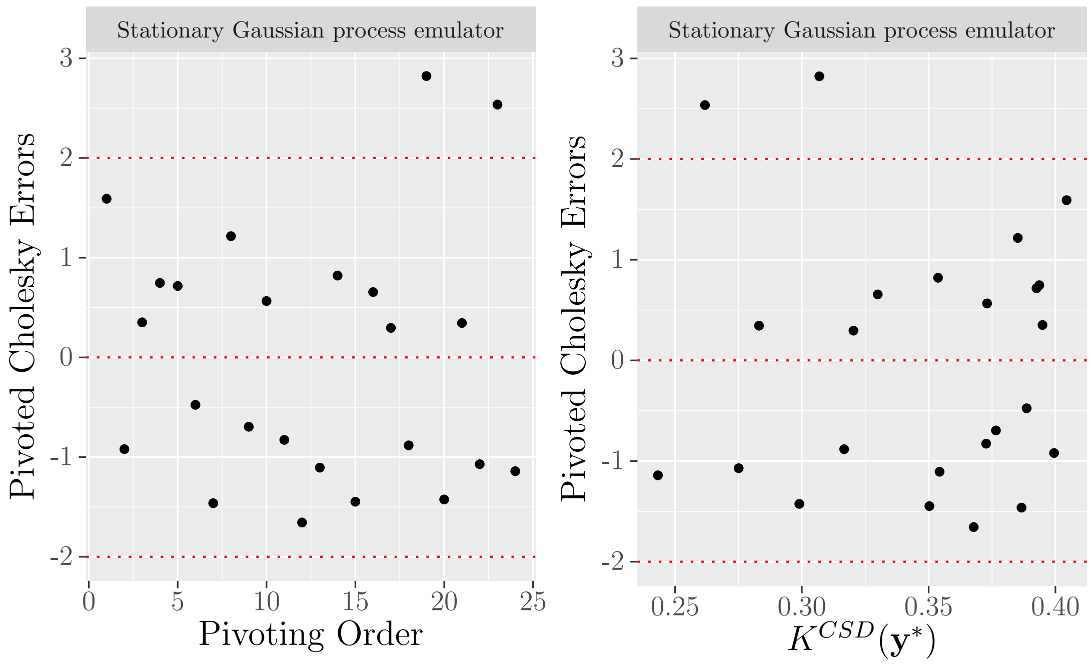

Figure 5.11: Plots of the pivoted Cholesky errors against the pivoting order and the pivoted Cholesky errors against the scaled conditional standard deviations for the stationary Gaussian process emulator of the modified borehole model based on 100 training points and 24 validation points. Large conditional standard deviations are seen in the plot.

Figure 5.11 shows the plots of the pivoted Cholesky errors against the pivoting order and the pivoted Cholesky errors against the scaled conditional standard deviations for the stationary Gaussian process emulator of the modified borehole model. The posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, and the posterior variance, $V[\mathbf{y}^*|\mathbf{y}, \boldsymbol{\delta}]$, of the stationary Gaussian process emulator are given in equations (2.3.28) and (2.3.29).

It can be shown that most of the pivoted Cholesky errors are small with only two errors lying outside the bounds. The scaled conditional standard deviations, however, can be considered slightly large since most of them are between 0.3 and 0.4.

Figure 5.12 shows the plot of pivoted Cholesky errors against the pivoting order and the plot of pivoted Cholesky errors against the scaled conditional standard deviations for the TGP and the CGP emulators. The posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}]$, and the posterior variance of the TGP emulator obtained using the `tgp` package. The posterior mean, $\mathrm{E}[\mathbf{y}^*|\mathbf{y}]$, and the posterior variance of the CGP emulator are given in equations (5.3.13) and (5.3.14).

It can be seen that most of the pivoted Cholesky errors for the TGP emulator are small, but three pivoted Cholesky errors lie outside the bounds. The scaled conditional standard deviations are small since most of them are close to zero. The scaled conditional standard deviations are smaller than those from the stationary Gaussian process emulator. We can also notice that the scaled conditional standard deviations for partition 1 are separated from the scaled conditional standard deviations for partition 2 (see the ranges of the scaled conditional standard deviations for the partitions in Figure 5.13). The uncertainty in partition 2 is larger than the uncertainty in partition 1 and this may be because there are fewer design points in partition 1 than in partition 2. This confirms that the plot of the pivoted Cholesky errors against the scaled conditional standard deviations is more informative than the plot of the pivoted Cholesky errors against the pivoting order.

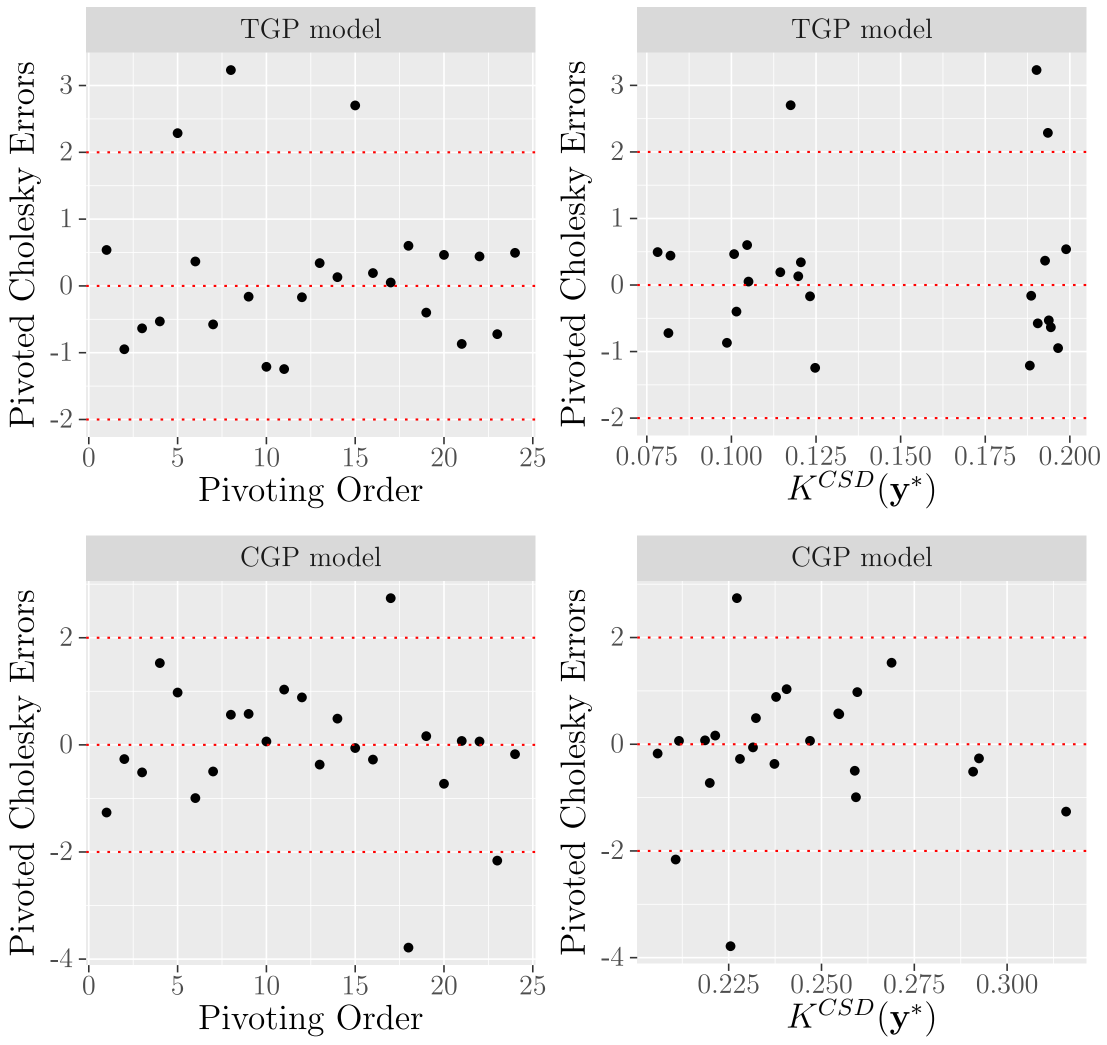

Figure 5.12: The plot of the pivoted Cholesky errors, $K^{PC}(\mathbf{y}^*)$, against the pivoting order and the plot of the pivoted Cholesky errors against the scaled conditional standard deviations, $K^{CSD}(\mathbf{y}^*)$, for TGP and CGP emulators of model (5.5.1) based on 100 training points and 24 validation points. The scaled conditional standard deviations are small.

For the CGP emulator, it can be shown that most of the pivoted Cholesky errors are small, but three pivoted Cholesky errors lie outside the bounds. The scaled conditional standard deviations are small and they are between 0.2 and 0.325, indicating also that the points are close to each other and have almost the same level of uncertainty. The scaled conditional standard deviations are also smaller than those from the stationary Gaussian process emulator. We can also notice that the large pivoted Cholesky errors correspond to small scaled conditional standard deviations. In addition, the pivoted Cholesky errors decrease as the scaled conditional standard deviations increase.

We also considered the plot of pivoted Cholesky errors against the pivoting order and the plot of pivoted Cholesky errors against the scaled conditional standard deviations for the emulators in each partition of the tree, Figure 5.13.

Most of the pivoted Cholesky errors in partition 1 are very small, but two pivoted Cholesky errors lie outside the bounds. The scaled conditional standard deviations are small. We can see that the pivoted Cholesky errors decrease as the scaled conditional standard deviations increase. For partition 2, most of the pivoted Cholesky errors are also small with only one pivoted Cholesky error lying outside the bounds. The scaled conditional standard deviations are also small. The scaled conditional standard deviations in partition 2 are smaller than those for partition 1.

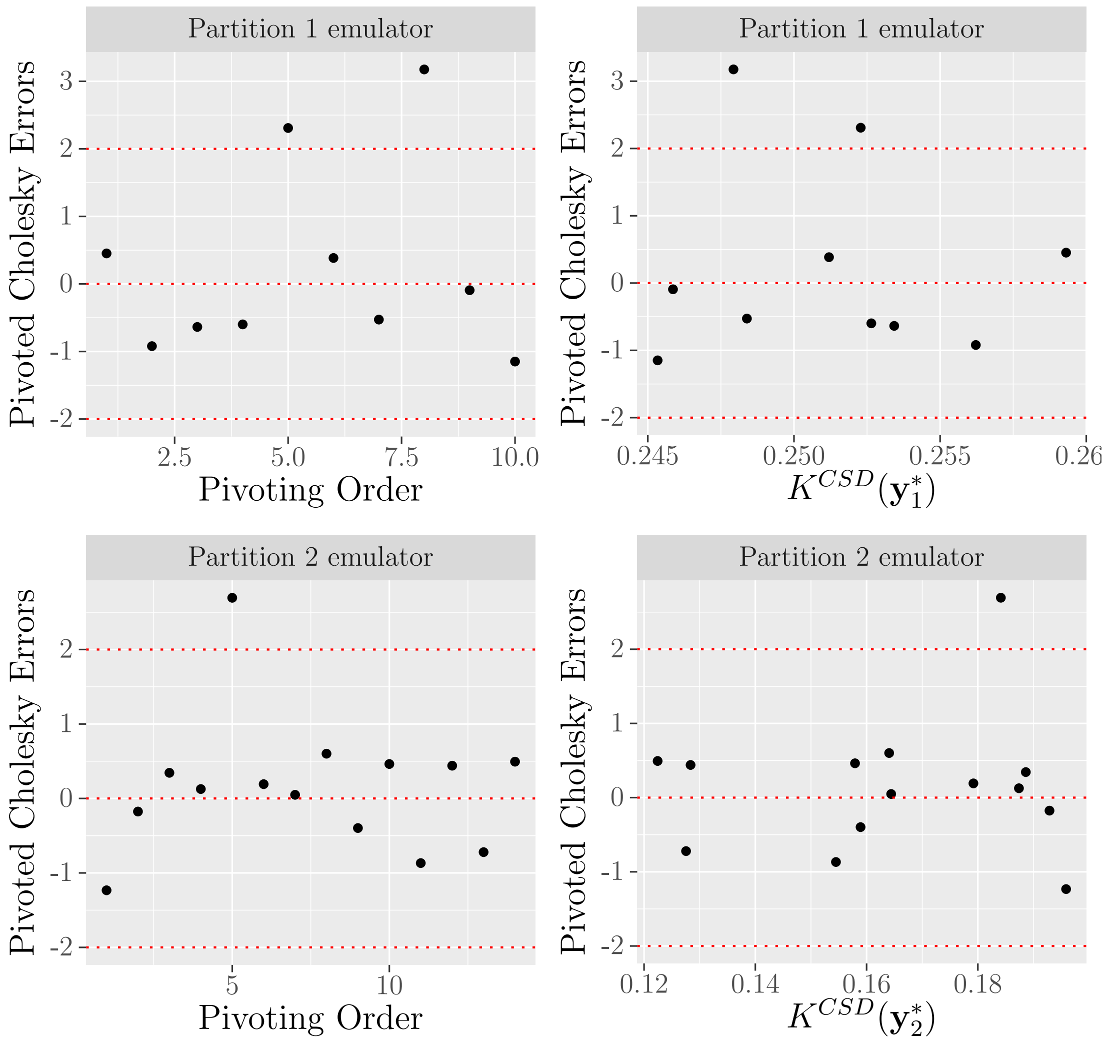

Figure 5.13: The plot of the pivoted Cholesky errors against the pivoting order and the plot of the pivoted Cholesky errors against the scaled conditional standard deviations for the emulators in each of the tree partitions. The emulator in partition 1 is based on 40 training points and 10 validation points whereas the emulator in partition 2 is based on 60 training points and 14 validation points. The scaled conditional standard deviations for partition 1 are larger than those for partition 2.

5.6 Conclusion

In this chapter, we have reviewed some complex Gaussian process models that can be used for dealing with nonstationary models. In addition, we have investigated the performance of diagnostic methods for these advanced emulators. We have shown how coverage interval diagnostic can detect whether the Gaussian process assumption is suitable for building both the TGP emulator and the CGP emulator or not. We have also examined the performance of some diagnostic methods for TGP and CGP emulators with an illustrative example.

The coverage interval diagnostic measures the coverage properties of $(1-\alpha)100\%$ posterior credible intervals. Thus, it depends on the validation outputs, the lower and the upper bounds of the $(1-\alpha)100\%$ posterior credible intervals for the validation outputs. Moreover, the distribution of the coverage interval diagnostic based on sampling simulated outputs from the posterior distribution of $f(\cdot)$ and the coverage interval diagnostic for these samples. Hence, as long as we can obtain the posterior emulator, calculating the coverage interval diagnostic and obtaining its distribution will be straightforward with no matter how complex the emulator is. So the coverage interval diagnostic and its distribution can be applied for stationary Gaussian process emulator and advanced Gaussian process emulators.

Chapter 6

Conclusions

In this chapter, the main contributions of this thesis are reviewed. The thesis was concerned with diagnostic methods for validating Gaussian process emulators. In Section 6.1, the summary and the main finding of the thesis chapters are reviewed. Section 6.3 presents a number of recommendations based on the finding of our results. Section 6.4 presents several possible ideas for the future work.

6.1 Summary of the thesis chapters and key developments

The present thesis consists of six chapters. In Chapter 1, an introduction to computer models was given with a number of applications of computer models in different areas of science. In addition, there was an explanation for the need of surrogate models for tackling the computationally expensive problem of computer models.

In Chapter 2, the concept of Gaussian process emulators was introduced with literature search for the components of Gaussian process emulators as well as applications for Gaussian process emulators in different fields. We reviewed several possible choices for the mean and covariance functions that have been used in building Gaussian process emulators. Methods that have been used in literature for estimating the correlation parameters were also reviewed in this chapter. Finally, we presented several designs for training and validation inputs in building Gaussian process emulators.

In Chapter 3, the concept of the overconfidence and underconfidence of Gaussian process emulators was explained. The discussions concerned with situations when assumptions of Gaussian process emulators may not be well specified. The concept of diagnostics for Gaussian process emulators was presented with two different methods: separate validation sets and the cross-validation method. Moreover, we reviewed two kinds of current diagnostics for validating Gaussian process emulators: simple diagnostics and diagnostics that consider predictions uncertainty. A modification of an existing diagnostic was developed in order to make the plot of this diagnostic more informative. It was important to check the performance of some diagnostic methods for examining assumptions used in building Gaussian process emulators. Thus, we applied several current simple diagnostics and diagnostics that consider uncertainty in emulator predictions on different examples of simulators. The need for more than one diagnostic method for validating Gaussian process emulators was explained.

The purpose of Chapter 4 was to develop the coverage interval diagnostic that can be used to examine the Gaussian process assumption in building emulators. This graphical diagnostic method investigated the coverage properties of $(1-\alpha)100\%$ posterior credible intervals for the validation outputs. It was possible to compare the performance of the coverage interval diagnostic with the QQ-plot and show that the coverage interval diagnostic is more informative. It was clear that the distribution of the coverage interval diagnostic cannot be found analytically. Thus, it was therefore necessary to develop a simulation-based procedure for obtaining the distribution of any diagnostic. The importance of this method is that it can be used to obtain the reference distribution of diagnostics which cannot be obtained analytically. This simulation-based procedure was presented to obtain samples of the coverage interval diagnostic using both the cross-validation method and separate validation set for validating emulators. We also investigated the performance of our diagnostic with data exhibiting different properties to that of a Gaussian process.

The final contribution was extending the diagnostic methods for validating complex Gaussian process emulators, where diagnostics methods were applied for validating nonstationary Gaussian process emulators. Thus, in Chapter 5, we reviewed two kinds of complex Gaussian process emulators that can deal with nonstationary functions. We described the idea of these complex Gaussian process models and the process of constructing these emulators. We presented the procedure of the coverage interval diagnostic and the procedure of the simulation-based method for obtaining the distribution of the coverage interval diagnostic for these complex Gaussian process emulators. The investigation of whether the Gaussian process assumption is suitable in building these complex Gaussian process emulators was assessed by the coverage interval diagnostic with a nonstationary simulator. Other diagnostic methods were also applied for these complex Gaussian process emulators.

6.2 The relationship between diagnostics

In this thesis, we have developed and reviewed a number of diagnostic methods that are based on the comparison between the emulator predictions and the simulator outputs. These diagnostic methods are related to each other and there is a correlation between these different diagnostics. Simple diagnostic methods are based on the differences between the validation outputs and the emulator predictions. Diagnostic methods that take into account uncertainty in the emulator predictions can be seen as extensions of simple diagnostic methods. They are not only depend on differences between the simulator outputs and the emulator predictions, but also consider the uncertainty in the emulator predictions.

The individual standardised errors, (3.4.10), are based on the differences between the validation outputs and the emulator predictions that are standardised by the predictive standard deviation. Each individual standardised error can be perceived as a diagnostic. The Mahalanobis distance, (3.4.13), can be seen as an extension of the individual standardised errors that summarises them in a single value. The Mahalanobis distance diagnostic measures the overall fit of the emulator. When the observed value of the Mahalanobis distance is extreme from its expected value, further investigation is needed. The individual standardised errors can be used to investigate if there is a local problem for some validation points. However, the individual standardised errors may be difficult to interpret due to the correlation among them.

An alternative decomposition of the Mahalanobis distance is the pivoted Cholesky decomposition which produces uncorrelated errors that are mapped to emulator predictions. The sum of squares of the pivoted Cholesky errors is the Mahalanobis distance. Each of the pivoted Cholesky errors can be liked with one of the validation outputs. This helps to investigate the individual large errors and see whether there is a local problem only for some validation points.

In some cases, we may obtain many extreme large and small errors and the value of the Mahalanobis distance is still acceptable. The coverage interval diagnostic is a supplement to the Mahalanobis distance in that it can detect this kind of problems. The coverage interval diagnostic provides a direct assessment of the coverage properties of the credible intervals and so we can investigate whether these $(1-\alpha)100\%$ credible intervals are meaningful or not.

However, when the number of training is large, the different diagnostics will be efficient and show similar results where the emulator predictions will more likely to be good approximations of the simulator outputs. Moreover, the uncertainty in the emulator predictions will be very small.

6.3 Recommendations

In this section, we present a number of recommendations based on our results and our search of the applications of Gaussian process emulators.

- In Table 2.1, most of the authors used simple diagnostics for validating their emulators, so we recommend using diagnostic methods that consider uncertainty in the emulator predictions.

- We propose using separate validation points, if they are available, rather than the cross-validation method for validate Gaussian process emulators. The cross-validation method can be unreliable with a small number of training inputs.

- For the validation points, we suggest using two times the number of the input variables, $2p$. We have seen that increasing the number of validation points has a minor impact on improving the diagnostic performance.

- We recommend using at least two diagnostic methods for validating Gaussian process emulators, but not simply two related methods such as the standardised root mean squared error and the predictivity coefficient.

- If the pivoted Cholesky errors are used as diagnostics, we recommend plotting them against the conditional standard deviations rather than the pivoting order as they are more informative.

- In order to test the Gaussian process assumption for building emulators, our recommendation is using the coverage interval diagnostic with separate validation sets rather than the QQ-plot. The coverage interval diagnostic measures the coverage properties of the $(1-\alpha)100\%$ posterior credible intervals for the validation outputs. This allows us to investigate whether the proportion of the credible intervals that contain the validation outputs is as we expect it to be or not. Moreover, when using the QQ-plot, it is difficult to investigate the overconfidence and the underconfidence of emulators.

- We recommend using the simulation-based method to obtain the distribution of diagnostics when their distribution may not be found analytically.

6.4 Future work

In Chapter 4, it was found that the coverage interval diagnostic with separate validation points succeeded in detecting the behaviour of data from a multivariate Student-t distribution. The coverage interval diagnostic indicated that the Gaussian process assumption is not suitable for multivariate Student-t data. Moreover, it was indicated that the emulator of the nonstationary variance data is underconfident. In order to make Gaussian process assumption suitable for these data, it is necessary to consider transformations methods for these data. It is then possible to build Gaussian process emulators on these transformed data and examine the performance of diagnostics for Gaussian process emulators for these transformed data.

Moreover, this study considered diagnostic methods for univariate (single-output) Gaussian process emulators. Hence, future studies are possible to investigate validating multivariate Gaussian process emulators with separable and nonseparable covariance functions and to examine the performance of diagnostic methods for multivariate Gaussian process emulators.

The distribution of the emulator depends on the correlation parameters. In this thesis, for several emulators, we used the given true values of the correlation parameters to investigate the consequence of estimating the correlation parameters. Moreover, for some other emulators, the plug-in method was considered for the correlation parameters where an estimate $\hat{\boldsymbol{\delta}}$ is considered as the true value of the correlation parameters, $\boldsymbol{\delta}$, without taking into account the uncertainty. However, a Markov Chain Monte Carlo (MCMC) algorithm can be used to obtain samples from the posterior distribution of the correlation length parameters. Thus, it is useful to employ a fully Bayesian analysis to account the uncertainty about the unknown true value of the correlation parameters, $\boldsymbol{\delta}$. This can be significant when the emulator predictions are not close to the simulator outputs or when a small number of design points is used. Hence, it is possible to investigate the performance of diagnostic methods for emulators that are based on samples from the posterior distribution of the correlation length parameters.

Bibliography

Amiri, H. A. A. (2012). The enhancement of a low-frequency electrical heating method by saltwater circulation, *Petroleum Science and Technology*, **30 (5)**: 489–502.

Anderes, E. B. and Stein, M. L. (2008). Estimating deformations of isotropic Gaussian random fields on the plane, *The Annals of Statistics*, pp. 719–741.

Andrianakis, I., Vernon, I. R., McCreesh, N., McKinley, T. J., Oakley, J. E., Nsubuga, R. N., Goldstein, M. and White, R. G. (2015). Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on HIV in Uganda, *PLoS computational biology*, **11 (1)**: e1003968.

Andrianakis, Y. and Challenor, P. (2011). Parameter estimation for Gaussian process emulators.

Ba, S. and Joseph, V. R. (2012). Composite Gaussian process models for emulating expensive functions, *The Annals of Applied Statistics*, **6 (4)**: 1838–1860.

Baggaley, A. W., Boys, R. J., Golightly, A., Sarson, G. R., Shukurov, A. et al. (2012a). Inference for population dynamics in the Neolithic period, *The Annals of Applied Statistics*, **6 (4)**: 1352–1376.

Baggaley, A. W., Sarson, G. R., Shukurov, A., Boys, R. J. and Golightly, A.

(2012b). Bayesian inference for a wave-front model of the Neolithization of Europe, *Physical Review E*, **86 (1)**: 016105.

Barbillon, P., Barthélémy, C. and Samson, A. (2015). Parametric estimation of complex mixed models based on meta-model approach, *arXiv preprint arXiv:1506.03313*.

Bastos, L. (2010). Validating Gaussian process models in computer experiments.

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

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

Bayarri, M. J., Berger, J. O., 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 (2)**.

Ben-Ari, E. N. and Steinberg, D. M. (2007). Modeling data from computer experiments: an empirical comparison of kriging with MARS and projection pursuit regression, *Quality Engineering*, **19 (4)**: 327–338.

Bijak, J., Hilton, J., Silverman, E. and Cao, V. D. (2013). Reforging the Wedding Ring: Exploring a Semi-Artificial Model of population for the United Kingdom with Gaussian process emulators, *Demographic Research*, **29 (27)**: 729–766.

Bounceur, N., Crucifix, M., Wilkinson, R. et al. (2014). Global sensitivity analysis of the climate–vegetation system to astronomical forcing: an emulator-based approach, *Earth System Dynamics Discussions*, **5 (2)**: 901–943.

Bowman, V. E. and Woods, D. C. (2016). Emulation of multivariate simulators using thin-plate splines with application to atmospheric dispersion, *SIAM/ASA Journal on Uncertainty Quantification*, **4 (1)**: 1323–1344.

Chang, E. T., Strong, M. and Clayton, R. H. (2015). Bayesian sensitivity analysis of a cardiac cell model using a Gaussian process emulator, *PloS one*, **10 (6)**: e0130252.

Chang, W., Haran, M., Applegate, P. and Pollard, D. (2016). Calibrating an ice sheet model using high-dimensional binary spatial data, *Journal of the American Statistical Association*, **111 (513)**: 57–72.

Crookston, N. L., Rehfeldt, G. E., Dixon, G. E. and Weiskittel, A. R. (2010). Addressing climate change in the forest vegetation simulator to assess impacts on landscape forest dynamics, *Forest Ecology and Management*, **260 (7)**: 1198–1211.

Currin, C., Mitchell, T., Morris, M. and Ylvisaker, D. (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments, *Journal of the American Statistical Association*, **86 (416)**: 953–963.

Emanuel, K. (2002). A simple model of multiple climate regimes, *Journal of Geophysical Research: Atmospheres*, **107 (D9)**.

Fang, K.-T., Li, R. and Sudjianto, A. (2010). *Design and modeling for computer experiments*, CRC Press.

Fox, J., Weisberg, S., Adler, D., Bates, D., Baud-Bovy, G., Ellison, S., Firth, D., Friendly, M., Gorjanc, G., Graves, S. et al. (2016). Package car.

Galanti, S. and Jung, A. (1997). Low-discrepancy sequences: Monte Carlo simulation of option prices, *The Journal of Derivatives*, **5 (1)**: 63–83.

Gómez-Dans, J. L., Lewis, P. E. and Disney, M. (2016). Efficient emulation of radiative transfer codes using Gaussian processes and application to land surface parameter inferences, *Remote Sensing*, **8 (2)**: 119.

Gramacy, R. (2007). tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models, *Journal of Statistical Software*, **19 (1)**: 1–46.

Gramacy, R. B. and Lee, H. K. (2012). Bayesian treed Gaussian process models with an application to computer modeling, *Journal of the American Statistical Association*.

Gu, M., Berger, J. O. et al. (2016). Parallel partial Gaussian process emulation for computer models with massive output, *The Annals of Applied Statistics*, **10 (3)**: 1317–1347.

Han, M. and Yong Tan, M. H. (2016). Integrated parameter and tolerance design with computer experiments, *IIE Transactions*, **48 (11)**: 1004–1015.

Jandarov, R., Haran, M., Bjørnstad, O. and Grenfell, B. (2014). Emulating a gravity model to infer the spatiotemporal dynamics of an infectious disease, *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **63 (3)**: 423–444.

Johnson, J., Gosling, J. and Kennedy, M. (2011). Gaussian process emulation for second-order Monte Carlo simulations, *Journal of Statistical Planning and Inference*, **141 (5)**: 1838–1848.

Katurji, M., Nikolic, J., Zhong, S., Pratt, S., Yu, L. and Heilman, W. E. (2015). Application of a statistical emulator to fire emission modeling, *Environmental Modelling & Software*, **73**: 254–259.

Kenett, R. and Zacks, S. (1998). *Modern Industrial Statistics: Design and Control of Quality and Reliability*, Duxbury Press, ISBN 9780534353704.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63 (3)**: 425–464.

Kim, Y.-J. and Park, C.-S. (2017). Multi-criterion stochastic optimal selection of a double glazing system, in *Building Simulation*, vol. 10, pp. 1–9, Springer.

Kolachalama, V. B., Bressloff, N. W. and Nair, P. B. (2007). Mining data from hemodynamic simulations via Bayesian emulation, *Biomedical engineering online*, **6 (1)**: 47.

Lan, S., Bui-Thanh, T., Christie, M. and Girolami, M. (2015). Emulation of higher-order tensors in manifold Monte Carlo methods for Bayesian inverse problems, *arXiv preprint arXiv:1507.06244*.

Le Gratiet, L., Marelli, S. and Sudret, B. (2016). Metamodel-based sensitivity analysis: Polynomial chaos expansions and Gaussian processes, *arXiv preprint arXiv:1606.04273*.

Lee, L., Carslaw, K., Pringle, K., Mann, G. and Spracklen, D. (2011). Emulation of a complex global aerosol model to quantify sensitivity to uncertain parameters, *Atmospheric Chemistry and Physics*, **11 (23)**: 12253–12273.

Li, Q., Augenbroe, G. and Brown, J. (2016). Assessment of linear emulators in lightweight Bayesian calibration of dynamic building energy models for parameter estimation and performance prediction, *Energy and Buildings*, **124**: 194–202.

Liu, X. and Guillas, S. (2016). Dimension reduction for emulation: application to the influence of bathymetry on tsunami heights, *arXiv preprint arXiv:1603.07888*.

Machac, D., Reichert, P., Rieckermann, J. and Albert, C. (2016). Fast mechanism-based emulator of a slow urban hydrodynamic drainage simulator, *Environmental Modelling & Software*, **78**: 54–67.

Marrel, A., Marie, N. and De Lozzo, M. (2015). Advanced surrogate model and sensitivity analysis methods for sodium fast reactor accident assessment, *Reliability Engineering & System Safety*, **138**: 232–241.

McDonnell, J., Schunck, N., Higdon, D., Sarich, J., Wild, S. and Nazarewicz, W. (2015). Uncertainty quantification for nuclear density functional theory and information content of new measurements, *Physical review letters*, **114 (12)**: 122501.

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

Miller, R. L., Harding, L. B., Davis, M. J. and Gray, S. K. (2012). Bi-fidelity fitting and optimization, *The Journal of chemical physics*, **136 (7)**: 074102.

Montagna, S. and Tokdar, S. T. (2016). Computer emulation with nonstationary Gaussian processes, *SIAM/ASA Journal on Uncertainty Quantification*, **4 (1)**: 26–47.

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments, *Journal of statistical planning and inference*, **43 (3)**: 381–402.

Nash, J. E. and Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part ia discussion of principles, *Journal of hydrology*, **10 (3)**: 282–290.

Novak, J., Novak, K., Pratt, S., Vredevoogd, J., Coleman-Smith, C. and Wolpert, R. (2014). Determining fundamental properties of matter created in ultrarelativistic heavy-ion collisions, *Physical Review C*, **89 (3)**: 034917.

Oakley, J. and O'Hagan, A. (2002). Bayesian inference for the uncertainty distribution of computer model outputs, *Biometrika*, **89 (4)**: 769–784.

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

Oakley, J. E. and Youngman, B. D. (2017). Calibration of stochastic computer simulators using likelihood emulation, *Technometrics*, **59 (1)**: 80–92.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction, *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–42.

Olson, R., Sriver, R., Goes, M., Urban, N. M., Matthews, H. D., Haran, M. and Keller, K. (2012). A climate sensitivity estimate using Bayesian fusion of instrumental observations and an Earth System model, *Journal of Geophysical Research: Atmospheres (1984–2012)*, **117 (D4)**.

Overstall, A. M. and Woods, D. C. (2016). Multivariate emulation of computer simulators: model selection and diagnostics with application to a humanitarian relief model, *Journal of the Royal Statistical Society: Series C (Applied Statistics)*.

Paciorek, C. J. and Schervish, M. J. (2003). Nonstationary covariance functions for Gaussian process regression., in *NIPS*, pp. 273–280.

Petropoulos, G., Wooster, M., Carlson, T., Kennedy, M. and Scholze, M. (2009). A global Bayesian sensitivity analysis of the 1d SimSphere soil–vegetation–atmospheric transfer (SVAT) model using Gaussian model emulation, *Ecological Modelling*, **220 (19)**: 2427–2440.

Qian, P. Z. (2012). Sliced Latin hypercube designs, *Journal of the American Statistical Association*, **107 (497)**: 393–399.

Rasmussen, C. E. and Williams, C. K. (2006). Gaussian processes for machine learning, *Gaussian Processes for Machine Learning, by CE Rasmussen and CKI Williams. ISBN-13 978-0-262-18253-9*.

Reich, S. and Cotter, C. (2015). *Probabilistic forecasting and Bayesian data assimilation*, Cambridge University Press.

Rojnik, K. and Naveršnik, K. (2008). Gaussian process metamodeling in Bayesian value of information analysis: a case of the complex health economic model for breast cancer screening, *Value in health*, **11 (2)**: 240–250.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989b). Design and analysis of computer experiments, *Statistical science*, pp. 409–423.

Santner, T. J., Williams, B. J. and Notz, W. I. (2003). *The design and analysis of computer experiments*, Springer.

Sarkar, D., Contal, E., Vayatis, N. and Dias, F. (2016). Prediction and optimization of wave energy converter arrays using a machine learning approach, *Renewable Energy*, **97**: 504–517.

Sergienko, E., Gamboa, F. and Busby, D. (2013). Shape invariant model approach for functional data analysis in uncertainty and sensitivity studies, *arXiv preprint arXiv:1304.0861*.

Sexton, J. and Everingham, Y. (2014). Global sensitivity analysis of key parameters in a process-based sugarcane growth model: A Bayesian approach.

Sham Bhat, K., Haran, M., Olson, R. and Keller, K. (2012). Inferring likelihoods and climate system characteristics from climate models and multiple tracers, *Environmetrics*, **23 (4)**: 345–362.

Spracklen, D., Pringle, K., Carslaw, K., Chipperfield, M. and Mann, G. (2005). A global off-line model of size-resolved aerosol microphysics: I. Model development and prediction of aerosol properties, *Atmospheric Chemistry and Physics*, **5 (8)**: 2227–2252.

Tokmakian, R., Challenor, P. and Andrianakis, Y. (2012). On the use of emulators with extreme and highly nonlinear geophysical simulators, *Journal of Atmospheric and Oceanic Technology*, **29 (11)**: 1704–1715.

Vernon, I., Goldstein, M., Bower, R. G. et al. (2010). Galaxy formation: a Bayesian uncertainty analysis, *Bayesian Analysis*, **5 (4)**: 619–669.

Vitsas, P. A. (2016). Commercial simulator applications in flight test training, *Journal of Aerospace Engineering*, **29 (4)**: 04016002.

Wan, H.-P., Mao, Z., Todd, M. D. and Ren, W.-X. (2014). Analytical uncertainty quantification for modal frequencies with structural parameter uncertainty using a Gaussian process metamodel, *Engineering Structures*, **75**: 577–589.

Warner, T. T. (2010). *Numerical weather and climate prediction*, Cambridge University Press.

Weaver, A. J., Eby, M., Wiebe, E. C., Bitz, C. M., Duffy, P. B., Ewen, T. L., Fanning, A. F., Holland, M. M., MacFadyen, A., Matthews, H. D. et al. (2001). The UVic Earth system climate model: Model description, climatology, and applications to past, present and future climates, *Atmosphere-Ocean*, **39 (4)**: 361–428.

Worley, B. A. (1987). Deterministic uncertainty analysis, Tech. rep., Oak Ridge National Lab., TN (USA).

Xing, W., Shah, A. A. and Nair, P. B. (2015). Reduced dimensional Gaussian process emulators of parametrized partial differential equations based on Isomap, in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 471, p. 20140697, The Royal Society.

Xiong, Y., Chen, W., Apley, D. and Ding, X. (2007). A non-stationary covariance-based kriging method for metamodelling in engineering design, *International Journal for Numerical Methods in Engineering*, **71 (6)**: 733–756.

Zhang, R., Lin, C. D. and Ranjan, P. (2016). Local Gaussian process model for large-scale dynamic computer experiments, *arXiv preprint arXiv:1611.09488*.

Zidek, J. V., Shaddick, G., White, R., Meloche, J. and Chatfield, C. (2005). Using a probabilistic model (pCNEM) to estimate personal exposure to air pollution, *Environmetrics*, **16 (5)**: 481–493.