Gaussian Process Emulators in coastal wave modelling

Sajni Malde

Submitted for the degree of Master of Philosophy

School of Mathematics and Statistics

November 2018

Supervisor: Prof. Jeremy Oakley

The University of Sheffield

To HR Wallingford

Acknowledgements

I would like to express my sincere gratitude to my supervisor at The Univeristy of Sheffield: Professor Jeremy Oakley and my industrial supervisors at HR Wallingford: Ben Gouldby, David Wyncoll, and Ye Liu, for providing me with an excellent opportunity to work on my thesis as part of the knowledge transfer partnership (KTP) program. Their technical knowledge and support have been essential for me to complete this thesis.

In particular, I would like to thank Jeremy for sharing his expertise on GPEs in a wider context and for being supportive throughout the project. I would also like to thank David and Ye for their invaluable advice and support related to programming and statistical background knowledge.

Importantly, I would like to thank Innovate UK and HR Wallingford for providing the financial support for this project.

I would also like to thank Nigel Tozer and Dirk Diederen for sharing their technical knowledge of coastal modelling, the SWAN simulator and for various other technical discussions that have greatly impacted my thesis.

Finally, I would like to show my appreciation to my family and friends who have supported me throughout my studies without which this would not have been possible.

Abstract

A majority of the coastal wave modelling analysis require using historical data from physical observations or from computer simulations. Such simulators are often computationally expensive (takes long for a single evaluation run) and therefore it is normally a bottleneck in the analysis. Meta models are increasingly used as surrogates of the complex simulators to improve the efficiency of the bottleneck step. The performance of the meta model is vital when selecting the model as this would greatly influence the conclusions that are drawn from the analysis.

In this thesis we apply the Gaussian Process Emulator as a meta model of a wave transformation simulator, SWAN. The GPE is advantageous compared to other meta models as the predictions from the GPE are in the form of a distribution (mean and variance) and predicting at an event used to train the GPE returns perfect predictions with no uncertainty. Univariate and multivariate approaches of the GPE are presented and compared in case studies. In addition simple diagnostics to validate the GPE are discussed.

Look–up table (LUT) approach is a commonly used traditional meta model in coastal modelling. This is based on multidimensional linear interpolation of points on a regular grid. A case study shows the performance improvement that can be gained by using GPE over this traditional LUT approach. The GPE needs less than 2% of the simulator runs required for the LUT to obtain a similar accuracy.

When introducing the multivariate GPE we identify two types of multiple outputs. We present a principal component GPE (PC-GPE) and a separable GPE for highly correlated and high dimensional output. These methods are compared to fitting multiple univariate GPE's. In terms of accuracy the multiple univariate GPE outperformed the other methods however the PC-GPE tends to be more efficient with only a small compromise on accuracy. For low dimensional output that is weekly correlated we present the linear model of coregionalisation (LMC) GPE which is a more flexible technique than the separable GPE. We compared this with the separable GPE and to fitting multiple univariate GPEs. The LMC GPE gave similar results as the multiple univariate GPE, but it is unstable and took a significant amount of time to fit.

Finally, we describe three approaches of selecting a design (simulator runs used to train the GPE). We aim to select a design that will maximise the information we can get from the simulator in order to inform the GPE given the limited simulator runs.

The aim of this thesis is to present the GPE methodology in a concise manner with running examples throughout. The novelty here is to show the application of GPEs to coastal wave modelling in order to help alleviate the computational burden and improve accuracy when using meta-models to avoid the bottleneck in the analysis.

Contents

List of Tables **xi**

List of Figures **xii**

1 Introduction **1**

1.1 Motivation 1

1.1.1 Coastal modelling 1

1.1.2 Simulators and “events” 2

1.1.3 Meta models and emulators 3

1.1.4 The SWAN model 4

1.1.5 Design selection 4

1.1.6 A typical GPE application 5

1.2 Knowledge Transfer Partnership 6

1.3 Software 7

1.4 Overview of Thesis 7

2 The simulator: SWAN **8**

2.1 The SWAN model 8

2.2 Case studies 11

2.2.1 Case study 1: Farasin Islands, Red Sea 11

2.2.2 Case study 2: Greater Wash, UK 13

2.2.3 Case study 3: Dataset 15

3 GPE for univariate output 17

3.1 Introduction 17

3.2 Use of Gaussian process emulators 17

3.3 Notation and terminology 19

3.3.1 Toy example: one input, one output simulator 20

3.4 The formulation of a GPE 22

3.4.1 Partitioning property of multivariate normal distributions . . 22

3.4.2 Application of partition property to GPEs 23

3.4.3 Prior beliefs of $\eta(\boldsymbol{x})$ 23

3.4.4 Deriving the posterior distribution: $\eta(\boldsymbol{x})|\boldsymbol{y}, \boldsymbol{\delta}$ 26

3.5 Diagnostics for validating GPEs 32

3.6 Case Study - Greater Wash, UK 39

3.6.1 Results 42

3.6.2 Discussion of results 51

3.7 Conclusion 52

Appendices 54

3.A Perfect prediction at training event 54

4 GPEs for multivariate output 57

4.1 Introduction 57

4.2 Multivariate output and common emulation techniques 58

4.3 Field type outputs 61

4.3.1 Toy example: one input, two field output simulator 62

4.3.2 Principal Components GPE (PC-GPE) 63

4.3.3 Separable GPE 71

4.3.4 Case Study - Farasin Islands, Red Sea 78

4.4 Multiple-type output 88

4.4.1 Toy example: one input, two multiple-type outputs simulator . 89

4.4.2 LMC GPE 91

4.4.3 Case Study - Greater Wash, UK 100

4.5 Conclusion 108

5 Design Selection 110

5.1 Introduction 110

5.2 Common design selection methods 111

5.3 Criterion 1 114

5.3.1 Maximum dissimilarity algorithm (MDA) 115

5.4 Criterion 2 119

5.4.1 Weighted maximum dissimilarity algorithm (WMDA) 120

5.5 Criterion 3 122

5.5.1 Pivoted Cholesky decomposition 123

5.6 Case Study 125

5.6.1 Results and discussion 128

5.7 Conclusion 133

6 Conclusion 135

6.1 Other applications of GPE at HR Wallingford 138

6.1.1 Bayonet GPE 138

6.1.2 SWAN calibration using GPE 138

6.1.3 Applying GPE to other simulators 139

6.1.4 Wave forecasting 139

Bibliography 141

List of Tables

4.1 Correlation length parameters for the LMC GPE at n = 500 102

4.2 Correlation length parameters for the Separable and Independent GPE at n = 500 102

List of Figures

1.1 Flowchart for using a GPE in a typical application 5

2.1 Location map of Farasin Islands in the Red Sea showing the wave prediction point for case study 1 marked by the green triangle 12

2.2 Location map of Greater Wash, UK showing the wave prediction point for case study 2. The depth of this area is relatively shallow and hence of interest for wind farms. 14

3.1 Univariate toy simulator - the black points indicate the training runs and the red crosses are the events which we wish to predict 21

3.2 Univariate toy GPE - the points indicate the training runs, the red crosses are the events which we predict at, the red dashed line shows the GPE predictions and the shaded region represents the uncertainty around the predictions 32

3.3 Design points selected using the Regular grid (black asterisks – lower triangular matrix) and the MDA approach (black circles – upper triangular matrix). The lower triangular matrix shows poor coverage in each dimension compared to the upper triangular matrix. 43

3.4 NRMSE for Wave height (measured in metres), period (seconds)and direction (degrees) using LUT (dashed line – with 16000 design events) and GPE (solid line – using between 10 to 1000 design events). The GPE outperforms the LUT using only 200 design points. Further gains in accuracy with the GPE are possible with more design points. 44

3.5 Comparison of approximations using LUT (upper panel) and GPE-MDA (lower panel) with true SWAN output. The LUT has a wider spread of points compared to the GPE method which indicates worse performance. The off-diagonal cluster of points for the wave direction plots are due to the periodic nature of this variable. 45

3.6 Time series plot of Significant wave height, wave period and wave direction showing the predictions from LUT (dotted blue line) and GPE (dashed red line). The shaded region represents the 95% credible interval from the GPE. The true SWAN output is presented with a solid black line (or dots in the bottom panel) 46

3.7 Validation plot of the GPE fit for Wave height ($Hsig$) 48

3.8 Validation plot of the GPE fit for Wave period (Tm_{02}) 49

4.1 Multivariate (field-type output) toy simulator. The black points highlight the training runs, the red crosses are the events which we wish to predict at and the red points are true simulator output for these events. 63

4.2 Multivariate (field-type output) toy simulator plotted with the principal components (component 1 and component 2) of the data. . . . 64

4.3 Multivariate (field-type output) simulator output plotted on transformed axis - the principal components. 65

4.4 Multivariate (field-type output) toy GPE. The black points highlight the training runs, the red crosses are the events we predicted using the GPE and the red points are the predicted simulator output. . . . 70

4.5 Design selected as training inputs using the MDA method. The black dots represent the design and the grey dots in the background represent all the events. 79

4.6 Correlation between the simulator outputs 1:189 80

4.7 Proportion of variance explained by each additional principal component 82

4.8 Effect of increasing PC on NRMSE. The red line represents the NRMSE generated from the IND method. The blue dotted line shows the NRMSE generated by fitting the first 6 components. 83

4.9 Predicted and simulated values for one output variable using the SEP GPE 85

4.10 NRMSE for all the output variables and methods for the field-type output. 86

4.11 Coverage values for all the output variables and methods for the field-type output. 87

4.12 Multivariate (multiple-type output) toy simulator (1 input, 2 output) 90

4.13 Multivariate (multiple-type output) toy simulator (1 input, 2 output) 100

4.14 Correlation between the simulator outputs for the multiple-type output. 101

4.15 Comparison of time taken to fit GPEs to all the output variables for each method. 103

4.16 NRMSE for all the output variables and methods for the multiple-type outputs. 105

4.17 Coverage values for all the output variables and methods for the multiple-type outputs. The dotted line represents 95% coverage. . . . 106

4.18 Comparison of predicted and simulated values for all output variables using all the methods. 107

5.1 Explaining the MDA diagrammatically - Having selected points 1 and 3, the next most dissimilar point to be selected is point 2. 117

5.2 Explaining the WMDA diagrammatically - Having selected points 1 and 3, the next most dissimilar point to be selected using weighted distances is point 4. 121

5.3 Weight assigned to each event based on the events input wave height and wave direction values. 127

5.4 Designs selected using a random selection (RG1 and RG2) 129

5.5 Designs selected using MDA (upper triangle) and WMDA (lower triangle) designs 130

5.6 Comparison of NRMSE of GPE's fitted to different designs for varying sizes of designs 131

5.7 Comparison of weighted RMSE of GPE's fitted to different designs for varying sizes of designs 132

Chapter 1

Introduction

1.1 Motivation

1.1.1 Coastal modelling

Coastal modelling applications typically include the analysis of ocean waves to design and manage offshore and coastal structures (e.g. harbours, bays, piers, jetties, dykes, wind farms, etc.). Following a detailed analysis, engineers and contractors may need to adapt the design of a structure to wave loads on site, they may want the structure to be able to withstand extreme wave conditions, or they may also want to perform fatigue analysis of the structure (analysing the weakening effect of constant load on a structure). The analysis may also be used to study the operability of the structure, how or when it would be safe to construct the structure or access it for maintenance, or to analyse the safety of a structure that may, for example, be prone to flooding. Other applications of coastal modelling include performing risk analysis of a certain event happening (e.g. coastal flood risk analysis), prediction of waves at a particular location and defining the wave climate (a long term distribution of sea states) which maybe be useful for applications that require, for example, the analysis of sediment transport.

These applications typically require data related to wave properties at a particular site near or on the shore. Often buoy measurements, if available, may be many kilometers away from the site of interest. Moreover, it is seldom possible to get a long enough time series from these buoys for an extreme value analysis, so modelled data is often used instead. Additionally, the data that is readily available from regional or global models (such as the WAM model (The WAMDI Group, 1988) or the WAVEWATCH3 model (Tolman, 2009)) are based on coarse grids (between 5-20km) and generously account for deep water processes. However for applications such as the ones mentioned earlier, it is necessary for the data to account for shallow water processes that may be specific to the path travelled by a wave. Factors such as the slope and geometry of the sea bed and local wind play a significant role in how waves transform as they approach the shore compared to how waves behave in deep waters. Moreover, as per The British Standards Institution (2013), it is necessary for the deeper water waves to be "transformed to the point of interest, taking account of shallow water effects which act at the location such as bed friction, refraction, shoaling, breaking, reflection and diffraction." Often computer models (henceforth referred to as simulators) are used to model such wave transformation processes.

1.1.2 Simulators and "events"

Simulators, in general, are a set of computer codes used to compute a set of output variable(s) based on a given set of input variables. Very simply, we can think of a simulator as a function, $\eta(.)$ where we have some inputs $\boldsymbol{x}$ and the function returns some output y such that $y = \eta(\boldsymbol{x})$. We suppose that the relationship between the inputs and outputs is complex enough that a closed form expression is not available.

We define a *simulator run* as a single evaluation of $\eta(.)$ at a certain choice of inputs, $\boldsymbol{x}$. An *event* is defined as a vector of inputs, $\boldsymbol{x}$, that is required for a single simulator run.

Simulators can be computationally expensive to run if the underlying physical processes represented by the model are complex. For example, coastal process simulations, flood risk analysis and flood forecasting can all require substantial computational resources. When uncertainty, and the associated sensitivity analysis, also needs to be evaluated, the computing demand can increase by orders of magnitude. It is therefore common practice to apply statistical approximation techniques to predict the outcome of computationally intensive models, without necessarily running the simulator for every case of interest.

1.1.3 Meta models and emulators

Meta-models (defined by Thode (1988) as a "model of the model") are used to approximate complex simulators by constructing lower-cost statistical models that are used as a surrogate to the complex simulators. Modern meta-modelling approaches include: Polynomial regression (Jin et al., 2001), simple look up table approaches, (Artificial) Neural Networks (Kalra et al., 2005; Maier and Dandy, 2000), Multivariate Adaptive Regression Splines (Friedman, 1991), Radial basis functions (Buhmann, 2003) and Gaussian Process Emulators (GPE). A comparison of several combinations of these methods in different applications can be found in Jin et al. (2001), Kalra et al. (2005), Ajeesh and Deka (2015), Clarke et al. (2005) and Storlie et al. (2009).

These approaches typically draw upon a smaller subset of representative events defined by some selection criteria. The simulator is used to derive the output variables for this representative subset. A meta-model is then fitted to these events and the corresponding outputs. It is then used as an approximation tool to predict the output for any new event. The accuracy of the meta-model will depend on the size and design of the representative subset and the interpolation technique used.

This research focuses on using a Gaussian Process Emulaotr (GPE) - which is a meta-model that takes a Bayesian approach in modelling. Compared with other meta-modelling techniques, the GPE has two significant advantages, (Santner et al., 2013): the Bayesian approach means that the prediction is in the form of a distribution, thus giving point wise predictions and credible interval bounds around this prediction as an indication of how uncertain the GPE prediction may be at a particular point. The second advantage is that the GPE interpolates the available simulator runs exactly: predicting at an event that was used to train the GPE results in a perfect prediction with zero uncertainty.

1.1.4 The SWAN model

In this thesis, we use the SWAN wave transformation model (**S**imulating **WA**ves **N**earshore), as the computationally expensive simulator that we would like to approximate. The SWAN model is used to derive information about the wave properties near the shore (or in shallow waters) given information about the wave properties offshore. A single run of the SWAN model can take between a few minutes and a few hours to evaluate depending on the complexities of the model set up. Therefore running the model for 1000 events can become computationally expensive and time- consuming. The objective of the GPE implementation is to reduce the computational time and resources used to manageable levels, without significantly compromising on the accuracy of the predictions.

It is worth mentioning that the meta-models mentioned here can potentially be used in a Bayesian setting, however there is no implementation in literature yet.

1.1.5 Design selection

Suppose $\mathcal{X} \in \mathbb{R}^p$ is a collection of events at which we want to run the simulator. Usually, the input domain can be represented in the form $[0, 1]^p$ and the values in each dimension can be selected independently. Here we are choosing training events from a predetermined set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ generated from a multivariate probability distribution of interest. This is atypical and we will be investigated further in Chapter 4.

1.1.6 A typical GPE application

Figure 1.1 shows a flowchart of a typical application where a GPE would be used. We often have a large number of events which we cannot afford to run the simulator for. So instead, we choose (filter) a representative subset of the events to run through the simulator (SWAN). We run the simulator for the selected events. Then, we train the GPE using the selected events and the corresponding simulator output. We then use the trained GPE fit to predict the output at the remaining events.

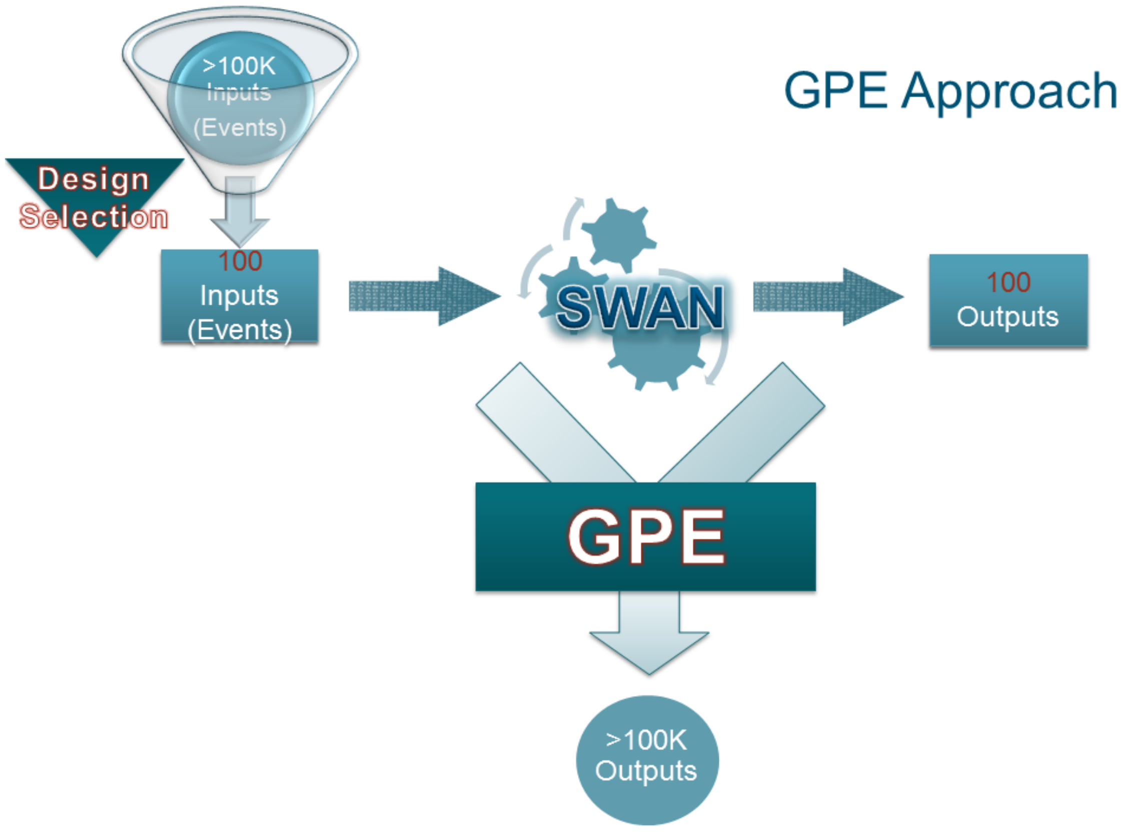

Figure 1.1: Flowchart for using a GPE in a typical application

1.2 Knowledge Transfer Partnership

This thesis was part of a knowledge transfer partnership (KTP) program. This is a part-government funded program that enables universities or academic institutions that are knowledgeable on latest methodologies and innovation, to collaborate with industries where these methodologies could be applied to increase efficiency, accuracy or improve methods. This is done by a suitable associate who will lead a strategic business project.

In the case of this project, the KTP was a collaboration between the University of Sheffield and HR Wallingford. HR Wallingford is an independent research organisation and consultancy in civil engineering, environmental hydraulics and management of water and water resources.

As part of their analysis, HR Wallingford have to run several numerical models for thousands of events which can be computationally expensive and time consuming. Before the start of this project, HR Wallingford used simple meta-modelling techniques to help them overcome some of the computational burden. The objective of this project was to research and implement advanced meta-modelling approaches using GPE to enable them to increase their efficiency when completing their projects and to reduce the computational burden. Some of the these techniques and comparisons have been recorded in this thesis.

The novelty in this research is in the application of GPE to a particular simulator used at HR Wallingford. HR Wallingford had provided us with real life examples used in the industry where improved meta modelling approaches are beneficial in terms of increasing accuracy and improving efficiency. As part of the research we were able to compare their traditional approach with the GPE. We presented the results and published a paper for the FloodRisk 2016 conference, (Malde et al., 2016). Moreover we have submitted a paper to the Journal of Coastal Engineering highlighting some of the results (Malde et al., 2018).

1.3 Software

In collaboration with other colleagues in this project, the methods discussed have been programmed into an R package that is available on Github (https://github.com/OakleyJ/MUCM.git). We have called this package MUCM (Managing Uncertainty in Computer Models). The package includes the univariate and multivariate GPE methodologies discussed here and it also contains a function to select the designs using the methods discussed. We have ensured that the package is robust, optimised and valid.

1.4 Overview of Thesis

We begin in Chapter 2 by briefly introducing the SWAN model, and detailing the context of the case studies that we will use in the later chapters. Chapter 3 begins with an introduction to univariate (one output variable) GPE, followed by a discussion of the diagnostics for validating a GPE. We end the chapter with a case study presenting the performance improvement gained by using a GPE over a traditional approach.

In practice simulators often have more than one output variable (multivariate). It is common for users to model each of these multiple outputs independently using a univariate GPE. In Chapter 4 we present three approaches to handle multiple outputs, and compare the performances amongst them. In Chapter 5 we introduce some basic design selection methods based on three criteria and compare the use of these methods in a case study. The thesis ends with a conclusion in Chapter 6 of the main topics discussed within the thesis along with other related work carried out as spin offs from this program.

Chapter 2

The simulator: SWAN

The SWAN (Simulating WAves Nearshore) model is a widely used simulator in applications of coastal engineering that require coastal wave modelling. Booij et al. (1999) compare and contrast SWAN to other wave models and conclude that the results of the SWAN simulator agree well with the analytical solutions, laboratory observations and generalised field observations. In this chapter, we briefly introduce the SWAN model and describe the inputs and outputs that are most commonly used. Finally, we outline the details of the three case studies that are used in the chapters. The case studies that we present in the thesis represent typical applications of coastal modelling with SWAN. Although SWAN is faster than other types of wave models, it is reasonably demanding and is often the bottleneck in the complete analysis. In the remaining chapters we describe how we use a Gaussian process emulator to approximate the SWAN model in order to mitigate computational expenses and time taken to run the model.

2.1 The SWAN model

SWAN, developed by the Delft University of Technology, is a third generation spectral wave model for obtaining realistic estimates of wave parameters in coastal areas, lakes and estuaries for given wind, depth and current conditions (Booij et al., 2004). The model computes the wave transformation from a point offshore to a near shore point taking into consideration different tidal, wind and offshore wave conditions.

Basic linear wave theory states that as a wave approaches shallow water the speed of the wave decreases, the wave length (the distance between successive crests of a wave) decreases and the wave height (the vertical distance between a trough and a crest) increases. However, the wave period, which is the time taken for two crests of a wave to pass a fixed point, does not change. These differences between deep water and shallow water wave transformations justify the use of such a simulator over global models that only represent deep water processes. The SWAN model also represents the following processes:

- Refraction - the bending of the waves due to varying water depths or currents. Waves in shallow water move slower compared to waves in deeper water, thus as waves approach the shore, they bend; (Bascom, 1964)
- Shoaling – the change in wave height as waves enter a shallow water region; (Komar, 1976)
- Reflection - the change in direction of a wave as it encounters an obstacle;
- Diffraction - the propagation of a wave around an obstacle.

However it is not recommended to use the model in areas where variations in wave height are large (within a horizontal scale of a few wavelengths), as the wave field computed by SWAN will generally not be accurate in the immediate vicinity of obstacles, for example for predicting waves in ports and harbours.

The SWAN model is ideally suited to the transformation of wave energy spectra in relatively large areas as is often the case with such applications. This is particularly true where the features of the seabed (such as offshore banks and reefs) cause depth induced wave breaking and non-linear wave-wave interactions. The model can also include wave generation by wind within the model area and thus is useful for areas where waves can be locally generated.

The input to SWAN consists of variables that describe the properties of the wave conditions along the offshore boundary, the seabed bathymetry and the wind forcing. These may include: significant wave height (average wave height from trough to crest of the highest third of the waves), measured in meters; peak wave period (T_p), measured in seconds; wave and wind direction, given in degrees; wind speed, measured in meters per second and water level, measured in meters.

The output of SWAN comprises transformed wave conditions near shore having taken account of shallow water processes. The outputs that we are typically interested in are near shore wave height, wave period, including mean wave period (T_{m01}, T_{m02}, T_{m-10}) and wave direction.

Depending on the complexities of the study area and the associated wave boundary conditions, which often govern the model grid extent and resolution, each event (or time step) can take from several minutes to a few hours to run on a regular computer. Thus to evaluate SWAN for 1000's of events (which would be necessary when for example we need to hindcast several years of conditions, or in a multivariate extreme value analysis) can be time consuming. The objective of the GPE implementation is to reduce the computational time and resources used to tolerable levels, without significantly compromising on the accuracy of the predictions.

In the context of the examples and case studies that we discuss in this thesis, we make an assumption that the full set of events which we want to run the model for are pre-determined. These events are often pre-determined because they result from Monte Carlo sampling which is carried out to generate large samples of synthetic events from historical events. This is done in order to model extreme sea conditions using multivariate extreme value methods in order to account for global warming, for example, (Gouldby et al., 2014). This is atypical to other GPE applications and the design selection methods they use.

2.2 Case studies

In this section we present three case studies that we are going to use throughout the thesis to demonstrate the application of GPE. The case studies we present are past projects that HR Wallingford worked on, and we believe that these case studies in particular are sufficiently complex to provide realistic test cases to assess the robustness and accuracy of the GPE. Here, we detail the scope of the study and where the data comes from.

2.2.1 Case study 1: Farasin Islands, Red Sea

Saudi Aramco, a firm in Saudi Arabia, intends to construct a 400,000 barrels crude per day grassroots refinery and terminal facilities within the economic centre of Jazan, Red Sea. HR Wallingford was commissioned to undertake a range of specialist hydraulic studies necessary to support the design and development of the works including: metocean study, hydraulic design for seawater intake, berths hydraulic studies, scour protection and sediment dispersion studies. This case study focuses on the metocean study which entails the study of the meteorological (weather) conditions and physical processes within the sea.

The Farasin Islands are a large group of coral islands located in the Red Sea approximately 40 km offshore from the Port City of Jazan, in the south west of Saudi Arabia (see Figure 2.1). Within this area, due to the sheltering effect of the islands, wave conditions are typically locally generated. This means that some of the islands are exposed to waves from offshore and there are large variations in fetch lengths and bathymetry. This makes it a challenging area to predict wave conditions using simple desk-based techniques, and therefore require the use of a simulator such

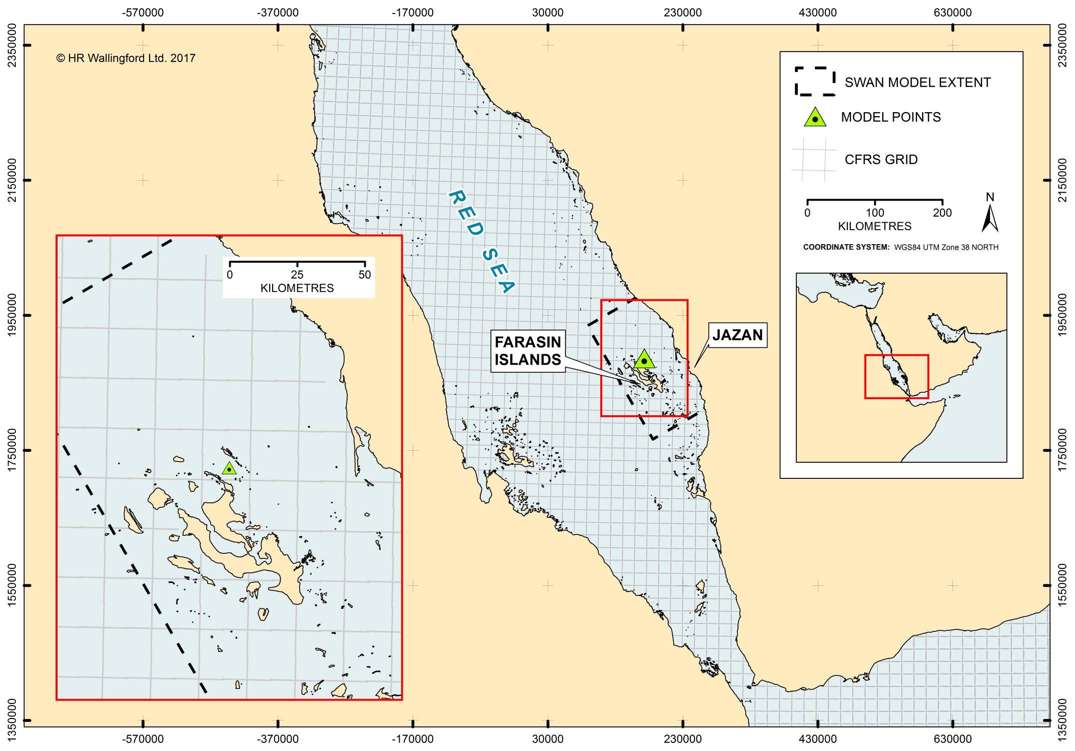

Figure 2.1: Location map of Farasin Islands in the Red Sea showing the wave prediction point for case study 1 marked by the green triangle

as SWAN. The complexity of the geometry and bathymetry makes this area an ideal site to test the performance of a GPE and various design selection methods. Note, in this case study, we only focus on the part of the project that requires the use of the SWAN model, i.e. only the wave transformation from offshore, to a point within the area of interest.

Offshore wind and wave conditions were derived from CFSR (Climate Forecast System Reanalysis - a high resolution coupled atmosphere-ocean-land surface-sea ice reanalysis, (National Center for Atmospheric Research Staff (Eds), 2016)) data. These were used as input conditions for the SWAN model to derive the wave climate at points around the Farasin Islands.

For this case study, the input SWAN boundary conditions were a long term time series of significant wave height (meters), peak wave period (seconds), wind speed (meters per second), wave direction (degrees) and wind direction (degrees) given by the regional wind and Wavewatch III wave model (National Center for Atmospheric Research Staff (Eds), 2016; Tolman, 2009) and water lever (meters). The tidal range in the Red Sea is relatively small, so the tide range was artificially enhanced to provide a more severe test of the methods. The SWAN model predicts the corresponding wave conditions of significant wave height, mean wave periods (e.g. T_{m02}, T_{m-10}), peak wave period (T_p) and mean wave direction for all events across the model domain as shown in Figure 2.1.

The full set of events we wish to hindcast are a time series of offshore wave and wind conditions at 3 hourly time steps from 1st December 1983 to 31st December 2009 which in total leads to approximately 70,000 time steps or events. A validation dataset was generated comprising SWAN model simulations for a subset of these events, 2 years of continuous data containing approximately 5600 events.

2.2.2 Case study 2: Greater Wash, UK

The Greater Wash is a region on the east coast of England, UK, where the seabed is characterised by a series of shallow banks. Due to the shallow water, open aspect, relative stability of the banks, short distance from the coast and good wind resource, this area is the site of several offshore wind farm arrays, including sites at Race Bank and Docking shoal (as shown in Figure 2.2). For the design of such structures, and planning construction and maintenance work, knowledge of the long term wave climate is essential. Wave conditions at the site are a combination of those generated locally by winds from the South and the West, as well as those generated in the North Sea that propagate towards the site from the North, East and South East. Whilst there is measured data at the sites of interest suitable for model calibration, in the absence of long term measured data spanning several decades, there was a requirement for a hindcast analysis. To provide a robust hindcast it was necessary

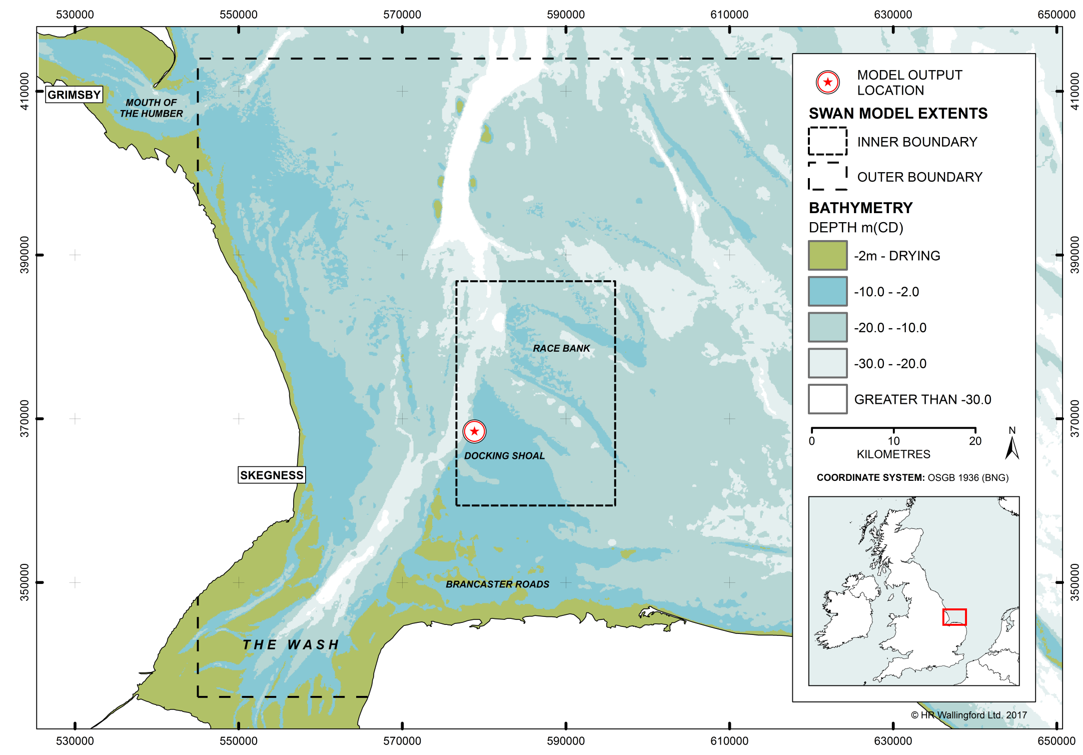

Figure 2.2: Location map of Greater Wash, UK showing the wave prediction point for case study 2. The depth of this area is relatively shallow and hence of interest for wind farms.

to accurately resolve the extensive and complex bathymetry features in the area at a relatively high spatial resolution. This requirement places computational constraints on decadal simulations and thus meta-modelling approaches lend themselves to these situations.

Offshore wave conditions are applied to the northern and eastern boundaries of the SWAN model and a constant wind field applied over the area. The wind and offshore wave conditions were from the UK Met Office European wave model. This model provides 3 hourly wind (20m elevation) and wave records on a 25 km spatial grid. The model is operated primarily for forecasting purposes, taking boundary wave data from a global wave model and surface wind data from a weather model also run for forecasting purposes. Wind data are also provided at every model point.

The six input variables for SWAN in this case study are significant wave height (meters), peak wave period (seconds), wind speed (meters per second), water level (meters), wave direction (degrees) and wind direction (degrees). SWAN calculates the corresponding transformed wave conditions at the point of interest (marked by the star in Figure 2.2) which are significant wave height, mean period and mean wave direction at all events across the model domain, taking account of the relevant shallow water processes.

The full set of events at which we want to know the output from the SWAN model are a time series of offshore wave conditions at 3 hourly time steps starting from January 1986 to 2009, approximately 78,000 time steps. Each time-step was considered a separate event and represented as a steady-state simulation. A dataset containing 5430 events from 1987 and 2009 was set aside as a validation dataset which is used for assessing the performance of the model. The output time series provides important information on the expected sea conditions, e.g. the directional distribution and the seasonality, for further steps in the hindcast analysis.

2.2.3 Case study 3: Dataset

For this case study, we cannot reveal the background of the study or the location of the study due to client confidentiality agreements. However we explain the dataset that we were allowed to use as a test case.

We use data from the SWAN model with 7 input variables at an offshore location namely; significant wave height, period, direction of wave, spread of wave, water level, wind speed and wind direction. The output variable considered is the significant wave height at 189 near shore points, which are separated by approximately a mile in distance from each other. SWAN has been run for 500 events which were selected using a design selection technique (Maximum Dissimilarity Algorithm - MDA) from a much larger set of events. We use 250 events as training data and a further 250 events as validation data. We do not have access to the full set of events from which the MDA events were originally chosen from.

Chapter 3

GPE for univariate output

3.1 Introduction

A Gaussian Process (GP) is defined as an infinite collection of random variables such that any finite subset has a multivariate normal distribution, (Rasmussen, 2006). In this chapter, we first describe the history and applications of Gaussian process emulators (GPEs). Then we introduce the notation and terminologies used and show a detailed derivation of the univariate GPE model. We go through the derivation of the model with a simple toy example. We review methods to diagnose a GPE model and assess whether it can be used for prediction. We conclude the Chapter by illustrating the benefits of using a GPE over a traditional approach using one of the case studies from Chapter 2.

3.2 Use of Gaussian process emulators

GPs were first used as emulators for computer models by Sacks et al. (1989). They base the model from a kriging (Matheron, 1963) perspective (spatial statistics). Currin et al. (1991) presented a Bayesian version of the model which he thought was a more natural way of interpreting the model for the common application areas.

GPEs are commonly used for solving optimisation problems, model calibration, and performing uncertainty and sensitivity analysis. Optimisation problems typically involve finding out what input values to use to get a desired output value. Jones et al. (1998) used GPEs to replace an expensive black box function to efficiently solve a global optimisation problem. Wood et al. (2015) compare the use of GPEs with genetic algorithms for optimising the energy use of buildings. They find that the GPE approach leads to more stable output, even with much fewer simulation runs.

Simulators may need to be calibrated using observed data before using them to make estimations of real world scenarios. Calibration problems involve finding input values such that the output values match observed data. Kennedy and O'Hagan (2001) have used GPEs for model calibration using a Bayesian approach. Holden et al. (2010) present a more advanced use of the GPE for model calibration combined with approximate Bayesian computation. The GPE allows for investigating the uncertainty in climate sensitivity as a function of the input parameters to the GENIE-1 model.

GPEs are also used to speed up sensitivity analysis and uncertainty analysis. Sensitivity analysis seeks to investigate how simulator outputs respond to changes in simulator inputs, (Oakley, 2011). Uncertainty analysis is about quantifying uncertainty in model outputs induced by uncertainty in model inputs (O'Hagan, 2006). If the simulator is computationally expensive than these analysis are often skipped or not thoroughly done. GPEs allow for the simulator to be "replaced" by a GPE once it has been trained on some data from the simulator. This allows for more robust analysis. Fricker et al. (2011) present two approaches to carry out probabilistic uncertainty analysis and demonstrate this on practical examples. Moreover they compare this approach to alternative approaches used in practice and conclude that the suggested approach using GPEs gives a true representation of the uncertainty in the model. Becker et al. (2011 and 2012) suggest approaches to perform uncertainty and sensitivity analysis using a GPE at a much lower cost. Bounceur et al. (2015) also explores the use of GPEs with principal component analysis to perform global sensitivity analysis on a climate vegetation system.

A GPE fits in to all the above applications because there is often a simulator involved in the analysis that is computationally expensive. In most cases, the user will have limited computational power or a time constraint and thus will only be able to run a limited number of simulator runs. A GPE can be trained on the limited simulator runs and then be used to make predictions of the simulator output at untried events.

3.3 Notation and terminology

We first define the notation and terminology used. Let a deterministic simulator be represented by a function $\eta(.)$. By deterministic we mean that the simulator will always produce the same output for a given set of input variables (no randomness involved). We define a vector of all input variables required for a single simulator run as an *event*. Each event is denoted as a p-element vector of input variables $\boldsymbol{x} = (x_1, \ldots, x_p)$. We denote the output of the simulator as y, such that $y = \eta(\boldsymbol{x})$.

The requirement is to evaluate the simulator N times at $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, where N is a large number. Suppose there is however, for practical computational reasons or limited time, a restriction on the number of simulator runs that can be performed. These are restricted to n, where n is much smaller than N.

We carefully select n events as a representative subset of the full set of events, $\mathcal{X}$. We define the selected events as the *design* and denote it as $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \subseteq \mathcal{X}$. We run the simulator at the design events and the corresponding outputs are represented as $\boldsymbol{y}^T = (y_1 = \eta(\boldsymbol{x}_1), \ldots, y_n = \eta(\boldsymbol{x}_n))$.

We use these design events and corresponding simulator output to train the GPE. Then we use the GPE to predict the output for the $(N - n)$ non-design events, $\mathcal{X} \setminus X = (\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_N)$ where we did not run the simulator. We derive a probability distribution for $(\eta(\boldsymbol{x}_{n+1}), \ldots, \eta(\boldsymbol{x}_N))$ given the training runs, $\boldsymbol{y} = \eta(X)$. This distribution is derived by modelling $\eta(.)$ as a Gaussian Process (GP) and can be used to give predictions for the simulator output evaluated at non-design events $(\mathcal{X} \setminus X)$. The GPE can also give an uncertainty estimation about its predictions which serves as an indicator for how confident the GPE is in predicting at that input value. For simplicity in the mathematical detail below we use $\boldsymbol{x}$ (no subscript) to indicate an event that we would like to predict at.

In the Bayesian approach, we treat $\eta(.)$ as a random variable simply because its unknown. The simulator can give us the value of $\eta(\boldsymbol{x})$ exactly, but without running the simulator, we are uncertain about the value of $\eta(\boldsymbol{x})$. We choose to represent the uncertainty about $\eta(.)$ with a GP.

A GP distribution is fully specified by its mean and covariance functions. In this section, we illustrate how to use a GPE as a method to predict the output, $\boldsymbol{y}$, for new input values $\boldsymbol{x}$ given the training runs $(X, \boldsymbol{y})$. This Section is based heavily on Oakley (1999).

3.3.1 Toy example: one input, one output simulator

We describe a toy simulator with $p = 1$ input, and $q = 1$ output. We use this simulator as a running example as we illustrate the methodology of fitting a univariate GPE. We define the simulator function as $y = 0.2x^2 + 3\exp(-x)\cos(2\pi x)$ where $x \in [-1, 2]$, but we assume that the simulator is computationally expensive, and so we can only evaluate the function at a limited number of values. There are many ways to select a subset of points over the input space, however for this example we just pick 9 points that are equally spread over the input space. The training inputs and the corresponding training outputs, from the toy simulator function, are given below and we wish to predict the simulator at two input values: $(0.5, 1.75)$,

$$X = \begin{pmatrix} -1 \\ -0.85 \\ 0 \\ 0.25 \\ 0.4 \\ 0.75 \\ 1.2 \\ 1.5 \\ 2 \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} 8.35 \\ 4.27 \\ 3 \\ 0.01 \\ -1.59 \\ 0.11 \\ 0.57 \\ -0.22 \\ 1.21 \end{pmatrix}.$$

Figure 3.1 shows the output of the toy simulator plotted against the input values that range over the input space [-1,2]. The black points represent the training runs, and the red crosses are the events where we wish to predict the simulator output.

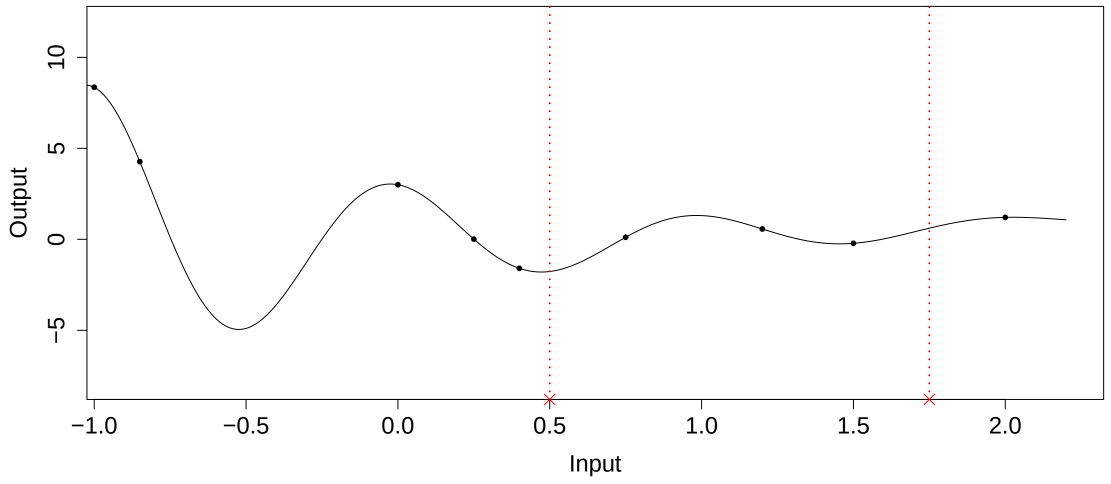

Figure 3.1: Univariate toy simulator - the black points indicate the training runs and the red crosses are the events which we wish to predict.

3.4 The formulation of a GPE

3.4.1 Partitioning property of multivariate normal distributions

A useful property of multivariate normal (MVN) distributions which carries over to GPEs is its partitioning property. Krzanowski (1988) goes through the proof in his book, we choose only to present the result. Suppose

$$\boldsymbol{\mathcal{Z}} \sim MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\mathcal{Z}}$ is an $N \times \mathit{1}$ vector and has a multivariate normal distribution with mean $\boldsymbol{\mu}$, and variance $\boldsymbol{\Sigma}$. Now partition $\boldsymbol{\mathcal{Z}}$ into $\begin{bmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{bmatrix}$ where $\boldsymbol{y_1}$ is an $n \times \mathit{1}$ vector and $\boldsymbol{y_2}$ is an $(N-n) \times \mathit{1}$ vector and $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ are partitioned accordingly into $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{bmatrix}$, and $\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$ where $E(\boldsymbol{y_i}) = \boldsymbol{\mu_i}$ and $cov(\boldsymbol{y_i}, \boldsymbol{y_j}) = \Sigma_{ij}$. Then $\boldsymbol{f}$ is given by:

$$\boldsymbol{y_2}|\boldsymbol{y_1} = \boldsymbol{f} \sim MVN(\boldsymbol{\mu_{2|1}}, \boldsymbol{\Sigma_{2|1}}),$$

where

$$\boldsymbol{\mu_{2|1}} = \boldsymbol{\mu_2} + \Sigma_{21}\Sigma_{11}^{-1}(\boldsymbol{f} - \boldsymbol{\mu_1}), \tag{3.1}$$

$$\boldsymbol{\Sigma_{2|1}} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}. \tag{3.2}$$

3.4.2 Application of partition property to GPEs

Suppose we model $\eta(.)$ as a GP where,

$$\begin{bmatrix} (\eta(\boldsymbol{x}_1), ..., \eta(\boldsymbol{x}_n))^T \\ (\eta(\boldsymbol{x}_{n+1}), \ldots, \eta(\boldsymbol{x}_N))^T \end{bmatrix} \sim MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.3}$$

Using the partition property, we know $(\eta(\boldsymbol{x}_{n+1}), \ldots, \eta(\boldsymbol{x}_N))^T \mid (\eta(\boldsymbol{x}_1), ..., \eta(\boldsymbol{x}_n))$ has a multivariate normal distribution, and thus we can evaluate the conditional mean and variance for the distribution following the procedure discussed in Subsection 3.4.1. The prediction of the simulator output at non-design events is given by the conditional mean. The conditional variance describes the uncertainty around this prediction. Hence, having chosen to model $\eta(.)$ as a GP, we need to decide how to obtain $E[\eta(\boldsymbol{x})]$ and $cov(\eta(\boldsymbol{x}_i), \eta(\boldsymbol{x}_j))$.

For the toy example, we can rewrite Equation 3.3 as

$$\begin{bmatrix} (\eta(-1), \eta(-0.85), \ldots, \eta(2))^T \\ (\eta(0.5), \quad \eta(1.75))^T \end{bmatrix} \sim MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.4}$$

3.4.3 Prior beliefs of $\eta(\boldsymbol{x})$

Prior mean of $\eta(\boldsymbol{x})$

We can describe our prior expectation of $\eta(\boldsymbol{x})$ generally as:

$$E[\eta(\boldsymbol{x})|\boldsymbol{\beta}] = \boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{\beta}, \tag{3.5}$$

where the vector $\boldsymbol{h}(.)$ consists of m known regression functions of $\boldsymbol{x}$ incorporating our beliefs about $\eta(.)$, and the vector $\boldsymbol{\beta}$ consists of m unknown coefficients. We expect that using this general form of prior mean will always make it possible to describe our beliefs and choose an appropriate form of $\boldsymbol{h}(.)$. For instance, if we believe $\eta(.)$ has linear trends in its inputs $(m = p+1)$, then we can state $\boldsymbol{h}(X)^T = (\mathbf{1}, X^T) = \big(\mathbf{1}, (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^T\big)$ and $\boldsymbol{\beta} = (\beta_1, ..., \beta_m)^T$.

For our toy example, we assume that the simulator has a linear mean function and as discussed later we assume that β has a weak prior. Therefore we write our prior expectation as:

$$E[\eta(\boldsymbol{x})|\boldsymbol{\beta}] = \begin{pmatrix} 1 & -1 \\ 1 & -0.85 \\ 1 & 0 \\ 1 & 0.25 \\ 1 & 0.4 \\ 1 & 0.75 \\ 1 & 1.2 \\ 1 & 1.5 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}.$$

Prior covariance of $\boldsymbol{\eta(x)}$

We consider how we expect the true function to deviate from this prior expectation. We expect there to be a high correlation between $\eta(\boldsymbol{x})$ and $\eta(\boldsymbol{x'})$ if $\boldsymbol{x}$ and $\boldsymbol{x'}$ are sufficiently close, and we expect this correlation to decrease with an increase in distance between $\boldsymbol{x}$ and $\boldsymbol{x'}$. Formally we write the covariance between $\eta(\boldsymbol{x})$ and $\eta(\boldsymbol{x'})$ as

$$Cov(\eta(\boldsymbol{x}), \eta(\boldsymbol{x'})|\sigma^2) = \sigma^2 c(\boldsymbol{x}, \boldsymbol{x'}), \tag{3.6}$$

for some correlation function, $c(\boldsymbol{x}, \boldsymbol{x'})$ which will respect our expectations above. The parameter σ^2 is an unknown variance parameter.

Numerous kinds of correlation functions have been used in the past, but in this Chapter we present two correlation functions. A common and convenient (tractable) correlation function adapted from Sacks et al. (1989) is the Gaussian correlation function. This is given as:

$$c(\boldsymbol{x}, \boldsymbol{x'}) = \prod_{i=1}^{p} \exp\left[-\left(\frac{x_i - x_i'}{\delta_i} \right)^2 \right]. \tag{3.7}$$

where $\boldsymbol{\delta}$ is the correlation length vector with length p with the i - th element denoted δ_i. The parameter, δ_i describes the strength of the correlation in the i-th dimension of its input. The correlation between $\eta(\boldsymbol{x})$ and $\eta(\boldsymbol{x'})$ depends on the distance between $\boldsymbol{x}$ and $\boldsymbol{x'}$. The effect of $\boldsymbol{\delta}$ is to re-scale the distance between $\boldsymbol{x}$ and $\boldsymbol{x'}$. A low value of δ_i, suggests that the output values are weakly correlated over a narrow range of the i-th input $\boldsymbol{x}_i$. More generally, this most commonly used correlation function is often preferred because its infinitely differential and thus is very smooth. However, Stein (1999) (cited by Rasmussen (2006)) argues that such strong smoothness assumptions may be too unrealistic for many applications and suggests the Matern correlation function.

The Matern correlation function is given in the general form as:

$$c(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\sqrt{2\nu} r \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} r \right) \tag{3.8}$$

where Γ is the gamma function, K_ν is the modified Bessel function, and r is the squared sum of distance between inputs x and x', scaled by $\boldsymbol{\delta}$ as:

$$r = \left[\frac{(x_1 - x_1')^2}{\delta_1^2} + ... + \frac{(x_p - x_p')^2}{\delta_p^2} \right]^{0.5},$$

(Rasmussen, 2006). In general, the correlation function would be chosen based on any prior information that you have about the model. This would include information on the shape of the function, how “wiggly” it is, how differentiable it is, etc. The choice of the covariance functions becomes more crucial as n gets smaller.

Rasmussen (2006, Chapter 4) discusses these in more detail.

Prior distribution for the hyper-parameters of $\eta(\boldsymbol{x})$

The prior mean and covariance of $\eta(\boldsymbol{x})$ depend on the hyper-parameters $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\delta}$, respectively. For the prior distribution of $\boldsymbol{\beta}$ and σ^2 a weak prior is used generally:

$$p(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}. \tag{3.9}$$

However, it is possible to include prior knowledge of $\eta(.)$ through the prior distribution of these hyper-parameters. Oakley (2002) introduces a way of including expert prior knowledge about the simulator in the GPE. He also shows that where the number of training runs is limited, then including proper prior beliefs can be beneficial in reducing the posterior variance of the GPE.

We estimate the value of $\boldsymbol{\delta}$ by maximum likelihood estimation instead of a full proper Bayesian inference. Therefore we do not consider a prior for it, and we discuss the estimation of $\boldsymbol{\delta}$ in the next Section.

In summary, the prior model for $\eta(.)$ is given by

$$\eta(.)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim N\left(\boldsymbol{h}(.)^T\boldsymbol{\beta}, \quad \sigma^2 c(.,.)\right). \tag{3.10}$$

3.4.4 Deriving the posterior distribution: $\eta(\boldsymbol{x})|\boldsymbol{y}, \boldsymbol{\delta}$

Next, we observe $\eta(.)$ at the design events $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. This produces simulator output data:

$$\boldsymbol{y}^T = (y_1 = \eta(\boldsymbol{x}_1), ..., y_n = \eta(\boldsymbol{x}_n)).$$

We then update the prior distribution of $\eta(.)$ using the partitioning property of multivariate normal distributions. Given

$$\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim MVN(H\boldsymbol{\beta}, \sigma^2 A) \tag{3.11}$$

where

$$H^T = (\boldsymbol{h}(\boldsymbol{x}_1), ..., \boldsymbol{h}(\boldsymbol{x}_n)), \tag{3.12}$$

$$A = \begin{pmatrix} 1 & c(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & c(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ c(\boldsymbol{x}_1, \boldsymbol{x}_2) & 1 & & \vdots \\ \vdots & & \ddots & \\ c(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & & 1 \end{pmatrix}, \tag{3.13}$$

and equation (3.10), we can apply the partitioning property of multivariate normal distribution to get

$$\eta(.)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}, \boldsymbol{y} \sim GP(m^*(.), \sigma^2 c^*(., .)), \tag{3.14}$$

where,

$$\begin{aligned} m^*(\boldsymbol{x}) &= \boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{\beta} + \boldsymbol{t}(\boldsymbol{x})^T A^{-1}(\boldsymbol{y} - H\boldsymbol{\beta}), \\ c^*(\boldsymbol{x}, \boldsymbol{x}') &= c(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{t}(\boldsymbol{x})^T A^{-1}\boldsymbol{t}(\boldsymbol{x}'), \\ \boldsymbol{t}(\boldsymbol{x})^T &= (c(\boldsymbol{x}, \boldsymbol{x}_1), ..., c(\boldsymbol{x}, \boldsymbol{x}_n)), \\ \boldsymbol{y}^T &= (\eta(\boldsymbol{x}_1), ..., \eta(\boldsymbol{x}_n)). \end{aligned}$$

The parameters, $\boldsymbol{\beta}$ and σ^2 are regarded as nuisance parameters, and thus we aim to derive the distribution of $\eta(.)|\boldsymbol{\delta}, \boldsymbol{y}$ unconditional on $\boldsymbol{\beta}$ and σ^2. To do this we combine (3.9) and (3.11) using Bayes' Theorem to get a posterior distribution for

$\boldsymbol{\beta}|\delta, \boldsymbol{y}$ and $\sigma^2|\delta, \boldsymbol{y}$. But first, we consider the likelihood of $\boldsymbol{\beta}$ and σ^2:

$$f(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}) = (2\pi\sigma^2)^{(-\frac{n}{2})} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y} - H\boldsymbol{\beta})^T A^{-1}(\boldsymbol{y} - H\boldsymbol{\beta})\right\}, \tag{3.15}$$

and note that

$$(\boldsymbol{y} - H\boldsymbol{\beta})^T A^{-1}(\boldsymbol{y} - H\boldsymbol{\beta}) = (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T H^T A^{-1} H(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + (n - m - 2)\hat{\sigma}^2, \tag{3.16}$$

where

$$\hat{\boldsymbol{\beta}} = (H^T A^{-1} H)^{-1} H^T A^{-1} \boldsymbol{y} \tag{3.17}$$

$$\hat{\sigma}^2 = \frac{\boldsymbol{y}^T (A^{-1} - A^{-1} H (H^T A^{-1} H)^{-1} H^T A^{-1}) \boldsymbol{y}}{n - m - 2}. \tag{3.18}$$

The resulting posteriors are proper as long as $n \geqslant m + 2$. Then applying the Bayes' Theorem, we note that $\boldsymbol{\beta}, \sigma^2|\boldsymbol{\delta}, \boldsymbol{y}$ has a normal inverse gamma distribution:

$$f(\boldsymbol{\beta}, \sigma^2|\boldsymbol{\delta}, \boldsymbol{y}) \propto \sigma^{2^{-\frac{n+2}{2}}} \exp\left\{\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T H^T A^{-1} H(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + (n - m - 2)\hat{\sigma}^2\right\}. \tag{3.19}$$

Given the properties of normal inverse gamma distribution, it can be seen that

$$\boldsymbol{\beta}|\sigma^2, \boldsymbol{\delta}, \boldsymbol{y} \sim N\Big(\hat{\boldsymbol{\beta}}, (H^T A^{-1} H)^{-1}\Big), \tag{3.20}$$

$$\sigma^2|\boldsymbol{\delta}, \boldsymbol{y} \sim (n - m - 2)\hat{\sigma}^2 \chi^{-2}_{n-m}. \tag{3.21}$$

If we combine the distribution of $\eta(.)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}, \boldsymbol{y}$ (3.14) with the distribution of $\boldsymbol{\beta}|\sigma^2, \boldsymbol{\delta}, \boldsymbol{y}$ (3.21) and integrate out $\boldsymbol{\beta}$, we obtain

$$\eta(.)|\sigma^2, \boldsymbol{\delta}, \boldsymbol{y} \sim GP(m^{**}(.), \sigma^2 c^{**}(., .)), \tag{3.22}$$

where,

$$\begin{aligned} m^{**}(\boldsymbol{x}) &= \boldsymbol{h}(\boldsymbol{x})^T\hat{\boldsymbol{\beta}} + \boldsymbol{t}(\boldsymbol{x})^T A^{-1}(\boldsymbol{y} - H\hat{\boldsymbol{\beta}}), \\ c^{**}(\boldsymbol{x}, \boldsymbol{x}') &= c^*(\boldsymbol{x}, \boldsymbol{x}') + \Big(\boldsymbol{h}(\boldsymbol{x})^T - \boldsymbol{t}(\boldsymbol{x})^T A^{-1}H\Big) \\ &\qquad \times (H^T A^{-1} H)^{-1}\Big(\boldsymbol{h}(\boldsymbol{x})^T - \boldsymbol{t}(\boldsymbol{x})^T A^{-1}H\Big)^T, \end{aligned}$$

Similarly, if we combine the distribution of $\eta(.)|\sigma^2, \boldsymbol{\delta}, \boldsymbol{y}$ (3.22) with the distribution of $\sigma^2|\delta, \boldsymbol{y}$ (3.21) and integrate out σ^2, we obtain the following distribution conditional on $\boldsymbol{\delta}$

$$\frac{\eta(\boldsymbol{x}) - m^{**}(\boldsymbol{x})}{\sqrt{\frac{n-m-2}{n-m}} \quad \hat{\sigma}\sqrt{c^{**}(\boldsymbol{x}, \boldsymbol{x})}}|\boldsymbol{\delta}, \boldsymbol{y} \sim t_{n-m}, \tag{3.23}$$

where t_{n-m} is a student t distribution with $n-m$ degrees of freedom.

This is written in Bastos and O'Hagan (2009) as:

$$\eta(.)|\boldsymbol{y}, \boldsymbol{\delta} \sim StudentProcess(n-m, \quad m^{**}(.), \quad \hat{\sigma^2} c^{**}(.,.)) \tag{3.24}$$

This multivariate t distribution gives us a quick approximation of $\eta(.)$ for any $\boldsymbol{x}$ as $m^{**}(\boldsymbol{x})$ does not depend on $\eta(\boldsymbol{x})$. For input point $\boldsymbol{x}_T$, the point estimate is given by the posterior mean, $m^{**}(\boldsymbol{x}_T)$ and the posterior variance of this point estimate is given by $c^{**}(\boldsymbol{x}_T, \boldsymbol{x}_T)$. At training inputs, X, the posterior mean passes through the observed points exactly (as shown in (3.39)) with zero variance (3.43), as expected (explained further in Appendix 3.A).

Estimating the correlation length parameter, $\boldsymbol{\delta}$

The posterior result of the GPE is still conditional on $\boldsymbol{\delta}$. This is because the correlation functions $c(.,.)$ contain a vector $\boldsymbol{\delta}$ of parameters which describe the strength of correlation between outputs in each of the input dimension. The values of these parameters are unknown since the function $\eta(.)$ is unknown, and there is no analytic way of dealing with this uncertainty, unlike with parameters $\boldsymbol{\beta}$ and σ^2. The simplest option is to keep $\boldsymbol{\delta}$ fixed at a particular value, indicating an appropriate distribution for the roughness of $\eta(.)$. Prior knowledge of the roughness of $\eta(.)$ may suggest a value for $\boldsymbol{\delta}$, however, alternatively we could estimate $\boldsymbol{\delta}$ from the data.

One of the ways this can be done is by using the posterior mode, as suggested by Haylock (1997). The density function of $\boldsymbol{y}$ conditional on $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$ is given as

$$f(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}) = \frac{|A|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{1}{2}n}(2\pi)^{\frac{n}{2}}} \exp\left[-(\boldsymbol{y} - H\boldsymbol{\beta})^T \frac{A^{-1}}{2\sigma^2}(\boldsymbol{y} - H\boldsymbol{\beta})\right], \tag{3.25}$$

which can be thought of as a likelihood function for $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{\delta}$. Taking the product of this likelihood with the weak prior of $\boldsymbol{\beta}$ and σ^2 (3.9), and independently an uninformative uniform prior on $\boldsymbol{\delta}$, the posterior distribution can be obtained as:

$$f(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta}|\boldsymbol{y}) \propto \frac{|A|^{-\frac{1}{2}}}{(\sigma^2)^{\frac{1}{2}(n+2)}(2\pi)^{\frac{p}{2}}} \exp\left[-(\boldsymbol{y} - H\boldsymbol{\beta})^T \frac{A^{-1}}{2\sigma^2}(\boldsymbol{y} - H\boldsymbol{\beta})\right]. \tag{3.26}$$

Integrating out $\boldsymbol{\beta}$ gives:

$$f(\sigma^2, \boldsymbol{\delta}|\boldsymbol{y}) \propto \frac{|A|^{-\frac{1}{2}}|H^T A^{-1} H|^{-1}}{(\sigma^2)^{\frac{1}{2}(n+2-m)}} \exp\left[-(\boldsymbol{y} - H\hat{\boldsymbol{\beta}})^T \frac{A^{-1}}{2\sigma^2}(\boldsymbol{y} - H\hat{\boldsymbol{\beta}})\right], \tag{3.27}$$

and finally integrating out σ^2 and recognising that the resulting equation is proportional to an inverse gamma density function gives:

$$f(\boldsymbol{\delta}|\boldsymbol{y}) \propto (\hat{\sigma}^2)^{-\frac{(n-m)}{2}}|A|^{-\frac{1}{2}}|H^T A^{-1} H|^{-\frac{1}{2}}. \tag{3.28}$$

We find an optimal value of $\boldsymbol{\delta}$ by maximising the output of this likelihood function.

Toy GPE

For our toy example, we first calculate an estimate of δ by maximising the likelihood function (3.28). We choose to use the Gaussian correlation function (3.7). We estimate the value as $\delta = -0.65$ and using this value we can calculate $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ using equations 3.17 and 3.18 respectively. We note that both these calculations require the A matrix, which is a correlation matrix as shown in (3.13). We show the calculations for one element of the matrix, $A_{1,2} = c(x_1, x_2)$. We note that $p = 1$ and thus (3.7) reduces to

$$c(x_i, x_i') = \exp\left[-\left(\frac{x_i - x_i'}{\delta}\right)^2\right].$$

Substituting the relevant values will result in

$$\begin{aligned} c(x_1, x_2) &= \exp\left[-\left(\frac{-1 - -0.85}{-0.65}\right)^2\right]. \\ &= 0.92. \end{aligned}$$

We use the same approach to calculate every element in the A matrix. Below we show the values (to the nearest two decimal places) of $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$:

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} 3.92 \\ -2.7 \end{pmatrix}.$$

$$\hat{\sigma}^2 = 27.33.$$

We derive the GPE predictions and associated uncertainty around these predictions as

$$m^{**}\begin{pmatrix} 0.5 \\ 1.75 \end{pmatrix} = \begin{pmatrix} -1.83 \\ 0.72 \end{pmatrix} \tag{3.29}$$

$$\hat{\sigma}^2 c^{**}\begin{pmatrix} 0.5 & 0.5 \\ & , \\ 1.75 & 1.75 \end{pmatrix} = \begin{pmatrix} 0.01 & -0.05 \\ -0.05 & 1.3 \end{pmatrix}. \tag{3.30}$$

We use this GPE to predict the simulator output and the associated prediction uncertainty at more input values. We plot the result in Figure 3.2. The predicted values (red dashed line) are very close to the true simulator output (black solid line) for input values 0 and above. For the input values between $[-0.8,\ 0]$ the prediction uncertainty is quite large. This suggests that the GPE is very uncertain about predicting at these input values. Moreover, the prediction in this area is not even within the 95% interval. We can improve this by adding more training runs within that interval. In the next Section we look at diagnostics which are useful to assess the performance of the GPE and suggest ways to improve the GPE.

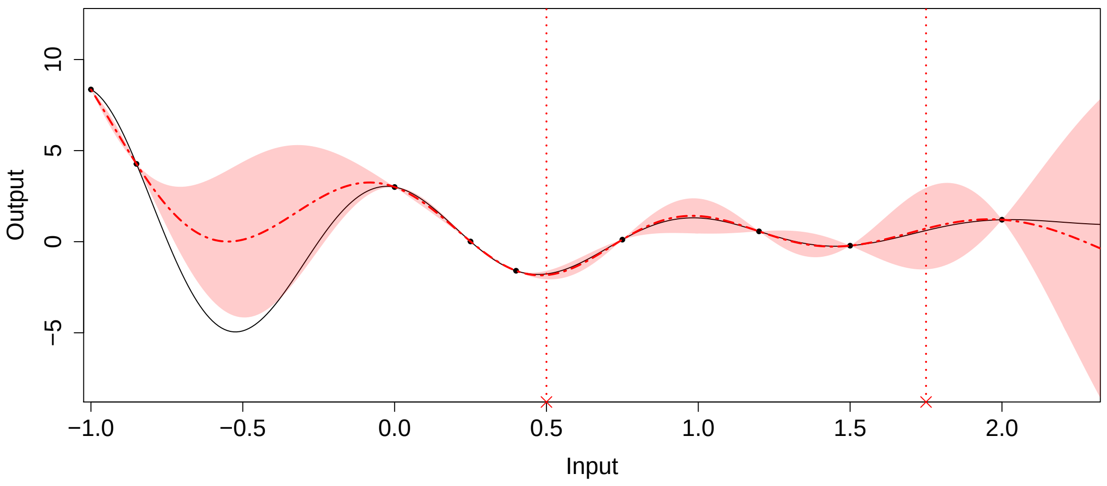

Figure 3.2: Univariate toy GPE - the points indicate the training runs, the red crosses are the events which we predict at, the red dashed line shows the GPE predictions and the shaded region represents the uncertainty around the predictions

3.5 Diagnostics for validating GPEs

It is important for the GPE to correctly represent the simulator, otherwise, inferences made using the GPE may not be valid. It is therefore, necessary to validate the GPE before using it for prediction and any further analysis. Invalid predictions could be a result of certain inappropriate assumptions that are made when fitting the GPE. It could be that the specified prior mean and correlation function do not appropriately represent the simulator. This could be due to the simulator being very responsive to small changes in the inputs between certain values therefore invalidating the stationarity assumption for the prior correlation function. If this is the case, then we expect extremely wide (narrow) credible intervals of the GPE predictions in regions of low (high) responsiveness. A GPE may also be invalid if the estimates of the hyper-parameters are inappropriate, which could be due to an unfortunate choice of the training data, resulting in predictions being consistently too low (or high) in areas of the input space.

A beneficial property of GPEs that we have seen is its ability to perfectly predict the simulator output at a design event with zero uncertainty. This means that when validating the GPE, we have to use events that are not included in the design. We could leave out some of the design events when fitting the GPE so that we can use them for validation. Suppose that the events we keep aside for validation, which we refer to as validation events, are denoted as $\tilde{X} = (\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_v)$. The result of the GPE at validation events modelled by $\eta(\tilde{X})|\boldsymbol{y}$, is a posterior mean, $E[\eta(\tilde{X})|\boldsymbol{y}]$, and a posterior variance, $V[\eta(\tilde{X})|\boldsymbol{y}]$ and $\sqrt{V[\eta(\tilde{X})|\boldsymbol{y}]}$ is the posterior standard deviation. We refer to the true simulator output for the validation events as *simulator output* denoted by $\tilde{\boldsymbol{y}} = (\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_v)$. We refer to the predicted output for the validation events as *GPE output* denoted as

$$E[\eta(\tilde{X})|\boldsymbol{y}] = (E[\eta(\tilde{\boldsymbol{x}}_1)|\boldsymbol{y}, \ldots E[\eta(\tilde{\boldsymbol{x}}_v)|\boldsymbol{y}]) \, .$$

A common calculation that we will use when performing the diagnostics is the residual error (or the prediction error), $\boldsymbol{\varepsilon}$. This is given by ε_i where $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_v)^T$ and

$$\varepsilon_i = (\tilde{\boldsymbol{y}}_i - E[\eta(\tilde{\boldsymbol{x}}_i)|\boldsymbol{y}]) \, . \tag{3.31}$$

In this Section we discuss a series of diagnostic tests carried out to assess the performance of GPE output against the simulator output. These diagnostics assess the validity of some assumptions or predictions made. In some cases an appropriate reference distribution is used for comparison. If the observed value of the diagnostics is too extreme comparatively then we have what we call a validation failure. If this is the case, and especially when these failures are numerous and too extreme, then modifications need to be applied to the GPE by either rebuilding the GPE with different training runs (or with more data) or choosing different priors. However, if there are only a few diagnostic errors or minor validation failures then the GPE can be declared valid and is often a logical decision. The presentation in this Section is based on Bastos (2010) and Bastos and O'Hagan (2009).

Coverage Statistic

We define *coverage* as the proportion of simulator output values (for the validation events) that lie between the $100(1-\alpha)\%$ range of the GPE predicted credible interval. A typical value for α is 0.05 such that we are looking for the 95% credible interval. If we choose a higher α, we increase the probability of having a higher proportion of values that lie within the interval. Suppose for residual $\boldsymbol{\varepsilon}_i$, the function $g(\boldsymbol{\varepsilon}_i)$ is given by:

$$g(\boldsymbol{\varepsilon}_i) = \begin{cases} 1, & \text{if } |\boldsymbol{\varepsilon}_i| < t_{n-p;(1-\alpha/2)} \times \sqrt{V[\eta(\tilde{\boldsymbol{x}}_i)|\boldsymbol{y}]}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.32}$$

Then the coverage value can be calculated by the following formula:

$$Coverage = \frac{\sum_{i=1}^{n}\left(g(\boldsymbol{\varepsilon}_i)\right)}{n}, \tag{3.33}$$

where $t_{n-p;(1-\alpha/2)}$ is the $100(1-\alpha)\%$ quantile of a student t distribution. The closer the coverage value is to $100(1-\alpha)$, the better the GPE calibration of the uncertainty intervals. This statistic can however be misleading if the average width of the credible interval of the GPE output is high.

Normalised Root Mean Square Error (NRMSE)

RMSE is a another measure of the residual error. The RMSE is calculated by the following formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{v} \varepsilon_i^2}{v}}.$$

RMSE penalises large residual errors more heavily than smaller residual errors. The RMSE value is in the same unit as the value being predicted and its value is relative to the range of the output. Stating the range of the output values allows for judging how good the GPE is. A drawback for using RMSE is that it cannot be used to compare with different outputs or even between GPE fits that use different training data.

NRMSE is a unit form of RMSE which normalises the RMSE to the range of the simulator output, where lower values indicate a better approximation. NRMSE is calculated using the following formula:

$$NRMSE = \frac{RMSE}{(\tilde{y}_{max} - \tilde{y}_{min})}.$$

Unlike the RMSE, this statistic can be used to compare results across different GPE's.

Standardised prediction errors (SPE)

Mathematically, the SPE is written as

$$D_i^I(\tilde{y}_i) = \frac{\varepsilon_i}{\sqrt{V[\eta(\tilde{\boldsymbol{x}}_i)|\boldsymbol{y}]}}. \qquad (3.34)$$

Each $D_i^I(\tilde{y}_i)$ is a validation diagnostic on its own, with the standard normal distribution as a reference distribution. Thus $\left|D_i^I(\tilde{y}_i)\right| > 1.96$ indicates a conflict between the simulator and the GPE. If there are a few SPE values that meet this criterion, they could be ignored, or they may indicate a local problem around those events. Further investigation can be carried out by adding more validation events around the problematic ones to assess the situation further. A large number of values of the same sign that meet this criterion may indicate that the prior mean function or the estimate of $\boldsymbol{\beta}$ may not be appropriate. It could also mean that the stationarity assumption is invalid. These problems could be the case especially if the errors are of the same sign. If the large values are close to the design events then it may indicate an overestimation of the correlation parameters, otherwise the estimate of the hyper-parameter σ^2 may be inappropriate. Complementary diagnosis apply to unusually small SPE values.

Mahalanobis Distance (MD)

Bastos and O'Hagan (2009) define Mahalanobis Distance, D_{MD}, as a summary statistic of the SPE between the GPE and the simulator output as:

$$D_{MD}(\tilde{\boldsymbol{y}}) = \boldsymbol{\varepsilon}^T \big(V(\eta(\tilde{X})|\boldsymbol{y}\big)^{-1} \boldsymbol{\varepsilon}, \tag{3.35}$$

which takes into account correlation amongst the output. The reference distribution for $D_{MD}(\tilde{\boldsymbol{y}})$ is the scaled F-Snedecor distribution with degrees of freedom as v and $(n-m)$ where m is the dimension of $h(.)$. If $D_{MD}(\tilde{\boldsymbol{y}})$ is too large (small) compared to the reference distribution then it indicates that the GPE is over-confident (under-confident). This suggests that the uncertainty expressed in the GPE predicted posterior variance is too low (high) compared to the observed residuals, β has been poorly estimated and, σ^2 is underestimated (overestimated) or correlation length parameters $\boldsymbol{\delta}$ have been overestimated (underestimated). A moderate value of $D_{MD}(\tilde{\boldsymbol{y}})$ suggests the GPE is valid.

We choose to present this diagnostic graphically. We plot the probability density of random samples drawn from a F-Snedecor distribution. We also plot the value of $D_{MD}(\tilde{\boldsymbol{y}})$ as a vertical line. We expect that the vertical line lies within the boundaries of the distribution. If the vertical line is on the left of distribution the MD is said to be too small. Conversely, if it is on the right of the distribution the MD is said to be too large.

Variance Decomposition

Interpreting SPE's may be misleading as the individual errors may be correlated, and we may fail to recognise conflicts between GPE and simulator. Bastos and O'Hagan (2009) define G to be a standard deviation matrix such that $V[\eta(\tilde{X})|\boldsymbol{y}] = GG^T$. Then the vector of transformed errors

$$D_G(\tilde{\boldsymbol{y}}) = G^{-1}\boldsymbol{\varepsilon}, \tag{3.36}$$

are uncorrelated and have unit variances. There are different choices of the decomposition of G. One choice is a Cholesky decomposition, which is an upper triangular matrix (with positive diagonal entries), denoted as R, such that $V[\eta(\tilde{X})|\boldsymbol{y}] = R^T R$.

An arbitrary permutation of rows and columns of matrix $V[\eta(\tilde{X})|\boldsymbol{y}]$ can lead to a different choice of R, and thus we may be able to detect different interpretations of the problem. Bastos and O'Hagan (2009) suggest that the most effective permutation would be such that the first pivoting element index has the largest variance conditional on the training data, the second pivoting element index has the largest variance conditional on training data and the first pivoted element in the sequence, the third pivoting element index has the largest variance conditional on training data and the first two pivoted elements, and so on. They define the vector of transformed errors with this way of permuting the elements as the pivoted Cholesky errors, which can be obtained by applying the pivoted Cholesky decomposition. This returns a matrix G such that $P^T V[\eta(\tilde{X})|\boldsymbol{y}]P = R^T R$, where P is a permutation matrix and R is the Cholesky decomposition matrix of $V[\eta(\tilde{X})|\boldsymbol{y}]$. Permutation matrices are orthogonal, thus $V[\eta(\tilde{X})|\boldsymbol{y}]$ can be rewritten as $V[\eta(\tilde{X})|\boldsymbol{y}] = PR^T RP^T$ such that $G = PR^T$ is the Pivoted Cholesky Decomposition (PCD) of $V[\eta(\tilde{X})|\boldsymbol{y}]$ in (3.36) which we refer to as $D_{PC}(\tilde{\boldsymbol{y}})$.

We present this graphically by plotting the elements of the vector $D_{PC}(\tilde{\boldsymbol{y}})$ against the pivoting index. The pivoting index is based on the order of the pivoted Cholesky errors. The error is expected to fluctuate around 0 with a constant variance and no special patterns. Too many large errors indicate an underestimation of variance and vice versa. Both cases can also suggest a non-stationary simulator. Extremely large (or small) errors at the beginning of the plot (i.e. on the left side) indicates underestimation (or overestimation) of predictive variance or non-stationarity. However, extreme large (small) errors at the end of the plot (i.e., on the right side) indicates overestimation (or underestimation) of the correlation length parameters.

Quantile-Quantile (QQ) plots

Another diagnostic that we look at is the QQ plot of the pivoted Cholesky errors (which aren't independent). In a QQ plot if the points lie close to the 45 degree-line through the origin, then the normality assumption of errors is a reasonable assumption. If the points cluster around in a line with slope greater (or lesser) than one, then the plot suggests that the predictive variability was underestimated (or overestimated). Curvature in the plot suggests non-normality, while outliers at either end of the plot suggests non-stationarity or local fitting problems.

Cross Validation (CV)

Suppose, that it is only possible to model n runs from the simulator, and all the n runs are crucial to train the GPE. In this case validation can be performed using cross validation. This is where the dataset is split into k smaller sets and validation is done on each of these sets in a loop using the other sets as training data. This is known as the k-fold CV.

Given a set of n model runs in a random order, divide the training runs into k folds. This should result in k chunks of approximately $\frac{n}{k}$ model runs each. Then, for $i = 1, \ldots, k$, we train the GPE using all the model runs not in fold i, and test the GPE using the model runs in fold i using diagnostics mentioned in this section. By using k fold CV, we may be able to identify bad GPEs, however it may result in higher variances of the predicted values. The predicted values obtained from performing CV can then be diagnosed using similar methods mentioned in this section. We do not apply cross validation techniques in this thesis, but we choose to mention it for guidance purposes.

3.6 Case Study - Greater Wash, UK

In this case study we aim to illustrate how to use a GPE in a real application context, the benefits of using a GPE and how it performs in comparison to a traditional method. We explore these using the context of case study 2 as described in Chapter 2.2.2.

Traditionally, at HR Wallingford, a relatively simple meta-modelling technique has been applied to a range of applications. The technique used is called the “look up table“ (LUT) approach. It is widely applied in relation to coastal flood forecasting and wave modelling applications. Some examples include work done by Environmental Agency and Defra (2004), Maddux et al. (2006), Chawla et al. (2012), and

Deltares (2017), where they use the LUT table approach to speed up their analysis.

The Traditional LUT Approach

Numerous projects carried out at HR Wallingford require the SWAN model to be evaluated at a large number of events. Typically a computational burden or time constraint is imposed. Thus to avoid running the simulator at all the events a representative subset of events are selected to be run by the simulator. These events set the basis of an interpolation matrix to predict the output of the large number of events.

The LUT approach uses a regular grid technique to select the representative design events. For the selection of the design events, the input data are discretised into a LUT matrix, where a user selects a number of values from each input dimension and chooses to run the model at all combinations. In practice, the modeller would typically place constraints on the resolution and limits for the design points based on prior knowledge of, for example, the geography of the site and boundary conditions. We note that with this approach the selected design does not cover the input parameter space well, as there are only a few selected design points for each input (see Figure 3.3). Additionally, the number of events required in the design can become excessive if the problem is defined by a large number of input variables.

Once the design is selected, the simulator is run for these events and model evaluations are obtained. Multi-dimensional linear interpolation is then used to evaluate the model at all the remaining events that we wish to run the simulator for. On inspection of the results additional events can be run and used to fill in parts of the space where it is apparent there are limitations relating to the use of linear interpolation, i.e. where there are non-linearities in the output response surface.

For the LUT approach in this case study the time series of offshore wave and wind conditions were discretised into a five dimension LUT comprising 8 discrete wave heights, 9 wave directions, 7 wind speeds, 12 wind direction sectors and 3 water levels. Excluding combinations that did not occur in the offshore time series, this gave a total of approximately 16,000 design points.

The GPE Approach

For the GPE approach, we also select a suitable subset of design points that represent the full set of events. In general, in order for the GPE to give robust predictions, it is important that the design points are selected in a way that ensures they are well spread out, preferably as far as possible from each other, covering the entire input parameter space. There are many different ways of selecting design points. The regular grid method, used in the LUT approach, does not work well with GPEs because of its collapsing property, i.e. multiple points have a fixed coordinate value when projected onto a variable axis, (Camus et al., 2011b). For each particular dimension (keeping values of other dimensions constant), there would only be a few unique values to estimate from. This is because of the matrix like structure when selecting the data.

There are alternative methods to select design points (detailed discussed in Chapter 5), however here we choose to use a standard maximum dissimilarity algorithm (MDA) approach as described by (Camus et al., 2011a'b), and as applied in the context of coastal analysis by Camus et al. (2011a' 2013); Gouldby et al. (2014). This algorithm analyses the data using a Euclidean measure of distance between points in the multidimensional space. Having normalized the input variables, and given an initial value (the starting point), the MDA selects the next point that is the furthest away in Euclidean distance in the multidimensional space. This method outputs a set of design points which efficiently represents the full set of events. Using this approach we select a 1000 events to represent the input parameter space.

Again, once the design is selected, the simulator is run for these events and model evaluations are obtained. We then fit a univariate GPE to each of the outputs from the simulator (wave height, wave period, and wave direction) given the design events. Note however, due to the periodic nature of the offshore wave direction, we convert wave direction to x and y components as described further in Section 5.3. We choose a linear mean function and a Gaussian covariance function to describe our prior beliefs of the simulator for each of the outputs.

3.6.1 Results

Figure 3.3 shows a plot of the selected design for the two approaches overlaid on the full set of events. For the regular grid from the LUT approach (lower triangular matrix) the complete set of approximately 16,000 events are plotted; for the GPE approach (upper triangular matrix) only the first 150 of the 1,000 design events selected using MDA are plotted. The full set of events, $\mathcal{X}$ are plotted in a lighter colour, and the selected design points are plotted in black. Visual inspection of Figure 3.3 shows the design points for the LUT approach are inefficient in providing coverage of the input variable space when compared to the MDA approach. For example, the plot of wind speed against water level appears to have much fewer events for the LUT approach. This is because they are in the form of a recti-linear grid, and many events “collapse” on top of each other. For example see the water level against wind speed in Figure 3.3; there are only three unique water levels and seven unique wind speeds, however other combinations do exist.

This figure helps us visualise the data behind each of the approaches, and thus gives us more confidence in the GPE approach as the MDA method ensures that we are efficiently representing the full set of events, even with a small number of points.

The SWAN model was run for the 16,000 design points for the LUT approach and the subsequent results at the near shore point were used to populate the LUT. This LUT was then used to predict the output for the validation events using mul-

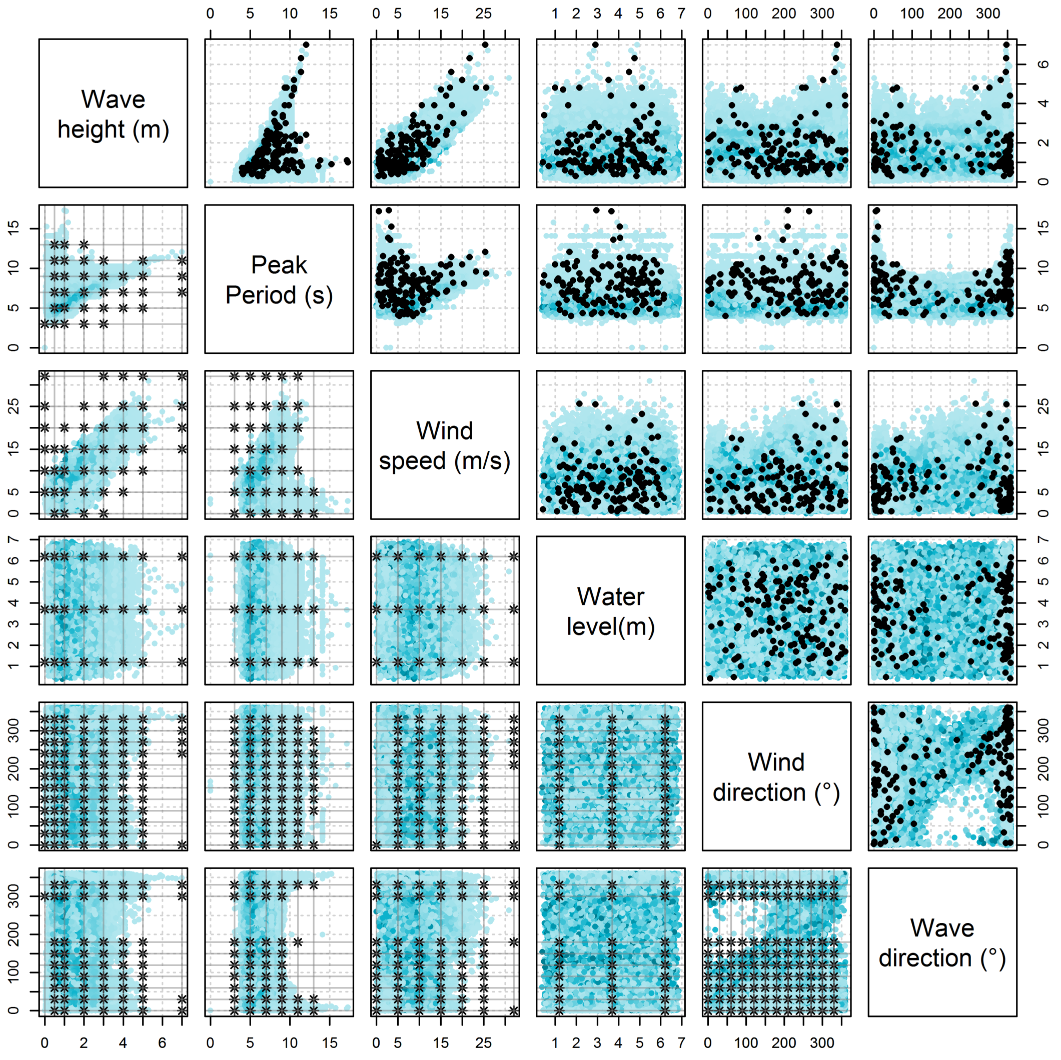

Figure 3.3: Design points selected using the Regular grid (black asterisks – lower triangular matrix) and the MDA approach (black circles – upper triangular matrix). The lower triangular matrix shows poor coverage in each dimension compared to the upper triangular matrix.

tidimensional linear interpolation. For the GPE approach, SWAN was run for the 1,000 design events selected using MDA. To gain an understanding of the performance of the GPE as the number of design events increases, separate GPEs were fitted to increasing number of design points, ranging from 10 to 1,000. The NRMSE statistics of the LUT and GPE are presented in Figure 3.4, where the dashed line shows the LUT NRMSE and the solid line shows the GPE NRMSE for various

GPEs.

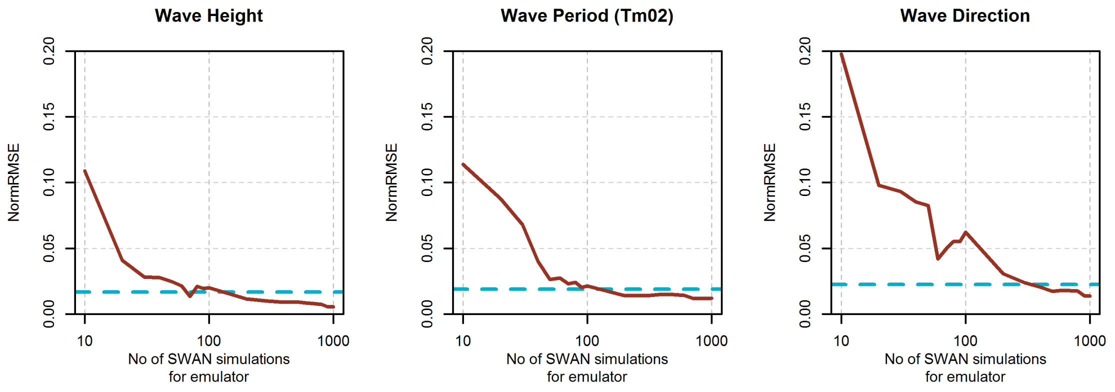

Figure 3.4: NRMSE for Wave height (measured in metres), period (seconds)and direction (degrees) using LUT (dashed line – with 16000 design events) and GPE (solid line – using between 10 to 1000 design events). The GPE outperforms the LUT using only 200 design points. Further gains in accuracy with the GPE are possible with more design points.

The intersection of the dashed and solid line in Figure 3.4 is where the GPE approach starts to outperform the LUT approach. It is evident, from Figure 3.4 that in this case study the GPE approach achieves a similar level of accuracy as the LUT approach measured by the NRMSE over the validation dataset using only about 200 design events, in contrast to the 16,000 used for the LUT. This is a big saving on computational time. Moreover, we can see that further gains in accuracy can be made if more design events were used. It is noticeable that there are a few jumps in the prediction errors in the GPE approach, on average however, the NRMSE decreases with an increase in design events. For the subsequent analysis on this case study, a GPE fitted to 500 design events has been used.

Figure 3.5 shows a scatter plot of the LUT (top panel) and GPE (bottom panel) predictions against the true SWAN output for the full validation dataset. We note that in reality it is seldom possible to obtain such a large validation dataset, however we choose to present the following diagnostics using the full dataset to show the comparison on performance against the LUT approach. Later, we will use a subset

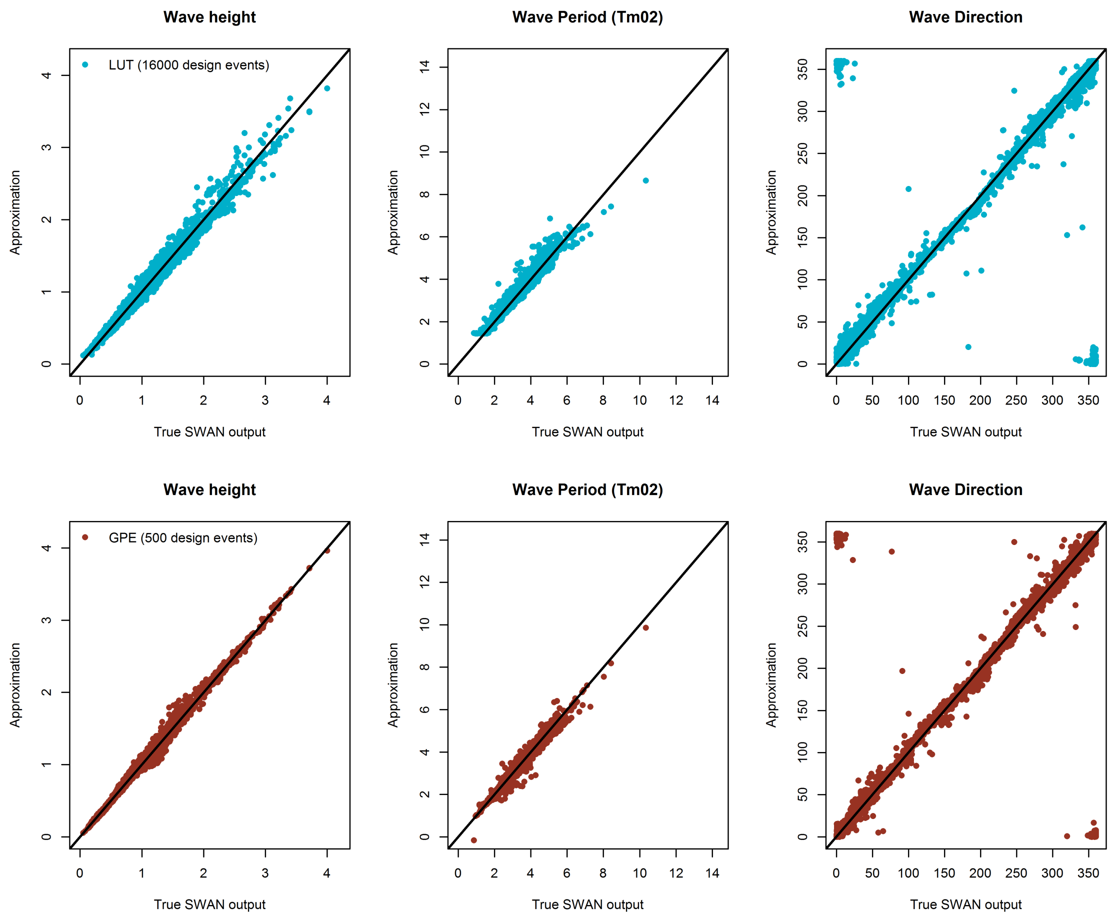

Figure 3.5: Comparison of approximations using LUT (upper panel) and GPE-MDA (lower panel) with true SWAN output. The LUT has a wider spread of points compared to the GPE method which indicates worse performance. The off-diagonal cluster of points for the wave direction plots are due to the periodic nature of this variable.

of the MDA selected design events for validating the GPE. From Figure 3.5 it is evident that the predictions using the LUT approach show slightly more scatter whereas the predictions from the GPE approach, appear to have a better fit. The wave height plot shows that both methods perform well. This may be because of a high correlation between the inputs and wave height compared to the other outputs. From Figure 3.5 we can conclude that, overall the GPE approach results in better predictions compared to the LUT approach and were achieved with much fewer design points (500) compared with the LUT. Note the off diagonal cluster of points for the wave direction plot in Figure 3.5 are due to the periodic nature of this variable, where 0° is the equivalent to 360° for example.

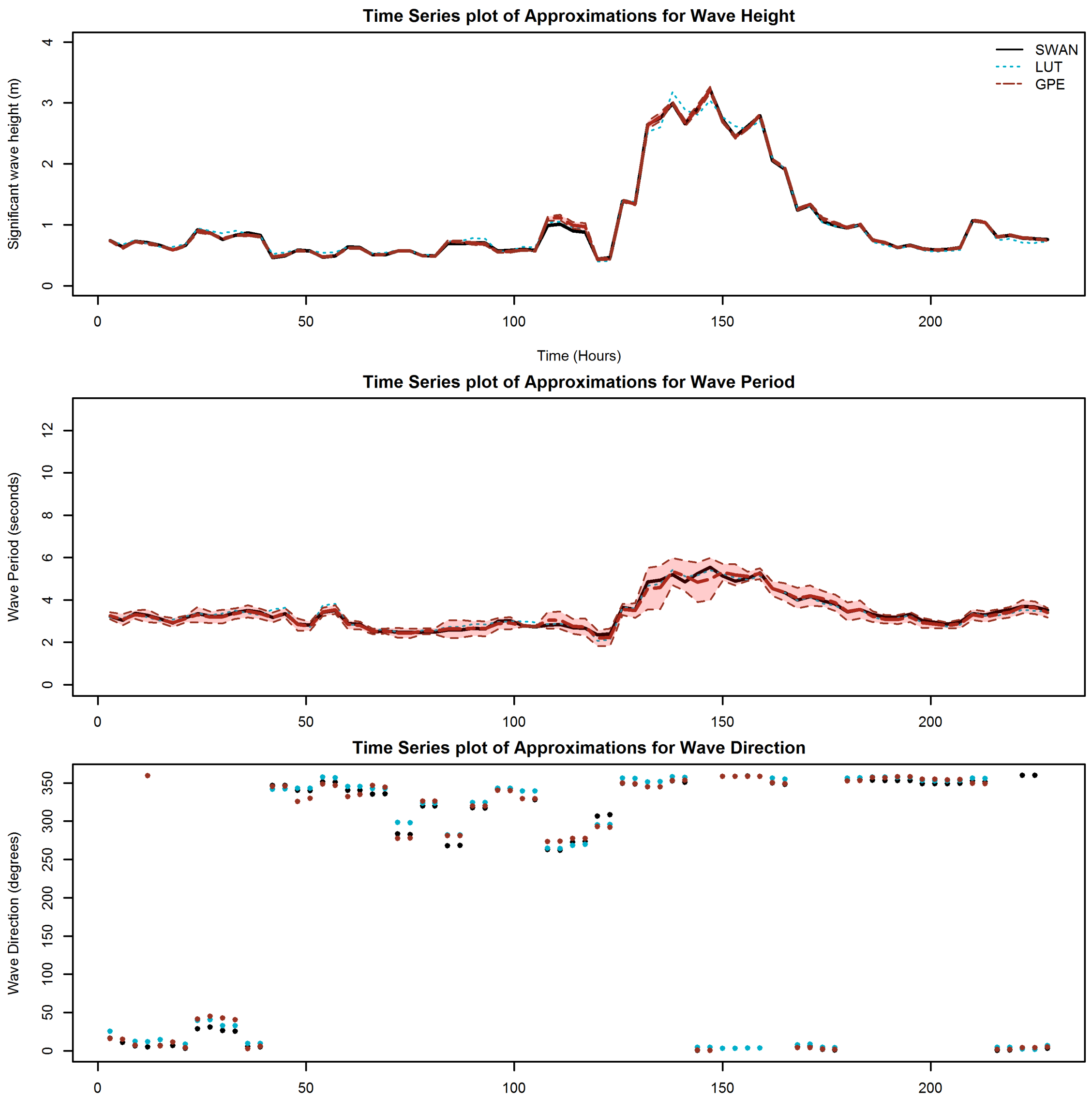

Figure 3.6: Time series plot of Significant wave height, wave period and wave direction showing the predictions from LUT (dotted blue line) and GPE (dashed red line). The shaded region represents the 95% credible interval from the GPE. The true SWAN output is presented with a solid black line (or dots in the bottom panel)

The 5430 validation points correspond to events occurring at different times. We select a subset of 75 consecutive times and plot their means and confidence intervals. Figure 3.6 illustrates the predictions from both the approaches by showing a time series of plot of these events for near shore wave height, wave period and wave direction for a sample within the validation dataset. In each plot, the dotted (blue) line represents the predictions from the LUT approach and the dashed (red) lines represent the GPE approach. The shaded region in the top and middle panel represents the 95% credible intervals for the GPE approach which represents the emulator uncertainty around the GPE prediction. For the top panel it is evident from the graph that both approximation methods very accurately reproduce the SWAN model predictions of near shore wave height for this time period. Moreover the 95% credible interval is very narrow suggesting that the GPE has high levels of confidence about its prediction.

The middle panel shows that the GPE approach provides a better (or similar) estimate of SWAN output than the LUT approach as the GPE line appears closer to the SWAN line compared to the LUT line. Moreover, it also shows that the SWAN output appears between the 95% credible intervals of the GPE for a majority of time. It is also evident that the credible intervals widens and narrows depending on how uncertain the predictor may be at a particular event based on how close it is to a design event. Note that for when time (x axis) is about 105 hours it is evident that the GPE credible interval reduces to zero and the prediction for the GPE approach is exact. This is because that event is a design event used to train the GPE. This property holds for all design events in the GPE approach. From the bottom panel it is also evident that the LUT predictions are slightly further away from SWAN when compared to the GPE predictions.

Next, we validate the GPE's (fitted with 500 design events) that we use to predict near shore wave height and wave period (we do not include direction here, because we have used x and y components rather than degrees (more details in Section 5.3)). In a more realistic validation scenario we would not have a large number of events where we can validate the GPE. Here, we choose to validate the GPE based on the next 100 MDA selected events (i.e. MDA event numbers 501 to 600). We use the

MDA selected events because these are the events from the full dataset that are the most distinct from each other, hence enabling us to validate the GPE on the most difficult subset for the GPE.

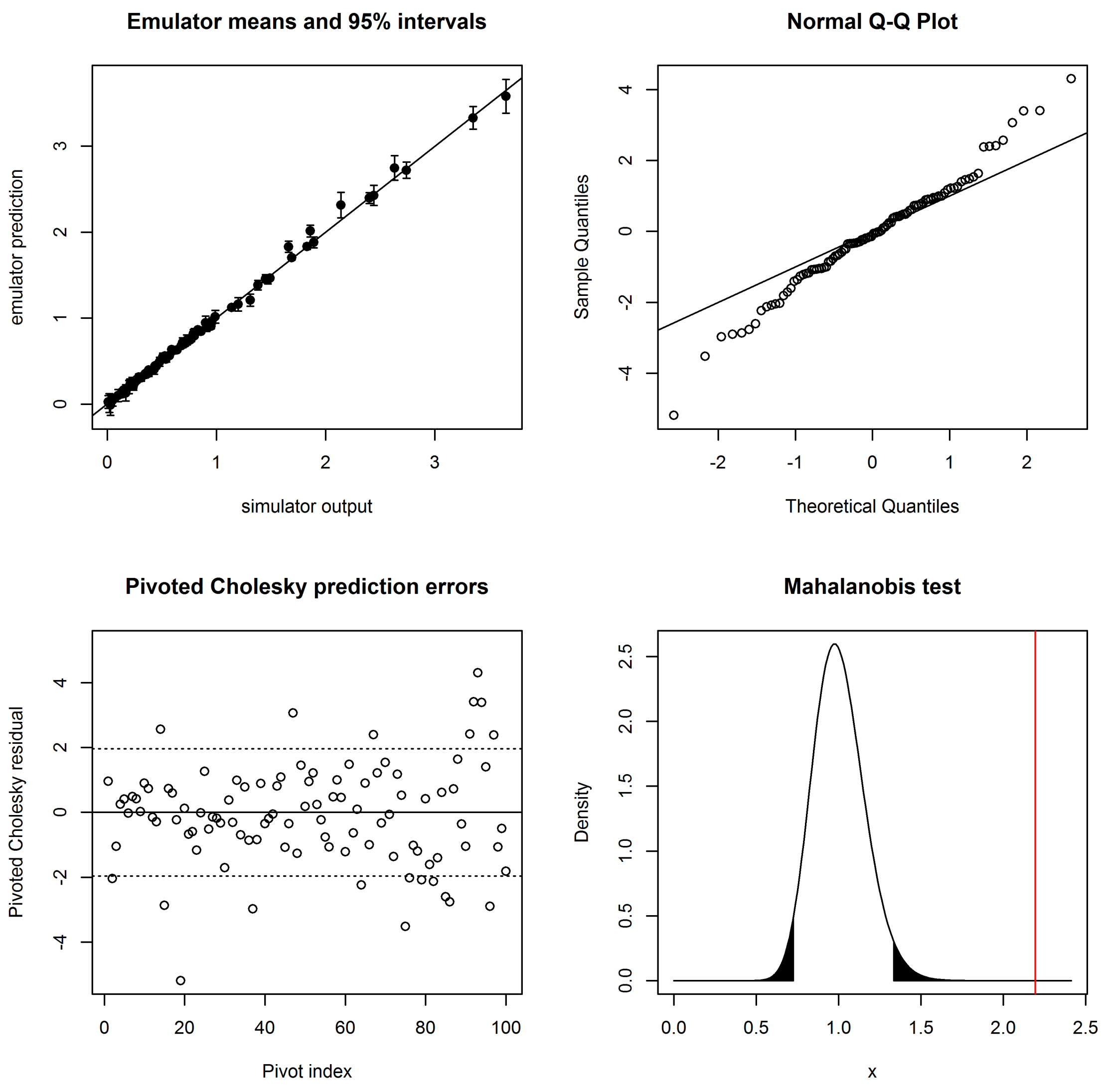

Figure 3.7: Validation plot of the GPE fit for Wave height ($Hsig$)

The top left plot in Figure 3.7 and Figure 3.8 shows the GPE predictions plotted against the true simulator output. The diagonal line represents perfect predictions and the error bars represent the 95% credible interval from the GPE prediction. This plot helps to add value to the coverage statistic. The coverage statistic summarises

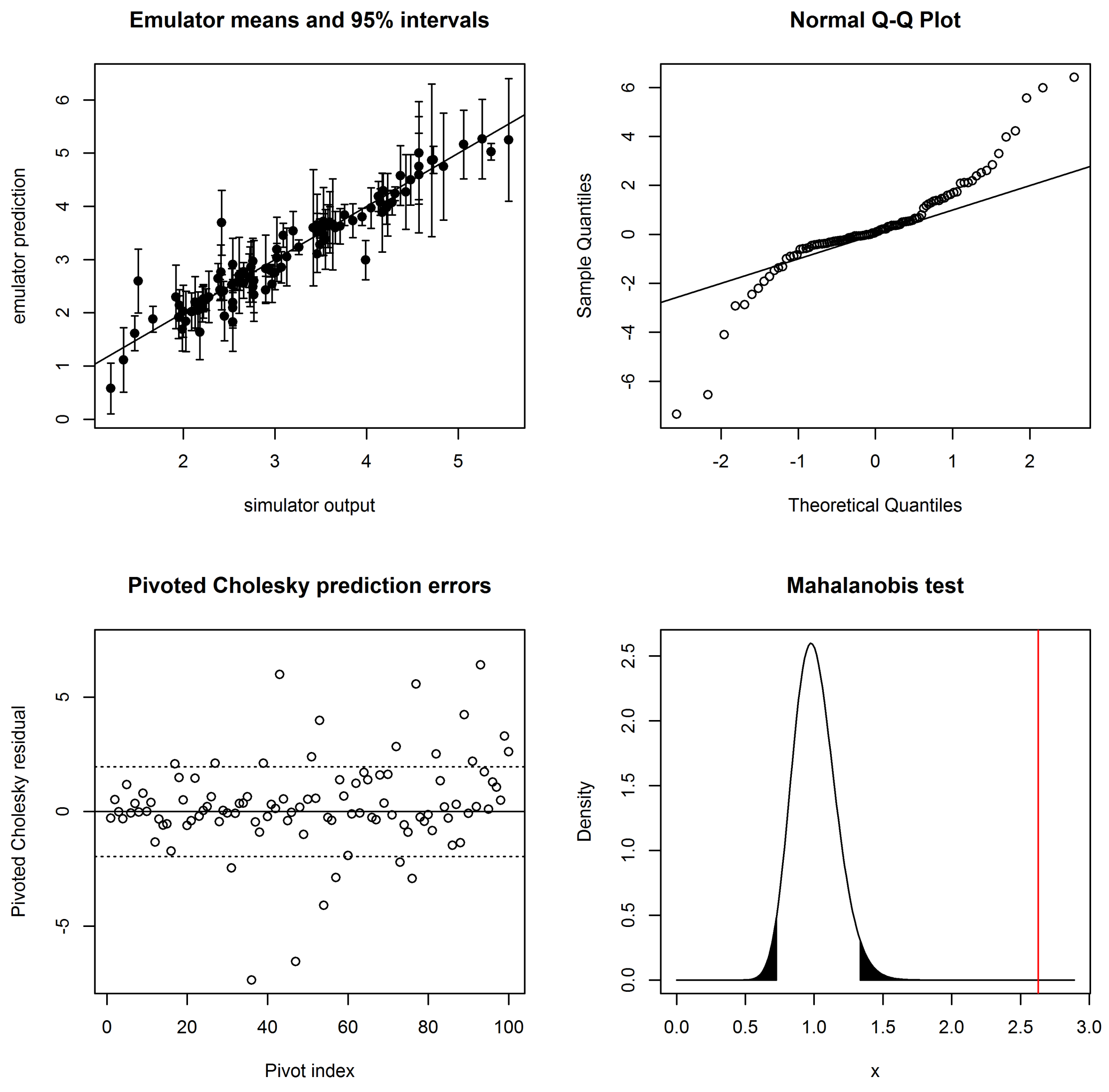

Figure 3.8: Validation plot of the GPE fit for Wave period (Tm_{02})

the percentage of error bars that pass through the straight line. We also note how wide the error bars are and if the they are consistently too wide or too narrow.

We note that the coverage value for GPE predictions for wave height is 0.88 and for wave period is 0.84. This means that true simulator wave height and wave period lie within the GPE predicted uncertainty 88% and 84% of the times. We can deduce that the GPE is slightly over confident about its predictions. It is evident from the top left plot in Figure 3.7 and Figure 3.8 that although it looks like the

GPE predicts SWAN well, the estimated uncertainty is quite small for around 15% of the points. The latter is less clear in Figure 3.7.

We note that the NRMSE for wave height GPE predictions is 0.01 and for wave period is 0.07. This suggests that the predictions are really close to the true simulator value.

The top right plot in Figure 3.7 and Figure 3.8 shows a QQ plot of the pivoted Cholesky errors. It is evident from Figure 3.7 that most of the points lie close to the 45 degree-line. We note that the cluster of points has a gradient greater than one, which again suggests the posterior variance was underestimated, (Bastos and O'Hagan, 2009). Similarly for Figure 3.8, we note that the gradient of the cluster of points is greater than one. However we also note that the points curve off in the tail of the distribution which may suggest non-normality. Moreover, we note for both figures that they are a few outliers in the tail of distribution which suggest local fitting problems.

The bottom left plot in Figure 3.7 and Figure 3.8 shows a plot of the pivoted Cholesky errors against the pivoting index. We note that most of the points lie between the two dotted lines that represent two standard deviations of a standard normal distribution. For both plots there are a few points that are above and below the dotted lines, but there does not seem to be a recognisable pattern. This can suggest a non-stationarity simulator but the errors are not large enough to suggest critical failures.

The bottom right plot in Figure 3.7 and Figure 3.8 shows the MD statistic (the red vertical line) overlaid on a reference distribution. For both the plots the red line lies on the right hand side of the distribution. It is evident that the $D_{MD}(\tilde{\boldsymbol{Y}})$ value is moderately big and suggests that the GPE is slightly overconfident.

For the GPE for wave height and wave period, the validation failures are due to the overconfidence in the GPE and hence the predicted variance is low. To some extent this is a tolerable validation failure and can be overlooked. Alternatively, if time permits, we could try to overcome this problem by refitting the GPE to a different set of design events, choose different priori, or we could increase the number of our design events, although this may not always be possible.

3.6.2 Discussion of results

The performance of two meta-modelling techniques has been compared for the spectral wave model SWAN, which can be expensive to run. The traditional LUT approach uses a regular grid and multidimensional linear interpolation techniques, whilst the GPE approach uses a design selection method that chooses the most dissimilar events from a dataset (MDA) and then uses the GPE to approximate.

We have shown the regular grid does not usually cover the marginal input space efficiently except possibly when combined with the modeller's prior knowledge so that the grid resolution may be increased around important areas. The number of design points required increases quickly with the number of input dimensions when using the regular grid approach. The linear interpolation technique, although easy to understand and implement, only utilises the input-output relationship from the closest design points rather than from potentially a much wider range as the GPE does. As a result, the LUT approach requires a much higher number of training runs in order to match the accuracy of the MDA approach, thus proving to be computationally expensive in comparison with the MDA approach.

In contrast, for the GPE approach, the design points provide efficient coverage both marginally and multi-dimensionally and the selection can be performed with limited prior knowledge. The GPE approximation technique is also superior to linear interpolation in a way that the trend of the output is estimated from all points in the design, rather than only the nearest points, as is in linear interpolation. The GPE allows modelling of fairly complicated functional forms. Another advantage of the

GPE is that the output is a distribution. This can be used as an input for subsequent models, or sensitivity and uncertainty analysis as uncertainty can be appropriately evaluated using the relevant posterior distributions. For this case study, it has been shown that the GPE approach is better than the traditional LUT approach in terms of overall computational efficiency and accuracy. The GPE approach can achieve similar RMSE with less than 10% of the design points used by the LUT approach.

We note that it is necessary to realise the context of the application, i.e. if accurate predictions are all that is necessary, over or under estimation of the predictive uncertainty may not be worthwhile investigating further. However if the predictions and uncertainty around these predictions are both required for further analysis then it may be more critical to investigate these errors. Nevertheless, it is always advisable to examine the diagnostics to ensure that nothing unexpected is going on.

3.7 Conclusion

In this Chapter, we have shown that a basic univariate GPE approach, with only a limited number of design events, can give quite accurate predictions and prediction uncertainty estimates. We also describe several diagnostics that can be used to assess the performance of the GPE. We go on to compare a basic GPE with a traditional approach and prove that the GPE can give accurate predictions compared to the conventional approach. In the case study presented in this chapter, we use a univariate GPE for each output assuming that the output variables are independent. Further gains are expected to be made using multivariate GPE approaches which will enable us to model all the outputs jointly to incorporate the dependencies between output variables. The next Chapter will look at approaches to modelling multiple outputs jointly.

Moreover, by requiring fewer simulator runs to training a GPE compared to traditional methods, the use of the GPE approach allows a potential increase in the spatial resolution or complexity of the model. For example, one could compare how close to the physical observation SWAN is and focus on how to close this gap, through modelling more of the physical processes and at higher resolutions. The GPE approach also facilitates an efficient uncertainty and sensitivity analysis.

This method has proven to be effective in practice at HR Wallingford. A recent national scale analysis undertaken in England has used the MDA and GPE technique to generate a large near shore wave dataset. This data set has the potential to be used for a wide-range of purposes including a coastal flood risk assessment, and for use in climate change impact assessments and coastal flood forecasting (Wyncoll et al., 2016; Gouldby et al., 2016).

Appendix

3.A Perfect prediction at training event

In this appendix, we first show that the posterior mean is equal to the simulator output if predicting at one of the training events. Then, we show that this prediction has zero posterior variance.

We recall that the posterior mean of $\eta(.)|\delta, \boldsymbol{y}$ is given by the equation:

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

Note that

$$H\hat{\boldsymbol{\beta}} = \begin{pmatrix} \boldsymbol{h}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_n)^T \end{pmatrix} \hat{\boldsymbol{\beta}}. \tag{3.37}$$

Note also that we can write $A = (\boldsymbol{t}(\boldsymbol{x}_1), \ldots, \boldsymbol{t}(\boldsymbol{x}_n))$. It then follows that

$$AA^{-1} = \begin{pmatrix} \boldsymbol{t}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{t}(\boldsymbol{x}_n)^T \end{pmatrix} A^{-1} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

and that

$$\boldsymbol{t}(\boldsymbol{x}_i)^T A^{-1} = \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{pmatrix} \tag{3.38}$$

Then for a single point $\boldsymbol{x}_i$

$$\begin{aligned} m^{**}(\boldsymbol{x}_i) &= \boldsymbol{h}(\boldsymbol{x}_i)^T \hat{\boldsymbol{\beta}} + \boldsymbol{t}(\boldsymbol{x}_i)^T A^{-1}(\boldsymbol{y} - H\hat{\boldsymbol{\beta}}) \\ &= \boldsymbol{h}(\boldsymbol{x}_i)^T \hat{\boldsymbol{\beta}} + (y_i - \boldsymbol{h}(\boldsymbol{x}_i)^T \hat{\boldsymbol{\beta}}) \quad \text{(from equation (3.37) and (3.38))} \\ &= y_i \end{aligned} \tag{3.39}$$

Similarly, the posterior variance of $\eta(.)|\delta, \boldsymbol{y}$ is given by the equation: $\sigma^2 c^{**}(.,.)$, where:

$$\begin{aligned} c^{**}(\boldsymbol{x}, \boldsymbol{x}') = c^*(\boldsymbol{x}, \boldsymbol{x}') &+ \Big(\boldsymbol{h}(\boldsymbol{x})^T - \boldsymbol{t}(\boldsymbol{x})^T A^{-1} H\Big) \\ &\times (H^T A^{-1} H)^{-1} \Big(\boldsymbol{h}(\boldsymbol{x})^T - \boldsymbol{t}(\boldsymbol{x})^T A^{-1} H\Big)^T. \end{aligned}$$

We note that for any input i

$$t(\boldsymbol{x}_i) = c(\boldsymbol{x}_i, \boldsymbol{x}_i) \tag{3.40}$$

and

$$c(\boldsymbol{x}_i, \boldsymbol{x}_i) = 1 \tag{3.41}$$

For a single point $\boldsymbol{x}_i$

$$\begin{aligned} c^*(\boldsymbol{x}_i, \boldsymbol{x}_i) &= c(\boldsymbol{x}_i, \boldsymbol{x}_i) - \boldsymbol{t}(\boldsymbol{x}_i)^T A^{-1} \boldsymbol{t}(\boldsymbol{x}_i) \quad \text{(from equation 3.14)} \\ &= 1 - \boldsymbol{t}(\boldsymbol{x}_i) \quad \text{(from equation 3.40 and 3.41)} \\ &= 0 \quad \text{(from equation 3.41)} \end{aligned} \tag{3.42}$$

The posterior variance for point $\boldsymbol{x}_i$ is

$$\begin{aligned}
c^{**}(\boldsymbol{x}_i, \boldsymbol{x}_i) &= c^*(\boldsymbol{x}_i, \boldsymbol{x}_i) + \Big(\boldsymbol{h}(\boldsymbol{x}_i)^T - \boldsymbol{t}(\boldsymbol{x}_i)^T A^{-1}\boldsymbol{h}(\boldsymbol{x}_i)^T\Big) \\
&\qquad \times (H^T A^{-1} H)^{-1}\Big(\boldsymbol{h}(\boldsymbol{x}_i)^T - \boldsymbol{t}(\boldsymbol{x}_i)^T A^{-1}\boldsymbol{h}(\boldsymbol{x}_i)^T\Big)^T \\
&= \Big(\boldsymbol{h}(\boldsymbol{x}_i)^T - \boldsymbol{h}(\boldsymbol{x}_i)^T\Big)(H^T A^{-1} H)^{-1}\Big(\boldsymbol{h}(\boldsymbol{x}_i)^T - \boldsymbol{h}(\boldsymbol{x}_i)^T\Big)^T \\
&\qquad \text{(from equation 3.42 and 3.38)} \\
&= 0 \qquad (3.43)
\end{aligned}$$

Chapter 4

GPEs for multivariate output

4.1 Introduction

In this Chapter we first discuss types of multivariate output and common approaches of emulating these multiple outputs. We split the remainder of the Chapter to the two types of outputs discussed in Section 4.2; field type outputs (same variable varying across time or space) and multiple-type outputs (multiple types of variables, representing varied quantities).

Under field type outputs in Subsection 4.3.2 we review an approach to use principal component analysis (PCA) to emulate multiple field type outputs. In Subsection 4.3.3 we describe work presented by Conti and O‘Hagan (2010) on separable GPEs. This appeals as it is a simple extension to the univariate emulator theory presented in Chapter 3. However, it assumes that all the outputs have the same correlation lengths, which may be too restrictive in some cases. We compare these approaches against emulating each output independently in a case study.

Under multiple-type output, in Section 4.4, we review the linear model of coregionalisation (LMC) method of handling multiple-type output. We assume that the multiple outputs represent a variety of different quantities, hence it would not be sensible to impose a parametric form for the between output dependencies, (Fricker et al., 2013). We review work carried out by Fricker et al. (2013) on the LMC model, which is presented as a more flexible method of emulating multiple output as it allows for variation in correlation lengths amongst outputs. We compare this approach to the separable GPE and to emulating each output independently in a case study. We conclude the Chapter by discussing the approaches and best practices to use when dealing with multiple output simulators.

4.2 Multivariate output and common emulation techniques

Gaussian process emulators have become popular in the statistical analysis of deterministic simulators, particularly those that require a long time to run each observation. In practice, simulators often have more than one output variable. For a nonexpert, a common practice to model multivariate output using GPE is to apply univariate emulators for each output independently. In a situation where the simulator, and essentially the emulator, has to be run for many input values for multiple outputs, the computational time taken to generate results can be very high and the task can be laborious. More importantly, it is possible that multiple outputs from the simulator are correlated in some way, and applying univariate emulators to each output independently may result in loss of information. Some vital information about between output correlations may also be lost due to modelling the prior beliefs independently.

Fricker et al. (2013) categorise multivariate output into two classes. One class is where multiple outputs represent a certain value at various locations in the field, or at various timestamps. Often the name of the output is like an index to specify the particular location or time. This is referred to as *field output*. An example of this kind of output using SWAN as the simulator would be where there is one offshore point, but we are interested in transformed wave height at multiple near shore locations. The number of near shore locations could range between two to a couple hundred locations. It would be really time consuming to fit a GPE for each location. Moreover, it is unlikely that the near shore locations that are close together will have that much variance in output. In other words, these multiple field outputs tend to be highly correlated with each other.

One approach to emulating multiple field type output is by using dimension reduction techniques. There exists numerous methods for both linear and non-linear dimension reduction. Maaten et al. (2009) believe that PCA is the most popular linear dimension reduction technique. They compare numerous non-linear techniques using the PCA as a benchmark and conclude that these techniques do not outperform the traditional PCA yet. Therefore, and similar to Higdon et al. (2008), in this thesis, we choose to focus on emulating a simulator with high dimensional field output with principal component analysis to reduce the dimensionality.

We use PCA to condense the information from a large number of variables into a smaller number of principal components. These principal components are independent from each other and thus we can safely use multiple univariate GPEs independently to predict the outcome of the simulator at other input variables. These predictions will be in terms of the principal components and thus we will need to convert them back to the form of the simulator output.

Chen et al. (2011) also use a similar approach to emulate their complex simulator with high dimensional time-space-dependent output. They demonstrate that the use of PCA in conjunction with the GPE was effective and efficient. Other examples include work done by Wilkinson (2010), Bounceur et al. (2015), Gómez-Dans et al. (2016) and Camus et al. (2013). Xing et al. (2014) use kernel PCA, a non-linear dimension reduction technique as they believe that the linear representation of the high dimensional output is not a faithful representation of their data. Liu and Guillas (2016) use gradient-based kernel dimension reduction technique in the input space, but this could easily be applied in the output space as well.

Another approach that we consider in this thesis, is the separable GPE. Conti and O‘Hagan (2010) propose this emulator as a simple extension of the univariate emulator discussed in Chapter 3 with a strong assumption that the outputs have the same correlation lengths. They compare this GPE to independent univariate GPE's and to a time input emulator (where time is included as an extra input). They conclude that the separable GPE is the simplest in terms of computation. However in terms of flexibility it is more restrictive than multiple independent univariate GPE. This is because the separable GPE shares a common set of correlation length parameters amongst all the output whereas the latter allows each output to be fitted with unique parameters. This restricted flexibility can be reasonable in some cases (where changes in input settings have similar effects to all outputs) and can be efficient.

Kennedy and O'Hagan (2001) consider emulating field output by including the index as an input parameter, and then applying standard univariate GPEs, however this becomes impractical for emulating whole maps (very high dimension) of output. Rougier (2008) demonstrates yet another type of multivariate GPE which he calls the outer-product emulators. He states that the separable GPE is a special case of the outer-product emulator. He also claims that the outer-product emulator is faster to compute, is more memory-efficient and more stable. Boukouvalas and Cornford (2008) consider alternative ways to reduce dimensionality and other structural forms of GPEs that can be used. However these approaches are beyond the scope of this thesis.

The second class of multivariate output according to Fricker et al. (2013) “arises from simulators that simulate different types of quantities simultaneously, and the index of the output is merely a label." This is referred to as *multiple-type output*. Often each output may also vary in units. We have already seen an example of this type of output with the SWAN model. This model outputs the properties of the waves at near shore locations, which include the wave height, wave period and wave direction. We note that these output variables all represent different quantities and have different units. Multiple-type output is often not highly correlated and not high dimensional thus dimension reduction techniques are not effective. Moreover, because the outputs represent a variety of quantities, a shared correlation length parameter would be too restrictive and hence separable GPE's are also not appropriate.

Fricker et al. (2013) propose a more flexible approach that allows each output to have different correlation length parameters. They propose two nonseparable GPE approaches; one approach is using convolution methods. We do not discuss this approach in this thesis, but some references include Ver Hoef and Barry (1998); Boyle and Frean (2005); Álvarez and Lawrence (2009); Higdon (2002). Another approach Fricker et al. (2013) suggests is the LMC model, which they extend from geostatistics (Goulard and Voltz, 1992; Journel and Huijbregts, 1978; Wackernagel, 1995; Gelfand et al., 2005). This method allows the variations in correlation length parameters of the multiple outputs to occur on different scales as output processes are constructed as linear combinations of independent univariate GPEs.

4.3 Field type outputs

In general, simulator output can have many dimensions. The higher the dimensions of the output, the harder it is and the longer it takes to emulate the simulator. Two methodologies are presented here to emulate multiple field type outputs using GPE. We first introduce the PCA methodology followed by the separable GPE. In this section we assume that the outputs that we are dealing with are field type and therefore the conclusions drawn in this section may not apply for other output types.

4.3.1 Toy example: one input, two field output simulator

We use a toy example in this section to illustrate the mathematical detail as we present the two methodologies. Here, we describe a toy simulator with $p = 1$ input, and $q = 2$ outputs. We assume this simulator is computationally expensive and so we can only evaluate it at a limited number of values. Let's assume in this example, a simulator is represented by $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{x})$, where $\boldsymbol{y}$ can be written as $\boldsymbol{y} = (y_1, y_2) = \big(\eta_1(\boldsymbol{x}), \eta_2(\boldsymbol{x})\big)$. Suppose we run the simulator seven times for input values $(\boldsymbol{x}_1, ..., \boldsymbol{x}_7)$, which we denote X, which are selected randomly over the input space. The simulator output for these values is denoted $Y = \boldsymbol{\eta}(X) = (\boldsymbol{y}_1, ..., \boldsymbol{y}_7)$, where $\boldsymbol{y}_i = (y_{i,1}, y_{i,2})$. The values for X and Y are given as below:

$$X = \begin{pmatrix} -4 \\ -2.5 \\ -1 \\ 1 \\ 2.25 \\ 3 \\ 4 \end{pmatrix}, \; Y = \begin{pmatrix} -4 & -3.7 \\ -2.5 & -2.75 \\ -1 & -1.3 \\ 1 & 0.95 \\ 2.25 & 2.5 \\ 3 & 3.2 \\ 4 & 3.8 \end{pmatrix}.$$

and we wish to predict the simulator at four input values: (-3, -2, 0, 2).

Figure 4.1 shows the outputs of the toy simulator plotted against the input values that range over the input space [-4,4]. The black points represent the training runs, and the red crosses are the events where we wish to predict the simulator output. The red points illustrate the true output at the prediction points.

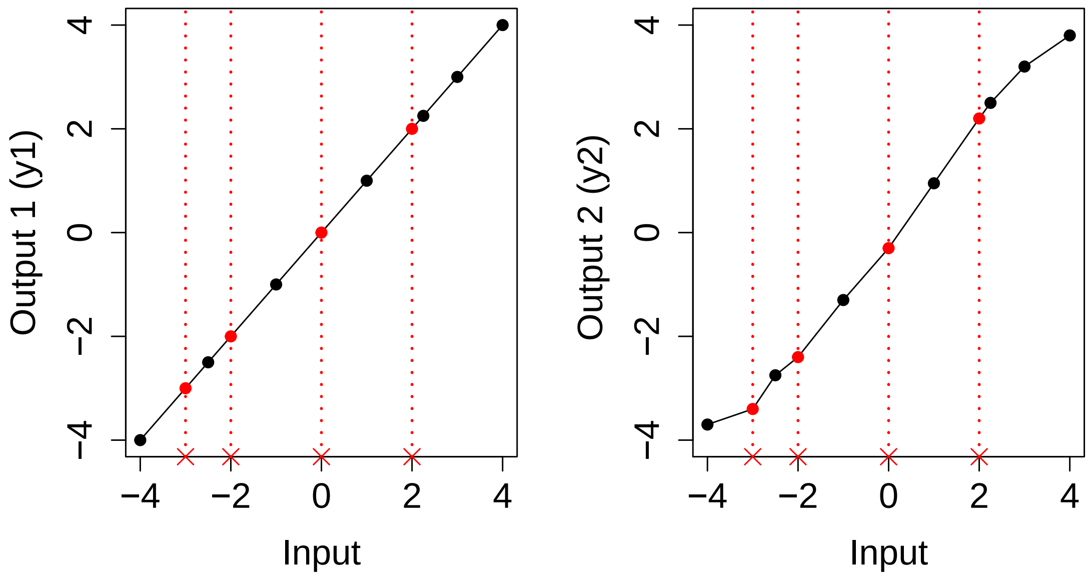

Figure 4.1: Multivariate (field-type output) toy simulator. The black points highlight the training runs, the red crosses are the events which we wish to predict at and the red points are true simulator output for these events.

4.3.2 Principal Components GPE (PC-GPE)

One way of emulating multiple field type outputs is to use PCA. We use PCA to transform the data so that it is expressed as a set of orthogonal coordinates such that the sample variances of the data with respect to these coordinates are in decreasing order of magnitude. In other words, the projection of the points onto the first principal component has maximal variance among all such linear projections. The projection of the points onto the second principal component has orthogonal projection to that of the first principal component, and so on. We can reduce the dimensions of the transformed data set by ignoring some of the latter components, and then performing emulation with the reduced transformed dataset as the output variables.

Dimension reduction techniques are more appropriate for emulating multiple field output as with high correlation and high dimensional output, the variation in the outputs can be represented by only a few components.

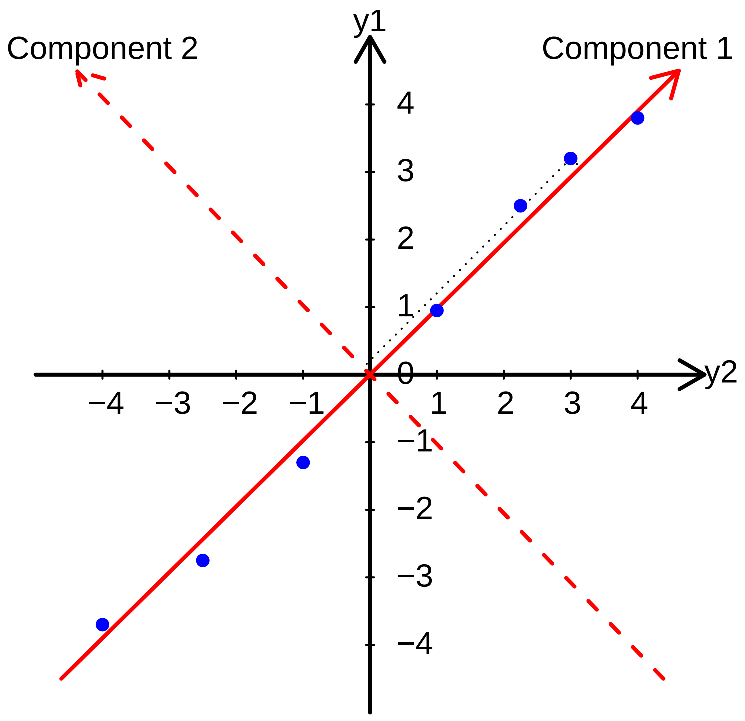

Figure 4.2: Multivariate (field-type output) toy simulator plotted with the principal components (component 1 and component 2) of the data.

Figure 4.2, shows a scatter plot of the simulator output, Y (blue points). The arrow labelled “Component 1” represents the first principal component and the arrow labelled “Component 2” represents the second principal component. These two vectors can be interpreted as the “new” axes over which the transformed set of orthogonal coordinates have been plotted, as shown in Figure 4.3. It is evident from this graph that the data does not vary much in the direction of Component 2 .

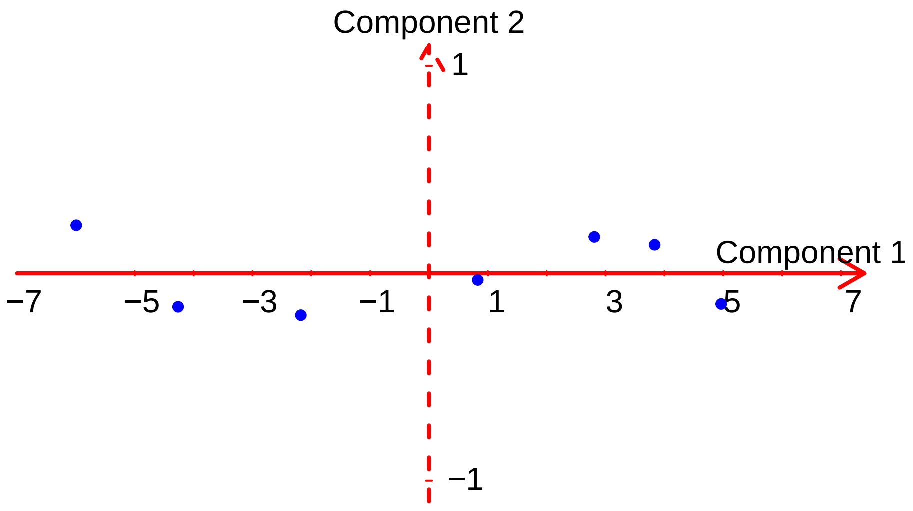

Figure 4.3: Multivariate (field-type output) simulator output plotted on transformed axis - the principal components.

The mean value of each output in Y and the variance of Y, denoted $\bar{Y}$ and S respectively, are as follows:

$$\bar{Y} = \begin{pmatrix} \sum_{i=1}^{7} \frac{y_{i,1}}{n} & \sum_{i=1}^{7} \frac{y_{i,2}}{n} \end{pmatrix} = \begin{pmatrix} 0.393 & 0.386 \end{pmatrix} \tag{4.1}$$

$$S = \begin{pmatrix} 8.87 & 8.88 \\ 8.88 & 8.96 \end{pmatrix}. \tag{4.2}$$

Let λ_1 and λ_2 denote the eigenvalues of S in descending order of magnitude, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ denote the corresponding normalised eigenvectors and B denote the matrix of all the eigenvectors. The eigenvalues and eigenvectors are computed and given as below:

$$\lambda_1 = 17.8 \tag{4.3}$$

$$\lambda_2 = 0.03 \tag{4.4}$$

$$B = \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 \end{pmatrix} \tag{4.5}$$

$$= \begin{pmatrix} 0.705 & -0.709 \\ 0.709 & 0.705 \end{pmatrix}. \tag{4.6}$$

Define $\boldsymbol{\zeta}(\boldsymbol{x}) = \begin{pmatrix} \zeta_1(\boldsymbol{x}) & \zeta_2(\boldsymbol{x}) \end{pmatrix}$ to represent a transformation of $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{x})$ using PCA, where

$$\boldsymbol{\zeta}(\boldsymbol{x}) = \left(\boldsymbol{\eta}(\boldsymbol{x}) - \bar{Y}\right) B. \tag{4.7}$$

Then $\boldsymbol{\zeta}(\boldsymbol{x})$ gives the coordinates of $\boldsymbol{y}$ on the transformed axes and

$$\boldsymbol{\eta}(\boldsymbol{x}) = \bar{Y} + \boldsymbol{\zeta}(\boldsymbol{x}) B^T. \tag{4.8}$$

Define $\boldsymbol{\zeta}(X) = \begin{pmatrix} \zeta_1(X) & \zeta_2(X) \end{pmatrix}$ to represent the transformation of Y as in Equation 4.7. $\boldsymbol{\zeta}(X)$ is calculated as follows:

$$\boldsymbol{\zeta}(X) = \left[\begin{pmatrix} y_1 & y_2 \end{pmatrix} - \bar{Y} \right] \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 \end{pmatrix}$$
$$= \begin{pmatrix} -5.99 & 0.23 \\ -4.26 & -0.16 \\ -2.18 & -0.2 \\ 0.83 & -0.03 \\ 2.81 & 0.18 \\ 3.83 & 0.14 \\ 4.96 & -0.15 \end{pmatrix}.$$

The blue points in Figure 4.3 have coordinates given by the values in $\boldsymbol{\zeta}(X)$. It is clear to see from the values of $\boldsymbol{\zeta}(X)$ that $\zeta_2(X) \approx 0 \; \forall \boldsymbol{x}$, so we use the approximation

$$\boldsymbol{\eta}(\boldsymbol{x}) \approx \bar{Y} + \begin{pmatrix} \zeta_1(\boldsymbol{x}) & 0 \end{pmatrix} B^T. \tag{4.9}$$

Effectively, we have reduced the dimension of the data, as now we only need to emulate $\zeta_1(\boldsymbol{x})$, which is much easier to emulate using a univariate emulator. If there were more components at this stage, then each of the components would be emulated independently using univariate GPEs. The principal components are independent from each other thus independently emulating the multiple principal components does not risk loss of information.

The ratio $\frac{\lambda_j}{\lambda_1+\lambda_2}$ indicates the proportion of the variance of the data represented by the j-th component (j = 1 or 2). Component 1 represents 99.82% of the variance of Y and Component 2 only represents 0.18% of the variance. These ratios can be used to evaluate the importance of each component.

Next, we fit an emulator using training inputs X and the values for the first component $\zeta_1(X)$. We then use this to predict the output values of $\zeta_1(\tilde{X})$,where $\tilde{X} = (\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_5)^T = \begin{pmatrix} -3 & -2 & 0 & 2 \end{pmatrix}^T$. Let $E(\zeta_1(\tilde{X})|\zeta_1(X))$ denote the posterior mean of the emulator, which is the predicted output of $\zeta_1(\tilde{X})$. Similarly, let $V(\zeta_1(\tilde{X})|\zeta_1(X))$ denote the posterior variance covariance matrix of the emulator prediction.

The values for the emulator prediction $E(\zeta_1(\tilde{X})|\zeta_1(X))$ and the true output, $\zeta_1(\tilde{X})$ are given below. The calculation for the RMSE is also shown.

$$E(\zeta_1(\tilde{X})|\zeta_1(X)) = \begin{pmatrix} -4.88 & -3.6 & -0.72 & 2.43 \end{pmatrix}^T, \tag{4.10}$$

$$\zeta_1(\tilde{X}) = \begin{pmatrix} -5.08 & -3.66 & -0.76 & 2.42 \end{pmatrix}^T, \tag{4.11}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^4 \left(\zeta_1(\tilde{\boldsymbol{x}}_i) - E\big(\zeta_1(\tilde{\boldsymbol{x}}_i)|\zeta_1(\boldsymbol{x}_i)\big)\right)^2}{4}} \tag{4.12}$$

$$= 0.17. \tag{4.13}$$

The RMSE suggests a good level of accuracy given that the data ranges from -5.08 to 2.42. The NRMSE is 0.03.

In order to get $E(\zeta_1(\tilde{X})|\zeta_1(X))$ in terms of y_1 and y_2, which we denote as $\tilde{Y}^*$, we apply Equation 4.9 as follows:

$$\tilde{Y}^* = \begin{pmatrix} 0.393 & 0.386 \end{pmatrix} + \begin{pmatrix} -4.88 & 0 \\ -3.6 & 0 \\ -0.72 & 0 \\ 2.43 & 0 \end{pmatrix} \begin{pmatrix} 0.705 & 0.709 \\ -0.709 & 0.705 \end{pmatrix}$$

$$= \begin{pmatrix} -3.05 & -3.07 \\ -2.14 & -2.16 \\ -0.12 & -0.13 \\ 2.11 & 2.11 \end{pmatrix}.$$

The true output of the simulator evaluated at $\tilde{X}$ is

$$\boldsymbol{\eta}(\tilde{X}) = \begin{pmatrix} -3 & -3.4 \\ -2 & -2.4 \\ 0 & -0.3 \\ 2 & 2.2 \end{pmatrix}.$$

Again we calculate the RMSE as follows:

$$\sqrt{\frac{\sum_{i=1}^{4}\left(\tilde{\boldsymbol{y}}_i - \tilde{\boldsymbol{\eta}}(\tilde{\boldsymbol{x}}_i)^2\right)}{4}} = 0.1.$$

The range of the data here is between -3.4 and 2.2, hence this value suggests a good level of accuracy. The NRMSE is 0.01. Note that, we have lost some information about the simulator by not emulating component 2, however, it was a compromise to having to emulate only one component. We have to keep that in mind while judging the RMSE.

Similarly, for each new input $\tilde{\boldsymbol{x}}_i$, the posterior variance $V(\zeta(\tilde{\boldsymbol{x}}_i)|\zeta_1(X))$ can be written in terms of $y_1,\ y_2$, denoted, $\tilde{V}_i$, using the following formula where the result is a variance covariance matrix of y_1 and y_2:

$$\tilde{V}_i = \boldsymbol{b}_1^T V(\zeta(\tilde{\boldsymbol{x}}_i)|\zeta_1(X))\boldsymbol{b}_1^T \tag{4.14}$$

We calculate $\tilde{V}_1$ for the new input $\boldsymbol{x}_1$ as follows:

$$\begin{aligned}\tilde{V}_1 &= \begin{pmatrix} 0.705441 \\ 0.708768 \end{pmatrix} \begin{pmatrix} 0.000485 \end{pmatrix} \begin{pmatrix} 0.705441 & 0.708768 \end{pmatrix} \\ &= \begin{pmatrix} 2e-04 & 2e-04 \\ 2e-04 & 2e-04 \end{pmatrix}.\end{aligned}$$

The diagonal elements of $\tilde{V}_1$ represent the uncertainty gauged by the emulator for its prediction of $\eta_1(\boldsymbol{x}_1)$ and $\eta_2(\boldsymbol{x}_1)$.

Figure 4.4 shows the result of the GPE predictions on a plot. The red points here represent the GPE predictions at the input values marked with crosses. We have not plotted the error bars as the uncertainty around these predictions is calculated to be minimal. It is evident that the red dots lie very close to the true simulator output (black line).

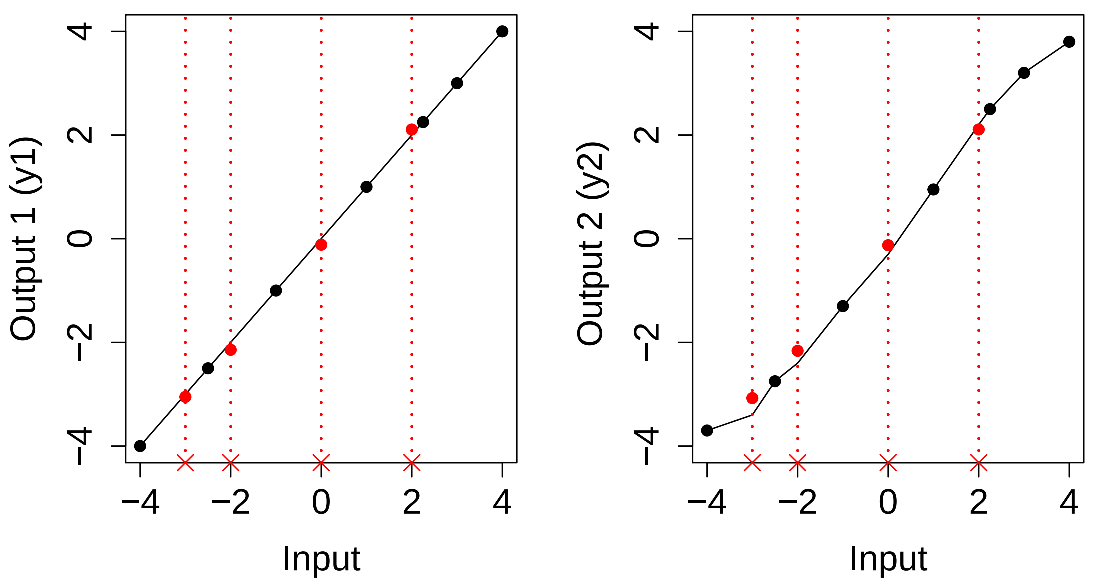

Figure 4.4: Multivariate (field-type output) toy GPE. The black points highlight the training runs, the red crosses are the events we predicted using the GPE and the red points are the predicted simulator output.

In general, for q dimensional simulators, B is the matrix of eigenvectors $\boldsymbol{b}_1, ..., \boldsymbol{b}_q$, that correspond to eigenvalues $\lambda_1, ..., \lambda_q$. Let $\boldsymbol{\zeta}(X) = \begin{pmatrix} \zeta_1(X) & \dots & \zeta_q(X) \end{pmatrix}$ represent a transformation of Y, given by Equation 4.8. The ratio $\frac{\sum_i^c \lambda_i}{\sum_i^q \lambda_i}$ indicates the total proportion of the variance of the data represented by the first c components. Choose c such that components greater then c represent minimal information of the data that can be ignored. Set $\zeta_j(X) = \mathbf{0}\ \forall j > c$.

Next, we fit a univariate GPE with inputs X to each of the outputs $\zeta_1(X), ..., \zeta_c(X)$ and use it to predict the output values of $\zeta_1(\tilde{X}), ..., \zeta_c(\tilde{X})$ for inputs $\tilde{X}$. To transform the emulator predictions of the components back in terms of the original data, we use a generalisation of Equation 4.9

$$\boldsymbol{\eta}(\boldsymbol{x}) \approx \bar{Y} + \begin{pmatrix} \zeta_1(\tilde{X}) & \dots & \zeta_c(\tilde{X}) & 0 & \dots & 0 \end{pmatrix} B^T. \tag{4.15}$$

Similarly, to transform the posterior variance for each new input, $V(\boldsymbol{\zeta}(\tilde{\boldsymbol{x}}_i)|\zeta_1(X))$, (for convenience we use the shorthand $V(\zeta(\tilde{\boldsymbol{x}}_i))$, we use a generalisation of Equation 4.14

$$\tilde{V}_i = \begin{pmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_c \end{pmatrix} \begin{pmatrix} V(\zeta_1(\tilde{\boldsymbol{x}}_i)) & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & V(\zeta_c(\tilde{\boldsymbol{x}}_i)) \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_1 & \dots & \boldsymbol{b}_c \end{pmatrix}.$$

Generally, if the multiple outputs are highly correlated then PCA can be used to significantly lower the number of dimensions with minimal loss of accuracy.

4.3.3 Separable GPE

Another approach to emulating multiple field type output is using a separable GPE. This type of GPE is a simple extension of the univariate GPE. This approach is particularly useful when dimensionality is not a problem, i.e. it is not necessary to reduce the dimensions of the simulator. It is also useful when the simulator outputs are highly correlated. This method shares a common set of correlation length parameters across all outputs and therefore it efficiently fits a single GPE for all the outputs. However, if the simulator output is not well correlated, a common set of correlation length parameters may be an unrealistic assumption. The rest of this Subsection is based on Conti and O'hagan (2010).

Following a similar setup as used to describe the univariate GPE approach in Chapter 3, we derive a probability distribution of $\boldsymbol{\eta}(\boldsymbol{x}_{n+1}), ..., \boldsymbol{\eta}(\boldsymbol{x}_N))$ given the training runs, Y by modelling $\boldsymbol{\eta}(.)$ as a q-dimensional Gaussian Process

$$\boldsymbol{\eta}(\boldsymbol{x})|B, \Sigma, \boldsymbol{r} \sim GP(\boldsymbol{m}(.), \Sigma c(.,.))$$

conditional on hyper- parameters B, Σ and $\boldsymbol{r}$. These hyper-parameters are unknown parameters in the mean and covariance functions which are described in a later subsection.

Prior beliefs of $\boldsymbol{\eta}(\boldsymbol{x})$

We can describe our prior mean and covariance for $\boldsymbol{\eta}(\boldsymbol{x})$ as:

$$E[\boldsymbol{\eta}(\boldsymbol{x})|B, \Sigma, \boldsymbol{r}] = \boldsymbol{m}(\boldsymbol{x}) \tag{4.16}$$

$$Cov[\boldsymbol{\eta}(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x'})|B, \Sigma, \boldsymbol{r}] = \Sigma c(\boldsymbol{x}, \boldsymbol{x'}) \tag{4.17}$$

where $c(.,.)$ is a positive definite function that provides spatial correlation over the inputs such that $c(\boldsymbol{x}, \boldsymbol{x}) = 1 \; \forall \boldsymbol{x}$ and Σ which should also be a positive definite matrix represents the between outputs covariance at any given input. This structure assumes the spatial correlation and the between output covariances can be separated. That is why this approach is called the separable GPE.

Similar to the univariate GPE we model the mean function using a generic specification which will always make it possible to describe our beliefs about the simulator. We choose the correlation function to be the Gaussian correlation function expressed in multivariate notation as follows:

$$\boldsymbol{m}(\boldsymbol{x}) = B^T \boldsymbol{h}(\boldsymbol{x})^T, \tag{4.18}$$

$$c(\boldsymbol{x}, \boldsymbol{x'}) = exp[-(\boldsymbol{x} - \boldsymbol{x'})^T R(\boldsymbol{x} - \boldsymbol{x'})]. \tag{4.19}$$

Here, the vector $\boldsymbol{h}(.)$ consists of m known regression functions $h_1(\boldsymbol{x}), ..., h_m(\boldsymbol{x})$ shared by each individual function $\boldsymbol{\eta}_j(.)$ where $j = 1, ..., q$. The matrix $B = [\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_q]$ is a $m \times q$ matrix of regression coefficients and R is a diagonal matrix of p correlation length parameters such that $R = diag\{\boldsymbol{r}\} = diag\{r_i^{-2}\}$. (Note that Conti and O'Hagan (2010) refer to R as the matrix of roughness parameters which is the squared inverse of the correlation length parameter). The form of the correlation function in (4.19) is a generalisation of the univariate Gaussian correlation function and assumes that $\boldsymbol{\eta}_j(.)$ are analytical functions, which may not be a realistic assumption for some simulators. However, the Gaussian correlation function is convenient for illustrating the theory due to its mathematical tractability.

In summary, the prior distribution for $\boldsymbol{\eta}(\boldsymbol{x})$ is given by

$$\boldsymbol{\eta}(\boldsymbol{x})|B, \Sigma, \boldsymbol{r} \sim N(B^T \boldsymbol{h}(\boldsymbol{x})^T, \Sigma c(.,.)) \tag{4.20}$$

The prior mean and variance for $\boldsymbol{\eta}(\boldsymbol{x})$ depend on the hyper-parameters B, Σ, and $\boldsymbol{r}$. Similar to the prior for the hyperparameters of the univariate GPE prior, a weak prior for the prior distribution for B and Σ independent of $\boldsymbol{r}$ is used:

$$p(B, \Sigma|\boldsymbol{r}) \propto |\Sigma|^{-\frac{q+1}{2}}. \tag{4.21}$$

Analogous to the estimation of $\boldsymbol{\delta}$ in the univariate GPE here we also estimate the value of $\boldsymbol{r}$ by maximum likelihood estimation instead of a full proper Bayesian inference which we discuss in a latter Subsection.

Deriving the posterior distribution: $\boldsymbol{\eta}(\boldsymbol{x})|\boldsymbol{r}, Y$

Next, we observe $\boldsymbol{\eta}(X)$ at the training inputs $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, which produces simulator output data

$$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) = Y = \boldsymbol{\eta}(X) = (\boldsymbol{\eta}(\boldsymbol{x}_1)), \ldots, \boldsymbol{\eta}(\boldsymbol{x}_n)).$$

We then update the prior distribution of $\boldsymbol{\eta}(.)$ using the partitioning property of multivariate normal distributions. Given that the joint matrix-Normal distribution of Y conditional on the hyper-parameters B, Σ and $\boldsymbol{r}$ is:

$$Y|B, \Sigma, \boldsymbol{r} \sim N_{n,q}(HB, A, \Sigma) \tag{4.22}$$

where $H^T = [\boldsymbol{h}(\boldsymbol{x}_1), ..., \boldsymbol{h}(\boldsymbol{x}_n)] \in \mathbb{R}_{m,n}$ and $A = [c(\boldsymbol{x}, \boldsymbol{x'})] \in \mathbb{R}^+_{n,n}$, and equation (4.20), we can apply the partitioning property of multivariate normal distribution to get

$$\boldsymbol{\eta}(\boldsymbol{x})|B, \Sigma, \boldsymbol{r}, Y \sim GP(\boldsymbol{m^*}(.), \Sigma c^*(., .)), \tag{4.23}$$

where,

$$\begin{aligned}
\boldsymbol{m^*}(\boldsymbol{x}) &= B^T \boldsymbol{h}(\boldsymbol{x})^T + (Y - HB)^T A^{-1} \boldsymbol{t}(\boldsymbol{x}), \\
c^*(\boldsymbol{x}, \boldsymbol{x'}) &= c(\boldsymbol{x}, \boldsymbol{x'}) - \boldsymbol{t}(\boldsymbol{x})^T A^{-1} \boldsymbol{t}(\boldsymbol{x'}), \\
\boldsymbol{t}(\boldsymbol{x})^T &= (c(\boldsymbol{x}, \boldsymbol{x}_1), ..., c(\boldsymbol{x}, \boldsymbol{x}_n)).
\end{aligned}$$

To derive the posterior distribution of $\boldsymbol{\eta}(\boldsymbol{x})$ conditional only on $\boldsymbol{r}$, we integrate (4.23) with respect to the posterior distribution of B and Σ. We first integrate out B from the product of (4.23), (4.22) and (4.28) which gives:

$$\boldsymbol{\eta}(\boldsymbol{x})|\Sigma, \boldsymbol{r}, Y \sim GP(\boldsymbol{m^{**}}(.), \Sigma c^{**}(., .)), \tag{4.24}$$

where,

$$\begin{aligned}
\boldsymbol{m^{**}}(\boldsymbol{x}) &= \hat{B}^T \boldsymbol{h}(\boldsymbol{x})^T + (Y - H\hat{B})^T A^{-1} \boldsymbol{t}(\boldsymbol{x}), \\
c^{**}(\boldsymbol{x}, \boldsymbol{x'}) &= c^*(\boldsymbol{x}, \boldsymbol{x'}) + \\
&\qquad [\boldsymbol{h}(\boldsymbol{x}) - H^T A^{-1} \boldsymbol{t}(\boldsymbol{x})].(H^T A^{-1} H)^{-1} [\boldsymbol{h}(\boldsymbol{x'}) - H^T A^{-1} \boldsymbol{t}(\boldsymbol{x'})]
\end{aligned}$$

and

$$\hat{B} = (H^T A^{-1} H)^{-1} H^T A^{-1} Y, \tag{4.25}$$

is the generalised least squares (GLS) estimator of B. The resulting posteriors are proper as long as $n \geq m + q$.

Similarly, we integrate out Σ and obtain the following q-variate T process conditional on $\boldsymbol{r}$

$$\boldsymbol{\eta}(.)|\boldsymbol{r}, Y \sim \mathcal{T}(\boldsymbol{m}^{**}(.),\ \hat{\Sigma} c^{**}(.,.);\ n - m), \tag{4.26}$$

where $\mathcal{T}$ is such that the distribution of an arbitrary collection of vectors is matrix variate T with $n - m$ degrees of freedom, and

$$\hat{\Sigma} = (n - m)^{-1}(Y - H\hat{B})^T A^{-1}(Y - H\hat{B}) \tag{4.27}$$

which denotes the GLS estimator of Σ.

Estimating the correlation length parameter $\boldsymbol{r}$

We cannot derive analytically an estimate for the correlation length parameter and carrying out a full MCMC based marginalisation of (4.26) with respect to unknown smoothness in R would be too computationally expensive. Similar to the univariate GPE, the simplest option is to plug in estimates of $\boldsymbol{r} = (r_1^{-2}, ..., r_p^{-2})$. Prior knowledge or experience of the smoothness of the simulator may suggest a value. Alternatively, we could estimate $\boldsymbol{r}$ from the data by maximising the likelihood function as derived below.

The prior distribution for the hyper-parameters has the form:

$$p(B, \Sigma, \boldsymbol{r}) \propto p_{\boldsymbol{R}}(\boldsymbol{r}) |\Sigma|^{-\frac{q+1}{2}}. \tag{4.28}$$

Using Bayes' theorem to combine this prior (4.28) with the likelihood function 4.22, the full posterior is

$$p(B, \Sigma, \boldsymbol{r}|Y) \propto p_{\boldsymbol{R}}(\boldsymbol{r})|A|^{-\frac{q}{2}}|\Sigma|^{-\frac{n-m+q+1}{2}} \\ \exp\Big[-\frac{1}{2}\Big(Tr[Y^T GY\Sigma^{-1}] + Tr[(B-\hat{B})^T H^T A^{-1} H(B-\hat{B})\Sigma^{-1}]\Big)\Big].$$

Integrating out the hyper-parameter matrices B and Σ gives:

$$p(\boldsymbol{r}|Y) \propto p_{\boldsymbol{R}}(\boldsymbol{r})|A|^{-\frac{q}{2}}|H^T A^{-1} H|^{-\frac{q}{2}}|Y^T GY|^{-\frac{n-m}{2}}, \tag{4.29}$$

where, $G = A^{-1} - A^{-1}H(H^T A^{-1} H)^{-1} H^T A^{-1}$. The estimates of $\boldsymbol{r}$ can then be obtained my maximising (4.29).

Toy GPE

For our toy example, we assume that this multivariate simulator has a linear mean function. We write this as:

$$\boldsymbol{m}(\boldsymbol{x}) = \begin{pmatrix} 1 & -4 \\ 1 & -2.5 \\ 1 & -1 \\ 1 & 1 \\ 1 & 2.25 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} \beta_{1,1} & \beta_{2,1} \\ \beta_{1,2} & \beta_{2,2} \end{pmatrix}.$$

We also assume that this simulator has a Gaussian correlation function (4.19). We estimate the value of r as 0.66 by maximising the likelihood function (4.29). Using this value we can calculate $\hat{B}$ and $\hat{\Sigma}$ with equations (4.25) and (4.27) respectively. We note that both these calculations require the A matrix. We show the calculations for one element of the matrix, $A_{1,2} = c(x_1, x_2)$. We note that $p = 1$ and substituting the relevant values we get

$$\begin{aligned} c(x_1, x_2) &= \exp\left[-\left(\frac{-4 - -2.5}{0.66}\right)^2\right]. \\ &= \exp\left[-0.6\right] \\ &= 0.55. \end{aligned}$$

We use the same approach to calculate every element in the A matrix. Below we show the values of $\hat{B}$ and $\hat{\Sigma}$:

$$\hat{B} = \begin{pmatrix} 0 & -0.05 \\ 1 & 0.94 \end{pmatrix}, \hat{\Sigma} = \begin{pmatrix} 0 & 0 \\ 0 & 0.3 \end{pmatrix}. \tag{4.30}$$

We can then derive the GPE predictions and associated uncertainty around these predictions as

$$m^{**}\begin{pmatrix} -3 \\ -2 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} -3 & -3.09 \\ -2 & -2.33 \\ 0 & -0.21 \\ 2 & 2.21 \end{pmatrix} \tag{4.31}$$

$$\hat{\Sigma}c^{**}\begin{pmatrix} -3 & -3 \\ -2 & -2 \\ 0 & 0 \\ 2 & 2 \end{pmatrix}, = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0059 & -0.0049 & 0.0049 & -3e-04 \\ 0 & 0 & 0 & 0 & -0.0049 & 0.0046 & -0.0057 & 4e-04 \\ 0 & 0 & 0 & 0 & 0.0049 & -0.0057 & 0.0108 & -0.0011 \\ 0 & 0 & 0 & 0 & -3e-04 & 4e-04 & -0.0011 & 2e-04 \end{pmatrix}. \tag{4.32}$$

Note that the posterior variance is a Kronecker product of $\hat{\Sigma}$ - the between outputs correlation, and $c^{**}(.,.)$ the covariance amongst the inputs that we are interested in predicting.

The RMSE for the separable GPE is 0.08 and the NRMSE is 0.01. This RMSE and NRMSE is lower than that of the PC-GPE, remember however that we loss information by reducing the dimension for the PC-GPE. We choose not to plot the posterior toy GPE because the predictions are very close to the truth and the uncertainty bands around the predictions are extremely small that they would not be visible on the plot. In the next Subsection we compare the two approaches in a case study using the SWAN simulator example.

4.3.4 Case Study - Farasin Islands, Red Sea

In this case study we aim to compare three ways of emulating multiple field type output: fitting a univariate GPE independently to each of the simulator output (IND), fitting a univariate GPE independently to a reduced number of principal components of the simulator output (PC-GPE) and fitting a single separable GPE to all of the simulator output (SEP). We explore these methods using the context of case study 3 as described in Chapter 2.2.3.

As a summary, the SWAN model used in this case study has $p = 7$ input variables which we denote $\boldsymbol{x} = (x_1, ..., x_7)$. The output variable considered is the significant wave height at $q = 189$ near shore points denoted as $\boldsymbol{y}_1, ..., \boldsymbol{y}_{189}$. This is the field output class of multivariate data. We use $n = 250$ runs as training data and a further 250 runs as validation data to compare how accurate the predictions are.

Figure 4.5 shows a pair wise plot of the 500 events of input data. The 250 runs selected as training data are highlighted in black. This plot shows that the training data effectively represents the entire input dataset. Figure 4.6 shows correlation matrix of all the output variables. We can see that the majority of the variables are

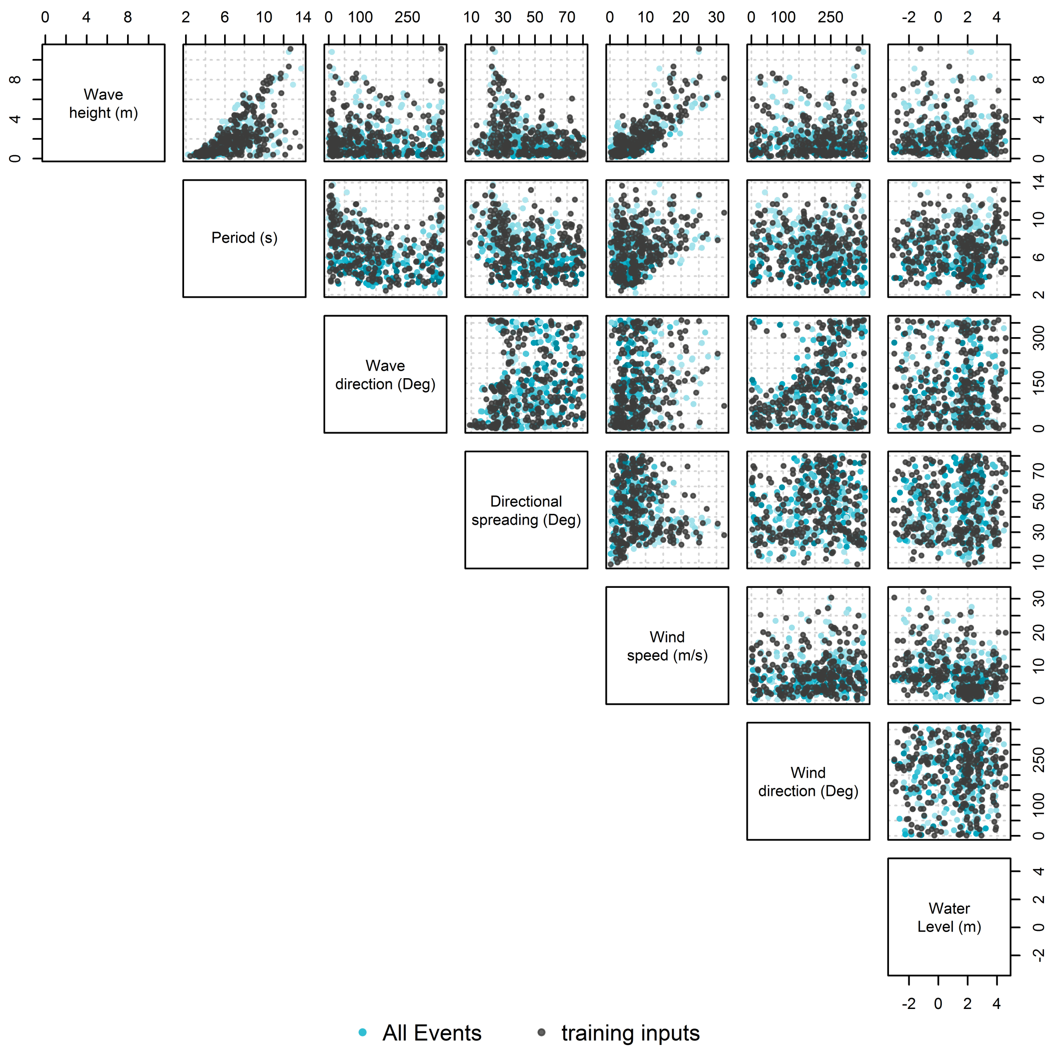

Figure 4.5: Design selected as training inputs using the MDA method. The black dots represent the design and the grey dots in the background represent all the events.

highly correlated.

Independent method

For the IND method, we fit a univariate GPE (Chapter 3) to each SWAN output (189). We use the same input values for all the GPEs. We use a linear prior mean and a Gaussian correlation function to describe our prior beliefs of the simulator for

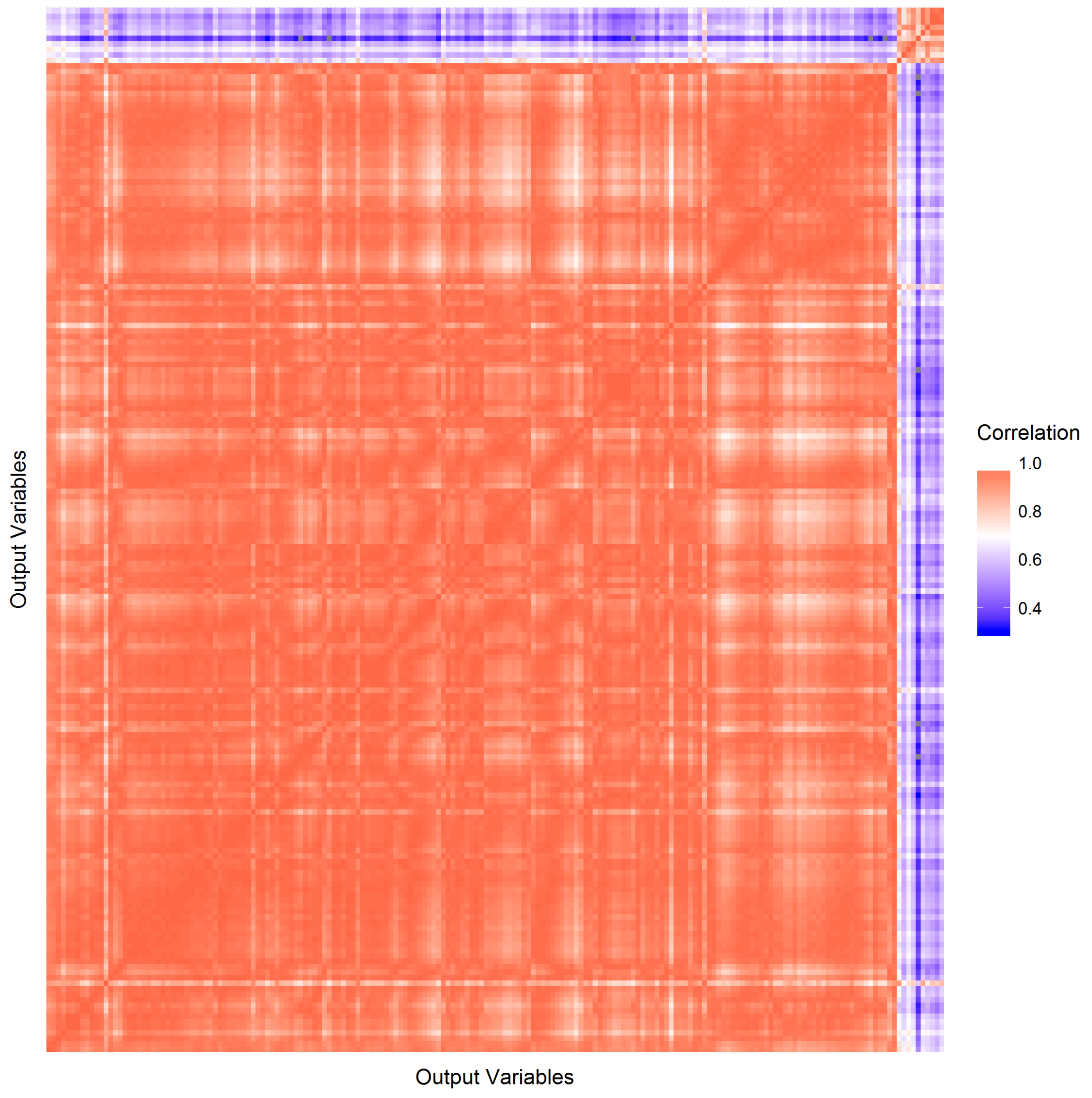

Figure 4.6: Correlation between the simulator outputs 1:189

each of the outputs. We handle the directional variables using X and Y components due to the periodic nature of the offshore wave direction as described further in Subsection 5.3.1.

Using the corresponding GPE fit, we predict the simulator values for each output variable. We also look at the validation plots, as described in Chapter 3, for each GPE fit and analyse them in the same way as mentioned in Chapter 3. Once we are happy with the validation plots, we compare the predictions to the true simulator output. In the case where the validation plot shows a significant validation failure we would try to refit the model by removing one row in the training dataset. Another solution would be to consider a different correlation function. We note the NRMSE and the coverage values for a comparison with the other methods, discussed in the results section.

Note that by fitting a univariate GPE to each of the simulator outputs, we assume that the outputs are independent and as a result we may have lost information about the between output dependence.

PC-GPE method

As we can see in Figure 4.6, the multiple outputs are highly correlated, and hence we could effectively use PCA to reduce the dimensions of the data. First, we convert the simulator output to principal components. Figure 4.7 shows the proportion of variance explained by each additional principal component. We can see that the first component explains approximately 91% of the variance in the simulator data. The second and third component represents 4% and 3% of the data, respectively. In other words, the first three components represent 98% of the variability of the simulator data. This is very useful as we can potentially reduce the dimensions of our dataset from 189 to three with minimal loss of information.

Figure 4.8 shows the effect that the number of principal components chosen has on the NRMSE. This data has been obtained by fitting a univariate GPE to c components, predicting at new input values, converting the predictions from the GPE to the simulator output variables and then calculating the NRMSE of the predicted values, where c ranges from 1 to 45. The red line shows the NRMSE generated from the IND method. We aim to get as close as possible to the red line using the least number of principal components. The blue dotted line shows the NRMSE generated by fitting only the first 6 principal components $(\zeta_1(X), ..., \zeta_6(X))$. It is evident from

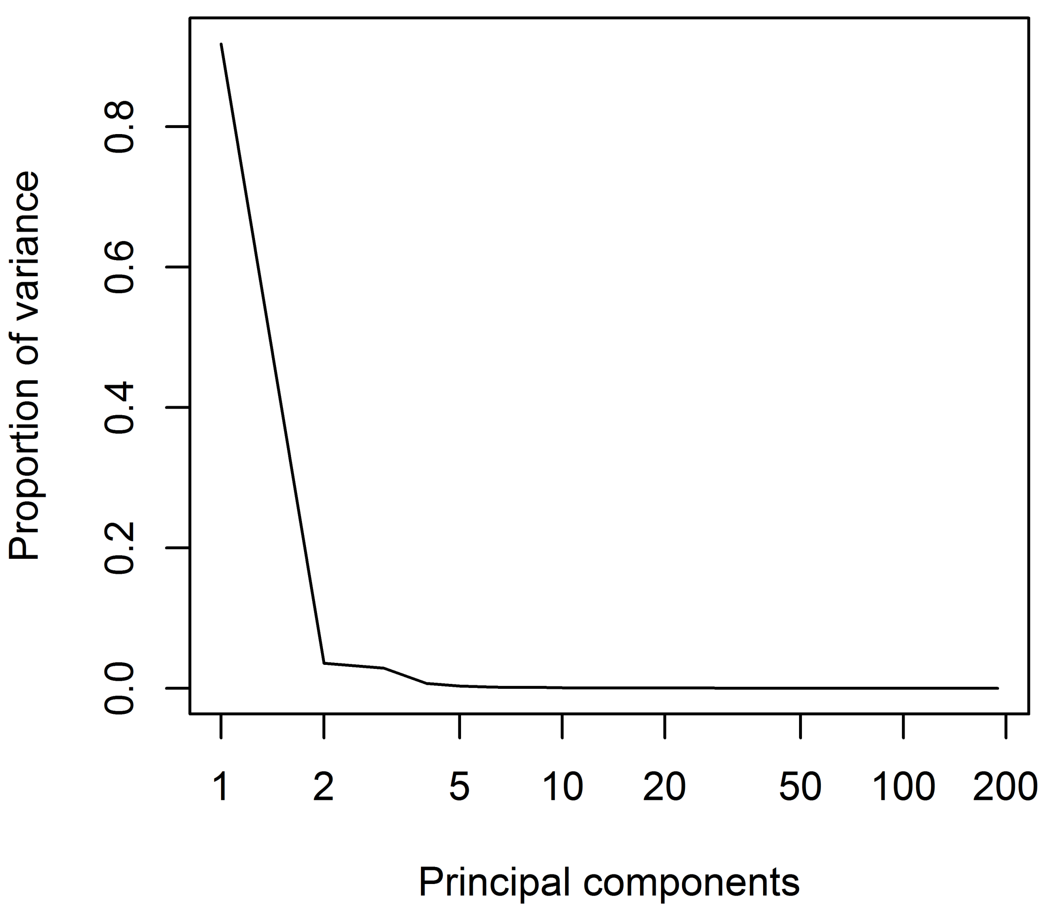

Figure 4.7: Proportion of variance explained by each additional principal component

the graph that increasing the number of principal components higher than $c = 6$ does not lead to a noteworthy reduction in NRMSE. Any additional component here would not help to significantly increase the accuracy of the predictions of the simulator output.

In a realistic application, we would often not have time or resources to calculate the results plotted on Figure 4.8. In this case, we can set a threshold of the percentage of the data that the principal components should represent (for example, 99%),

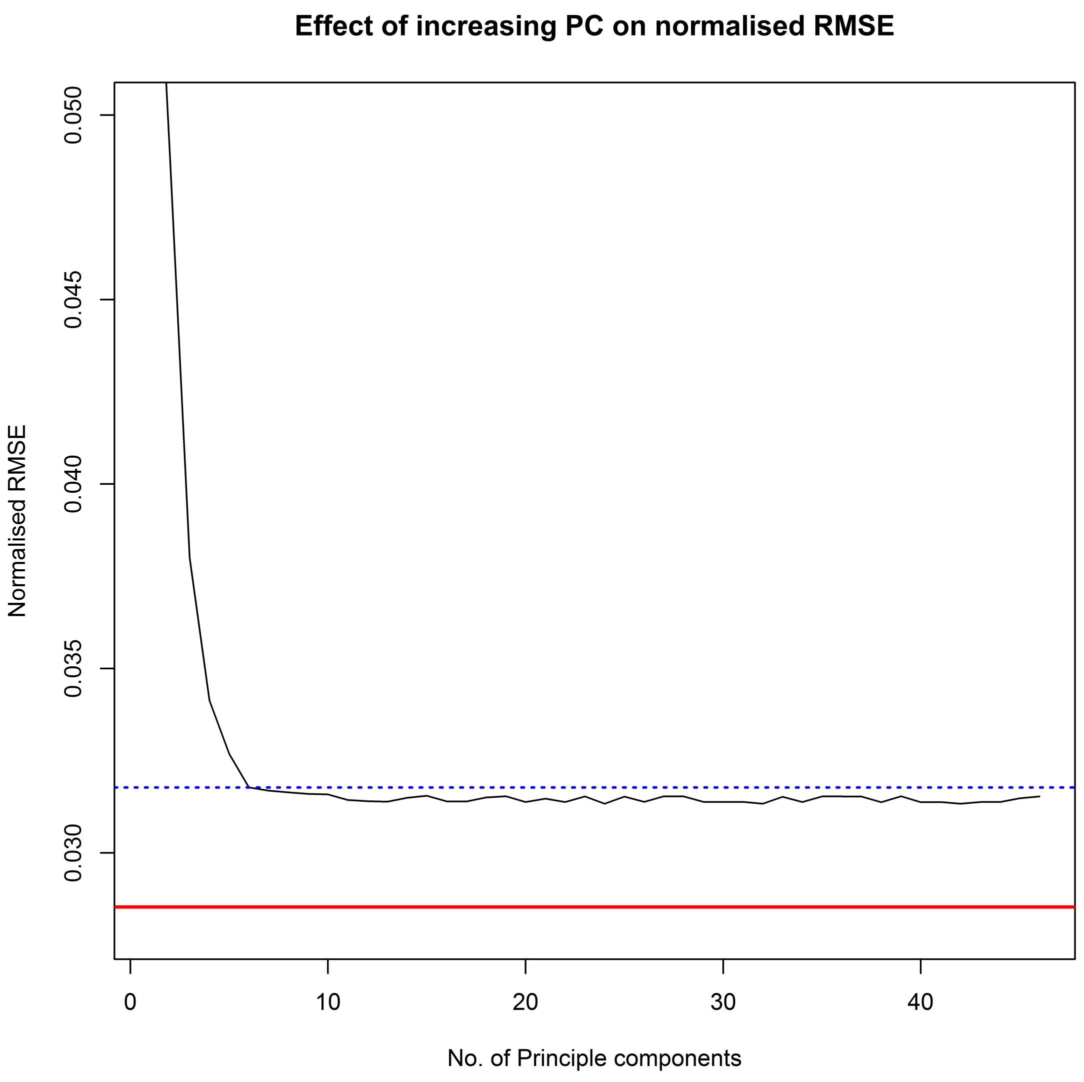

Figure 4.8: Effect of increasing PC on NRMSE. The red line represents the NRMSE generated from the IND method. The blue dotted line shows the NRMSE generated by fitting the first 6 components.

and using the cumulative proportion of the variance we can select the number of principal components to use. The first six components represents 99.4% of the data.

We choose to fit a univariate GPE to 6 principal components. Theoretically, the 6 principal components are independent to each other, therefore we can reasonably assume that no information is lost by modelling these components independently. We use these GPE fits, to predict the outcome of the new input values in terms of the principal components. Next, we assess the validation plots for each of the GPE fits. If there are no validation failures, we convert the predicted values (posterior mean and variance) back to the original unit of the simulator. We compare the predictions with the true output in terms of the NRMSE and the coverage values. We use this for comparison with the other methods.

Note that although the PCA method may not give the same accuracy of results as fitting each emulator independently, it reduces computational time by over 95%.

Separable method

For the SEP method, we fit a single separable GPE to all the output variables. We use a linear prior mean and a Gaussian correlation function to describe our prior beliefs of the simulator. We predict the simulator output for all the output variables. Validation of the separable GPE fit is done using the same techniques as mentioned in Section 3.5, one output variable at a time.

Figure 4.9 shows a plot of the predicted values using the SEP GPE and simulator output for one variable. Notice that the 95% confidence interval ribbon is quite wide, which may suggest that the SEP GPE is under confident about its predictions.

Results

Figure 4.10 shows the NRMSE obtained using different emulation techniques for each output variable. Notice that overall, the IND and PC-GPE method are quite comparable to each other, however the SEP GPE is worse off. This may be because of the common correlation length parameters for all the outputs. However note that overall the NRMSE is roughly centred around 0.065 for all values. For the PC-GPE the NRMSE is higher (even higher than SEP for some outputs) for simulator output 175 onwards (remember these were the less well correlated outputs from Figure 4.6).

Figure 4.11 shows the coverage values for all the methods for all the output

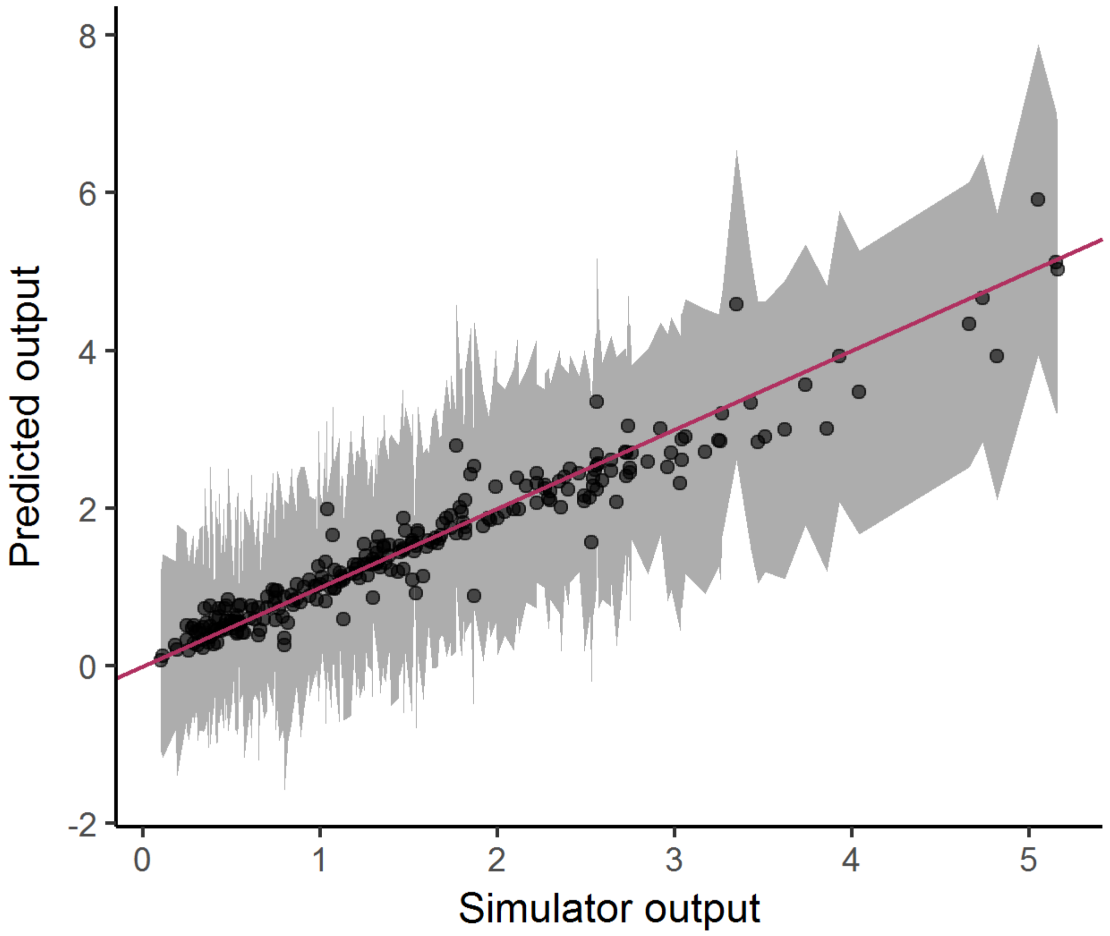

Figure 4.9: Predicted and simulated values for one output variable using the SEP GPE

variables. For the SEP method the coverage value is 1 for all the output variables. This (and the validation plots) suggest that the GPE is under confident about its predictions and hence the 95% credible intervals are wide and the residual errors always fall within the interval. For the PC-GPE, the coverage values centred around 0.8, which is reasonable, however, it suggests that the PC-GPEs were over confident about the predictions. For the IND method the coverage values are around 0.85 which is also reasonable.

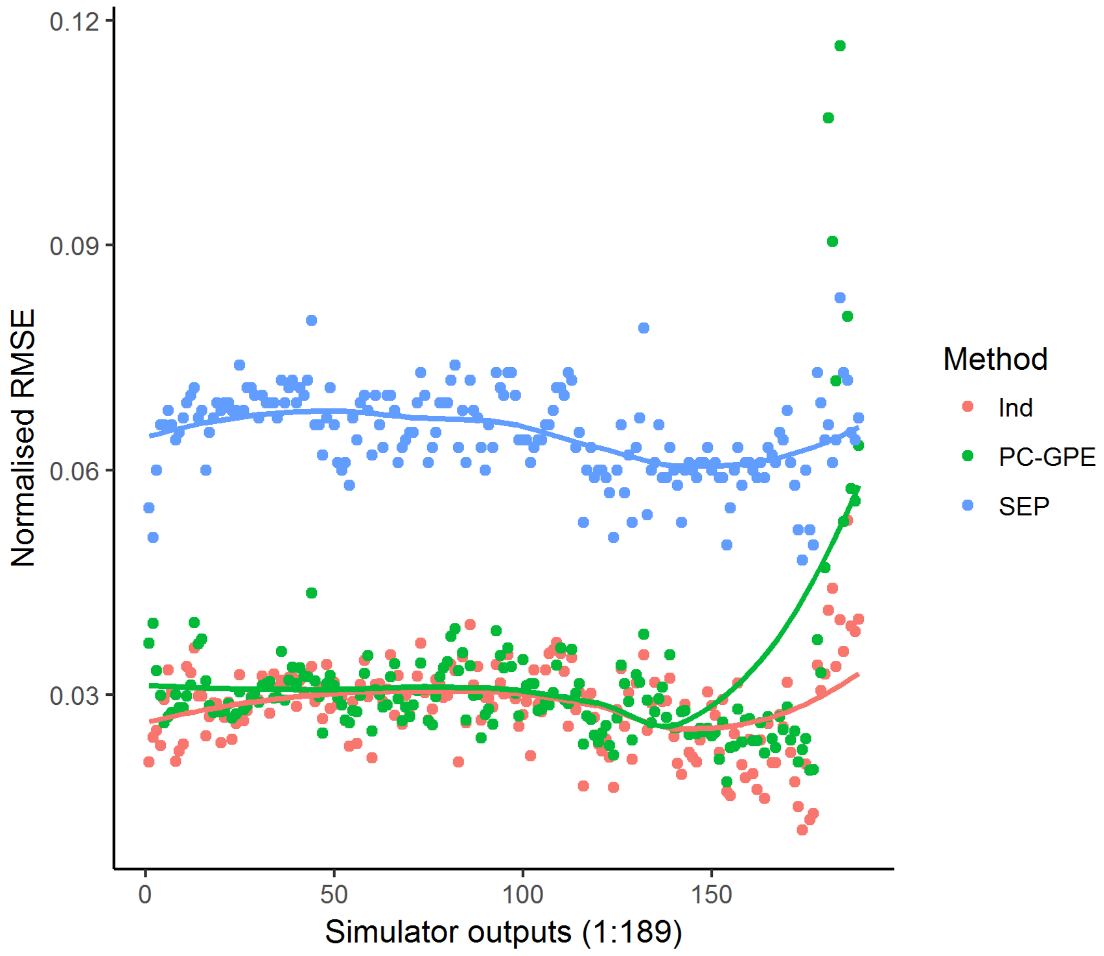

Figure 4.10: NRMSE for all the output variables and methods for the field-type output.

Discussion

From the results above we can see that the IND method outperformed the other methods in terms of posterior mean and variance. However it is laborious to fit 189 univariate GPEs and predicting the simulator output using the corresponding fit. It may also be very time consuming depending on the size of the data and number of output variables.

One of the main arguments against the IND method is that we would lose information regarding the between output correlations. In this case study we have only compared the marginal predictions of individual outputs, but if we compared

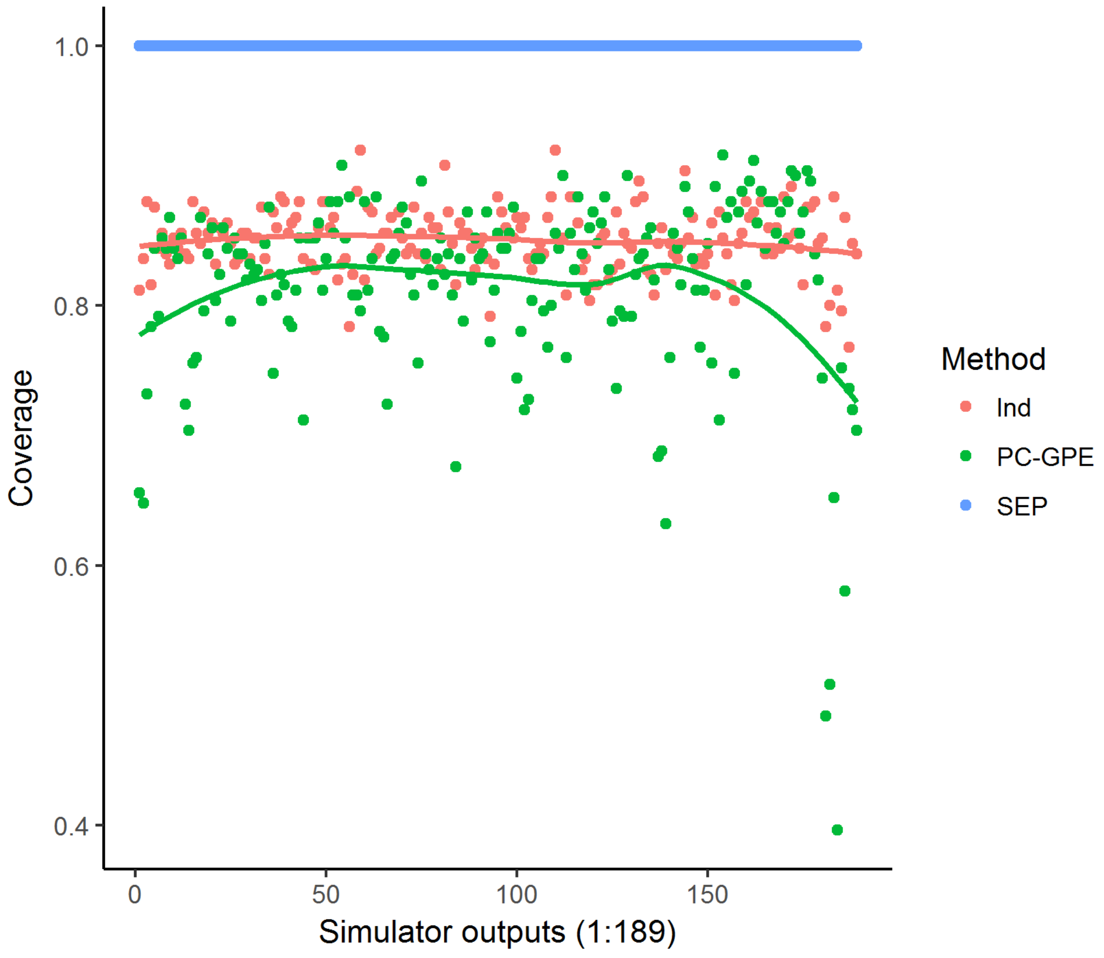

Figure 4.11: Coverage values for all the output variables and methods for the field-type output.

a scalar-valued function of these outputs, we would find that the predictions would likely be underconfident or overconfident.

The PC-GPE method is also comparable to the IND method in terms of marginal predictions and coverage values. Moreover, we only have to fit GPEs to a reduced set of principal components. Note here that there is no risk of loosing information about the between output correlation as the principal components are independent. This can help speed up analysis by over an order of magnitude. To improve the results for the PC-GPE, we could try using different prior mean assumptions.

The SEP method requires the least amount of effort and the point predictions are quite accurate, however the uncertainty associated around these predictions is often underestimated.

It is also worth noting, that for this particular case study, it took 21 seconds to fit a single univariate GPE. Therefore the total time taken to fit all the GPEs for the IND method was approximately 4000s, compared to 126s for the PC-GPE method. The SEP GPE took only 80s. This would play a big role in helping to decide which method to pick when the dataset is large.

4.4 Multiple-type output

We saw in the previous section how PC-GPE and separable GPE can be used to model multiple field outputs. In this section we shift our interest towards modelling multiple-type output. When modelling multiple-type outputs, it is important to note that multiple outputs often represent different physical quantities. In practice a typical simulator can have numerous outputs.

PC-GPE is less likely to be effective when applied to these scenarios as each output may vary on a different scale. Additionally, if outputs do not have a high correlation, and the number of outputs is small than we are unlikely to achieve useful dimension reduction.

Moreover, a considerable drawback for separable GPE's is that when the multiple outputs represent different physical quantities, one spatial correlation function amongst all outputs is too restrictive. This leads to poor performance of the emulator.

In this section we look at non-separable emulators as an alternate method. This method is promising as the covariance function allows multiple spatial correlation functions to be incorporated into the analysis. Fricker et al. (2013) describe two approaches to specify non-separable covariance functions: convolution and coregionalisation models. We chose to only focus on the latter.

In the Subsections that follow, we describe an approach to modelling multiple-type outputs using the linear model of coregionalisation (LMC) GPE as presented by Fricker et al. (2013). We first introduce a toy simulator and use this simulator as a simple example as we go through the detailed mathematical concepts. Later, we present a case study comparing the LMC GPE to the separable GPE.

4.4.1 Toy example: one input, two multiple-type outputs simulator

We use an example in this section to illustrate the detail as we present the LMC GPE. Here we describe a toy simulator with $p = 1$ input variable, and $q = 2$ output variables. Let's assume in this example, a simulator is represented by $\boldsymbol{y} = \boldsymbol{\eta}(\boldsymbol{x})$, where $\boldsymbol{y}$ can be written as $\boldsymbol{y} = (y_1, y_2) = \big(\eta_1(\boldsymbol{x}), \eta_2(\boldsymbol{x})\big)$. Suppose we run the simulator nine times for input values $(\boldsymbol{x}_1, ..., \boldsymbol{x}_9)$, which we denote X. The simulator output for these values is denoted $Y = \boldsymbol{\eta}(X) = (\boldsymbol{y}_1, ..., \boldsymbol{y}_9)$, where $\boldsymbol{y}_i = (y_{i,1}, y_{i,2})$. Using the GPE approach, we derive a probability distribution of $\boldsymbol{\eta}(\boldsymbol{x}_{10}), \ldots, \boldsymbol{\eta}(\boldsymbol{x}_N)$ given the training runs: X, Y by modelling $\boldsymbol{\eta}(.)$ as a 2-dimensional Gaussian Process.

Again, we assume that this simulator is computationally expensive and so we can only evaluate it at a limited number of values. There are many ways to select a subset of points over the input space, however for this example we just pick nine points that are equally spread over the input space.

Figure 4.12 shows the output of the toy simulator plotted against input values that range over the input space [0,5]. The training inputs and the corresponding

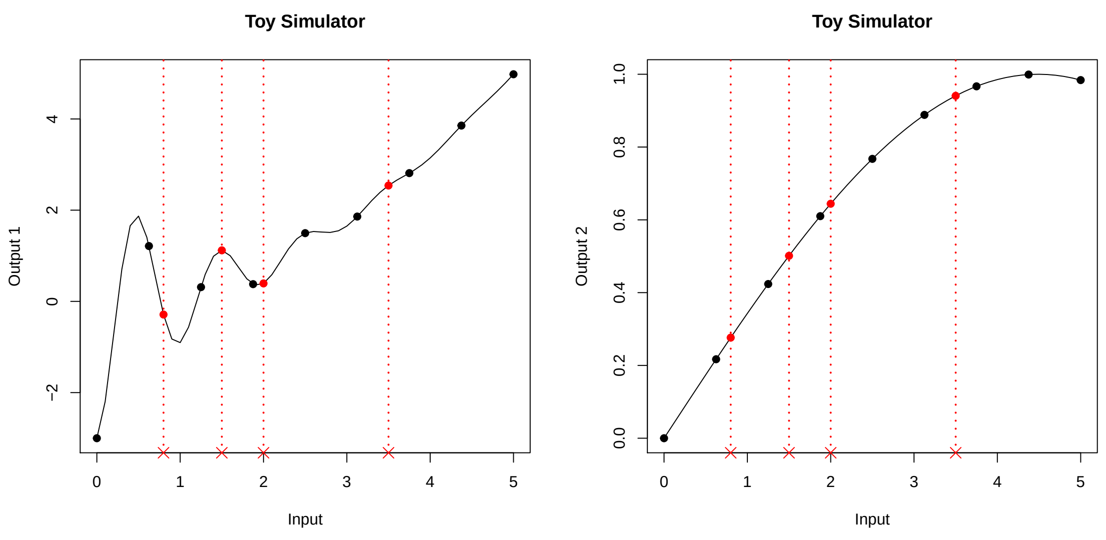

Figure 4.12: Multivariate (multiple-type output) toy simulator (1 input, 2 output). The black points highlight the training runs, the red crosses are the events which we wish to predict at and the red points are true simulator output for these events.

training outputs from the toy simulator function are given as follows:

$$X = \begin{pmatrix} 0 \\ 0.62 \\ 1.25 \\ 1.88 \\ 2.5 \\ 3.12 \\ 3.75 \\ 4.38 \\ 5 \end{pmatrix}, \quad Y = \begin{pmatrix} -3 & 0 \\ 1.21 & 0.22 \\ 0.31 & 0.42 \\ 0.38 & 0.61 \\ 1.5 & 0.77 \\ 1.86 & 0.89 \\ 2.81 & 0.97 \\ 3.85 & 1 \\ 4.98 & 0.98 \end{pmatrix}.$$

We wish to evaluate the function at the four input values: 0.8, 1.5, 2, 3.5.

4.4.2 LMC GPE

The LMC was developed within geostatistics as a tool for modelling multivariate spatial processes (Journel and Huijbregts (1978), Wackernagel (1995), Goulard and Voltz (1992)).

For a model with q outputs, we represent our uncertainty in the function $\boldsymbol{\eta}(.)$ by the multivariate GP prior:

$$\begin{aligned} \boldsymbol{\eta}(.) &= \boldsymbol{m}(.) + \boldsymbol{z}(.), \\ \boldsymbol{m}(.) &= (I_q \otimes \boldsymbol{h}(.)^T)\beta, \\ \boldsymbol{z}(.)|\theta &\sim GP_q[0, \boldsymbol{C}(.,.)], \end{aligned} \tag{4.33}$$

where the function $\boldsymbol{m}(.)$ represents the mean function, and $\boldsymbol{C}(.,.)$ represents the covariance function for the GP modelled by $\boldsymbol{z}(.)$.

Fricker et al. (2013) describe that the approach to LMC is to construct output processes $\boldsymbol{z}(.)$ as linear combinations of a number of building-block processes, written as

$$\boldsymbol{z}(.) = \boldsymbol{R}\boldsymbol{u}(.).$$

Here we construct an LMC with q building-block processes where $\boldsymbol{R}$ is a $q \times q$ full rank matrix, and $\boldsymbol{u}(.)$ is a vector of q independent GPs with zero mean and unit variance with spatial correlation functions $c_1(.,.), \ldots, c_q(.,.)$ that depend on some hyperparameters, $\boldsymbol{\theta}$. Each correlation function $c_i(.,.)$ could be any function as described in Chapter 3.3.

Two types of correlations are taken into consideration within $\boldsymbol{C}(.,.)$. First, the between-output correlation - which considers the dependencies of all the output parameters to the same inputs of the simulator. Second, the spatial correlation for each output over the input space - which relies on the assumption that neighbouring points in the input parameter space have similar output values.

The corresponding variance covariance function for the residual process, $\boldsymbol{z}(.)$ is given by

$$\boldsymbol{C}(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{R}[diag\{c_1(.,.),\ldots,c_q(.,.)\}]\boldsymbol{R}^T \tag{4.34}$$

$$= \sum_{j=1}^{q} \boldsymbol{\Sigma_j} c_j(\boldsymbol{x},\boldsymbol{x}'). \tag{4.35}$$

where for $j = 1\ldots q$, $\boldsymbol{\Sigma_j} = \boldsymbol{r_j}\boldsymbol{r_j}^T$ with $\boldsymbol{r_j}$ the $j-th$ column of $\boldsymbol{R}$ which is a $q \times q$ full rank matrix.

The variance covariance matrix

$$\boldsymbol{v}(X,X) = \sum_{j=1}^{q} \boldsymbol{\Sigma_j} \otimes c_j(X,X)$$

is a combination of correlation functions for each output as a weighted sum of functions $c_1(.,.),\ldots,c_q(.,.)$. These weights are given by $\boldsymbol{\Sigma_j}$ which is known as the *coregionalisation matrix.* This way of representing the covariance matrix allows for incorporating variation occurring on different scales.

Note that since square root of matrices are not unique, the between-outputs covariance for LMC emulator, $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{R}^T$ is not unique and can lead to different models with the same $\boldsymbol{\Sigma}$. To overcome this, Fricker et al. (2013) choose to parameterise the model over Σ and then calculating a specific $\boldsymbol{R}$ by an eigendecomposition.

Fricker et al. (2013) emphasize that choosing a decomposition method requires considerable amount of attention, and a method should not simply be chosen because of its computational ease. Gelfand et al. (2004) use the Cholesky decomposition mainly for its computational ease, however this decomposition method is not invariant to the order of the outputs. In most real world applications there may be no obvious hierarchy of dependence of outputs.

The eigendecomposition of $\boldsymbol{\Sigma}$ is given by $\boldsymbol{R} = \boldsymbol{\Lambda}\sqrt{\boldsymbol{D}}\boldsymbol{\Lambda}^T$, where $\boldsymbol{\Lambda}$ is the orthogonal matrix of normalised eigenvectors of $\boldsymbol{\Sigma}$, $\boldsymbol{D}$ is a diagonal matrix of the eigenvalues of $\boldsymbol{\Sigma}$, and the resulting matrix $\boldsymbol{R}$ is a symmetric matrix. Here, the order of outputs will have no impact on the end result.

We can describe our prior mean of $\boldsymbol{\eta}(.)$ as:

$$E[\boldsymbol{\eta}(\boldsymbol{x})|\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\Phi}] = \boldsymbol{m}(\boldsymbol{x}) \tag{4.36}$$

where we model the mean functions as $\boldsymbol{m}(\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{x})\boldsymbol{\beta}$, where $\boldsymbol{\beta} = [\beta_{1,1}, \ldots, \beta_{1,m_1}, \ldots, \beta_{q,1}, \ldots, \beta_{q,m_q}]$ is a column vector of unknown coefficients (hyperparameters) of length qm and $\beta_{i,1}, \ldots, \beta_{i,m_i}$ refer to the unknown coefficients of output i. The mean function is typically non-constant, and represents the global trend of the model output across input space. The mean function helps the emulator to predict outputs in regions of the input space that are sparsely populated and when predicting outputs that are outside the input space (extrapolation).

We define the vector $\boldsymbol{h}_i(.)$ to consist of m_i regressor functions of $\boldsymbol{x}$ incorporating our beliefs of the i-th output of computer model $\boldsymbol{\eta}(.)$. We also define $H_i = \big(\boldsymbol{h}_i(\boldsymbol{x}_1), \ldots, \boldsymbol{h}_i(\boldsymbol{x}_n)\big)^T$ to be a $n \times m_i$ matrix representing the regressor functions for output i. Then, $\boldsymbol{H}$ is a $qn \times qm$ matrix ($m = m_1 + \ldots + m_q$) written as

$$\boldsymbol{H} = \begin{pmatrix} H_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & H_2 & & \vdots \\ \vdots & & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & H_q \end{pmatrix}. \tag{4.37}$$

The prior mean and variance for $\boldsymbol{\eta}(\boldsymbol{x})$ depend on the hyperparameters $\boldsymbol{\beta}, \boldsymbol{\Sigma}$, and $\boldsymbol{\Phi}$. The hyperparameters in the LMC covariance function are $\boldsymbol{\Sigma}$ and $\Phi = \{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_q\}$, where $\boldsymbol{\phi}_i$ denotes the collection of hyperparameters from the basis correlation function $c_i(\boldsymbol{x}, \boldsymbol{x})$ (equivalent to $\boldsymbol{\delta}$ in the univariate GPE). The prior distributions of $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ are covered separately in a latter Subsection. We use a weak prior for $\boldsymbol{\beta}$:

$$p_{\boldsymbol{\beta}}(\boldsymbol{\beta}) \propto 1. \tag{4.38}$$

In summary, the prior distribution for $\boldsymbol{\eta}(\boldsymbol{x})$ is given by

$$\boldsymbol{\eta}(\boldsymbol{x})|\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\Phi} \sim N\big(\boldsymbol{H}(\boldsymbol{x})\boldsymbol{\beta}, \boldsymbol{v}(\boldsymbol{x}, \boldsymbol{x}')\big) \tag{4.39}$$

The linear model of coregionalisation (LMC) emulator

We observe $\boldsymbol{\eta}(X)$ at the training inputs $X = \boldsymbol{x}_1, ..., \boldsymbol{x}_n$, which produces data as tabulated in Y such that:

$$Y = \begin{pmatrix} \boldsymbol{\eta}_1(\boldsymbol{x}_1) & \boldsymbol{\eta}_2(\boldsymbol{x}_1) & \cdots & \boldsymbol{\eta}_q(\boldsymbol{x}_1) \\ \boldsymbol{\eta}_1(\boldsymbol{x}_2) & \boldsymbol{\eta}_2(\boldsymbol{x}_2) & \cdots & \boldsymbol{\eta}_q(\boldsymbol{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{\eta}_1(\boldsymbol{x}_n) & \boldsymbol{\eta}_2(\boldsymbol{x}_n) & \cdots & \boldsymbol{\eta}_q(\boldsymbol{x}_n) \end{pmatrix}. \tag{4.40}$$

Let $\vec{\boldsymbol{Y}}$ be a column vector of Y such that

$$\vec{\boldsymbol{Y}}^T = \big(\boldsymbol{\eta}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{\eta}_1(\boldsymbol{x}_n), \ldots, \boldsymbol{\eta}_q(\boldsymbol{x}_1), \ldots, \boldsymbol{\eta}_q(\boldsymbol{x}_n)\big).$$

The likelihood function of the data is given as

$$\vec{\boldsymbol{Y}}|\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\Phi} \sim N_{n,q}(\boldsymbol{H}\boldsymbol{\beta}, V), \tag{4.41}$$

where $\boldsymbol{H} \in \mathbb{R}^+_{qn,mq}$ and $\boldsymbol{V} = \boldsymbol{v}(X, X) = \sum_{j=1}^{q} \boldsymbol{\Sigma_j} \otimes c_j(X, X) \in \mathbb{R}^+_{qn,qn}$.

Let $\acute{X} = \boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_N$, denote the new points that we would like to evaluate the function at. Then, given the prior distribution (4.38) and the likelihood function, we update the distribution of $\boldsymbol{\eta}(.)$ using the partitioning property of multivariate normal distributions to get

$$\boldsymbol{\eta}(\acute{X})|\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\Phi}, \vec{\boldsymbol{Y}} \sim GP(\boldsymbol{m}^{*}(.), \boldsymbol{v}^{*}(.,.)), \tag{4.42}$$

where,

$$\begin{aligned}\boldsymbol{m}^{*}(\acute{X}) &= \boldsymbol{H}(\acute{X})^T\boldsymbol{\beta} + \boldsymbol{u}(\acute{X})V^{-1}(\vec{\boldsymbol{Y}} - \boldsymbol{H}\boldsymbol{\beta})^T \\ \boldsymbol{v}^{*}(\acute{X}, \acute{X}) &= \boldsymbol{v}(\acute{X}, \acute{X}) - \boldsymbol{u}(\acute{X})^T V^{-1}\boldsymbol{u}(\acute{X}) \\ \boldsymbol{u}(\acute{X})^T &= \boldsymbol{v}(X, \acute{X}).\end{aligned}$$

To derive the posterior distribution of $\boldsymbol{\eta}(\acute{X})|\boldsymbol{\Sigma}, \boldsymbol{\Phi}$, we integrate (4.42) with respect to $\boldsymbol{\beta}$ which gives

$$\boldsymbol{\eta}(\acute{X})|\boldsymbol{\Sigma}, \boldsymbol{\Phi}, \vec{\boldsymbol{Y}} \sim GP(\boldsymbol{m}^{**}(.), \boldsymbol{v}^{**}(.,.)), \tag{4.43}$$

where,

$$\boldsymbol{m}^{**}(\acute{X}) = \boldsymbol{H}(\acute{X})^T\hat{\boldsymbol{\beta}} + \boldsymbol{u}(\acute{X})V^{-1}(\vec{\boldsymbol{Y}} - \boldsymbol{H}\hat{\boldsymbol{\beta}})^T \tag{4.44}$$

$$\boldsymbol{v}^{**}(\acute{X}, \acute{X}) = \boldsymbol{v}(\acute{X}, \acute{X}) - \boldsymbol{u}(\acute{X})^T V^{-1}\boldsymbol{u}(\acute{X}) + \tag{4.45}$$

$$(\boldsymbol{H}(\acute{X}) - \boldsymbol{u}(\acute{X})^T V^{-1}\boldsymbol{H})(\boldsymbol{H}^T V^{-1}\boldsymbol{H})^{-1}(\boldsymbol{H}(\acute{X}) - \boldsymbol{u}(\acute{X})^T V^{-1}\boldsymbol{H})^T \tag{4.46}$$

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{H}^T V^{-1}\boldsymbol{H}^T)^{-1}\boldsymbol{H}^T V^{-1}\vec{\boldsymbol{Y}} \tag{4.47}$$

The resulting posteriors are proper as long as $n \geqslant mq + q$. This multivariate approach for approximating $\boldsymbol{\eta}(\acute{X})$ is conditional on hyperparameters $\boldsymbol{\Sigma}$ and $\boldsymbol{\Phi}$ which cannot be analytically derived. Therefore, these hyperparameters need to be estimated by maximising their likelihood function (See the following Subsection). Similar to the univariate and separable emulators, the posterior mean, $\boldsymbol{m}^{**}(.)$ smoothly interpolates the training data to predict the output of the computer model at new inputs. The posterior variance covariance matrix $\boldsymbol{v}^{**}(.,.)$ quantifies the uncertainty associated with these predictions. Note that the two important properties of the emulator still hold: predicting at the training inputs replicates the training outputs exactly and with zero uncertainty.

Estimating the hyperparameters $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$

The prior distribution for the hyperparameters has the form:

$$p(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\Phi}) \propto p(\boldsymbol{\Phi})|\Sigma|^{-\frac{q+1}{2}}. \tag{4.48}$$

Using Bayes' theorem to combine 4.41 with 4.48, the full posterior is

$$p(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\Phi}|\vec{\boldsymbol{Y}}) \propto p(\boldsymbol{\Sigma}, \boldsymbol{\Phi})|V|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\vec{\boldsymbol{Y}} - \boldsymbol{H}\boldsymbol{\beta})^T V^{-1}(\vec{\boldsymbol{Y}} - \boldsymbol{H}\boldsymbol{\beta})\right] \tag{4.49}$$

Integrating out the matrix $\boldsymbol{\beta}$ and simplifying gives:

$$p(\boldsymbol{\Sigma}, \boldsymbol{\Phi}|\vec{\boldsymbol{Y}}) \propto p(\boldsymbol{\Sigma}, \boldsymbol{\Phi})|V|^{-\frac{1}{2}}|\boldsymbol{H}^T V^{-1}\boldsymbol{H}|^{-\frac{1}{2}} \tag{4.50}$$

$$\exp\left[-\frac{1}{2}(\vec{\boldsymbol{Y}} - \boldsymbol{H}\hat{\boldsymbol{\beta}})^T V^{-1}(\vec{\boldsymbol{Y}} - \boldsymbol{H}\hat{\boldsymbol{\beta}})\right]. \tag{4.51}$$

We find an optimal value for $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$ by maximising the likelihood function.

Toy GPE

For our toy example, we assume the simulator has a linear mean function. We write this as:

$$\boldsymbol{m}(\boldsymbol{x}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0.62 & 0 & 0 \\ 1 & 1.25 & 0 & 0 \\ 1 & 1.88 & 0 & 0 \\ 1 & 2.5 & 0 & 0 \\ 1 & 3.12 & 0 & 0 \\ 1 & 3.75 & 0 & 0 \\ 1 & 4.38 & 0 & 0 \\ 1 & 5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0.62 \\ 0 & 0 & 1 & 1.25 \\ 0 & 0 & 1 & 1.88 \\ 0 & 0 & 1 & 2.5 \\ 0 & 0 & 1 & 3.12 \\ 0 & 0 & 1 & 3.75 \\ 0 & 0 & 1 & 4.38 \\ 0 & 0 & 1 & 5 \end{pmatrix} \begin{pmatrix} \beta_{1,1} \\ \beta_{1,2} \\ \beta_{2,1} \\ \beta_{2,2} \end{pmatrix}.$$

We also assume that this simulator has a Gaussian correlation function (4.19). We estimate the value of $\boldsymbol{\Phi}$ as $(0.46, 2.04)$ and $\boldsymbol{\Sigma}$ as below by maximising the likelihood function (4.50). Using these values we can calculate $\hat{\boldsymbol{\beta}}$ with equation (4.47).

$$\hat{\boldsymbol{\Sigma}} = \begin{pmatrix} 53.53 & 3.4 \times 10^{-5} \\ 3.4 \times 10^{-5} & 1.9 \end{pmatrix}. \tag{4.52}$$

We note that this calculation requires the V matrix. We show the calculations for one element of the matrix, $v(\boldsymbol{x}_2, \boldsymbol{x}_1) = \boldsymbol{\Sigma_2} c_2(x_2, x_1)$. We note that

$$c(x_2, x_1) = \exp\left[-\left(\frac{x_2 - x_1}{\phi_2}\right)^2\right].$$

Substituting the relevant values will get

$$\begin{aligned} c(x_2, x_1) &= \exp\left[-\left(\frac{0 - 0.62}{2.04}\right)^2\right]. \\ &= \exp\left[-0.01\right] \\ &= 0.99. \end{aligned}$$

To calculate the V matrix, we first calculate the A matrix (like in separable GPE or Univariate GPE) for all outputs. Also calculate the corresponding $\boldsymbol{\Sigma_i}$ for each output. Then, add together the Kronecker products of $\boldsymbol{\Sigma_i}$ and A_i for each output.

Below we show the values of $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} y1 & y2 \\ -9.22 & -0.52 \\ 3.58 & -0.05 \end{pmatrix}. \tag{4.53}$$

We derive the GPE predictions and associated standard deviation around these predictions as:

$$m^{**}\begin{pmatrix}0.8\\1.5\\2\\3.5\end{pmatrix} = \begin{pmatrix}1.2 & 0.28\\4e-02 & 0.5\\0.63 & 0.64\\2.33 & 0.94\end{pmatrix} \tag{4.54}$$

$$v^{**}\begin{pmatrix}0.8 & 0.8\\1.5 & 1.5\\2 & 2\\3.5 & 3.5\end{pmatrix}, = \begin{pmatrix}4e-04 & -3e-04 & 1e-04 & 2e-04 & 0 & 0 & 0 & 0\\-3e-04 & 2e-04 & -1e-04 & -1e-04 & 0 & 0 & 0 & 0\\1e-04 & -1e-04 & 0 & 1e-04 & 0 & 0 & 0 & 0\\2e-04 & -1e-04 & 1e-04 & 2e-04 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{pmatrix}. \tag{4.55}$$

Figure 4.13 shows the posterior distribution of the toy simulator given it was fit to the points indicated by the black dots. The red dots indicate the predictions made using the LMC GPE, and the red ribbon indicates the 95% uncertainty around these predictions.

The NRMSE for the LMC GPE is 0.33 for output 1 and 2.64×10^{-8} for output 2. We can see that for the toy example the GPE predicted the second output variable accurately with minimal uncertainty. For output 1, the prediction and the uncertainty have been poorly estimated. In this case, it may be worth adding more training events for the GPE.

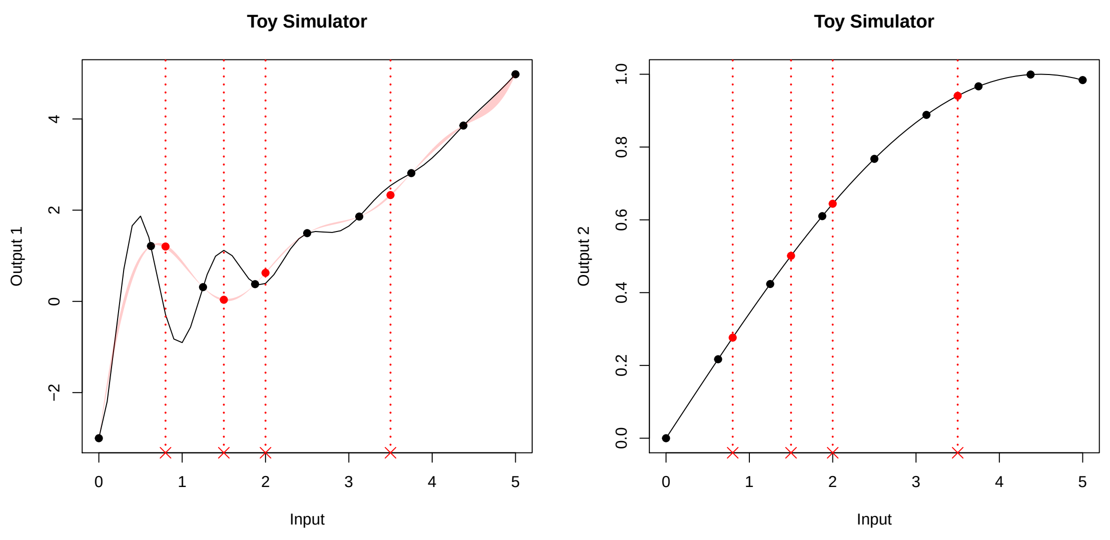

Figure 4.13: Multivariate (multiple-type output) toy simulator (1 input, 2 output). The black points highlight the training runs, the red crosses are the events which we wish to predict at and the red points are GPE predictions for these events. The red shaded region represents the GPE uncertainty around these predictions.

4.4.3 Case Study - Greater Wash, UK

In this case study we aim to illustrate the LMC GPE method in comparison to independently fitting univariate GPEs to the output (IND) and to fitting the separable GPE (SEP) mentioned in Section 4.3.3. We explore these methods using the context of case study 2 as described in Chapter 2.2.2 and as an extension to the case study in Section 3.6.

For all the methods, we use a design selected using the MDA method where the first 150 design events are plotted on the upper triangular matrix in Figure 3.3. We use a linear prior mean function and a Gaussian correlation function to describe our prior beliefs of the simulator for all methods. We also handle the directional input and output variables by converting them to X and Y components to account for the periodicity (see Subsection 5.3.1 for more details).

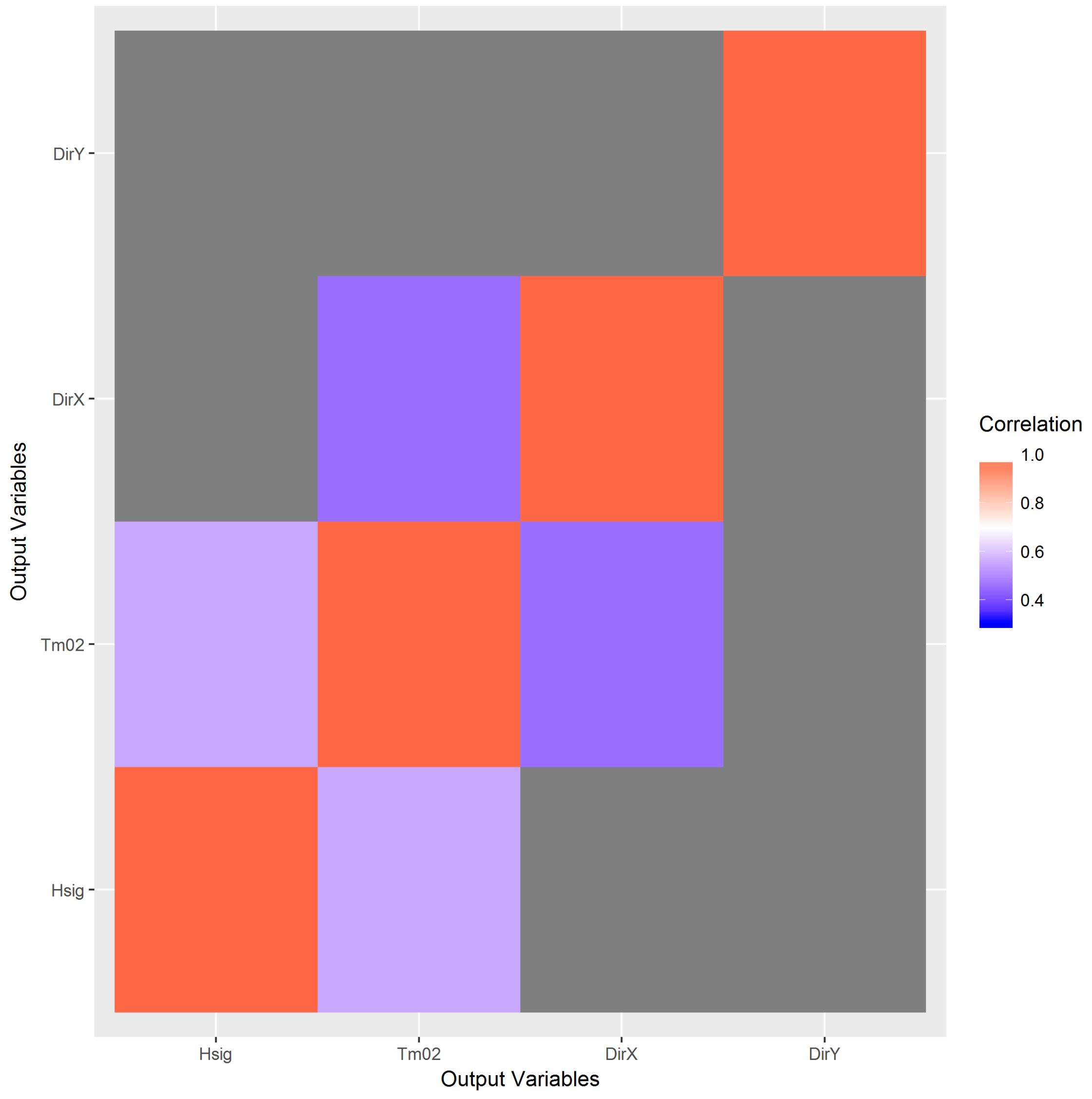

Figure 4.14: Correlation between the simulator outputs for the multiple-type output.

Results

Figure 4.14 shows a correlation plot of the output variables. From this plot, it is evident that these variables are not strongly correlated to each other and with only four output variables PC-GPE may not achieve useful dimension reduction. Moreover, we expect that one set of correlation length parameters may be too restrictive to explain the variance in all the outputs.

Table 4.1 and 4.2 show the estimated correlation length parameters for all the

	LMC.Hsig	LMC.Tm02	LMC.DirX	LMC.DirY
WindSpeed	1.98	-0.01	1.05	1.40
WaveHeight	0.64	0.25	6.17	0.70
PeakWavePeriod	2.24	1.44	0.55	1.87
WaterLevel	2.42	2.70	2.49	2.07
WindDirX	1.30	2.19	-0.07	0.58
WindDirY	1.69	2.16	0.17	-0.37
WaveDirX	0.48	0.55	-0.04	0.36
WaveDirY	0.84	1.14	0.03	-0.24

Table 4.1: Correlation length parameters for the LMC GPE at n = 500

	SEP	IND.Hsig	IND.Tm02	IND.DirX	IND.DirY
WindSpeed	1.16	1.97	-0.00	1.06	1.41
WaveHeight	0.18	0.59	0.25	5.06	0.72
PeakWavePeriod	1.84	2.22	1.44	0.56	1.87
WaterLevel	2.01	2.35	2.72	2.44	2.06
WindDirX	0.76	1.21	2.14	-0.07	0.57
WindDirY	0.42	1.63	2.17	0.18	-0.37
WaveDirX	0.14	0.51	0.54	-0.04	0.35
WaveDirY	0.39	0.86	1.15	0.02	-0.25

Table 4.2: Correlation length parameters for the Separable and Independent GPE at n = 500

methods. For the LMC and IND methods, we have a different set of correlation length parameters for each output. Notice that for each output, the correlation length parameters from the IND and LMC method are very similar, unlike the SEP method. This is expected, as the LMC model estimates correlation length parameters for each correlation function per output variable. On the other hand, the SEP method estimates one value to be used amongst all the output variables. Note also that there is a variation in the correlation length parameters amongst the inputs.

For the SEP method and for each GPE in the IND method, we estimate (using maximum likelihood estimation) eight values for the correlation length parameters (number of inputs including the X and Y components of the directional variables). For the LMC GPE, we have to estimate 10 values for Σ which is a 4×4 symmetric matrix and $8 \times 4 = 32$ correlation length parameters. This can get time consuming and challenging to optimise. Optimising such large numbers of values can lead to convergence issues where the optimiser can get stuck in a local minima or maxima. The time taken to fit the three GPEs is shown in Figure 4.15 which is plotted on a log scale. We can see that the LMC method takes very long to fit, and we know that it is spending a majority of the time optimising the hyperparameters. The SEP method is the quickest.

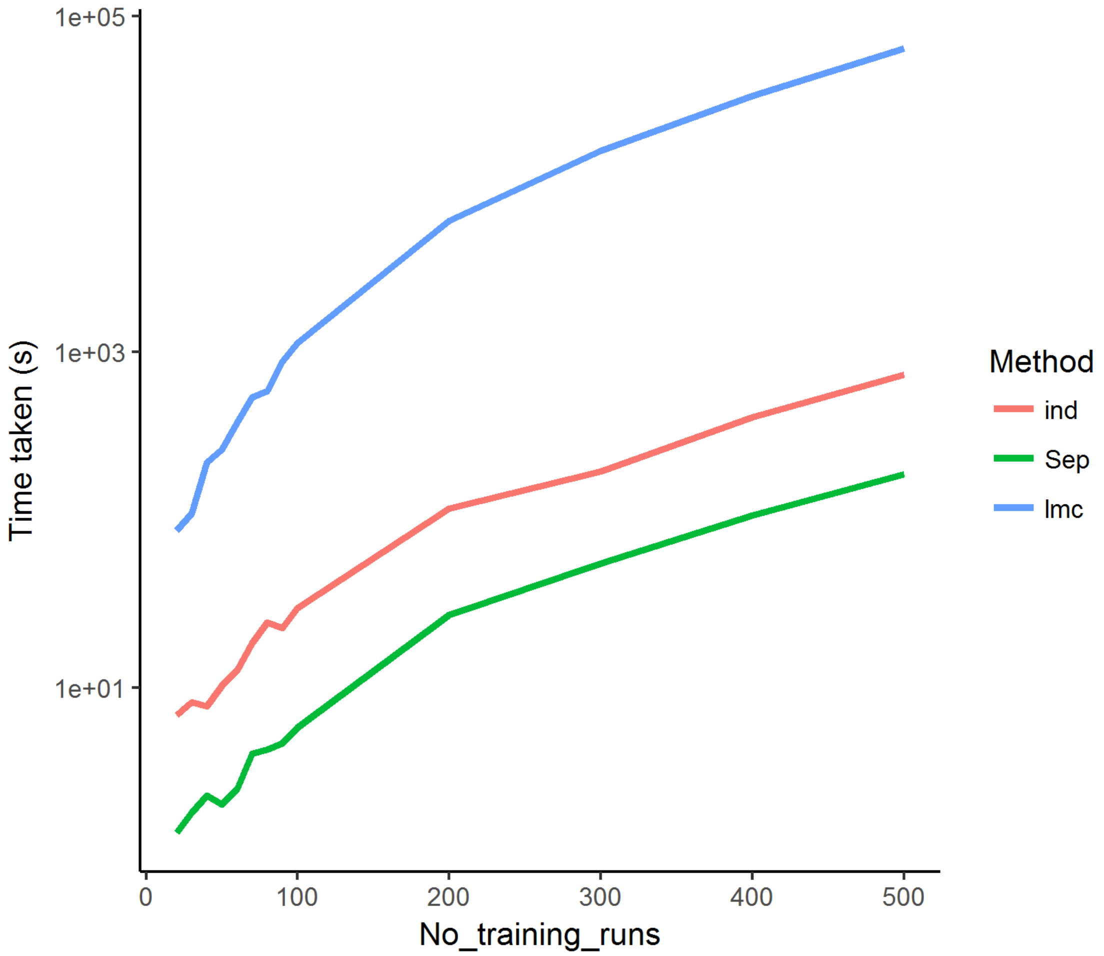

Figure 4.15: Comparison of time taken to fit GPEs to all the output variables for each method.

Figure 4.16 shows the NRMSE for the output variables and for Wave Direction (in degrees) and steepness where steepness is a function of Hsig and Tm02 given as:

$$steepness = \frac{2 \times \pi \times Hsig}{9.81 \times Tm02^2}.$$

We introduce steepness here as a function of two of the simulator outputs in order to assess the accuracy of the joint prediction distribution. This function to calculate steepness is used within HR Wallingford to feed into the next model in the chain.

We can see that all the methods lead to very similar NRMSE. The IND method and the LMC method are often very similar and better than the SEP method, which is expected. The cases where the LMC is worse off in comparison to the IND method may suggest convergence problems with the optimisation. Notice how the difference between the methods for output variable Hsig and Tm02 change when we compare steepness predictions, particularly at $n = 500$. The IND method and the LMC method are significantly worse off than the SEP method. This is not the case for $n < 500$ which may suggest that for the LMC GPE the correlation between Hsig and Tm02 may not have been captured properly. For the IND GPE, we expect that it is worse off when predicting the joint distribution of multiple outputs.

Figure 4.17 shows the coverage values for the output variables. We can see that in general, all the methods converge quickly to good and comparable coverage values. For Hsig, however, the IND and LMC GPE are overconfident about their predictions. This could be because of the high correlation between input wave height and output wave height.

Figure 4.18 shows the prediction output against the simulator output for the different output variables and methods. We used $n = 500$ here. Similar to the observation we made on the previous plot, all the methods are over confident about their predictions for Hsig. However, we note that this is not a problem as the predictions are very accurate. For the other output variables, we can see that each of the methods have very similar uncertainty bands and prediction accuracy and that neither are noticeably over or under confident about their predictions.

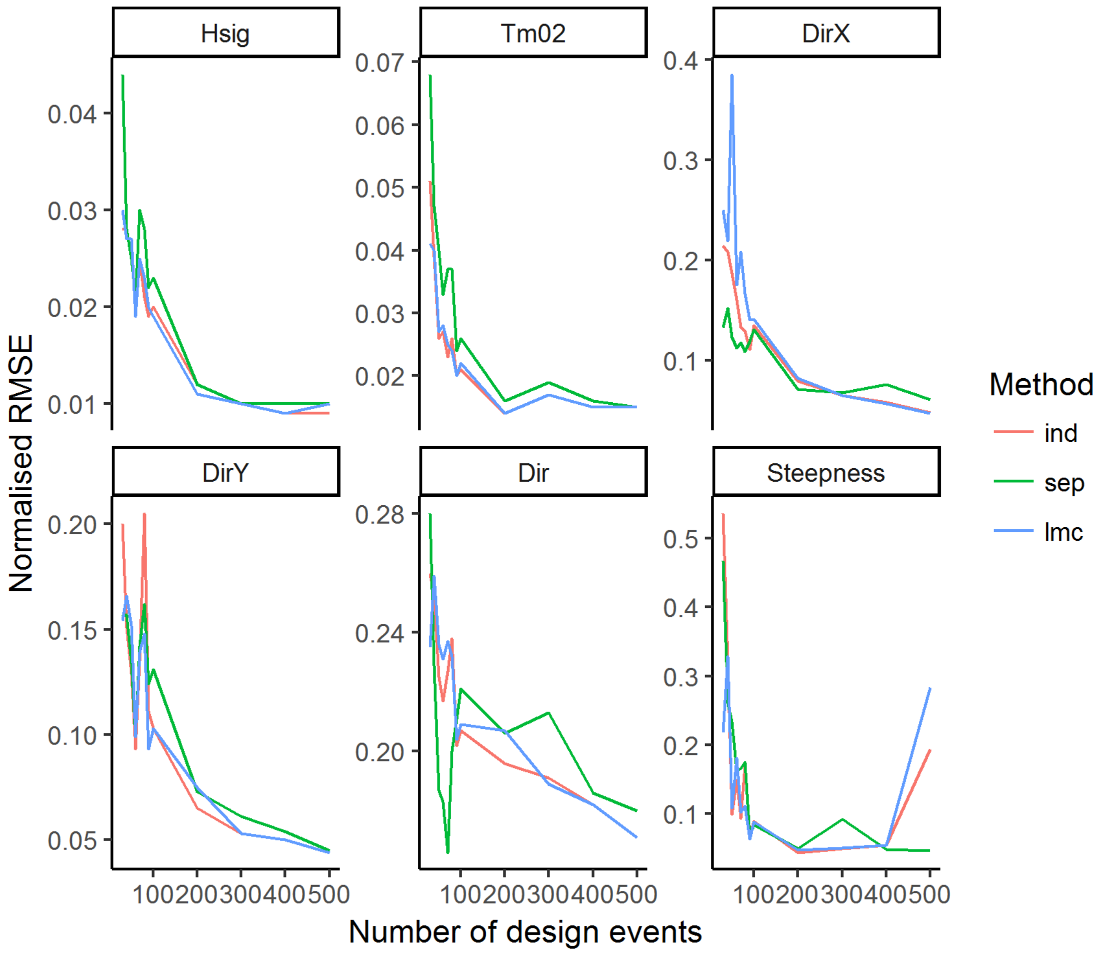

Figure 4.16: NRMSE for all the output variables and methods for the multiple-type outputs.

Discussion of results

We compared three methods of emulating multiple-type outputs. All the methods lead to very good and comparable results. The SEP method on the whole is slightly worse off than the other two methods, but there is little evidence to suggest that the LMC GPE method is better than the IND method.

It may be worth comparing the predicted joint distribution of a function that combines correlated outputs. In this case we tried to show this with steepness, however the outputs (Hsig and Tm02) were moderately correlated. Moreover they may have higher correlations with input values that may affect the prediction.

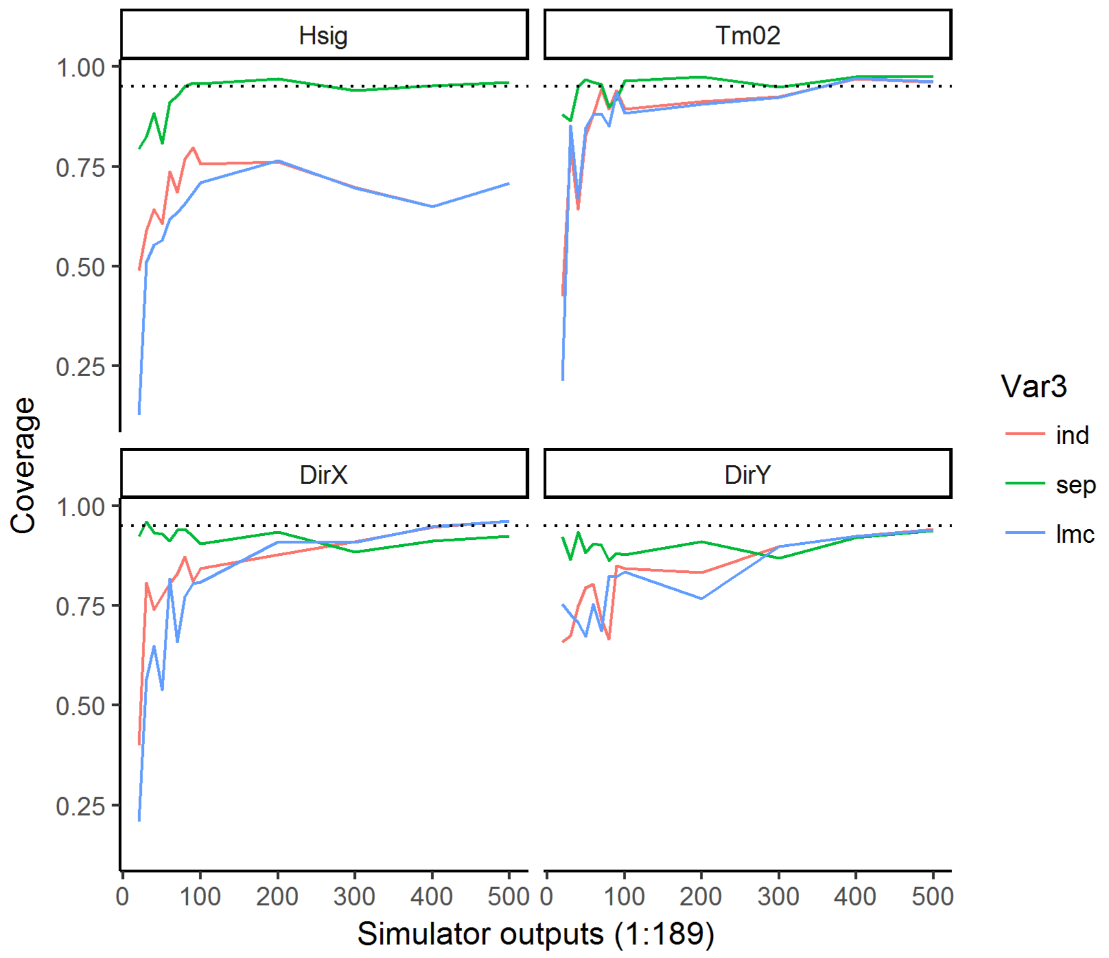

Figure 4.17: Coverage values for all the output variables and methods for the multiple-type outputs. The dotted line represents 95% coverage.

Ultimately the decision depends on what the user would like to do with the predictions. If we are only interested in using the marginal point estimates and uncertainties, then it can be argued that the simple IND approach may be used. However if predictions of joint distributions of correlated outputs are of interest than it may be worth exploring the LMC GPE and SEP GPE to the particular application.

In theory, we expect that LMC GPE would be better than the SEP GPE and the IND method, however because of the high number of values the LMC GPE needs to optimise, there is a possibility of false convergence of the optimiser which can lead

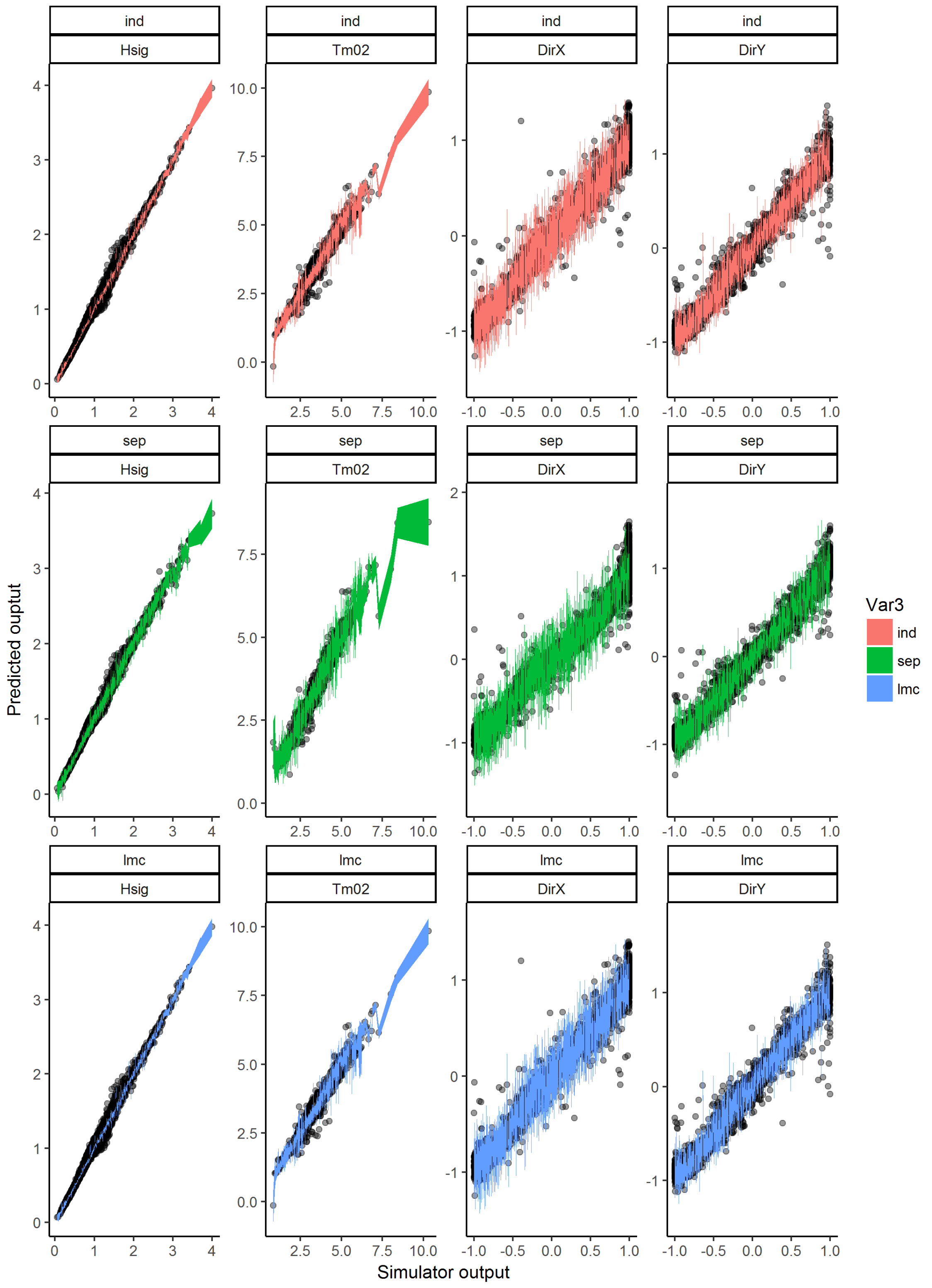

Figure 4.18: Comparison of predicted and simulated values for all output variables using all the methods.

to a poor GPE. Moreover the time taken to fit a LMC GPE is significantly higher than the other methods, therefore this method may still not be appealing.

4.5 Conclusion

In this Chapter we have discussed ways of emulating simulators which have multiple outputs. We split the multiple outputs to field type outputs and multiple-type outputs. Under each type of output we discussed one or two ways of emulating and compared the methods against each other and to fitting multiple univariate GPEs independently.

Under the field type outputs, we noted that these outputs are highly correlated and often high dimensional. We discussed two methods that are commonly used. The PC-GPE is an ideal way of reducing the number of univariate GPE's to fit and yet maintain a high level of accuracy. We also discussed the separable GPE which performed well in terms of RMSE, however it was under confident about its predictions.

Under the multiple-type outputs, for uncorrelated outputs, we discussed using the LMC GPE. There was not enough evidence to suggest that the LMC GPE is better than the independent GPE even though the theory suggests otherwise. Potentially, if we are able to help the optimiser by giving it sensible starting values, then the LMC may be able to select better estimates for the hyperparameters and it may take less time to fit a LMC GPE.

Overall, when it comes to deciding which method to choose when emulating simulators which have multiple outputs, it is advisable to categorise the type of outputs into either field type outputs or multiple-type outputs. Once categorised, a test can be carried out on a small subset of the data to see which method under the chosen category is most accurate for that particular simulator, bearing in mind what the end goal is (marginal versus joint predictions). This way a more informed decision can be made on the choice of methodology. Note that the time taken to fit the GPE may also be a factor that would need to be taken into consideration.

Chapter 5

Design Selection

5.1 Introduction

For a given set of events, $X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, that are sampled from a certain distribution of interest, we want to evaluate $\eta(\boldsymbol{x}_1), \ldots, \eta(\boldsymbol{x}_N)$, where $\eta(.)$ represents a computationally intensive simulator. However, we are restricted to evaluating only a limited subset of these events through the simulator due to time or computational constraints. A GPE will be used to predict the response for the events where we cannot afford to run the simulator. In this Chapter we discuss various methods for selecting the "best" representative subset of events, referred to as *design*, D, from the set of all events X.

Due to the limit on number of events that can be run through the simulator, we propose methods to fine tune the selection of the design. We do this by first defining some criterion that describes a good design, and then selecting the design such that it best satisfies the criterion. In this Chapter we look at three design criteria. We compare the results of the performance of the GPE fit to data selected using each of the suggested methodologies in a case study. We end the Chapter with a discussion on design selection methods in a more generic context.

5.2 Common design selection methods

In general, in order for the GPE to give robust predictions, it is important that the design events are selected in a way that ensures they are well spread out, preferably as far as possible from each other, covering the entire input variable space, (Iooss et al., 2010). There are many different ways of selecting design points.

The regular grid method, used in the LUT approach explored in Section 3.6, does not work well with GPEs because of its collapsing property. This is when multiple points have a fixed coordinate value when projected onto a variable axis, (Camus et al., 2011b). Moreover, with the regular grid method, the number of events in the design increases rapidly as we increase the dimensions.

A popular design selection method for GPEs is Latin Hypercube Sampling (LHS), see for example (Urban and Fricker, 2010), where a set of design points are selected subject to an input probability density constraint that ensures that across each input dimension, values are evenly spread. It is similar to the regular grid method, in that a single value is selected within each defined grid (or hypercube), however, unlike the regular grid method, no two design events have the same value for a single parameter. Urban and Fricker (2010) compare the Latin hypercube and grid ensemble designs for a multivariate GPE. They discuss the advantages and disadvantages of each of the methods. They recommend that although the grid design selection may be appropriate in some cases when looking at sensitivity analysis, the Latin hypercube designs should be used over the grid ensemble designs when the primary concern is to use the GPE for prediction. Iooss et al. (2010) discuss that as LHS is merely a form of stratified random sampling, it is not related to a particular criterion, and therefore the GPE prediction may have poor accuracy. Further enhancements have been applied to LHS to adopt an optimality criterion such as entropy, maximin and minimax distances. More information about these designs can be found in Morris and Mitchell (1995); Johnson et al. (1990); Jones and Johnson (2009); Oakley (1999).

Other approaches that have been considered in the past are sequential design selections. This is when a number of events are selected to run initially, a GPE is fit to these events. Then predictions are made on the simulator output, and further events are selected based on the predicted response to minimise (or maximise) a certain criterion. Typically this criterion is such that the events chosen in the iterative steps are those with highest prediction variance. Such designs can improve the performance of the GPE in a very efficient way, (Iooss et al., 2010), but of course they have to be combined with another design selection method to make the initial choice of design events. Sobol sequences are a common technique under this kind of design selection and are advantageous over Latin hypercube design because they can be built sequentially, (Caflisch and Morokoff, 1994). However, they do not guarantee that each dimension will have a uniform selection of events.

In this thesis, we assume that the data used here cannot be defined by closed-form expressions of probability density. In other words, we assume that we already have a set of predetermined events where we would like to run the simulator. Other combinations of parameter values for an event may not be a plausible scenario to model. Therefore, the Latin hypercube approach is less appropriate for our application.

As an alternative, in this thesis, we explore the maximum dissimilarity algorithm (MDA) approach as described by (Camus et al., 2011a), and as applied in the context of coastal analysis by (Camus et al., 2011b; Gouldby et al., 2014). This algorithm analyses the events using a measure of the distance between points in the multidimensional space. Having normalized the input variables, and given an initial event (the starting point), the MDA selects the next point that is the furthest away in Euclidean distance in the multidimensional space. This method outputs a set of design points which efficiently represents all the events in X.

Selecting the "best" design

Selecting the "best" design depends on how the user would describe best. Would they like to minimise the error over the entire input space? Would they like to minimise the posterior variance over the input space? Would they like simply to classify events and the actual value of the output is not as important? It is evident that selecting the best design is quite subjective and depends on what the user would like to use the GPE predictions for. It is important that these are clear before selecting a design selection method and ultimately a design.

In most applications, it may be of interest to minimise the root mean square prediction error of all the events in X. Mathematically, the criterion we are trying to minimise can be written as:

$$\sqrt{\frac{\sum_{i=1}^{N}\Big(\eta(\boldsymbol{x}_i) - E\big[\eta(\boldsymbol{x}_i)|D\big]\Big)^2}{N}},$$

where $\eta(\boldsymbol{x})$ represents the simulator output and $E\big[\eta(\boldsymbol{x}_i)|D\big]$ denotes the posterior mean prediction of an event derived from a GPE fitted using the design, D. In this case, the design we select would include events that are well spread out and efficiently cover the entire event parameter space. The design selection method that we propose for this criterion is the MDA technique.

In other applications, it may be of interest to minimise the RMSE error on a pre-determined selection of events. In this case the design may include more events that would fall into the pre-determined selection and may be more clustered compared to a design selected using MDA. The design selection method that we propose for this criterion is a weighted MDA. This is similar to MDA but has a weight associated with each event. The Euclidean distance between two events is then multiplied with the weights associated to the events. The weighted distance of two higher weighted events will appear to be larger than that of two lower weighted events given a fixed

Euclidean distance between them.

Finally we also look at an application where it may be of interest only in the correct classification of events. This applies to scenarios where the accuracy of the predicted simulator output is not important, but the classification is. For this case, we propose a sequential design where first we select a small amount of events (using MDA for example), fit a GPE to it. Then we use the GPE to predict the simulator output for all other events, and select design events based on the uncertainty around its **classification** (using both the mean and standard deviation from the GPE predictions).

In the Sections that follow we consider these three different applications and explore the associated design selection method in more depth. Moreover we present a case study to compare the proposed design selection methods with randomly selected designs.

5.3 Criterion 1

In this Section, we consider the most common criterion of design selection. We assume that the modeller would like to accurately predict the outcome for every event. In this case, every event is equally important regardless of the input or output value(s) of the event. An example of this scenario would be when the modeller would like to model the entire wave climate for a given number of years.

In this application we are trying to minimise the following criterion:

$$\sqrt{\frac{\sum_{i=1}^{N}\Big(\eta(\boldsymbol{x}_i)-E\big[\eta(\boldsymbol{x}_i)|D\big]\Big)^2}{N}}$$

where D will be the design selected to fit the GPE.

Camus et al (2011) use MDA to select a representative set of events. They found that the MDA selects events diversely distributed over the data, exploring the edges of the parameter space and sampling a variety of events, which in turn improves the emulator.

5.3.1 Maximum dissimilarity algorithm (MDA)

MDA is a sampling technique that is based on the Euclidean distance in multi-dimensional space from each event. Given an initial event, the algorithm chooses the remaining events iteratively based on the events that are the furthest away in Euclidean distance from all of the events that have already been selected.

More formally, for a given sample of data $X = \boldsymbol{x}_1, \dots \boldsymbol{x}_N$, we want to select n events that represent our design, $\mathcal{D}$. Let $\mathcal{D}_R = \{\boldsymbol{d}_0, \dots, \boldsymbol{d}_R\} \in X$ contain the selected events at iteration R in the selection process. The vector $\boldsymbol{d}_0$ is pre-defined by the user as an initial starting event(s).

At iteration R, the dissimilarity between an event i in the data sample $X_R = X \backslash \mathcal{D}_R$ and the j-th event in the subset $\mathcal{D}_R$ is calculated as follows:

$$\boldsymbol{diss_{i,j}} = \|\boldsymbol{x}_i - \boldsymbol{d}_j\| \quad \text{for} \quad i = 1, \dots, (N-R) \quad \text{and} \quad j = 1, \dots, R, \tag{5.1}$$

where

$$\|\boldsymbol{x}_i - \boldsymbol{d}_j\| = \sqrt{\sum_{k=1}^{p} (x_{i,k} - d_{j,k})^2},$$

and $\|.\|$ denotes Euclidean distance. Subsequently, the dissimilarity $\boldsymbol{diss_{i,\mathcal{D}_R}}$, between the vector i and the subset $\mathcal{D}_R$, is calculated as:

$$\boldsymbol{diss_{i,\mathcal{D}_R}} = min\{\boldsymbol{diss_{i,j}}\} \quad \text{for} \quad i = 1, \dots, (N-R) \quad \text{and} \quad j = 1, \dots, R.$$

Finally, the point that is selected is the point with largest dissimilarity such that

$$\boldsymbol{d}_{R+1} = \boldsymbol{x}_{max\{\boldsymbol{diss_{i,\mathcal{D}_R}}\}}.$$

Conceptually, calculating $\boldsymbol{diss_{i,j}}$ for $i = 1$ and $j = 1, \ldots, R$ gives the Euclidean distances between all the selected points ($\mathcal{D}_R$) and x_i. To select the point that is furthest away from all the previously selected points we must maximise the minimum distance between the selected points and the point that is chosen.

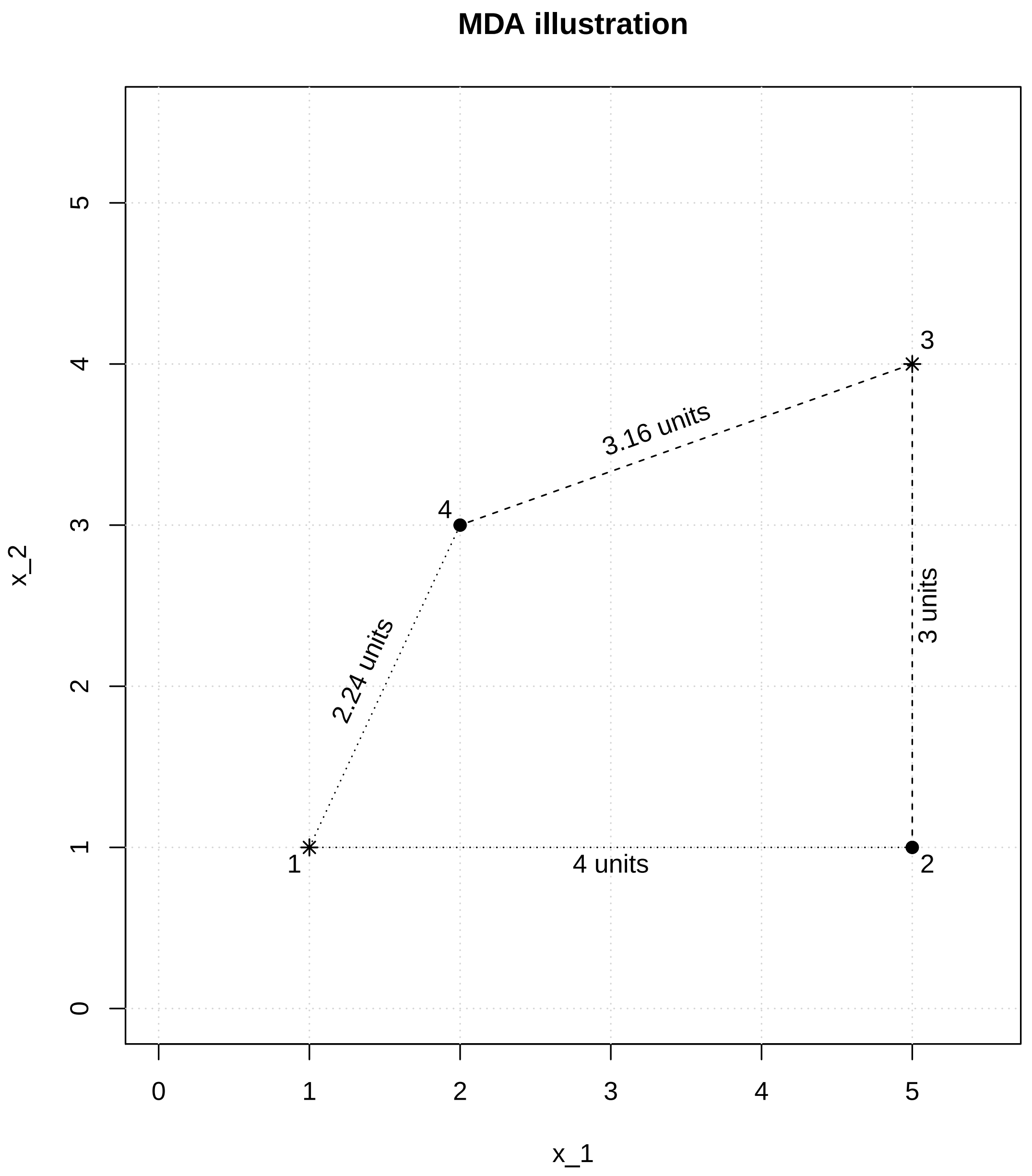

Figure 5.1: Explaining the MDA diagrammatically - Having selected points 1 and 3, the next most dissimilar point to be selected is point 2.

Figure 5.1 shows an illustration of how MDA works using an example. Suppose we have already selected the points marked with an asterix (points 1 and 3). At the current iteration we have to select the most dissimilar point that has not yet been selected (point 2 and 4). We calculate the distances between point 1 and point 2 and 4, and the distances between point 3 and point 2 and 4. These are indicated in the figure using dotted lines and dashed lines respectively with the calculated distances displayed nearby. With each unselected point, we note the selected point that is closest. Point 1 is closest to point 4 with a distance of 2.24 units and point 3 is closest to point 2 with a distance of 3 units. The point that is chosen is the point that has the largest distance to the closest point. Therefore, point 2 is selected.

Before applying the MDA technique to a dataset, we need to make sure that $\{x_1, \ldots, x_p\}$ have the same unit of measure. One way to overcome this is by normalising the data. This will ensure that equal weighting is applied to each distance.

In the SWAN model, at least one of the parameters is a directional variable ranging between 0° and 360°. When calculating the distance for this variable we have to ensure that we take the smallest angular distance between the two points. To do this, Camus et al (2011) suggest using a Euclidean-circular (EC) distance where the 'E' is for the Euclidean distance in the scalar variables and 'C' is for the circular (or angular) distance in directional variables. The scalar variables are normalised between 0 and 1 using the formula below:

$$\frac{x_i - min(x)}{max(x) - min(x)} \qquad for \quad i = 1, \ldots n.$$

For the directional variables, we note that the maximum difference between any two directions x_i and x_j, $max(x_i - x_j) = 180°$, while $min(x_i - x_j) = 0°$. Therefore the normalisation formula from above reduces to $\frac{x_i}{180°}$ for directional variables. This effectively rescales the directional variables between $[0, 2]$ such that the circular distance between any two directional variables would range between $[0, 1]$.

Following this, if, for example, we want to calculate the EC distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ where $\boldsymbol{x}_i = [h_i, \theta_i]$ and h denotes normalised wave height ranging between [0,1] and θ denotes normalised wave direction ranging between [0,2]. Then the EC distance is calculated as:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{(h_i - h_j)^2 + min\Big(|\theta_i - \theta_j|, (2 - |\theta_i - \theta_j|)\Big)^2}$$

and $\| \quad \|$ denotes the EC distance.

Another solution to the problem associated with directional variables would be to convert the direction, θ, to vector components in x (horizontal) and y (vertical) directions such that

$$\mathrm{x} = \cos\left(\frac{2\pi\theta}{360}\right) \qquad \text{and} \qquad \mathrm{y} = \sin\left(\frac{2\pi\theta}{360}\right).$$

This will result in x and y values ranging between 0 and 1. However, a disadvantage of this method is that for every directional variable you have, you will have one extra parameter in the input space. This may affect your model if you already have many parameters. Nevertheless if you only have a few parameters, then we found that this method works better in terms of performance of the GPE.

5.4 Criterion 2

In this Section, we consider another criterion for design selection. We assume that the modeller would like to accurately predict the outcome for certain events more than others. These events have to be pre-determined. Let's suppose that for $\boldsymbol{x}_i \in X, w_i$ represents the weighting associated to the event, where $w_i \in [0, 1]$. This weighting represents the modellers interest in the importance of accurately predicting event i, where $w_i = 1$ represents highest importance, and $w_i = 0$ represents the least importance. To account for this, we can expect that the design selection method will bias event selection to higher weight.

An example of this scenario would be when the modeller would be more interested in wave heights above a specified threshold (for instance where waves below 0.5 meters may not have any significant impact on a particular study). Another example would be when waves from certain directions may be assumed to have a higher impact to the offshore wave height due to the bathymetry for a particular site.

In this application, we aim to minimise the weighted Euclidean distances by minimising the following criterion:

$$\sqrt{\frac{\sum_{i=1}^{N} w_i \Big(\eta(\boldsymbol{x}_i) - E\big[\eta(\boldsymbol{x}_i)|D\big]\Big)^2}{N}}.$$

5.4.1 Weighted maximum dissimilarity algorithm (WMDA)

The design selection method we propose here is a slight modification of the MDA method defined earlier. The method we propose here is a weighted MDA (WMDA). This algorithm chooses points based on the *weighted* Euclidean distance in multidimensional space from each event.

The intention is that with the limited number of events that we can evaluate the simulator for, we want to make the best choice of design so that the events that we are most interested in (higher w_i) are as accurate as possible with a slight compromise on the accuracy of the lower weighted events.

Following the same setup as the MDA method, we calculate the dissimilarity between an event i in the data sample X_R and the $j-th$ event in the subset $\mathcal{D}_R$ as follows:

$$\boldsymbol{diss_{i,j}} = \|\boldsymbol{x}_i - \boldsymbol{d}_j\| \times w_i w_j \quad \text{for} \quad i = 1, \ldots, (N-R) \quad \text{and} \quad j = 1, \ldots, R, \quad (5.2)$$

where $\|.\|$ denotes the Euclidean distance.

Multiplying the Euclidean distance by a weight factor has the effect of lengthening or shortening the distance by factor w. If i and j are highly weighted events, then $\boldsymbol{diss_(i,j)}$ would be greater than if i and j were unfavourably weighted events. Assigning a weight of zero to an event is effectively saying that this event is not important at all and thus will not get chosen. The rest of the analysis remains the same as MDA.

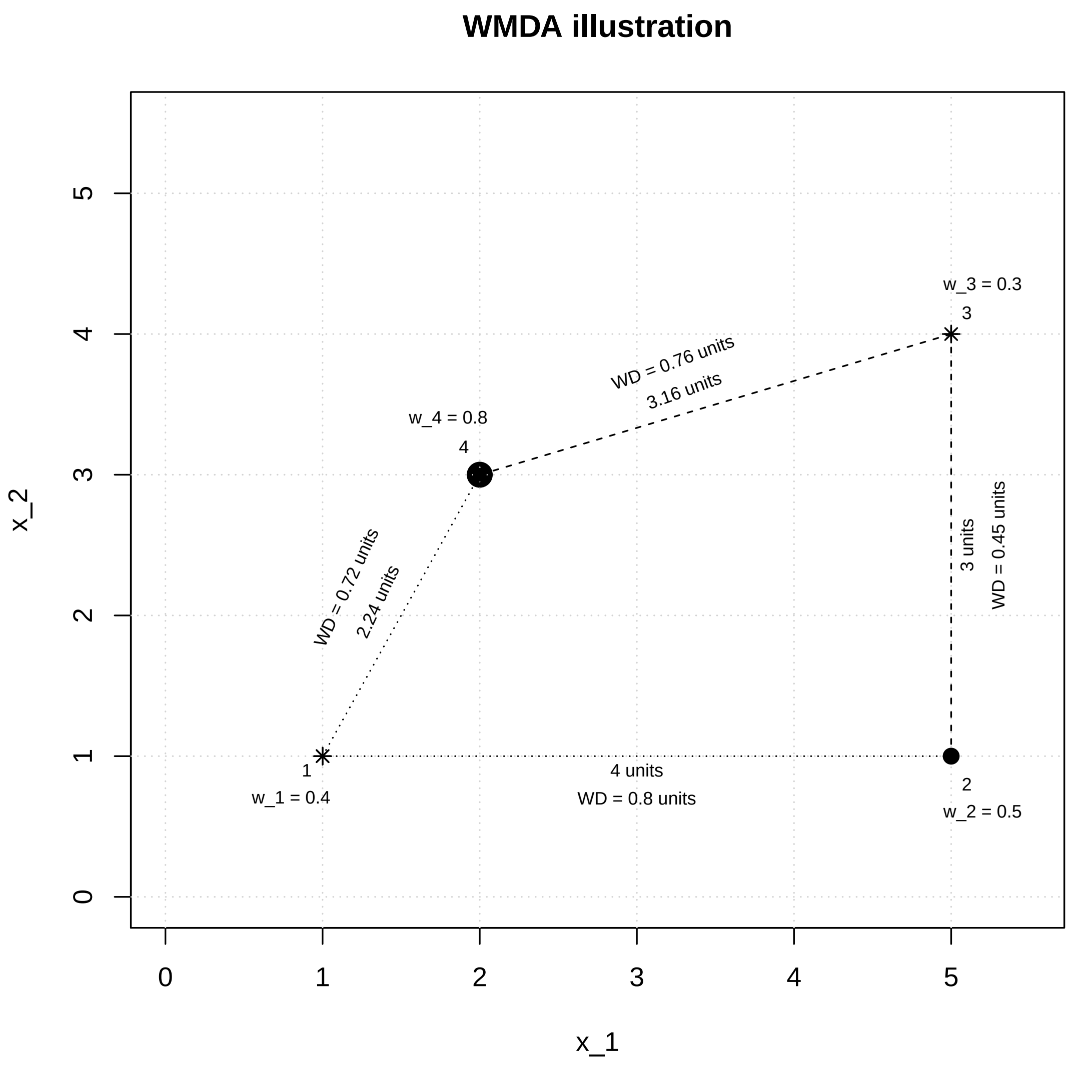

Figure 5.2: Explaining the WMDA diagrammatically - Having selected points 1 and 3, the next most dissimilar point to be selected using weighted distances is point 4.

Figure 5.2 shows an illustration of how WMDA works using an example following the same set up as Figure 5.1. We assign weighting to each of the points as indicated on the graph. Suppose we have already selected the points marked with an asterix (points 1 and 3). At the current iteration we have to select the most dissimilar point that has not yet been selected (point 2 and 4). The Euclidean distances between the points are shown on the graph. We also calculate the weighted Euclidean distances between the points (abbreviated as WD on the graph). This would be a multiplication of the weights of each point and the Euclidean distance between them. For each unselected point, we note the selected point that is closest. Point 1 is closest to point 4 with a weighted Euclidean distance of 0.72 units and point 3 is closest to point 2 with a weighted Euclidean distance of 0.45 units. The point that is chosen is the point that has the largest distance to the closest point. Therefore, point 4 is selected.

5.5 Criterion 3

In this Section, we assume that the modeller would like to accurately classify a set of events and where accurate predictions of the simulator output itself are unimportant.

An example of this scenario would be when the modeller is assessing a flood defence of a certain height. The modeller does not care about the specific value of the output wave height, but whether the wave height is higher than the height of the flood defense.

We present here a sequential design where we first propose to fit a GPE to a space filling design (e.g. MDA) for a small number of points. We then predict the output of the non design events using the fitted GPE. The idea is to update our design based on the uncertainty associated with the correct classification of the predicted events (by taking into consideration the posterior mean and variance).

Suppose a modeller is interested in correctly classifying whether the simulator output will lie within a certain interval or not. we define such an interval as $R =$

$[a, b]$. The criterion that we are trying to minimise is as follows:

$$\sum_{x' \in X \setminus \mathcal{D}} \left| \mathbb{1}_R(\eta(\boldsymbol{x}')) - \mathbb{1}_R(E\big[\eta(\boldsymbol{x}')|\mathcal{D}\big]) \right|$$

where

$$\mathbb{1}_R(\boldsymbol{z}) = \begin{cases} 1, & \text{if } \boldsymbol{z} \in R. \\ 0, & \text{if } \boldsymbol{z} \notin R. \end{cases}$$

5.5.1 Pivoted Cholesky decomposition

Formally, suppose that the main interest is in determining whether, for event $\boldsymbol{x}$, the simulator output $\eta(\boldsymbol{x})$ will lie in some interval $R = [a, b]$. We define a new function $g(\boldsymbol{x})$ such that:

$$g(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \eta(\boldsymbol{x}) \in R, \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

For a given sample of data $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, suppose that we have already selected an initial design, $\mathcal{D}_1 = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{n_1})\}$ with a small number of design events where $n_1 < n$. We run the simulator at these events, and use the output to fit a GPE. We then use this GPE to predict the simulator output at all events in X. Note that evaluating $\eta(\boldsymbol{x})$ will also give us a value of $g(\boldsymbol{x})$. For the second design in the sequence, the idea is to select the events where:

- we have the greatest uncertainty about $g(\boldsymbol{x})$
- and by avoiding events that are close together: if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close together, evaluating $\eta(\boldsymbol{x}_i)$ is likely to give us an accurate value of $g(\boldsymbol{x}_j)$ (in addition to $g(\boldsymbol{x}_i)$), because once we learn $\eta(\boldsymbol{x}_i)$, we will have little uncertainty about $\eta(\boldsymbol{x}_j)$.

Next, we obtain the variance-covariance matrix V of $\{g(\boldsymbol{x}_1), \dots, g(\boldsymbol{x}_N)\}$, and use pivoted Cholesky decomposition on V to find the event $\boldsymbol{x}_{(1)}$ where $g(\boldsymbol{x}_{(1)})$ has the largest variance, the event $\boldsymbol{x}_{(2)}$ where $g(\boldsymbol{x}_{(2)})$ has the largest variance conditional on $g(\boldsymbol{x}_{(1)})$ and so on. The rest of this Section explains how to calculate V.

From the emulator, for a non design event, $\boldsymbol{x}_i$ (not used to fit the GPE), suppose we have

$$\eta(\boldsymbol{x}_i) \sim N(m_i, s_i^2).$$

where $m_i = E\{\eta(\boldsymbol{x}_i)|\mathcal{D}_1\}$ and $s_i^2 = var\{\eta(\boldsymbol{x}_i)|\mathcal{D}_1\}$. We define $g(\boldsymbol{x}_i)$ to have a Bernoulli distribution with probability p_i as follows:

$$g(\boldsymbol{x}_i) \sim Bernoulli(p_i),$$

where

$$p_i = \Phi\left(\frac{b - m_i}{s_i}\right) - \Phi\left(\frac{a - m_i}{s_i}\right)$$

and Φ is the CDF of the standard normal distribution. We can compute

$$Var\{g(\boldsymbol{x}_i)\} = p_i(1 - p_i)$$

to give us the diagonal elements of V. For the off-diagonal elements of V we need

$$\begin{aligned} Cov\{g(\boldsymbol{x}_i), g(\boldsymbol{x}_j)\} &= E\{g(\boldsymbol{x}_i)g(\boldsymbol{x}_j)\} - E\{g(\boldsymbol{x}_i)\}E\{g(\boldsymbol{x}_j)\} && (5.4)\\ &= E\{g(\boldsymbol{x}_i)g(\boldsymbol{x}_j)\} - p_i p_j, && (5.5) \end{aligned}$$

where

$$\begin{aligned} E\{g(\boldsymbol{x}_i)g(\boldsymbol{x}_j)\} &= P\{g(\boldsymbol{x}_i) = 1, g(\boldsymbol{x}_j) = 1\} && (5.6)\\ &= P\{\eta(\boldsymbol{x}_i) \in [a, b], \eta(\boldsymbol{x}_j) \in [a, b].\} && (5.7) \end{aligned}$$

This probability can be calculated using a bivariate normal CDF with

$$\begin{pmatrix} \eta(\boldsymbol{x}_i) \\ \eta(\boldsymbol{x}_j) \end{pmatrix} \sim N\left(\begin{pmatrix} m_i \\ m_j \end{pmatrix}, \begin{pmatrix} s_i^2 & c_{i,j} \\ c_{i,j} & s_i^2 \end{pmatrix} \right),$$

where $c_{i,j}$ is the emulator posterior covariance between $\eta(\boldsymbol{x}_i)$ and $\eta(\boldsymbol{x}_j)$.

Now that we can define V, we calculate the pivoted Cholesly decomposition of V. This will result in a ranked order matrix of events that have the largest variance conditional on the previous events. We select the top $(n - n_1)$ events that we need to update our design.

Although this method appears theoretically sound, when implementing it to a large dataset we faced severe computational burdens and hence were not able to implement in to the case study that follows. However simpler test cases have proved that there is definitely some scope for this method where a modeller is faced with such a scenario.

5.6 Case Study

The aim of this case study is to compare design selection methods. We explore the design selection methods using the context of case study 1 as described in Chapter 2.2.1.

The design selection methods that we are going to compare are: two randomly generated designs (abbreviated as RG1 and RG2), one design generated using the MDA method (abbreviated as MDA), and one design generated using the WMDA method (abbreviated as WMDA). We will try to minimise criterion 1 to illustrate how the MDA compares to RG1 and RG2 and simultaneously try to minimise criterion 2 to see how WMDA compares to MDA, RG1 and RG2.

We evaluate the SWAN model at the designs obtained by each design selection method. Next, we fit separate GPEs to each of the selected designs for each SWAN output. We use these fits to predict the simulator output for a fixed subset of events. We compare the design selection methods by comparing the performance of each GPE in predicting the events.

The basic setup is as follows. We have a full set of events, $N = 69494$ events where we would like to evaluate the SWAN model. Suppose that we are limited to only $n = 500$ SWAN model runs. The input variables of SWAN are wind speed, wave height, peak period, wind and wave direction. The output variables are nearshore wave properties described with wave height (Hsig), wave period (Tm_{02}) and wave direction (Dir). We have true SWAN output for approximately 5600 events which we will use for assessing the design selection method. For MDA and WMDA, we initialise the algorithm at the event where the wave height is the highest.

Note that two of our input variables and one output variable are directional variables: wave direction, wind direction and nearshore wave direction. We calculate the x and y components for each variable as discussed in Subsection 5.3.1. It follows that our input space is now increased by 2 variables from five to seven. This method of handling the directional variables was selected as it gave a better RMSE for initial tests.

For the design of coastal structures, in general, the higher wave conditions are likely to be of more relevance. Assuming that higher offshore waves lead to higher waves at the site of interest, rather than proceed with the MDA algorithm, it is desirable to focus the design points to cover higher wave conditions, i.e. targeting increased resolution of design points towards an area of the input parameter space more likely to result in higher wave heights. This can be achieved with the WMDA approach by weighting high waves more than low waves. Moreover the selected wave prediction point (as shown in Figure 2.1) is sheltered to waves from several directional sectors. Therefore, it would not be as useful to run SWAN model for the events which have these wave directions as the SWAN output may result in low wave conditions. As a result higher weights were assigned to each event proportional to wave height and to events where the offshore wave direction was between 170 to 200 and 300 to 330 degrees. Figure 5.3 shows the weighting applied according to offshore wave height and offshore wave direction.

It is important to note that the weighting can only be based on the input variables as the SWAN output will not be available. Additionally note that the stronger the weighting, the greater the effect WMDA will have to bias the selection of events to the higher weighted events. Note this effect is also based on how correlated the input variables are to the output variable of interest.

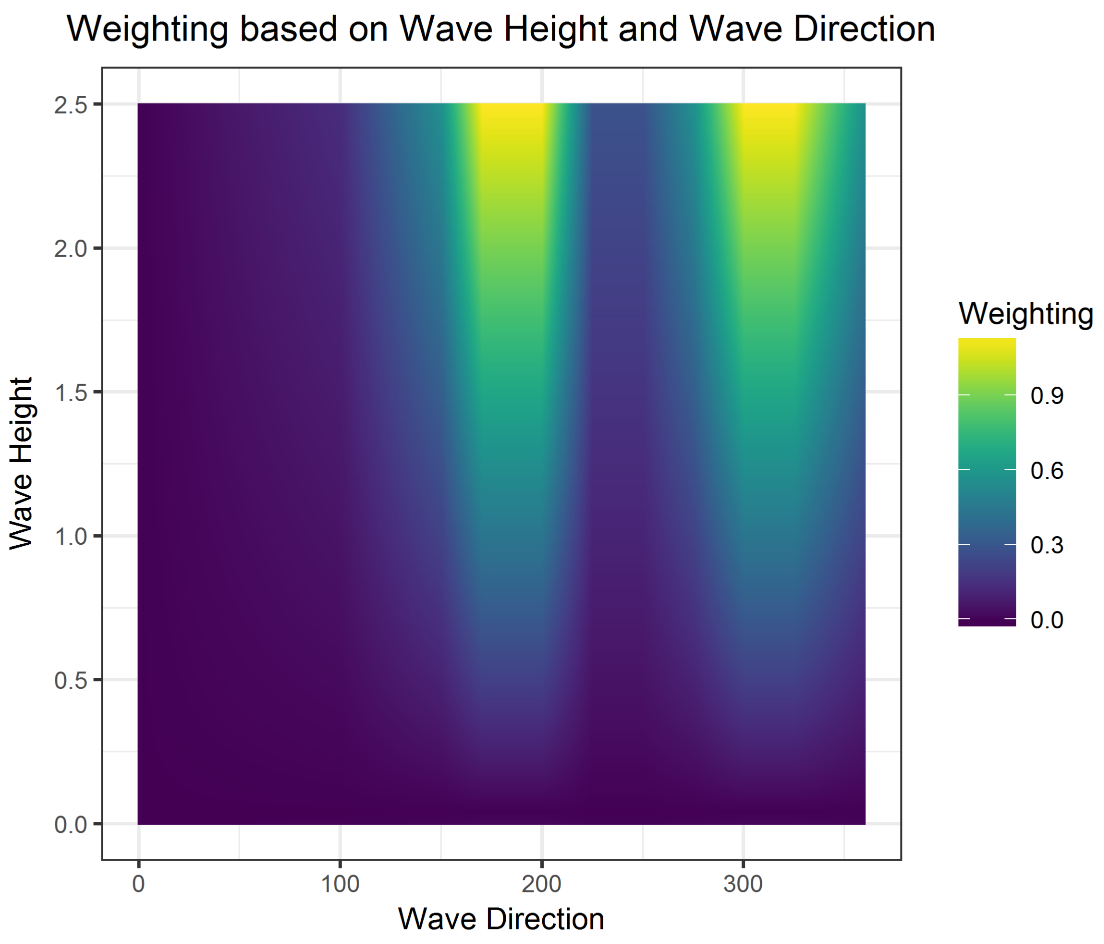

Figure 5.3: Weight assigned to each event based on the events input wave height and wave direction values.

5.6.1 Results and discussion

Figure 5.4 shows the design for the two randomly selected design points, while Figure 5.5 shows the design selected using MDA and WMDA. In both figures the blue dots represent the entire set of events where we would like to evaluate SWAN, while the grey dots represent the selected design events. Notice how for RG1 and RG2 the selected events are mainly concentrated around the more dense regions of the data, as the events have a higher chance of getting picked. Also notice that there is a large variation between the two designs (RG1 and RG2). This is due to the random element in the method.

On the other hand notice that for the MDA method, the selected events are more spread out and cover the entire space of the data for all parameters. Each time MDA (or WMDA) is used to select a design with the same initial conditions and variables, the resulting design is identical. For the WMDA method, the selected events are concentrated where the wave height is bigger and the wave direction is between 170 to 200 and 300 to 330 degrees.

For the GPE, the closer the prediction event is to the design event, the more accurate the prediction. Thus the design for RG1 and RG2 might work well with smaller n as most of the prediction events are centred around the design, but as we increase n, RG1 and RG2 will continue picking points that are in the denser regions. This is where MDA is better as it ensures that the event it chooses at each iteration is the furthest away from all the selected events, thus effectively exploring the entire input parameter space.

Once we have selected the design and run these through the SWAN model, the next step is to fit the GPE to the events. We fit a separate GPE for each design and each output variable. After fitting the GPEs, we can use it to predict the output of the model at the validation events. We compare the performance of predictions made using the four designs.

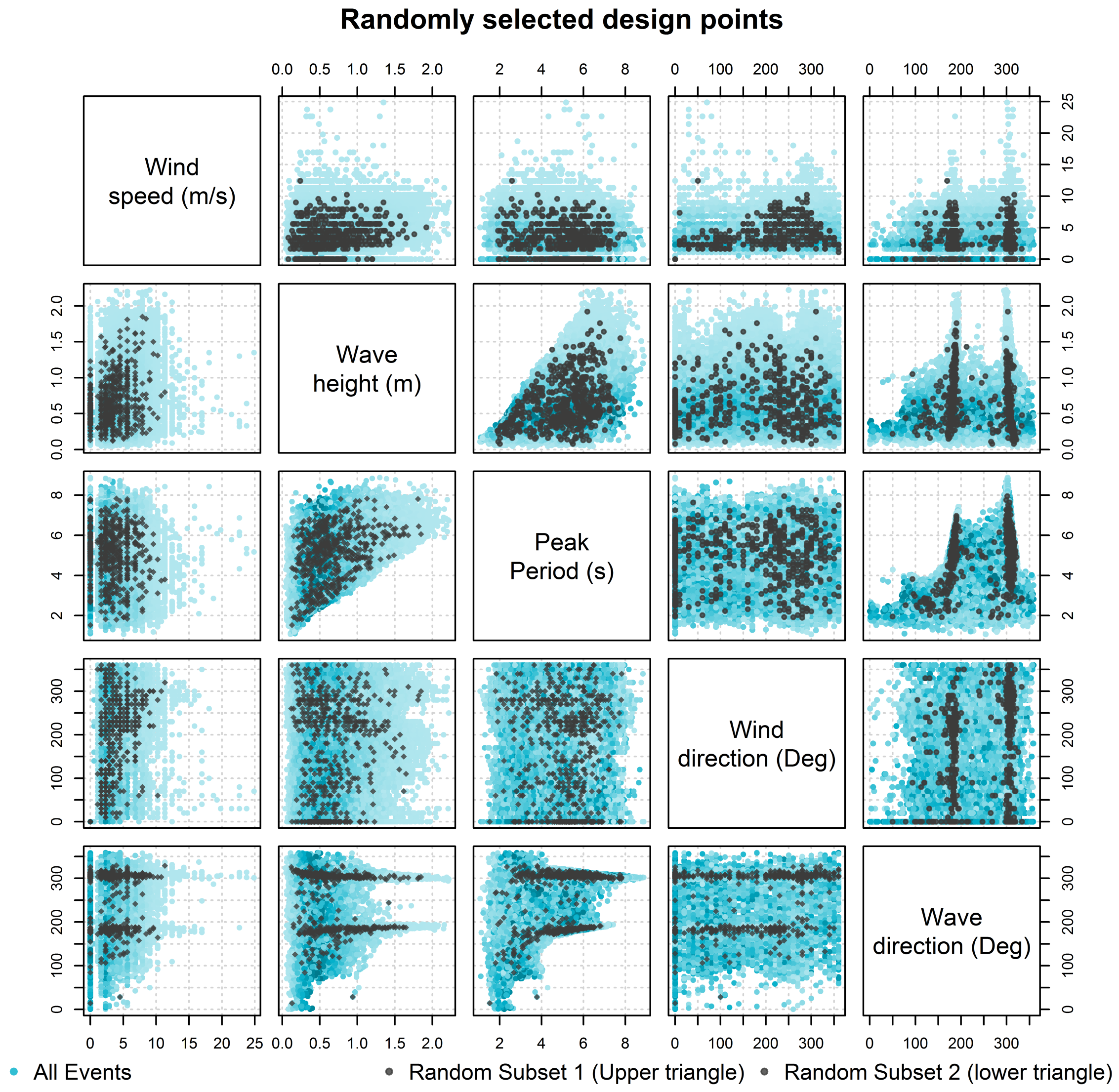

Figure 5.4: Designs selected using a random selection (RG1 and RG2)

In order to minimise criterion 1, we can look at the RMSE error. Figure 5.6 shows how the NRMSE varies as we increase the number of design events used to fit the GPE for each of design selection methods. Notice in this column that where n is small, all the methods seem to be predicting the SWAN output with similar accuracy. On some occasions the random designs seem to be better than the MDA. However, as you increase n, the MDA (and WMDA) method outperforms RG1 and RG2. This is because the MDA method ensures that the design events

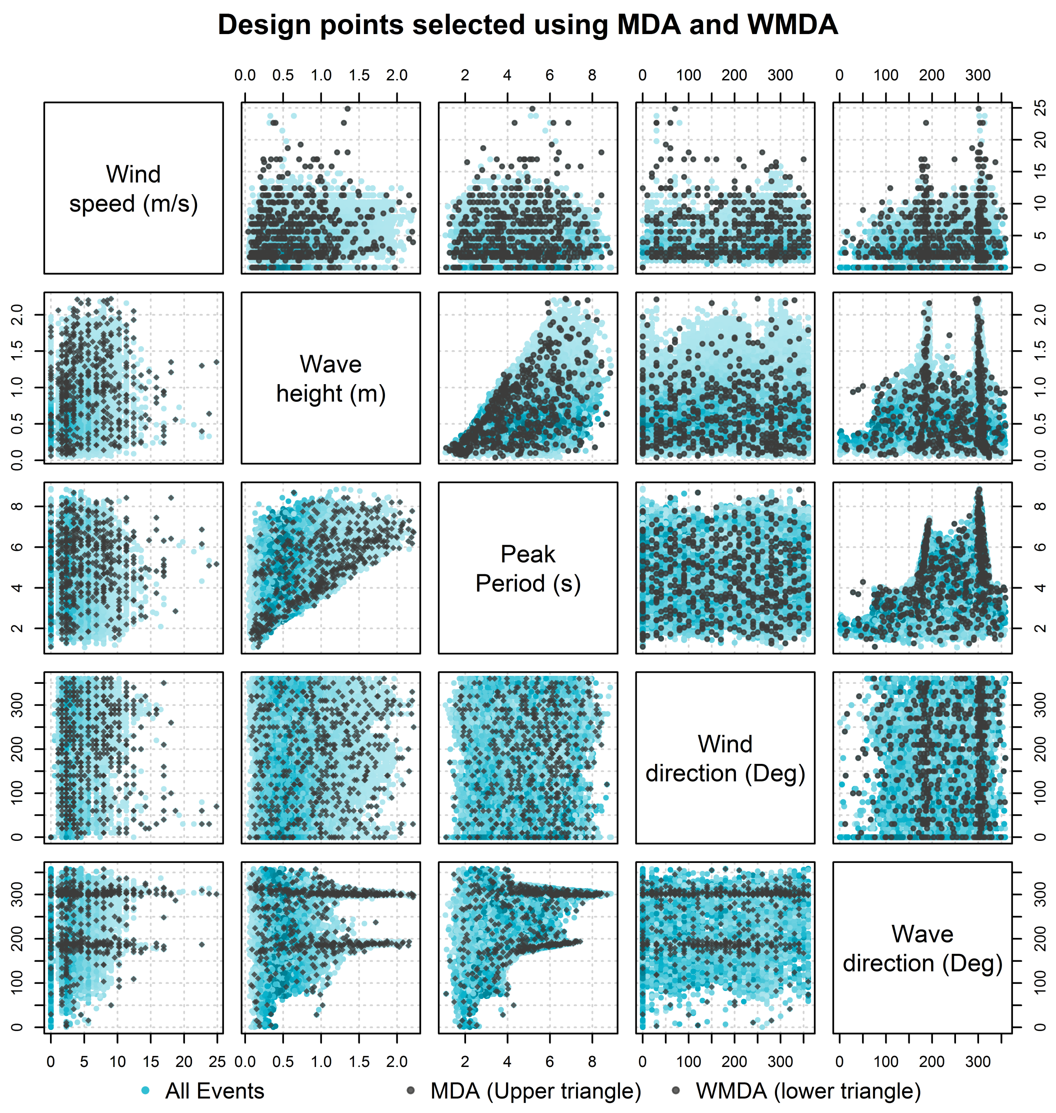

Figure 5.5: Designs selected using MDA (upper triangle) and WMDA (lower triangle) designs

are as far away from each other, whereas for the random designs, the events which are clustered together have a higher chance of being picked and thus most of the events may be close to each other. Therefore, adding more events does not seem to be informative for the GPE.

In general, the NRMSE of the MDA method follows a downward trend as you increase the number of training runs. This is not consistently the case for RG1 and RG2 especially for wave height and wave period. The overall performance for the

RG1 and RG2 is inconsistent and unreliable compared to the MDA method.

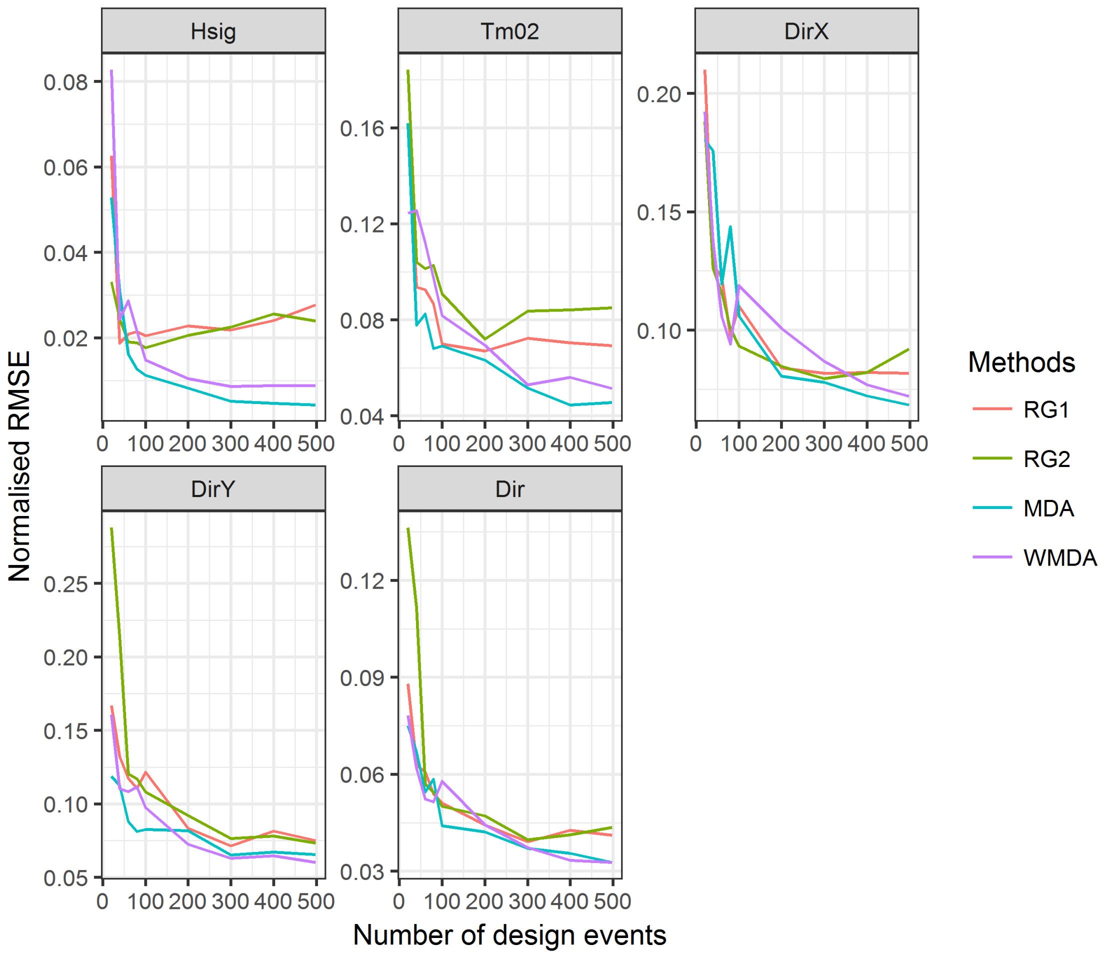

Figure 5.6: Comparison of NRMSE of GPE's fitted to different designs for varying sizes of designs

In order to minimise criterion 2, we can look at the weighted RMSE error which is calculated as:

$$\sqrt{\frac{\sum_{i=1}^{N}\frac{1}{w_i}\Big(\eta(\boldsymbol{x}_i) - E\big[\eta(\boldsymbol{x}_i)|D\big]\Big)^2}{N}}.$$

Figure 5.7 shows a plot of the weighted RMSE against the number of design events. The random design selection methods are clearly worse off than the other methods. The MDA and the WMDA method follow similar patterns. In this case, the MDA method outperforms the WMDA for Hsig and Tm02. This may be because of the low correlation between the input wave height and the output wave height and wave period for this particular nearshore point that we are predicting at. We can see that for output wave direction (Dir), the WMDA performs better than the MDA for most of the number of design events. On average, for wind direction WMDA can achieve the same accuracy as MDA with fewer (about 100) design events. In a more realistic use case, this gain in accuracy in the higher weighted regions would be very significant.

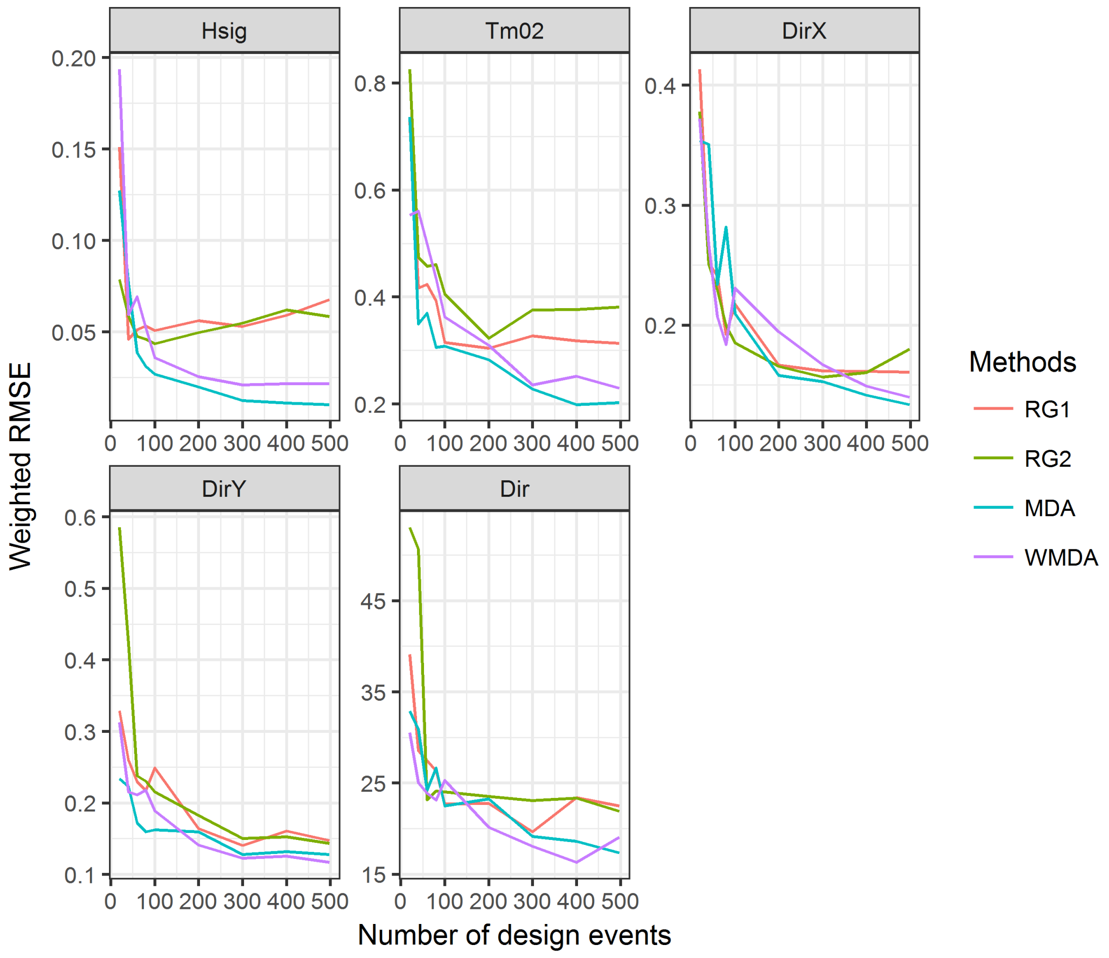

Figure 5.7: Comparison of weighted RMSE of GPE's fitted to different designs for varying sizes of designs

Unfortunately, in this case study, we cannot demonstrate the benefits of the WMDA design selection method as well as we intended. In the paper, (Malde et al., 2018), we show another case study of WMDA where it outperforms the MDA.

In general, the effect of weighting depends on how well the assumptions between the input and output correlations are understood. These often vary case by case and thus is difficult (without expert knowledge) to appropriately assign a weight to each event in advance of running the simulator. A more appropriate weight could be assigned by using a sequential approach. A standard design selection method (MDA) could be applied at first to choose a small number of design events. The simulator can be run for these events, and a GPE can be fitted to the simulator input and output. A prediction of the simulator output can be made at all other events. This may be a poor prediction, but it can be used to get an understanding of the input and output correlation structure. This can help with assigning more appropriate weights to each event.

Other factors that we have found to affect the performance of WMDA in comparison to MDA, is how strong a weight applied to a particular event is in comparison to the weighting of the remaining events. In general, the stronger the weight, the higher the chances of it being picked. However, this may have adverse effects on the performance of predicting the lower weighted events.

5.7 Conclusion

In this Chapter we have identified three different criteria that a user maybe interested in achieving. For each criterion, we have presented a design selection method that would be appropriate to implement. In the case study, we showed that the MDA method resulted in the better GPE predictions then all the other methods. We showed an example of a case where WMDA can be applied, and although we did not prove that it was better than the MDA method, we referenced a paper which goes through a similar comparison and shows that it can be. The performance of WMDA is heavily dependent on the assigned weights and the assumed relationship between the inputs and outputs.

In this thesis, we are limited to a design selection method that allows us to select a design from a predetermined set of events. However, without this restriction there are many design selection methods available. The way to choose to the most appropriate method will depend on the specific use case of the final output. This objective should be defined earlier in order to target events appropriately when selecting the design.

In general, a good design is one where, events are well spread out in all input dimensions; events cover the extent of the input parameter space (such that we are interpolating when predicting events), and where design is representative of all events.

It is also important to consider how the simulator is going to be run. By this we mean, is it going to run numerous runs in parallel? If so running one simulator run would take the same time as running multiple. It may be worth taking this into account when selecting the design to avoid "wastage of resources." This may also be used to determine how many runs to select for each step in a sequential design selection method.

Chapter 6

Conclusion

A majority of the coastal modelling applications require using historic data from physical observations or from computer simulations. A variety of simulators exist and range in complexity and accuracy of approximating real world scenarios. In general, the really accurate simulators are often very complex and take a long time to complete a single run and require a lot of computer resources. Ideally for a complete study, a large quantity of data is required. If this is to be derived from the simulator, it can often be challenging. Hence companies such as HR Wallingford, have adapted meta models to help reduce the burden.

In this thesis we have considered the application of Gaussian process emulators (GPE) to coastal wave modelling. We have explored different types of GPEs and applied it to a variety of applications in the case studies presented. We used a wave transformation model (SWAN) as the simulator that we fit the GPE to, however the details discussed in this thesis are easily applicable to other simulators which are computationally expensive and time consuming to run.

We choose to use GPEs compared to other meta-models as it has two significant advantages. Since GPE uses a Bayesian approach in modelling, the predictions of the simulator are in terms of a distribution. This means that we can get point wise predictions and credible intervals around these predictions which represent the GPEs uncertainty around the predicted value. Moreover using the GPE to predict at an event that was used to train it, will give 100% accurate predictions with zero uncertainty. Additionally, although not explored in this thesis, GPEs can also be used to carry out uncertainty and sensitivity analysis much more efficiently.

In this thesis we first introduced a univariate GPE in Chapter 3 where we used a toy simulator to explain the mathematical details. In the case study we illustrated the benefit of using GPEs over the traditional look up table approach that was used. We showed that the GPE approach required much fewer number of simulator runs and resulted in higher accuracy than the look up table approach. We note that limited expert knowledge about the simulator is required. This was not the case in the look up table approach.

In the next Chapter we extended the basic univariate GPE to applications where the simulator has multiple outputs. We identified two types of multiple output and illustrated the methodology to emulate the simulator for each type. For high dimensional and strongly correlated outputs, we illustrate the use of principal component analysis to reduce the dimension of the output and then fit the GPE to the reduced number of principal components. We showed that this method works well if the output is highly correlated. We reviewed an alternative multivariate GPE approach referred to as the separable GPE. The limitation with this method is that it only uses a single correlation length parameter amongst all the outputs, which tends to be quite restrictive.

For low dimensional output that is not strongly correlated we explore the linear model of coregionalisation (LMC) GPE. This is more flexible than the separable GPE as the output processes are constructed as linear combinations of independent univariate GPEs. Therefore this method allows a different set of correlation length parameters for each output. A limitation to this method is the number of parameters it has to estimate using maximum likelihood estimation. This can result in the optimiser falsely converging at a local maxima/minima which can lead to a poor GPE. There is scope for improving the computer implementation of this methodology in terms of making it more efficient and robust to optimise the hyperparameters.

In separate case studies we demonstrated which method would be more suitable depending on the type of output. We concluded that it is best to carry out a small test case where the methods are compared based on what the end goal is. In general, if the interest is in using just the marginal point predictions of the output variables, then it may suffice to use multiple univariate GPEs independently if the number of outputs is not too large. Otherwise, principal component analysis can be used. However, if the interest is in using a joint distribution then it may be better to use the separable or LMC GPE depending on how similar the output variables are to each other.

An important aspect of fitting a GPE to a simulator is the design selection. In Chapter 5 we discuss ways of selecting a design such that we maximise the information we can get from the simulator to inform the GPE when we are limited to a number of simulator runs. In this thesis we are limited to selecting a design based on a set of pre-determined events. We discuss three approaches to select a design based on three different criteria we may want to optimise over. However, without this limitation there are a number of other methods that can be applied. In general an objective should be defined prior to choosing the design selection method. For instance, an objective could be to minimise the RMSE of the predictions.

Knowledge Transfer Partnership (KTP)

This thesis was written as part of the KTP program. The objective of this program, which was to research and implement advanced meta-modelling approaches has been successfully met. At HR Wallingford, there has already been a noticeable reduction in computational burden and an improvement of accuracy in approximating the simulator compared to their traditional methods. Moreover, at HR Wallingford, there has been an interest in other applications where GPEs can be applied. These are explained in more detail in the next section.

6.1 Other applications of GPE at HR Wallingford

6.1.1 Bayonet GPE

Together with other colleagues at HR Wallingford, we have designed an overtopping model using GPE which will be used industry wide, (Pullen et al., 2018). The new model offers several advantages over the previous neural network based model. It provides better accuracy and a more explicit and complete representation of uncertainty. Moreover the predictions can be obtained instantaneously. The design used here consisted of all the data from an existing database of physically modelled results.

GPEs have zero variance at the design points and away from the design points the variance is non-zero, realistically reflecting uncertainty through consideration of distance from the design points. Where it is known the data themselves contain uncertainty (noise), as is the case with data from physical model experiments, this is included within the model fitting through an error term known as a “nugget”, (Andrianakis and Challenor, 2012). The output predictions from the GPE are therefore capable of including the sources of uncertainty associated with model fit in relation to distance from design points and this additional source of uncertainty.

6.1.2 SWAN calibration using GPE

Calibration of simulators such as SWAN can be a laborious and an expensive task. Automatic calibration techniques exist e.g. based on the minimisation of a prescribed objective function, but rely on many simulator runs. For simulators such as SWAN taking several minutes or hours to run, this is typically not a viable option. However, a GPE can be setup to cover the space not only of the offshore wind, wave and water conditions, but also the model settings used to tune the simulator. Then, automatic calibration becomes viable. Early tests show that the minimisation of an objective function, e.g. the RMSE in wave height, for a time series or storm peaks can be determined. The downside is that separate GPEs are (probably) required for different formulations for model settings to be considered, but this requires significantly smaller number of SWAN runs. More tests are ongoing.

6.1.3 Applying GPE to other simulators

SWAN is one of several simulators used in coastal modelling studies, but is generally limited in use to open waters. In order to accurately represent the important processes of diffraction and reflection and the associated interference patterns within enclosed areas such as ports and harbours we require the application phase resolving simulators. Several model types exist including simulators based on the linear Mild Slope Equation and non-linear Boussinesq type equations. These simulators are significantly more computationally expensive compared with SWAN, so may also benefit from use of GPEs. A GPE of the Mild Slope Equation model - ARTEMIS at HR Wallingford has already proven to be successful, and further work could be done using a GPE for the non-linear Boussinesq type simulators.

6.1.4 Wave forecasting

The computational efficiency of the GPE compared to the model provides the potential for applications requiring efficient computing. Wave forecasting is one such area, where the model runs need to be sufficiently fast to produce a forecast of future conditions, not only before the conditions have occurred, but to provide sufficient time to enable appropriate action, e.g placement of flood gates, evacuation, etc. Currently look-up-table approaches are sometimes applied, but the advantages of GPEs should mean that GPEs will take over this role, particularly where probabilistic forecasting using ensembles are used. For example, downscaling a deterministic offshore wave forecast to nearshore which comprises of one forecast for each time step using SWAN is often viable. However, downscaling a probabilistic offshore wave forecast comprising e.g. 12, 24 or 50 member ensemble for each time step without a supercomputer requires a meta-modelling technique such as GPE. Testing is currently underway to see if GPEs can be used.

Bibliography

Ajeesh, K. and Deka, P. C. (2015). Forecasting of Significant Wave Height Using Support Vector Regression, *Int. Conf. Adv. Comput. Commun.*.

Álvarez, M. A. and Lawrence, N. D. (2009). Computationally Efficient Convolved Multiple Output Gaussian Processes, *Jmlr*, **12**: 1459–1500.

Andrianakis, I. and Challenor, P. G. (2012). The effect of the nugget on Gaussian process emulators of computer models, *Comput. Stat. Data Anal.*, **56 (12)**: 4215–4228.

Bascom, W. (1964). Waves and beaches; the dynamics of the ocean surface.

Bastos, L. (2010). *Validating Gaussian Process Models in Computer Experiments*, Ph.D. thesis, University of Sheffield.

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

Becker, W., Oakley, J. E., Surace, C., Gili, P., Rowson, J. and Worden, K. (2012). Bayesian sensitivity analysis of a nonlinear finite element model, *Mech. Syst. Signal Process.*, **32**: 18–31.

Becker, W., Rowson, J., Oakley, J. E., Yoxall, A., Manson, G. and Worden, K. (2011). Bayesian sensitivity analysis of a model of the aortic valve, *J. Biomech.*, **44 (8)**: 1499–1506.

Booij, N., Haagsma, I. J. G., Holthuijsen, L. H., Kieftenburg, A., Ris, R. C., Van Der Westhuysen, A. J. and Zijlema, M. (2004). SWaN cycle III version 40.41 user manual, *Delft Univ. Technol.*, **115**.

Booij, N., Ris, R. C. and Holthuijsen, L. H. (1999). A third-generation wave model for coastal regions 1 . Model description and validation, *J. Geophys. Res.*, **104**: 7649–7666.

Boukouvalas, A. and Cornford, D. (2008). Dimension Reduction for Multivariate Emulation, **(July 2008)**.

Bounceur, N., Crucifix, M. and Wilkinson, R. D. (2015). Global sensitivity analysis of the climate-vegetation system to astronomical forcing: An emulator-based approach, *Earth Syst. Dyn.*, **6 (1)**: 205–224.

Boyle, P. and Frean, M. (2005). Dependent Gaussian Processes, *Adv. Neural Inf. Process. Syst.*, pp. 217–224.

Buhmann, M. D. (2003). *Radial Basis Functions: Theory and Implementations*, Cambridge University Press.

Caflisch, R. E. and Morokoff, W. J. (1994). Quasi-random sequences and their discrepancies, *SIAM J. Sci. Comput.*, **15 (6)**: 1251–1279.

Camus, P., Mendez, F. J. and Medina, R. (2011a). A hybrid efficient method to downscale wave climate to coastal areas, *Coast. Eng.*, **58 (9)**: 851–862.

Camus, P., Mendez, F. J., Medina, R. and Cofiño, A. S. (2011b). Analysis of clustering and selection algorithms for the study of multivariate wave climate, *Coast. Eng.*, **58 (6)**: 453–462.

Camus, P., Mendez, F. J., Medina, R., Tomas, A. and Izaguirre, C. (2013). High resolution downscaled ocean waves (DOW) reanalysis in coastal areas, *Coast. Eng.*, **72**: 56–68.

Chawla, A., Spindler, D. and Tolman, H. L. (2012). 30 Year Wave Hindcasts using WAVEWATCH III R with CFSR winds, Tech. rep.

Chen, T., Hadinoto, K., Yan, W. and Ma, Y. (2011). Efficient meta-modelling of complex process simulations with time-space-dependent outputs, *Comput. Chem. Eng.*, **35 (3)**: 502–509.

Clarke, S. M., Griebsch, J. H. and Simpson, T. W. (2005). Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses, *J. Mech. Des.*, **127 (6)**: 1077.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models, *J. Stat. Plan. Inference*, **140 (3)**: 640–651.

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, *Source J. Am. Stat. Assoc.*, **86 (416)**: 953–963.

Deltares (2017). Building with Nature.

Environmental Agency and Defra (2004). Best practice in coastal flood forecasting, Tech. rep.

Fricker, T. E., Oakley, J. E., Sims, N. D. and Worden, K. (2011). Probabilistic uncertainty analysis of an FRF of a structure using a Gaussian process emulator, *Mech. Syst. Signal Process.*, **25 (8)**: 2962–2975.

Fricker, T. E., Oakley, J. E. and Urban, N. M. (2013). Multivariate Gaussian Process Emulators with Nonseparable Covariance Structures, *Technometrics*, **55 (1)**: 47–56.

Friedman, J. (1991). Multivariate adaptive regression splines, *Ann. Stat.*, **19 (1)**: 1–141.

Gelfand, A. E., Banerjee, S. and Gamerman, D. (2005). Spatial process modelling for univariate and multivariate dynamic spatial data, *Environmetrics*, **16 (5)**: 465–479.

Gelfand, A. E., Schmidt, A. M., Banerjee, S. and Sirmans, C. F. (2004). Nonstationary multivariate process modeling through spatially varying coregionalization, *Test*, **13 (2)**: 263–312.

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 Sens.*, **8 (2)**: 1–32.

Goulard, M. and Voltz, M. (1992). Linear coregionalisation model: tools for estimation and choice of cross-variogram Matrix, *Math. Geol.*, **24 (3)**: 269–286.

Gouldby, B., Méndez, F. J., Guanche, Y., Rueda, A. and Mínguez, R. (2014). A methodology for deriving extreme nearshore sea conditions for structural design and flood risk analysis, *Coast. Eng.*, **88**: 15–26.

Gouldby, B., Wyncoll, D., Panzeri, M., Franklin, M., Hunt, T., Tozer, N., Dornbusch, U., Hames, D., Pullen, T. and Hawkes, P. (2016). National scale multivariate extreme value modelling of waves , winds and sea levels, in *E3S Web Conf.*, vol. 7, p. 1007.

Haylock, R. G. E. (1997). *Bayesian inference about outputs of computationally expensive algorithms with uncertainty on the inputs*, Ph.D. thesis, University of Nottingham.

Higdon, D., Gattiker, J., Williams, B. and Rightley, M. (2008). Computer model calibration using high dimensional output, *J. Am. Stat. Assoc.*, **103 (482)**: 570–583.

Higdon, D. M. (2002). Space and space-time modeling using process convolutions, *Quant. methods Curr. Environ. issues*, pp. 37–54.

Holden, P. B., Edwards, N. R., Oliver, K. I. C., Lenton, T. M. and Wilkinson, R. D. (2010). A probabilistic calibration of climate sensitivity and terrestrial carbon change in GENIE-1, *Clim. Dyn.*, **35 (5)**: 785–806.

Iooss, B., Feuillard, V., Boussouf, L. and Marrel, A. (2010). Numerical studies of the metamodel fitting and validation processes, *Int. J. Adv. Syst. Meas. 3*.

Jin, R., Chen, W. and Simpson, T. W. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria, *Struct. Multidiscip. Optim.*, **23 (1)**: 1–13.

Johnson, M. E., Ylvisaker, D. and Moore, L. M. (1990). Minimax and maximin distance designs, *J. Stat. Plan. Inference*, **26**: 131–148.

Jones, B. and Johnson, R. T. (2009). Design and Analysis for the Gaussian Process Model, *Qual. Reliab. Eng. Int.*, **25**: 515–524.

Jones, D. R., Schonlau, M. and Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions, *J. Glob. Optim.*, **13 (4)**: 455–492.

Journel, A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*, Academic press.

Kalra, R., Deo, M. C., Kumar, R. and Agarwal, V. K. (2005). Artificial neural network to translate offshore satellite wave data to coastal locations, *Ocean Eng.*, **32 (16)**: 1917–1932.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models, *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **63 (3)**: 425–464.

Komar, P. D. (1976). *Beach processes and sedimentation*, Prentice-Hall Englewood Cliffs, N.J, ISBN 0130725951.

Krzanowski, W. J. (1988). Principles of multivariate analysis: a user's perspective. Clarendon.

Liu, X. and Guillas, S. (2016). Dimension reduction for emulation: application to the influence of bathymetry on tsunami heights, pp. 1–26.

Maaten, L. V. D., Postma, E. and Herik, J. (2009). Dimensionality Reduction : A Comparative Review, **(April)**.

Maddux, T. B., Cowen, E. A., Foster, D., Haller, M. C. and Stanton, T. P. (2006). The cross-shore sediment transport experiment (CROSSTEX), *Coast. Eng. Proc.*, **3 (October)**: 2714–2725.

Maier, H. R. and Dandy, G. C. (2000). Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications, *Environ. Model. Softw.*, **15 (1)**: 101–124.

Malde, S., Oakley, J., Tozer, N., Liu, Y., Gouldby, B. and Wyncoll, D. (2018). Applying Gaussian process emulators for coastal wave modelling, *Coast. Eng.*.

Malde, S., Wyncoll, D., Oakley, J., Tozer, N. and Gouldby, B. (2016). Applying emulators for improved flood risk analysis, in *E3S Web Conf.*, vol. 04002.

Matheron, G. (1963). Principles of geostatistics, *Econ. Geol.*, **58 (8)**: 1246–1266.

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments, *J. Stat. Plan. Inference*, **43 (3)**: 381–402.

National Center for Atmospheric Research Staff (Eds) (2016). The Climate Data Guide: Climate Forecast System Reanalysis (CFSR).

Oakley, J. (1999). *Bayesian uncertainty analysis for complex computer codes*, Ph.D. thesis, University of Sheffield.

Oakley, J. (2002). Eliciting Gaussian process priors for complex computer codes, *J. R. Stat. Soc. Ser. D Stat.*, **51 (1)**: 81–97.

Oakley, J. E. (2011). Modelling with Deterministic Computer Models, in *Simplicity, Complex. Model.*, pp. 51—-67, John Wiley & Sons, Ltd, ISBN 9781119951445.

O'Hagan, A. (2006). Bayesian analysis of computer code outputs: A tutorial, *Reliab. Eng. Syst. Saf.*, **91 (10-11)**: 1290–1300.

Pullen, T., Liu, Y., Otinar Morillas, P., Wyncoll, D., Malde, S. and Gouldby, B. (2018). Bayonet gpe: A generic and practical overtopping model that includes uncertainty, *ICE's Maritime Engineering Journal*.

Rasmussen, C. E. (2006). Gaussian processes for machine learning.

Rougier, J. (2008). Efficient Emulators for Multivariate Deterministic Functions, *J. Comput. Graph. Stat.*, **17 (January 2015)**: 827–843.

Sacks, J., Welch, W. J., Mitchell, J. S. B., Henry, P. W., Mitchell, T. J. and Wynn, H. P. (1989). Design and Experiments of Computer Experiments, *Stat. Sci.*, **4 (4)**: 409–423.

Santner, T. J., Williams, B. J. and Notz, W. I. (2013). *The design and analysis of computer experiments*, Springer Science & Business Media.

Stein, M. L. (1999). *Interpolation of spatial data : some theory for kriging*, Springer series in statistics, New York: Springer, ISBN 0387986294.

Storlie, C. B., Swiler, L. P., Helton, J. C. and Sallaberry, C. J. (2009). Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models, *Reliab. Eng. Syst. Saf.*, **94 (11)**: 1735–1763.

The British Standards Institution (2013). BS 6349-1-1:2013. Maritime works, general code of practice for planning and design for operations, Tech. rep.

The WAMDI Group (1988). The WAM Model—A Third Generation Ocean Wave Prediction Model, *J. Phys. Oceanogr.*, **18 (12)**: 1775—-1810.

Thode, H. C. (1988). Statistical Tools for Simulation Practitioners, *Technometrics*, **30 (4)**: 464.

Tolman, H. L. (2009). User manual and system documentation of WAVEWATCH-IIITM version 3.14, *Tech. note*, **(3.14)**: 220.

Urban, N. M. and Fricker, T. E. (2010). A comparison of Latin hypercube and grid ensemble designs for the multivariate emulation of an Earth system model, *Comput. Geosci.*, **36 (6)**: 746–755.

Ver Hoef, J. M. and Barry, R. P. (1998). Constructing and fitting models for cokriging and multivariable spatial prediction, *J. Stat. Plan. Inference*, **69 (907)**: 275–294.

Wackernagel, H. (1995). *Multivariate geostatistics*, Springer.

Wilkinson, R. D. (2010). Bayesian Calibration of Expensive Multivariate Computer Experiments, *Large-Scale Inverse Probl. Quantif. Uncertain.*, **(January 2011)**: 195–215.

Wood, M., Eames, M. and Challenor, P. (2015). A comparison between Gaussian process emulation and genetic algorithms for optimising energy use of buildings, *14th Int. Conf. IBPSA - Build. Simul. 2015, BS 2015, Conf. Proc.*, pp. 680–687.

Wyncoll, D., Haigh, I., Gouldby, B., Hames, D., Laeger, S., Wall, A. and Hawkes, P. (2016). Spatial analysis and simulation of extreme coastal flooding scenarios for national-scale emergency planning, *E3S Web Conf.*, **7**: 01001.

Xing, W., Shah, A. A. and Nair, P. B. (2014). Reduced dimensional Gaussian process emulators of parametrized partial differential equations based on Isomap, *Proc. R. Soc. A Math. Phys. Eng. Sci.*, **471 (2174)**: 20140697–20140697.