Generalising history matching for enhanced calibration of computer models

Wenzhe Xu

Supervisor: Daniel Williamson

Peter Challenor

College of Engineering, Mathematics and Physical Sciences

University of Exeter

Submitted by Wenzhe Xu to the University of Exeter as a thesis for the degree of

Doctor of Philosophy in Mathematics, March 2021

Abstract

History matching using Gaussian process emulators is a well-known methodology for the calibration of computer models. It attempts to identify the parts of input parameter space that are likely to result in mismatches between simulator outputs and physical observations by using emulators. These parts are then ruled out. The remaining "Not Ruled Out Yet (NROY)" input space is then searched for good matches by repeating the history matching process.

The first section of this thesis illustrates an easily neglected limitation of standard history matching: the emulator must simulate the target NROY space well, else good parameter choices can be ruled out. We show that even when an emulator passes standard diagnostic checks on the whole parameter space, good parameter choices can easily be ruled out. We present novel methods for detecting these cases and a Local Voronoi Tessellation method for a robust approach to calibration that ensures that the true NROY space is retained and parameter inference is not biased.

The remainder of this thesis looks into developing a generalised history matching for calibrating computer models with high-dimensional output. We address another limitation of the standard (PCA-based) history matching, which only works well when the parameters are responsible for the strength of various patterns. We show that when the parameters control the position of patterns, e.g. shifting currents, current approaches will not generally be able to calibrate these models. To overcome this, we extend history matching to kernel feature space, where output space for moving patterns can be compared with the observations. We develop kernel-based history matching as a generalisation to history matching and examine the multiple possible interpretations of the usual implausibility measure and threshold for defining NROY. Automatic kernel selection based on expert modeller judgement is introduced to enable the experts to define important features that the model should be able to reproduce.

Acknowledgements

First and foremost, I want to thank my supervisor Daniel Williamson for the continuous support of my PhD study and research. His guidance helped me in all the time of research and writing of this thesis. I would also like to thank Peter Challenor for his continuous encouragement and insightful comments. My sincere thank also goes to my collaborators from HIGH-TUNE project. In particular, I would like to thank Frédéric Hourdin, for spending a great deal of time helping me with the `Shiny` app.

I would like to thank my family. I am extremely grateful to my parents, for their love, prayers, caring and supporting. I am very much thankful to my husband Ge for his love, understanding, and continuing support to complete this research work.

Finally, I would like to say thanks to my friends and research colleagues over the last few years, who have made my time in Exeter so enjoyable. Special mention goes Victoria, Louise, Evan and Heba for all of the fun we have had in the last four years. I would also like to thank my best friends, Diana and Qiaorong, for standing with me all the time.

Publications

The majority of the results of Chapter 3 have published in the following:

Wenzhe Xu, Daniel B. Williamson and Peter Challenor "Local Voronoi tessellations for robust multi-wave calibration of computer models." International Journal for Uncertainty Quantification (2020).

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Computer models have been widely used in many areas of science to learn about features of the real-world. Determining the settings of a computer model's input parameters, so that the outputs are consistent with real-world observations, is an important problem. Before using these models to perform inference about the past, current and future states of a complex system, careful parameter calibration (the climate modelling community refers to calibration as 'tuning') is required to give parameters that lead to accurate representations of the real world. Calibration has been used in a variety of applications, such as climate systems, epidemiology, galaxy formation and agro-ecosystem modelling (Andrianakis et al., 2015; Lehuger et al., 2009; Salter and Williamson, 2016; Vernon et al., 2010; Williamson et al., 2015). However, computer models are usually expensive and/or take a long time to run. For example, high-resolution climate models can take days or even weeks to run on supercomputers (Hourdin et al., 2017). When it is not possible to run the model often enough to calibrate directly, a small, carefully chosen set of model runs, often termed a 'design' or 'ensemble', can be run and used to explore the input parameter space.

The field of Uncertainty Quantification (UQ) provides different approaches to construct an 'emulator' or 'surrogate': an inexpensive statistical model used to approximate the computer model. We use the common choice, Gaussian process emulators, as fast approximations for computer model output at input parameters x. Emulators generate a prediction for the computer model together with uncertainty on the prediction. After assessing the adequacy of a proposed GP emulator to represent a model response, calibration can then be carried out using the emulator to efficiently explore the input space.

Within the uncertainty quantification literature, there are different approaches to calibration. The method that I consider in detail in this thesis is history matching. History matching attempts to identify the parts of the input parameter space that are likely to result in mismatches between computer outputs and observations by iteratively removing those regions of parameter space in which we are virtually certain that there are no good matches. In particular, the retained regions of input space, termed "Not Ruled Out Yet" (NROY) space is then searched for good matches by repeating the history matching process.

During this PhD, my supervisor, Prof. Williamson, has worked closely with the climate modellers at LMDZ and Météo-France as part of the HIGH-TUNE project (The French National Research Agency). The aim of HIGH-TUNE is to use high-resolution simulators to improve and tune boundary-layer cloud parameterisations. The climate modellers are interested in developing tools to automatically tune boundary layer cloud parameterisations within their models, based on history matching. As a member of Exeter UQ team, I helped Prof. Williamson to deliver a workshop with the climate modellers aimed at developing and maintaining software for tuning the French climate model. The collaboration involves providing methods to both emulate and history match to a large number of process-based metrics, rapidly and automatically, enabling the modellers to use the tools independently. With multiple unsupervised calibrations, it is important that history matching is robust enough to withstand potential ensemble issues.

Some unexplored limitations of standard history matching were found through this close collaboration and these are the subject of this thesis. The first, easily neglected, limitation can occur when an emulator is unable to simulate the target NROY space effectively, even if it seems to pass all standard emulator diagnostic checks. Poor simulation may result in true NROY space being ruled out without any indication for the analyst that this has occurred. For simulators that are constantly under development, such as climate models, this could be a costly mistake that causes parameterizations or even computational methods and hardware to be needlessly revisited, even though the model was already fit for purpose. We present novel methods for detecting these cases and a Local Voronoi Tessellation method for a robust approach to history matching that ensures that the true NROY space is retained and parameter inference is not biased. These methods have been published in Xu et al. (2021).

When history matching computer models with high-dimensional output (e.g. a time series, a spatial field, or a spatio-temporal field), it is common to use dimension reduction techniques: to represent the high dimensional output as linear combinations of a fixed set of low dimensional basis vectors, reducing the complexity of calculations. However, when we try to match spatio-temporal fields (or spatial fields), sometimes what is important to the credibility of the model is that the key physical patterns are present, even if they may not be in the right place or be an exact replica. This is particularly true for climate models when parameter values compatible with emergent phenomenon (such as large-scale circulations) are often the target of the exercise, but where it is not expected that these phenomena occur in exact the same place (or at the same time) as in observations of the real climate. We found this limitation of standard history matching by applying standard PCA-based history matching to the clouds model at the HIGH-TUNE workshop, but this limitation can happen for any model. Existing statistical methods for calibrating are only good at finding stronger or weaker signals in fixed locations. In this thesis, we enhance history matching using kernel methods. A kernel-based history matching (KHM) method is proposed to perform history matching in a higher-dimensional feature space, where output space for moving patterns can be compared with the observations.

The innovations in the thesis aim to 'robustify' history matching in order to allow non-statisticians to independently perform automatic tuning of computer models. In particular, we first develop a diagnostic to check whether a globally good emulator failed locally near the target NROY space, which is a crucial step for checking whether performing history match with the current emulator is dangerous or not. If the emulator cannot be trusted to calibrate directly, a Local Voronoi Tessellation method provides a way to safely and automatically isolate any possible target NROY regions of parameter space where we do not trust the emulator, and to history match in the remaining space. This approach allows the same emulator to be used appropriately for that emulation without having to waste a whole wave or any further cost with further runs.

The remaining part of this thesis is focused on calibration with computer models with high-dimensional output. We explored KHM as a generalisation to history matching and examine the multiple possible interpretations of the usual implausibility measure and threshold for defining NROY. Particularly important is model discrepancy: what it means and how it is treated in kernel approaches. Given our preferred approaches, we establish kernel selection methods: based on expert input via a `Shiny` app. We applied KHM to IPSL-CM, the inspiring French climate model that failed with the standard history match, to establish the effectiveness and accuracy of KHM. KHM is a contribution to UQ, involve harnessing expert input to deliver semi-automatic calibration.

1.1 Thesis Outline

In Chapter 2, we introduce the current literature in uncertainty quantification (UQ), with a focus on Gaussian process emulation, the diagnostics used to validate and assess the adequacy of a Gaussian process emulator for representing the simulator, history matching, both for models with scalar output and multivariate output, and we briefly discuss Bayesian calibration.

Chapter 3 presents a novel Local Voronoi Tessellation design that can be used for robust multi-wave calibration of computer models. We present a novel detecting method, taking place after the emulator diagnostic check, that attempts to determine whether the emulator could have failed near the target NROY space. A Local Voronoi Tessellation design is introduced after the detecting step. We compare our approach to standard history matching and assess the performance for two illustrative examples and a climate model, IPSL-CM.

In Chapter 4, we discuss the drawbacks of history matching for calibrating computer models with high-dimensional output. A KHM method is proposed to perform history matching in a high dimensional space ('feature' space). We reframe history matching in the feature space and introduce new distance measures to define the implausibility in feature space. We propose new cut-off thresholds for the implausibility to account for all sources of uncertainty. We finish Chapter 4 by comparing our methods to standard history matching based on PCA on an idealised numerical example.

In Chapter 5, we investigate the important step in KHM: choosing a suitable kernel for each application. We present an automatic optimization algorithm for kernel selection that considers 'expert'/'modeller' prior knowledge. Moreover, we also prove that standard history matching can be achieved by KHM with a specific kernel function, which shows our approach is a generalisation of standard history matching.

In Chapter 6, we illustrate KHM with the French climate model, IPSL-CM, with the goal of satisfying all of the modeller's calibration targets. We start by designing a new interactive R `Shiny` app to collate expert's judgement ('acceptable' runs in the training data). We first apply the optimisation algorithm of Chapter 5 to select a kernel for this climate model, and then illustrate the KHM of Chapter 4 by performing three iterations.

We conclude in Chapter 7, and highlight a number of potential areas for future work.

Chapter 2

Background

2.1 Computer experiments and simulators

Computer models, or simulators, are systems of physical equations that are implemented as computer code to make inferences about the real-world. Computer models are used across many disciplines, such as in climate and environment science (Bony and Dufresne, 2005; Edwards, 2001; Taylor et al., 2012), cosmology (Bower et al., 2010; Kaufman et al., 2011; Vernon et al., 2010), social sciences (Sun et al., 2006), engineering (Ankenman et al., 2010; Kirkpatrick, 2000) and biological applications (Andrianakis et al., 2015, 2017). The physical processes being studied are usually representative of complex systems, making experimentation and measurement difficult or expensive over the relative space. In order to learn about features of the real-world, computer models can be used to represent the complex system so it can be studied. There are two components of a computer model; inputs and outputs. Santner et al. (2003) classify computer model inputs into three classes based on the role they play in the code; control variables, environmental variables and model variables. Control variables are usually set by engineers to control the outputs. Environmental variables can also affect the computer outputs, but the effect varies for specific users. Model variables, also known as model parameters or tuning parameters, are usually unknown, or given with a subjective probability distribution, characterise the behaviour of the simulator. For example, often a goal is to calibrate these parameters so the model behaviour is close to the behaviour of the real system.

The output of a computer model can take many different forms such as a single value, a spatial field, a time series or a combination of these. For climate models, the output is produced in a grid of boxes over the globe with several different output fields. For example, IPSL-CM is an atmosphere model that is used to predict planetary atmospheres, including that of the Earth and other planets (Mars, Titan, Venus), as well as regional climate process studies (Bony and Dufresne, 2005; Hourdin et al., 2017; Voldoire et al., 2013). It simulates the different process of the world over a horizontal and vertical grid, which can be arranged to give outputs in each grid box over any given regional scale over time (Hourdin et al., 2006).

The outputs of a computer model usually correspond to process in the real world. Since these simulator outputs are not able to perfectly represent the real world, uncertainties in the outputs are inevitable. Therefore, uncertainty quantification is required to quantify and reduce the uncertainties in the outputs of a computer model (Kennedy and O'Hagan, 2001).

2.2 Uncertainty Quantification

There are many uncertainties associated with a computer model's construction and application. Uncertainty Quantification (UQ) refers to the methodologies which are used to quantify these uncertainties of computer models (Smith, 2013).

The various sources of uncertainty in the computer models are grouped into classes by Kennedy and O'Hagan (2001). Parameter uncertainty occurs when the computer model contains unknown parameters whose exact values cannot be controlled in experiments or defined by physical knowledge. To solve this issue, calibration is a commonly used approach to estimate unknown parameter inputs by comparing computer model outputs with partial observations of the modelled processes.

Even after eliminating parameter uncertainty, there is still no computer model that can represent a real world process perfectly without any error. Structural uncertainty, also called model discrepancy, is introduced to measure the uncertainty that comes from computer model inadequacy. Model discrepancy will be discussed further in Section 2.4.1. If it is possible for there to be a difference between the observed value and true value of a real world process, then the difference is referred to as an observation error (or measurement error) (Kennedy and O'Hagan, 2001). There are many possible causes of this error, such as human error and the limitation of instruments.

When a computer model is expensive and/or takes long time to run, it might be not possible to run the model for every set of inputs that we are interested in. So, it is necessary to construct a fast surrogate model to represent the simulator (Section 2.3). An extra source of uncertainty known as code uncertainty is introduced in this situation (O'Hagan, 2006).

There are other sources of uncertainties particular to specific model types. For example, with climate models, the predictions are of a chaotic nature, and are sensitive to the initial value of state variables used in simulators (Palmer et al., 2005). A slight difference in the initial conditions due to observation uncertainty would lead to a very different prediction of future weather. Tebaldi and Knutti (2007) define this uncertainty as "initial condition uncertainty". Boundary condition uncertainty is also mentioned by Tebaldi and Knutti (2007), which is caused by human influences, future anthropogenic emissions and unpredictable natural phenomena.

To quantify the uncertainties in computer experiments, a number of frameworks have been developed in the uncertainty quantification literature. Apart from calibration (details in Section 2.4), there are other approaches used to estimate the different sources of uncertainties, such as uncertainty analysis and sensitivity analysis (SA). Uncertainty analysis, also known as uncertainty propagation, quantity the uncertainty in model outputs introduced by uncertainty in the inputs (mainly parameter uncertainty) (Oakley and O'Hagan, 2002). SA is a study related to an uncertainty analysis: the goal of SA is to identify how model inputs affect the model outputs (Oakley and O'Hagan, 2004; Saltelli et al., 2000). The idea behind SA is to study the sensitivity of the computer model's output, with respect to each input parameter. As parameter combinations may include non-linear interactions, it is advised to investigate all parameters simultaneously not one by one (Saltelli et al., 2005).

2.3 Emulation

Emulators are computationally cheap statistical models which are used to approximate expensive computer simulators (Currin et al., 1991; Haylock and O'Hagan, 1996). Most standard techniques of uncertainty quantification require simulators to be evaluated at a very large number of design variables. For instance, hundreds of millions of model runs could be required in a Monte Carlo approximation (Kennedy and O'Hagan, 2001) for uncertainty analysis. The majority of computer models, however, are expensive and time consuming to run, hindering comprehensive future analyses. For example, a global climate model may take several months to complete a single run (Rougier et al., 2009). In such situations, it is not possible to run the computer model the number of times required to obtain valid results (Williamson et al., 2017). In order to mitigate this issue, an emulator is considered as a more efficient surrogate model of the simulator for further analysis, which can significantly improve computational efficiency (Asher et al., 2015).

A computer model is a function, f, that maps a vector of inputs, $\mathbf{x}$, from input space $\mathcal{X}$ into an output space with model output $f(\mathbf{x})$. An emulator treats a computer model as a black box, and the mapping from $\mathbf{x}$ to $f(\mathbf{x})$ is learned by the emulator without necessarily having any information of the inner workings of f. To build an emulator, a small ensemble of model runs based on a design in the parameter inputs is generated. With the knowledge gained from this ensemble, an emulator is built to represent the computer model. For the given design input, the emulator generates the same value as the computer model output with no uncertainty. At other inputs, the emulator provides the entire range of possible values for $f(\mathbf{x})$ rather than a single approximation value.

Whilst there are many different approaches to emulation, the general form of an emulator is considered as the sum of two independent processes (Sacks et al., 1989)

$$f_i(\mathbf{x}) = \sum_{j=1}^{k} \beta_{ij} h_j(\mathbf{x}) + \epsilon_i(\mathbf{x}), \tag{2.1}$$

where β_{ij} are unknown regression coefficients, $h_j(\mathbf{x})$ are chosen regression functions and $\epsilon(\mathbf{x})$ is a correlated residual process representing the difference between $f(\mathbf{x})$ and the linear model. Starting from this general form, we devote the rest of this section to presenting Gaussian process emulation.

2.3.1 Gaussian process emulation

A Gaussian Process (GP) is a stochastic process. Any finite number of random variables from a Gaussian process has a joint Gaussian distribution (Rasmussen and Williams, 2006). Specifically, $f(\mathbf{x})$ is a Gaussian process, if for any finite collection of runs, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, $n > 1$, the vector of model output $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$ has a multivariate normal distribution. A Gaussian process for $f(\mathbf{x})$ is determined by a mean function and a covariance function

$$f(\mathbf{x})|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathrm{GP}\Big(m(\mathbf{x}),\ \sigma^2 c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})\Big), \tag{2.2}$$

where σ^2 is a hyper-parameter that controls the scaling of the process and $\boldsymbol{\delta}$ is a vector of correlation length parameters used to define the correlation function $c(\mathbf{x}, \mathbf{x}')$. A detailed review of covariance functions is given in Section 2.3.2.

In the formulation of equation (2.1), if we model the residual term $\epsilon(\mathbf{x})$ as a Gaussian process with mean zero, this is equivalent to setting

$$m(\mathbf{x}) = h(\mathbf{x})^T \boldsymbol{\beta}, \tag{2.3}$$

and

$$\mathrm{Cov}\left[\epsilon(\mathbf{x}), \epsilon(\mathbf{x}')\right] = \sigma^2 c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta}). \tag{2.4}$$

2.3.2 Covariance function (Kernel)

The choice of covariance function, or kernel $k(\mathbf{x}, \mathbf{x}')$, is one of the key elements of the Gaussian Process. A covariance function is defined with a user-specified correlation function $c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\delta})$ (see equation (2.4)). The correlation function defines the similarity or nearness between inputs: two inputs which are immediate neighbours are likely to give similar outputs (Rasmussen and Williams, 2006). There are many choices of correlation functions within the kernel-based methods literature. We give some of the most popular below.

A widely-used choice is the Gaussian or squared exponential correlation function (Kennedy and O'Hagan, 2000),

$$c(\mathbf{x}, \mathbf{x}') = \exp\left\{-\sum_{i=1}^{p}\left(\frac{x_i - x_i'}{\delta_i}\right)^2\right\}. \tag{2.5}$$

The distance between two inputs for each input dimension is scaled by the corresponding correlation length parameter $\delta_i, i = 1, \ldots, n$. The squared exponential correlation function is infinitely differentiable, which leads to the Gaussian process being infinitely mean-square differentiable and very smooth (Rasmussen and Williams, 2006).

A general form of the squared exponential correlation function is the power exponential correlation function,

$$c(\mathbf{x},\mathbf{x}') = \exp\left\{-\sum_{i=1}^{p}\left(\frac{x_i - x_i'}{\delta_i}\right)^{\kappa_i}\right\},$$

with $0 < \kappa_i \leq 2$. κ_i is estimated for each dimension of the inputs i. When κ_i is less than 2, the smoothness of a GP with a power exponential correlation function is lower than a GP with squared exponential correlation function.

Another standard choice is the Matérn correlation function,

$$c(\mathbf{x},\mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left[\frac{\sqrt{2\nu}}{\delta}|\mathbf{x}-\mathbf{x}'|\right]^{\nu} K_\nu\left[\frac{\sqrt{2\nu}}{\delta}|\mathbf{x}-\mathbf{x}'|\right], \tag{2.6}$$

where $\Gamma()$ is the gamma function, K_ν is the modified Bessel function of the second kind of order ν, and δ is correlation length parameter (Abramowitz, 1985). For $\nu \to \infty$, the Matérn correlation function is the same as squared exponential correlation function (Nychka et al., 2002). When ν is a half-integer, $\nu = \frac{1}{2} + p$ where p is non-negative integer, the Matérn correlation function becomes a product of an exponential and a polynomial of order p,

$$c(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\sqrt{2\nu}|\mathbf{x}-\mathbf{x}'|}{\delta}\right)\frac{\Gamma(q+1)}{\Gamma(2q+1)}\sum_{i=0}^{p}\frac{(p+1)!}{i!(p-1)!}\left(\frac{\sqrt{8\nu}|\mathbf{x}-\mathbf{x}'|}{\delta}\right)^{p-i}.$$

Two common choices of ν are $\nu = 3/2$ and $\nu = 5/2$ (Rasmussen, 2003). In addition to these correlation functions, we present further discussion of kernels in Chapter 4.

2.3.3 The nugget parameter

In the Gaussian process model shown in equation (2.2), the emulator interpolates the model runs exactly at the given design input, with zero variance. This may not always be a useful property. For instance, in climate models, different model outputs for the same input could occur by varying the initial conditions (Williamson and Blaker, 2014), it is inappropriate to interpolate the model runs exactly for this application. Craig et al. (1996) modify the emulation definition in equation (2.1) by adding a nugget term, $\upsilon(\mathbf{x})$,

$$f(\mathbf{x}) = h(\mathbf{x})^T\boldsymbol{\beta} + \epsilon(\mathbf{x}) + \upsilon(\mathbf{x}). \tag{2.7}$$

The nugget term $\upsilon(\mathbf{x})$ is independent and identically distributed of other terms with mean zero and prior variance σ_υ^2 for all inputs. More precisely,

$$\text{Cov}\left[\upsilon(\mathbf{x}), \upsilon(\mathbf{x}')\right] = \begin{cases} \sigma_\upsilon^2, & \mathbf{x} = \mathbf{x}' \\ 0, & otherwise. \end{cases} \tag{2.8}$$

One interpretation for equation (2.13) is that the emulators contains some variability, σ_υ^2, which is not introduced by the inputs (Andrianakis and Challenor, 2012). The addition of the nugget modifies the probabilistic specification for simulator $f(x)$ in equation (2.2),

$$f(\mathbf{x})|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\delta} \sim \mathcal{GP}\left(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta}, \sigma_\upsilon^2)\right), \tag{2.9}$$

with the same defined mean function as equation (2.3) and a new covariance function

$$k(\mathbf{x}, \mathbf{x}'; \sigma^2, \boldsymbol{\delta}, \sigma_\upsilon^2) = \sigma^2 c(\mathbf{x}, \mathbf{x}'; \delta) + \sigma_\upsilon^2 \mathbf{1}\{\mathbf{x} = \mathbf{x}'\}, \tag{2.10}$$

where the indicator function is

$$\mathbf{1}\{\mathbf{x} = \mathbf{x}'\} = \begin{cases} 1, & \mathbf{x} = \mathbf{x}' \\ 0, & otherwise. \end{cases} \tag{2.11}$$

However, this parameterisation leads to a marginalisation of σ^2 which is analytically intractable, implying that σ^2 would have to be estimated jointly with $\boldsymbol{\delta}$ or even marginalised numerically (Andrianakis and Challenor, 2012). Another way to add a nugget parameter to the covariance function by Andrianakis and

Challenor (2012); Gramacy and Lee (2012) is

$$k(\mathbf{x},\mathbf{x}';\sigma^2,\boldsymbol{\delta},\sigma_\upsilon^2) = \sigma^2 \left(c(\mathbf{x},\mathbf{x}';\delta) + \sigma_\upsilon^2 \mathbf{1}\{\mathbf{x}=\mathbf{x}'\} \right), \tag{2.12}$$

where $\sigma^2\sigma_\upsilon^2$ represents the variability which is not captured by the correlated part.

Craig et al. (1997, 1996) divide the inputs into active inputs $\mathbf{x}_{Active}$ and inactive inputs $\mathbf{x}_{Inactive}$ based on their effect on the output: only active inputs are used to build an emulator. An emulator representation for $f(\mathbf{x})$ then becomes

$$f(\mathbf{x}) = h(\mathbf{x}_{Active})^T\boldsymbol{\beta} + \epsilon(\mathbf{x}_{Active}) + \upsilon(\mathbf{x}). \tag{2.13}$$

The nugget term $\upsilon(\mathbf{x})$ could be used to account for uncertainty in the inactive inputs. This model could significantly reduce the input dimensionality of $\epsilon(\mathbf{x}_{Active})$, and hence provide significant computational savings.

There are other possible reasons to include a nugget term in deterministic model emulators. Gramacy and Lee (2012) demonstrate that a nugget term could be used to account for the discrepancies between the Gaussian process emulator and computer model, which can lead to a better performing emulator. Moreover, adding a nugget parameter on to the principal diagonal of the design correlation matrix can be used to alleviate numerical problems in fitting Gaussian processes to data (Neal, 1997). Numerical problems occur when the covariance matrix for the design points is ill-conditioned, mostly occurs with the squared exponential correlation function, so its inversion might be inaccurate or not feasible (Andrianakis and Challenor, 2012).

2.3.4 Fitting a Gaussian process Emulator

A Bayesian approach is typically used for fitting a Gaussian process emulator (Currin et al., 1991). Let the computer model run at n points, $\mathbf{x} = (x_1,\ldots,x_n)^T \in \mathcal{X}$, where $\mathcal{X}$ is the p-dimensional input space. Also, let $\mathbf{F} = (f(x_1),\ \ldots,\ f(x_n))$ represent known outputs of the model at the inputs. According to equation (2.2), the output

F has a multivariate normal distribution,

$$\mathbf{F}|\boldsymbol{\beta},\sigma^2,\boldsymbol{\delta} \sim \text{MVN}(\mathbf{H}^T\boldsymbol{\beta},\ \sigma^2\mathbf{A}),$$

where,

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

$$\mathbf{A} = \begin{bmatrix} 1 & c(\mathbf{x}_1,\mathbf{x}_2) & \ldots & c(\mathbf{x}_1,\mathbf{x}_n) \\ c(\mathbf{x}_2,\mathbf{x}_1) & 1 & & \vdots \\ \vdots & & \ddots & \\ c(\mathbf{x}_n,\mathbf{x}_1) & \ldots & & 1 \end{bmatrix}.$$

Given the emulator hyperparameters and the distribution of $f(x)$ in equation (2.2), the posterior distribution of f at a new input $\mathbf{x}$, given ensemble $\{\mathbf{X},\mathbf{F}\}$, is

$$f(\mathbf{x})|\{\mathbf{X},\mathbf{F}\},\boldsymbol{\beta},\boldsymbol{\delta},\sigma^2 \sim \mathcal{GP}(m^*(\mathbf{x}),\ \sigma^2 c^*(\mathbf{x},\mathbf{x}')), \tag{2.14}$$

with well-known analytic expressions for $m^*(\mathbf{x})$

$$m^*(\mathbf{x}) = h^T(\mathbf{x})\boldsymbol{\beta} + t(\mathbf{x})^T\mathbf{A}^{-1}(\mathbf{F}-\mathbf{H}\boldsymbol{\beta}), \tag{2.15}$$

and $c^*(x,x')$

$$c^*(\mathbf{x},\mathbf{x}') = c(\mathbf{x},\mathbf{x}') - t(\mathbf{x})^T\mathbf{A}^{-1}t(\mathbf{x}'), \tag{2.16}$$

where

$$t(\mathbf{x})^T = (c^*(\mathbf{x},\mathbf{x}_1),\ldots,c^*(\mathbf{x},\mathbf{x}_n)).$$

There are different approaches to handling hyperparameters β, $\boldsymbol{\delta}$ and σ^2. Currin et al. (1991) adopt a maximum likelihood method to fit the hyperparameters. The likelihood in this case is

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

The hyperparameters can be estimated by maximising the likelihood equation,

$$\hat{\boldsymbol{\beta}} = (\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{A}^{-1}\mathbf{F},$$

$$\hat{\sigma^2} = \frac{(\mathbf{F}-\mathbf{H}\hat{\boldsymbol{\beta}})\mathbf{A}^{-1}(\mathbf{F}-\mathbf{H}\hat{\boldsymbol{\beta}})}{n},$$

and

$$\hat{\boldsymbol{\delta}} = \underset{\boldsymbol{\delta}}{\operatorname{argmin}}\left[p(\mathbf{F}|\hat{\boldsymbol{\beta}},\ \hat{\sigma}^2,\ \boldsymbol{\delta})\right].$$

This approach has a drawback, in that the hyperparameters are usually highly confounded, leading to a ridge on the likelihood surface for large δ and σ^2. One possible resolution to this problem is to specify δ.

Haylock and O'Hagan (1996) propose a 'non-informative' prior:

$$P(\beta,\sigma^2) \propto \sigma^{-2}. \tag{2.17}$$

A benefit of 'non-informative' prior is that the posterior analysis is tractable. By integrating out $\boldsymbol{\beta}$, the posterior distribution can be written down conditioned on the ensemble and parameters:

$$f(\mathbf{x})|\boldsymbol{\delta},\sigma^2 \sim \mathcal{GP}(m^{**}(\mathbf{x}),\sigma^2 c^{**}(\mathbf{x},\mathbf{x}')),$$

with mean

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

and variance

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

for $\hat{\boldsymbol{\beta}} = (\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})\mathbf{H}^T\mathbf{A}^{-1}\mathbf{F}$. By integrating out σ^2, the posterior distribution for $f(\mathbf{x})|\boldsymbol{\delta}$ is a multivariate student t-distribution with $n-q$ degrees of freedom,

$$\frac{f(\mathbf{x}) - m^{**}(\mathbf{x})}{\sqrt{\frac{\mathbf{F}^T(\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{H}(\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{A}^{-1})\mathbf{F}c^{**}(\mathbf{x},\mathbf{x}')}{n-q-2}}} \sim t_{n-q}, \tag{2.18}$$

where q is the rank of the matrix $\mathbf{H}$. Therefore, Gaussian process emulators can be used to make predictions of the simulator output for an input $\mathbf{x}$ given the values of the correlation parameters $\boldsymbol{\delta}$.

This formulation does not account for the inclusion of the nugget in the Gaussian process. However, the method of Haylock and O'Hagan (1996) can be adapted to add a nugget to covariance function, with the form

$$k(\mathbf{x},\mathbf{x}';\sigma^2,\boldsymbol{\delta},\sigma_v^2) = \sigma^2\left(c(\mathbf{x},\mathbf{x}';\delta)+\sigma_v^2\mathbf{1}\{\mathbf{x}=\mathbf{x}'\}\right). \tag{2.19}$$

The original covariance function can then be replaced, and the addition of a nugget term does not change the posterior distribution for $f(x)$ with a non-informative prior for β and σ^2.

Different prior distribution choices for the hyperparameters are also well established in the literature. Oakley and O'Hagan (2004) use an informative Normal-inverse gamma prior for β and σ^2,

$$P(\beta,\sigma^2) \propto \sigma^{-(p+q+2)/2}\exp\left(-\left((\boldsymbol{\beta}-\mathbf{z})^T\mathbf{V}^{-1}(\boldsymbol{\beta}-\mathbf{z})+a\right)/2\sigma^2\right), \tag{2.20}$$

where $\mathbf{z}$, $\mathbf{V}$ and q are user-specified parameters that allow the user to incorporate prior knowledge about f into the model. Given the data, the posterior of $f(\mathbf{x})$ can be shown to be

$$\frac{f(\mathbf{x})-m^*(\mathbf{x})}{\hat{\sigma}\sqrt{c^*(\mathbf{x},\mathbf{x}')}} \sim t_{p+n}, \tag{2.21}$$

with

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

and

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

for

$$\hat{\boldsymbol{\beta}} = \mathbf{V}^*(\mathbf{V}^{-1}\mathbf{z}+\mathbf{H}^T\mathbf{A}^{-1}\mathbf{F}),$$

$$\hat{\sigma^2} = \left(a + \mathbf{z}^T \mathbf{V}^{-1} \mathbf{z} + \mathbf{F}^T \mathbf{A}^{-1} \mathbf{F} - \hat{\boldsymbol{\beta}} (\mathbf{V}^*)^{-1} \hat{\boldsymbol{\beta}}\right) / (n + p + 2),$$

$$\mathbf{V}^* = (\mathbf{V}^{-1} + \mathbf{H}^T \mathbf{A} \mathbf{H})^{-1}.$$

The posterior distribution is a multivariate t-distribution with $p+n$ degrees of freedom.

Higdon et al. (2008), Gramacy and Lee (2012) and Volodina and Williamson (2020) fit GPs via Full Bayes Markov chain Monte Carlo (MCMC) methods, with a benefit that prior distributions over all parameters can be used to penalise the ridge on the likelihood surface. In this case, we cannot analytically define the posterior distribution, hence MCMC is used to sample it. This approach allows flexible prior specification, but comes at an expensive computational cost. Gu et al. (2018) estimate GP emulator hyperparameters by a marginal posterior mode estimator, which provides stable results for emulator with lower predictive errors.

2.3.5 Multivariate emulation

In the previous sections, we introduced some approaches to constructing a GP for computer models with univariate output. However, computer models can give a number of different forms of multivariate output, for instance, a time-series output could be generated by dynamic simulators to make inference of time-evolving systems. A single run of such simulators consists of an extensive simulation over time for each input. In general, the output length is a multiple of the length of the time-series output, which can be extremely large. The form of the emulation depends on the forms of the computer model output. There is a wide literature on multivariate emulation approaches, in particular those that can handle computer models with different forms of multivariate outputs (Conti and O'Hagan, 2010; Higdon et al., 2008; Liu et al., 2009; Overstall and Woods, 2016).

For computer models with spatial or spatio-temporal output, the univariate emulation can be directly extended to multivariate emulation by building the emulator for each of the responses separately (each model grid cell). Lee et al.

(2013) apply this approach for emulating the global model simulations of cloud condensation nuclei (CCN). In their paper, the model output is the monthly mean CCN for each model grid cell. They build independent emulators for every month, and every model grid cell, with no correlation across outputs. However, since the simulator generates a massive quantity of data for every coordinate during each simulator run, building individual emulators is computationally expensive. Another drawback of this approach is that no account is taken of spatial or temporal correlation.

In a similar way, Gu et al. (2016) build emulators for every spatial and temporal output independently. In that paper, they are working with the test bed simulator TITAN2D (Patra et al., 2005). The specific simulator they emulate is a volcanic pyroclastic flow simulator, which will generate up to 10^9 outputs over a space-time grid of coordinates during each simulator run. To achieve a computationally efficient emulator, they use a joint mean function, and the correlation parameters are only estimated once. These estimated correlation parameters are adopted for every emulator of the outputs.

To construct emulators for computer models with a time-series output, Kennedy and O'Hagan (2001) treat time as an input variable and consider time in the correlation structure. Therefore they can emulate the response via the univariate GP emulator. Instead of emulating a complete multi-step run of the simulators, Conti et al. (2009) emulate a single-step simulator and then use the emulator iteratively. The fundamental assumption is that simulator output at time $t = T$, $f_T(\cdot)$ can be expressed iteratively in terms of the single-step simulator (Conti et al., 2009; Mohammadi et al., 2019). For a computer model f, the l-dimensional Gaussian process is

$$f(\mathbf{x})|B, \Sigma, R \sim \text{GP}\left(m(\mathbf{x}), c(\mathbf{x}, \mathbf{x}')\Sigma\right), \tag{2.22}$$

this implies that, for any input $\mathbf{x}$, the expectation is

$$\text{E}\left[f(\mathbf{x})|\boldsymbol{\beta}, \Sigma, R\right] = \boldsymbol{\beta}^T h(\mathbf{x}),$$

for any $\mathbf{x}$ and $\mathbf{x}'$, and the covariance is

$$\mathrm{Cov}\left[f(\mathbf{x}), f(\mathbf{x})' | \boldsymbol{\beta}, \Sigma, R\right] = c(\mathbf{x}, \mathbf{x}')\Sigma,$$

where $\boldsymbol{\beta}$ is the $q \times l$ coefficient matrix, $c(\mathbf{x}, \mathbf{x}')$ is the correlation functions over input space dependent on parameters R and Σ is the $l \times l$ covariance matrix across the outputs at an input. For any given input, they assume a common correlation length parameter to make the computing efficient, but for different inputs, individual covariance matrices for the outputs need to be calculated, which leads to computationally expensive matrix operations.

Rougier (2008) introduces an efficient emulating framework for simulators with multivariate outputs, known as the outer product emulator (OPE). Given l different outputs $s_1, \ldots s_l$ of the computer model $f(\cdot)$, Rougier (2008) fitted an emulator with a similar form to equation (2.22):

$$f_i(\mathbf{x}) = \sum_{j=1}^{p} \boldsymbol{\theta}_{ij} h_j(\mathbf{x}, s_j) + \epsilon_t(\mathbf{x}, s_i).$$

He assumes the residual covariance function is separable over the outputs and inputs,

$$c((\mathbf{x}, s), (\mathbf{x}', s')) = c^x(\mathbf{x}, \mathbf{x}') \times c^s(s, s'),$$

where the $c^x()$ and $c^s()$ are covariance functions over input space and output space individually. Rougier (2008) shows that the OPE is an efficient approach to building multivariate emulators, even with hundreds of simulator outputs or/and simulator evaluations. However, using a common covariance structure across the whole output space might be unsuitable for some cases. For example, for time-series data, the covariance might change over time.

Liu et al. (2009) propose another emulation approach for dynamic simulators. The time-series output computer model is modelled via a time-varying auto-regressive model (TVAR), with independent Gaussian processes for every time

$$f_t(\mathbf{x}) = \sum_{j=1}^{p} \boldsymbol{\theta_t} f_{t-1}(\mathbf{x}) + \epsilon_t(\mathbf{x}),$$

where the lag p is specified, $\epsilon_t(\mathbf{x})$ is a zero mean Gaussian process with $\mathrm{Cov}\left[\epsilon_t(\mathbf{x}), \epsilon_t(\mathbf{x}')\right] = \sigma_t c(\mathbf{x}, \mathbf{x}')$ and $\boldsymbol{\theta_t} = (\theta_{t1}, \ldots, \theta_{tp})$ is a vector of auto-regressive parameters. They assume the auto-regressive parameters vary over time, so that $\boldsymbol{\theta_t}$ is modelled following a random walk through time,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-01} + w_t, w_t \sim N(0, \varsigma^2\ W_t),$$

for a matrix W_t. This approach specification is extended for computer models that may exhibit chaotic behaviour in Williamson and Blaker (2014).

Another approach for handling spatio-temporal output is to use dimension reduction techniques to represent the high dimensional output as linear combinations of a fixed set of low dimensional basis. Bayarri et al. (2007) suggest that the wavelet decomposition would be a suitable basis representation: the emulators are built on the wavelet coefficients, and can be transformed back to the output space. Williamson et al. (2012) use B-splines, and constructed emulators on the linear coefficients of the basis. Principal component analysis (PCA) is the most usual default approach for emulating the high-dimensional model output due to its simplicity (Higdon et al., 2008; Wilkinson, 2010).

Principal component analysis (PCA) is a feature extraction method that transforms a number of correlated data into a set of uncorrelated variables called principal components (Jolliffe, 2011; Wold et al., 1987). Higdon et al. (2008) apply PCA to computer outputs $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$, where the model output $f(\mathbf{x}_i)$ is a vector of length l, $\mathbf{F}$ is a matrix that has dimension $l \times n$, and the n uncertain inputs are $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$. Basis vectors (principal components) can be obtained via singular value decomposition (SVD) of the standardised output matrix $\tilde{\mathbf{F}}$,

$$\tilde{\mathbf{F}} = (f(\mathbf{x}_1) - \mathbf{u}, \ldots, f(\mathbf{x}_n) - \mathbf{u}),$$

where $\mathbf{u}$ is the ensemble mean $\mathbf{u} = (u_1, \ldots, u_n)$, and u_i is the mean of i-th output,

$$u_i = \frac{1}{n}\sum_{j=1}^{n} f_i(\mathbf{x}_j).$$

In the majority of applications, the ensemble size n is typically less than the number of outputs l. Therefore, the basis vectors will not be full rank (it only represents the ensemble with n orthogonal directions). We denote the matrix of basis vectors as Γ, where

$$\Gamma = (\gamma_1, \ldots, \gamma_{n-1}),$$

is a collection of orthogonal vectors with length $n-1$. Each individual basis vector γ_i is a vector of length l, and there are $n-1$ basis vectors because the n-th as the ensemble mean has been removed. PCA proposes that the majority of the variability in F is explained by the first few basis vectors, $\Gamma_q = (\gamma_1, \ldots, \gamma_q)$ (Jolliffe, 2011). Following this property, the number of the components q could be selected by requiring that the majority of the variance in the ensemble is explained by projection onto the basis. For instance, Γ_q should explain more than 99% of the total variance of the data, as suggested by (Higdon et al., 2008). However, the later basis vectors explain low percentages of the variability in the ensemble, making accurate emulation for later coefficients difficult, and even if only the first few are used, the model output may still be correctly represented. More discussion will be given in Section 4.2

The vector of coefficients for the projection of the computer model output onto a given basis is

$$\mathbf{c}(\mathbf{x}) = \Gamma_q^T (f(\mathbf{x}) - \mathbf{u}),$$

where $\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), \ldots, c_q(\mathbf{x}))^T$. The output can then be represented as:

$$f(\mathbf{x}) = \Gamma_q \mathbf{c}(\mathbf{x}) + \mathbf{u} + \epsilon,$$

where ϵ is the reconstruction error, and the elements of coefficient vector $\mathbf{c}(\mathbf{x})$ are GPs over the input space. Higdon et al. (2008) fit univariate Gaussian process emulators for each set of coefficients $c_i(\mathbf{x})$ separately,

$$c_i(\mathbf{x}) \sim GP(0,\ \lambda_{wi}^{-1} c(\mathbf{x}, \mathbf{x}')).$$

They use a power exponential correlation function $c(.,.')$ and precision parameter λ_{wi}. Wilkinson (2010) fits GPs, with mean functions as in equation (2.2). The Gaussian process emulators' expectation and variance at $\mathbf{x}$ are given by

$$\mathrm{E}\left[c(\mathbf{x})\right] = (\mathrm{E}\left[c_1(\mathbf{x})\right],\ \ldots,\ \mathrm{E}\left[c_q(\mathbf{x})\right]),$$

and

$$\mathrm{Var}\left[c(\mathbf{x})\right] = \mathrm{diag}(\mathrm{Var}\left[c_1(\mathbf{x})\right],\ \ldots,\ \mathrm{Var}\left[c_q(\mathbf{x})\right]).$$

The $\mathrm{E}\left[c(\mathbf{x})\right]$ and $\mathrm{Var}\left[c(\mathbf{x})\right]$ can be transformed into the l-dimensional model output space:

$$\mathrm{E}\left[f(\mathbf{x})\right] = \Gamma_q \mathrm{E}\left[c(\mathbf{x})\right],$$

and

$$\mathrm{Var}\left[f(\mathbf{x})\right] = \Gamma_q \mathrm{Var}\left[c(\mathbf{x})\right] \Gamma_q^T + \Gamma_{-q} \Sigma_{-q} \Gamma_{-q},$$

where Γ_{-q} is Γ with the first q columns removed and Σ_{-q} is a diagonal matrix with diagonal elements containing the discarded eigenvalues (Salter et al., 2019; Wilkinson, 2010). We will revisit this approach with calibration in Section 2.5.3, Chapter 4, 5 and 6.

2.3.6 Diagnostics for Gaussian process emulators

To construct an emulator, several assumptions are made. Inappropriate assumptions can lead to poor emulator predictions of simulator outputs. Therefore, before using an emulator with other approaches, diagnostics must be used to validate and assess the adequacy of a Gaussian process emulator for representing the simulator.

The most popular diagnostic method is a 'leave one out' validation approach. In this approach, the output at one data point is removed from the ensemble runs, and an emulator is built with the remaining data. Then, we predict the removed simulation point using this emulator. This procedure is repeated for all runs. Prediction intervals are calculated for each run using the emulator's prediction: if the true function value does not lie within 3 (2 also commonly used) standard deviations of the mean (it can also be outside 90% or 95% prediction intervals), there will be a conflict between the emulator and the simulator (Rougier et al., 2009).

Bastos and O'Hagan (2009) also propose diagnostics that compare Gaussian process emulator predictions with simulation outputs. They separate the data set into two clusters, training data and validation data. The training data is used to build the emulator and the validation data is used to evaluate the performance of the emulator.

Let $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ represent the emulator training data with inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, and let $\mathbf{F}' = (f(\mathbf{x}'_1), \ldots, f(\mathbf{x}'_m))$ be the validation data with the validation input $\mathbf{X}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_m)$. For each validation input $\mathbf{x}'_i$, $i = 1, 2, \ldots, m$, the emulator gives predictive mean $\mathrm{E}\left[f(\mathbf{x}'_i)\right]$ with variance $\mathrm{Var}\left[f(\mathbf{x}'_i)\right]$.

Individual prediction errors are given by the difference between the simulator outputs $f(\mathbf{x}'_i)$ and the Gaussian process emulator predictive mean $\mathrm{E}\left[f(\mathbf{x}'_i)\right]$, for $i = 1, 2, \ldots, m$ at the same validation inputs. This method considers each standardised prediction error as a diagnostic,

$$D_i^I(f(\mathbf{x}'_i)) = \frac{f(\mathbf{x}'_i) - \mathrm{E}\left[f(\mathbf{x}'_i)\right]}{\sqrt{\mathrm{Var}\left[f(\mathbf{x}'_i)\right]}}, \quad i = 1, 2, \ldots, m. \tag{2.23}$$

Bastos and O'Hagan (2009) state that standardised large errors (larger than 2) suggest that there could be a conflict between the emulator and the simulator. We should expect 5% of points to fail this test if we have not been under-confident, and we may have extrapolation issues on the input space boundaries. In practice, if less than 5% of the errors are large and there is no systematic problem (e.g. all large errors are in the same region of parameter space) an emulator is considered to have been 'validated'.

In order to summarise the collection of individual standardised errors $D_i^I(f(\mathbf{x}_i'))$ into a single diagnostic, a χ^2 test can be used as a validation approach (Hills and Trucano, 1999). Define $D_\chi^2(\mathbf{F}')$ via

$$D_\chi^2(\mathbf{F}') = \sum_{i=1}^{m} D_i^I(f(\mathbf{x}_i'))^2. \tag{2.24}$$

The distribution of $D_\chi^2(f(\mathbf{x}_i'))$ converges to a chi-squared distribution with m-degrees freedom when the emulator has a large training data set.

A natural extension of equation (2.24) is the Mahalanobis distance between simulator and emulator outputs for the validation data set:

$$D_{MD}(\mathbf{F}') = (\mathbf{F}' - \mathrm{E}[f(\mathbf{X}')])^T (\mathrm{Var}[f(\mathbf{X}')])^{-1} (\mathbf{F}' - \mathrm{E}[f(\mathbf{X}')]). \tag{2.25}$$

The correlation among the outputs is captured/accounted for by the Mahalanobis distance. Similarly to individual prediction errors, extreme values (unexpectedly large or small) of $D_{MD}(y^*)$ indicate the existence of a conflict between the emulator and the simulator.

Individual prediction errors are correlated, so that they may be ineffective in finding the conflict between the emulator and the simulator. For example, if we found that two individual errors are individually small but were of opposite sign, there might be a conflict when they are strongly positively correlated (Bastos and O'Hagan, 2009).

Graphical methods are also frequently-used in diagnostics for Gaussian process emulators. For instance, we might plot the individual errors against the emulator's predictions, or plot the errors against the index and quantile-quantile plots(QQ-plots). Demonstrations of these graphical methods are presented within the subsequent Chapters.

If an emulator passes the validation tests, then it is usually assumed that the emulators have represented the simulators adequately. The test described above may be sufficient to assess the global performance of an emulator, but for some uncertainty quantification approaches, the primary concern should be the local performance of the emulator within the specific area, e.g. history matching (one of the calibration methods which will be introduced in Section 2.5). For example, to pass the 'leave one out' validation check, we would expect no more than 5% of points lie outside of 2 (or 3) standard deviation prediction intervals. However, the failed points might be near a specific local area which could result in biased results for uncertainty quantification without any indication for the analyst that this has occurred. No current methods exist for highlighting or solving this issue that we are aware of. We discuss this in details in Chapter 3, and present a novel contribution to address it.

2.4 Calibration

Hundreds of parameters can be introduced when constructing an computer model. For climate models, for example, these parameters control the behaviour of the atmosphere, oceans and a variety of other processes. Before using the model to study the real world, a parameter calibration (the climate modelling community refers calibration as 'tuning') step needs to be considered. Calibration of computer models broadly involves using partial and imperfect observations of the real world to learn which values of the model's input parameters lead to outputs that are consistent with real-world observations, given relevant uncertainties such as measurement error and model discrepancy (Kennedy and O'Hagan, 2001; Rougier, 2007). Calibration has been seen in a variety of applications, including oil reservoir modelling, climate systems, epidemiology, galaxy formation and agro-ecosystem modelling (Andrianakis et al., 2015; Lehuger et al., 2009; Salter and Williamson, 2016; Vernon et al., 2010; Williamson et al., 2015). Calibration with GP emulators is widely used to find input parameter values that give outputs consistent with the observation.

Kennedy and O'Hagan (2001) present a Bayesian approach to calibration. They split the input parameter into two parts: control variables $\mathbf{x}_{con}$ and calibration variables $\mathbf{x}_{cal}$, so that $\mathbf{x} = (\mathbf{x}_{con}, \mathbf{x}_{cal})$. Given control variable inputs, $\mathbf{x}_{con}$, the true value of the real process is denoted as $\varsigma(\mathbf{x}_{con})$.

Bayesian calibration requires a 'best input' assumption. Denoting $\mathbf{x}^*$ as the best calibration input, the computer model output $f(\mathbf{x}^*, \mathbf{x}_{con})$ with the best inputs $\mathbf{x}^*$ returns the best representation of the real system $\varsigma(\mathbf{x}_{con})$. A statistical model then links the computer model and reality via

$$\varsigma(\mathbf{x}_{con}) = \rho f(\mathbf{x}^*, \mathbf{x}_{con}) + \eta(\mathbf{x}_{con}), \tag{2.26}$$

where ρ is an unknown regression parameter, (the simplest choice is $\rho = 1$), and $\eta(\cdot)$ is the model discrepancy function, which is independent of $f(\mathbf{x})$ and $\mathbf{x}^*$. To learn about $\mathbf{x}^*$, we study the calibration data, comprised of the m observations $\mathbf{z} = (z_1, \ldots, z_m)^T$. For each observation i, z_i is an observation of $\varsigma(\mathbf{x}^i_{con})$

$$z_i = \varsigma(\mathbf{x}^i_{con}) + e_i, \tag{2.27}$$

where e_i is the observation error for the i-th observation and follows an independent normal distribution with zero mean.

Kennedy and O'Hagan (2001) assign Gaussian process prior as the prior information for both unknown functions, $f(\cdot)$ and $\eta(\cdot)$. The posterior distribution for the best input can then be derived from equation (2.32). They use the same equation as history matching to derive the posterior distribution, which we will consider in Section 2.5. In the thesis, we mainly focus on history matching, we will not go details of Bayesian calibration.

Rougier (2007) discusses calibration with only one set of the observations which have the same control variable. Therefore, there is only one suitable setting of the calibration input. Assumptions of this type are typically made in the context of climate models. There, they denote climate as a vector $y = (y_h, y_f)$, where y_h and y_f are corresponding to historical or current climate and future climate respectively. y_h depends on the available data, and the observation data of historical climate is denoted by z. The real climate and the observation is linked by an uncertain observation error e,

$$y_h = z + e. \tag{2.28}$$

there is a unique defined input $\mathbf{x}^*$, for which

$$y_h = f(\mathbf{x}^*) + \eta, \tag{2.29}$$

where η is the discrepancy of the computer model. Rougier (2007) specifies that the best input, model discrepancy and observation error are independent of each other. The distributions of the error terms are assumed to have the following form:

$$\eta \sim N(0,\, \Sigma_\eta), \quad e \sim N(0,\, \Sigma_e). \tag{2.30}$$

When the computer model has a one-dimensional output, these variances take a single value. In the higher dimensional case, the variances are covariance matrices that can represent different uncertainties across the different dimensions of the outputs. By applying calibration techniques to the historical data, a probability distribution over future data can be derived. This allows probabilistic inference for future climate to be made using an ensemble of climate model evaluations.

The approach developed by Kennedy and O'Hagan (2001) create a large impact for calibration area (Bayarri et al., 2007; Han et al., 2009; Higdon et al., 2004, 2008). However, Gramacy et al. (2015) state concerns over using the method by Kennedy and O'Hagan (2001). Tuo et al. (2015) show that the method by Kennedy and O'Hagan (2001) can lead to unreasonable estimates: the posterior distribution of the parameter will depend on the prior distribution of the discrepancy, even with a large number of observations. Based on this justification, Tuo and Wu (2016) suggest a new mathematical framework for calibration: they define L_2 consistency as a justification for the calibration approach. Based on this justification, which is called L_2 calibration, the standard modelling assumption for this calibration is the same as equations (2.32) and (2.27). The 'best input' $\mathbf{x}^*$ is defined as the parameter choice that minimise the L_2 norm between the physical process, $\varsigma(\cdot)$, and the computer output, $f(\mathbf{x},\cdot)$,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} \| \hat{\varsigma(\cdot)} - \hat{f}(\mathbf{x},\cdot) \|_{L_2(\mathcal{X})}, \tag{2.31}$$

where $\hat{f}(\mathbf{x},\cdot)$ is the emulator for $f(\mathbf{x},\cdot)$.

Plumlee (2017) proposes a new method for Bayesian calibration. In that method, the core methods of Bayesian calibration by Kennedy and O'Hagan (2001) are left intact, but they overcome the concerns associated where a parameter's posterior depends on the choice of the bias prior. Plumlee (2017) define the minimisation of loss as a parameter. For example, the generalised loss function used in Tuo et al. (2015). Under this assumption, the prior distribution on the bias should be orthogonal to the gradient of the computer model, and problems associated with Bayesian calibration are mitigated.

To calibrate in higher dimensions, the approach of Kennedy and O'Hagan (2001) can be extended to multivariate models by using a multivariate emulator for $f(\cdot)$. However, for a computer model with high dimensional output, this might be computationally expensive. To overcome these challenges, Higdon et al. (2008) use the basic framework of Kennedy and O'Hagan (2001) with principal components (as discussed in Section 2.3.5) to reduce the dimensionality of the observation and speed up the computations.

2.4.1 Discrepancy

In order to do calibration appropriately, no matter which method is used, specifying the model discrepancy term is an important step. Brynjarsdóttir and O'Hagan (2014) demonstrate the importance of model discrepancy, by illustrating that calibration without model discrepancy might lead to biased parameter estimation, and that this bias persists with increasing observations. Extrapolation and parameters inferences can be biased with a non informal discrepancy model. Therefore, including an accurate discrepancy is critical.

We have already discussed alternative discrepancy approach based on altering best input (e.g. Plumlee (2017) and Tuo and Wu (2016)) above. With the standard framework, a discrepancy variance (or GP kernel) needs to be specified, corresponding to beliefs about the extent to which the computer model represents the reality. Kennedy and O'Hagan (2001) represent the model discrepancy $\eta(\mathbf{x}_{con})$ as a Gaussian process with mean 0 and variance $\sigma^2 c(\mathbf{x}_{con}, \mathbf{x}'_{con}|\delta)$, where $c(\cdot,\cdot|\delta)$ is the squared exponential correlation function, expressing a prior belief that the discrepancy is a smooth function. The zero mean shows there is no prior expectation that discrepancy term is negative or positive. Brynjarsdóttir and O'Hagan (2014) believe that formulating discrepancy prior information is the key to the application of calibration. However, translating a simulator's deficiencies into a discrepancy prior is a difficult task. Moreover, if a weak prior belief is set, the posterior will rely heavily on the ensemble runs from the computer model in particular the difference between the ensemble members and the observation.

Chang et al. (2014, 2016) introduce a discrepancy model for calculating the discrepancy associated with ice sheet model calibration. The ice sheet model has high-dimensional binary spatial data. They calculate the signed proportion of mismatch between computer model output and the observation for each location s_j,

$$r_j = \sum_{i=1}^{p} \mathrm{sgn}(f_j(x_i) - z_j) I(f_j(x_i) \neq z_j),$$

where $\mathrm{sgn}(\cdot)$ is the sign function. Based on r_j, they define the discrepancy as

$$\boldsymbol{\eta}_j = \begin{cases} \log(\frac{1+r_j}{1-r_j}), & |r_j > c|, \\ 0, & otherwise, \end{cases}$$

for a chosen threshold c. The binary spatial data works well with this discrepancy method, because only a small number of model outputs are in conflict with the observations in the example, so a non-zero discrepancy is set for these locations.

For high dimensional output calibration, Salter et al. (2019) model the discrepancy as a multivariate Gaussian process with mean zero and covariance matrix,

$$\Sigma_{\eta}^{ij} = v_i v_j c(s_i,\ s_j),$$

where $c(.,.')$ is the correlation function between two locations s_i and s_j, and they specify the variance v_i based on whether the location i is in the key feature set S. This method specifies the different tolerance of mismatch between observation and model output for different locations, and the method could be extended to k sets of indicators for spatial output.

Zhang et al. (2018) adopt a L_2 discrepancy model for the field output calibration,

$$\eta(\mathbf{x}) = \parallel z - f(\mathbf{x}) \parallel_{L_2},$$

where $\| \cdot \|_{L_2}$ is the L_2 norm. This calibration locates the best input, which minimises the discrepancy model. For spatial output, a L_2 discrepancy term is not able to capture correctional errors across the computer outputs. In fact, it weights each error independently. For a field output computer or time-series model, specifying a discrepancy covariance matrix would carry more physical information than defining a L_2 discrepancy.

2.5 History matching

History matching is another prominent technique for computer model calibration (Craig et al., 1996; Vernon et al., 2010; Williamson et al., 2013). History matching attempts to identify the parts of the input parameter space that are likely to result in mismatches between computer outputs and observations by iteratively removing those regions of parameter space where a good match is very unlikely to exist. Previous research has applied history matching to many fields, including oil reservoir modelling (Craig et al., 1997; Cumming and Goldstein, 2010), epidemiology (Andrianakis et al., 2015, 2017), galaxy formation and (Bower et al., 2010; Rodrigues et al., 2017; Vernon et al., 2010) climate systems (Edwards et al., 2011; Gladstone et al., 2012; McNeall et al., 2013; Salter and Williamson, 2016; Williamson and Blaker, 2014; Williamson et al., 2013, 2015).

To employ the history matching method, the inconsistencies between the computer model and the physical reality must be examined at the beginning (Goldstein and Rougier, 2009). In history matching, similar to Bayesian calibration, simulator inadequacy can be expressed through a relation of the form

$$y_i = f_i(\mathbf{x}^*) + \eta_i, \tag{2.32}$$

where y_i represents reality, $\mathbf{x}^*$ is the 'best input' of the computer model and η_i is the model discrepancy. The discrepancy term is independent of $f_i(\mathbf{x})$ and $\mathbf{x}^*$ (Craig et al., 1996; Goldstein and Rougier, 2004). To learn about $\mathbf{x}^*$, we have observations z_i with unknown measurement error e_i such that

$$z_i = y_i + e_i, \tag{2.33}$$

where e has zero mean and is uncorrelated with of η_i. Because of the majority of computer models are expensive and can take a long time to run, emulators are used to predict the output of the computer model $f(\mathbf{x})$. For any parameter setting, $\mathbf{x}$, the emulator gives an expectation, $\mathrm{E}[f(\mathbf{x})]$, with variance $\mathrm{Var}[f(\mathbf{x})]$. The implausibility measure is used to eliminate any parameter settings which produce outputs which are 'too far' from the observations. For a single output at a given value, $\mathbf{x}$, implausibility is defined as

$$\mathcal{I}_i(\mathbf{x}) = \frac{|z_i - \mathrm{E}[f_i(\mathbf{x})]|}{\sqrt{\mathrm{Var}[z_i - \mathrm{E}[f_i(\mathbf{x})]]}}. \tag{2.34}$$

Under equations (2.32) and (2.33), when $\mathbf{x} = \mathbf{x}^*$

$$\begin{aligned}
\mathrm{Var}\left[z_i - \mathrm{E}\left[f_i(\mathbf{x})\right]\right] &= \mathrm{Var}\left[z_i - f_i(\mathbf{x}^*) + f_i(\mathbf{x}^*) - \mathrm{E}\left[f_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[z_i - y_i + y_i - f_i(\mathbf{x}^*) + f_i(\mathbf{x}^*) - \mathrm{E}\left[f_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[e_i + \eta_i + f_i(\mathbf{x}^*) - \mathrm{E}\left[f_i(\mathbf{x})\right]\right] \qquad (2.35) \\
&= \mathrm{Var}\left[e_i\right] + \mathrm{Var}\left[\eta_i\right] + \mathrm{Var}\left[f_i(\mathbf{x}^*) - \mathrm{E}\left[f_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[e_i\right] + \mathrm{Var}\left[\eta_i\right] + \mathrm{Var}\left[f_i(\mathbf{x})\right],
\end{aligned}$$

such that only the observation, observation error variance, discrepancy variance and emulator prediction variance is required to calculate the implausibility (Salter et al., 2019; Williamson et al., 2017). However, because of the emulator uncertainty, a small value of implausibility can either occur when the distance between the observation and emulator prediction is small, or when the emulator is extremely uncertain. Therefore, the input with the optimal of implausibility cannot guarantee that the model is consistent with observations (Williamson et al., 2015).

Instead of finding a good choice of calibration input, large values of $\mathcal{I}_i(\mathbf{x})$ can indicate that $f(\mathbf{x})$ is too far from the observations, even with all the uncertainties. History matching uses the implausibility measure to rule out parameter settings that deviate most significantly from the observations. The space that has not yet been ruled out, 'Not Ruled Out Yet' (NROY) space, $\mathcal{X}_{NROY}$, is defined as

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_i(\mathbf{x}) \leq T\}, \qquad (2.36)$$

where T is a chosen threshold. A common choice is $T = 3$ based on the three sigma rule (Pukelsheim, 1994), which states that for any unimodal continuous probability distribution 95% of the population lies within three standard deviations of the mean.

2.5.1 Refocusing

History matching does not seek to identify NROY space using a single set of computer model evaluations, but through iteratively designed experiments known as 'waves' (Vernon et al., 2010). In the first wave, the emulator is constructed based on an ensemble at a set of points which cover the whole input space, $\mathcal{X}$. History matching is used to rule out space from the initial space through equation (2.42) and the NROY space after wave 1 is denoted as $\mathcal{X}^1$. A new emulator can then be constructed using an ensemble $f(\mathbf{x}_2)$ at a new set of points from the first wave NROY space $\mathbf{x}_2 \in \mathcal{X}^1$. After that, the second wave of history matching can be carried out. This process is called 'refocusing'. In general, at wave k, a new emulator is built from a new set of points $\mathbf{x}_k$ drawn from the wave $k-1$ NROY space, $\mathbf{x}_k \in \mathcal{X}^{k-1}$, and the wave k NROY space is defined as

$$\mathcal{X}^k = \{x \in \mathcal{X}^{k-1} | \mathcal{I}^k(x) \leq T_k\}, \tag{2.37}$$

where $\mathcal{I}^k(x)$ is the implausibility (equations 2.34 and 2.38) defined on $\mathcal{X}^{k-1}$ and with the wave k emulator and T_k is a selected threshold for wave k.

The refocusing process first builds emulators using an ensemble in the full parameter space, and then the implausibility is used to cut down parameter space. As later waves are reached, the density of model runs is increased in the reduced space. Since the emulators only need to be accurate over the reduced parameter space at later waves, the emulator is more accurate than the initial wave, where we need it to be. For each wave, applying different implausibility measurements can provide a lot of flexibility (Williamson et al., 2017).

A key question for this process is how many waves to run or when to stop. Williamson et al. (2015) suggest stopping the process when the entire space is ruled out by a certain metric. The stopping criteria could also rely on the emulator variance. When the emulator variance is small enough, such that it will not significantly change the NROY space in the next wave, it would be meaningless to continue the process (Williamson et al., 2017). However, in real applications, these two scenarios are rare. Either budget or a time limit would typically be the reason why the process stops.

2.5.2 Implausibility in Many Dimensions

When computer model output represents multiple metrics, a multivariate version of the implausibility must be used (Craig et al., 1997),

$$\begin{aligned}\mathcal{I}(\mathbf{x}) &= (\mathbf{z}-\mathrm{E}[f(\mathbf{x})])^T\mathrm{Var}[\mathbf{z}-\mathrm{E}[f(\mathbf{x})]]^{-1}(\mathbf{z}-\mathrm{E}[f(\mathbf{x})]) \\ &= (\mathbf{z}-\mathrm{E}[f(\mathbf{x})])^T(\mathrm{Var}[e]+\mathrm{Var}[\eta]+\mathrm{Var}[f(\mathbf{x})])^{-1}(\mathbf{z}-\mathrm{E}[f(\mathbf{x})]),\end{aligned} \tag{2.38}$$

where $\mathbf{z}$ and $\mathrm{E}[f(\mathbf{x})]$ are vectors with length r, and $\mathrm{Var}[e]$,$\mathrm{Var}[\eta]$ and $\mathrm{Var}[f(\mathbf{x})]$ are $r \times r$ variance matrices. This implausibility takes account of the correlation between different components of the model outputs (Vernon et al., 2010), and is especially useful for climate models (Williamson et al., 2017). To calculate the implausibility, the variance matrices for the observation error and discrepancy need to be specified.

For setting the cutoff threshold, the implausibility $\mathcal{I}(\mathbf{x})$ can be compared with a Chi-squared distribution with r degrees of freedom (Vernon et al., 2010). Then the threshold T could be 95% or 99.5% of a Chi-squared distribution with r degrees of freedom. A NROY space $\mathcal{X}_{NROY}$ can then be defined with this threshold

$$\mathcal{X}_{NROY} = \left\{\mathbf{x} \in \mathcal{X} | \mathcal{I}(\mathbf{x}) \leq \chi^2_{r,\,0.995}\right\}. \tag{2.39}$$

However, there are some limitations of using the implausibility measure defined in equation (2.38). Firstly, that measurement does not allow different thresholds for different dimensions of the model output: the measurement can give a larger value when a single poorly matching component occurs in one dimension (Craig et al., 1997).

When specifying discrepancy error and observation error for each of the outputs only, separate implausibility measures for each metric $\mathcal{I}_i(\mathbf{x})$ can be evaluated with individual emulators. Even if the model outputs are correlated, accurate emulators for each metric of the outputs could be as good as the multivariate emulator. In this case, the implausibility can be defined as the maximum of the implausibility measures for each output (Craig et al., 1997). The maximum implausibility measure is

$$\mathcal{I}_M(\mathbf{x}) = \max_i \mathcal{I}_i(\mathbf{x}). \tag{2.40}$$

Using the maximum implausibility measure might be too sensitive, meaning that there is a chance to rule out good parameter choices. When the cutoff threshold is set as 3, it is expected that we would rule out up to 5% of model outputs which are consistent with the observations. Vernon et al. (2010); Williamson et al. (2017) use the second or third largest implausibility, the second implausibility is given by

$$\mathcal{I}_{2M}(\mathbf{x}) = \max_i \left(\mathcal{I}_i(\mathbf{x}) \backslash \mathcal{I}_M(\mathbf{x})\right),$$

and the third largest implausibility is

$$\mathcal{I}_{3M}(\mathbf{x}) = \max_i \left(\mathcal{I}_i(\mathbf{x}) \backslash \{\mathcal{I}_M(\mathbf{x}),\ \mathcal{I}_{2M}(\mathbf{x})\}\right).$$

In general, the q-th maximum implausibility is

$$\mathcal{I}_{qM}(\mathbf{x}) = \max_i \left(\mathcal{I}_i(\mathbf{x}) \backslash \{\mathcal{I}_M(\mathbf{x}),\ \mathcal{I}_{2M}(\mathbf{x}),\ \ldots,\ \mathcal{I}_{(q-1)M}(\mathbf{x})\}\right). \tag{2.41}$$

The NROY space with the q-th maximum implausibility is then also given by

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_{qM}(\mathbf{x}) \leq T\}. \tag{2.42}$$

All of the implausibilities (e.g. $\mathcal{I}(\mathbf{x})$ and $\mathcal{I}_{qM}(\mathbf{x})$) are computed with emulator predictions, which indicates that the performance of history matching relies on the emulator prediction. Before cutting areas of parameter space, diagnostics must be used to assess the adequacy of an emulator. We presented a variety of diagnostics that compare Gaussian process emulator predictions and simulation outputs at validation points in Section 2.3.6. Whilst the test described in Section 2.3.6 may be adequate to assess the global performance of an emulator, when history matching the primary concern should be the local performance of the emulator near $\mathbf{x}^*$. When the emulator is inaccurate near $\mathbf{x}^*$, good parameter choices can easily be ruled out. We present novel methods for detecting these cases and a Local Voronoi Tessellation method for a robust approach to calibration that ensures that the true NROY space is retained and parameter inference is not biased in Chapter 3.

2.5.3 Multivariate history matching using basis projection methods

For computer model with high-dimensional output, history matching with the basis projection method for emulation is proposed (multivariate emulation is introduced in Section 2.3.5 (Salter et al., 2019). For basis emulators, multivariate history matching can be applied in different ways.

Firstly, the implausibility can be calculated on the coefficient space. The univariate implausibility for each coefficient $c_i(\mathbf{x})$ follows equation (2.34), and is calculated as

$$\mathcal{I}_i(\mathbf{x}) = \frac{|c_i(\mathbf{z}) - \mathrm{E}\left[c_i(\mathbf{x})\right]|}{\sqrt{\mathrm{Var}\left[c_i(\mathbf{z}) - \mathrm{E}\left[c_i(\mathbf{x})\right]\right]}}, \tag{2.43}$$

where $c_i(\mathbf{z})$ is defined as the projection of the observation onto the i-th column of Γ, and $\mathbf{c}(\mathbf{z}) = (c_1(\mathbf{z}), \ldots, c_q(\mathbf{z}))$, where

$$\mathbf{c}(\mathbf{z}) = \Gamma^T(\mathbf{z} - \mathbf{u})) = \Gamma^T(\mathbf{z} - \mathbf{u})). \tag{2.44}$$

The variance term $\mathrm{Var}\left[c_i(\mathbf{z}) - \mathrm{E}\left[c_i(\mathbf{x})\right]\right]$ can also be written as in terms of the observation error and discrepancy on the coefficient space:

$$\begin{aligned}
\mathrm{Var}\left[c_i(\mathbf{z}) - \mathrm{E}\left[c_i(\mathbf{x})\right]\right] &= \mathrm{Var}\left[c_i(\mathbf{z}) - c_i(\mathbf{x}^*) + c_i(\mathbf{x}^*) - \mathrm{E}\left[c_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[c_i(\mathbf{z}) - c_i(y_i) + c_i(y_i) - c_i(\mathbf{x}^*) + c_i(\mathbf{x}^*) - \mathrm{E}\left[f_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[c_i(e) + c_i(\eta) + c_i(\mathbf{x}^*) - \mathrm{E}\left[c_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[c_i(e)\right] + \mathrm{Var}\left[c_i(\eta)\right] + \mathrm{Var}\left[c_i(\mathbf{x}^*) - \mathrm{E}\left[c_i(\mathbf{x})\right]\right] \\
&= \mathrm{Var}\left[c_i(e)\right] + \mathrm{Var}\left[c_i(\eta)\right] + \mathrm{Var}\left[c_i(\mathbf{x})\right],
\end{aligned}$$

where $c_i(\eta)$ and $c_i(e)$ are defined as the projection of the observation error and discrepancy onto the i-th column of Γ respectively. For each basis, the variance of the projection of observation error $\mathrm{Var}\left[c_i(e)\right]$ and discrepancy $\mathrm{Var}\left[c_i(\eta)\right]$ could be set as the i^{th} vector of $n \times n$ matrices $\mathrm{Var}\left[\mathbf{c}(e)\right]$ and $\mathrm{Var}\left[\mathbf{c}(\eta)\right]$,

$$\mathrm{Var}\left[\mathbf{c}(e)\right] = \mathrm{Var}\left[\Gamma^T(e - \mathbf{u})\right] = \Gamma^T \mathrm{Var}\left[e\right]\Gamma,$$

and

$$\mathrm{Var}\left[\mathbf{c}(\eta)\right] = \mathrm{Var}\left[\Gamma^T(\eta - \mathbf{u})\right] = \Gamma^T \mathrm{Var}\left[\eta\right]\Gamma.$$

Given equations (2.40) and (2.41), the maximum implausibility measure and q-th maximum implausibility can be applied on the coefficient space.

Another option is to use the multivariate implausibility in equation (2.38) on coefficient space. Following the emulators in equation (2.2), the multivariate implausibility in equation (2.38) then becomes

$$\begin{aligned}
\mathcal{I}_c(\mathbf{x}) &= \left(\mathbf{c}(\mathbf{z}) - \mathrm{E}\left[c(\mathbf{x})\right]\right)^T \mathrm{Var}\left[\mathbf{c}(\mathbf{z}) - \mathrm{E}\left[c(\mathbf{x})\right]\right]^{-1}\left(\mathbf{c}(\mathbf{z}) - \mathrm{E}\left[c(\mathbf{x})\right]\right) \\
&= \left(\mathbf{c}(\mathbf{z}) - \mathrm{E}\left[c(\mathbf{x})\right]\right)^T \left(\mathrm{Var}\left[\mathbf{c}(\eta)\right] + \mathrm{Var}\left[\mathbf{c}(e)\right] + \mathrm{Var}\left[c(\mathbf{x})\right]\right)^{-1}\left(\mathbf{c}(\mathbf{z}) - \mathrm{E}\left[c(\mathbf{x})\right]\right).
\end{aligned} \tag{2.45}$$

One final way to do multivariate history matching is analogous to the multivariate calibration in Wilkinson (2010), where $\mathrm{E}\left[c(\mathbf{x})\right]$ and $\mathrm{Var}\left[c(\mathbf{x})\right]$ are transferred back to the original model output space. The implausibility measurement in equation (2.38) can then be applied.

Salter et al. (2019) discuss how PCA-based methods can cause a 'terminal case analysis', where the calibration can fail when the observations are not in the linear subspace defined by the PCA. Choosing the low dimensional space that best explains the maximum variability in the ensemble, cannot guarantee that the key features of the observations are preserved. Salter et al. (2019) use a rotation algorithm to find a calibration-optimal basis which considers the observation reconstruction error in the basis selection step. The observation reconstruction error measures how accurately the observation can be represented by a basis Γ_q:

$$\mathcal{R}_{\mathbf{W}}(\Gamma_q, \mathbf{z}) = ||\mathbf{z} - \Gamma_q(\Gamma_q^T \mathbf{W}^{-1}\Gamma_q)^{-1}\Gamma_q^T \mathbf{W}^{-1}\mathbf{z}||_{\mathbf{W}}, \tag{2.46}$$

where $||\mathbf{v}||_{\mathbf{W}} = \mathbf{v}^T\mathbf{W}^{-1}\mathbf{v}$ is the norm of vector $\mathbf{v}$, $\mathbf{W}$ is a $l \times l$ positive definite weight matrix. By setting $\mathbf{W} = \mathrm{Var}[\eta] + \mathrm{Var}[\eta]$, $\mathcal{R}_{\mathbf{W}}(\Gamma_q, \mathbf{z})$ is equivalent to equation (2.38) if the emulator variance is zero. If $\mathcal{R}_{\mathbf{W}}(\Gamma_q, \mathbf{z})$ is bigger than the history matching threshold, the representation of z on Γ_q would be ruled out. To avoid this situation, they developed a calibration-optimal basis vector, $\Gamma^* = \Gamma\Lambda$ (Λ is a rotation matrix), to minimise the reconstruction error, $\mathcal{R}_{\mathbf{W}}(\Gamma_q^*, \mathbf{z})$, subject to a constrain that Γ_q^* explains the enough variability in the ensemble for building emulators for the new coefficients.

The projections for $f(\mathbf{x})$, $\mathbf{z}$, $\mathrm{Var}[\eta]$ and $\mathrm{Var}[e]$ onto basis Γ_q with the given $\mathbf{W}$ are

$$\begin{aligned}
\mathbf{c}(\mathbf{x}) &= (\Gamma_q^T\mathbf{W}^{-1}\Gamma_q)^{-1}\Gamma_q^T\mathbf{W}^{-1}(f(\mathbf{x}) - \mathbf{u}),\\
\mathbf{c}(\mathbf{z}) &= (\Gamma_q^T\mathbf{W}^{-1}\Gamma_q)^{-1}\Gamma_q^T\mathbf{W}^{-1}(\mathbf{z} - \mathbf{u}),\\
\mathrm{Var}[\mathbf{c}(\eta)] &= (\Gamma_q^T\mathbf{W}^{-1}\Gamma_q)^{-1}\Gamma_q^T\mathbf{W}^{-1}\mathrm{Var}[\eta]\,\mathbf{W}^{-1}\Gamma_q(\Gamma_q^T\mathbf{W}^{-1}\Gamma_q)^{-T},\\
\mathrm{Var}[\mathbf{c}(e)] &= (\Gamma_q^T\mathbf{W}^{-1}\Gamma_q)^{-1}\Gamma_q^T\mathbf{W}^{-1}\mathrm{Var}[e]\,\mathbf{W}^{-1}\Gamma_q(\Gamma_q^T\mathbf{W}^{-1}\Gamma_q)^{-T}.
\end{aligned} \tag{2.47}$$

When $\mathbf{W}$ is the identity matrix, these projections are the same as standard PCA. The coefficient implausibility following equation (2.45) could be calculated with the projections in equation (2.47), and we denote the implausibility as $\mathcal{I}_{\mathbf{W}}(\mathbf{x})$. Salter and Williamson (2019) show that computer models with large output fields can be history matched efficiently with $\mathcal{I}_{\mathbf{W}}(\mathbf{x})$. By setting $\mathbf{W} = \text{Var}[\eta] + \text{Var}[e]$, we have

$$\mathcal{I}(\mathbf{x}) = \mathcal{I}_{\mathbf{W}}(\mathbf{x}) + \mathcal{R}_{\mathbf{W}}(\Gamma_q, \mathbf{z}). \tag{2.48}$$

Hence, the calculation of the expensive implausibility $\mathcal{I}$ using $l \times l$ matrix inversions only requires $q \times q$ matrix inversions at each $\mathbf{x}$, without any loss of information.

When we try to calibrate computer models with high-dimensional output, the model has to be able to represent the pattern we want our model to replicate. PCA-based history matching introduced by Salter et al. (2019) works well when the parameters are responsible for the strength of various patterns. However, if the parameters control the position of patterns e.g. shifting currents, existing basis calibration methods will fail. We will explore kernel based methods from machine learning for history matching to overcome the limitation of current basis methods.

In this thesis, we are interested in improving history matching. Two limitations of history matching will be demonstrated in the following chapters. The first shows that standard diagnostic checks for GP emulators may be satisfied yet lead to good regions of parameter space being ruled out. In Chapter 3, we illustrate the problem, offer a new diagnostic to test for this situation and propose an extension to history matching for cases when this diagnostic raises a flag that ensures that current emulator can safely be used for remove space. The second limitation addressed is related to the above discussion on PCA-based history matching for high dimensional output. We argue that current methods are only really effective if key signals do not move around the output space as the inputs are charged and if the model and reality can, should and do have the timing and location of these signals in common. When this is not the case, similar to the issue presented by Salter and Williamson (2019), we are using a basis that leads to a terminal case. To overcome this, we extend history matching to kernel feature space. In so doing, we discover that we need to revisit the notion of what it means for a model to be implausible and to generalise the concept of history matching accordingly.

Chapter 3

Local Voronoi tessellations for robust multi-wave calibration of computer models

3.1 Introduction

Chapter 2 reviews the well-known calibration methodology. History matching uses emulators to rule out parameter settings which lead to models that deviate most significantly from observations. The remaining space is Not Ruled Out Yet (NROY) space. A limitation of history matching occurs when an emulator is unable to simulate the unknown target NROY space, $\mathcal{X}^*$, effectively, even if it seems to pass all standard emulator diagnostic checks (introduced in Section 2.3.6). Emulator diagnostic checks may be adequate to assess the global performance of an emulator, but when using history matching the primary concern should be the local performance of the emulator near the target NROY space. Note, there is no history matching tailored diagnostic.

When an emulator at a given wave is incorrect outside the target NROY space, the worst thing that could happen is that a poor parameter choice is retained. Although, that parameter choice could still be removed by a future wave with a more accurate emulator. However, when the emulator is incorrect inside the target NROY space, good parameter choices could be irrecoverably ruled out. For instance, the "leave one out" validation approach usually expects about 5% validation points to lie outside of the two standard deviation prediction interval. Once the emulators meet this requirement, it is viewed as appropriate to use history matching. However, if the points that failed the validation test are near the target NROY space, that could lead to biased calibration results. Since the target NROY is unknown, to confirm whether the failed points are in the target NROY space or not, we compare those points with the observations. If an isolated large error is found near the observations, there is a chance that the emulator has not accurately simulated the target NROY space. Poor simulation may result in the true NROY space being ruled out without any indication to the analyst that this has occurred.

Figure 3.1 demonstrates the performance of a GP emulator and the results of the first wave of history matching, compared to the true NROY space found for the model directly. The 1D function used was first considered by Xiong et al. (2007), which takes the form

$$y(x) = \sin(30(x-0.9)^4)\cos(2(x-0.9)) + \frac{(x-0.9)}{2}.$$

We use a 10-run maximin Latin Hypercube (LHC) (Morris and Mitchell, 1995) to design the runs to train the emulators used in this example.

The true function (black line) and the corresponding emulator posterior mean (blue line) with uncertainty (blue dashed lines) is shown in the top right panel. We can see that it is hard to predict the region $[0, 0.4]$ by directly comparing the true function and the emulator prediction. The leave one out diagnostic plot (top left panel) indicates that the emulator has failed at one point, but this single failure would not usually be deemed serious enough to invalidate the emulator. The result of history matching with this emulator is shown in the bottom left panel, and the bottom right panel shows a zoomed in version of the result. The blue interval defines the NROY space after the first wave of history matching, and the red interval defines the true NROY. Comparing the true model calibration results with the emulator calibration results, we find that nearly one third of the true NROY space is ruled out.

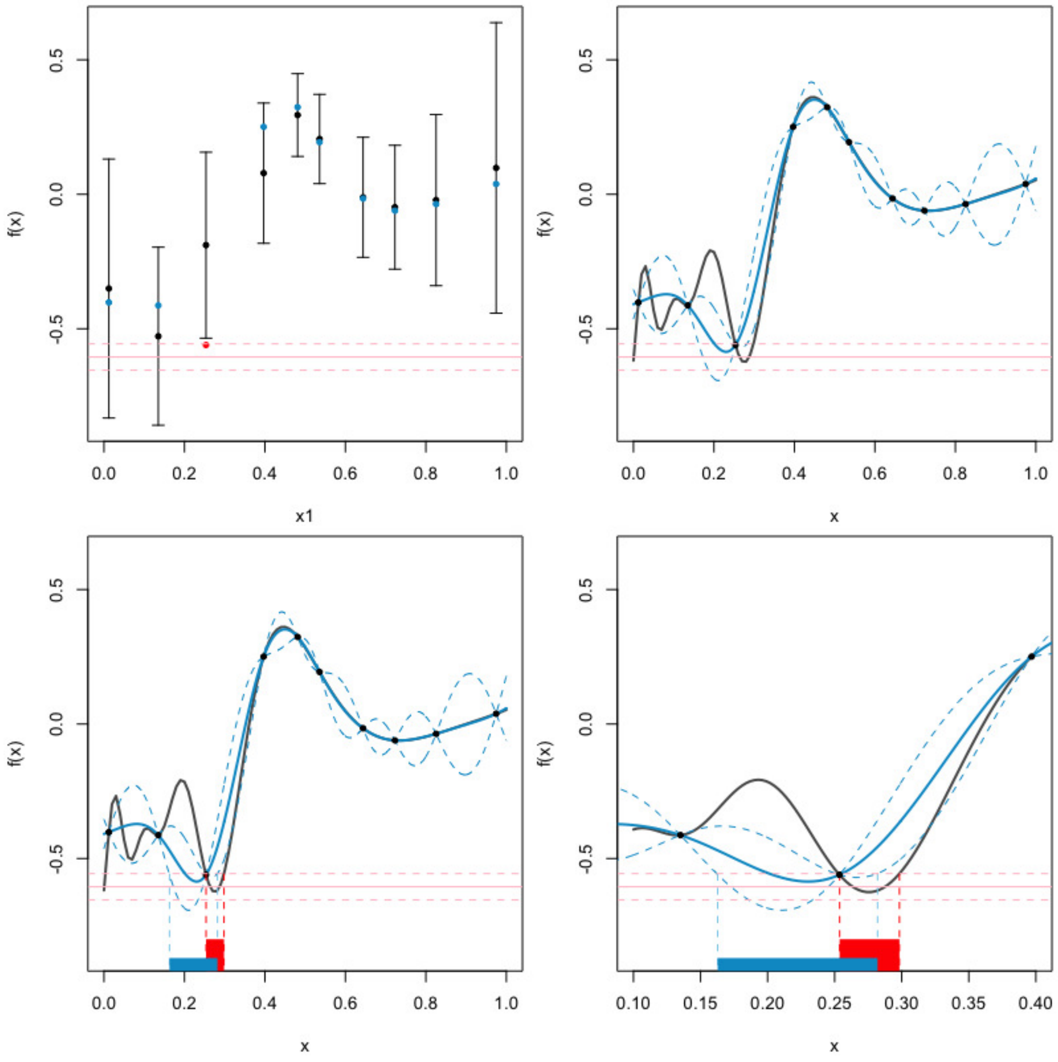


Fig. 3.1 *Top left:* Leave One Out diagnostic plot against x. The emulator prediction and two standard deviation intervals are given in black. The true function values are in blue if they lie within two standard deviation prediction intervals, or red otherwise. The pink line and the pair of red dotted lines represent the observation with observation error and discrepancy in all 4 panels. *Top right:* Emulator performance for the 1D model. The true function is represented by the black curve and ten black points are inputs used to train the emulator. The blue line represents the emulator posterior mean, and the blue dotted lines give the two standard deviation prediction intervals. *Bottom left:* History matching results and the true NROY region. The blue interval defining the NROY space after first wave, the red interval defining true NROY $\mathcal{X}^*$. *Bottom right:* As with bottom left but enlarged over the NROY regions.

In an application with many emulators being used to cut a high dimensional parameter space using many metrics, such critical cases may often occur and be difficult to catch. A trainable nugget would enable simpler functions to fit the data Gramacy and Lee (2012), and this might alleviate the problem in some cases, particularly if we have achieved what looks like an acceptable fit by over-fitting. We would generally use a trainable nugget when building emulators, for these reasons. However, in most cases where we see this pathology, the emulator fits well across the parameter space, but near the true NROY there is an issue which would not normally raise a diagnostic flag. In these cases, it is likely that the overall fit is good, but that there is some local non-stationarity near where the function behaves like the data. In such cases, if a trainable nugget had not already been used, one would be unlikely to solve the problem and may lead to reduction in global performance (i.e. we may still have a good emulator from a validation perspective, but might rule out less space given that there will be a larger posterior variance). With or without a trainable nugget, we should still expect 5% of predicted points to lie outside our prediction intervals. If most or all of these occur near true NROY, we may still rule out good parts of that space by mistake.

The history matching literature usually recommends only ruling space out if 3 or more outputs have large implausibility, and this might insure against ruling out true NROY in some cases (if we have more than 1 or 2 metrics). However, a poor emulator near the true NROY region may often be a feature of the design that can appear for emulators of all metrics, and flagging this issue in a key region might make us wary of trusting emulators for other metrics in that region.

Larger cutoff thresholds are sometimes used in earlier waves to retain more space until we are more confident about ruling out. If this is done routinely, it may still be that true NROY is ruled out using the larger threshold. If this is done to ensure that all points where there may be an issue are not ruled out, the cutoff may have to be set so high as to ensure that no space would be ruled out at all. Indeed, no current methods exist for highlighting this potential issue that we are aware of, nor of proceeding robustly with history matching following detection. In this chapter, we will discuss which factors can contribute towards ruling out good parameter choices, and we will then present a novel Local Voronoi Tessellation design that can be used for robust multi-wave calibration, ensuring that the true NROY space is retained without biasing the parameter inference.

This chapter has the following structure. Section 3.2 demonstrates a novel detecting method, taking place after the emulator diagnostic check, that attempts to determine whether the emulator could have failed near the target NROY space. Section 3.3 introduces our Local Voronoi Tessellation design, alongside other tested methods. Section 3.4 presents the procedure for the robust multi-wave history matching of computer models. In Section 3.5, we apply our methods to two illustrative examples. In Section 3.6, we apply our method to the output of the French climate model, IPSL-CM, an atmosphere model that is used to predict planetary atmospheres, including the Earth and other planets (Mars, Titan, Venus) (Bony and Dufresne, 2005; Hourdin et al., 2017; Voldoire et al., 2013). We then conclude in Section 3.7 with a discussion.

3.2 Detection

In practice, emulators are typically fitted before applying history matching. Once the emulator passes diagnostic checks, history matching can be applied. Above used a simple one dimensional example to show that an emulator could pass a diagnostic check across the whole input space, but still fail near the target NROY space. A detection method is required before applying history matching, to test whether this situation occurs in practice. We describe our approach to detection below.

Let $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ represent the emulator training data with inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. In cases where we do not have validation runs, we use leave one out cross-validation as an initial diagnostic: one run is removed from the ensemble runs, an emulator is built with the remaining data, and this emulator is used to predict the removed run. This procedure is repeated for all runs. For each input $\mathbf{x}_i$, the emulator gives predictive mean $\mathrm{E}[f(\mathbf{x}_i)]$ with variance $\mathrm{Var}[f(\mathbf{x}_i)]$. As presented in Section 2.3.6, using the prediction results we can compute standardised errors $D_i(f(\mathbf{x}_i))$ which are given by the differences between the simulator outputs and the Gaussian process emulator predictive mean at the same input:

$$D_i(f(\mathbf{x}_i)) = \frac{f(\mathbf{x}_i) - \mathrm{E}[f(\mathbf{x}_i)]}{\sqrt{\mathrm{Var}[f(\mathbf{x}_i)]}}. \tag{3.1}$$

Bastos and O'Hagan (2009) propose that standardised large errors suggest that there could be a conflict between the emulator and the simulator. Based on that, we define the points with large $D_i(f(\mathbf{x}_i))$ as candidate problem points. We firstly identify the emulator 'failed' set, $\mathbf{X}_F \subseteq \mathbf{X}$, with

$$\mathbf{X}_F = \{x_i \in \mathbf{X} | D_i(f(x_i)) > T_F\}, \tag{3.2}$$

where T_F is a threshold which is usually set as 2 (or even 1.96 with the argument that if the emulator were a good fit, 5% of these points should raise a flag). If $\mathbf{X}_F$ is not empty then $\mathbf{X}_F$ is a candidate set of points that may indicate the existence of an emulator that could remove regions of target NROY, $\mathcal{X}^*$. This could only happen if a point in $\mathbf{X}_F$ were inside or close to the target NROY space.

To determine whether the failure points are near the target NROY space, we define the NROY space found by the computer model directly (without an emulator) as "true" NROY space or target NROY. Target NROY space $\mathcal{X}^*$ is defined as

$$\mathcal{X}^* = \left\{ \mathbf{x} \in \mathcal{X} | \frac{|z - f(\mathbf{x})|}{\sqrt{\mathrm{Var}[e] + \mathrm{Var}[\eta]}} \leq T \right\}, \tag{3.3}$$

where T is the history matching threshold, z is the observation, e is the observation error and η is the model discrepancy. For each point $x_m \in \mathbf{X}_F$, $f(\mathbf{x}_m)$ is compared with the observation z to determine whether the emulator failure points are near $\mathcal{X}^*$, and we form a set of 'doubt points' that could be close enough to target NROY to cause an issue. The doubt points set, $\mathbf{X}_D$, is defined using equation (3.3) as

$$\mathbf{X}_D = \left\{ \mathbf{x}_m \in \mathbf{X}_F \mid |z - f(\mathbf{x}_m)| \leq T\sqrt{\text{Var}[e] + \text{Var}[\eta]} \right\} \tag{3.4}$$

We define the set of remaining points $\mathbf{X}_N$, $\mathbf{X}_N = \mathbf{X} \backslash \mathbf{X}_D$.

Standard history matching can be applied directly if $\mathbf{X}_D$ is empty. Otherwise, in principle, with existing methods we might have to seek to add further runs from the computer model and/or find a more complex or bespoke emulation. The latter may not always be easy or even possible. Emulation and history matching are increasingly popular with modellers as a way of calibrating their own models. Developing a bespoke emulator using a tailored kernel or mean function may be possible for UQ experts in any given problem, but it raises barriers to wider application in general that may not be necessary. Further model runs near $\mathbf{X}_D$ will likely enable standard methods to work well and fix the issue in many cases. However, in applications like climate modelling where running the model requires specialist equipment (e.g. supercomputers) and scientist time, it often the case that runs need to be done in batches, and time/budget constraints mean that only a small number of batches will be possible. Wasting one of these resources just to improve part of an emulator may sacrifice a whole potential wave of history matching. To automatically tune the computer model, such as the climate model in the HIGH-TUNE project, we propose a robust multi-wave history matching.

Our method is based on the notion that the emulator works well enough in most of the parameter space so that it can be used for history matching anyway. However, in regions of space near $\mathbf{X}_D$, it would be safer not to remove space at all, and to re-sample that space in wave two. Essentially, we will add further runs of our simulator to correct the errors in this region, but we will firstly cut out all of the space that can safely be cut out with the existing emulator. The goal then, when $\mathbf{X}_D$ is not empty, is to separate the whole input space into two regions, one containing $\mathbf{X}_D$, $\mathcal{X}_D \supseteq \mathbf{X}_D$, and the other containing $\mathbf{X}_N$, $\mathcal{X}_N \supseteq \mathbf{X}_N$. History matching will be only applied on $\mathcal{X}_N$ and the $\mathbf{X}_D$ will be retained throughout in such a way that we can ensure that history matching in the latter region only will not discard parts of $\mathcal{X}^*$.

3.3 Finding $\mathcal{X}_D$

Given the doubt points set, $\mathbf{X}_D$, and the normal data set $\mathbf{X}_N$, we require a partition $\mathcal{X} = \mathcal{X}_D \cup \mathcal{X}_N$. The doubt region, $\mathcal{X}_D$, will be retained in the next wave to ensure that good parameter choices will not be ruled out. To find $\mathcal{X}_D$, several different approaches can be employed to partition the input space into two distinct regions. One conventional approach is to use a classification method (Bailey, 1994; Clifford et al., 1975; Duda et al., 2012). In our case, the response variables are "doubt points" and "normal data".

We first attempted to use common classification methods on our problem, such as logistic regression (Menard, 2002). However, these classification methods failed because the data is very unbalanced. To deal with imbalanced data, a new Local Voronoi Tessellation is used as a classification method to identify the doubt region, $\mathcal{X}_D$ (Murphy, 1990). We develop the approach below.

3.3.1 Failed classification methods

Logistic regression is a statistic model that uses a logistic function to model the discrete binary outputs (Hosmer Jr et al., 2013; Kleinbaum et al., 2002; Menard, 2002). Logistic regression is commonly used for classification with binary outputs: given the model inputs, the logistic function itself models the probability of the binary outputs. Mathematically, the logistic function is defined as

$$\log\left(\frac{P(\mathbf{x})}{1 - P(\mathbf{x})}\right) = \beta\mathbf{x},$$

where $\mathbf{x}$ is the model inputs, β is a length $m+1$ coefficient vector, $\beta = [\beta_0, \ldots, \beta_{m+1}]$, and $P(\mathbf{x})$ is the probability of response being 1 at given input x. There are multiple methods for estimating the value of parameters β. Maximum likelihood estimation is a commonly used approach to determine the values of the parameters: the value of β can be estimated by maximising the likelihood function:

$$L(\beta) = \prod_{i=1}^{n} P(\mathbf{x}_i)^{\mathrm{y}_i}(1 - P(\mathbf{x}_i))^{1-\mathrm{y}_i},$$

where $(\mathbf{x}, \mathbf{y})$ is the training data. To use logistic regression to classify the doubt and normal regions, we denote y is 1 when the simulator output is in the doubt points set, else y is 0, $y(\mathbf{x}_i)$. Apart from that, maximising a posterior (MAP) and Markov chain Monte Carlo (MCMC) can be used within a Bayesian setting. A common prior used for logistic regression is $\beta_j \sim N(\mu_j,\, \sigma_j^2)$, where μ_j is usually set as zero and σ_j^2 is usually chosen from the range 10 to 100.

We use the simulator inputs as the logistic function inputs, for a given input x, the logistic function gives the probability for a unclassified sample point to be class 1, $P(y = 1|\beta,\, \mathbf{x}) = P(\mathbf{x})$. Solving $P(\mathbf{x})$ gives,

$$P(\mathbf{x}) = \frac{\exp(\beta\mathbf{x})}{1 + \exp(\beta\mathbf{x})} = \frac{1}{1 + \exp(-\beta\mathbf{x})}.$$

We have that $y = 1$ with the probability $P(\mathbf{x})$ and $y = 0$ with the probability $1 - P(\mathbf{x})$. To use the predication as a classifier, we set a cutoff level T_{cl}. Classifier accuracy is a widely used measurement to choose the cutoff level. The classifier gives a set of classes based on the probability $P(\mathbf{x})$,

$$\text{Class}(\mathbf{x}) = \begin{cases} 1, & P(\mathbf{x}) > T_{cl} \\ 0, & \text{otherwise.} \end{cases}$$

We consider the performance of logistic regression classification on the 1D example introduced in Section 3.1. The results of using the detection clearly show that there exist one doubt point. By setting this doubt point response as 1 and others as 0, we can apply a logistic regression classification. Figure 3.2 demonstrates the example classification results. To set a suitable cutoff level, we

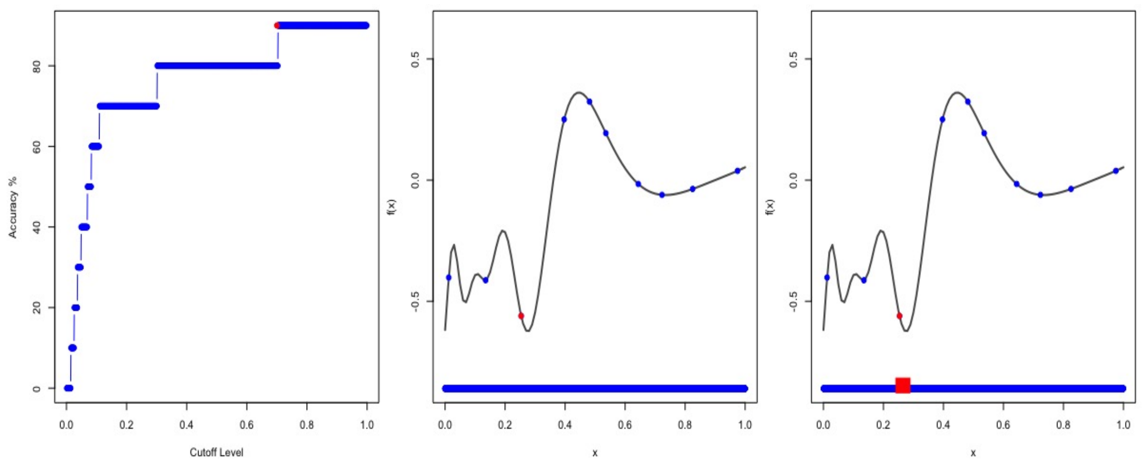


Fig. 3.2 Logistic regression classification plots. *Left:* The logistic regression cutoff level against the classification accuracy, the red dot is the automatic selection of the threshold which returns the highest accuracy. *Middle:* Logistic regression classification results with the automatic selection threshold. The true function is plotted in black, red dot is in the doubt points set and the blue dots are normal points. The blue bar shows the classification results, which means all the input space is in the normal region. *Right:* True classification results, the blue narrow bar should be the normal region and the red bar is the doubt region.

plot the classifier accuracy against the cutoff level by leave one out cross-validation in the left panel. By automatically choosing the cutoff to reach the highest accuracy (90%), the cutoff level becomes 0.8. The classification results with automatically selected cutoff level are plotted in the middle panel. To quantify the classification performance, we compare the classification result with the "true" doubt region (overlapped region of the target NROY region (red bar) and the emulator failed region (red bar) in Figure 3.1). On the middle panel and the right panel, the real function is plotted in black, the red point is the only doubt point, and the blue points are the normal points. The results of the classification method are shown by the blue bar and red bar on the bottom, which represent the normal region $\mathcal{X}_N$ and doubt region $\mathcal{X}_D$ respectively.

The classifier attempts to put all points in the normal region, and thus fails to capture the doubt region adequately. By classifying all of the points into the normal region, the classifier can attain the highest possible accuracy (around 90%). To get an idea of why this classification problem is performing poorly, we check the proportion of each category. The doubt data $\mathbf{X}_D$ only contains one data point, which is only 10% of the whole training data. The classification training data will always be imbalanced, not only in this one-dimensional example, because the doubt data set is constructed by the failed emulator data, and it thus should not exceed 5-10% (else we would not consider it a good emulator). Indeed, most classification data sets do not have equal amounts of data in each class. The slight difference rarely matters, but when the data is excessively imbalanced it will lead to problems: classifiers are very likely to predict all the points in the normal class, and a high accuracy can be easily achieved.

To ensure that the target NROY space is retained after history matching, when there might be a doubt region, we need to identify the whole doubt region. Therefore, we should avoid the potential error of predicting the actual doubt points as lying in the normal region. Such an error is called a Type I error, also known as false positive (FP). A type II error, also called false negative (FN), occurs when a classifier incorrectly predicts a normal point to be in the doubt class (Casella and Berger, 2002). A significant type II error might retain a vast region of the input space that makes the history matching wave meaningless.

Instead of using accuracy to set the cutoff level, we want the cutoff level to be used to quantify the tradeoff between type I and type II error. A Receiver Operating Characteristic curve (ROC) can be used to decide which cutoff value to choose (Zweig and Campbell, 1993). The ROC curve is a graphical method, allowing us to balance the True Positive Rate (TPR) and the False Positive Rate (FPR).The TPR and the FPR are calculated by using the True Positive counts (TP), True Negative counts (TN), FP and FN, as:

$$TPR = \frac{TP}{TP+FN},$$

$$FPR = \frac{FP}{TN+FP}.$$

A ROC curve is generated by plotting the TPR (the y axis) against the FPR (x axis) at various cutoff settings. By specifying the different cost for making a type I error and a type II, the total cost can be calculated.

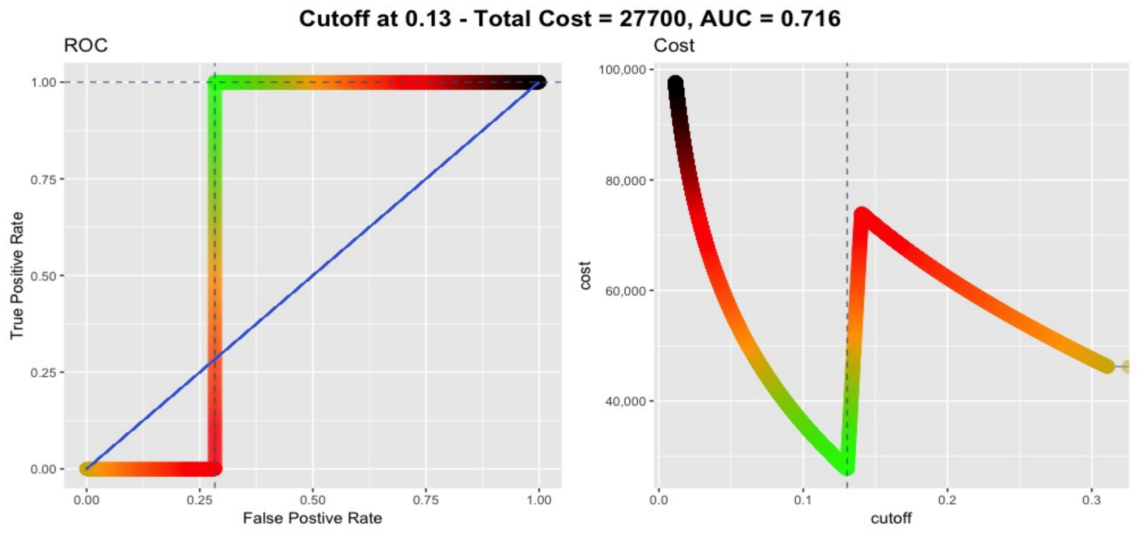


Fig. 3.3 The optimization choice of cutoff level by ROC. *Left:* The ROC curve, the colour on the ROC curve shows the cost corresponding with each point, which is associated with the right panel. Green represents the lowest total cost, and black means the highest total cost. The tilted blue line declares the boundary of an average model, with a 0.5 area under the curve. *Right:* The total cost against different cutoff value choice. The black dotted line denotes where that optimal cutoff value and minimum total cost lies.

Figure 3.3 shows the ROC curve for the one-dimensional toy model in the left panel, and a total cost against various cutoff settings in the right panel. By assigning the cost for a type I error and type II error to be 200 and 10 respectively (we set a large weight for type I error to find all possible doubt region). The optimal cutoff to minimise the total cost is calculated to be 0.13, with the corresponding total cost 27700. To compare the new cutoff level 0.13 selected by a ROC curve with the cutoff level 0.8, we calculate the type I error and type II error for the two cutoff choices with a new set of validation data. The results of this are shown in Table 3.1. The real doubt region is around 2.8 per cent of the whole input space. With a cutoff level 0.13, we catch all of the valid doubt region, but around 36.7% of the normal region is misclassified. With a cutoff level of 0.8, the classifier classifies the whole right doubt region wrongly, but the accuracy of the classifier is around 90%. This example demonstrates that type I error can be controlled as zero by choosing a suitable logistic regression cutoff value, but, accordingly, it leads to low accuracy, with a prominent type II error. Both logistic regression classifiers results for this simple one-dimensional case are not ideal. In practice, we might expect more than one doubt point when we use extensive training data with high

	Type I error	Type II error
Cutoff level=0.1	0	36.7%
Cutoff level=0.8	2.8%	0

Table 3.1 Type I and type II errors for the two different cutoff level of the logistic regression classifier.

dimensional inputs. Using logistic regression for problems such as this leads to a large chance of ruling out good parameter choices, especially if the structure of the doubt points is complicated, such as being located in different regions of the input space. The new cutoff seems to offer an alternative way to do logistic regression classification with unbalanced training data, but the results show that the method is far away from being as required.

To overcome the imbalanced data, under-sampling and oversampling are two techniques which can produced class-balanced data. The simpler undersampling method randomly removes sampling from the majority class to strike a rough balance of two classes. However, undersampling should/can not be applied to our extreme unbalanced data, which may only contain one or two inputs in the minority class.

Oversampling involves the generation of more training data in the minority class. To avoid running the simulator additional times, we use oversampling techniques to add more data to the doubt class. One machine learning method that can be used for classification on class-imbalanced data is the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002). SMOTE uses synthetic data generation to increase the number of samples in the minority class, so that the data set becomes balanced. SMOTE first finds the n-nearest neighbours in the minority class for each of the samples in the class, then random samples are generated on the lines between the neighbours. Though promising, SMOTE requires at least two points in $\mathbf{X}_D$ which in many instances will not apply.

Whilst well-known methods for classification are typically built based on the logical or statistical reasoning, those approaches may fail in cases with imbalanced data. Primarily, we want the classification to be able to identify a small and unreliable region around the doubt points which gives nearly zero type I error and a small type II error. Inspired by finding the local region around doubt points, we propose a strategy for classification based on partitioning of the input space through Voronoi tessellation, where the doubt region could be set as a local Voronoi tile area.

3.3.2 Local Voronoi Tessellation

A Voronoi tessellation works with a finite number, n, of points in the Euclidean plane, where $2 < n < \infty$. The n points are labelled by $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T \in \mathcal{X}$, each $\mathbf{x}_i$ is distinct from any other point, $\mathbf{x}_i \neq \mathbf{x}_j$. Supposing the n points are a set of centres of an n-cell Voronoi tessellation, a Voronoi tessellation then partitions the space into n convex cells which are called Voronoi regions, $\mathcal{V}_i$ (Gallier, 2008). A Voronoi region is defined as the set of points in $\mathcal{X}$, whose 'nearest' point is $\mathbf{x}_i$, given by

$$\mathcal{V}_i = \left\{\mathbf{x} \in \mathcal{X} | d(\mathbf{x}, \mathbf{x}_i) \leq d(\mathbf{x}, \mathbf{x}_j)\right\}, \qquad \forall j \in \{1, \ldots, n\} \backslash i. \tag{3.5}$$

where $d(\mathbf{x}, \mathbf{x}')$ is a distance function, commonly defined as the Euclidean distance,

$$d_E(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{p} (x_i - x_i')^2},$$

or the Manhattan distance

$$d_M(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{p} |x_i - x_i'|.$$

In Figure 3.4 we provide a visual demonstration of Voronoi regions on a two-dimensional input space. We used a 20-run maximin Latin Hypercube(LHC) design to sample 20 input points, which are shown in the figure with black points. The input space is separated into 20 Voronoi cells $\mathcal{V}_i$, $i = 1, \ldots, 20$ associated with the 20 input points. Each Voronoi cell has a boundary made up of Voronoi edges: the Voronoi edges are the black boundary lines shown in the figure. Mathemati-

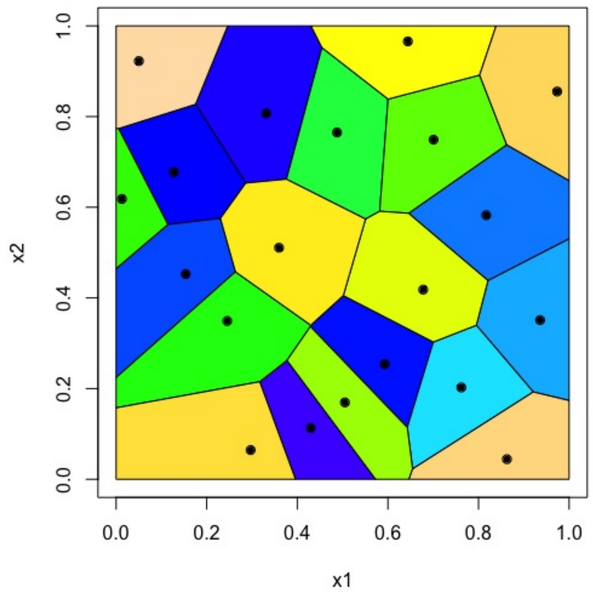


Fig. 3.4 Voronoi diagrams of 20 points under a Euclidean distance function.

cally, if

$$\mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset,$$

we set the $\mathcal{V}_i \cap \mathcal{V}_j$ set to be a Voronoi edge.

When history matching, our correlation function, $c(x,x')$, provides an appropriate notion of distance between inputs. Our n inputs $x \in \mathcal{X}$ can be used as the centres of a Voronoi tessellation. We cannot use the correlation directly (as the distance between the two points increases, the correlation decreases), therefore we define a Voronoi Tessellation $\mathcal{V}_i$ with the emulator posterior correlation function as

$$\mathcal{V}_i = \left\{ x \in \mathcal{X} \,\middle|\, |c^*(x,x_i)| \geq |c^*(x,x_j)| \right\}, \qquad \forall j \in \{1, \ldots, n\} \backslash i. \tag{3.6}$$

Finding a Voronoi tessellation can be computationally challenging when the design is large or when the input dimensions become even moderately sized (e.g. > 4). However, we do not need to map the whole parameter space. Our goal is to find the local Voronoi tiles that cover $\mathbf{X}_D$, ensuring that all possible values we have not run, but might doubt our emulator for near the true NROY are included. Specifically, a local Voronoi tessellation will partition the input space into a doubt region, $\mathcal{X}_D \supseteq \mathbf{X}_D$, and normal region, $\mathcal{X}_N \supseteq \mathbf{X}_N$, by finding $\mathcal{X}_D$. We define a local

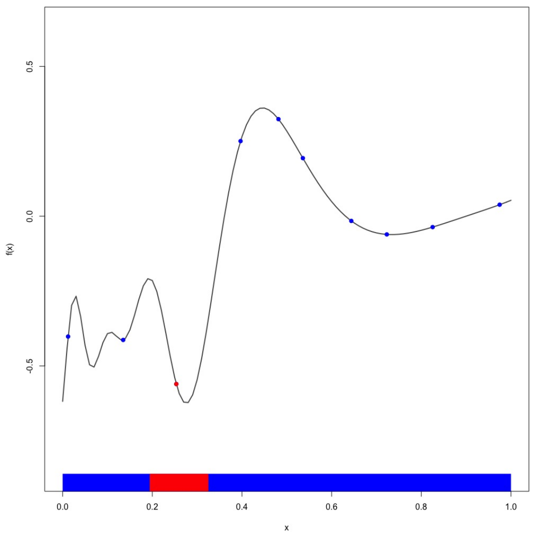


Fig. 3.5 Voronoi Tessellation classification results. The blue bar denotes the normal region which can be employed in history matching. The red bar represents the retained doubt region.

Voronoi tessellation $\mathcal{X}_D = \bigcup_{\{i:x_i \in \mathbf{X}_D\}} \mathcal{V}_i$, with

$$\mathcal{V}_i = \{x \in \mathcal{X} \,\Big|\, |c^*(x,x_i)| \geq |c^*(x,x_j)|\}, \qquad \forall j \text{ s.t. } x_j \in \mathbf{X}_N.$$

To test the local Voronoi tessellation classification, we apply it to the 1-dimensional toy model, for which we present the results of classification in Figure 3.5. As before, the blue points and red point represent the normal data and the doubt data respectively, the doubt region is highlighted in a red bar on the bottom and the normal region is highlighted in a blue bar. To compare with the logistic regression classification method, we also compute the classification accuracy, type I error and type II error for the local Voronoi tessellation classification, given in Table 3.2. Similarly to logistic regression classification by ROC-set cutoff, the local Voronoi tessellation classification guarantees that the Type I error is zero: all the true doubt regions are identified. In addition, the results also shows that there is a large increase of both accuracy and Type II error associated with these improvements.

	Accuracy	Type I error	Type II error
VT classification	91%	0	8.09%

Table 3.2 Local Voronoi tessellation classification results.

3.3.3 Local augmentation

The local Voronoi tessellation, $\mathcal{X}_D$, represents the union of convex sets around the doubt points. Given that the emulator failed to predict the doubt points, but was able to predict the surrounding normal points, we can deduce that there is a region between each normal point and each doubt point where the emulator is reliable (it predicts the normal points well) and a region near the doubt points where it is not. Though $\mathcal{X}_D$ will contain much of the doubt region, if not all, there is no guarantee that it should contain the whole badly performing region. We therefore include an augmentation step to ensure that as much of the region where the emulator cannot be trusted (near target NROY) is included in $\mathcal{X}_D$ as is possible.

For any design point x_i, the design point x_j with the largest value of $c^*(x_i,x_j)$ is the point with the most influence on $f(x)$ in the Gaussian process. For $x_i \in \mathbf{X}_N$, we want to ensure that the most influential points are not doubt points where we do not trust our emulator, as this would indicate a possibility that some part of the region bordering $\mathcal{X}_D$ and near to x_j is unreliable. Our augmentation step adds all points from $\mathbf{X}_N$ with this property to $\mathbf{X}_D$ before arriving at a final $\mathcal{X}_D$.

Specifically, for each $x_i \in \mathbf{X}_N$, let

$$x_{k(i)} = \arg \max_{k:x_k \in \mathbf{X}} c^*(x_i,x_k).$$

Let $\mathbf{X}_{D'} = \left\{x_i : x_{k(i)} \in \mathbf{X}_D\right\}$, be the set of points in $\mathbf{X}_N$ whose most influential point is a doubt point. We then augment the doubt set by $\mathbf{X}_{D'}$ so that $\mathbf{X}_D = \mathbf{X}_D \cup \mathbf{X}_{D'}$, and compute the local Voronoi tessellation on the augmented set as before.

3.4 Robust history matching

Having isolated a region of parameter space, $\mathcal{X}_N = \mathcal{X} \setminus \mathcal{X}_D$ in which we feel confident enough in our emulator to rule out parameter space, we can history match in just that region. Specifically, we define

$$\mathcal{X}'_N = \{x \in \mathcal{X}_N : \mathcal{I}(x) \leq T\} \tag{3.7}$$

where $\mathcal{I}(x)$ is the implausibility measurement. The NROY space, $\mathcal{X}^1_{NROY}$, after the first wave is defined as

$$\mathcal{X}^1_{NROY} = \mathcal{X}_D \cup \mathcal{X}'_N. \tag{3.8}$$

The detection step is adopted before history matching in each wave, and the Local Voronoi tessellation approach is applied when there is a sign of doubt points. Our robust history matching uses the following algorithm:

1. Fit an emulator using the training data $\mathcal{X}$.
2. Calculate the emulator's cross-validation mean and variance for the training data, and compute the standardised errors for each training data input by equation (3.1).
3. Identify the failed set $\mathbf{X}_F$, and then, for each point in the failed set, compare the failed data output with the observation via equation (3.4) to discover a doubt point set, $\mathbf{X}_D$.
4. Apply history matching if $\mathbf{X}_D = \emptyset$. Else, set a local Voronoi region for the doubt points set to identify the doubt region $\mathcal{X}_D$, the rest part of input space is denoted as the normal region $\mathcal{X}_N$.
5. Apply history matching in $\mathcal{X}_N$, The NROY space $\mathcal{X}_{NROY}$ is defined as

$$\mathcal{X}_{NROY} = \mathcal{X}_D \cup \mathcal{X}'_N, \tag{3.9}$$

where $\mathcal{X}'_N$ is the retained part (NROY space) of normal region,

$$\mathcal{X}'_N = \{x \in \mathcal{X}_N : \mathcal{I}(x) \leq T\}. \tag{3.10}$$

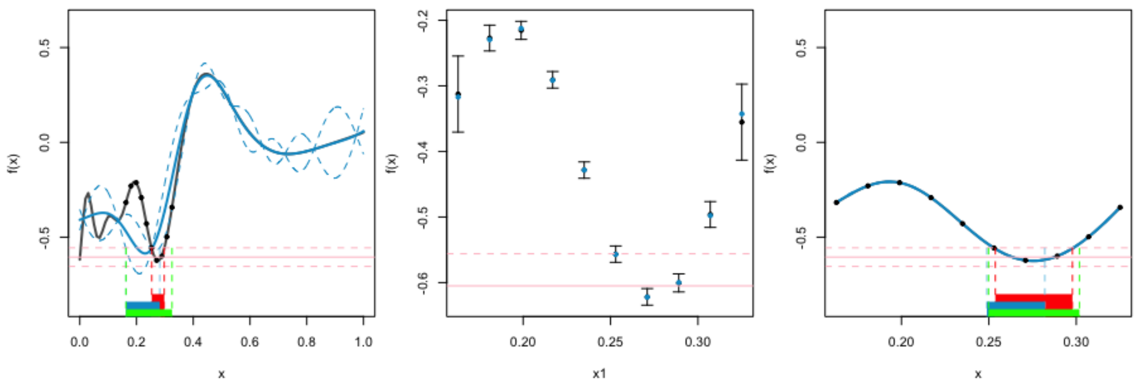


Fig. 3.6 The results of multi-wave calibration of a 1-dimensional model. *Left:* the results of robust history matching after one wave. The true function is represented by the black curve and the ten black points are input points. The blue line represents the emulator posterior mean, and the blue dotted lines give the two standard deviation prediction intervals. The red interval defines the true NROY space, the blue interval defines the NROY space by standard history matching and the green interval defines the NROY space by our robust history matching approach. *Centre:* leave One Out diagnostic plot against x for a second wave emulator. The emulator prediction and two standard deviation intervals are given in black. The true function values are in blue if they lie within two standard deviation prediction intervals, or red otherwise. The pink line and the pair of red dotted lines represent the observation with observation error and discrepancy. *Right:* history matching second wave result. The green interval defining the NROY space after the second wave, the red interval defining true NROY $\mathcal{X}^*$.

3.5 Numerical examples

We apply the methodology of the last section to the 1-dimensional function from the introduction and a 5-dimensional function described below. We use the R package `DiceKriging` (Roustant et al., 2012) to construct the emulators throughout.

3.5.1 The 1-dimensional function

We present our robust history match of the one-dimensional toy function in Figure 3.6. The wave one result is shown in the left panel, where the green interval defines the first wave NROY space, and the red interval defines the true NROY space. To compare with standard history matching results, we plot the blue interval, which is the first wave NROY space. After the first wave, our approach retains 16.24% of the input space, which covers all of the true NROY space (4.6% input space). A second wave is performed with ten randomly selected runs within NROY space:

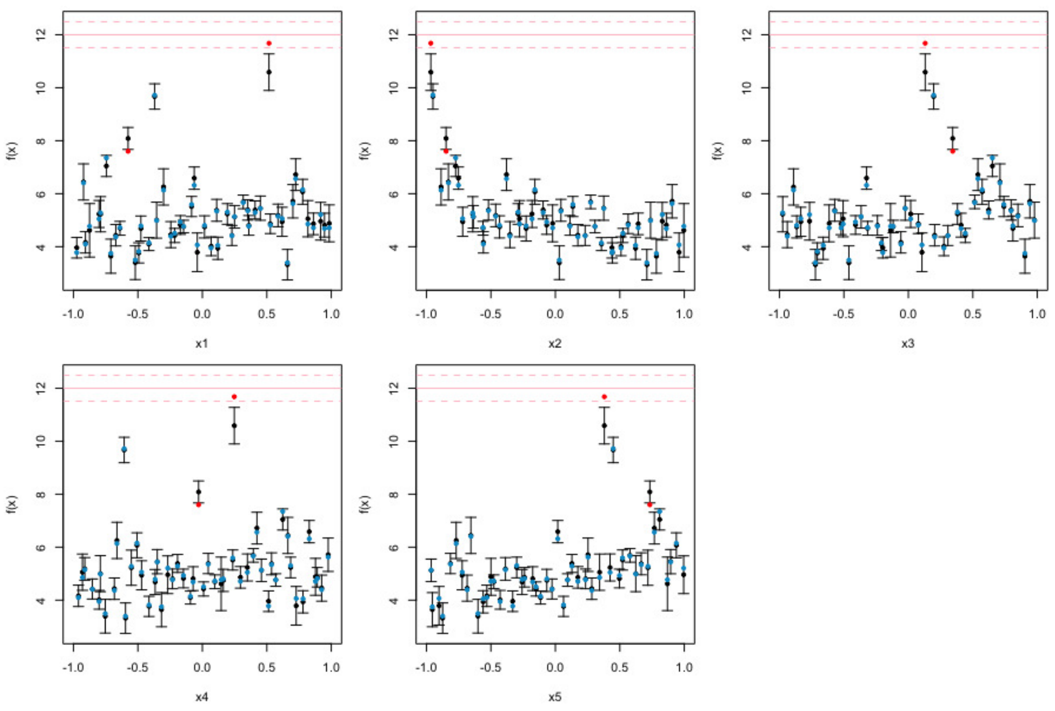


Fig. 3.7 Leave one out diagnostic plots. Each panel represents leave one out predictions from an emulator against one of the 5 inputs. Black points and error bars are from the emulator posterior mean and two standard deviation prediction intervals. The true function values are in green if they lie within two standard deviation prediction intervals, or red otherwise.

the leave one out diagnostic plot against inputs for the second wave emulator is shown in the middle panel, highlighting that there are no doubt points. Having passed the diagnostic check, we apply the standard history matching on the first wave NROY space. The right panel shows the second wave results. After the second wave, the NROY space retains 5.3% input space, and the figure shows that all of the target NROY space is retained. The blue interval shows the second wave results following standard history matching, and even though the second wave performs well, more than one third of the target NROY space has already been ruled out by the first wave.

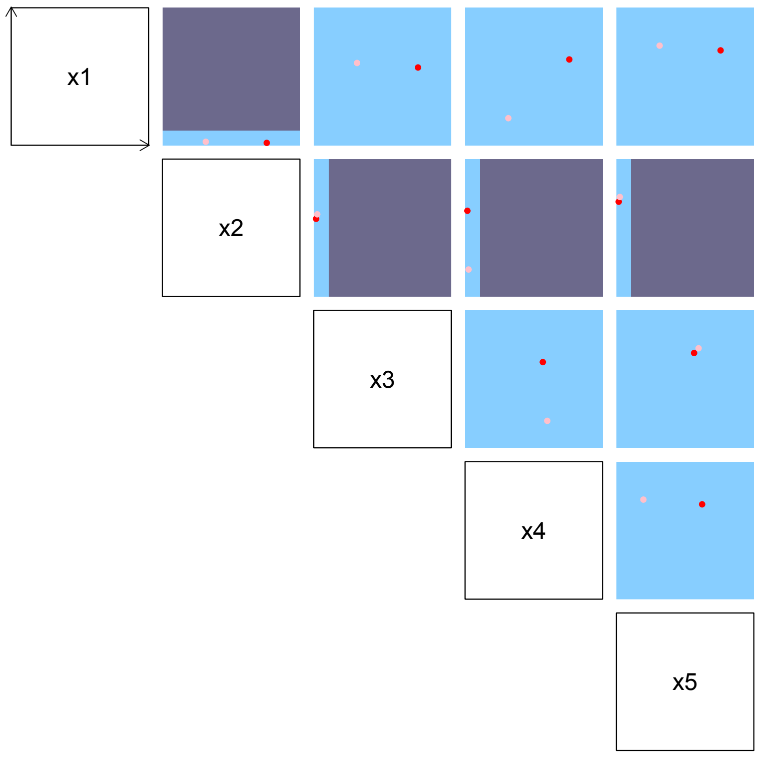


Fig. 3.8 Local Voronoi cell plots over each pair of parameters. The red point is the doubt point and the pink points are selected by our augmentation step. The blue region is the Local Voronoi cell of the doubt points which is the doubt region.

3.5.2 A 5-dimensional function

In order to examine the performance of our method in higher dimensions, we look at a 5-dimensional function,

$$f(\mathbf{x}) = \sqrt{x_1} + \frac{1}{\sqrt{x_2}} + x_3 + \sin(x_4) + \exp(x_5).$$

Note that this function tends to infinity as x_2 tends to zero which could happen in a physical model as an input approaches a hard physical boundary. We use a 50 member maximin Latin Hypercube (LHC) to select points for the first wave, and use the function evaluations to construct an emulator.

Leave one out diagnostics against each input are presented in Figure 3.7. Black points and error bars are calculated from the emulator posterior mean and two standard deviation prediction intervals. The true function values are in red if they do not lie within two standard deviation prediction intervals, which are then assigned to the emulator failed data set. By eye, we see that the emulator

	Standard wave 1	Standard wave 2	Standard wave 3	Robust wave 1	Robust wave 2	Robust wave 3
Retained NROY volume	0.6373%	0.4445%	0.2664%	4.6660%	3.9132%	0.2251%
Retained target NROY volume	24.667%	29.744%	24.615%	99.643%	99.643%	99.449%

Table 3.3 Standard vs robust history matching with top row as the percentage of the original space as NROY and the bottom the percentage of target NROY retained.

has individual large errors near the observations, which might indicate that the emulator does not simulate the target 'NROY' space effectively. Using equation 3.4 we identified one doubt point. Following robust history matching, a local Voronoi tessellation using the local augmentation approach is applied to identify the doubt region. The local Voronoi tessellation plot for both inputs is presented in Figure 3.8. The red point is the doubt point and the pink point is selected by the augmentation step. The blue range is the Local Voronoi tessellation. We apply the robust history matching algorithm described in Section 3.4, retaining the local Voronoi tiles as part of the NROY space, and apply the usual constraint to the rest of the space.

We compare our robust method with standard history matching in Figure 3.10. Figure 3.9 shows the target NROY space as a reference. In these density plots, each pixel on any panel represents the proportion of points behind that pixel in the other 3 dimensions of the parameter space that is NROY. The scale corresponds to the colours in the upper triangles, whilst plots on the lower triangle mirror the upper triangle but with independent scales so as to reveal any structure hidden by the comparative colour scheme.

The left panel of Figure 3.10 shows the first wave and third wave NROY space following standard history matching. Comparing with the target NROY space, we can see the first wave following standard history matching has incorrectly cut out a large corner of the target region (low x_1 and low x_2, x_3, x_4 and the lower half of x_5). Wave 1 of robust history matching, shown in the bottom left panel of Figure 3.10, does not have this issue and cuts out less parameter space overall (as

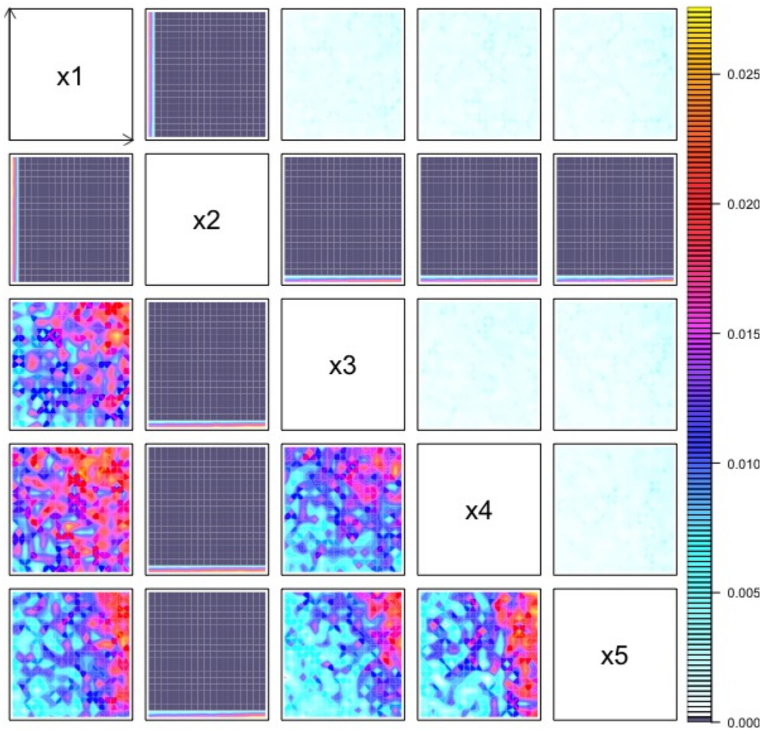


Fig. 3.9 Target NROY space.

expected). We continue to perform robust history matching for 2 further waves, for which waves 2 and 3 produced no doubt points, and are thus the same as standard history matching (but from a different wave 1). The wave 3 NROY space is shown in bottom right panel of Figure 3.10.

Table 3.3 shows the volume of NROY space as a percentage of the original space (top row) and the percentage of target NROY retained following each wave of history matching (bottom row). Target NROY is 0.09% of the original space. Though standard history matching cuts more space than our robust method in wave 1, it cuts out nearly 75% of target NROY, whilst we only cut 0.2%. After 2 further waves of history matching, we have still retained the target NROY, but have reduced our NROY to 0.17% of the original space.

This example shows a case where history matching can be non-robust in 5 dimensions and that our robust history matching effectively enables us to continue the analysis, without having to run a new wave 1. We now show a case from our work on tuning climate models where this issue has presented itself, and show how our method performs.

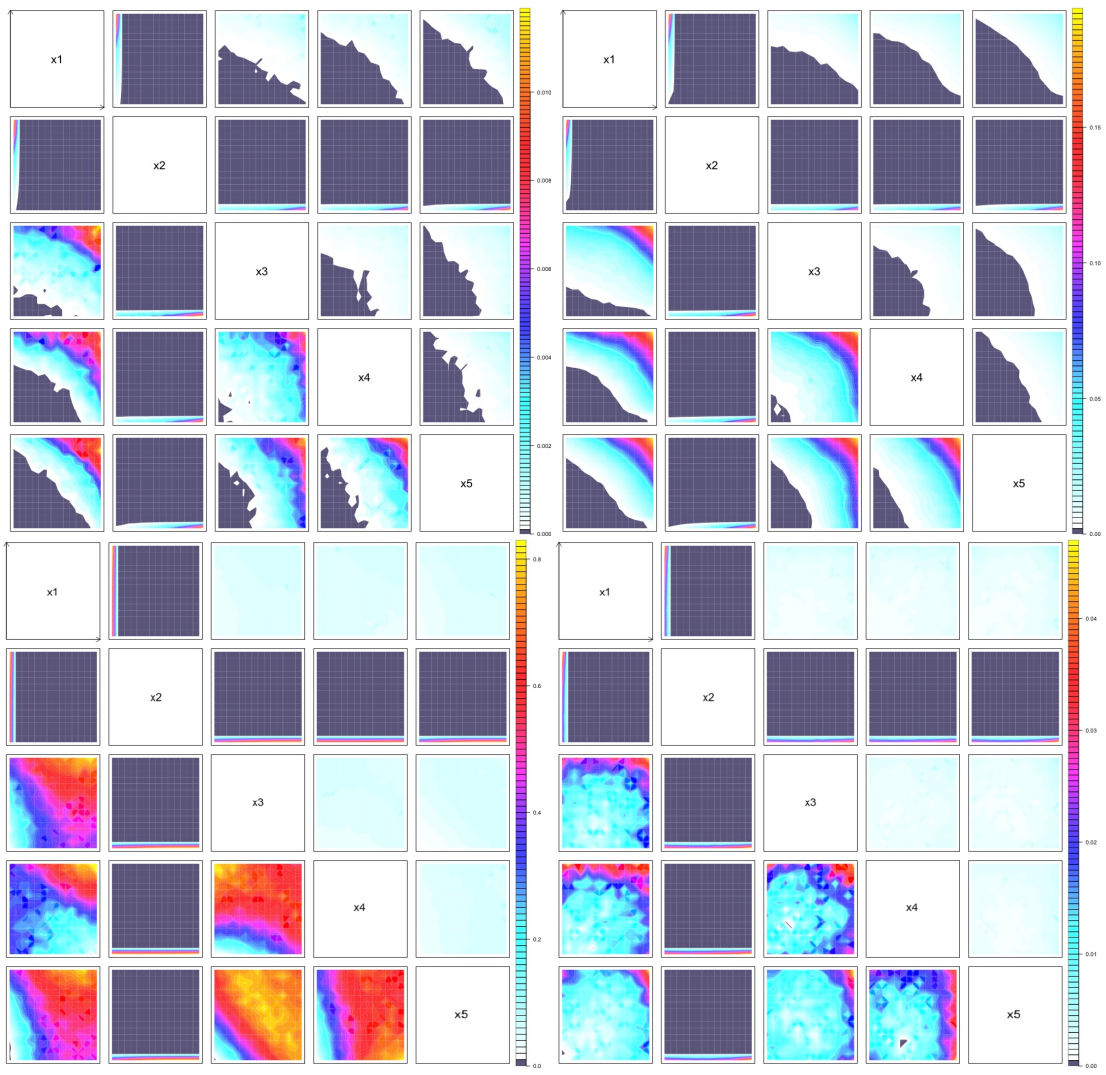


Fig. 3.10 NROY density plots for 2-D projections of NROY space. *Top left:* Wave 1 NROY space following standard history matching. *Top right:* Wave 3 NROY space following standard history matching. *Bottom left:* Wave 1 NROY space following robust history matching. *Bottom right:* Wave 3 NROY space after robust history matching. The scale corresponds to the colours in the upper triangles, whilst plots on the lower triangle mirror the upper triangle but with independent scales so as to reveal any structure hidden by the comparative colour scheme (the change from light blue, blue to red indicates that the density is rising).

3.6 Application: process-based tuning of climate models

We introduced the literature on climate model tuning in Chapter 2. Hourdin et al. (2017) state there still exist challenges when using calibration approaches. The main challenge is the computational cost of running climate simulators. For high-resolution climate models, only the calibration methods which can be used with the emulators can be considered. Also, a challenge for automatic tuning methods is that tuning based on a handful of metrics may lead to over-tuning because the improved performance in those metrics may risk achieving bad performance in metrics which are not involved in the tuning process. Process-based tuning of climate models explicitly avoids the over-tuning issue. The developers of the French climate models CNRM-CM and IPSL-CM are developing tools to automatically tune boundary layer cloud parameterisations within their models based on history matching to high resolution Large Eddy Simulations. Our collaboration involves providing methods to both emulate and history match to a large number of process-based metrics quickly and automatically, so that the modellers can use the tools independently.

With multiple unsupervised history matches, it is important that our methods are robust to possible ensemble issues, and so the method we describe in this paper should be part of our set of tools. We illustrate its importance through an example of a metric that fails our tests in IPSL-CM. IPSL-CM is an atmosphere model that is used to predict planetary atmospheres, including that of the Earth and other planets (Mars, Titan, Venus), as well as regional climate process studies (Bony and Dufresne, 2005; Hourdin et al., 2017; Voldoire et al., 2013). We run a single column version of the model (SCM) and perturb 4 cloud parameters chosen by the modellers. The model is run for a particular boundary layer case (in this case SANDU/REF, which captures transitions from cumulus to stratocumulus clouds, where stratocumulus clouds are low-level patches of clouds.) with the idea of seeing which parameter choices lead to a reasonable representations of clouds in these region types (as compared to high-resolution simulations). The parameters are `thermals fact epsilon`, `thermals ed dz`, `cld lc lsc` and `cld tau lsc`, where parameter ranges were determined by the project, and in our analysis we have mapped the parameters onto $[-1, 1]$ for fitting emulators and history matching.

We generate a 30-member design as the first 2 LHCs in a 150-member extended LHC composed of ten 15-member LHCs following (Williamson, 2015) (each additional LHC ensures that the composite design is orthogonal and fills the space in each extension phase). Leave one out diagnostic plots for our fitted emulator are presented in the top row of Figure 3.11. To history match, we use an observation of 12.18, and the observation error variance and discrepancy variance are both set as 0.0006.

Figure 3.11 shows there are 2 failed points near the observation, which might indicate that the emulator does not simulate the target NROY space well. Using equation (3.4) we identify 1 doubt point, and another doubt point is defined by our augmentation step. Since the target NROY is unknown in the climate model, in order to fairly compare our method with standard history matching, we use the remaining 120 data points (from our 150 member LHC) as validation data. The validation results are shown in Figure 3.11 in the middle and bottom rows. The first wave emulator training data are presented in black, and the rest of the data are from the validation set. The validation inputs are in green if they are retained in the NROY after wave 1 history matching, or grey otherwise. The middle panel shows the validation results following the standard history matching. In this small data set, we have 11 points in target NROY space, the standard history matching misses one target point after wave 1 (around 9% target NROY space is missed). The bottom panel shows the validation results after wave 1 following robust history matching, where the red point is the original doubt point, and the orange point is the doubt point selected by our augmentation step. The results show that our method retains all the true NROY. After a validation test with a small data set, to fairly compare two methods, we do three waves with each methodology.

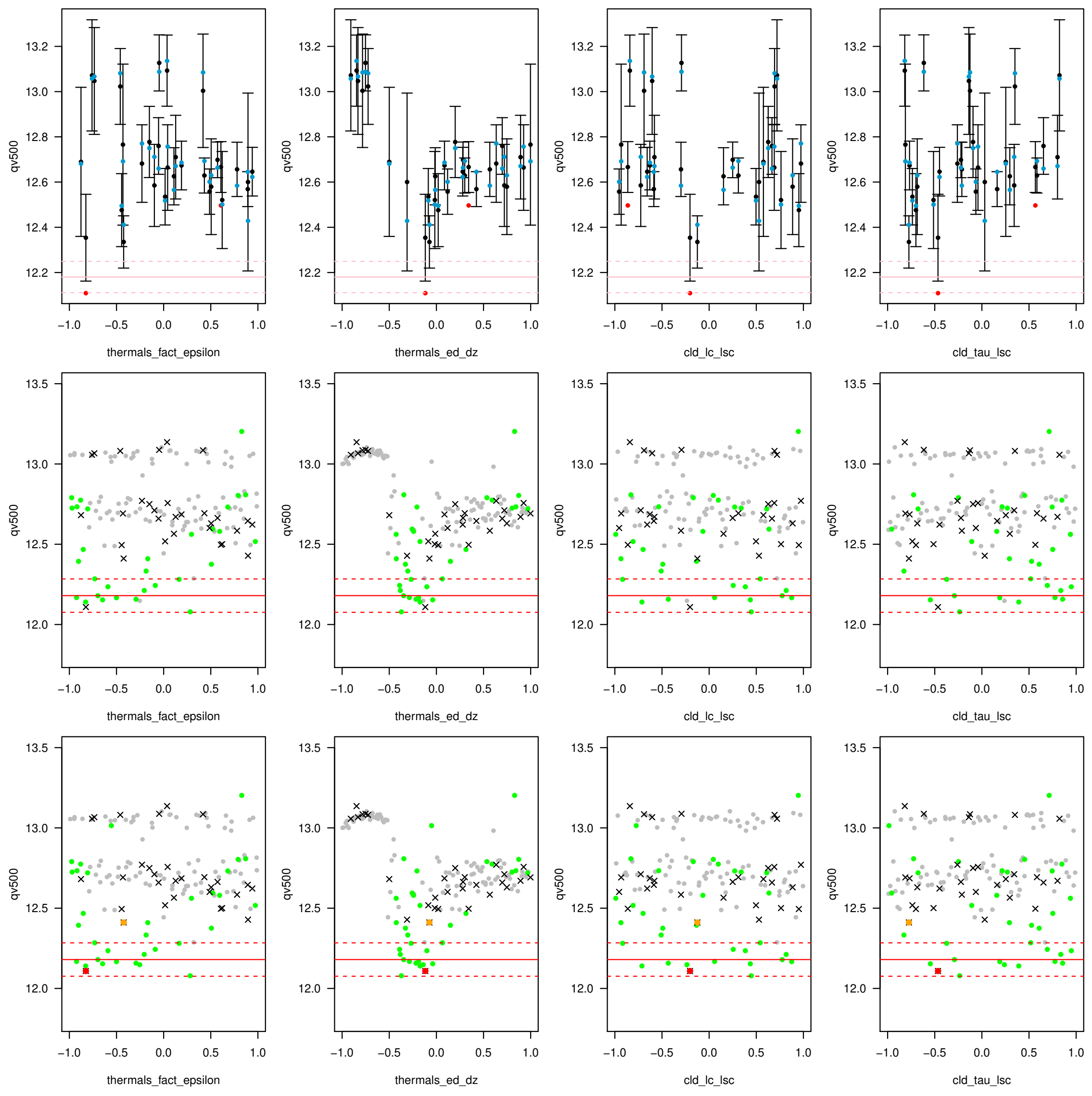


Fig. 3.11 *Top:* Leave one out diagnostic plots. Each panel represents one left-out emulator predicted, black points and error bars are from the emulator posterior mean and two standard deviation prediction intervals. The true function values are in blue if they lie within two standard deviation prediction intervals, or red otherwise. The observation with observation error are in red and dotted red line respectively. *Middle:* Validation results after wave 1 following standard history matching. All the points are from 150-member LHC sampling, emulator training data are presented in black. The remaining data are used as validation data which are in green if they are retained in the NROY after wave 1 history matching, or grey otherwise. *Bottom:* Validation results after wave 1 following robust history matching. The red point is the original doubt point and the orange point is the doubt point selected by our augmentation step.

The NROY density plots are presented in Figure 3.12, and a comparison of retained NROY volume after each wave for these two methods is presented in Table 3.4. Only the first wave was detected with doubt points, and thus standard history matching is applied to wave 2 and 3 for both approaches (when there is no

	Wave 1 retained NROY volume	Wave 2 retained NROY volume	Wave 3 retained NROY volume
Standard History matching	20.952%	17.376%	15.649%
Robust History matching	22.088%	17.63%	16.839%

Table 3.4 Comparison between standard history matching our method.

doubt point, the robust history matching is the same as standard history matching). We can see that the standard history matching cuts off more space in the third wave, which might be because of the emulator or the training data sample. As mentioned, the later waves give a more accurate emulators in the reduced space. We compare the later wave's NROY with the first wave doubt region, to detect whether the first wave doubt region is still retained in the final NROY space. By comparing the standard history matching wave 1 NROY density plot with robust wave 1's NROY density plot, we can see that our method retains more space in the first wave (around 1%). We can see from Figure 3.12 that this retained space is in the centre of the space spanned by `thermals fact epsilon` and `thermals ed dz`. The wave 3 NROY density plot of robust history matching shows the doubt area is still in the NROY space, showing that standard history matching incorrectly ruled out part of target NROY parameter space.

This application showed that incorrectly ruling out parameter space can occur in practice, in this case when history matching to tune climate models. For climate models in particular, this mistake could prove very costly as history matching is used to assess the quality of a given parametersiation or an alternative. If good models are accidentally discarded, the parameterisation or even the resolution of the model or its implementation might be needlessly changed, wasting the time and/or resources of the modelling centre.

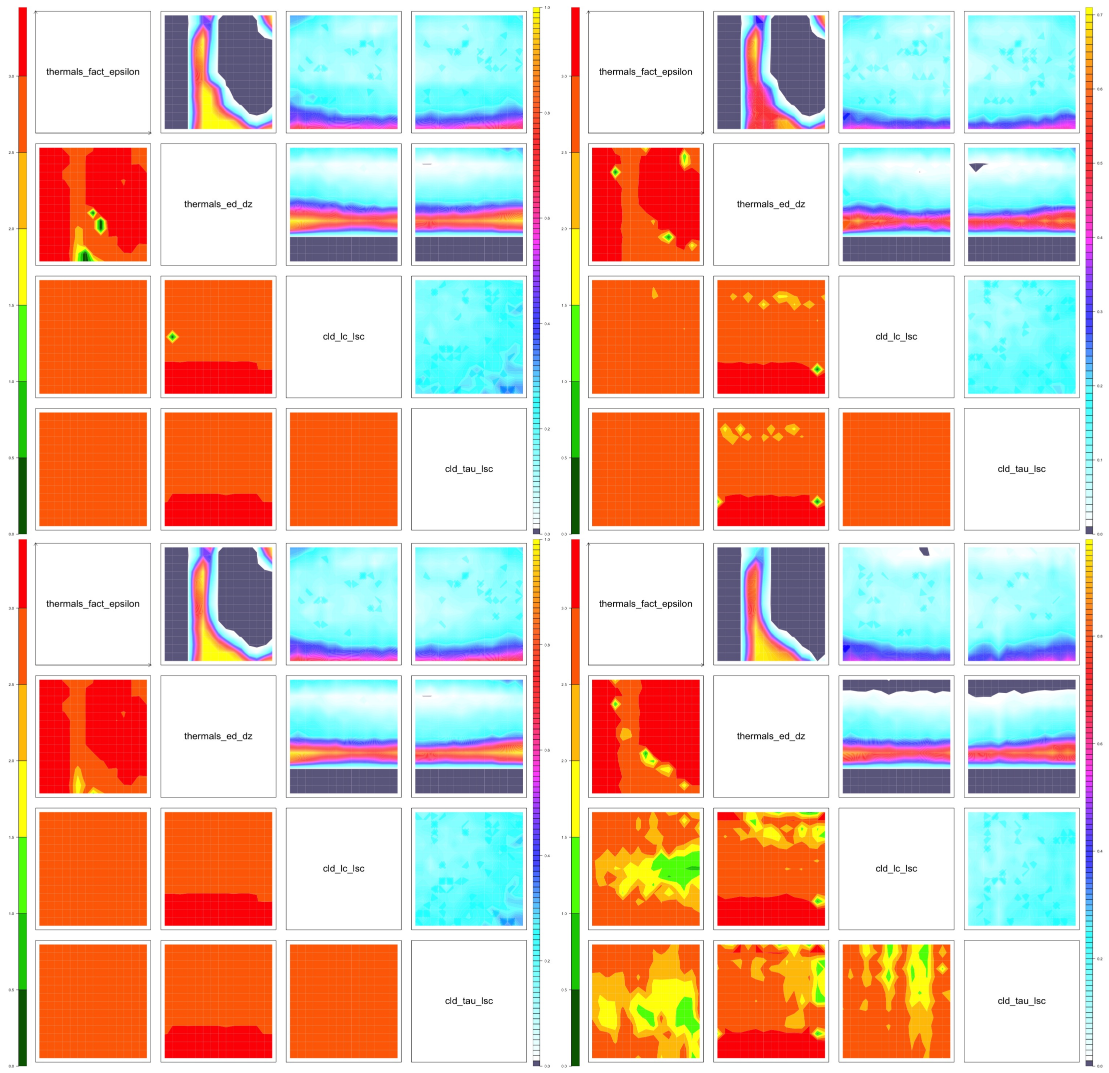


Fig. 3.12 *Top left*: Wave 1 NROY space for LMDZ-SANDU after robust history matching. *Top right*: Wave 3 NROY space for LMDZ-SANDU after robust history matching. *Bottom left*: Wave 1 NROY space for LMDZ-SANDU following standard history matching. *Bottom right*: Wave 3 NROY space for LMDZ-SANDU following standard history matching.

3.7 Discussion

In this chapter we demonstrated a potential issue with history matching that occurs when emulators that seem to validate well by most standard analyses do not simulate the (unknown) target subspace well enough. We showed that this can lead to good parts of parameter space being ruled out unintentionally. We developed a method for detecting these cases based on standard diagnostics. We then presented a robust history matching method based on using a tailored local Voronoi tessellation designed to capture the region where the emulator is not as good as it needs to be, and isolate it so that the rest of the input space can be pruned as normal, without having to re-run the simulator.

We demonstrated the efficacy of our method in comparison to standard history matching for 2 numerical examples designed to demonstrate the issue, and then applied the method to a process metric from a single column version of the French climate model. We showed that, unlike standard history matching, our method manages to effectively cut parameter space whilst ensuring that the target space is preserved. There are several possible extensions to our developed approaches. Our detect step based on standard diagnostic can easily identify the doubt region when the emulation failed runs lie in the true NROY space. However, where no model runs in the true NROY space, there may still be inaccuracies in history matching if the failure points are close to NROY space. More emulation diagnostic approaches could be considered for these possible situations in the future.

Whilst it may be possible to observe the diagnostic issue we have highlighted, and to offer a bespoke history match for a particular quantity in any given application, this is unlikely to be feasible in applications where tens, hundreds or even thousands of emulators are built, and are to be compared with observations (see, e.g. Gu et al., 2016; Lee et al., 2012). We also want methods that do not require frequent intervention by an experienced statistician. Hence our robust method provides a way to safely and automatically isolate any regions of parameter space where it would be dangerous to history match with the current emulator, but still allow the same emulator to be used appropriately without requiring a bespoke analysis. To achieve this aim, we will continue study history matching for different cases in the following chapters, especially for computer model with high-dimensional outputs. The climate model will also be studied further in Chapter 6.

Chapter 4

Kernel-based history matching for high-dimensional computer model output

4.1 Introduction

There are a number of different approaches for emulation and calibration of computer models with high-dimensional output (e.g. a time series, a spatial field, or a spatio-temporal field), as discussed in Section 2.3.5 and Section 2.5.3. The most commonly used method to handle high-dimensional output is principal component analysis (PCA), which projects the computer model output onto a low dimension basis, derived from an ensemble of computer runs (Higdon et al., 2008; Wilkinson, 2010). PCA reduces the dimensionality of the model output significantly, such that a straight-forward and efficient emulation can be built for the coefficients, either univariately or multivariately.

Both history matching and calibration can be performed in conjunction with the PCA basis projection method for emulation (see Section 2.4 and Section 2.5.3 for more details). Section 3.1 provides a short summary of the general advantages of history matching. Particularly for climate models, a prior distribution for the discrepancy error is hard to estimate. Bayesian calibration, introduced by Kennedy and O'Hagan (2001) specifically requires discrepancy information. Furthermore, due to the complexity of the physical processes being studied, climate models often require vast amounts of computing even for a single run, meaning that only a small number of runs can be evaluated. Using Bayesian calibration to optimise a vast state vector concerning only a relatively small amount of data could easily lead to overfitting/overtuning (Hourdin et al., 2017). Hence, we will only focus on history matching for computer models with high-dimensional output in this chapter.

For emulators built using basis coefficients, history matching can be performed either on these coefficients, or on reconstructions of the original field. Salter et al. (2019) investigate the properties of the PCA basis method for history matching: they present a 'terminal case' to show there is a chance that the basis vectors fail to represent the key features in the observations in which the calibration is interested. To solve that limitation, they offer a rotation algorithm to find a calibration-optimal PCA basis. Moreover, Salter and Williamson (2019) present an efficient way to compute calibration for high-dimensional computer model output via history-matching.

However, when we try to match spatial output, even the rotated PCA-based history matching might fail when the output space is not well approximated by a linear subspace. To explain the reasons why the existing approaches can fail, consider the following analogy. Imagine a room with spotlights which form patterns (like the basis vectors), and each of them has a dimmer which is responsible for the strength of various patterns (similar to the coefficient controlled by the input parameters). The rotated PCA-based history matching can find the right way to set the dimmers to make the lights stronger or weaker, which gives certain lighting conditions consistent with the key pattern in the observation. But when the locations of the spot lights are unrestricted in the 2-D space of the output, the positions of key features in the output space can be different for each run, and can be misaligned compared to observations. When we try to calibrate such spatial output, what is important for the credibility of the model (in our application) is that the key physical patterns are present, even if they may not be in the right place or exact. The current state of the art statistical methods for calibrating these models cannot handle these problems well, and are typically only good at finding stronger or weaker signals in fixed locations.

This kind of situation can happen for simulators with a large spatial output where the model input parameters control the position of the pattern. Such situations are prevalent in climate models; resolution-dependent currents or signals may move around the output space. When we try to calibrate these computer models, sometimes the goal is that the model has to be able to represent key features that the modellers want their model to replicate, even if they may not happen in exactly the same place in reality.

Kernel methods for pattern recognition and feature extraction from machine learning could hold the key to overcoming the limitations of current calibration methods. We investigate the application of kernel methods to history matching. A novel method is introduced as kernel-based history matching. We will show that kernel-based history matching is not only an extension of current calibration approaches, but a generalised version of the traditional history matching using PCA. Through our investigating, we will generalise the notion of the method and the meaning of key elements, such as discrepancy, implausibility and NROY space as needed.

In this chapter, we first adopt kernel methods for model outputs which can be represented by a spatial field to highlight the key features that the modellers want to calibrate. Essentially, there is a mapping function that maps our model output from the simulator output space to another space (a higher-dimensional "feature space"), in which the simulator exhibits a more linear behaviour. There is no requirement that we know the explicit expression of the mapping function. Kernel methods allow us to get rid of the need to compute the coordinates of the mapped model output in feature space: the same procedures are achieved with a kernel function which computes dot products between the data in feature space.

With a user specified kernel function, we position the calibration problem in the corresponding feature space. We perform history matching as defined within the same framework introduced before, but in feature space. This method is based on the distance (implausibility) between the simulator outputs and observation in feature space, with respect to observation error and discrepancy (though the meaning of the latter will be revisited), and we use this distance to rule out regions of input space that give model outputs that are 'too far' from the observation.

To avoid building thousands of emulators for high-dimensional outputs, linear PCA is performed on the feature space. This procedure is known as kernel principal component analysis (kernel PCA), which is a non-linear extension of PCA using kernel methods to process nonlinear data sets. Kernel PCA can extract the nonlinear characteristics of data, and obtain the best description of data, while keeping the data variance unchanged. By projecting the output onto a low dimensional basis space (a sub-space of feature space), we only need a few emulators similar to other basis emulation methods introduced in Chapter 2. Kernel-based history matching will be performed on the feature space with these emulators, giving both efficiency and effectiveness. We will introduce kernel-based history matching over two chapters: this chapter gives the background of kernel methods and introduces kernel-based history matching. Getting the calibration right may require tailoring our kernels to physical knowledge, so an algorithm for optimal kernel selection in kernel-based history matching will be developed in Chapter 5.

This chapter has the following structure. In Section 4.2 we introduce the background of kernel methods and kernel PCA. The multivariate emulation approach with kernel PCA is also introduced. We present our proposed kernel-based history matching on the feature space in Section 4.3, and demonstrate three potential approaches with newly defined implausibility and cut-off threshold in Section 4.4, 4.5, 4.6, and 4.7 . In Section 4.8, we apply our method to a numerical example. We finish off with a discussion in Section 4.9.

4.2 Kernel methods

Kernel methods are a category of approaches for pattern analysis which have become popular in machine learning over the past 2 decades (Joachims, 2002; Schölkopf et al., 2002; Shawe-Taylor et al., 2004). For many linear algorithms in pattern analysis which can only efficiently detect linear relationships within data (e.g. PCA), kernel methods are introduced to extend linear hypotheses to nonlinear relationship by embedding the original data set in a feature space, $\mathcal{F}$, in which algorithms based on linear algebra can be applied to identify patterns in embedded data (Burges, 1998; Soentpiet et al., 1999).

The transformation from original space to a feature space is a user-specified mapping function $\phi(\cdot)$, which maps the data from an initial data space, $\mathcal{M}$, into feature space $\mathcal{F}$, $\phi : x \longrightarrow \phi(x)$. However, a feature space usually has higher dimensionality than the original space, and could even be the infinite-dimensional space, which means the mapping function and it's inverse function are difficult to find, or even impossible. In contrast, kernel methods require only a user-specified kernel, without needing to compute the coordinates of the data in feature space, but rather by only needing to compute the inner product between two vectors in the feature space (Bishop, 2006; Genton, 2001; Hofmann et al., 2008; Scholkopf and Smola, 2001). For all x and x' from the initial space, $\mathcal{M}$, of dimension m, a kernel function $k(x,x')$ defines an inner product of $\phi(x)$ and $\phi(x')$ in the respective feature space $\mathcal{F}$ of dimension D, $\mathcal{F} \subset \mathbb{R}^D$

$$k(x,x') = <\phi(x),\ \phi(x')>, \tag{4.1}$$

where the feature space $\mathcal{F}$ is a dot product space and features are finite dimensional vectors (Hofmann et al., 2008), so that the inner product can be written as a dot product:

$$<\phi(x),\phi(x')> = \ \phi(x)^T\phi(x').$$

Instead of explicitly computing the mapped data $\phi(x)$ in the higher dimensional feature space, a kernel method allows us to represent the data $\phi(x)$ and $\phi(x')$ only through a set of pairwise comparisons between the original data x and x'. This approach is called the kernel trick (Lanckriet et al., 2004; Shawe-Taylor et al., 2004). Using the kernel trick is often computationally cheaper than computing an explicit mapping function. This approach works because the algorithms can be implemented such that they only require dot products between embedded data. There are many algorithms capable of operating with kernels, such as SVM, PCA and ridge regression. We illustrate the kernel trick thought our account of Kernel PCA in Section 4.2.2.

4.2.1 Kernels

Theoretically, $k(x,x')$ is required be positive-definite (Gehler, 2009; Mercer, 1909). To give the definition of a positive-definite kernel, we denote $x_1,\ldots x_n$ as n input data from initial data space $\mathcal{M}$. Given a kernel function $k(x,x')$, a $n \times n$ kernel matrix $\mathbf{K} = [K_{ij}]$ can be calculated, where $K_{ij} = k(x_i,x_j)$. The semi-definite matrix, $\mathbf{K}$, is positive definite when

$$\sum_{i,j=1}^{n} c_i c_j K_{ij} \geq 0 \tag{4.2}$$

is true for any nonzero constants $c_1, \ldots, c_n \in \mathbb{R}$ of the same sign. When the kernel matrix is positive definite for any given input, then its associated kernel function is a positive definite kernel. Aronszajn (1950) presents the Moore-Aronszajn theorem: for a positive definite kernel: there is a Hilbert space $\mathcal{H}$ and an implicitly defined map function ϕ such that $\phi : \mathcal{X} \to \mathcal{H}$ and $k(x,x^{'}) =< \phi(x), \phi(x^{'}) >$. Hence, as long as the feature space $\mathcal{F}$ is an inner product space, an explicit representation for ϕ is not required.

In our research, we mainly study kernel methods that work with PCA. Hofmann et al. (2008); Hsu et al. (2003); Schölkopf et al. (2002); Souza (2010) present different types of positive definite kernel functions for kernel PCA (we will introduce the kernel PCA algorithm in Section 4.2.2). The most commonly used kernel functions are introduced below.

1. Linear kernels are the most simple kernel functions with

$$k(x,x') = xx' + c, \tag{4.3}$$

for constant c. When $c = 0$, $k(x,x')$ is called the homogeneous linear kernel. The linear kernel function needs fewer control parameters and calculation is fast, making their use computationally attractive.

2. Polynomial kernels are popular in image processing (Meijering et al., 1999). The most commonly used form for a polynomial kernel is

$$k(x,x') = (\alpha x^T x' + c)^d, \tag{4.4}$$

where c is a constant, α is an adjustable slope parameter and d is a positive integer referring to the polynomial degree.

3. Gaussian kernels are general-purpose kernels, which have been popular in machine learning (Souza, 2010). The Gaussian kernel is written

$$k(x,x') = \exp\left(-\frac{\| x - x' \|^2}{2\sigma^2}\right), \tag{4.5}$$

where $\| x - x' \|^2$ is the L_2 norm, σ is an adjustable parameter known as the length scale. The Gaussian kernel (squared exponential correlation function) is also a default kernel for Gaussian process emulators, which were introduced in Section 2.3.2. The Gaussian kernel is universal, and it is infinitely differentiable. It can project the mapped data $\phi(x)$ in to a infinite dimensional feature space. Gaussian kernels are examples of radial basis function (RBF) kernels, i.e. a kernel that depends only on the distance between the two arguments, $\| x - x' \|$.

4. The (absolute) exponential kernel is also an RBF kernel, similar to the Gaussian kernel, the expression is

$$k(x,x') = \exp\left(-\frac{\| x-x' \|}{2\sigma^2}\right), \tag{4.6}$$

where $\| x-x' \| = \sqrt{\| x-x' \|^2}$, and σ is the length scale hyper-parameter. Unlike the squared exponential function, the exponential kernel is only continuous, and it is not differentiable. The exponential kernel can be a good choice for fitting non-differentiable functions.

5. Laplacian kernels are similar to the exponential kernel (it is also a RBF kernel). The Laplacian kernel has the form

$$k(x,x') = \exp\left(-\frac{\| x-x' \|}{\sigma}\right). \tag{4.7}$$

For multidimensional input, by default the same length scale is adopted for each input dimension if we use the kernels introduced above. Automatic Relevance Determination (ARD) kernels are introduced for multivariate input data in Van Gestel et al. (2001); Wang et al. (2010). Most basic kernel functions have an ARD extended version. A general form for linear kernels can be found by adding a positive definite matrix, Σ, onto the input components

$$k(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\Sigma^{-1}\mathbf{x}' + c. \tag{4.8}$$

For Gaussian kernels, the general form is given by

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-(\mathbf{x}-\mathbf{x}')^T\Sigma^{-1}(\mathbf{x}-\mathbf{x}')\right). \tag{4.9}$$

When Σ is a diagonal matrix, $k(\mathbf{x},\mathbf{x}')$ can be written as

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\sum_{i=1}^{n}\frac{(\mathbf{x}_i-\mathbf{x}'_i)^2}{\sigma_i^2}\right), \tag{4.10}$$

where σ_i is the length scale of dimension i. When σ_i is infinity, the corresponding dimension is ignored. If every dimension is ignored then this kernel would be a constant. If there is a dimension that contains important information, then we could find the optimal corresponding length scale, σ_i, to scale that dimension appropriately.

Which kernel to use and how to choose or fit its properties is an important question and depends on the specific problem at hand. We will discuss the kernel (and kernel parameter) selection problem in the next chapter.

4.2.2 Kernel principal component analysis for emulation

Kernel principal component analysis (kernel PCA) is a nonlinear extension of PCA using kernel methods, which can extract the nonlinear characteristics of data. As our goal is to generalise PCA approaches to emulation and calibration, we introduce kernel PCA over the outputs of a computer experiment, analogous to Section 2.3.5.

Let f represent the computer model and $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) \in \mathcal{M} \subseteq \mathbb{R}^m$ represent a set of simulator runs with inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X} \subseteq \mathbb{R}^p$, where $\mathcal{X}$ is the p-dimensional simulator input space and $\mathcal{M}$ is the m-dimensional simulator output space. As we introduced in Section 4.2, kernel PCA maps the m-dimensional output data into a higher D-dimensional feature space $\mathcal{F} \subset \mathbb{R}^D$ via a mapping function $\phi(\cdot) : x \to \phi(x)$ which is defined by a kernel function

$$k(f(\mathbf{x}'), f(\mathbf{x})) = \phi(f(\mathbf{x}'))^T \phi(f(\mathbf{x})). \tag{4.11}$$

PCA is then applied to the feature space $\mathcal{F}$, via the $D \times n$ matrix representing the mapped ensemble, $\Phi = (\phi(f(\mathbf{x}_1)), \ldots, \phi(f(\mathbf{x}_n)))$. The majority of variability in the training data (on feature space) can be explained by the first few orthogonal directions. Define the mean of the mapped ensemble $\bar{\phi} = (\bar{\phi}_1, \ldots, \bar{\phi}_D)$, where

$$\bar{\phi}_j = \frac{1}{n}\sum_{i=1}^{n} \phi_j(f(\mathbf{x}_i)),$$

and define the centred mapped ensemble $\tilde{\mathbf{\Phi}} = (\tilde{\phi}(f(\mathbf{x}_1)), \ldots, \tilde{\phi}(f(\mathbf{x}_n)))$, where $\tilde{\phi}(f(\mathbf{x}_i)) = \phi(f(\mathbf{x}_i)) - \bar{\phi}$. Note, throughout, we will not have access to $\phi(f(\mathbf{x}))$ or $\bar{\phi}$, but can access terms involving $\phi(f(\mathbf{x}))^T\phi(f(\mathbf{x}))$ using the kernel. Analogous to PCA, we find the eigen-decomposition of the $D \times D$ covariance matrix,

$$G = \frac{1}{n}\sum_{i=1}^{n} \tilde{\phi}(f(\mathbf{x}_i))\tilde{\phi}(f(\mathbf{x}_i))^T. \tag{4.12}$$

We have to find eigenvalues $\lambda > 0$ and eigenvectors $\mathbf{W} \in \mathcal{F}$ satisfying

$$G\mathbf{W} = \lambda\mathbf{W}, \tag{4.13}$$

where λ is a $D \times D$ diagonal matrix, and $\mathbf{W}$ is a $D \times n$ matrix of eigenvectors, $\mathbf{W} = (W_1, \ldots, W_n)$, and each W_i has length D. Note that, as with PCA, for the kth eigenvector W_k, $W_k \in \text{span}(\tilde{\phi}(f(\mathbf{x}_1)), \ldots, \tilde{\phi}(f(\mathbf{x}_n)))$. There is a $n \times n$ matrix of coefficients, A, with the k-th column vector of A, $\alpha_k = (\alpha_{k1}, \ldots, \alpha_{kn})^T$, such that

$$W_k = \sum_{i=1}^{n} \alpha_{ki}\tilde{\phi}(f(\mathbf{x}_i)). \tag{4.14}$$

However, directly solving for the eigenvectors and eigenvalues of G in equation (4.13) may not be feasible (ϕ, $\bar{\phi}$ and $\tilde{\mathbf{\Phi}}$ are typically unknown). The eigen-decomposition can be performed instead using the kernel trick (Hoffmann, 2007; Ma and Zabaras, 2011; Schölkopf et al., 1997). The eigen-decomposition of equation (4.13) is equivalent to the eigen-decomposition of the centred kernel matrix $\tilde{\mathbf{K}}$ (the full proof is given by Schölkopf et al. (1998b)),

$$\lambda A = \frac{1}{n}\tilde{\mathbf{K}}A, \tag{4.15}$$

where $\tilde{\mathbf{K}}$ is the centred kernel matrix defined via

$$\begin{aligned}
\tilde{K}_{ij} &= \tilde{\phi}(f(\mathbf{x}_i))^T \tilde{\phi}(f(\mathbf{x}_j)) \\
&= (\phi(f(\mathbf{x}_i)) - \bar{\phi})^T (\phi(f(\mathbf{x}_j)) - \bar{\phi}) \\
&= \phi(f(\mathbf{x}_i))^T \phi(f(\mathbf{x}_j)) - \frac{1}{n}\sum_{l=1}^{n} \phi(f(\mathbf{x}_i))^T \phi(f(\mathbf{x}_l)) - \frac{1}{n}\sum_{k=1}^{n} \phi(f(\mathbf{x}_k))^T \phi(f(\mathbf{x}_j)) \\
&+ \frac{1}{n^2}\sum_{l=1}^{n}\sum_{k=1}^{n} \phi(f(\mathbf{x}_k))^T \phi(f(\mathbf{x}_l)) \\
&= K_{ij} - \frac{1}{n}\sum_{l=1}^{n} K_{il} - \frac{1}{n}\sum_{l=1}^{n} K_{kj} + \frac{1}{n^2}\sum_{l=1}^{n}\sum_{k=1}^{n} K_{kl},
\end{aligned} \tag{4.16}$$

where $\tilde{K}_{ij}$ is the ij-th element in the centred kernel matrix, $\tilde{\mathbf{K}}$. By defining a centre matrix $\mathbf{H} = \mathbf{I} - \mathbf{1}_n$, where $\mathbf{I}$ is the $n \times n$ identity matrix, n is the number of data points. and $\mathbf{1}_n$ is a $n \times n$ matrix with all elements equal to $\frac{1}{n}$, the centred kernel matrix can be expressed as

$$\tilde{\mathbf{K}} = \mathbf{HKH}. \tag{4.17}$$

Equation (4.15) shows that α_k is an eigenvector of $\tilde{\mathbf{K}}$, such that the eigenvectors, W, of covariance matrix G can be represented by the eigenvectors of $\tilde{\mathbf{K}}$. Also, given the eigenvalues of $\tilde{\mathbf{K}}$, $\tilde{\lambda}$, the eigenvalues λ of covariance matrix G are determined by $\tilde{\lambda} = n\lambda$. By requiring that the corresponding eigenvectors of G be normalised, i.e. $1 = W_k^T W_k$, the eigenvectors of the centred kernel matrix can be normalised:

$$\begin{aligned}
1 &= \sum_{i=1,j=1}^{n} \alpha_{ki}\alpha_{kj}\tilde{\phi}(f(\mathbf{x}_i))^T \tilde{\phi}(f(\mathbf{x}_j)) \\
&= \sum_{i=1,j=1}^{n} \alpha_{ki}\alpha_{kj} k(f(\mathbf{x}_i), f(\mathbf{x}_j)) \\
&= \alpha_k^T \tilde{\mathbf{K}} \alpha_k \\
&= \tilde{\lambda}_k \alpha_k^T \alpha_k .
\end{aligned} \tag{4.18}$$

Let the normalised eigenvector matrix, A, be $\tilde{A}$. For the ki-th entry of $\tilde{A}$, we have $\tilde{\alpha}_{ki} = \frac{\alpha_{ki}}{\sqrt{\tilde{\lambda}_{ki}}}$. Note that, we can compute α_k and $\tilde{\mathbf{K}}$. Therefore, the k-th eigenvector

W_k becomes

$$W_k = \sum_{i=1}^{n} \tilde{\alpha}_{ki}\tilde{\phi}(f(\mathbf{x}_i)) = \tilde{\alpha}_k\tilde{\mathbf{\Phi}}. \tag{4.19}$$

Although we cannot calculate the mapped data $\tilde{\mathbf{\Phi}}$ without the explicit mapping function (which is generally unavailable), the projection of the mapped data $\tilde{\phi}(f(\mathbf{x}))$ can be calculated directly via the kernel trick. We denote the projection of the mapped data $\tilde{\phi}(f(\mathbf{x}))$ onto the eigenvector W_k as $C_k(\mathbf{x})$, where

$$\begin{aligned} C_k(\mathbf{x}) &= W_k^T\tilde{\phi}(f(\mathbf{x})) \\ &= \sum_{j=1}^{n}\tilde{\alpha}_{kj}\tilde{\phi}(f(\mathbf{x}_j))^T\tilde{\phi}(f(\mathbf{x})) \\ &= \sum_{j=1}^{n}\tilde{\alpha}_{kj}\tilde{k}(f(\mathbf{x}),\ f(\mathbf{x}_j)). \end{aligned} \tag{4.20}$$

As with PCA, we can truncate to the first few eigenvectors, giving $\phi_r(f(\mathbf{x}))$ as

$$\phi_r(f(\mathbf{x})) = \sum_{k=1}^{r} W_k C_k(\mathbf{x}) + \bar{\phi}, \tag{4.21}$$

and

$$\phi(f(\mathbf{x})) = \phi_r(f(\mathbf{x})) + \epsilon, \tag{4.22}$$

where ϵ is a residual vector of length D which is orthogonal to each $W_k, k = 1, \ldots, n$. Equation (4.22) can be written in matrix form as:

$$\phi(f(\mathbf{x})) = \mathbf{W}^r\mathbf{C}_r(\mathbf{x}) + \bar{\phi} + \varepsilon, \tag{4.23}$$

where $\mathbf{C}_r(\mathbf{x})$ is the projections of $\tilde{\phi}(f(\mathbf{x}))$ onto the first r eigenvectors, $\mathbf{C}_r(\mathbf{x}) = [C_1(\mathbf{x}),\ \ldots,\ C_r(\mathbf{x})]^T$ and $\mathbf{W}^r = [W_1, \ldots, W_r]$. $\mathbf{C}_r(\mathbf{x})$ can be written as:

$$\mathbf{C}_r(\mathbf{x}) = (\mathbf{W}^{rT}\mathbf{W}^r)^{-1}\mathbf{W}^{rT}\left(\phi(f(\mathbf{x})) - \bar{\phi}\right) = \mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x})), \tag{4.24}$$

where $\mathbf{W}^{rT}\mathbf{W}^r = \mathbf{I}$.

Note that a kernel PCA with a linear kernel,

$$k(f(\mathbf{x}'), f(\mathbf{x})) = f(\mathbf{x}')^T f(\mathbf{x}),$$

is exactly equivalent to standard PCA. This is because the kernel matrix $\mathbf{K}$ with entries $K_{ij} = k(f(\mathbf{x}_i), f(\mathbf{x}_j)) = f(\mathbf{x}_i)^T f(\mathbf{x}_j)$ is equal to the standard Gram matrix $G_{ij} = f(\mathbf{x}_i)^T f(\mathbf{x}_j)$. Hence the principal components will not change.

4.2.3 Gaussian process emulators: Basis method with kernel PCA

Computing $C_k(\mathbf{x}_i)$, $k = 1, \ldots, r$ for each ensemble member $f(\mathbf{x}_i)$, transforms $\mathbf{F}$ $(m \times n)$ into an $r \times n$ matrix, $\mathbf{C} = (\mathbf{C}_r(\mathbf{x}_1), \ldots, \mathbf{C}_r(\mathbf{x}_n))$, where $\mathbf{C}_r(\mathbf{x}_i) = [C_1(\mathbf{x}_i), \ldots, C_r(\mathbf{x}_i)]^T$. By approximating r coefficients for each input, rather than m initial simulator outputs, the dimensionality of the simulator outputs is reduced. Either univariate or multivariate Gaussian process emulators could be built for the coefficients (Xing et al., 2016). Before building emulators, the number of components, r, needs to be specified. The common approach selects r by requiring that the majority of the variance in the ensemble is explained by projection onto $\mathbf{W}^r$. Denote the total proportion of ensemble variability to be explained by the first r basis vectors as $V(\mathbf{W}^r) = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^n \lambda_i}$. We require that

$$V(\mathbf{W}^r) > T_v,$$

where T_V is set by the user and is the proportion of ensemble variability we want to be explained (often $T_v = 0.95$), and setting T_v too high can lead to a large r being selected. However, the later basis vectors explain low percentages of the variability in the ensemble, making accurate emulation for later coefficients difficult. Higdon et al. (2008) show that, for PCA, Gaussian process emulation works well for the first few components, but not so for the later components. They suggest that one does not take take more than 5 basis vectors in practice, we follow this suggestion for kernel PCA.

By constructing univariate Gaussian process emulators for the coefficients $C_k(\mathbf{x})$ for each basis vector separately,

$$C_k(\mathbf{x}) \sim \mathcal{GP}(m_k(\mathbf{x}),\ \sigma_k^2 c_k(\mathbf{x},\mathbf{x})), \quad k = 1, \ldots, r, \tag{4.25}$$

with the emulator expectation for each of the r basis vectors given by

$$\mathrm{E}[\mathbf{C}_r(\mathbf{x})] = [\mathrm{E}[C_1(\mathbf{x})], \ldots, \mathrm{E}[C_r(\mathbf{x})]]^T,$$

and the associated emulator variance matrix:

$$\mathrm{Var}[\mathbf{C}_r(\mathbf{x})] = \mathrm{diag}[\mathrm{Var}[C_1(\mathbf{x})], \ldots, \mathrm{Var}[C_r(\mathbf{x})]].$$

The expressions for $\mathrm{E}[\phi(f(\mathbf{x}))]$ and $\mathrm{Var}[\phi(f(\mathbf{x}))]$ can be extracted for history matching, which can be written in terms of the coefficient emulators,

$$\mathrm{E}[\phi(f(\mathbf{x}))] = \mathbf{W}^r \mathrm{E}[\mathbf{C}_r(\mathbf{x})] + \bar{\phi}, \tag{4.26}$$

$$\mathrm{Var}[\phi(f(\mathbf{x}))] = \mathbf{W}^r \mathrm{Var}[\mathbf{C}_r(\mathbf{x})]\, \mathbf{W}^{rT}. \tag{4.27}$$

Note that $\mathbf{W}^r$ and $\bar{\phi}$ are unavailable directly, but we will show that neither the explicit form of $\mathbf{W}^r$ or $\bar{\phi}$ are required in history matching.

4.2.4 Observation in feature space

Consider observations, z, of the modelled process that we intend to use for history matching. Denote $\phi(z)$ as the mapped observation in feature space, and the projection of the mapped observation, $\phi(z)$, onto $\mathbf{W}$ as $\mathbf{C}(z)$, where $\mathbf{C}(z)$ is a n vector, $\mathbf{C}(z) = [C_1(z), \ldots, C_n(z)]^T$. Note that $\phi(z)$ is unknown with no explicit representation for the mapping function, but its projection, $\mathbf{C}(z)$, can be obtained via the kernel trick. To calculate $\mathbf{C}(z)$, the mapped ensemble mean will first be removed from the $\phi(z)$, with the centred mapped observation is defined as

$$\tilde{\phi}(z) = \phi(z) - \bar{\phi}.$$

The k-th projection $C_k(z)$ is given by:

$$\begin{aligned}
C_k(z) &= W_k^T \tilde{\phi}(z) \\
&= \sum_{i=1}^{n} \tilde{\alpha}_{ki} \tilde{\phi}(f(\mathbf{x}_i))^T [\phi(z) - \bar{\phi}] \\
&= \sum_{i=1}^{n} \tilde{\alpha}_{ki} [\phi(f(\mathbf{x}_i)) - \bar{\phi}]^T [\phi(z) - \bar{\phi}] \\
&= \sum_{i=1}^{n} \tilde{\alpha}_{ki} [\phi(f(\mathbf{x}_i))^T \phi(z) - \bar{\phi}^T \phi(z) - \phi(f(\mathbf{x}_i))^T \bar{\phi} + \bar{\phi}^T \bar{\phi}] \\
&= \tilde{\alpha}_k^T \mathbf{K}_z - \frac{1}{n} \mathbf{1}^T \mathbf{K}_z (\tilde{\alpha}_k^T \mathbf{1}) - \frac{1}{n} \tilde{\alpha}_k^T (\mathbf{K}\mathbf{1}) + \frac{1}{n^2} \mathbf{1}^T \mathbf{K}\mathbf{1} (\tilde{\alpha}_k^T \mathbf{1}).
\end{aligned} \tag{4.28}$$

Here $\mathbf{K}_z = [k(z, f(\mathbf{x}_1)), k(z, f(\mathbf{x}_2)),\ \ldots,\ k(z, f(\mathbf{x}_n)))]$, and $\mathbf{1} = [1, \ldots, 1]^T$ is an $n \times 1$ vector. For computation, equation (4.28) can be written as:

$$C_k(z) = W_k^T \tilde{\phi}(z) = \tilde{\alpha}_k^T \tilde{\mathbf{K}}_z, \tag{4.29}$$

where $\tilde{\mathbf{K}}_z = [\tilde{k}(z, f(x_1)), \tilde{k}(z, f(x_2)),\ \ldots,\ \tilde{k}(z, f(x_n))]$. Each element $\tilde{k}(z, f(\mathbf{x}))$ can be calculated as:

$$\begin{aligned}
\tilde{k}(z, f(\mathbf{x})) &= \tilde{\phi}(z)^T \tilde{\phi}(f(\mathbf{x})) \\
&= [\phi(z) - \bar{\phi}]^T [\phi(f(\mathbf{x})) - \bar{\phi}] \\
&= k(z, f(\mathbf{x})) - \frac{1}{n} \sum_{i=1}^{n} k(z, f(\mathbf{x}_i)) - \frac{1}{n} \sum_{i=1}^{n} k(f(\mathbf{x}), f(\mathbf{x}_i)) + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(f(\mathbf{x}'), f(\mathbf{x}_i)) \\
&= k(z, f(\mathbf{x})) - \frac{1}{n} \mathbf{1}^T \mathbf{K}_z - \frac{1}{n} \mathbf{1}^T \mathbf{K}_f(\mathbf{x}) + \frac{1}{n^2} \mathbf{1}^T \mathbf{K}\mathbf{1}.
\end{aligned}$$

$\tilde{\mathbf{K}}_z$ can then be expressed as:

$$\begin{aligned}\tilde{\mathbf{K}}_z &= [\tilde{k}(z, f(x_1)), \tilde{k}(z, f(x_2)), \ \ldots, \ \tilde{k}(z, f(x_n)))] \\ &= \mathbf{K}_z - \frac{1}{n}\mathbf{1}^T\mathbf{K}_z\mathbf{1} - \frac{1}{n}\mathbf{K}\mathbf{1} + \frac{1}{n^2}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1} \\ &= \mathbf{H}(\mathbf{K}_z - \frac{1}{n}\mathbf{K}\mathbf{1}),\end{aligned}$$

where $\mathbf{H}$ is the centring matrix, and $\mathbf{I}$ is the $n \times n$ identity matrix.

If a mapped observation is projected onto the first r basis vectors, $\mathbf{W}^r$, giving the projections $C_k(z)$. We use $C_k(z)$ to reconstruct $\phi_r(z)$ on the D-dimensional feature space,

$$\phi_r(z) = \sum_{k=1}^{r} W_k C_k(z) + \bar{\phi}. \tag{4.30}$$

This reconstruction of the mapped observation $\phi_r(z)$ cannot replace $\phi(z)$ without quantifying the difference between these two terms, which is usually called the observation reconstruction error (Salter et al., 2019). The observation reconstruction error ε_z is defined as

$$\varepsilon_z = \tilde{\phi}(z) - \tilde{\phi}_r(z). \tag{4.31}$$

Note that equations (4.30) and (4.31) only give expressions for $\phi_r(z)$ and ε_z, but they cannot be computed in general. In the following sections, we will show how to use these expressions for history matching in feature space by transforming the calculations with a kernel trick.

4.2.5 Kernel PCA and reconstruction

The simplest approach to history matching with kernel PCA would be to use the kernel approach to emulation, but to perform the history matching in the $m-$dimensional output space where the observations live. This is the idea in standard basis history matching and calibration when wishing to avoid comparison in the linear subspace (Salter et al., 2019; Wilkinson, 2010). This requires us to be able to reconstruct an emulator predictions in the output space. A straightforward approach for this is would be use an inverse mapping function, ϕ^{-1} which can map the data back from feature space to the original space. However, without an explicit expression of mapping function ϕ, it is impossible to know the form of ϕ^{-1} and so find the "pre-image" of $\phi(f(\mathbf{x}))$.

Schölkopf et al. (1998a) propose a method without requiring ϕ^{-1}. Given the expression of data, $\phi(f(\mathbf{x}))$, in feature space, $\phi(f(\mathbf{x})) \in \mathcal{F}$, a point in the original simulator output space, $\hat{f}(\mathbf{x}) \in \mathcal{M}$, is the pre-image of $\phi(f(\mathbf{x}))$ if it satisfies

$$\phi(f(\mathbf{x})) = \phi(\hat{f}(\mathbf{x})),$$

where $\phi(f(\mathbf{x}))$ is usually given by its reconstruction $\phi_r(f(\mathbf{x}))$. With any given vector in feature space, the exact pre image may not always exist. So Schölkopf et al. (1998a) define an approximate pre-image, $\hat{f}(\mathbf{x})$, for $\phi(f(\mathbf{x}))$ that satisfies

$$\phi(f(\mathbf{x})) \approx \phi(\hat{f}(\mathbf{x})).$$

Mika et al. (1999) find the pre-image of $\phi(f(\mathbf{x}))$ by minimising the squared distance between $\phi(\hat{f}(\mathbf{x}))$ and $\phi(f(\mathbf{x}))$, for any vector $\hat{f} \in \mathcal{M}$, such that

$$\hat{f}(\mathbf{x}) = \arg\min_{\hat{f}\in\mathcal{M}} \parallel \phi(\hat{f}) - \phi_r(f(\mathbf{x})) \parallel^2 .$$

By replacing the terms that do not involve $\phi(\hat{f})$ with Ω in the calculation, we have

$$\hat{f}(\mathbf{x}) = \arg\min_{\hat{f}\in\mathcal{M}} \left(k(\hat{f},\hat{f}) - 2\phi(\hat{f})^T \phi_r(f(\mathbf{x})) + \Omega \right). \tag{4.32}$$

Substituting equation (4.21) into equation (4.32), we arrive an expression which is written in terms that are known

$$\hat{f}(\mathbf{x}) = \arg\min_{\hat{f}\in\mathcal{M}} \left(k(\hat{f},\hat{f}) - 2\sum_{k=1}^{r} C_k(\mathbf{x}) \sum_{j=1}^{n} \tilde{\alpha}_{kj} k(\hat{f}, f(\mathbf{x}_i)) + \Omega \right). \tag{4.33}$$

Consequently, $\hat{f}(\mathbf{x})$ can be obtained once the kernel function is specified for any vector $f \in \mathcal{M}$. In the original paper, Mika et al. (1999) use a fixed-point iterative algorithm. For a Gaussian kernel with the formula $k(f(\mathbf{x}), f(\mathbf{x}')) = \exp(-\delta \parallel (f(\mathbf{x}) - f(\mathbf{x}')) \parallel^2)$ (and thus $k(f(\mathbf{x}), f(\mathbf{x}))$ is a constant for any $f(\mathbf{x})$). We deduce from equation (4.32) that we have to maximise $\phi(\hat{f})^T \phi_r(f(\mathbf{x}))$. Hence

$$\hat{f}(\mathbf{x}) = \arg\max_{\hat{f} \in \mathcal{M}} \left(\sum_{i=1}^{n} \tilde{\gamma}_i k(\hat{f}, f(\mathbf{x}_i)) \right). \tag{4.34}$$

where $\tilde{\gamma}_i = \sum_{k=1}^{r} C_k(\mathbf{x}) \alpha_{ki}$. For a maximum, the gradient with respect to $\hat{f}(\mathbf{x})$ is zero:

$$\sum_{i=1}^{n} \tilde{\gamma}_i \exp(-\delta \parallel (\hat{f}(\mathbf{x}) - f(\mathbf{x}_i)) \parallel^2)(\hat{f}(\mathbf{x}) - f(\mathbf{x}_i)) = 0.$$

This leads to a necessary condition for the maximum: $\hat{f}(\mathbf{x})$ should satisfy

$$\hat{f}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \tilde{\gamma}_i \exp(-\delta \parallel (\hat{f}(\mathbf{x}) - f(\mathbf{x}_i)) \parallel^2) f(\mathbf{x}_i)}{\sum_{i=1}^{n} \tilde{\gamma}_i \exp(-\delta \parallel (\hat{f}(\mathbf{x}) - f(\mathbf{x}_i)) \parallel^2)}. \tag{4.35}$$

As the kernel function is smooth, there is a neighbourhood of the extreme value of equation (4.35) in which the denominator is not equal to zero. So that $\hat{f}(\mathbf{x})$ can also be computed iteratively by

$$\hat{f}(\mathbf{x})_{t+1} = \frac{\sum_{i=1}^{n} \tilde{\gamma}_i \exp(-\delta \parallel (\hat{f}(\mathbf{x})_t - f(\mathbf{x}_i)) \parallel^2) f(\mathbf{x}_i)}{\sum_{i=1}^{n} \tilde{\gamma}_i \exp(-\delta \parallel (\hat{f}(\mathbf{x})_t - f(\mathbf{x}_i)) \parallel^2)}. \tag{4.36}$$

Many other approaches have been also proposed to solve the pre-image problem. A non-iterative approach of calculating the distance constraint has been proposed by Kwok and Tsang (2004). They propose a new method to constrain the embedding of the pre-image based on distance constraints (this will be introduced in Section 4.2.6), where the reconstruction of a new point is described by n neighbours based on least-squares solutions following (Gower, 1968).

The pre-image approach addresses the reconstruction method by minimising the squared distance between the mapped data in the feature space, which suggests that a small squared distance in feature space indicates a small squared distance in simulator output space. To figure out the relationship between a distance in simulator output space and the corresponding distance in feature space, distance constraints are introduced. History matching using a pre-image approach and distance constraints will be introduced in Section 4.3.

4.2.6 Distance constraints

For any two simulator outputs, $f(\mathbf{x})$ and $f(\mathbf{x}')$, in the simulator output space, we denote their mapped data in feature space as $\phi(f(\mathbf{x}))$ and $\phi(f(\mathbf{x}'))$, the simulator output space distance between these two vectors as $d_{f(\mathbf{x}),f(\mathbf{x}')}$, and the feature space distance as $d_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))}$. Without an explicit mapping function, any calculations in feature space must rely on a kernel trick, so that we can only compute the squared distance in feature space. We therefore give the relationships between the squared simulator output space distance $d^2_{f(\mathbf{x}),f(\mathbf{x}')}$ and the squared feature space distance $d^2_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))}$ for the different kernels.

We first consider RBF kernels, with a general form $k(f(\mathbf{x}), f(\mathbf{x}')) = \zeta(- \| (f(\mathbf{x}) - f(\mathbf{x}')) \|^2)$, where ζ is a function which is typically invertible. Kwok and Tsang (2004) present a simple relationship between squared simulator output space distance $d^2_{f(\mathbf{x}),f(\mathbf{x}')}$ and squared feature space distance $d^2_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))}$

$$\begin{aligned} d^2_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))} &= \| \phi(f(\mathbf{x})) - \phi(f(\mathbf{x}')) \|^2 \\ &= \phi(f(\mathbf{x}))^T \phi(f(\mathbf{x})) + \phi(f(\mathbf{x}'))^T \phi(f(\mathbf{x}')) - 2\phi(f(\mathbf{x}))^T \phi(f(\mathbf{x}')) \\ &= k(f(\mathbf{x}), f(\mathbf{x})) + k(f(\mathbf{x}'), f(\mathbf{x}')) - 2\zeta(- \| (f(\mathbf{x}) - f(\mathbf{x}')) \|^2) \\ &= k(f(\mathbf{x}), f(\mathbf{x})) + k(f(\mathbf{x}'), f(\mathbf{x}')) - 2\zeta\left(d^2_{f(\mathbf{x}),f(\mathbf{x}')}\right). \end{aligned} \tag{4.37}$$

Hence, the relationship between squared simulator output space distance $d^2_{f(\mathbf{x}),f(\mathbf{x}')}$ and squared feature space distance $d^2_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))}$ can be expressed as

$$\zeta\left(d^2_{f(\mathbf{x}),f(\mathbf{x}')}\right) = \frac{1}{2}\left(k(f(\mathbf{x}), f(\mathbf{x})) + k(f(\mathbf{x}'), f(\mathbf{x}')) - d^2_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))}\right). \tag{4.38}$$

For instance, with the squared exponential kernel, the function ζ is given as an exponential function,

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \exp\left(-\theta(f(\mathbf{x}) - f(\mathbf{x}'))^T(f(\mathbf{x}) - f(\mathbf{x}'))\right).$$

We then have the following relationship

$$d^2_{f(\mathbf{x}),f(\mathbf{x}')} = -\frac{1}{\theta}\log\left(\frac{1}{2}\left(k(f(\mathbf{x}), f(\mathbf{x})) + k(f(\mathbf{x}'), f(\mathbf{x}')) - d^2_{\phi(f(\mathbf{x})),\phi(f(\mathbf{x}'))}\right)\right).$$

In addition, for dot product kernels of the form $k(f(\mathbf{x}), f(\mathbf{x}')) = \zeta(f(\mathbf{x})^T f(\mathbf{x}'))$, there is a relationship between dot product, $\phi(f(\mathbf{x}))^T\phi(f(\mathbf{x}'))$, in the feature space and dot product, $f(\mathbf{x})^T f(\mathbf{x}')$, in the model output space (Kwok and Tsang, 2004; Williams, 2001),

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \phi(f(\mathbf{x}))^T\phi(f(\mathbf{x}')) = \zeta(f(\mathbf{x})^T f(\mathbf{x}')).$$

Note that ζ is often a invertible function, for example, for a degree-p polynomial kernels

$$k(f(\mathbf{x}), f(\mathbf{x}')) = (f(\mathbf{x})^T f(\mathbf{x}') + c)^p,$$

where $c \geqslant 0$ is a free parameter. When p is odd, we have

$$f(\mathbf{x})^T f(\mathbf{x}') = k(f(\mathbf{x}), f(\mathbf{x}'))^{\frac{1}{p}} - c = \phi(f(\mathbf{x}))^T\phi(f(\mathbf{x}')^{\frac{1}{p}} - c.$$

Likewise, for the sigmoid kernel

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \tanh(\gamma f(\mathbf{x})^T f(\mathbf{x}') - c),$$

where c and γ are free parameters. We then have

$$f(\mathbf{x})^T f(\mathbf{x}') = \frac{\tanh^{-1}(k(f(\mathbf{x}), f(\mathbf{x}'))) + c}{\gamma}.$$

With the given $f(\mathbf{x})^T f(\mathbf{x}')$ in these examples, the corresponding squared distance in the original space, $d^2_{f(\mathbf{x}),f(\mathbf{x}')}$ can be computed by kernel functions.

Distance constraints have wide applications for kernel methods. As we described, they are used to solve the pre-image problem (Kwok and Tsang, 2004). Williams (2001) uses distance constraints for a metric multidimensional scaling algorithm. We will use these relationships to quantify uncertainties in the calibration process, details of which are given in Section 4.5.

4.3 History matching in feature space

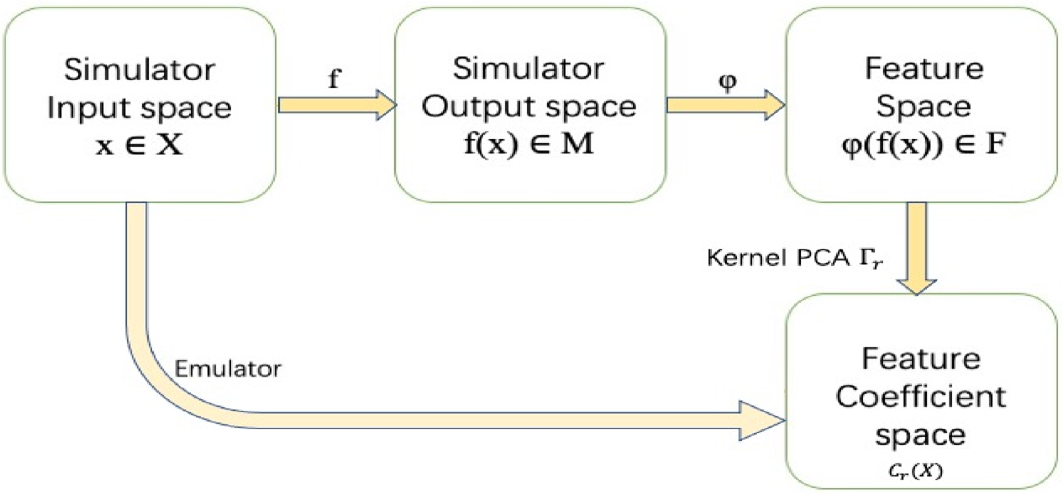

Simulator Input space x ∈ X
f
Simulator Output space f(x) ∈ M
φ
Feature Space φ(f(x)) ∈ F
Kernel PCA Γr
Emulator
Feature Coefficient space Cr(X)

Fig. 4.1 The relationship between metric spaces.

We introduced several metric spaces in the last section, and Figure 4.1 shows the relationship between these spaces, where f is the simulator function, and ϕ is the mapping function that maps the model outputs from $\mathcal{M} \subset \mathbb{R}^m$ into a higher-dimension feature space, $\mathcal{F} \subset \mathbb{R}^D$, determined by a kernel function.

Emulators and history matching are both traditionally performed on the simulator output field (or a subspace of simulator output space). But, with kernel PCA, we build emulators for the kernel PCA coefficients in a 'feature coefficient space' given the reconstructions $\phi(f(\mathbf{x}))$ on the feature space. To achieve standard history matching in the simulator output metric space with kernel PCA, we need to find the pre-image for $\phi(f(\mathbf{x}))$ for every parameter choice we used in the calibration process. However, there are two challenges. Firstly, pre-image approaches reconstruct data with no uncertainty. Yet we have emulator uncertainty throughout which must be accounted for. The second issue is that finding the pre-image (which relies on nonlinear optimization) for massive data is inevitably expensive and slow, which works against the initial computational motivation. These challenges makes traditional approaches to history matching in the simulator output metric space with kernel PCA arguably infeasible.

Instead of finding an approximate pre-image for each $\phi(f(\mathbf{x}))$, and comparing it with the observation (with uncertainties accounted for), we perform history matching in feature space, inspired by the pre-image reconstruction of kernel PCA (Mika et al., 1999). The comparison between the pre-image and history matching problems is shown in Figure 4.2. If we reverse the pre-image problem, then the process is exactly what is required for history matching.

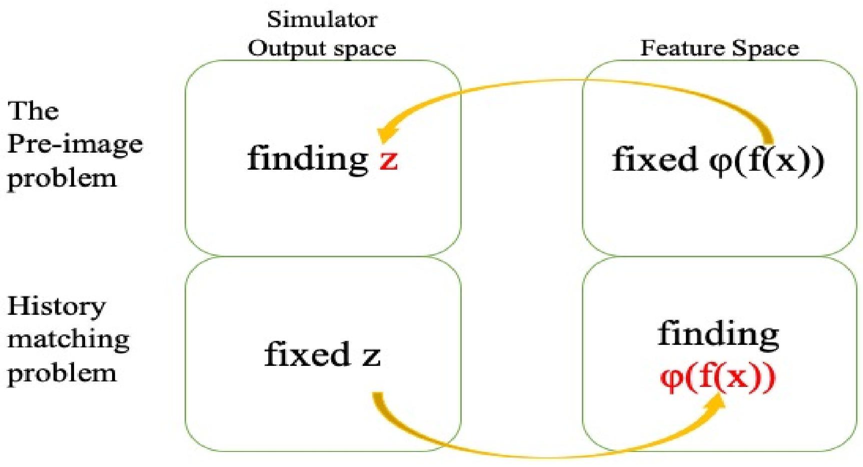


Fig. 4.2 The comparison between the pre-image and history matching.

Mika et al. (1999) address the pre-image problem by comparing $\phi(f(\mathbf{x}))$ and its mapped reconstruction in feature space. We perform a history matching on the feature space with the same idea. The observation is mapped into feature space, which can be used to compare with mapped simulator outputs $\phi(f(\mathbf{x}))$. Given an implausibility function in feature space, we can rule out input parameter settings, $\mathbf{x}$, which lead to $\phi(f(\mathbf{x}))$ that are not consistent with the mapped observation $\phi(z)$. To achieve this idea, we define a new implausibility in feature space.

4.3.1 Implausibility in feature space

The implausibility (introduced in Section 2.5), $I(\mathbf{x})$, is defined to measure the distance between the simulator outputs and observations. We can write the implausibility function as a distance,

$$I(\mathbf{x}) = ||z - f(\mathbf{x})||_f, \tag{4.39}$$

where the $||\cdot||_f$ represents an appropriate measure that accounts for all sources of uncertainties, a Mahalanobis-type function is a natural choice for $||\cdot||_f$,

$$I(\mathbf{x}) = (z - f(\mathbf{x}))^T (\mathrm{Var}[z - f(\mathbf{x})])^{-1} (z - f(\mathbf{x})). \tag{4.40}$$

When we use the emulator predictions to represent the simulator outputs, the implausibility can be written as

$$I(\mathbf{x}) = ||z - \mathrm{E}[f(\mathbf{x})]\,||_f. \tag{4.41}$$

Using a Mahalanobis-type function for $||\cdot||_f$, the implausibility can then be written as

$$I(\mathbf{x}) = (z - \mathrm{E}[f(\mathbf{x})])^T (\mathrm{Var}[z - \mathrm{E}[f(\mathbf{x})]])^{-1} (z - \mathrm{E}[f(\mathbf{x})]), \tag{4.42}$$

where

$$\mathrm{Var}[z - \mathrm{E}[f(\mathbf{x})]] = \mathrm{Var}[e] + \mathrm{Var}[\eta] + \mathrm{Var}[f(\mathbf{x})],$$

and e is the observation error and η is the discrepancy. The implausibility $I(\mathbf{x})$ is then used to rule out regions of input space that give model outputs that are 'too far' from observations. In standard history matching, described in Section 2.5, for simulator $f(\cdot)$ and observation z, a lager value of implausibility $I(\mathbf{x})$ at any given input $\mathbf{x}$ implies that, relative to the uncertainties, it is implausible that the output of simulator at $\mathbf{x}$, $f(\mathbf{x})$, is consistent with the observations. A threshold, T, is chosen to define the NROY space, such that any inputs of $I(\mathbf{x}) > T$ are implausible.

Analogous to the standard implausibility in equation (4.39), we define the implausibility as a distance in feature space,

$$\mathcal{I}(\mathbf{x}) = ||\phi(z) - \phi(f(\mathbf{x}))||_f, \tag{4.43}$$

where $\phi(z)$ is the mapped observation, $\phi(f(\mathbf{x}))$ is the mapped simulator output in feature space and $||\cdot||_f$ represents an appropriate metric. The pre-image approach addresses the reconstruction problem by minimising the distance between the mapped output and its reconstruction. We perform a calibration on the feature space with the same goal: the distance between mapped observation and output is parallel to the implausibility function used for ruling out space in history matching. However, unlike history matching, the pre-image approach does not consider any uncertainties in the distance function. This suggests that the distance between a mapped data and its mapped exact pre-image should be zero. History matching does not expect an exact match, due to the observation error and discrepancy being included in the distance (Craig et al., 1996; Vernon et al., 2010; Williamson et al., 2013). To make meaningful comparisons between simulator outputs and observations in feature space, all of the sources of uncertainty need to be addressed in history matching. Therefore, before determining an appropriate measure for $||\cdot||_f$, all sources of uncertainties need to be quantified first. Due to the uncertainties naturally belonging in different spaces, we need to quantify all sources of uncertainty in separate processes. We need to determine discrepancy and observation errors in feature space first, then consider emulator uncertainty, discrepancy and observation errors together in feature space history matching.

To determine discrepancy and observation errors in feature space history matching, we have explored a number of potential approaches. According to different uncertainty quantification approaches, we roughly divide feature space history matching into three classes: history matching with projected uncertainties, history matching with distance constraints, and kernel-based history matching. The details of the potential approaches are introduced individually in the following sections. The numerical examples given in Section 4.8 compare these approaches.

4.4 History matching with projected uncertainties

For researchers who are experienced with history matching, projecting observation error and discrepancy into feature space is the most natural approach to consider. To perform history matching in feature space, we first propose to project discrepancy and observation error into feature space. After that, we define an implausibility using these projected uncertainties.

4.4.1 Projecting uncertainties into feature space

To project observation error and discrepancy into feature space, we recall the statistical model from Section 2.5, which links the observation to simulator output via

$$z = f(\mathbf{x}^*) + e + \eta,$$

where $\mathbf{x}^*$ is the best input and where the observation error, e, and the discrepancy, η, are uncorrelated mean-zero terms, with positive definite variance matrices Σ_e and Σ_η, e.g.

$$e \sim N(0,\, \Sigma_e), \quad \eta \sim N(0,\, \Sigma_\eta).$$

Similar to the projection of observations introduced in Section 4.2.4, we can compute the projections of uncertainties in the coefficient space. We denote the mapped observation error as $\phi(e)$, the mapped discrepancy as $\phi(\eta)$, and $\mathbf{C}(e)$ and $\mathbf{C}(\eta)$ as their projections onto the basis $\mathbf{W}$, where $\mathbf{C}(e)$ and $\mathbf{C}(\eta)$ are length n vectors, $\mathbf{C}(e) = [C_1(e),\, \dots,\, C_n(e)]^T$ and $\mathbf{C}(\eta) = [C_1(\eta),\, \dots,\, C_n(\eta)]^T$. Following equation (4.28), $C_k(e)$ and $C_k(\eta)$ are given as

$$C_k(e) = W_k^T \tilde{\phi}(e) = \tilde{\alpha}_k^T \tilde{\mathbf{K}}_e, \tag{4.44}$$

and

$$C_k(\eta) = W_k^T \tilde{\phi}(\eta) = \tilde{\alpha}_k^T \tilde{\mathbf{K}}_\eta, \tag{4.45}$$

where

$$\tilde{\mathbf{K}}_e = [\tilde{k}(e, f(x_1)), \tilde{k}(e, f(x_2)), \ \ldots, \ \tilde{k}(e, f(x_n))],$$

and

$$\tilde{\mathbf{K}}_\eta = [\tilde{k}(\eta, f(x_1)), \tilde{k}(\eta, f(x_2)), \ \ldots, \ \tilde{k}(\eta, f(x_n))].$$

Given Σ_e and Σ_η, we can sample e and η and then compute the sample projections $\mathbf{C}(e)$ and $\mathbf{C}(\eta)$ by equations (4.44) and (4.45). We can then use these samples to obtain the expectation and variance of the sampling distribution for $\mathbf{C}(e)$ and $\mathbf{C}(\eta)$. Expectations and variances in feature space can be written in terms of $\mathbf{C}_r(e)$. To give the expression of the observation error in feature space, we have

$$\mathrm{E}\left[\phi(e)\right] = \mathbf{W}^r \mathrm{E}\left[\mathbf{C}_r(e)\right] + \bar{\phi}, \tag{4.46}$$

and

$$\mathrm{Var}\left[\phi(e)\right] = \mathbf{W}^r \mathrm{Var}\left[\mathbf{C}_r(e)\right] \mathbf{W}^{rT}. \tag{4.47}$$

Note, we cannot compute these expressions ($\mathbf{W}^r$ is unknown). For giving the expression of the discrepancy in feature space, we have

$$\mathrm{E}\left[\phi(\eta)\right] = \mathbf{W}^r \mathrm{E}\left[\mathbf{C}_r(\eta)\right] + \bar{\phi}, \tag{4.48}$$

and

$$\mathrm{Var}\left[\phi(\eta)\right] = \mathbf{W}^r \mathrm{Var}\left[\mathbf{C}_r(\eta)\right] \mathbf{W}^{rT}. \tag{4.49}$$

Given the projections of discrepancy, the projections of observation error, the expression of the uncertainties in feature space, we want to calibrate using all available information. Standard history matching uses a Mahalanobis-type distance for $||\cdot||_f$, such as equations (4.40) and (4.42), to capture all sources of uncertainty. However, without the explicit expression of mapping function $\phi(\cdot)$ or even knowledge of it's dimension, it is impossible to apply the same choice for $||\cdot||_f$ in feature space. Instead, it is attractive to apply a low-dimensional basis approach to history match for computer model with high-dimensional outputs (Salter and Williamson, 2019). Analogous to multivariate history matching using SVD/PCA basis projection methods, we define a 'coefficient implausibility' for history matching in feature space.

4.4.2 Coefficient implausibility

Calibration of expensive computer models with high-dimensional output fields can be performed with basis methods, including probabilistic calibration (Chang et al., 2014, 2016; Higdon et al., 2008; Sexton et al., 2012) and history matching (Salter and Williamson, 2019). For history matching with basis methods, we have introduced the multivariate implausibility on coefficient space in Section 2.5.3, as:

$$\mathcal{I}_c(\mathbf{x}) = (\mathbf{C}_r(\mathbf{z}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T (\mathrm{Var}[\mathbf{C}_r(\eta)] + \mathrm{Var}[\mathbf{C}_r(e)] + \mathrm{Var}[\mathbf{C}_r(\mathbf{x})])^{-1} (\mathbf{C}_r(\mathbf{z}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]). \tag{4.50}$$

Given emulators for the kernel PCA coefficients on basis $\mathbf{W}^r$, observation projections, $\mathbf{C}_r(\mathbf{z})$ (by equation (4.28)), and the projections $\mathbf{C}(e)$ and $\mathbf{C}(\eta)$ (by equations (4.44) and (4.45)), we can history match in the subspace defined by the basis $\mathbf{W}^r$, using the coefficient implausibility in equation (4.50).

Analogous to standard history matching with coefficient implausibility, large values of $\mathcal{I}_c(\mathbf{x})$, indicate that it is implausible that $\mathbf{x} = \mathbf{x}^*$. Using $\mathcal{I}_c(\mathbf{x})$, 'Not Ruled Out Yet' (NROY) space contains all not implausible $\mathbf{x}$ defined as

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_c(\mathbf{x}) \leq T\}, \tag{4.51}$$

for a threshold bound T. All of the runs less than T will be retained in the NROY space, even if they are not good. Small values of $\mathcal{I}_c(\mathbf{x})$ do not necessarily imply good models, as small values can occur when the uncertainties are large. If the implausibility is larger than T, we can be sure that the model output is too far from the observations and we can safely cut this parameter choice. To choose a suitable threshold T, we introduce the following methods.

4.4.3 Threshold T

In standard history matching, for setting up the cutoff threshold, the implausibility $\mathcal{I}(\mathbf{x})$ in equation (4.42) can be compared with a Chi-squared distribution with l degrees of freedom, where l is the dimension of the model output (Vernon et al., 2010). Then the threshold T could be $95th$ percentile or $99.5th$ percentile of a Chi-squared distribution with l degrees of freedom. However, this approach for setting the cutoff threshold is not available for our implausibility in equation (4.50), since $\mathcal{I}_c(\mathbf{x})$ does not follow a Chi-squared distribution with r degrees of freedom (the details are given in Section 4.4.4).

There are also a number of different ways to set the value of threshold T. For example, using a rule-of-thumb as in Salter et al. (2018), where ice sheet simulations with an extent error of $> 23\%$ or 25% were discarded based on experts judgement (the tolerance was chosen based on comparing computer model outputs to observations, and judging which are acceptable).

Similarly, we can use expert judgement to set an appropriate cut-off value. This idea is inspired by expert tuning which is quite common in the climate community to adjust the model parameters manually (Bellprat et al., 2012; Mauritsen et al., 2012). By performing expert tuning for the computer outputs of the ensemble F, we have expert judgement that divides the ensemble members into acceptable runs, F^A, and unacceptable runs, F^U, and the corresponding inputs are separated as acceptable inputs, $\mathbf{x}^A$, and unacceptable inputs, $\mathbf{x}^U$. The F^A are selected by experts. This can be done, for example, with a strict requirement, that the acceptable runs are the members of the training data consistent with observations in the judgement of the modeller. If no such runs or very few exist in the ensemble, the acceptable runs could be set to be the "best" in the ensemble (where "best" is in the view of the experts).

With the above classification of the ensemble members (F^A, F^U), we can set the threshold as the tolerance of the differences between simulator outputs and observations. This is in keeping with a tolerance to error approach to discrepancy, which considers the tolerance given by the experts as discrepancy for the purposes of history matching (Couvreux et al., 2020; Williamson et al., 2017). The details of the tolerance approach are introduced in Section 2.4.1.

However, the expert's tolerance of the differences between simulator outputs and observation is not given directly. By using the implausibility function and experts classification information, an interval of possible tolerance values can be defined. Denote the minimum implausibility for the unacceptable runs as $\min(\mathcal{I}_c(\mathbf{x}^U))$, and the maximum implausibility for the acceptable runs as $\max(\mathcal{I}_c(\mathbf{x}^A))$, the interval can be written as:

$$[\max(\mathcal{I}_c(\mathbf{x}^A)), \min(\mathcal{I}_c(\mathbf{x}^U))]. \tag{4.52}$$

Only if the threshold is bigger than the maximum implausibility for the acceptable runs, can all of the acceptable runs be retained in NROY space, and only if the threshold is smaller than the minimum implausibility for the unacceptable runs, can all of the unacceptable runs be ruled out. But a range cannot be used as the threshold, a precise value for T is required. Due to the fact that the middle ground between $\max(\mathcal{I}_c(\mathbf{x}^A))$ and $\min(\mathcal{I}_c(\mathbf{x}^U))$ is unknowable, and there might be a chance that runs give implausibility within this range are acceptable, to avoid ruling out any 'good' runs, we use the minimum implausibility for the unacceptable runs $\min(\mathcal{I}_c(\mathbf{x}^U))$ as the threshold

$$T = \min(\mathcal{I}_c(\mathbf{x}^U)). \tag{4.53}$$

The advantage to setting implausibility thresholds by equation (4.53) is that expert judgement can be easily included into the automatic calibration process. In the next chapter, other approaches for setting implausibility thresholds by using expert judgement will be introduced.

4.4.4 Limitations

History matching in feature space (with the subspace defined by $\mathbf{W}^r$) with a kernel PCA basis can be achieved following the proposed implausibility in Section 4.4.2. However, there are two major issues with the methodology. Firstly, for standard history matching with the SVD/PCA basis, we perform the calibration on a subspace of the simulator output space, so that the model assumption (in Section (4.4.1)) still holds on the subspace defined by the SVD/PCA basis. Denote the SVD/PCA basis vector as $\mathbf{W}_{PCA_r}$, we have

$$\mathbf{W}_{PCA_r}^T z = \mathbf{W}_{PCA_r}^T f(\mathbf{x}^*) + \mathbf{W}_{PCA_r}^T e + \mathbf{W}_{PCA_r}^T \eta,$$

and this can be written as

$$\mathbf{C}_{PCA_r}(\mathbf{z}) = \mathbf{C}_{PCA_r}(\mathbf{x}^*) + \mathbf{C}_{PCA_r}(e) + \mathbf{C}_{PCA_r}(\eta). \tag{4.54}$$

Under the model assumptions in equation (4.54), we have:

$$\mathrm{Var}\left[\mathbf{C}_{PCA_r}(\mathbf{z}) - \mathrm{E}\left[\mathbf{C}_{PCA_r}(\mathbf{x})\right]\right] = \mathrm{Var}\left[\mathbf{C}_{PCA_r}(\eta)\right] + \mathrm{Var}\left[\mathbf{C}_{PCA_r}(e)\right] + \mathrm{Var}\left[\mathbf{C}_{PCA_r}(\mathbf{x})\right],$$

where $\mathbf{C}_{PCA_r} = \mathbf{C}_{PCA_r}(\mathbf{z})$. However, all of these relationships do not hold with nonlinear kernel based history matching. With a nonlinear mapping function, $\phi(\cdot)$, the linear relationship in the statistical model does not hold in the feature space,

$$\phi(z) \neq \phi(f(\mathbf{x}^*)) + \phi(e) + \phi(\eta). \tag{4.55}$$

$\phi(e)$ is not the difference between the observation with the true value of the system in feature space, and $\phi(\eta)$ cannot represent the difference between the simulator output given at the 'best' setting, $\phi(f(\mathbf{x}^*))$, and reality. If the model assumptions do not exist on the feature space, how do we account for the implausibility in the right way and use it for feature space history matching without biasing the parameter inference?

Additionally, another problem is that we can only compute the distance between observation projections and emulator predictions on the coefficient space rather than the feature space. Salter and Williamson (2019) present efficient calibration for high-dimensional computer model output using basis methods, but the method cannot be applied without knowing the mapping function, $\phi(\cdot)$ and basis vector $\mathbf{W}$. While these two problems cannot be solved, we still can history match with projected uncertainties, the results is shown in Section 4.8. To overcome these two problems, one idea is to use distance constraints to account for discrepancy and observation error. A new approach to do history matching in feature space is presented in Section 4.5.

4.5 History matching in feature space with distance constraints

4.5.1 Implausibility in feature space

The implausibility defined in equation (4.43) is written as a distance that measures the difference between the simulator outputs and observations. To define a calculable implausibility for history matching in feature space, we could adopt the Euclidean norm for $||\phi(z) - \phi(f(\mathbf{x}))||_f$, which can be calculated using the kernel trick, so that we do not need the explicit expression of the mapping function $\phi(\cdot)$. However, using the Euclidean norm for the distance function raises an obvious question: how should we account for the uncertainties (discrepancy, observation error and emulator uncertainty)? In contrast with the fixed threshold used in standard history matching, a proposed solution is to define the threshold as a function of $\mathbf{x}$ that accounts for all sources of uncertainty. We propose to use distance constraints to set this threshold function and this proposition is developed in Section 4.5.2.

Before setting the threshold function using distance constraint, the implausibility function is required. Given the observation, z, and the expressions for emulator predictions $\mathrm{E}[\phi(f(\mathbf{x}))]$ in equation (4.26), we define the implausibility as the distance between the mapped observation and the emulator prediction:

$$\mathcal{I}_D(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]). \tag{4.56}$$

Equation (4.56) can be efficiently calculated using a kernel trick, without requiring the mapping function $\phi(\cdot)$, as

$$\begin{aligned}
\mathcal{I}_D(\mathbf{x}) &= (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]) \\
&= \left(\phi(z) - (\mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})] + \bar{\phi})\right)^T\left(\phi(z) - (\mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})] + \bar{\phi})\right) \\
&= \left(\phi(z) - \bar{\phi} - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]\right)^T\left(\phi(z) - \bar{\phi} - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]\right) \\
&= \left(\tilde{\phi}(z) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]\right)^T\left(\tilde{\phi}(z) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]\right) \\
&= \tilde{\phi}(z)^T\tilde{\phi}(z) + (\mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]) - 2\tilde{\phi}(z)^T(\mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]) \\
&= \tilde{k}(z,z) + \left(\sum_{k=1}^{r} W_k\mathrm{E}[C_k(\mathbf{x})]\right)^T\left(\sum_{k=1}^{r} W_k\mathrm{E}[C_k(\mathbf{x})]\right) - 2\tilde{\phi}(z)^T\left(\sum_{k=1}^{r} W_k\mathrm{E}[C_k(\mathbf{x})]\right) \\
&= \tilde{k}(z,z) + \sum_{k=1}^{r}\mathrm{E}[C_k(\mathbf{x})]^T W_k^T W_k\mathrm{E}[C_k(\mathbf{x})] - 2\tilde{\phi}(z)^T\left(\sum_{k=1}^{r}\sum_{i=1}^{n}\tilde{\alpha}_{ki}\tilde{\phi}(f(x_i))\mathrm{E}[C_k(\mathbf{x})]\right) \\
&= \tilde{k}(z,z) + \sum_{k=1}^{r}\mathrm{E}[C_k(\mathbf{x})]^T\mathrm{E}[C_k(\mathbf{x})] - 2\left(\sum_{k=1}^{r}\sum_{i=1}^{n}\tilde{\alpha}_{ki}\tilde{\phi}(z)^T\tilde{\phi}(f(x_i))\mathrm{E}[C_k(\mathbf{x})]\right) \\
&= \tilde{k}(z,z) + \mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathrm{E}[\mathbf{C}_r(\mathbf{x})] - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathbf{A}\tilde{\mathbf{K}}_z,
\end{aligned} \tag{4.57}$$

where $\mathbf{A}$ is the matrix containing the first r eigenvectors of the centred kernel matrix, $\tilde{\mathbf{K}}$, and

$$\tilde{\mathbf{K}}_z = [\tilde{k}(z, f(\mathbf{x}_1)), \tilde{k}(z, f(\mathbf{x}_2)), \ \ldots, \ \tilde{k}(z, f(\mathbf{x}_n))].$$

The implausibility, $\mathcal{I}_D(\mathbf{x})$, represents the difference between the observation and the expectation of simulator output in feature space. In traditional history matching, a large value of the implausibility (shown in equation (4.42)), relative to all sources of uncertainty, at any input $\mathbf{x}$ implies that, the mapped simulator output at $\mathbf{x}$ is very far from where we would expect it to be. However, all sources of uncertainty are not considered in our new defined feature space implausibility, $\mathcal{I}_D(\mathbf{x})$. When the simulator output is 'close to' the observations (in terms of the relevant uncertainties), the squared distance between the mapped observation and the emulator prediction may not be small. We proposed to allow for this by using a variable cutoff threshold $T(\mathbf{x})$.

4.5.2 Threshold function $T(\mathbf{x})$

To define the threshold as a function of $\mathbf{x}$ to account for the uncertainties, we first consider the expectation and variance of $\mathcal{I}_D(\mathbf{x})$. However, its distribution is only available through sampling, so thresholds based on quantiles of $\mathcal{I}_D(\mathbf{x})$, as in Salter and Williamson (2019) are not efficiently available. To find a form of threshold function, we use the triangle inequality to prove that there is a upper bound, $L(\mathbf{x}^*)$ of the implausibility $\mathcal{I}_D(\mathbf{x}^*)$ at the best input $\mathbf{x}^*$,

$$\begin{aligned}\mathcal{I}_D(\mathbf{x}^*) &= \parallel \phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \parallel^2 \\ &= \parallel \phi(z) - \phi(f(\mathbf{x}^*)) + \phi(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \parallel^2 \\ &\leq \parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2 + \parallel \phi(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \parallel^2 \\ &= L(\mathbf{x}^*).\end{aligned} \tag{4.58}$$

Thus, the upper bound, $L(\mathbf{x}^*)$, is the sum of the L_2 distance between the observation and the model output at the best input in feature space, $\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2$, and the L_2 distance between the model output and it's expectation in feature space, $\parallel \phi(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \parallel^2$. The idea is to set the $\mathbf{x}$-dependent threshold $T(\mathbf{x})$ using the expectation and the variance of $L(\mathbf{x}^*)$,

$$\mathrm{E}\left[L(\mathbf{x}^*)\right] = \mathrm{E}\left[\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2\right] + \mathrm{E}\left[\parallel \phi(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \parallel^2\right],$$

$$\mathrm{Var}\left[L(\mathbf{x}^*)\right] = \mathrm{Var}\left[\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2\right] + \mathrm{Var}\left[\parallel \phi(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \parallel^2\right].$$

The discrepancy and observation error are accounted for in $\mathrm{E}\left[\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2\right]$ and $\mathrm{Var}\left[\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2\right]$, and the emulator uncertainties are accounted for in $\mathrm{E}[\| \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*)) \|]]$ and $\mathrm{Var}[\| \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*)) \|]]$ (details for computing these expectations and variances will be given in Section 4.5.4).

By letting the threshold for each input $T(\mathbf{x})$ $(T(\mathbf{x}) > 0)$ be

$$T(\mathbf{x}) = \mathrm{E}[L(\mathbf{x}^*)] + t\sqrt{\mathrm{Var}[L(\mathbf{x}^*)]}, \tag{4.59}$$

the Chebyshev–Cantelli inequality (one-sided Chebyshev inequality) gives that

$$\begin{aligned} &Pr\left(L(\mathbf{x}^*) - \mathrm{E}[L(\mathbf{x}^*)] > t\sqrt{\mathrm{Var}[L(\mathbf{x}^*)]}\right) \leq \frac{\mathrm{Var}[L(\mathbf{x}^*)]}{\mathrm{Var}[L(\mathbf{x}^*)] + t^2\mathrm{Var}[L(\mathbf{x}^*)]}, \\ &\Rightarrow Pr\left(L(\mathbf{x}^*) > \mathrm{E}[L(\mathbf{x}^*)] + t\sqrt{\mathrm{Var}[L(\mathbf{x}^*)]}\right) \leq \frac{1}{1+t^2}, \\ &\Rightarrow Pr\left(L(\mathbf{x}^*) > T(\mathbf{x})\right) \leq \frac{1}{1+t^2}, \end{aligned} \tag{4.60}$$

where the value of t $(t > 0)$ represents that for $L(\mathbf{x}^*)$, at least $\frac{t^2}{1+t^2}$ of it's distribution's values are greater than t standard deviations from the mean (Hazewinkel, 2001). By setting $t = 3$ for both equations (4.59) and (4.60), we have $T(\mathbf{x}) = \mathrm{E}[L(\mathbf{x}^*)] + 3\sqrt{\mathrm{Var}[L(\mathbf{x}^*)]}$, and the probability that $L(\mathbf{x}^*)$ smaller than $T(\mathbf{x})$ is bigger than 90%,

$$Pr\left(L(\mathbf{x}^*) < T(\mathbf{x})\right) \geq 90\%. \tag{4.61}$$

Given equations (4.58) and (4.62), we have that the probability that $\mathcal{I}_D(\mathbf{x}^*)$ smaller than $T(\mathbf{x})$ is bigger than 90%,

$$Pr\left(\mathcal{I}_D(\mathbf{x}^*) < T(\mathbf{x})\right) \geq 90\%. \tag{4.62}$$

For any input $\mathbf{x}$, if $\mathbf{x} = \mathbf{x}^*$, then the probability that it's implausibility, $\mathcal{I}_D(\mathbf{x})$, is bigger than the threshold $T(\mathbf{x})$ does not exceed 10%. Hence, this choice of t implies that we view a value of the implausibility $\mathcal{I}_D(\mathbf{x})$ bigger than threshold $T(\mathbf{x})$ as indicating that it is implausible that $\mathbf{x} = \mathbf{x}^*$. In order to compute $T(\mathbf{x})$ via equation

(4.59), we require expectation and variance of $L(\mathbf{x}^*)$ which will derive in the next two subsections.

4.5.3 Accounting for uncertainties using distance constraints

Following the model assumption introduced in Section (4.4.1), the sum of the errors in the simulator output space, $e+\eta$ can be represented as the distance between z and $f(\mathbf{x}^*)$,

$$d_{z,f(\mathbf{x}^*)} = z - f(\mathbf{x}^*) = e + \eta.$$

We give the projection of the errors in the feature space as the distance between $\phi(z)$ and $\phi(f(\mathbf{x}^*))$, $d_{\phi(z),\phi(f(\mathbf{x}^*))} = \phi(z) - \phi(f(\mathbf{x}^*))$. Recall the distance constraints introduced in Section 4.2.6: for many commonly used kernel functions, there is a relationship between the squared distance $d^2_{\phi(z),\phi(f(\mathbf{x}^*))}$ and $d^2_{z,f(\mathbf{x}^*)}$ (Kwok and Tsang, 2004), where

$$d^2_{\phi(z),\phi(f(\mathbf{x}^*))} = \parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2,$$

and

$$d^2_{z,f(\mathbf{x}^*)} = \parallel z - f(\mathbf{x}^*) \parallel^2 .$$

Therefore, for certain commonly used kernels, we could compute the L_2 norm of the sum of uncertainties in feature space, $d^2_{\phi(z),\phi(f(\mathbf{x}^*))}$ using $d^2_{z,f(\mathbf{x}^*)}$.

The distribution of e and η means that the distance $z - f(\mathbf{x}^*)$ has a Normal distribution

$$z - f(\mathbf{x}^*) \sim N(0,\ \Sigma_e + \Sigma_\eta).$$

By using the Normal property, we can derive the expectation and variance for the distribution of $\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2$. Note that the distribution for $\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2$ is different for different kernel functions. For instance, following equation (4.37), with a generalised Gaussian kernel $k(z, f(\mathbf{x}^*)) = \exp(-\sigma \parallel z - f(\mathbf{x}^*) \parallel^2)$, the relationship between $\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2$ and $||z - f(\mathbf{x}^*)||^2$ is

$$\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2 = 2 - 2\exp(-\sigma \parallel z - f(\mathbf{x}^*) \parallel^2). \tag{4.63}$$

We have the the expectation and variance (the full computation is given in Appendix A.3)

$$\mathrm{E}\left[\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2\right] = 2 - 2\frac{|(2\sigma\mathbf{I}_{m\times m} + (\Sigma_e + \Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{|\Sigma_e + \Sigma_\eta|^{\frac{1}{2}}},$$

$$\mathrm{Var}\left[\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2\right] = 4\frac{|(4\sigma\mathbf{I}_{m\times m} + (\Sigma_e + \Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{|\Sigma_e + \Sigma_\eta|^{\frac{1}{2}}} - 4\left(\frac{|(2\sigma\mathbf{I}_{m\times m} + (\Sigma_e + \Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{|\Sigma_e + \Sigma_\eta|^{\frac{1}{2}}}\right)^2.$$

For specific RBF kernels such as the Gaussian kernel, we can compute the expectation and variance of $\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2$. But for other kernels, such as dot kernels, we can only access the expectation and variance of $\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2$ by sampling. Using distance constraints, the discrepancy and observation error can be accounted for in the feature space, to compute $T(\mathbf{x})$. We will introduce treatment of the emulator uncertainty in the next section.

4.5.4 Emulator uncertainty

The other element of equation (4.58), $\| \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))] \|^2$, is the distance between the model outputs and the emulator predictions. This is the variance of the emulator in feature space. We want to compute the expectation and variance of $\| \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))] \|^2$. However, the expectation and the variance of $\phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))]$, shown in equations (4.26) and (4.27), includes the unknown basis vector $\mathbf{W}^r$. To compute the expectation and variance for $\| \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))] \|^2$, we first write it as

$$\begin{aligned}\| \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))] \|^2 &= \| \phi(f(\mathbf{x}^*)) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)] \|^2 \\ &= \| \phi_r(f(\mathbf{x}^*)) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)] \|^2 + ||\varepsilon_f||^2, \\ &= \| \mathbf{W}^r\mathbf{C}_r(\mathbf{x}^*) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)] \|^2 + ||\varepsilon_f||^2, \\ &= \| \mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)] \|^2 + ||\varepsilon_f||^2,\end{aligned} \tag{4.64}$$

where ε_f is the reconstruction error of the model output accounting for the uncertainty in the approximation,

$$||\varepsilon_f||^2 = \tilde{k}(f(\mathbf{x}^*), f(\mathbf{x}^*)) - (\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*)))^T(\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*))).$$

However, $||\varepsilon_f||^2$ cannot be computed without known $f(\mathbf{x}^*)$. A possible solution here is to use $\phi_r(f(\mathbf{x}^*))$ as a approximation of $\phi(f(\mathbf{x}^*))$, but ignoring $||\varepsilon_f||^2$ could give a threshold smaller than the actual value, and parameter space can be ruled out wrongly. We use the reconstruction error of the training data to approximate $||\varepsilon_f||^2$,

$$\begin{aligned} ||\varepsilon_f||^2 &\simeq \frac{1}{n}\sum_{i=1}^{n} ||\phi(f(\mathbf{x}_i)) - \phi_r(f(\mathbf{x}_i))||^2, \\ &= \tilde{k}(f(\mathbf{x}_i), f(\mathbf{x}_i)) + \mathrm{E}\left[\mathbf{C}_r(\mathbf{x}_i)\right]^T \mathrm{E}\left[\mathbf{C}_r(\mathbf{x}_i)\right] - 2\mathrm{E}\left[\mathbf{C}_r(\mathbf{x}_i)\right]^T \mathbf{A}\tilde{\mathbf{K}}_{f(\mathbf{x}_i)}, \end{aligned} \tag{4.65}$$

where $\mathbf{A}$ is the matrix containing the first r eigenvectors of the centred kernel matrix, $\tilde{\mathbf{K}}$, and

$$\tilde{\mathbf{K}}_{f(\mathbf{x})} = [\tilde{k}(f(\mathbf{x}),\ f(\mathbf{x}_1)),\ \tilde{k}(f(\mathbf{x}), f(\mathbf{x}_2)),\ \ldots,\ \tilde{k}(f(\mathbf{x}), f(\mathbf{x}_n))].$$

The proof of equation (4.64) are given in Appendix A.2.

Thus, the expectation and variance of $\|\ \phi(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right]\ \|^2$ can be computed if we have the expectation and variance $\|\ \mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}\left[\mathbf{C}_r(\mathbf{x}^*)\right]\ \|^2$. $\mathbf{C}_r(x)$ is given by Gaussian process emulators,

$$\mathbf{C}_r(\mathbf{x}) - \mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right] \sim N(0,\ \mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right]).$$

For univariate emulators built for each coefficient individually, $\mathrm{Var}\left[\mathbf{C}_r(x)\right])$ is a diagonal $r \times r$ matrix. For each coefficient,

$$\begin{aligned} &\frac{(C_k(\mathbf{x}) - \mathrm{E}\left[C_k(\mathbf{x})\right])^2}{\mathrm{Var}\left[C_k(\mathbf{x})\right]} \sim \chi_1^2, \\ &\Rightarrow (C_k(\mathbf{x}) - \mathrm{E}\left[C_k(\mathbf{x})\right])^2 \sim \mathrm{Var}\left[C_k(\mathbf{x})\right] \chi_1^2, \end{aligned}$$

where χ_1^2 is the chi-squared distribution with 1 degree of freedom, and $\mathrm{E}\left[\chi_1^2\right] = 1$ and $\mathrm{Var}\left[\chi_1^2\right] = 2$. The expectation and variance of $(C_k(\mathbf{x}) - \mathrm{E}[C_k(\mathbf{x})])^2$ are therefore

$$\mathrm{E}\left[(C_k(\mathbf{x}) - \mathrm{E}[C_k(\mathbf{x})])^2\right] = \mathrm{Var}[C_k(\mathbf{x})],$$

and

$$\mathrm{Var}\left[(C_k(\mathbf{x}) - \mathrm{E}[C_k(\mathbf{x})])^2\right] = 2\mathrm{Var}[C_k(\mathbf{x})]^2.$$

So that the expectation and variance of $\| \mathbf{C}_r(\mathbf{x}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})] \|^2$ are

$$\mathrm{E}\left[\| \mathbf{C}_r(\mathbf{x}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})] \|^2\right] = \sum_{k=1}^{r} \mathrm{Var}[C_k(\mathbf{x})], \tag{4.66}$$

$$\mathrm{Var}\left[\| \mathbf{C}_r(\mathbf{x}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})] \|^2\right] = 2\left(\sum_{k=1}^{r} \mathrm{Var}[C_k(\mathbf{x})]\right)^2. \tag{4.67}$$

Hence, the threshold, $T(\mathbf{x})$, given in equation (4.59) (with t=3) can be computed

$$\begin{aligned} T(\mathbf{x}) = &\sum_{k=1}^{r} \mathrm{Var}[C_k(\mathbf{x})] + \mathrm{E}\left[\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2\right] + ||\varepsilon_f||^2 \\ &+ 3\sqrt{2\left(\sum_{k=1}^{r} \mathrm{Var}[C_k(\mathbf{x})]\right)^2 + \mathrm{Var}\left[\| \phi(z) - \phi(f(\mathbf{x}^*)) \|^2\right]}. \end{aligned} \tag{4.68}$$

We have well defined implausibility measure $\mathcal{I}_D(\mathbf{x})$, the distance between mapped observation and model output in feature space, and an $\mathbf{x}$-dependent threshold, $T(\mathbf{x})$, that accounts for all sources of uncertainty. If $\mathcal{I}_D(\mathbf{x})$ is larger than $T(\mathbf{x})$ for some $\mathbf{x}$, we are confident that the computer model output is not consistent with the observation. The NROY space which contains all the remaining parameter space is defined as

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_D(\mathbf{x}) \leq T(\mathbf{x})\}. \tag{4.69}$$

However, as we discussed in Section 4.1, when the key features in the observations we want to replicate (e.g. a current like a gulf stream in an ocean model) exist but in a different part of the output space, it's hard to put the discrepancy on the output space. In fact, the discrepancy may not belong to the output space, that's the reason the pattern can move and standard history matching failed. Therefore, the discrepancy may need to be determined in the feature space, and the best way to put it here is through the kernel function itself.

4.6 Kernel-based history matching

4.6.1 Capturing uncertainty through the kernel functions

Kernel functions can be seen as the transformation from simulator output space to feature space, which carries all the information from the simulator output space to feature space. Uncertainty in the simulator output space is part of that information and so could be captured by the kernel. Kernel functions are sometimes also known as similarity functions, as they represent the similarity between the mapped data $\phi(f(\mathbf{x}))$ and $\phi(f(\mathbf{x}'))$ evaluated at two model outputs $f(\mathbf{x})$ and $f(\mathbf{x}')$. The kernel function describes the spatial or temporal covariance in the model output space, and the kernel matrix can quantify how similar or dissimilar different members of the ensemble are. Hence, rather than projecting uncertainties onto feature space via a kernel function as discussed in the last subsection, another approach is to use the kernel to represent key uncertainties.

To quantify uncertainties through the kernel, the most important requirement is that the observational and structural uncertainties must be captured. We extend our kernel to included these uncertainties in the following way. We use an $l \times l$ weight matrix, Υ, to reflect judgements regarding what are key regions of output space for matching observations and which are less important. For example, the Gaussian kernel with weight matrix Υ becomes

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \exp(-(f(\mathbf{x}) - f(\mathbf{x}'))^T \Upsilon^{-1} (f(\mathbf{x}) - f(\mathbf{x}'))/\sigma),$$

where σ is a vector of kernel parameter and the homogeneous linear kernel becomes

$$k(f(\mathbf{x}), f(\mathbf{x}^{'})) = f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}^{'}).$$

By setting Υ as the sum of the observation error and discrepancy variances, we ensure that these key uncertainties translate into feature space so that in regions of output space where we are most uncertain, the feature distance is smaller. Any calibration within this feature space naturally then includes discrepancy and observation error when looking at L_2 distance in feature space. Fundamentally, history matching looks to rule out models that are 'far' from observation using a distance metric that accounts for all sources of uncertainty. Placing observation uncertainty and discrepancy into a kernel can be seen as a natural generalisation of this idea. Note, whilst it is perhaps natural to place observation error and some sources of discrepancy at Υ, other structural errors (related to shifting patterns) cannot be captured like this. The kernel structural itself will account for these types of discrepancy, and we will revist the notion of what discrepancy actually means in kernel-based history matching in Chapter 5. Moreover, how to choose a suitable kernel (kernel parameters), and how to specify the discrepancy variance when it is not available, will be discussed in Chapter 5.

4.6.2 Implausibility for kernel-based history matching

The implausibility shown in equation (4.43), is defined to measure the distance between observations and the simulator outputs at a given input in feature space,

$$\mathcal{I}(\mathbf{x}) = ||\phi(z) - \phi(f(\mathbf{x}))||_f,$$

where $\phi(z)$ is the mapped observation, $\phi(f(\mathbf{x}))$ is the mapped simulator output in feature space and $||\cdot||_f$ represents an appropriate measure that accounts for all sources of uncertainties.

By having uncertainties in the kernel functions, when emulators are not required to represent the model, we use Euclidean norm, giving

$$\begin{aligned}\mathcal{I}_{F0}(\mathbf{x}) &= (\phi(z) - \phi(f(\mathbf{x})))^T (\phi(z) - \phi(f(\mathbf{x}))) \\ &= k(z,z) + k(f(\mathbf{x}), f(\mathbf{x})) - 2k(f(\mathbf{x}), z).\end{aligned} \tag{4.70}$$

When the simulator output is 'close to' the observations (given all the uncertainties on simulator output space), the values of $k(z,z)$, $k(f(\mathbf{x}), f(\mathbf{x}))$ and $k(f(\mathbf{x}), z)$ should be similar, so that the expectation of the squared distance between the mapped observation and the simulator output should be small.

When we require an emulator to evaluate $\mathrm{E}[\phi(f(\mathbf{x}))]$, we must embed emulator uncertainty within the norm $||\cdot||_f$. To achieve that, we investigate two different ways of doing history matching in feature space with emulators (when other uncertainties are embedded in the kernel). The first way we propose is to define a new implausibility, $\mathcal{I}_{F1}(\mathbf{x})$, using the Euclidean distance between the observations and the expectation of reconstructed data in feature space, and setting the threshold T as a function of $\mathbf{x}$ that accounts for emulator uncertainty (this is similar to our method in Section 4.5.1). The second idea is to define a new distance function for implausibility, $\mathcal{I}_{F2}(\mathbf{x})$, which embeds emulator uncertainty within the distance calculation. The following subsections detail these two approaches.

4.6.3 Implausibility $\mathcal{I}_{F1}(\mathbf{x})$: variable cut-off thresholds

Emulators are built for the coefficients on the first r kernel PCA basis vectors as described in Section 4.2.3, with the emulator expectation for each of the r basis coefficients $\mathrm{E}[\mathbf{C}_r(\mathbf{x})]$, and the associated emulator variance matrix, $\mathrm{Var}[\mathbf{C}_r(\mathbf{x})]$. We give the expressions for $\mathrm{E}[\phi(f(\mathbf{x}))]$ and $\mathrm{Var}[\phi(f(\mathbf{x}))]$ in terms of the coefficient emulators in equations (4.26) and (4.27). If $\mathrm{Var}[\mathbf{C}_r(\mathbf{x})] = 0$, we can use $\mathrm{E}[\phi(f(\mathbf{x}))]$ instead of $\phi(f(\mathbf{x}))$ in equation (4.70). The implausibility in feature space would then be:

$$\mathcal{I}_{F1}(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]). \tag{4.71}$$

$\mathcal{I}_{F1}(\mathbf{x})$ has the same form as $\mathcal{I}_D(\mathbf{x})$ in equation (4.57), so that

$$\mathcal{I}_{F1}(\mathbf{x}) = \tilde{k}(z,z) + \mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]^T \mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right] - 2\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]^T \mathbf{A}\tilde{\mathbf{K}}_z. \tag{4.72}$$

However, even though the form of $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_D(\mathbf{x})$ is the same, the meaning of these two implausibilities is different. In Section 4.5.1, we introduced $\mathcal{I}_D(\mathbf{x})$ as the distance between observations and emulator predictions in feature space without considering any sources of uncertainty. However, the implausibility $\mathcal{I}_{F1}(\mathbf{x})$ accounts for discrepancy and observation error by having these uncertainties in the kernel. When there is no emulator uncertainty, large values of $\mathcal{I}_{F1}(\mathbf{x})$ indicate that it is implausible that the output of the computer model at $\mathbf{x}$ is consistent with $\phi(z)$. However, emulator uncertainty is not ignorable in reality. In later waves of history matching, the emulator variance will reduce, but will rarely reach zero. To make kernel-based history matching meaningful with $\mathcal{I}_{F1}(\mathbf{x})$, we give a threshold function dependent on the emulator variance below.

Threshold choice for kernel-based history matching with $\mathcal{I}_{F1}(\mathbf{x})$

To set a cutoff threshold for $\mathcal{I}_{F1}(\mathbf{x})$, we adopt the same approach introduced in Section 4.5.1. We define the threshold as a function of $\mathbf{x}$, $T(\mathbf{x})$, to account for the emulator uncertainty. Equation (4.58) shows that there is an upper bound, $L(\mathbf{x}^*)$, of $\mathcal{I}_D(\mathbf{x}^*)$ at the best input $\mathbf{x}^*$. Similar to equation (4.58), we use the triangle inequality to prove that there is a upper bound, $Q(\mathbf{x}^*)$, of the implausibility $\mathcal{I}_{F1}(\mathbf{x}^*)$ at the best input $\mathbf{x}^*$,

$$\begin{aligned}
\mathcal{I}_{F1}(\mathbf{x}^*) &= \| \phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \|^2 \\
&= \| \phi(z) - \phi_r(f(\mathbf{x}^*)) + \phi_r(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \|^2 \\
&\leq \| \phi(z) - \phi_r(f(\mathbf{x}^*)) \|^2 + \| \phi_r(f(\mathbf{x}^*)) - \mathrm{E}\left[\phi(f(\mathbf{x}^*))\right] \|^2 \\
&= \| \phi(z) - \phi_r(f(\mathbf{x}^*)) \|^2 + \| \mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}\left[\mathbf{C}_r(\mathbf{x}^*)\right] \|^2 \\
&\leq a + \| \mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}\left[\mathbf{C}_r(\mathbf{x}^*)\right] \|^2 \\
&= Q(\mathbf{x}^*),
\end{aligned} \tag{4.73}$$

where a is an upper bound for $\| \phi(z) - \phi_r(f(\mathbf{x}^*)) \|^2$. Note that unlike Section 4.5.1, $\| \phi(z) - \phi_r(f(\mathbf{x}^*)) \|^2$ is not used to account for discrepancy and observation error. Instead of computing the expectation and variance of $\| \phi(z) - \phi_r(f(\mathbf{x}^*)) \|^2$, we find an upper bound, a, of it. Following the threshold introduced in Section 4.4.3, we can use expert judgement to set a. For the ensemble F, we have the acceptable runs, F^A, and unacceptable runs, F^U, given by experts. To retain all of the acceptable inputs, $\mathbf{x}^A$, in the NROY space and rule out all of the unacceptable runs input, $\mathbf{x}^U$, we have

$$a = \min(\| \phi(\mathrm{z}) - \phi_{\mathrm{r}}(\mathrm{f}(\mathbf{x}^{\mathrm{U}})) \|^2). \tag{4.74}$$

Other suggestions of the value of a will be discussed in Chapter 5. Therefore, we establish the upper bound $Q(\mathbf{x}^*)$ for $\mathcal{I}_{F1}(\mathbf{x}^*)$ at the best input,

$$Q(\mathbf{x}^*) = \min(\mathcal{I}_{\mathrm{F1}}(\mathbf{x}^{\mathrm{U}})) + \| \mathbf{C}_{\mathrm{r}}(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_{\mathrm{r}}(\mathbf{x}^*)] \|^2 . \tag{4.75}$$

We use the same approach introduced in Section 4.5.1: set the $\mathbf{x}$-dependent threshold, $T(\mathbf{x})$, using the expectation of $Q(\mathbf{x}^*)$ and variance of $Q(\mathbf{x}^*)$,

$$T(\mathbf{x}) = \mathrm{E}[Q(\mathbf{x}^*)] + 3\sqrt{\mathrm{Var}[Q(\mathbf{x}^*)]}. \tag{4.76}$$

Hence, all sources of uncertainty on the simulator output space are accounted for in the implausibility, and the emulator uncertainty for each input, $\mathbf{x}$, is used in the threshold $T(\mathbf{x})$. The expectation and variance of $\| \mathbf{C}_r(\mathbf{x}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})] \|^2$ are given in equations (4.66) and (4.67), where

$$\mathrm{E}\left[\| \mathbf{C}_r(\mathbf{x}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})] \|^2\right] = \sum_{k=1}^{r} \mathrm{Var}[C_k(\mathbf{x})], \tag{4.77}$$

$$\mathrm{Var}\left[\| \mathbf{C}_r(\mathbf{x}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})] \|^2\right] = 2\left(\sum_{k=1}^{r} \mathrm{Var}[C_k(\mathbf{x})]\right)^2 . \tag{4.78}$$

Hence, the threshold becomes

$$T(\mathbf{x}) = \sum_{k=1}^{r} \mathrm{Var}\left[C_k(\mathbf{x})\right] + 3\sqrt{2\left(\sum_{k=1}^{r} \mathrm{Var}\left[C_k(\mathbf{x})\right]\right)^2} + a. \tag{4.79}$$

As the variance of the emulator tends to 0, the distance between simulator outputs and emulator predictions tends to zero, so that the threshold function $T(\mathbf{x})$ will be a constant:

$$T(\mathbf{x}) = a = \min(\| \phi(\mathrm{z}) - \phi_{\mathrm{r}}(\mathrm{f}(\mathbf{x}^{\mathrm{U}})) \|^2) \qquad \text{as} \qquad \mathrm{Var}\left[\phi(\mathrm{f}(\mathbf{x}))\right] \longrightarrow 0.$$

4.6.4 Implausibility $\mathcal{I}_{F2}(\mathbf{x})$

Instead of using the euclidean norm, we can define a new measure for the implausibility in equation (4.80),

$$\mathcal{I}_{F2}(\mathbf{x}) = \left(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]\right)^T \left(\mathbf{1}_D + \mathrm{Var}\left[\phi(f(\mathbf{x}))\right]\right)^{-1} \left(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]\right), \tag{4.80}$$

where $\phi(z)$ and $\mathrm{E}\left[\phi(f(\mathbf{x}))\right]$ are vectors of length of D (D is unknown), $\mathbf{1}_D$ is the identity matrix of dimension $D \times D$ and $\mathrm{Var}\left[\phi(f(\mathbf{x}))\right]$ is the emulator's $D \times D$ covariance matrix in feature space. $\mathcal{I}_{F2}$ represents the distance between the observation and model output in feature space scaled by the emulator uncertainty. The addition of the identity matrix $\mathbf{1}_D$ ensures that the implausibility tends to the L_2 distance in feature space as the emulator uncertainty tends to zero, i.e.

$$\mathcal{I}_{F2}(\mathbf{x}) \longrightarrow \mathcal{I}_{F1}(\mathbf{x}) \qquad \text{as} \qquad \mathrm{Var}\left[\phi(f(\mathbf{x}))\right] \longrightarrow 0.$$

Calculating implausibility $\mathcal{I}_{F2}(\mathbf{x})$

Without knowing D or the explicit form of mapping function $\phi(\cdot)$, we offer an efficient way to compute $\mathcal{I}_{F2}(\mathbf{x})$. As in Section 4.4.2, we define the 'coefficient implausibility', analogous to equation (4.80) in the subspace defined by the basis vector, $\mathbf{W}^r$, as:

$$\mathcal{I}_{C2}(\mathbf{x}) = \left(\mathbf{C}_r(z) - \mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]\right)^T \left(\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right] + \mathbf{1}_r\right)^{-1} \left(\mathbf{C}_r(z) - \mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]\right). \tag{4.81}$$

Every term in equation (4.81) is known, so that we can compute the value of $\mathcal{I}_{C2}(\mathbf{x})$ for any input $\mathbf{x}$ with emulator predictions. We show that the feature space implausibility measure $\mathcal{I}_{F2}(\mathbf{x})$ can be written in terms of $\mathcal{I}_{C2}(\mathbf{x})$,

$$\mathcal{I}_{F2}(\mathbf{x}) = \mathcal{I}_{C2}(\mathbf{x}) + ||\varepsilon_z||^2, \tag{4.82}$$

where ε_z is the observation reconstruction error and

$$\begin{aligned} ||\varepsilon_z||^2 &= (\tilde{\phi}(z) - \tilde{\phi}_r(z))^T (\tilde{\phi}(z) - \tilde{\phi}_r(z)) \\ &= \tilde{k}(z,z) + \mathbf{C}_r(\mathbf{x})^T \mathbf{C}_r(\mathbf{x}) - 2\mathbf{C}_r(\mathbf{x})^T \mathbf{A}\tilde{\mathbf{K}}_z. \end{aligned} \tag{4.83}$$

The proof is a generalisation of the work by Salter and Williamson (2019), which relies on the well-known Woodbury formula (Higham, 2002; Woodbury and Woodbury, 1950),

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1},$$

for matrices A, U, C and V. Expanding $\mathcal{I}_{F2}(\mathbf{x})$,

$$\begin{aligned} \mathcal{I}_{F2}(\mathbf{x}) &= (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right])^T (\mathbf{1}_D + \mathbf{W}^r\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right]\mathbf{W}^{rT})^{-1} (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]) \\ &= (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right])^T \left(\mathbf{1}_D^{-1} - \mathbf{1}_D^{-1}\mathbf{W}^r(\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right]^{-1} + \mathbf{W}^{rT}\mathbf{1}_D^{-1}\mathbf{W}^r)^{-1}\mathbf{W}^{rT}\mathbf{1}_D^{-1}\right) (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\right. \\ &= (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right])^T \left(\mathbf{1}_D - \mathbf{W}^r(\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right]^{-1} + \mathbf{W}^{rT}\mathbf{W}^r)^{-1}\mathbf{W}^{rT}\right) (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]) \\ &= (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right])^T \left(\mathbf{1}_D - \mathbf{W}^r(\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right]^{-1} + \mathbf{1}_r)^{-1}\mathbf{W}^{rT}\right) (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]) \\ &= (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right])^T (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]) \\ &- (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right])^T \left(\mathbf{W}^r(\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x})\right]^{-1} + \mathbf{1}_r)^{-1}\mathbf{W}^{rT}\right) (\phi(z) - \mathbf{W}^r\mathrm{E}\left[\mathbf{C}_r(\mathbf{x})\right]). \end{aligned}$$

Applying the Woodbury formula again to the term of $(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]^{-1}+\mathbf{1}_D)^{-1}$, we have:

$$\begin{aligned}(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]^{-1}+\mathbf{1}_r)^{-1} &= \mathbf{1}_r^{-1}-\mathbf{1}_r^{-1}(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]+\mathbf{1}_r^{-1})^{-1}\mathbf{1}_r^{-1}\\ &= \mathbf{1}_r-(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]+\mathbf{1}_r)^{-1}.\end{aligned}$$

Therefore,

$$\begin{aligned}\mathcal{I}_{F2}(\mathbf{x}) &= (\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])\\ &\quad -(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T\left(\mathbf{W}^r(\mathbf{1}_r-(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]+\mathbf{1}_r)^{-1})\mathbf{W}^{rT}\right)(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})]).\\ &= (\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])\\ &\quad -(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T\mathbf{W}^r\mathbf{W}^{rT}(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])\\ &\quad +(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T\mathbf{W}^r(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]+\mathbf{1}_r)^{-1}\mathbf{W}^{rT}(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})]).\\ &= (\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])\\ &\quad -(\mathbf{W}^{rT}\phi(z)-\mathbf{W}^{rT}\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{W}^{rT}\phi(z)-\mathbf{W}^{rT}\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])\\ &\quad +(\mathbf{W}^{rT}\phi(z)-\mathbf{W}^{rT}\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]+\mathbf{1}_r)^{-1}(\mathbf{W}^{rT}\phi(z)-\mathbf{W}^{rT}\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})]).\\ &= (\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])\\ &\quad -(\mathbf{C}_r(z)-\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{C}_r(z)-\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})]),\\ &\quad +(\mathbf{C}_r(z)-\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\mathrm{Var}\,[\mathbf{C}_r(\mathbf{x})]+\mathbf{1}_r)^{-1}(\mathbf{C}_r(z)-\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})]).\end{aligned}$$

For the first two terms, we have

$$(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\phi(z)-\mathbf{W}^r\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])-(\mathbf{C}_r(z)-\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{C}_r(z)-\mathrm{E}\,[\mathbf{C}_r(\mathbf{x})])=||\varepsilon_z||^2. \tag{4.84}$$

The details of the proof of equation (4.84) is given in Appendix A.4. The last term is the coefficient implausibility, $\mathcal{I}_{C2}(\mathbf{x})$, which we defined in equation (4.81). Hence, we have proved that

$$\mathcal{I}_{F2}(\mathbf{x})=\mathcal{I}_{C2}(\mathbf{x})+||\varepsilon_z||^2.$$

With the calculable $\mathcal{I}_{C2}(\mathbf{x})$ in equation (4.81) and $||\varepsilon_z||^2$ in equation (4.83), we can now efficiently calculate the implausibility in feature space, $\mathcal{I}_{F2}(\mathbf{x})$, without requiring the explicit form of the mapping function.

Threshold choice for kernel-based history matching with $\mathcal{I}_{F2}(\mathbf{x})$

We define the threshold for $\mathcal{I}_{F2}(\mathbf{x})$ using 'leave one out' diagnostics. For each ensemble member, 'leave one out' diagnostics remove one run from the ensemble, and an emulator is then built using the remaining data $(n-1)$, and the removed run is predicted by this emulator. This procedure is repeated for all runs, so that 'leave one out' diagnostics will build n emulators, and each emulator gives the predictive mean $\mathrm{E}[\mathbf{C}_r(\mathbf{x}_i)]$ with variance $\mathrm{Var}[\mathbf{C}_r(\mathbf{x}_i)]$ for the i-th removed run, where $i = 1, \ldots, n$.

Our expert judgement divides the ensemble into acceptable runs and unacceptable runs as discussed in Section 4.4.3 and 4.6.3. Suppose there are q acceptable runs. In the calibration, we want to set a threshold that can keep all of the q acceptable, $\mathbf{x}^A$, in the NROY space and rule out of all the unacceptable inputs, $\mathbf{x}^U$. To achieve this we define the threshold T' for $\mathcal{I}_{F2}(\mathbf{x})$ following the judgement given in Section 4.4.3, as

$$T' = \min(\mathcal{I}_{F2}(\mathbf{x}_j^U)). \tag{4.85}$$

For each $\mathbf{x}_j$, we compute the implausibility $\mathcal{I}_{F2}(\mathbf{x}_j)$ using $\mathrm{E}\left[\mathbf{C}_r(\mathbf{x}_j)\right]$ and $\mathrm{Var}\left[\mathbf{C}_r(\mathbf{x}_j)\right]$ evaluated by leaving out $\mathbf{x}_j$. By setting the threshold T' though equation (4.85), we could account for both the emulator variance and expert judgement in the calibration.

4.7 Refocusing

Refocusing for standard history matching has been introduced previously in Section 2.5.1. History matching in feature space should also be performed waves. For history matching in feature space with distance constraints (in Section 4.5), the process is the same as for standard history matching. But for history matching with projecting uncertainties into feature space (in Section 4.4) and kernel-based history matching (in Section 4.6), we require expert judgement for each wave. This suggests the following multi-wave process. First, we choose an initial ensemble design, $\mathbf{X}_1 = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X}$, where $\mathcal{X}$ is the p-dimensional simulator input space. Let f represent the complex computer model, and then the set of simulator ensemble runs at the initial design, $\mathbf{F}_1 = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) \in \mathcal{M}$, where $\mathcal{M}$ is the m-dimensional simulator output space. By comparing the ensemble, $\mathbf{F}_1$, with the observation directly, experts will give the calibration judgements about the simulator runs, which can be easily translated into information on which runs are and are not deemed acceptable for this wave only. $\mathbf{X}_1$ is then separated into "acceptable" and "unacceptable" runs, $\mathbf{X}_1^A$ and $\mathbf{X}_1^U$. After this judgement is obtained, we apply the feature space history matching in a higher D-dimensional feature space $\mathcal{F} \subset \mathbb{R}^D$, using either $\mathcal{I}_D(\mathbf{x})$, $\mathcal{I}_{F1}(\mathbf{x})$ or $\mathcal{I}_{F2}(\mathbf{x})$, with the threshold value/function chosen via equations (4.53), (4.79) or (4.85) to define NROY space, $\mathcal{X}^1$. For instance, using $\mathcal{I}_{F1}(\mathbf{x})$, the NROY space is defined as:

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_{F1}(\mathbf{x}) \leq T(\mathbf{x})\},$$

where $T(\mathbf{x})$ is given in equation (4.79).

Refocusing involves iterating this process multiple times. In general, at wave k, a new ensemble design, $\mathbf{X}_k$, is generated from $\mathcal{X}^{k-1}$. The NROY space for wave k with the implausibility $\mathcal{I}_{F1}(\mathbf{x})$ is defined as

$$\mathcal{X}^k = \{\mathbf{x} \in \mathcal{X}^{k-1} | \mathcal{I}_{F1}(\mathbf{x}) \leq T_k(\mathbf{x})\},$$

where $T_k(\mathbf{x})$ is defined in equation (4.79), but based on expert judgement for all of the training data until wave k.

It is important to note that the expert judgement step is applied for each wave, which might give a different threshold value/function for each wave. For instance, in the first wave, the design for a limited number of ensemble members may not contain many perfect runs, and the experts might have a large tolerance for the difference between the observation and ensemble runs (they may not be sure the model can get very close to observations). As more 'good' runs are found, the tolerance might be reduced in later waves, and so the threshold value/function (T, T' or $T(\mathbf{x})$) would be reduced. In addition, T' and $T(\mathbf{x})$ for kernel-based history matching are also affected by the emulator uncertainty. As later waves are reached, the density of runs increases and the emulator uncertainty is reduced. For $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$, when the emulator uncertainty tends towards zero, we have $\mathcal{I}_{F2}(\mathbf{x}) \longrightarrow \mathcal{I}_{F1}(\mathbf{x})$ and $T(\mathbf{x}) \longrightarrow T'$.

For a stopping rule to determine how many waves the kernel-based history matching performs, we following the standard history matching stopping criteria. The details are discussed in Section 2.5.1.

4.8 Numerical study

To illustrate the performance of history matching in feature space, we design an idealised toy example. The toy example is built to imitate a situation where PCA approaches break down, where (for example) resolution-dependent currents or signals may move around the output space, but it is more important that they exist in approximately the right place. Therefore, we construct a toy model which gives roughly two kinds of model outputs. The first kind of model output contains no key features of the observation; this kind of output could be constructed by random errors. For another portion of the model outputs, the key features exist, but in various locations in the model output space. This imitates a situation where, for some parameter settings a key emergent process is not activated, and for others it is active but perhaps not in exactly the same place as its real world counterpart due to the inherent limitations of the model.

We build our toy example with 5 input parameters $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_5$. The parameter space $\mathcal{X}$ is $[-1,1]^5$ and the example gives a 10×10 spatial output. The full definition of the toy model is given in Appendix C.1.

The observation z is obtained by running the toy example at $\mathbf{x}_z$ with an observation error added. $\mathbf{x}_z$ is chosen as $\mathbf{x}_z = (0.2, 0.2, 0.3, 0.4, 0.3)$, and the value of the observation error is sampled from the distribution of the observation error

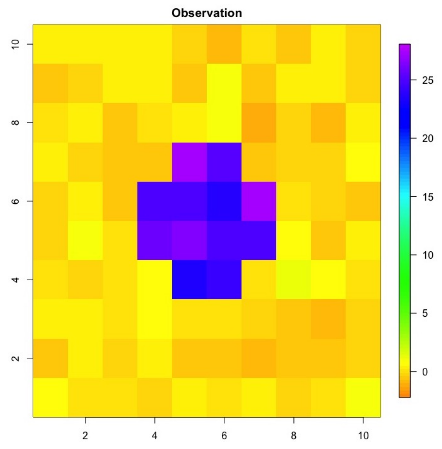


Fig. 4.3 The observations, z, for the toy function.

e, $N(0,\ \Sigma_e)$. To define Σ_e, we use a squared exponential correlation function as defined in equation (2.5), details given in Appendix C.1. The observation, z, is shown in Figure 4.3, the main feature of the observation is the purple/blue signal in the middle.

We sample 50 parameter settings, **X**, using a Latin Hypercube from the 5-dimensional parameter space, $\mathcal{X}$, giving an ensemble, **F**, with dimension 100×50. We plot the 50 ensemble members in order from the 1st run to the 50th in Figure C.1. As the constructor for the toy example, we could be viewed as the experts. As such, we define the model outputs with any part of the purple signal as acceptable runs: the acceptable runs F^A are the ensemble members 2, 9, 15, 18, 21, 22, 29, 36, 37, 38, 40, 43, 44 and 46. For some runs like ensemble member 5, which only contain a small part of the key pattern, we treat that part of the pattern as a noisy signal and these runs are deemed unacceptable.

In order to investigate the drawbacks of the standard history matching using PCA, a comparison between kernel-based history matching with standard history matching will presented. The limitation of standard history matching is presented in Section 4.8.2. Moreover, a comparison is given to illustrate the benefits and drawbacks of our approaches is given in Section 4.8.3. We first introduce the true NROY space for the toy example.

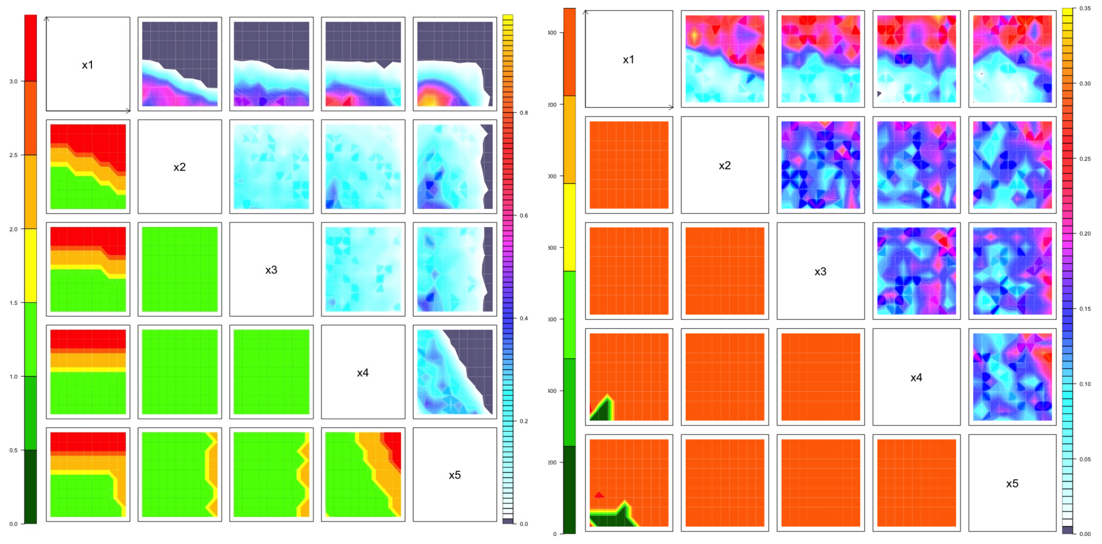


Fig. 4.4 *Left*: The true NROY density plots (upper triangle) and minimum implausibility plots (lower triangle). *Right*:Standard history matching $\mathcal{X}^*_{standard}$.

4.8.1 True NROY space

True NROY space is used to identify whether the model actually reproduces the observations, and is also used to judge the accuracy of the results of history matching. We have presented an approach to find true NROY space for the numerical examples in Chapter 3, by calculating the implausibility for each output directly, with no emulator variance. However, the true NROY space defined in this way cannot be used as a reference for different calibration methods, with implausibility measurements defined in different spaces and with different functions. We will show this in Section 4.8.2. Moreover, the full definition of the true NROY space for our method will be introduced in Section 5.3).

We define the true NROY space here using the structure of the toy example directly. We use 10000 runs sampled by a LHC design to represent the initial parameter space $\mathcal{X}$. We set the outputs with the key signal as "acceptable" runs, and the model outputs with no signal as unacceptable. Hence, the true NROY is constructed with all of the acceptable runs in these 10000 runs. We plot the target NROY space in the left panel of Figure 4.4 as a reference. True NROY is 14.34% of the original space.

4.8.2 The limitation of standard history matching

To illustrate the limitation of standard history matching clearly, we can apply standard history matching (or PCA-based standard history matching) for the toy example, and compare the NROY space generated by standard history matching with true NROY space to show the difference. However, the emulator uncertainties may influence the produced NROY space. To make a fair comparison, we find "true" NROY space of standard history matching for the toy example following the definition given in Chapter 3. We define the "true" NROY space for standard history matching, $\mathcal{X}^*_{standard}$, as the NROY space found by the computer model directly (without an emulator),

$$\mathcal{X}^*_{standard} = \{\mathbf{x} \in \mathcal{X} : (z - f(\mathbf{x}))^T (\Sigma_e + \Sigma_\eta)^{-1} (z - f(\mathbf{x})) \leq T\}, \tag{4.86}$$

where Σ_e is the observation error (already given with the observation) and Σ_η is the discrepancy. However, with the randomly moving signal, it's hard to specify the discrepancy on the output space. We set $\Sigma_\eta = \Sigma_e$ here, to make a fair comparison, the definition for Σ_η and Σ_e will be used in our methods as well. More discussion of Σ_η will be given later.

For multivariate history matching, the threshold T is generally set as 99.5% of a Chi-squared distribution with l degrees of freedom (Vernon et al., 2010). However, we find that such a threshold would nearly rule out all of the input space (we will discuss why later). To retain the same size (14.34%) of true NROY space, we achieve that by changing the threshold T. Standard history matching "true" NROY space can then be produced, $\mathcal{X}^*_{standard}$ is plotted in Figure 4.4 (right). By comparing standard history matching $\mathcal{X}^*_{standard}$ with true NROY density plots, we can see two opposing trends. The input space which has the highest density in the true NROY plot (Figure 4.4 (right)) have the lowest value of density in the plots for standard history matching NROY space (Figure 4.4 (left)).

The main reason for this is that the location of the key pattern in the ensemble is different with the observation. For model runs which identify the key pattern, but not in the same location as in observations, the distance in different locations will be counted twice, leading to a large implausibility. For model runs which do not have any patterns, and are relatively constant everywhere, the difference between the observation and the model output is because of the existence of the key pattern, and it will be only counted once, leading to a smaller implausibility than that of the good runs. These parameter choices with small values of implausibility are, in fact, those which need to be ruled out. This explains why, when we set the threshold based on the Chi-squared distribution, there are no parameter settings that were retained in the NROY space. The toy model shows that these types of simulators cannot be satisfactorily calibrated based on distance in the output field space. The issue is that we are using the wrong distance metric for comparing computer model runs to observations.

4.8.3 History matching in feature space

Three possible ways to perform history matching with kernel methods were presented in Sections 4.4, 4.5 and 4.6. For the third approach, kernel-based history matching, two possible ways to define an implausibility measurement are introduced. To make this comparison easier to follow and as a recap, we will call history matching with projected uncertainties (Section 4.4), using the coefficient implausibility

$$\mathcal{I}_c(\mathbf{x}) = (\mathbf{C}_r(\mathbf{z}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T (\mathrm{Var}[\mathbf{C}_r(\eta)] + \mathrm{Var}[\mathbf{C}_r(e)] + \mathrm{Var}[\mathbf{C}_r(\mathbf{x})])^{-1} (\mathbf{C}_r(\mathbf{z}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]),$$

as “method 1”. History matching with distance constraint (Section 4.5), using implausibility $\mathcal{I}_D(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])$, is called “method 2”. Kernel-based history matching is called “method 3” when it we use the implausibility $\mathcal{I}_{F1}(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])$, (note, $\mathcal{I}_D(\mathbf{x})$ and $\mathcal{I}_{F1}(\mathbf{x})$ are different due to the different kernel, as detailed in Section 4.6.3), and “method 4” when $\mathcal{I}_{F2}(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T(\mathbf{1}_D + \mathrm{Var}[\phi(f(\mathbf{x}))])^{-1}(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])$ is used.

To perform history matching in feature space, a kernel function is required. Up until now, we have not established a general rule as to which kernel should be used. Kernel selection will be introduced in Chapter 5. For this toy example, we use a Gaussian RBF kernel. For method 1 and method 2, we use

$$k_1(f(\mathbf{x}), f(\mathbf{x}')) = \exp(-\sigma(f(\mathbf{x}) - f(\mathbf{x}'))^T(f(\mathbf{x}) - f(\mathbf{x}')),$$

and for method 3 and method 4, we use

$$k_2(f(\mathbf{x}), f(\mathbf{x}')) = \exp(-\sigma(f(\mathbf{x}) - f(\mathbf{x}'))^T(\Sigma_e + \Sigma_\eta)^{-1}(f(\mathbf{x}) - f(\mathbf{x}'))),$$

where Σ_e is the observation error variance matrix, Σ_η is the discrepancy variance matrix, the specification for these two variances are same as before. For both kernels, we set $\sigma = 0.002$ (see chapter 5 for setting kernel parameters).

Given these kernel choices, we calculate the ensemble projections by applying the kernel PCA algorithm. For $k_1(f(\mathbf{x}), f(\mathbf{x}'))$, the algorithm suggests that 16 basis vectors are required to explain 95% of the ensemble variability. However, as we discussed in Section 4.2.3, no more than 5 basis vectors are adopted ((Higdon et al., 2008)). For $k_2(f(\mathbf{x}), f(\mathbf{x}'))$, only 4 basis vectors are required to explain 90% of the ensemble variability. We use the Bayesian Treed Gaussian Process method with the R package `tgp` to build the emulators for the coefficients (Gramacy and Lee, 2012) (emulation diagnostics for the toy model are provided in Appendix C.1). The data are hard to predict because of the discontinuous nature of the simulator, which give large uncertainties for both kernel emulators. To test which kernel function interprets the key feature better, we perform history matching.

Using the emulator predictions (given in Appendix C.1), we perform history matching in feature space with our four approaches. The NROY space for four methods are plotted in Figure 4.5. To clearly show the difference of the NROY spaces, we list the results in Table 4.1. For method 1 and method 2, with the same kernel and emulation, the retained volume of NROY space and retained true NROY are different. In general, these two methods work for the toy model,

	NROY volume	Retained true NROY
Method 1	45.36%	92.31%
Method 2	50.34%	99.11%
Method 3	43.72%	100%
Method 4	41.81%	100%

Table 4.1 History matching results of four methods.

	Implausibility	Fixed vs variable threshold	Where 'discrepancy and observation error' defined	Where 'emulator uncertainty ' accounted for
Method 1	$\mathcal{I}_c(\mathbf{x}) = (\mathbf{C}_r(\mathbf{z}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T$ $(\mathrm{Var}[\mathbf{C}_r(\eta)] + \mathrm{Var}[\mathbf{C}_r(e)] + \mathrm{Var}[\mathbf{C}_r(\mathbf{x})])^{-1}$ $(\mathbf{C}_r(\mathbf{z}) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])$	Fixed threshold	Implausibility	Implausibility
Method 2	$\mathcal{I}_D(\mathbf{x}) = \|\|\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]\|\|^2$	Variable threshold	Threshold	Threshold
Method 3	$\mathcal{I}_{F1}(\mathbf{x}) = \|\|\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]\|\|^2$	Variable threshold	Kernel function	Threshold
Method 4	$\mathcal{I}_{F2}(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T$ $(\mathbf{1}_D + \mathrm{Var}[\phi(f(\mathbf{x}))])^{-1}$ $(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])$	Fixed threshold	Kernel function	Implausibility

Table 4.2 Comparison between four methods.

however, there is still around 8% true NROY ruled out for method 1. This is an example of limitation of method 1 that was discussed in Section 4.4.4: perform history matching on the coefficient space with the projected discrepancy and observation error might be biased.

To compare these four approaches, we list the difference in Table 4.2. The implausibility for method 2, $\mathcal{I}_D(\mathbf{x})$, has the same expression as the implausibility for methods 3, $\mathcal{I}_{F1}(\mathbf{x})$, but the implied meaning is different due to the different kernel. The results show that method 3 ruled out more space than method 2 with all of the true NROY space retained. For the NROY plots for method 3 and method 4, we can see there is an area with smaller minimum implausibility than the other NROY spaces that is consistent with true NROY space (Figure 4.4). The plots for method 3 and 4 are most similar in 2D pattern to true NROY.

For method 3 and method 4, as discussed, when the emulator uncertainty vanishes, the expression for $\mathcal{I}_{F1}(\mathbf{x})$ is the same as the expression for $\mathcal{I}_{F2}(\mathbf{x})$, and the threshold T and threshold function $T(\mathbf{x})$ will also be the same. Hence, without considering emulator uncertainties, kernel-based history matching with $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$ will give the same NROY space. In other words, the difference between $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$ is entirely caused by the differences in how we use the emulator

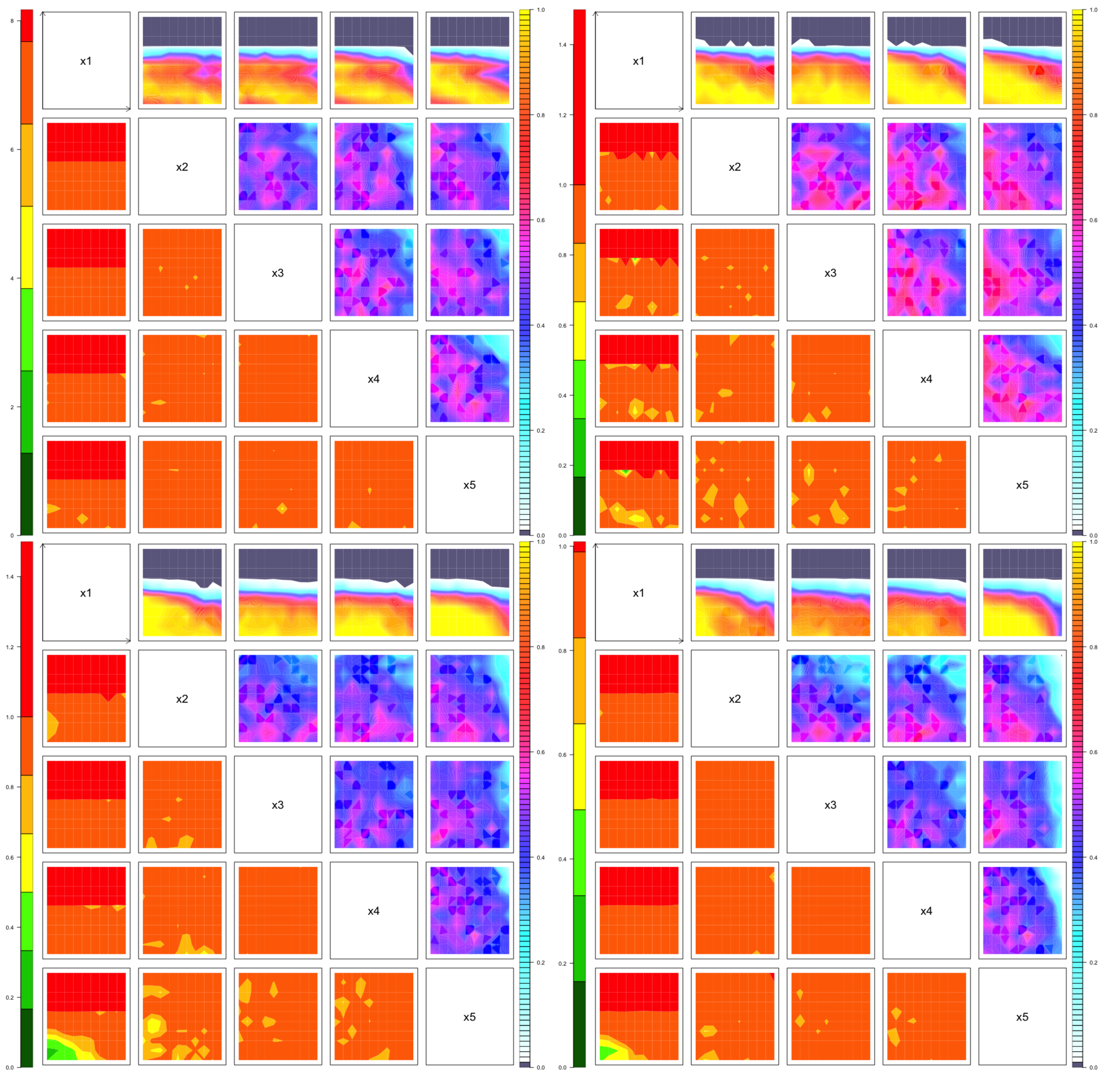


Fig. 4.5 *Top left*: Method 1 NROY space calculated by implausibility $\mathcal{I}_c(\mathbf{x})$. *Top right*: Method 2 NROY space calculated by implausibility $\mathcal{I}_D(\mathbf{x})$. *Bottom left*: Method 3 NROY space calculated by implausibility $\mathcal{I}_{F1}(\mathbf{x})$. *Bottom right*: Method 4 NROY space calculated by implausibility $\mathcal{I}_{F2}(\mathbf{x})$.

uncertainties. In this toy example, there is no big difference between method 3 and method 4.

For the first 2 methods, discrepancy is required to be defined first in the output space, and we then project it onto feature space. In fact, the discrepancy may not belong to the output space, that's the reason the pattern can move and standard history matching failed. The performance of history matching in feature space can be better once we set the discrepancy in feature space through the kernel, and this can be achieved by method 3 and method 4 (kernel-based history matching) only. Therefore, we suggest that kernel-based history matching is the best way to perform history matching in feature space. In the following chapters, we will refer it as KHM. In Chapter 5, we explore further development for KHM. We address the key question of how to select kernels, specify discrepancy and to choose implausibility thresholds. Refocusing of KHM will be demonstrated in the numerical study in Chapter 5.

4.9 Discussion

In this chapter, we enhanced history matching using machine learning approaches. A kernel-based history matching method is proposed to perform history matching in feature space. The idea of kernel-based history matching is based on the pre-image reconstruction of kernel PCA. We define the implausibility as the distance between the mapped observation and the mapped model outputs in feature space, so that we can search model output space for key features which are consistent with observations in the feature space.

We demonstrated the accuracy of kernel-based history matching in comparison to standard history matching based on PCA. By applying standard history matching to a toy example, we showed the limitations of current calibration methods: standard history matching cannot calibrate a computer model with high-dimensional output that contains moving patterns. To overcome this limitation, we introduced four different approaches to define the implausibility in feature space with all sources of uncertainty accounted for in different ways. By trialing our proposed methods on the toy example using a comparison study, we showed that kernel-based history matching in feature space can effectively cut parameter space whilst ensuring the true NROY space is preserved.

Unlike standard history matching, where discrepancy and observation error are accounted for in the implausibility function, kernel-based history matching captures all of the judgements in the model output space though the kernel. By projecting the model output and observations with this kernel, model output (with moving patterns) can be compared with the observations in the feature space. In this chapter, we claimed that placing observation uncertainty and discrepancy into a kernel is a natural generalisation of standard history matching. However, as we found in the numerical example, it's hard to define the discrepancy on the output space when the location of the signal is unfixed. How to define the discrepancy, and what discrepancy actually means in kernel-based history matching will be discussed in Chapter 5.

Given an ensemble, we typically do not know which kernel may work best. For a linear kernel, kernel PCA is exactly equivalent to PCA, and feature space would be the same as the model output space. It is crucial that we choose a suitable kernel for each individual application. To avoid choosing a wrong kernel, an automatic procedure for kernel optimization is proposed in the Chapter 5.

Chapter 5

Optimal kernel selection in kernel-based history matching

5.1 Introduction

The performance of kernel methods (e.g. support vector machine, kernel ridge regression and kernel PCA) relies on the choice of the kernel and its parameters (Alam and Fukumizu, 2014; Debnath and Takahashi, 2004; Kim et al., 2006). For kernel-based history matching (KHM), the kernel function can be seen as a measure of similarity for model outputs, and provides a bridge from the model output space to a feature space, where models can be compared with data, as described in Chapter 4. The choice of the kernel and its parameters will affect whether non-linear features in the initial data are extracted and can be evaluated. It is therefore important and necessary to construct a suitable kernel for specific data features. The toy example in the last chapter shows that contradictory results can be produced by performing history matching in different feature spaces (the simulator output space is equivalent to a feature space determined by a linear kernel, and was compared to a feature space determined by a Gaussian kernel). Hence, as with other kernel based methodologies, the efficiency of KHM is highly sensitive to the selection of the kernel.

To investigate suitable kernels and their parameters for history matching, we start by examining optimisation-based methods for kernel selection within the machine learning literature. The general approach for kernel selection is cross-validation (e.g., a bandwidth of a Gaussian RBF kernel for SVM (Chang et al., 2005)), with a learning objective function. The idea behind cross-validation is to separate the data into training data and test data, use the training data to run the algorithm, and test the data to check the performance (Camps-Valls et al., 2004). In choosing the hyperparameters of kernel PCA, Alam and Fukumizu (2014) proposed an approach based on cross-validation for comparing reconstruction errors of pre-images of kernel PCA. The pre-image of a feature vector, $\hat{f}$, is defined by an approximate inverse image of the feature map, $\phi(f(\mathbf{x}))$ (where $\phi(f(\mathbf{x}))$ is usually given by its reconstruction $\phi(f(\mathbf{x})) = \phi_r(f(\mathbf{x}))$, details are explained in Section 4.2.5). The reconstruction error of pre-images for $f(\mathbf{x})$, is defined as,

$$\epsilon_{re} = \| \phi(\hat{f}) - \phi_r(f(\mathbf{x})) \|^2 . \tag{5.1}$$

By minimising ϵ_{re}, the parameters in a kernel and the number of kernel principal components can be determined. However, a drawback of these cross-validation approaches is that they are not directly applicable for kernel PCA: the objective function given in equation (5.1) is on feature space which is determined by the kernel, thus the cross-validation errors are not comparable for different kernels. For instance, the maximum value of ϵ_{re} is 1 for the Gaussian kernel. However, the maximum value for polynomial kernels and linear kernels can be significantly larger. These kernel functions properties imply that the Gaussian kernel would be more likely to produce a smaller value ϵ_{re} than other kernels.

Other existing investigations into the selection of the kernel function are limited to the improvement of the existing common kernels introduced in Section 4.2.1 (Amari and Wu, 1999; Ayat et al., 2005; Vicente et al., 2017). This causes the limitation we discussed above: performance evaluations are not comparable for different kernels. In response to this, Smits and Jordaan (2002) developed a novel method of constructing a kernel function. A mixture kernel is used for improving

SVM regression and can be expressed as:

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \sum_{i=1}^{m} \omega_i k_i(f(\mathbf{x}), f(\mathbf{x}')), \tag{5.2}$$

where ω_i is an optimal mixing coefficient and $k_i(f(\mathbf{x}), f(\mathbf{x}'))$ are kernel functions. Shi et al. (2009) propose an optimised kernel PCA for fault detection based on mixture kernels. In this chapter, we define a specific mixture kernel: a combination of a non-linear kernel and a linear kernel for history matching. The observation error and discrepancy (where known) are included in the new defined kernel function (details will be given in Section 5.2.2). By changing the mixing coefficient, ω, we can achieve standard history matching with KHM. Thus, our mixture kernel will ensure that KHM is a generalisation of standard history matching.

Before implementing history matching, all of the unknown kernel parameters must be optimised. To find the optimal parameters, a performance evaluation to assess the accuracy and efficiency of the calibration results is required. There are no existing performance evaluations for history matching yet. We therefore investigate the use of expert judgement (the labelling of acceptable and unacceptable members of our training set, as detailed in Section 4.4.3) and the newly defined implausibility in feature space to optimise the parameters of our mixture kernel function.

Our cost function trades off two measurements: accuracy and efficiency. Accuracy aims to retain all of the acceptable runs and efficiency aims to rule out all of the unacceptable runs when carrying out history matching with the chosen kernel. By combining these two measurements into a score, we can optimise our chosen kernel for history matching. The implausibility cutoff threshold for kernel based history matching is a main factor in determining performance. For standard history matching, the cutoff threshold can be set as a static value based on the 3 sigma rule, or the quantiles of a Chi-squared distribution (for example). However, unlike with standard history matching, the value of the cutoff threshold for KHM must be influenced by the kernel function because kernel choices impact implausibility magnitudes. Therefore, we set the threshold as a function of the kernel, and an optimisation cutoff is given for the kernel in our optimisation algorithm.

This chapter is structured as follows: In Section 5.2, we introduce our mixture kernel. Specifically, in Section 5.2.3 we demonstrate that standard history matching can be achieved by KHM with a linear kernel. In Section 5.3, we introduce an optimisation algorithm for kernel selection. In Section 5.4 and Section 5.5, we apply the KHM method to two numerical examples. Section 5.6 concludes with a discussion.

5.2 A mixture kernel for kernel PCA

5.2.1 Kernel properties

Before constructing a mixture kernel that combines the linear kernel with a non-linear kernel, we will first formally introduce the properties associated with kernels. A kernel function must be symmetric, $k(\mathbf{x},\mathbf{x}') = k(\mathbf{x}',\mathbf{x})$, and strictly non negative: $k(\mathbf{x},\mathbf{x}') \geq 0$. Kernel functions must also satisfy the Cauchy-Schwartz inequality, $k(\mathbf{x},\mathbf{x}') \leq \sqrt{k(\mathbf{x},\mathbf{x})k(\mathbf{x}',\mathbf{x}')}$ (Genton, 2001).

Since kernels are positive semi-definite functions, we can construct new kernel functions from existing kernels. For example, the linear combination of two kernel functions k_1 and k_2 is also a kernel:

$$k(x,x') = a_1 k_1(x,x') + a_2 k_2(x,x'), \tag{5.3}$$

where a_1 and a_2 are two positive constants.

5.2.2 The structure of the mixture kernel

This study uses the structure of the mixture kernel introduced by Smits and Jordaan (2002), which is constructed following equation (5.3). We define the mixture kernel, $k(f(\mathbf{x}), f(\mathbf{x}'))$, as a combination of two kernels, $k_1(f(\mathbf{x}), f(\mathbf{x}'))$ and $k_2(f(\mathbf{x}), f(\mathbf{x}')$, via

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \omega k_1(f(\mathbf{x}), f(\mathbf{x}')) + (1-\omega) k_2(f(\mathbf{x}), f(\mathbf{x}')), \tag{5.4}$$

where $\omega \in [0,1]$ is a weight parameter. To maintain the ability to use both linear and non-linear kernels, we combine a linear kernel with a user-specified non-linear kernel. We set $k_1(f(\mathbf{x}), f(\mathbf{x}'))$ as a linear kernel,

$$k_1(f(\mathbf{x}), f(\mathbf{x}')) = f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}'),$$

where Υ is a $l \times l$ weight matrix. We also include a $l \times l$ uncertainty matrices, Υ' in the nonlinear kernel, $k_2(f(\mathbf{x}), f(\mathbf{x}'))$. For example, one possible choice is a Gaussian kernel,

$$k_2(f(\mathbf{x}), f(\mathbf{x}')) = \exp\left(-\frac{1}{\sigma}\left(f(\mathbf{x}) - f(\mathbf{x}')\right)^T \Upsilon'^{-1}\left(f(\mathbf{x}) - f(\mathbf{x}')\right)\right),$$

where σ is the Gaussian kernel parameter. For standard history matching, the usual choice for the uncertainty matrix is

$$\Upsilon = \Sigma_e + \Sigma_\eta,$$

where Σ_e is the observation error variance matrix and Σ_η is the discrepancy variance (original introduced by Salter et al. (2019)). We adopt this assumption for Υ, and we will show that standard history matching can be achieved by KHM when the linear kernel is picked with this assumption in the next section. For computationally reasons, we use the same assumption for Υ' here

$$\Upsilon' = \Upsilon = \Sigma_e + \Sigma_\eta.$$

However, this assumption might not be the best in terms of finding the best low-dimensional embedding of the output. It would be useful to consider in the future to provide different uncertainty matrices for Υ' for different nonlinear kernels which might lead to useful approaches.

It is often the case that the discrepancy variance is not well understood by the modellers, and needs to be modelled (we presented some discrepancy models in Chapter 2). Salter et al. (2019) model the discrepancy as a multivariate Gaussian process with mean zero for a simulator with high dimensional output, with unknown hyper-parameters that determine the covariance matrix. In some applications, e.g. the moving pattern situation introduced in the last chapter, the difference between computer model and reality (on the output space) is not meaningful, and there is no model discrepancy (under the usual HM definition). This renders Σ_η as an unknown term in the kernel that allows for distance between model output and reality in feature space. Υ is then the sum of observation error variance Σ_e and the unknown term Σ_η. We will fit the unknown kernel parameters (including Σ_η) in the optimization algorithm using our training data and expert classification.

Note that, as discussed in Section 4.6.1, by quantifying uncertainties through our kernel, what we mean by the distance between model output and observations is represented by the kernel. Specifically, a model run being 'close' to the data or 'far away' (so that it might be ruled out) is a judgement captured in the kernel, unlike in accounts of standard history matching, where these judgements appear in the implausibility. Note also that when $\omega = 1$, Σ_η will correspond to what we normally think of as model discrepancy, and KHM will produce the same results as standard history matching (we demonstrate this in Section 5.2.3).

5.2.3 Achieving standard history matching with KHM

A feature space which normally lies in a higher dimensional space than that of the original output space is determined by a kernel function. For the linear kernel, $k(f(\mathbf{x}'), f(\mathbf{x})) = f(\mathbf{x})^T f(\mathbf{x})$, the feature space is equivalent to the original space, and kernel PCA is exactly equivalent to standard PCA. The explicit formulation of the map function $\phi(\cdot)$ is $\phi(f(x)) = f(x)$.

The relationship between kernel PCA and standard PCA suggests that standard history matching can be achieved by KHM. We adopt the form of our mixture kernel defined in equation (5.4), and set the weight parameter, $\omega = 1$, the kernel function is then

$$k(f(\mathbf{x}), f(\mathbf{x}')) = f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}'),$$

Note that, Υ is symmetric and can be decomposed as $\Upsilon = Q^T H Q$, where Q is an orthogonal matrix, and H is a diagonal matrix with the eigenvalues of Υ as its diagonal elements. Hence, Υ^{-1} can be written as $\Upsilon^{-1} = Q^T H^{-1} Q$. Further, by writing $P = Q^T H^{-1/2}$, we have $\Upsilon^{-1} = PP^T$. The linear kernel is then

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \phi(f(\mathbf{x}))^T \phi(f(\mathbf{x})') = f(\mathbf{x})^T P P^T f(\mathbf{x}').$$

Since we are using the linear kernel for kernel PCA, the feature space is equivalent to the model output space, and the dimension of mapped data, D, is the same as the dimension of model output l. The explicit formulation of the map function in this case is:

$$\phi(f(\mathbf{x})) = P^T f(\mathbf{x}).$$

Therefore, performing kernel PCA on the original dataset $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ is equivalent to performing PCA with the $l \times n$ matrix of mapped ensemble members $\mathbf{\Phi} = (\phi(f(\mathbf{x}_1)), \ldots, \phi(f(\mathbf{x}_n)) = (P^T f(\mathbf{x}_1), \ldots, P^T f(\mathbf{x}_n))$. The mapped ensemble mean, $\bar{\phi}$, given by averaging across the rows of Φ, could be expressed as the original ensemble mean u, with

$$\bar{\phi} = P^T u.$$

Given the centred mapped ensemble $\tilde{\Phi}$, the PCA/SVD basis, $\mathbf{W}$, can be calculated via

$$\tilde{\Phi}^T = EB\mathbf{W}^T,$$

where E is a $n \times n$ orthonormal matrix, $\mathbf{W}$ is a $D \times D$ orthonormal matrix, and B is a $n \times D$ matrix with non-zero elements only along the main diagonal. $\mathbf{W}$ is truncated after the first r vectors, giving the truncated basis, $\mathbf{W}^r = (W_1, \ldots, W_r)$. The projection of a mapped output $\phi(f(\mathbf{x})) = P^T f(\mathbf{x})$ onto $\mathbf{W}^r$ is given by

$$\mathbf{C}_r(\mathbf{x}) = (\mathbf{W}^{rT}\mathbf{W}^r)^{-1}\mathbf{W}^{rT}(\phi(f(\mathbf{x})) - \bar{\phi}), \tag{5.5}$$

where $\mathbf{C}_r(\mathbf{x}) = [C_1(\mathbf{x}), \ldots, C_r(\mathbf{x})]^T$, and this projection is exactly the same as the projections for PCA-based history matching given in Salter and Williamson (2019):

$$\begin{aligned}\mathbf{C}_r(\mathbf{x}) &= (\mathbf{W}^T_{PCA_r}PP^T\mathbf{W}_{PCA_r})^{-1}\mathbf{W}^T_{PCA_r}P(\phi(f(\mathbf{x})) - \bar{\phi}) \\ &= (\mathbf{W}^T_{PCA_r}PP^T\mathbf{W}_{PCA_r})^{-1}\mathbf{W}^T_{PCA_r}PP^T(f(\mathbf{x}) - u) \\ &= (\mathbf{W}^T_{PCA_r}\Upsilon^{-1}\mathbf{W}_{PCA_r})^{-1}\mathbf{W}^T_{PCA_r}\Upsilon^{-1}(f(\mathbf{x}) - u),\end{aligned} \tag{5.6}$$

where $\mathbf{W}_{PCA}$ is the PCA basis, $\mathbf{W}_{PCA_r}$ is the truncated basis, and $P^T\mathbf{W}_{PCA} = \mathbf{W}$. This is because $\mathbf{W}_{PCA}$ is defined using the original model outputs from the ensemble,

$$F^T = E'B'\mathbf{W}^T_{PCA}.$$

Here, E' is a $n \times n$ orthonormal matrix, $\mathbf{W}_{PCA}$ is an $l \times l$ orthonormal matrix, and B' is an $n \times l$ matrix with non-zero elements only along the main diagonal. By the properties of the singular value decomposition, and as $\phi(f(\mathbf{x})) = P^T f(\mathbf{x})$, we have that $\mathbf{W} = P^T\mathbf{W}^T_{PCA}$.

To test whether KHM is the same as standard history matching based on the proposed kernel function, we compare the implausibility functions. When there is no emulator, the implausibility for standard history matching, $\mathcal{I}(\mathbf{x})$, is defined as the Mahalanobis distance between the observations and the computer model:

$$\mathcal{I}(\mathbf{x}) = (z - f(\mathbf{x}))^T(\Sigma_e + \Sigma_\eta)^{-1}(z - f(\mathbf{x})).$$

For the same input $\mathbf{x}$, the implausibility for KHM, $\mathcal{I}_{F0}(\mathbf{x})$, is defined as the euclidean distance between the observations and the computer model:

$$\mathcal{I}_{F0}(\mathbf{x}) = (\phi(z) - \phi(f(\mathbf{x})))^T (\phi(z) - \phi(f(\mathbf{x}))),$$

which can be written as:

$$\begin{aligned} \mathcal{I}_{F0}(\mathbf{x}) &= (P^T z - P^T f(\mathbf{x}))^T (P^T z - P^T f(\mathbf{x})) \\ &= (z - f(\mathbf{x}))^T P P^T (z - f(\mathbf{x})) \\ &= (z - f(\mathbf{x}))^T \Upsilon^{-1} (z - f(\mathbf{x})) \\ &= (z - f(\mathbf{x}))^T (\Sigma_e + \Sigma_\eta)^{-1} (z - f(\mathbf{x})). \end{aligned} \tag{5.7}$$

Hence, the implausibility for kernel PCA-based history matching and PCA history matching is the same.

When the emulators are required and built for the coefficients on the first r basis vectors, $\mathbf{C}_r(\mathbf{x}) = [C_1(\mathbf{x}), \ldots, C_r(\mathbf{x})]^T$,

$$C_k(\mathbf{x}) \sim \mathcal{GP}(m_k(\mathbf{x}), \sigma_k^2 c_k(\mathbf{x}, \mathbf{x})), \quad k = 1, \ldots, r, \tag{5.8}$$

with the emulator expectation for each of the r basis vectors given by $\mathrm{E}[\mathbf{C}_r(\mathbf{x})] = [\mathrm{E}[C_1(\mathbf{x})], \ldots, \mathrm{E}[C_r(\mathbf{x})]]^T$, and the associated emulator variance matrix: $\mathrm{Var}[\mathbf{C}_r(\mathbf{x})] = \mathrm{diag}[\mathrm{Var}[C_1(\mathbf{x})], \ldots, \mathrm{Var}[C_r(\mathbf{x})]]$. Note that the coefficients are same for PCA-based history matching and kernel PCA-based history matching. Thus, we retrieve the expectation and variance of $f(\mathbf{x})$ via:

$$\mathrm{E}[f(\mathbf{x})] = \mathbf{W}_{PCA_r} \mathrm{E}[\mathbf{C}_r(\mathbf{x})] + u, \quad \mathrm{Var}[f(\mathbf{x})] = \mathbf{W}_{PCA_r} \mathrm{Var}[\mathbf{C}_r(\mathbf{x})] \mathbf{W}_{PCA_r}^T, \tag{5.9}$$

and the expectation and variance of $\phi(f(\mathbf{x}))$ via:

$$\mathrm{E}[\phi(f(\mathbf{x}))] = \mathbf{W}^r \mathrm{E}[\mathbf{C}_r(\mathbf{x})] + \bar{\phi}, \quad \mathrm{Var}[\phi(f(\mathbf{x}))] = \mathbf{W}^r \mathrm{Var}[\mathbf{C}_r(\mathbf{x})] \mathbf{W}^{rT}. \tag{5.10}$$

Since $\mathbf{W} = P^T\mathbf{W}_{PCA}^T$ and $\bar{\phi} = P^T u$, we then can write the relationship between $\mathrm{E}\left[f(\mathbf{x})\right]$ and $\mathrm{E}\left[\phi(f(\mathbf{x}))\right]$: $\mathrm{E}\left[\phi(f(\mathbf{x}))\right] = P^T\mathrm{E}\left[f(\mathbf{x})\right]$, and the relationship between $\mathrm{Var}\left[f(\mathbf{x})\right]$ and $\mathrm{Var}\left[\phi(f(\mathbf{x}))\right]$: $\mathrm{Var}\left[\phi(f(\mathbf{x}))\right] = P\mathrm{Var}\left[f(\mathbf{x})\right]P^T$. Based on these relationships, we prove that the implausibility for standard history matching is the same the implausibility for KHM, we have:

$$\mathcal{I}(\mathbf{x}) = \mathcal{I}_{F0}(\mathbf{x}), \tag{5.11}$$

where

$$\mathcal{I}(\mathbf{x}) = \left(z - \mathrm{E}\left[f(\mathbf{x})\right]\right)^T \left(\Sigma_e + \Sigma_\eta + \mathrm{Var}\left[f(\mathbf{x})\right]\right)^{-1} \left(z - \mathrm{E}\left[f(\mathbf{x})\right]\right),$$

and

$$\mathcal{I}_{F2}(\mathbf{x}) = \left(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]\right)^T \left(\mathbf{1}_D + \mathrm{Var}\left[\phi(f(\mathbf{x}))\right]\right)^{-1} \left(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]\right).$$

A full proof of this result is given in Appendix A.5.

When the implausibility threshold is set as with the PCA choice, KHM will have the same NROY space as standard history matching. As such, KHM is a generalised version of standard history matching, and simulator output space is one possible choice of feature space we can perform history matching in.

5.3 Fitting the kernel parameters

Our mixture kernel function combines a nonlinear kernel with a linear kernel. To perform history matching, we must estimate the unknown parameters in this mixture kernel. We denote all of the unknown parameters in the mixture kernel as $\mathcal{K}_{par}$, including a weight parameter, $\omega \in [0,1]$, nonlinear kernel parameters κ, (e.g. σ, when a Gaussian kernel is applied), and unknown parameters, ϕ_η, used to specify Σ_η. We have $\mathcal{K}_{par} = (\omega, \kappa, \phi_\eta)$.

We define the kernel parameter selection problem as

$$\mathcal{K}^*_{par} = \arg\max_{\mathcal{K}_{par}} \mathcal{P}(\mathcal{K}_{par}), \tag{5.12}$$

where $\mathcal{K}^*_{par}$ is a set of optimal kernel parameter values, and $\mathcal{P}(\cdot)$ is a performance evaluation function to be introduced in the following section.

5.3.1 Evaluation of history matching performance

History matching attempts to identify the parts of the input parameter space that are likely to result in mismatches between computer outputs and observations. Given an implausibility function, $\mathcal{I}(\mathbf{x})$, and a cutoff threshold, T, an NROY space $\mathcal{X}_{NROY}$ can be defined. Comparing the difference between $\mathcal{X}_{NROY}$ and "true" NROY space, is the most obvious measure for history matching performance evaluation. In Chapter 3, we have defined the "true" NROY space, $\mathcal{X}^*$, for standard history matching, which is the NROY space found by the computer model directly (without an emulator):

$$\mathcal{X}^* = \left\{ x \in \mathcal{X} : \frac{|z - f(x)|}{\sqrt{\text{Var}\left[e\right] + \text{Var}\left[\eta\right]}} \leq T \right\}, \tag{5.13}$$

where $T = 3$. In order to evaluate standard history matching performance, we use $\mathcal{X}^*$ to compare with the NROY space found using an emulator. Following equation (5.13), we have $\mathcal{X}^*$ for KHM

$$\mathcal{X}^*(\mathcal{K}_{par}) = \{\mathbf{x} \in \mathcal{X} : \mathcal{I}_{F0}(\mathbf{x}) \leq T\}, \tag{5.14}$$

where $\mathcal{I}_{F0}(\mathbf{x}) = ||\phi(z) - \phi(f(\mathbf{x}))||^2$ (given in equation 4.70). Given the mixture kernel defined in equation (5.4), the NROY space found by the computer model directly, $\mathcal{X}^*(\mathcal{K}_{par})$, depends on kernel choices, $\mathcal{K}_{par}$. In order to define true NROY for KHM, a "best" kernel is required. We assume there is a modeller's (expert's) kernel (the mixture kernel with parameters, $\mathcal{K}^E_{par}$) that produces a modeller's NROY space. By setting this modeller's kernel as the "best" kernel, the modeller's

NROY space can be seen as the true NROY space,

$$\mathcal{X}^*_{KHM} = \mathcal{X}^*(\mathcal{K}^E_{par}).$$

Note that, we do not know $\mathcal{K}^E_{par}$, and the true NROY space, $\mathcal{X}^*_{KHM}$, cannot be assessed directly. We would not expect the modeller to have judgement about $\mathcal{K}^E_{par}$, but about $\mathcal{X}^*_{KHM}$ as a result of an understanding best captured through $\mathcal{K}^E_{par}$.

To learn the true NROY space, we label the training data based on the information given by experts. The ensemble members, $f(\mathbf{x})$, are labelled either as acceptable runs F^A or unacceptable runs F^U, with corresponding acceptable/unacceptable inputs $\mathbf{x}^A/\mathbf{x}^U$ (details are introduced in Section 4.4.3). Suppose there are p acceptable runs, so that the acceptable inputs set can be denoted as $\mathbf{X}^A = \{\mathbf{x}^A_1, \ldots, \mathbf{x}^A_p\}$, and the unacceptable inputs as $\mathbf{X}^U = \{\mathbf{x}^U_1, \ldots, \mathbf{x}^U_{n-p}\}$. These acceptable/unacceptable runs provide prior knowledge relating to which patterns are important and can be seen as partial information on true NROY, $\mathcal{X}^*_{KHM}$. With the "best" kernel, all of the acceptable runs should be retained in the NROY space, and all of the unacceptable runs should be ruled out (other situations accounting for human error will be discussed in Chapter 6). We treat the acceptable runs as $\mathcal{X}^A$, the aim of kernel optimization is then to find the kernel that represents $\mathcal{X}^A$ as well as possible for KHM.

In order to assess the performance of KHM with a given set of parameters, $\mathcal{K}_{par}$, we compare $\mathcal{X}^A$ with the produced NROY space $\mathcal{X}_{NROY}(\mathcal{K}_{par})$, where

$$\mathcal{X}_{NROY}(\mathcal{K}_{par}) = \{\mathbf{x} \in \mathcal{X} : ||(\phi(z) - \mathbf{W}^r \mathbf{C}_r(\mathbf{x})||^2 \leq T\}, \tag{5.15}$$

and $\mathbf{W}^r$ is a vector of the first r Kernel PCA basis vectors under $\mathcal{K}_{par}$. Unlike using $\mathcal{I}_{F0}(\mathbf{x})$ in equation (5.14), the implausibility function used in equation (5.15) is the distance between mapped observation, $\phi(z)$, and model output reconstruction, $\mathbf{W}^r\mathbf{C}_r(\mathbf{x})$. This enables us to fit Gaussian process emulators for the coefficients. Note that, by using equation (5.15) to access performance, we ensure that the chosen subspace defined by $\mathbf{W}^r$ can extract the key signal that we are calibrating for. To facilitate the comparison between $\mathcal{X}^*_{KHM}$ and $\mathcal{X}_{NROY}(\mathcal{K}_{par})$, two indicators are proposed to measure the accuracy and efficiency of KHM. We define these two indicators separately.

Accuracy

Given p acceptable runs, $\mathbf{X}^A$, and the produced NROY space, $\mathcal{X}_{NROY}(\mathcal{K}_{par})$, we can compute the number of cases when an acceptable input $\mathbf{x}^A$ is retained in $\mathcal{X}_{NROY}(\mathcal{K}_{par})$,

$$N_A(\mathcal{K}_{par}) = \sum_{\mathbf{x}\in\mathbf{X}^A} \mathbf{1}\Big(\mathcal{I}(\mathbf{x}) \le T\Big), \tag{5.16}$$

where $\mathbf{1}$ is the indicator function and $0 \le N_A(\mathcal{K}_{par}) \le p$. Given the value of $N_A(\mathcal{K}_{par})$, we define the accuracy of KHM as a function of $\mathcal{K}_{par}$ that computes the proportion of acceptable runs to be retained in $\mathcal{X}_{NROY}(\mathcal{K}_{par})$,

$$\mathcal{A}(\mathcal{K}_{par}) = \frac{1}{p} N_A(\mathcal{K}_{par}) \tag{5.17}$$

and $\mathcal{A}(\mathcal{K}_{par}) \in [0,1]$. The optimal situation for accuracy, $\mathcal{A}(\mathcal{K}_{par}) = 1$, can be reached when all of the acceptable runs, $\mathbf{x}^A \in \mathbf{X}^A$, are retained in the NROY space, $||(\phi(z) - \mathbf{W}^r\mathbf{C}_r(\mathbf{x}^A)||^2 \le T$. However, the optimal situation for accuracy does not necessarily represent the optimal situation for history matching. For instance, $\mathcal{A}(\mathcal{K}_{par}) = 1$ can be reached with a large cutoff T that can ensure that no space would be ruled out at all. Hence, although a high level of accuracy guarantees that acceptable runs are retained in the NROY space, it is also important to rule out most of the unacceptable runs. We introduce another measure to assess the efficiency of history matching.

Efficiency

To test efficiency, we usually compute the volume of $\mathcal{X}_{NROY}(\mathcal{K}_{par})$, to establish how much space has been removed. However, this does not mean the smaller size of NROY the better. Ideally NROY space is as close as possible to the true

NROY space. We define efficiency as a measure that tells us the proportion of unacceptable runs that history matching diagnoses as being acceptable. Denote the number of cases when unacceptable input, $\mathbf{x}^U$, is incorrectly retained in the produced NROY as N_U,

$$N_U(\mathcal{K}_{par}) = \sum_{\mathbf{x}\in\mathbf{X}^U} \mathbf{1}\Big(\mathcal{I}(\mathbf{x}) \leq T\Big), \tag{5.18}$$

We define the efficiency of a kernel as

$$\mathcal{E}(\mathcal{K}_{par}) = \frac{N_A(\mathcal{K}_{par})}{N_A(\mathcal{K}_{par}) + N_U(\mathcal{K}_{par})}. \tag{5.19}$$

The optimal situation for efficiency, $\mathcal{E}(\mathcal{K}_{par}) = 1$, can be reached when all of the unacceptable runs can be ruled out, $\mathcal{I}(\mathbf{x}^U)) > T$ for $\mathbf{x}^U \in \mathbf{X}^U$. Noted, with a smaller cutoff than any implausibility value, $\mathcal{E}(\mathcal{K}_{par})$ is undefined. To avoid this situation, we require that $T \geq \min_{\mathbf{x}\in\mathbf{X}}(\mathcal{I}(\mathbf{x}^U))$.

A successful calibration requires that all of the acceptable runs, $\mathbf{x}^A$, are retained in NROY space. History matching is, thus, seen as successful if a high number of parameter choices for unacceptable runs, $\mathbf{x}^U$, can be ruled out. By combining the accuracy measure and efficiency measure, our performance evaluation for history matching is:

$$\mathcal{P}(\mathcal{K}_{par}) = \alpha\mathcal{A}(\mathcal{K}_{par}) + (1-\alpha)\mathcal{E}(\mathcal{K}_{par}), \tag{5.20}$$

where α is an influence factor, which compromises between $\mathcal{A}(\mathcal{K}_{par})$ and $\mathcal{E}(\mathcal{K}_{par})$. In early waves of KHM, it is more important to keep all of the good runs. Thus, it is advisable to use a large value for α, e.g. $\alpha = 0.8$. The optimal situation, $\mathcal{P}(\mathcal{K}_{par}) = 1$, can be achieved when the kernel based-history matching satisfies

$$\max_{\mathbf{x}^A\in\mathbf{X}^A}(\mathcal{I}(\mathbf{x}^A)) \leq T < \min_{\mathbf{x}^U\in\mathbf{X}^U}(\mathcal{I}(\mathbf{x}^U)). \tag{5.21}$$

In this optimal situation, the produced NROY space, $\mathcal{X}_{NROY}(\mathcal{K}_{par})$, contains all of the acceptable runs and rules out all of the unacceptable runs.

5.3.2 Cutoff threshold: T

Ideally, we want the threshold, T, to be a clear 'boundary' between the acceptable and unacceptable runs, but determining where this 'boundary' lies will be impossible without an explicit mapping function. In Section 4.4.3, we set the value of T to depend on the kernel, because kernel choices impact the magnitude of the implausibility. We could use $\min(\mathcal{I}(\mathbf{x}^U))$ for $\mathbf{x}^U \in \mathbf{X}^U$ as one possible choice for threshold T. However, setting $T = \min(\mathcal{I}(\mathbf{x}^U))$ will not be suitable when equation (5.21) does not hold. In the same way as the example given in the last chapter, with a poor kernel function, unacceptable runs could be much closer to observations than the acceptable runs. Moreover, the optimal situation will never materialise if there exists any human error in the expert judgement (discussion and an illustration of this is given in the next chapter).

To help selection of a kernel that will be suitable, we determine the choice for the cutoff threshold within the kernel selection algorithm. Given the mixture kernel, we define the threshold as a function of $\mathcal{K}_{par}$, $T(\mathcal{K}_{par})$. The performance evaluation $\mathcal{P}(\cdot)$ is then defined as a function of $T(\mathcal{K}_{par})$ and $\mathcal{K}_{par}$. The optimisation of the threshold with given $\mathcal{K}_{par}$ can then be defined as

$$T^*(\mathcal{K}_{par}) = \arg\max_T \{\mathcal{P}(T(\mathcal{K}_{par}),\ \mathcal{K}_{par})\}. \tag{5.22}$$

Note that, with finite training data, the value of optimised threshold $T^*(\mathcal{K}_{par})$ is not unique. For example, given a suitable $\mathcal{K}_{par}$, any value of $T^*(\mathcal{K}_{par})$ in the range of $[\max(\mathcal{I}(\mathbf{x}^A)), \min(\mathcal{I}(\mathbf{x}^U))]$ gives $\mathcal{P}(T(\mathcal{K}_{par}), \mathcal{K}_{par}) = 1$. However, with different values of threshold between $[\max(\mathcal{I}(\mathbf{x}^A))$ and $\min(\mathcal{I}(\mathbf{x}^U))]$, the produced NROY spaces are different. To avoid ruling out any true NROY space, we keep the largest NROY space satisfying equation (5.22), establishing our optimal threshold as

$$T^{**}(\mathcal{K}_{par}) = \max\left\{T^* : T^* = \arg\max_T \left\{\mathcal{P}\left(T(\mathcal{K}_{par}),\ \mathcal{K}_{par}\right)\right\}\right\}. \tag{5.23}$$

The optimal threshold, $T^{**}(\mathcal{K}_{par})$, can be directly used for the implausibility measure introduced in equation (5.15) only. However, for the implausibilities introduced in Chapter 4 that involve emulators (equations (4.71) and (4.80)), the emulator variance also needs to be considered in the threshold. We first consider the threshold, $T(\mathbf{x})$, for implausibility

$$\mathcal{I}_{F1}(\mathbf{x}) = \parallel \phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right] \parallel^2 .$$

In Section 4.6.3, we derived a threshold for $\mathcal{I}_{F1}(\mathbf{x})$:

$$T(\mathbf{x}) = \sum_{k=1}^{r} \mathrm{Var}\left[C_k(\mathbf{x})\right] + 3\sqrt{2\left(\sum_{k=1}^{r} \mathrm{Var}\left[C_k(\mathbf{x})\right]\right)^2} + a,$$

where a is the maximum value for $\parallel \phi(z) - \phi(f(\mathbf{x}^*)) \parallel^2$. Instead of assuming $a = \min(\parallel \phi(\mathrm{z}) - \phi_{\mathrm{r}}(\mathrm{f}(\mathbf{x}^{\mathrm{U}})) \parallel^2)$ as in Section 4.6.3, here we set a as $T^{**}(\mathcal{K}_{par})$. The optimal threshold for $\mathcal{I}_{F1}(\mathbf{x})$ is then

$$T^*(\mathbf{x}) = \sum_{k=1}^{r} \mathrm{Var}\left[C_k(\mathbf{x})\right] + 3\sqrt{2\left(\sum_{k=1}^{r} \mathrm{Var}\left[C_k(\mathbf{x})\right]\right)^2} + T^{**}(\mathcal{K}_{par}).$$

Note that, as the variance of the emulator tends to 0, $\mathrm{Var}\left[\phi(f(\mathbf{x}))\right] \longrightarrow 0$, $T^*(\mathbf{x}) \longrightarrow T^{**}(\mathcal{K}_{par})$.

For implausibility

$$\mathcal{I}_{F2}(\mathbf{x}) = (\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right])^T (\mathbf{1}_D + \mathrm{Var}\left[\phi(f(\mathbf{x}))\right])^{-1} (\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]),$$

a possible threshold, T' is defined in Section 4.6.4 using 'leave one out' diagnostics, $T' = \min(\mathcal{I}_{F2}(\mathbf{x}_j^U))$. We define the optimisation of the threshold with given $\mathcal{K}_{par}$ for $\mathcal{I}_{F2}(\mathbf{x})$ as

$$T'^{**} = \max\left\{T'^{*} : T'^{*} = \arg\max_{T'} \left\{\mathcal{P}'\left(T', \mathcal{K}_{par}\right)\right\}\right\}, \tag{5.24}$$

where $\mathcal{P}'(\cdot)$ is the performance evaluation function for KHM with $\mathcal{I}_{F2}(\mathbf{x})$:

$$\mathcal{P}'(T', \mathcal{K}_{par}) = \alpha\mathcal{A}'(\mathcal{K}_{par}) + (1-\alpha)\mathcal{E}'(\mathcal{K}_{par}),$$

where $\mathcal{A}'(\mathcal{K}_{par})$ and $\mathcal{E}'(\mathcal{K}_{par})$ are defined in equations (5.17) and (5.19), but $N_A(\mathcal{K}_{par})$ and $N_U(\mathcal{K}_{par})$ are computed with $\mathcal{I}_{F2}(\mathbf{x})$:

$$N_A(\mathcal{K}_{par}) = \sum_{\mathbf{x}\in\mathbf{X}^A} \mathbf{1}\Big(\mathcal{I}_{F2}(\mathbf{x})) \leq T\Big),$$

and

$$N_U(\mathcal{K}_{par}) = \sum_{\mathbf{x}\in\mathbf{X}^U} \mathbf{1}\Big(\mathcal{I}_{F2}(\mathbf{x})) \leq T\Big).$$

Details for computing $\mathcal{I}_{F2}(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{X}^A \cup \mathbf{X}^U$ using 'leave one out' validation predictions for the training data are given in Section 4.6.4. Note that, as the variance of the emulator tends to 0, $\text{Var}[\phi(f(\mathbf{x}))] \longrightarrow 0$, $T^{'**} \longrightarrow T^{**}(\mathcal{K}_{par})$ and $\mathcal{P}'\left(T'(\mathcal{K}_{par}), \mathcal{K}_{par}\right) \longrightarrow \mathcal{P}\left(T'(\mathcal{K}_{par}), \mathcal{K}_{par}\right)$.

5.3.3 Kernel selection procedure

Given the mixture kernel in equation (5.4), we present our procedure for selecting an optimal kernel and a cutoff value for KHM with the objective function $\mathcal{P}(\mathcal{K}_{par}, T)$. In order to find $\mathcal{K}^*_{par}$, we use simulated annealing to optimise the objective function (Van Laarhoven and Aarts, 1987), though any procedure may be used.

Step 1: Perform kernel PCA for training data using the mixture kernel function with the given parameter setting $\mathcal{K}_{par}$.

Step 2: Calculate the implausibility for all of the ensemble members through

$$\mathcal{I}(\mathbf{x}_i) = ||(\phi(z) - \mathbf{W}^r\mathbf{C}_r(\mathbf{x}_i)||^2, \qquad i = 1, \ldots, n.$$

Step 3: Set the optimal cutoff threshold for given $\mathcal{K}_{par}$ as

$$T^{**}(\mathcal{K}_{par}) = \max\left\{T^* : T^* = \arg\max_T \left\{\mathcal{P}\left(T(\mathcal{K}_{par}), \mathcal{K}_{par}\right)\right\}\right\}.$$

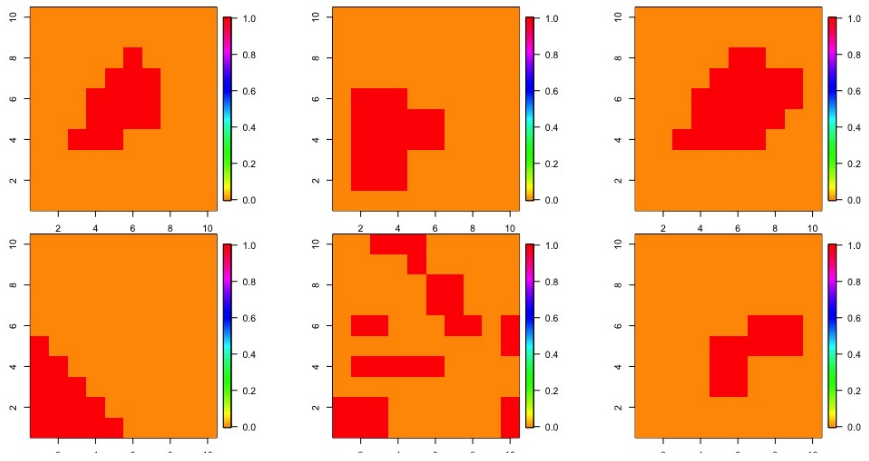


Fig. 5.1 The vectors $B_1, B_2, \ldots, B_6$ used to defined the toy example.

Step 4: Calculate the performance evaluation for given $\mathcal{K}_{par}$ and $T^{**}(\mathcal{K}_{par})$

$$\mathcal{P}(T^{**}(\mathcal{K}_{par}), \mathcal{K}_{par}) = \alpha\mathcal{A}(T^{**}(\mathcal{K}_{par}), \mathcal{K}_{par}) + (1-\alpha)\mathcal{E}(T^{**}(\mathcal{K}_{par}), \mathcal{K}_{par}).$$

For every perturbation of $\mathcal{K}_{par}$, first we must find $T^{**}(\mathcal{K}_{par})$, which is done by step 1 to step 3 in our algorithms, then we evaluate $\mathcal{P}(T^{**}(\mathcal{K}_{par}))$ in step 4. To optimise $\mathcal{K}^{*}_{par}$ we use simulated annealing to optimise the objective function $\mathcal{P}(T^{**}(\mathcal{K}_{par}))$. Because simulated annealing is a time-consuming algorithm, in practice we set a maximum time or a maximum performance evaluation, $M_{\mathcal{P}}$, we are willing to accept as a stopping criterion.

5.4 Numerical study 1

We apply the proposed KHM to a toy example, $f(\mathbf{x})$, with six input parameters, $x_1, \ldots, x_6$, on $[-1,1]^6$. The toy function is defined as

$$f(\mathbf{x}) = f(x_1, \ldots, x_6) = 4\pi_N((x_5+x_6),\ 1.3,\ 0.3)(B_3+B_1)$$
$$+1.5(x_1^2 B_6 + x_1 x_3 B_4 + \sin(x_4)B_2) + x_2 B_5 + e_n,$$

where $\pi_N((x_5+x_6),\ 1.3,\ 0.3)$ denotes the density function of the Normal distribution with mean 1.3 and variance 0.3, e_n gives a general background of the output that samples from a Normal distribution mean 15 (to ensure the output is positive) and variance 0.05, independently for each box in a 10×10 grid, and $B_1,\ B_2,\ \ldots,\ B_6$ are six vectors which are specified over the grid, as shown in Figure 5.1. With a given input parameter, $\mathbf{x}$, an output field $f(\mathbf{x})$ is given by combining the vectors $B_1,\ B_2,\ \ldots,\ B_6$. The most important of these, B_1 and B_3, contain the key pattern (the main pattern in the observation), and the toy model function shows that x_5 and x_6 in combination control the B_1 and B_3 vectors.

The observation, $z = f(\mathbf{x}_z) + e$, where $\mathbf{x}_z = (0.1,\ 0.01,\ 0.1,\ -0.1,\ 1,\ 0.3)$ and $e \sim N(0,\ \Sigma_e)$. To define Σ_e, we use a squared exponential correlation function (equation (2.5)). Denoting the spatial coordinates of the 10×10 grid as s (including horizontal coordinates and vertical coordinates), and setting the correlation length parameter between two inputs for each input dimension as 1, we then have the correlation between two inputs, s and s', as

$$c(s,s') = \exp\left\{-\frac{1}{2}\sum_{c=1}^{2}(s_c - s'_c)^2\right\}. \tag{5.25}$$

Hence the i, j^{th} entry of Σ_e is computed by the correlation function between input s^i and s^j, $c(s^i, s^j)$. The observation, z, is shown in the left panel of Figure 5.2. The main feature of the observation is the red and blue signal in the middle, other than which there are no clear patterns. The existence of the key pattern is much more important to us than the location of the key pattern in this idealised example.

A Latin hypercube sample of size 60 is taken from the 6-dimensional parameter space, $\mathbf{x} \in \mathcal{X}$, giving an ensemble $\mathbf{F}$ with dimension 100×60. We plot the mean output field of this ensemble in the right panel of Figure 5.2, and the 60 ensemble outputs in Figure 5.3. For these ensemble runs, only a few of them have the key pattern (in the same location as the observation), and we define the runs that contain the key pattern as the "acceptable runs" for the purpose of KHM. The acceptable runs, $f(\mathbf{x}^A)$, are the ensemble members 31, 52, 44, 33, 45, 19, and 60, about 10% size of the initial ensemble.

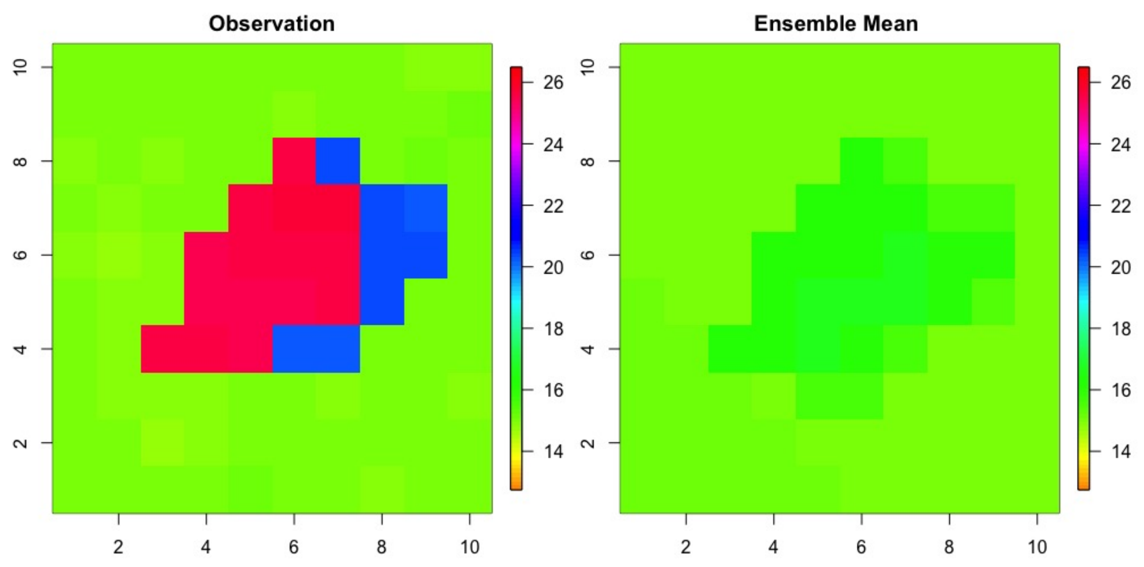


Fig. 5.2 *Left:* the observations, z, for the toy function. *Right:* the mean of the ensemble F.

5.4.1 Kernel selection for the toy function

Based on the selected acceptable runs, we use the algorithm above to select a kernel for KHM. We specify

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \omega f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}') + (1-\omega)\exp(-(f(\mathbf{x}) - f(\mathbf{x}'))^T \Upsilon^{-1} (f(\mathbf{x}) - f(\mathbf{x}'))/\sigma), \tag{5.26}$$

where ω is a weight parameter, $\omega \in [0,1]$, σ is a Gaussian kernel parameter, and Υ is a $l \times l$ positive defined weighted matrix, where $\Upsilon = \Sigma_e + \Sigma_\eta$. Σ_e is already given with the observation, without any unknown parameters. We specify the term Σ_η following Salter et al. (2019), $\Sigma_\eta = c(s, s')$, where $c(s, s')$ is given in equation (5.25) (details are given in Section 2.4.1). Instead of setting the value for the unknown correlation length parameters δ_1 and δ_2, before calibration, we estimate these two parameters though our kernel selection algorithm. So $\mathcal{K}_{par} = (\omega, \delta_1, \delta_2, \sigma)$, and we use the kernel selection algorithm.

By setting the influence factor, $\alpha = 0.8$, for the performance evaluation function, $\mathcal{P}(\cdot)$ (equation 5.20), our optimization algorithm suggests that the optimal situation, $\mathcal{P}(\mathcal{K}_{par}) = 1$, can be achieved when $T^{**}(\mathcal{K}_{par}) = \min(\mathcal{I}(\mathbf{x})^U)$, $\delta_1 = \delta_2 = 1$ and $\omega = 1$ (since the weight of the Gaussian kernel is zero, the choice of Gaussian

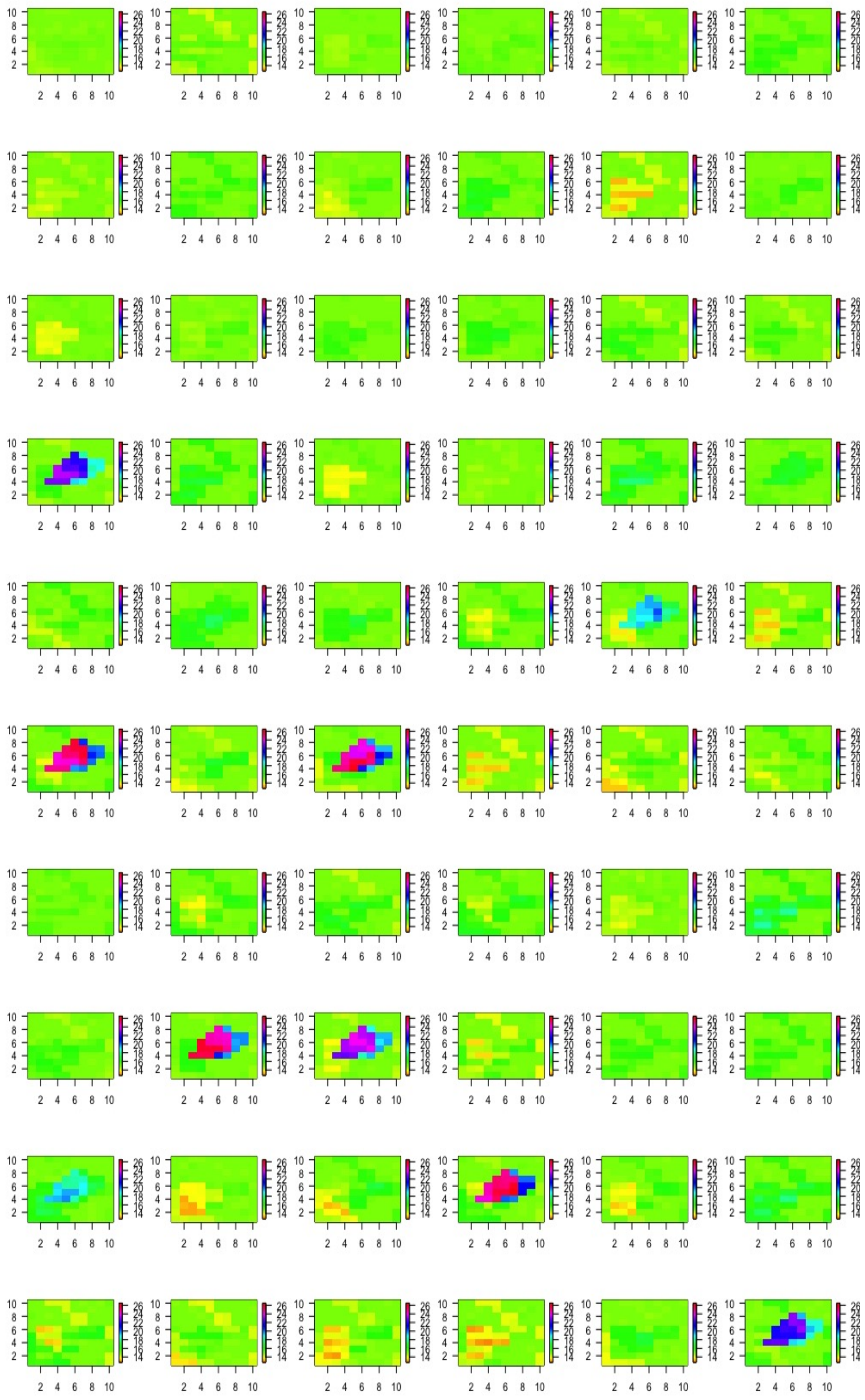


Fig. 5.3 The 60 ensemble members for wave 1.

kernel parameters is irrelevant). The optimal kernel function for the toy example is

$$k(f(\mathbf{x}), f(\mathbf{x}')) = f(\mathbf{x})^T (\Sigma_e + \Sigma_\eta)^{-1} f(\mathbf{x}').$$

KHM with the above kernel represents standard PCA-based history matching (Salter et al. (2019)), which implies that Σ_η corresponds to the discrepancy variance. It is not surprising that we find that the linear kernel is the optimal choice for this example. We built the toy function as a linear combination of six vectors, $B_1, \ldots, B_6$, the location of the key pattern is fixed, and the parameters are responsible for the strength of key pattern. So that the PCA-based standard history matching should work well.

We showed in Section 5.2.3 that the implausibility for KHM,

$$\mathcal{I}_{F2}(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T (\mathbf{1}_D + \mathrm{Var}[\phi(f(\mathbf{x}))])^{-1} (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]),$$

is same as the implausibility function for standard history matching. In addition to $\mathcal{I}_{F2}(\mathbf{x})$, we presented another implausibility measure for KHM in Section 4.6.3:

$$\mathcal{I}_{F1}(\mathbf{x}) = (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]).$$

We continue the example with these two implausibilities to demonstrate the performance of KHM.

5.4.2 KHM for the toy example

Given the optimal kernel function, we calculate the ensemble projections by applying the Kernel PCA algorithm. Our algorithm suggests that only 3 basis vectors are required, and these 3 basis vectors explain 96.99326% of the ensemble variability. The projected ensemble can be written as $\mathbf{C} = (\mathbf{C}_r(\mathbf{x}_1), \ldots, \mathbf{C}_r(\mathbf{x}_n))$ for n design points $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, where $n = 60$, and the dimension of $\mathbf{C}$ is 3×60. We construct a univariate Gaussian process emulator for each basis vector separately. The validation plots for these GP emulators are given in Figure 5.4, and

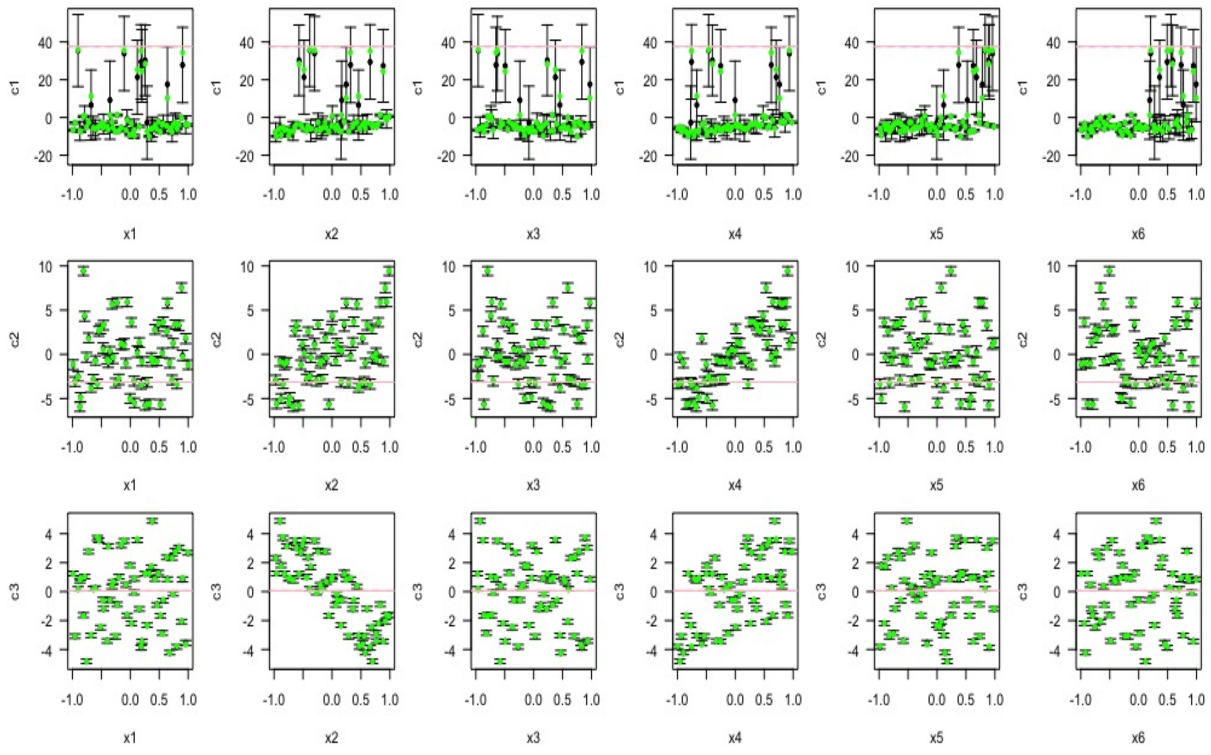


Fig. 5.4 Leave-one-out cross-validation plots for the emulators for the coefficients on the first three basis vectors.

are presented from top to bottom for the first three basis vectors. As before, black points and error bars are from the emulator posterior mean and two standard deviation prediction intervals. The true function values are in green if they lie within the two standard deviation prediction intervals, or red otherwise. We can see there are no failures for any of the three emulators. The emulator for the coefficient on the first basis vector has large uncertainties with some input parameter areas, because of nonstationarity.

A LHC of size 10000 is sampled to represent the initial parameter space $\mathcal{X}$, at which we compute the implausibility function. As the toy function is able to run at any parameter setting, we first compute the true NROY space, $\mathcal{X}^*$, to evaluate KHM performance. Given the linear kernel and model outputs, we have $\mathcal{X}^*$ following the definition of true NROY space given in equation (5.14):

$$\mathcal{X}^* = \{\mathbf{x} \in \mathcal{X} : (z - f(\mathbf{x}))^T(\Sigma_e + \Sigma_\eta)^{-1}(z - f(\mathbf{x})) \leq T\},$$

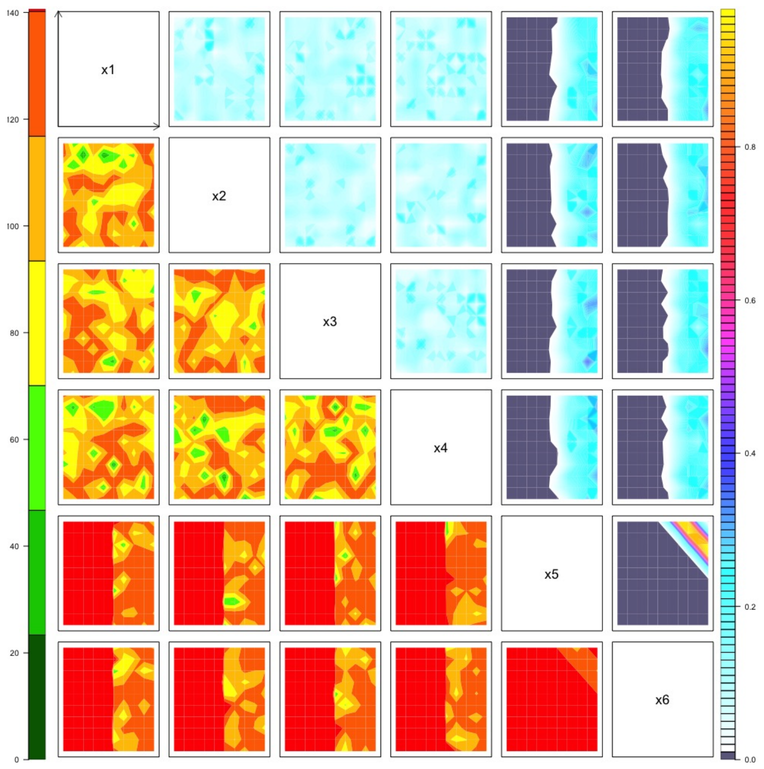


Fig. 5.5 True NROY space.

where T is set as $99.5th$ percentile of a Chi-squared distribution with 100 degrees of freedom. The threshold used here following standard history matching approach, to demonstrate our claim: KHM's performance is the same as standard history matching with our linear kernel function. True NROY space is plotted in Figure 5.5, and the size of the $\mathcal{X}^*$ is 8.37% of the original input space.

We first perform KHM with $\mathcal{I}_{F1}(\mathbf{x})$, the first wave NROY space for the toy model is,

$$\mathcal{X}_{I1} = \{\mathbf{x} \in \mathcal{X} : \mathcal{I}_{F1}(\mathbf{x}) \leq T^*(\mathbf{x})\},$$

where $T^*(\mathbf{x})$ is introduced in Section 5.3.2. NROY space, $\mathcal{X}_{I1}$, is illustrated in the left panel of Figure 5.6. We plot the NROY density plots (upper triangle) and "minimum implausibility" plots (lower triangle) for each pair of parameters. Note that, as the threshold $T^*(\mathbf{x})$ is unfixed, the "minimum implausibility" is computed as $\mathcal{I}_{F1}(\mathbf{x})/T^*(\mathbf{x})$. To make a comparison with KHM with $\mathcal{I}_{F2}(\mathbf{x})$, we demonstrate the example with $\mathcal{I}_{F2}(\mathbf{x})$ as well. The produced NROY space for first wave is

$$\mathcal{X}_{I2} = \{\mathbf{x} \in \mathcal{X} : \mathcal{I}_{F2}(\mathbf{x}) \leq T^{'**}\},$$

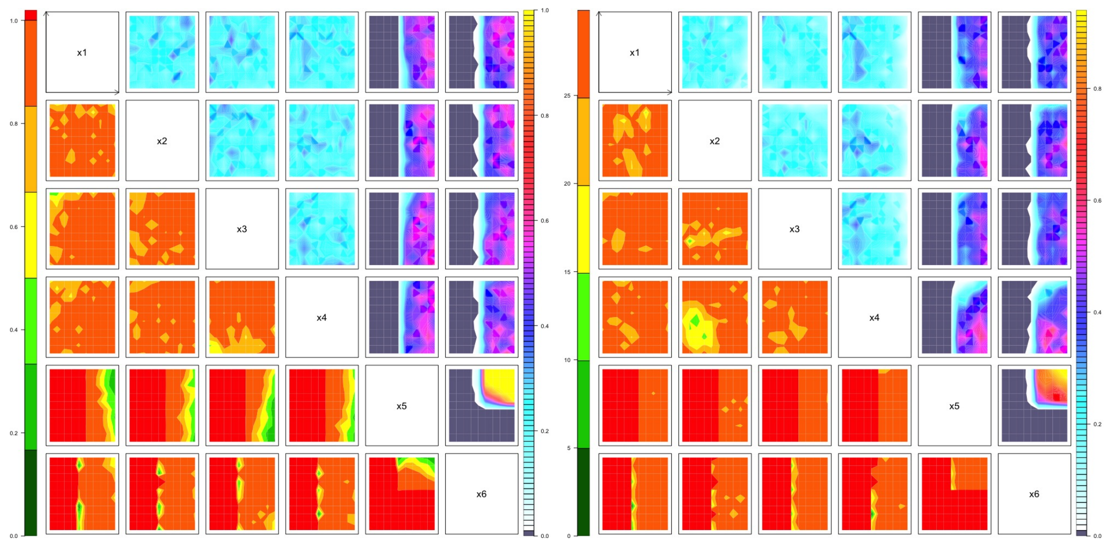


Fig. 5.6 *Left*: Wave 1 NROY space by performing KHM with $\mathcal{I}_{F1}(\mathbf{x})$. *Right*: Wave 1 NROY space by performing KHM with $\mathcal{I}_{F2}(\mathbf{x})$.

and it is illustrated in the right panel of Figure 5.6. The results of $\mathcal{X}_{I2}$ and $\mathcal{X}_{I1}$ are given in Table 5.1. From both NROY density plots, we observe a similar strong relationship between two parameters $\mathbf{x}_6$ and $\mathbf{x}_5$, which indicates the importance of these two parameters for calibration. Moreover, both NROY density plots show that the parameter that been most strongly been constrained is $\mathbf{x}_6$. The results show that the produced NROY space, $\mathcal{X}_{I1}$ and $\mathcal{X}_{I2}$, are similar. $\mathcal{I}_{F1}(\mathbf{x})$ retained a little bit more NROY space than $\mathcal{I}_{F2}(\mathbf{x})$ but 100% of the true NROY is retained in $\mathcal{X}_{I1}$. More discussion about the comparison of these two implausibilities will be given after numerical example 2. Moreover, because the emulator for the first basis vector has large uncertainties, the true NROY space is not immediately identified in the first wave. We take refocusing steps: two more waves are performed in Appendix C.2.

	NROY volume	Retained true NROY
$\mathcal{X}_{I1}$	24.31%	100%
$\mathcal{X}_{I2}$	21.34%	96.21%

Table 5.1 A comparison study between $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$.

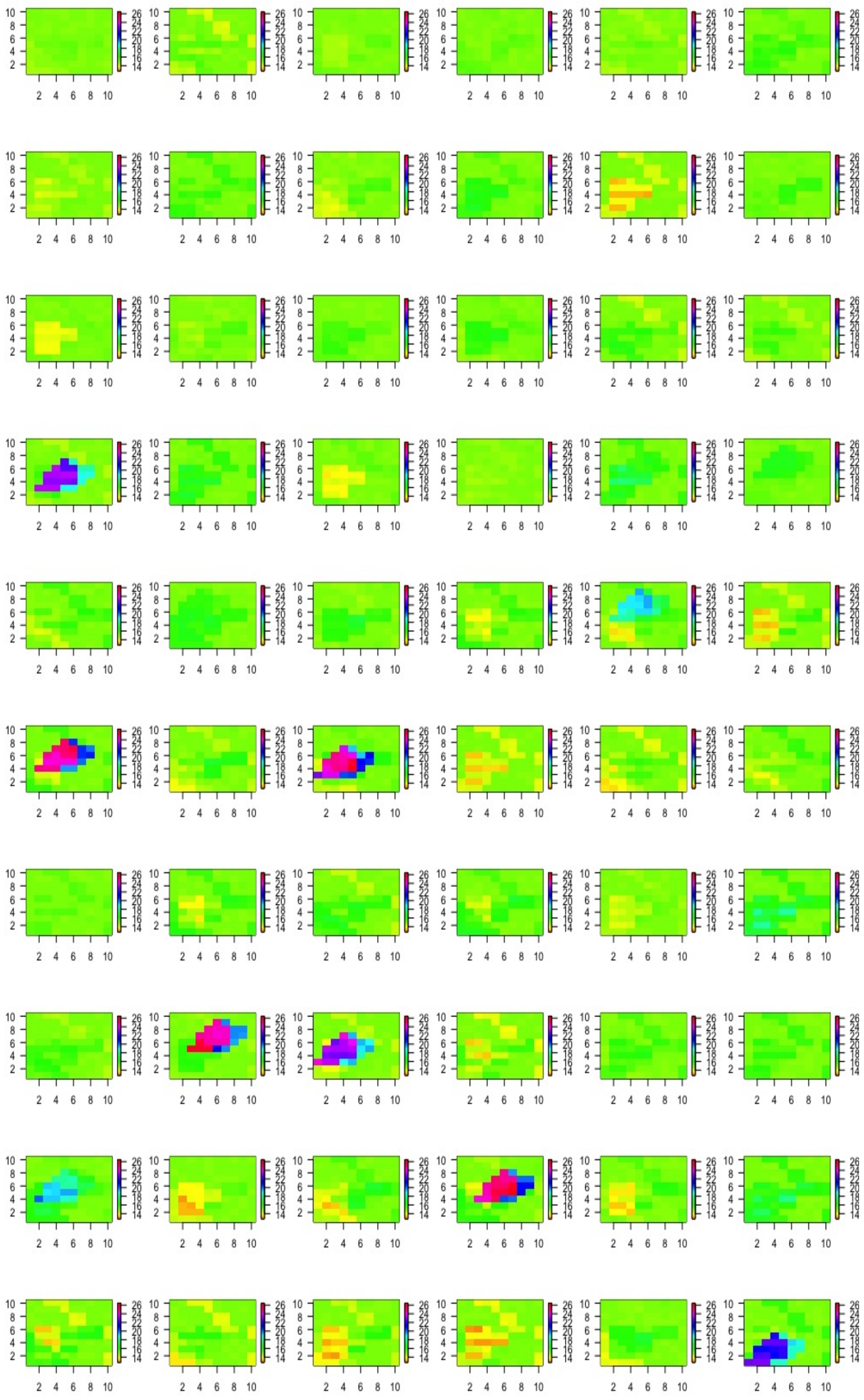


Fig. 5.7 The 60 ensemble members for wave 1.

5.5 Numerical study 2

In this section, we build a new toy example to demonstrate KHM. The toy function for this numerical study is the same as the toy function of the previous example, but we modify the vectors B_1 and B_3. In the first numerical example, we fixed the key pattern in vectors B_1 and B_3 to an unchangeable location. To test whether KHM can search for values of model parameters that lead to models containing the key pattern, no-matter where it is located, we change the toy model to be stochastic, so that it can produce different outputs under the same model conditions. In effect, the location of the key pattern in vectors B_1 and B_3 is produced randomly. Therefore, for the model outputs, the existence of the key pattern is deterministic, but the location of the key pattern is generated randomly. This toy example is not unrealistic and this situation could be happen in climate models. For example, the structure of currents or clouds might have some random components.

The observation error, Σ_e, and Σ_η is taken to be the same as the previous example. For Σ_η, the unknown parameters δ_1 and δ_2 need to be estimated through our kernel selection algorithm. Further, we use the same design, $\mathbf{x} \in \mathcal{X}$, as numerical study 1, and we plot the 60 ensemble outputs, $\mathbf{F}$, in Figure 5.7. By comparison with Figure 5.3, we can see the key pattern exists in the same ensemble members, but the location of the key pattern is different. For this experiment, the existence of the key pattern is much more important than the location of the key pattern, so the choice of acceptable runs is same as the previous example. Hence, all of the settings of this numerical example are the same as the first, other than the location of the key patterns in the vectors B_1 and B_3. As we are only looking for the input space containing the key pattern in the observations, the true NROY space of example 2 should be the same as for the previous example, which is plotted in Figure 5.5.

The same method for selecting a kernel is used as previously (Section 5.4). We use the same mixture kernel function as numerical study 1, given in equation (5.26). Note that, We specify Σ_e and Σ_η following the previous example. We denote all of the unknown parameters in the mixture kernel as $\mathcal{K}_{par}$, including a weight parameter, $\omega \in [0,1]$, Gaussian kernel parameters, σ, and δ_1, δ_2 used to specify Σ_η. We have $\mathcal{K}_{par} = (\omega, \sigma, \delta_1, \delta_2)$. By applying our optimisation algorithm, we find that the optimal weight parameter is $\omega = 0.00017$, $\delta_1 = 0.216717259, \delta_2 = 0.004824966$, and the optimal Gaussian kernel parameter is $\sigma = 0.01997$. The estimated value of ω is extremely small, largely due to the fact that the maximum value for the linear kernel is 30452.42, whilst the maximum value for the Gaussian kernel is 1. Given these parameters, the mixture kernel function for the toy example can be specified:

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \omega f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}') + (1-\omega)\exp(-(f(\mathbf{x}) - f(\mathbf{x}'))^T \Upsilon^{-1}(f(\mathbf{x}) - f(\mathbf{x}'))/\sigma).$$

In contrast to the first example, the linear kernel is not appropriate for this example because of the moving pattern. Once a linear kernel is adopted, the KHM (equal to standard history matching) will give a wrong NROY space. This situation was demonstrated in Chapter 4.

5.5.1 Wave 1

With the selected kernel function, we calculate the ensemble projections, $\mathbf{C} = (\mathbf{C}_r(\mathbf{x}_1), \ldots, \mathbf{C}_r(\mathbf{x}_n))$ (the dimension of $\mathbf{C}$ is 5×60), by applying the Kernel PCA algorithm. As we introduced before, the location of the pattern is random, the kernel projects these patterns into a feature space where only the presence of the pattern is important, and the KPCA coefficients should be near deterministic when projecting from this space. But when $\omega = 1$, then the projections, $\mathbf{C}$ would be stochastic, and $\mathbf{C}$ is getting more and more stochastic as ω approaches 1. For this example, ω is extremely small, so we treat the $\mathbf{C}'s$ as deterministic and construct univariate Gaussian process emulators for each basis vector. Leave one out diagnostics are plotted in Figure 5.8. We sample a LHC of size 10000 to represent the initial parameter space $\mathcal{X}$. To continue the comparison of KHM with $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$, we perform KHM with both implausibilities. The two NROY spaces are found following these two implausibilities, and are plotted in in Figure 5.9. The

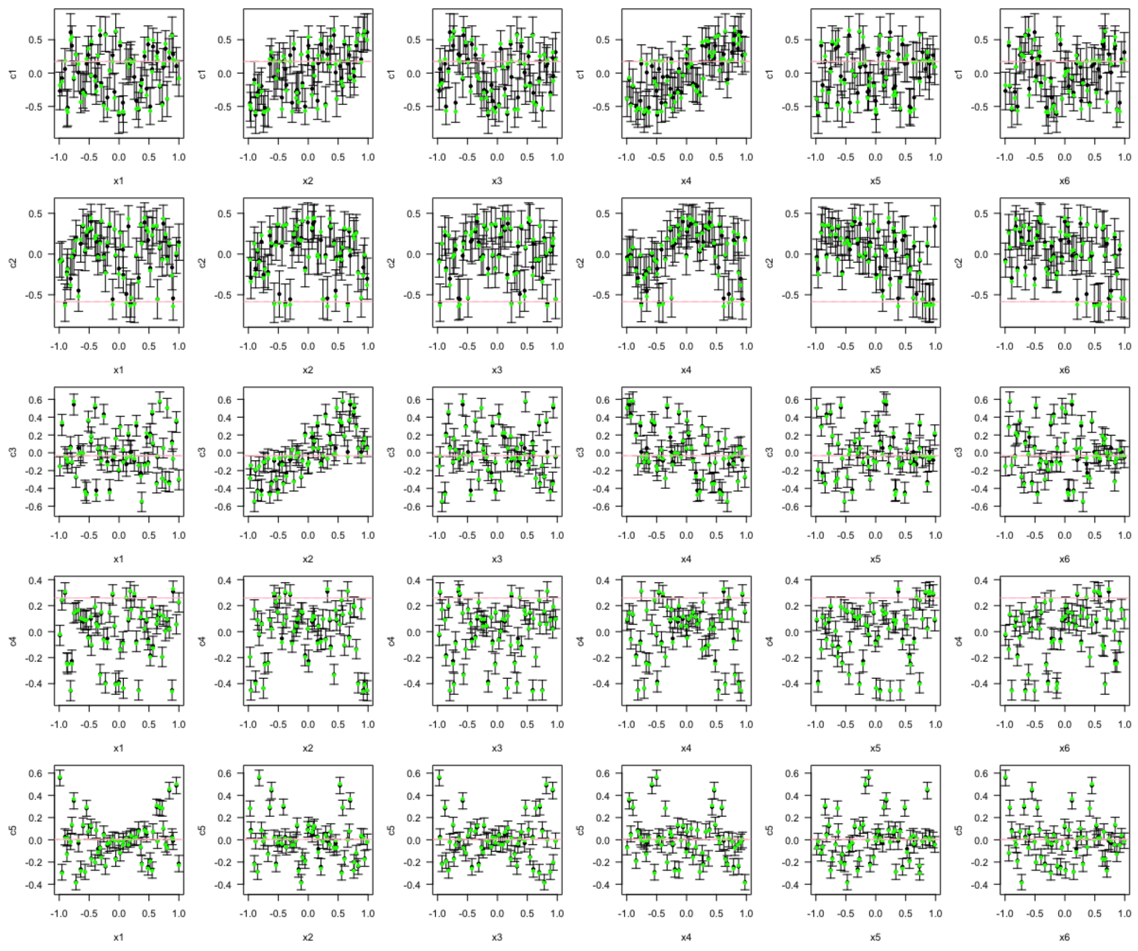


Fig. 5.8 Wave 1 Leave-one-out cross-validation plots for the emulators for the coefficients on the first 5 basis vectors.

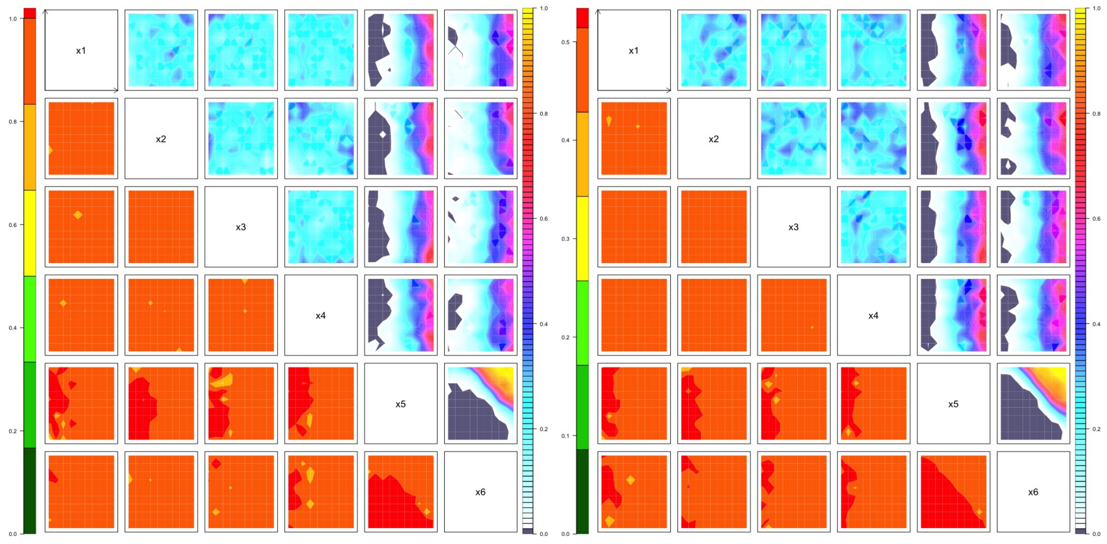


Fig. 5.9 *Left*: Wave 1 NROY space by performing KHM with $\mathcal{I}_{F1}(\mathbf{x})$. *Right*: Wave 1 NROY space by performing KHM with $\mathcal{I}_{F2}(\mathbf{x})$.

results are shown in Table 5.2. The produced NROY space for both implausibilities are similar (with the same emulators): large amounts of parameter space were ruled out with most of the true NROY retained.

5.5.2 Refocusing: wave 2

For consistency with the ensemble for the first wave, the new ensemble for wave 2 is designed to have 60 members. Using this new ensemble and its selected acceptable runs, we can perform kernel selection, emulation and KHM. KHM is most powerful when refocusing steps are taken with different forms of optimal mixture kernel. As later waves are reached, we are able to run the new ensemble in the reduced NROY space, and the number of acceptable runs will increase. Using the expert's judgement in later waves will allow us to produce a better kernel function and more precise threshold value/function. However, if expert judgement is not available in later waves, the optimal kernel function found in wave 1 can also be used to represent the expert's judgement. In this example, we select the best 19 runs as the acceptable runs and use this judgement in the kernel selection algorithm.

Given the new ensemble and the acceptable runs, we perform KHM for wave 2. The optimization algorithm suggests that using our mixture kernel with $\omega =$

	NROY volume	Retained true NROY
$\mathcal{I}_{F1}(\mathbf{x})$: Wave 1	23.36%	99.19%
$\mathcal{I}_{F1}(\mathbf{x})$: Wave 2	20.42%	97.02%
$\mathcal{I}_{F2}(\mathbf{x})$: Wave 1	25.78%	99.14%
$\mathcal{I}_{F2}(\mathbf{x})$: Wave 2	17.78%	96.21%

Table 5.2 KHM results for numerical example 2.

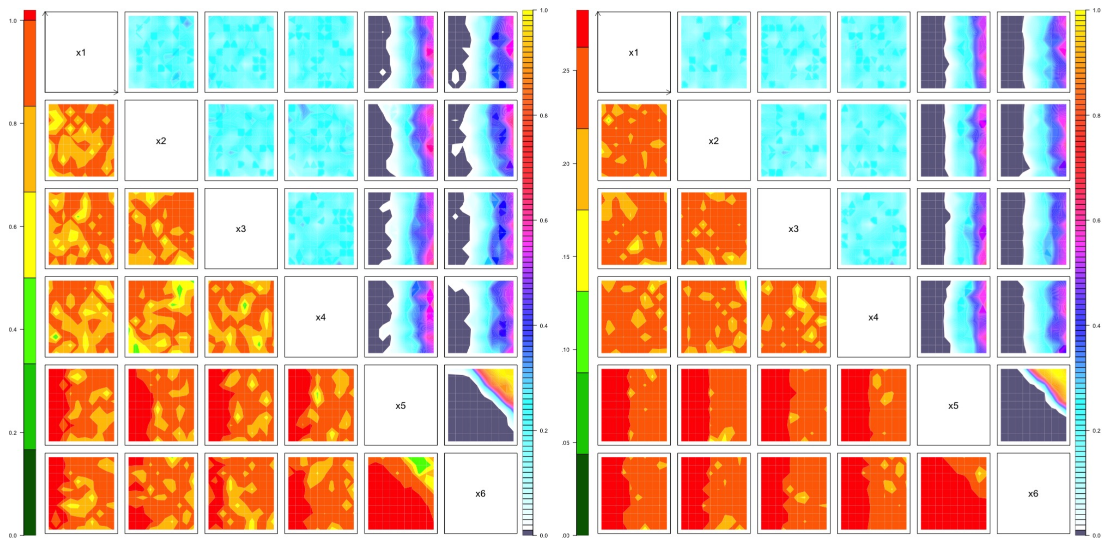


Fig. 5.10 *Left*: Wave 2 NROY space by performing KHM with $\mathcal{I}_{F1}(\mathbf{x})$. *Right*: Wave 2 NROY space by performing KHM with $\mathcal{I}_{F2}(\mathbf{x})$.

0.00731, $\sigma = 0.00014$. In this example, our optimization method suggested two different kernel functions for the two waves. By performing KHM with $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$, we find NROY spaces that are close to true NROY (plotted in Figure 5.10). By performing KHM with $\mathcal{I}_{F1}(\mathbf{x})$ in wave 2, we find an NROY space that is 20.42% of the initial parameter space. Performing wave 2 does not reduce the wave 1 NROY space significantly because the emulator uncertainties for both waves are quite large, giving a large threshold value. KHM with $\mathcal{I}_{F2}(\mathbf{x})$ gives a smaller NROY space but only 96.21% of the true NROY space is retained.

As discussed previously, when the emulator uncertainty vanishes, the expression for $\mathcal{I}_{F1}(\mathbf{x})$ is the same as the expression for $\mathcal{I}_{F2}(\mathbf{x})$, and the threshold will also be the same. Hence, without considering emulator uncertainties, KHM with $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$ will give the same NROY space. In other words, the difference between $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$ is entirely caused by the differences in how we handle emulator uncertainties. For the previous example and this example, KHM with

$\mathcal{I}_{F1}(\mathbf{x})$ always gives a similar result as KHM with $\mathcal{I}_{F2}(\mathbf{x})$ but with more true NROY space are retained. We believe that the bias of KHM with $\mathcal{I}_{F2}(\mathbf{x})$ here is caused by the threshold. As introduced in Section 5.3.2, the computation of the threshold for $\mathcal{I}_{F2}(\mathbf{x})$ is based on the emulator variances of the training data (leave one out emulators used for emulator diagnostics). When the emulator variance for the training data (given by leave-one out emulator diagnostics) is smaller than the emulator variance for the testing data, a low value of threshold $T^{'**}(\mathcal{K}_{par})$ could be given, which would rule out of more space than it's supposed to. Overall, we would suggest that $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$ can both be used for KHM when the emulator's predication is good in general, without particularly large uncertainties. Otherwise, using $\mathcal{I}_{F1}(\mathbf{x})$ for KHM is more conservative and more robust. Our detecting approach and the Local Voronoi Tessellation method in Chapter 3 can also be used for a robust approach to KHM that ensures the true NROY is safely retained.

5.6 Discussion

In this chapter, we introduced an automatic procedure for selecting the optimal mixture kernel for KHM. We developed a mixture kernel function, which opens up to the possibility for different feature spaces to be determined by different kernel parameters. In Section 5.2.3, we proved that KHM can achieve standard history matching with a linear kernel, and the projections of the model outputs are the same as the approach given in Salter and Williamson (2019); Salter et al. (2019). KHM with the mixture kernel function is then a generalised version of traditional history matching.

To determine the unknown kernel parameters in the mixture kernel function, we introduced an optimisation algorithm which can give the optimal mixture kernel for different applications. We developed an optimisation procedure based on expert judgement. KHM can calibrate to the features that the expert wants to see, which gives a good mix of statistical approaches and expert knowledge.

Moreover, the optimisation function not only selects the kernel function, but also the optimal threshold that gives the highest performance evaluation.

In our numerical studies, we built two simple and easily illustrated examples to show how the optimisation algorithm works. Numerical example 1 shows that the algorithm will suggest that a linear kernel is the best choice when standard history matching works (from wave 1 to wave 3). By slightly modifying numerical example 1, we built numerical example 2, which shows that with a suitable kernel, KHM can be used to improve the standard history matching results when key features in the observations we want to replicate exist but in different parts of the output space.

Many possible extensions could be introduced to the presented approach. We use wave 1 ensemble as the training data in the numerical examples to select a suitable kernel function. However, overfitting can happen when we only have a small number of runs. It would be useful to consider in future to evaluate the method out-of-sample.

During our simulation study, we demonstrate that KHM with both $\mathcal{I}_{F1}(\mathbf{x})$ and $\mathcal{I}_{F2}(\mathbf{x})$ have proved successful in 100-dimensional numerical studies. To test our methods with higher-dimensional output, we apply our method to the climate model LMDZ in the next chapter.

Chapter 6

Kernel-based history matching for climate models

6.1 Introduction

Climate models attempt to solve the Navier–Stokes equations on a rotating sphere to simulate the evolution of the Earth's climate Gettelman and Rood (2016). These models vary in their complexity, starting from simple radiative heat transfer models up to global climate models. Hundreds of parameters can be introduced to construct climate models, controlling the behaviour of the atmosphere, oceans and a variety of other processes (Hourdin et al., 2017; Williamson et al., 2017). Before using climate models to produce climate predictions of the future, a parameter calibration (the climate modelling community refers to calibration as 'tuning') step needs to be considered. Tuning is a necessary process that attempts to give values of the model parameters that allows the model to give the best representation of key observations so that we can trust its predictions (Hourdin et al., 2017).

History matching has been applied for many climate models with univariate model output. For example, Couvreux et al. (2020); Hourdin et al. (2020) apply standard history matching to calibrate climate models, aiming to improve and tune boundary-layer cloud parameterisations. By comparing the Single-

Column convection Model used in the global climate model with explicit 3D high-resolution Large Eddy Simulations, the free parameters were calibrated and the model performance was enhanced. More commonly, climate model outputs are high dimensional, spatio-temporal fields. History matching can be applied to this type of tuning problem to calibrate the free parameters in the climate model via a low dimensional basis representation (Salter et al., 2019). For example, Chang et al. (2014, 2016) adopt PCA based calibration to an ice sheet model that has high-dimensional binary spatial outputs.

As we claimed in Chapter 4, standard (PCA based) history matching can fail with high-dimensional model output which is not well approximated by a linear subspace, as we often find in climate models. For example, when considering the simulation of clouds in a single column convection scheme, the features of the evolution of a cloud are more important than the time at which they occur. Climate modellers view a 2D image of relative humidity in column height and time, viewing a cloud as realistic based on its pattern properties, without, for example, being concerned about when during the simulation, cloud formation occurs (e.g. Figure 6.1). We apply KHM to the boundary layer clouds of the French climate model, IPSL-CM, for a spatio-temporal output. To perform KHM on this climate model and to elicit the modeller's calibration target features, we design a new interactive R `Shiny` app to provide an elicitation platform for the expert classification our methodology requires.

This chapter is organised as follows. In Section 6.2, we describe the boundary layer cloud model, the rationale for using it in the climate community , and the data we will use for its calibration. In Section 6.3, we present our R `Shiny` app and its use by an expert at ISPL. In section 6.4, we perform KHM for the cloud model. In Section 6.5, we perform the second and third wave of KHM for this application. The chapter ends with a conclusion in Section 6.7.

6.2 Tuning the boundary layer clouds

Clouds from the bottom of stable boundary layers are called boundary layer clouds. They play an important role in the water cycle, atmospheric energy cycle and global surface temperatures (Bony and Dufresne, 2005). Boundary-layer clouds are much smaller than a grid cell of the climate model, so that their effect on the larger components of the model is accounted for by a parameterization: a mathematical model that represents the physical process (Holtslag et al., 2013; Nam et al., 2012). Each parameterization relies on a set of free parameters used to simulate the effect of the boundary-layer clouds, and therefore, tuning these free parameters is crucial to capturing and improving systematic biases in the global climate model (Hourdin et al., 2006, 2017).

The general approaches to tuning free parameters either follows so called traditional global model tuning or process-based tuning. Global model tuning considers a specific climate model performance metric in the tuning process, such as the temperature. However, Hourdin et al. (2017) state that global model tuning could lead to over-fitting or over tuning and that the good performance of a specific climate model according to global metrics can be achieved via compensating errors. To overcome these issues, Hourdin et al. (2017) suggest adopting process-based tuning at the first stage in model calibration. Process-based tuning uses process-oriented metrics for calibration, such as compositing cloud or precipitation characteristics by dynamical regimes (Bony and Dufresne, 2005). These process-oriented metrics can help relieve large-scale biases in specific subgrid-scale processes of the climate model (Hourdin et al., 2017). The calibration results of process-based tuning will be used for global climate model tuning.

The process-based tuning in the HIGH-TUNE project is based on comparison between single-column versions of the global model (SCM) with explicit 3D high-resolution Large Eddy simulations (LES) of the same boundary layer clouds (Brown et al., 2002; Hourdin et al., 2017). LES are mathematical simulations for turbulence used in clouds dynamics, to derive and evaluate the conceptual models at the root of the boundary layer and shallow cloud parameterizations (Guichard and Couvreux, 2017; Neggers et al., 2009). In fact, LES has been extensively used for evaluating the parameterizations of different cloud properties within many cloud regimes (Couvreux et al., 2005; Hourdin et al., 2002; Rio and Hourdin, 2008). SCMs are single-column models, where a single column is extracted from a 3D climate model, and can be run with the same boundary conditions as an LES simulation (Couvreux et al., 2020; Hourdin et al., 2020). The modellers from the HIGH-TUNE project are interested to find a subset of the input parameters of the boundary layer cloud parameterization scheme that gives SCM output close to LES. In particular, for LES and SCM, there is a selection of cases for different types of boundary-layer clouds such as continental and oceanic cumulus clouds. For this chapter, we consider 'SANDU', which is the composite stratocumulus to cumulus transition cases introduced by Sandu and Stevens (2011). In particular, the simulation outputs for our work are generated from the SANDU/REF case. The SANDU/REF simulation accounts for the increase in sea surface temperatures and reproduces the main features of observed sea surface temperatures in the boundary condition (Sandu and Stevens, 2011).

6.2.1 Simulation outputs

To apply KHM for the spatial–temporal fields output of the SCM, we run the SCM model with the SANDU/REF case and perturb 5 cloud parameters chosen by the modellers. The model parameters to be tuned are identified as `thermals fact epsilon`, `thermals ed dz`, `cld lc lsc`, `rad chaud1` and `z0min`, and the possible range of values were determined by the project. In our analysis we have mapped the parameters onto $[-1, 1]^5$ for fitting emulators and calibration. For the SANDU/REF case, the SCM model response is the cloud fraction from a compact stratocumulus layer to more broken fields of cumulus over 72 hours. The considered case corresponds to the reference simulation, LES, that is performed on a super computer with standard horizontal and vertical resolution. This LES reference is shown in Figure 6.1, which shows the time series of the hourly averages

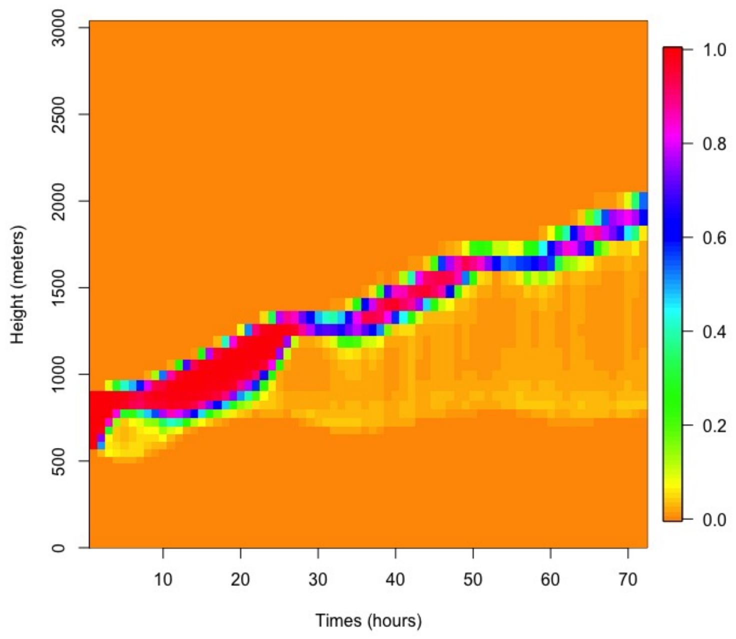


Fig. 6.1 LES reference for SANDU/REF case: time series of the hourly averages of the cloud fraction profiles.

of the cloud fraction profiles at different heights in the model. In this figure, the x axis reflects the time evolution of the cloud fraction (the time scale of the transition), which continues for 72 hours. The vertical axis is the height (meters) above ground level, and the obvious pattern (red and blue curve) is the clouds fraction (values are from 0 to 1). We will refer to this LES reference as the "observation" in the following experiments.

To perform KHM, we use the classical design of computer experiments, a Latin Hypercube (LHC), to generate the SCM ensembles. We start by generating a 3-extended LHC of size 30 following Williamson (2015). The 90-member LHC composed of 3, 30-member LHCs, where each additional LHC ensures that the composite design is orthogonal and fills the space in each extension phase. With all of the generated designs, we evaluate the SCM simulators to give the ensemble for wave 1. The 90 SCM runs are plotted in Figure 6.2. Each plot shows the hourly averages of the cloud fraction profiles during 72 hours of SCM simulation. The bottom axis, vertical axis and the colour patterns are the same as Figure 6.1. As in Chapters 4 and 5, we calibrate the high dimensional output fields via KHM in a feature space, and hence selecting a suitable kernel function is important. To apply the optimization kernel selection method of Chapter 5, we require expert

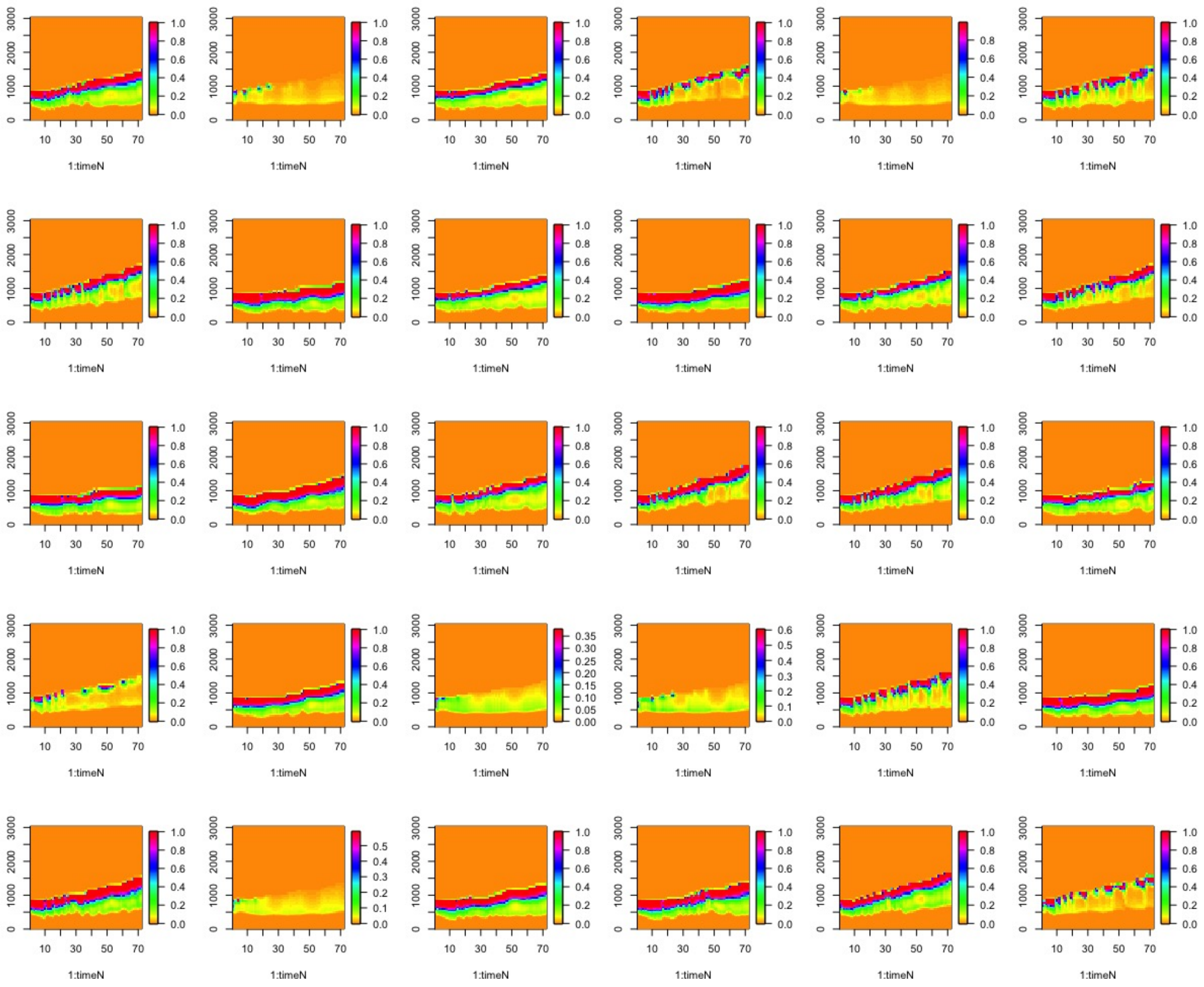


Fig. 6.2 Some wave 1 ensemble runs from SCM simulators: the ensemble outputs are plotted ordinarily from the 1st run to the 30th, the full ensemble that contains the rest 60 runs are plotted in the appendix. For each plot, it shows the hourly averages of the cloud fraction profiles during 72 hours of SCM simulation.

judgement to guide us to understand the important features of these LES clouds that should be captured by the SCM.

6.3 Expert judgement

In the HIGH-TUNE project, Couvreux et al. (2020); Hourdin et al. (2020) have applied standard history matching to improve the representation of boundary-layer clouds, but with scalar metrics at given times or averages over a given period. However the complex metrics in the model output, such as the time series or spatial fields can also be important for climate model tuning. For this application, our collaborators from the HIGH-TUNE project have already tried to reduce the dimensions of the outputs, using standard history matching with PCA (Salter et al., 2019). The project's inability to do this via standard methods inspired our work on KHM.

To perform KHM on the climate model output and to consider the modeller's judgement in the calibration process, we create a `Shiny` APP. We invite the senior model expert (within the IPSL development team), Dr. Frederic Hourdin, to use our app. Our app and Dr. Frederic Hourdin's choices are introduced in the following.

6.3.1 The `Shiny` app

`Shiny` is an R package that enables user to build interactive web apps in R. The purpose of using the app is to guide our experts in classifying fields (cloud patterns shown in Figure 6.2) as acceptable matches to the LES reference or not. There are three pages in the app, page 1 shows the observation or target field and the ensemble member figures, page 2 is the selection page where the experts can choose their acceptable runs, and page 3 is used to do a final check and save the selections. Full details of the App are given in Appendix B. We use this app to obtain expert judgement for each wave.

6.3.2 The expert's selection for wave 1

The cloud pattern is much more complex than the toy model introduced before, and we can likely expect human error to increase in the real application. To determine the degree of human error in the optimization algorithm, we ask Dr. Hourdin to make selections twice for the wave 1 ensemble runs, at different times, with the ensemble members plotted in a different order in the `Shiny` app. Initially, Dr. Hourdin selected 12 runs as acceptable, but in the second time, he had a different choice, where only 10 runs are selected as acceptable. During these two experiments, there are 8 acceptable runs are chosen in both sessions, 4 runs are only chosen as acceptable in the first session and 2 acceptable runs are only chosen in the second session. The result shows that either the expert's true NROY is different at different days and at different times, or that if true NROY exists and is fixed for an expert, during a time limited exercise they are only able to identify members of it with error. This indicates that it may be hard to reach the maximum value of the performance evaluation in our kernel-optimization algorithm. Hence, it is important to set the influence factor in the performance evaluation function (in Section 6.4.1), which can balance accuracy and efficiency. Details will be given when we apply the kernel optimization algorithm. We discuss the issue of the expert's "true NROY" further in the discussion section.

In total, we take all chosen 14 acceptable runs, combining all of the selections made over the 2 sessions (Figure 6.3). The cloud patterns in the acceptable runs are not exactly the same as the observation. To clearly demonstrate this, we plot a cloud fraction for a later hour (time=68) of the simulation with the spread of the ensemble of simulations used for wave 1 in the left panel of Figure 6.4. The wave 1 ensemble is presented in grey, green lines represent the acceptable runs selected by the experts and the reference LES in thick red. The altitude of the SCM runs are generally lower than the observation, and there are some runs which do not contain the cloud pattern. By comparing the green lines with the whole ensemble, we can observe that the patterns of the clouds are all contained in the acceptable runs, even though the patterns do not occur at the same altitude as

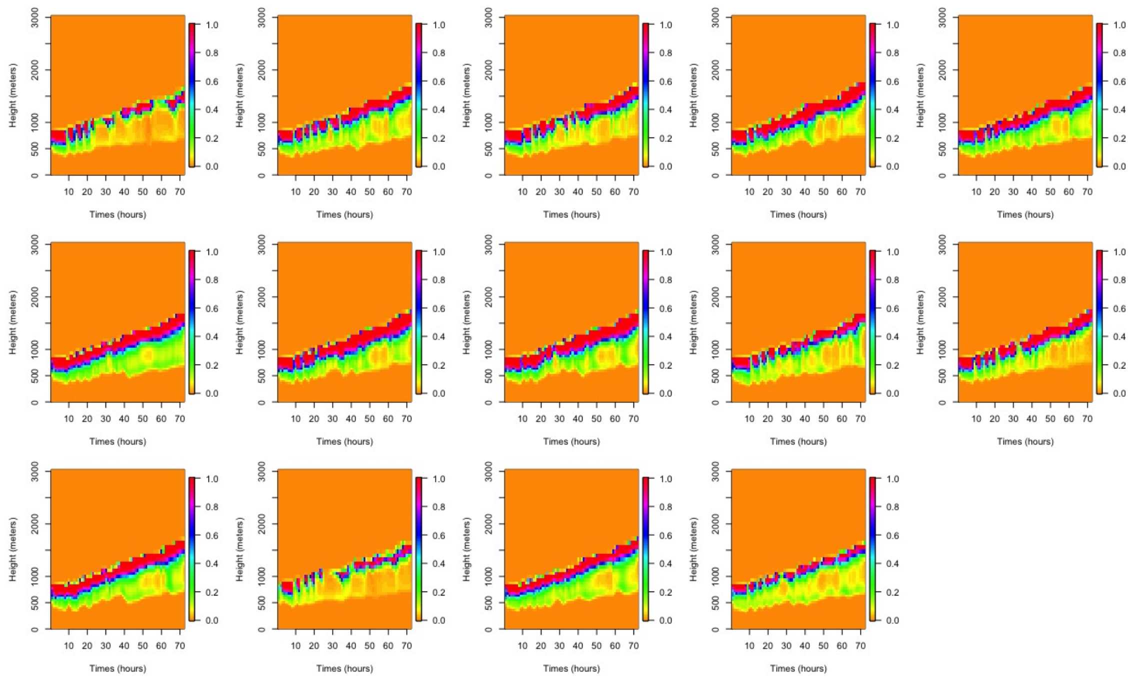


Fig. 6.3 The acceptable runs by expert's selection.

the observation. Moreover, to show the difference between the expert judgement and a pure distance-based judgement, we plot the first 14 'best' runs that are closest to the observation on the simulator output space (only observation error is considered in this distance, discrepancy is not given) in the right panel of Figure 6.4. The blue lines represents these 14 'best' runs, which all have almost no cloud fraction for the later hour. The plot again shows that the model outputs without the key pattern would be seen as the best (when we use the wrong distance). We now use the 14 acceptable runs selected here to perform KHM.

6.4 Kernel-based history matching

6.4.1 Kernel selection

Kernel selection is a necessary step in our approach. To select a suitable kernel function for the climate model, we apply the optimization kernel selection algorithm presented in Chapter 5, with the selected acceptable runs shown in Figure 6.3. To select the kernel function, we first must specify our kernel presented in

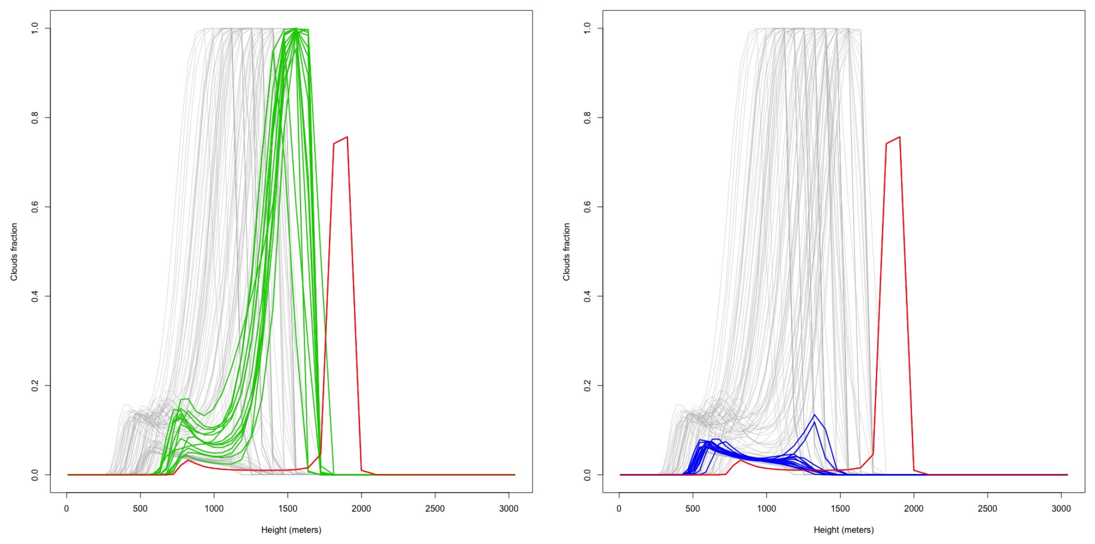


Fig. 6.4 The cloud fraction for the later hour (time=68) of the simulation with the spread of the ensemble of simulations used for wave 1. The wave 1 ensemble is presented in grey, green lines represent the acceptable runs selected by the experts, blue lines represent the first 14 'best' runs that are close to the observation in model output space, and the reference LES in thick red.

Chapter 5. We use the mixture kernel

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \omega f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}') + (1-\omega) g \exp(-(f(\mathbf{x}) - f(\mathbf{x}'))^T \Upsilon^{-1} (f(\mathbf{x}) - f(\mathbf{x}'))/\sigma), \tag{6.1}$$

where ω is a weight parameter, $\omega \in [0,1]$, σ is a Gaussian kernel parameter, Υ is a $l \times l$ positive definite weight matrix defined as the sum of the observation error (LES reference error) variance ,Σ_e, and another variance term, Σ_η, $\Upsilon = \Sigma_e + \Sigma_\eta$, where $e \sim N(0,\ \Sigma_e)$, and g is scale parameter, the value of g is given as the maximum value of the linear kernel of the training data to make the maximum value of both linear kernel and nonlinear kernel (the maximum value of Gaussian kernel is 1) are similar. Note again that Σ_η is not the discrepancy variance as defined by Kennedy and O'Hagan (2001), but it becomes discrepancy variance when $w = 1$. In this example, we follow the definition given by modellers, set Σ_e as a diagonal matrix, and compute the variance of these 2 LES runs as the main diagonal entries of Σ_e. Because the inverse of the 3600×3600 matrix, Υ, are required by each iteration of our optimization procedure, which is a time-consuming calculation, following the setting of Σ_e given by modellers, we also set Σ_η as a diagonal matrix to reduce the expensive computational cost in this example. However, if the computation time is accepted in the real application, the Gaussian covariance function would always be suggested to offer a more flexible structure for Σ_η. We also perform the first wave with different sets of Σ_η in Appendix C.5, a similar NROY space is produced by KHM even with different settings of the kernel function.

The performance evaluation function is

$$\mathcal{P}(\mathcal{K}_{par}) = \alpha\mathcal{A}(\mathcal{K}_{par}) + (1-\alpha)\mathcal{E}(\mathcal{K}_{par}), \tag{6.2}$$

where $\mathcal{A}(\mathcal{K}_{par})$ and $\mathcal{E}(\mathcal{K}_{par})$ were defined in equation (5.17) and (5.19). We set the goal for the first wave to retain all of the expert's choices, and the arbitrary influence factor is set as $\alpha = 0.8$. Note that α can be a very sensitive choice, further discussion will be given in Section 6.7. The optimization algorithm finds that $\omega = 0.92622$ is the best choice for the weight parameter, $\sigma = 0.00366$ for Gaussian kernel parameter, $g = 103928.3$, and the cutoff threshold is suggested as $T = 2353.828$. Moreover, though the value of ω is quite large, this does not mean that the optimal kernel tends to a linear kernel. In fact, due to the large g, the nonlinear kernel dominates.

Given this kernel function, we calculate the ensemble projections by applying the kernel PCA algorithm. The projected ensemble for 90 design points, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, can be written as $\mathbf{C}(\mathbf{X}) = (\mathbf{C}_r(\mathbf{x}_1), \ldots, \mathbf{C}_r(\mathbf{x}_n))$, where the dimension of $\mathbf{C}(\mathbf{X})$ is 5×90. Given the ensemble of the 5-dimensional input parameter space, and the coefficient projection for each $\mathbf{x}$ for each output in feature space, $\mathbf{C}_i(\mathbf{x})$, we build five univariate Gaussian process emulators for the first five basis vectors. Leave-one-out cross-validation plots are shown in Figure 6.5. The black dots and error bars show predictions together with 2 standard deviations from the leave-one-out emulator, whilst the green dots are true model output coefficient projections. From Figure 6.5, we can see there are no missed predictions, implying that the emulators are good representations of the metric of interest.

Through the kernel optimization, the variability explained by the first few basis vectors is not as important as for PCA-based history matching. In standard history

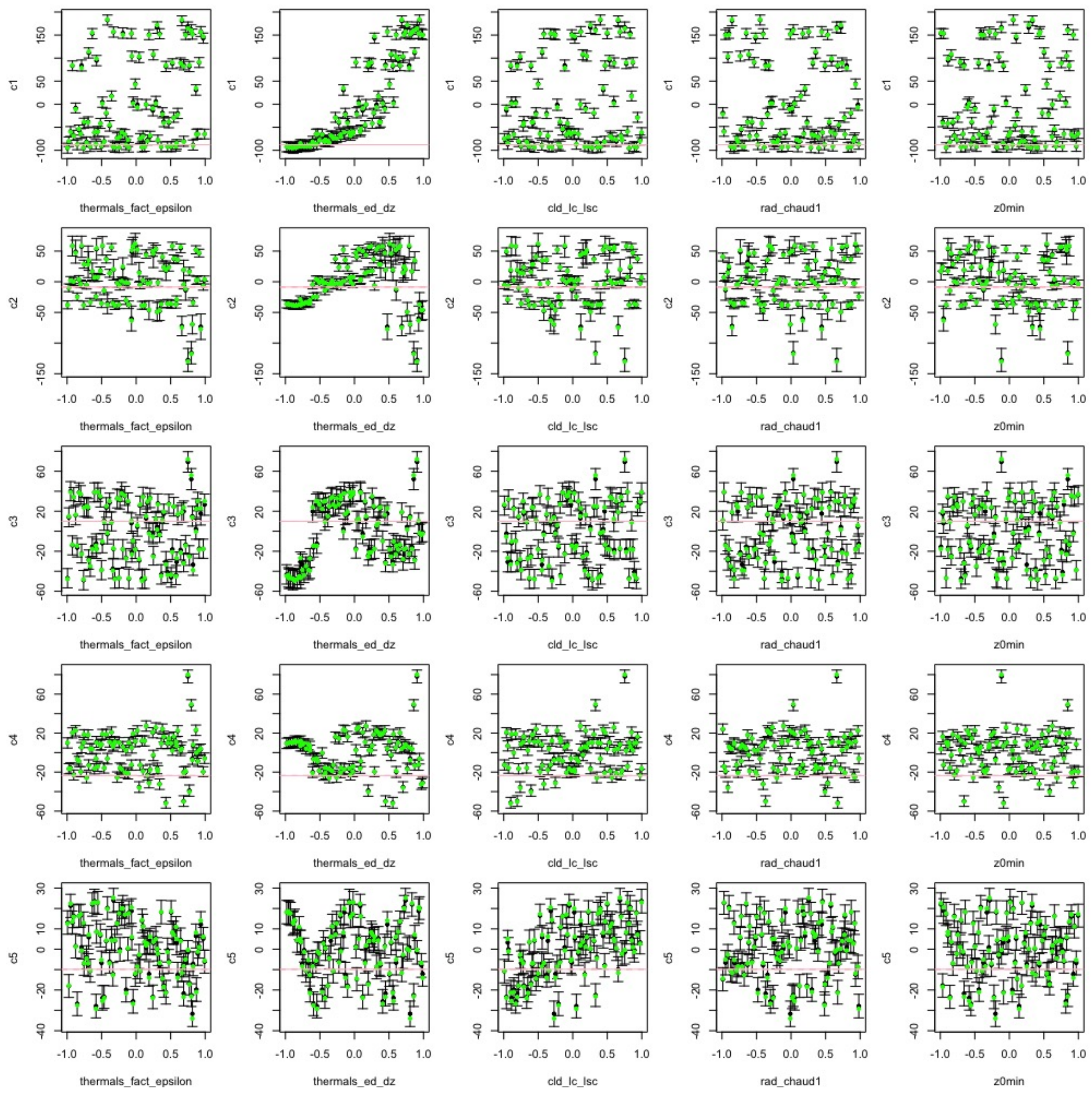


Fig. 6.5 Leave-one-out cross-validation plots: wave 1 Gaussian process emulators for $\mathbf{C}(\mathbf{X})$.

matching, a high value of the variability explained is required to ensure that we do not lose the signal in calibration. But for KHM, we first project the model output into feature space, and the kernel selection algorithm is performed based on the distance between the mapped observation, $\phi(z)$ and the reconstruction of the model output on feature space, $\mathbf{W}_5\mathrm{E}\left[\mathbf{C}_5(\mathbf{x})\right]$, which ensures that the chosen subspace defined by $\mathbf{W}_5$ can extract the key signal that we are calibrating for (see Section 5.3). Therefore, the kernel selection algorithm has already done a good separation of NROY and not NROY on the first few coefficients, and the signal experts want to calibrate is then built into the optimal kernel.

6.4.2 NROY space

We use KHM with $\mathcal{I}_{F1}(\mathbf{x})$ to rule out of regions of parameter space. The NROY space is

$$\mathcal{X}_1 = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_{F1}(\mathbf{x}) \leq T(\mathbf{x})\}, \tag{6.3}$$

where

$$\mathcal{I}_{F1}(\mathbf{x}) = \left(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]\right)^T \left(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right]\right), \tag{6.4}$$

and

$$T(\mathbf{x}) = \sum_{k=1}^{5} \mathrm{Var}\left[C_k(\mathbf{x})\right] + 3\sqrt{2\left(\sum_{k=1}^{5} \mathrm{Var}\left[C_k(\mathbf{x})\right]\right)^2} + 2353.828,$$

as presented in Section 5.4.2. The wave 1 NROY density plots and the minimum implausibility plots for each pair of parameters is shown in Figure 6.6. From the density plots, we observe a strong relationship between two parameters `thermals fact epsilon` and `thermals ed dz`, which indicates the importance of these two parameters for calibration. The minimum implausibility (lower triangle) plots show a similar orientation to the density plots (upper triangle). Red regions in this plot indicate the parameter setting is ruled out.

By performing a single wave, we have managed to cut out the initial parameter space $\mathcal{X}$ and achieved an NROY space $\mathcal{X}^1$ of size 43.01% of $\mathcal{X}$. The relationship between input parameters depicted in the parameter plots can not be judged

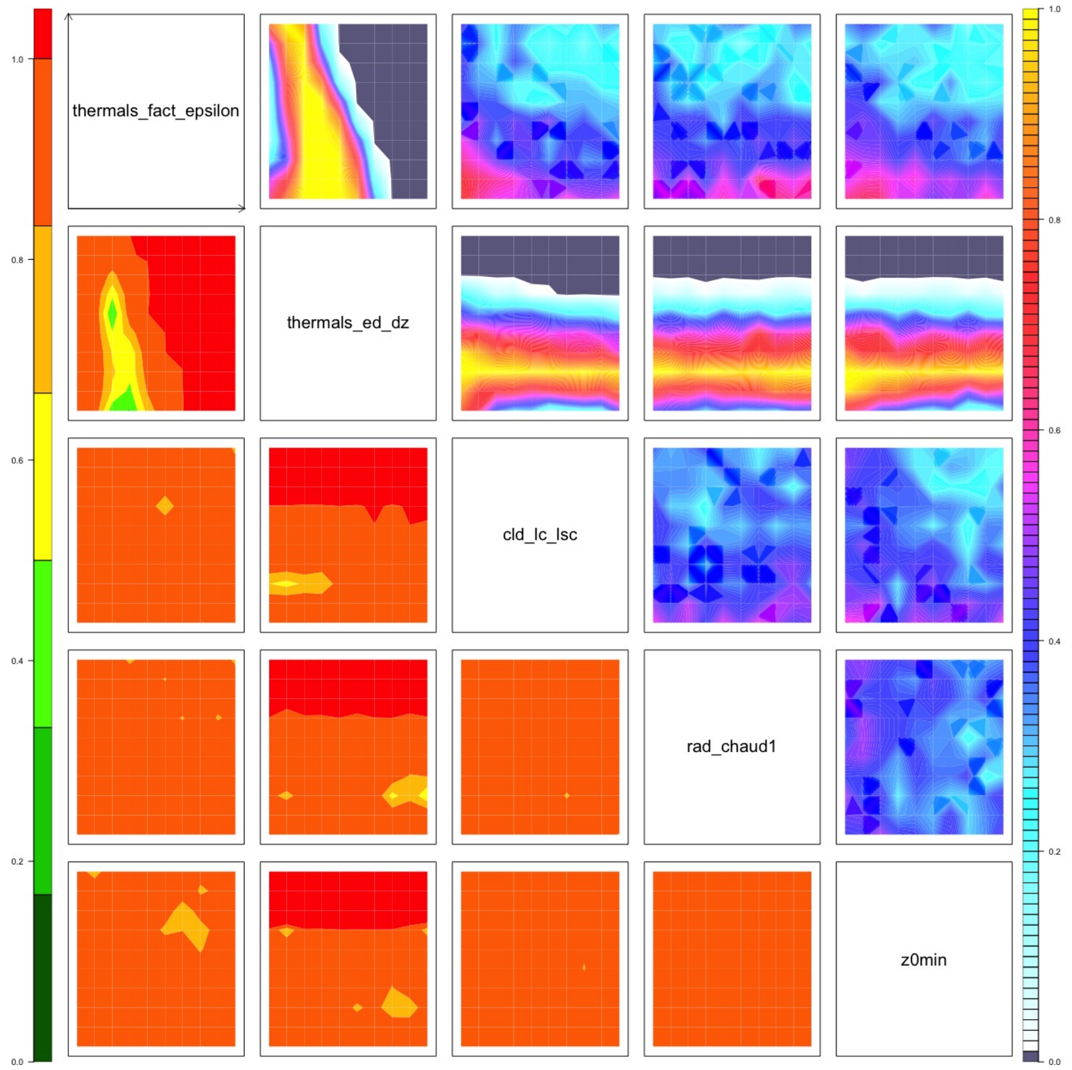


Fig. 6.6 *Upper triangle:* wave 1 NROY density plots for each pair of parameters. *Lower triangle:* minimum implausibility plots for each pair of parameters.

directly with the true NROY space in the real application, but all of the expert's acceptable runs are consistent with the wave 1 NROY space.

6.5 Refocusing

In Section 4.7, we have described KHM refocusing, within the last wave NROY space, a new ensemble is run and the procedure is repeated.

6.5.1 Wave 2 ensemble

To perform a second wave, a new ensemble is required. For consistency with the first wave ensemble, we select 90 members for the new ensemble. The new ensemble design, $\mathbf{X}_2$, is randomly generated from wave 1 NROY space $\mathcal{X}^1$,

$$\mathbf{X}_2 = (\mathbf{x}_{2,1}, \ldots, \mathbf{x}_{2,90})^T \in \mathcal{X}^1,$$

and then running the SCM at the design to generate

$$\mathbf{F}_2 = (f(\mathbf{x}_{2,1}), \ldots, f(\mathbf{x}_{2,90})).$$

The new ensemble members are presented in Appendix C.5, and we plot the cloud fraction for the later hour (time=68) of the simulation with the spread of the ensemble of simulations used for the different waves in Figure 6.7. By comparing Figure 6.7 with Figure 6.4, we can see that the ensemble for wave 2 is closer to the acceptable runs in wave 1. Although there are still some runs with no patterns, the simulator does perform better in the NROY space as compared to the initial parameter space, which indicates that the NROY space, $\mathcal{X}^1$, is closer to the true NROY space.

As we discussed in Section 4.7, the expert's selection criteria might be different in later waves. In the initial wave, the simulator is run within the whole parameter

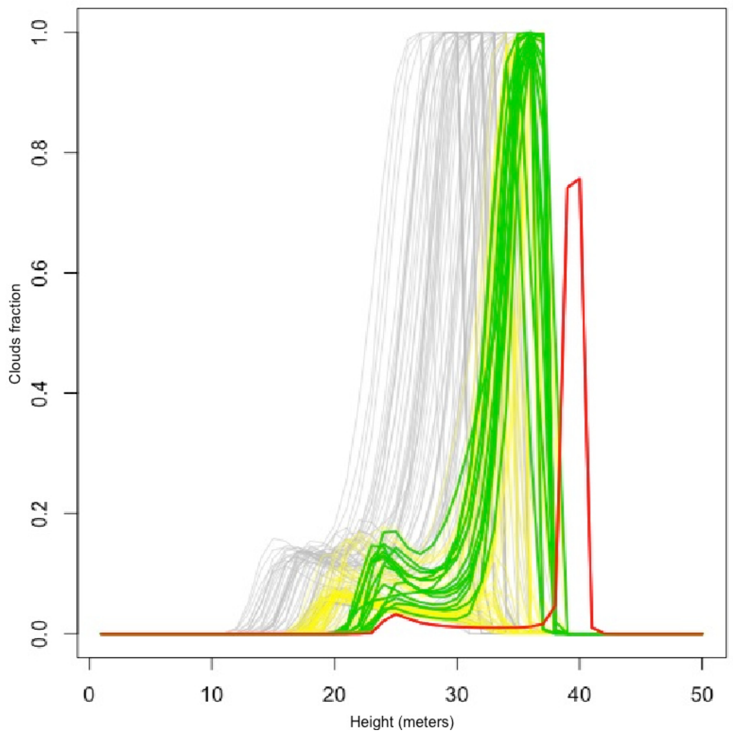


Fig. 6.7 The cloud fraction for the later hour (time=68) of the simulation with the spread of the ensemble of simulations used for the different waves indicated in different colours. The wave 1 ensemble is presented in grey, the wave 2 ensemble is presented in yellow, wave 1 acceptable runs are in green and the reference LES in thick red.

space, and we might expect all of the runs in wave 1 not to look good enough, so that the acceptable runs in wave 1 might be unacceptable when the probability of containing good runs becoming higher in later waves. To perform the kernel optimization algorithm, we asked Dr. Hourdin to select the new acceptable runs for wave 2. By using our R shiny app, we have 9 acceptable runs for wave 2, which are plotted in Figure 6.8. Note that, in this wave, we did not asked the experts to select twice, but the human error is still likely to be present in there.

Using the new ensemble, $\mathbf{F}_2$, and the acceptable runs we can perform KHM for wave 2. In standard history matching, the wave 1 training data retained in the wave 1 NROY space are added to the wave 2 ensemble to improve emulation performance (see Salter et al. (2019)). Following this suggestion, we add the acceptable runs from the previous wave into wave 2 to help to find a good kernel and build a good emulator. However, because the expert did not classify the wave 1 runs, the wave 1 acceptable runs are no longer seen as acceptable in wave 2. To

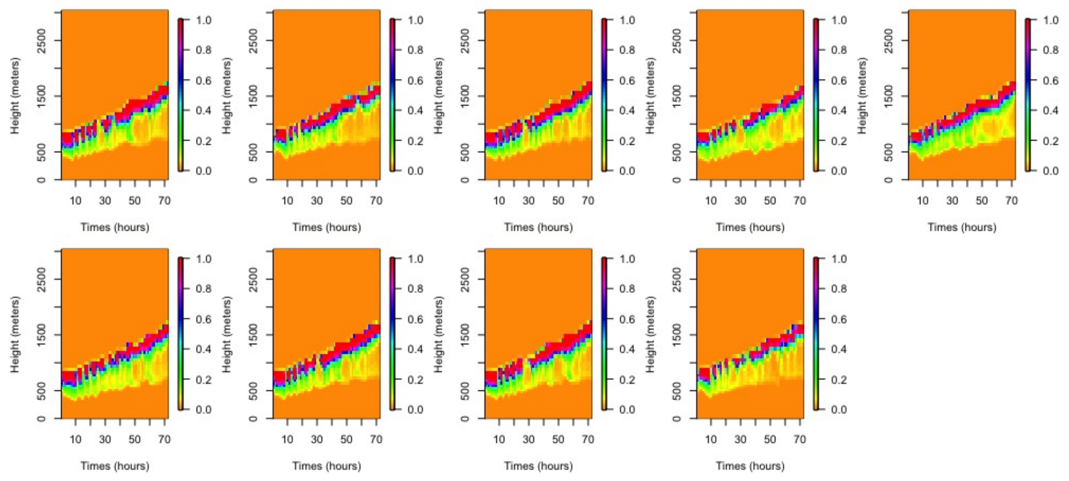


Fig. 6.8 The acceptable runs by expert's selection for wave 2.

use wave 1 acceptable runs, we include these runs into the first two steps of kernel selection procedure (given in Section 5.3.3), but only use the wave 2 training data to perform step 2 and 3 where the expert judgement is required.

6.5.2 Wave 2 NROY space

To select a suitable kernel function for wave 2, we apply the kernel selection algorithm used in wave 1. The algorithm finds $g = 10412.98$, $\omega = 0.321491060$, $\sigma = 0.002003871$ and the cutoff threshold without emulator uncertainty is suggested as $T = 1415.613$. The wave 2 optimization results are very different from the wave 1 results due to the big difference between wave 1 and wave 2 training data, as we presented in Figure 6.7.

The major aim of the kernel selection is to find a kernel that can classify the expert acceptable runs and expert unacceptable runs of each wave. In wave 1, all of the signal from model outputs were considered in the wave 1 kernel selection algorithm, but there are many discarded signals that were ruled out in wave 1, and they do not need to be considered in the following wave. Moreover, with the improvement of the model outputs from wave 1 to 2, the expert's standard for 'acceptable runs' might be stricter. With different training data, potentially different signals and an altered objective function, under normal circumstances, the kernel function should be different between waves. Note that, a special case

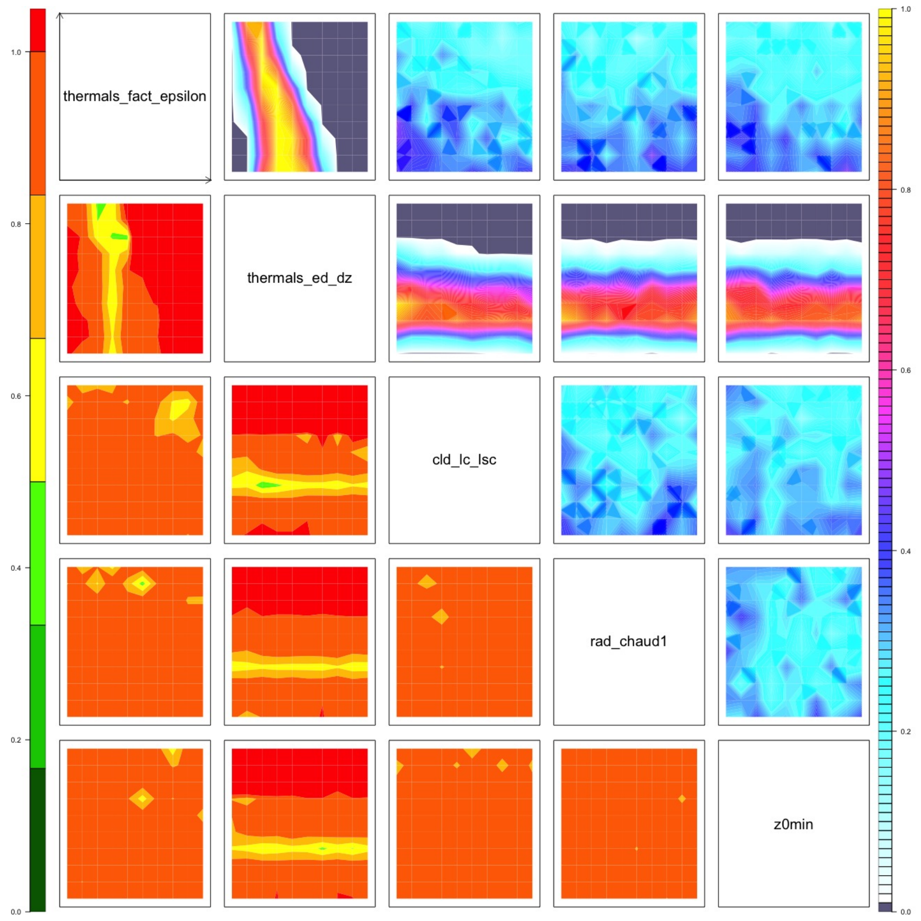


Fig. 6.9 *Upper triangle:* wave 2 NROY density plots for each pair of parameters. *Lower triangle:* minimum implausibility plots for each pair of parameters.

was introduced in Appendix C.2, when the model output space is the best choice for feature space (standard history matching is suitable), the linear kernel was always suggested as the optimal kernel function through 3 waves.

Given the ensemble of the 5-dimensional input parameter space and the optimal kernel function, we calculate the ensemble projections, $\mathbf{C}_r(\mathbf{x})$. We build five univariate Gaussian process emulators for the first five basis vectors. Leave-one-out cross-validation plots are shown in Figure C.20. We use KHM with $\mathcal{I}_{F1}(\mathbf{x})$ in equation (6.4) to rule out of regions of parameter space.

The wave 2 NROY density plots and the minimum implausibility plots for each pair of parameters, is shown in Figure 6.9. From the density plots, the strong relationship between two parameters `thermals fact epsilon` and `thermals ed dz` is similar to that seen in the wave 1 NROY density plots. Starting from the wave 1 NROY space consisting of 43.01% of initial parameter

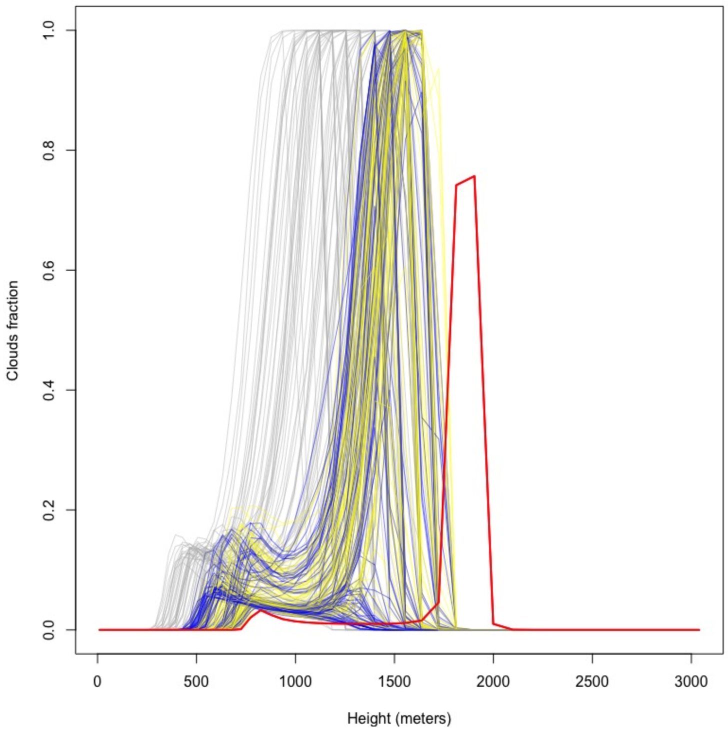


Fig. 6.10 The cloud fraction for the later hour (time=68) of the simulation with the spread of the ensemble of simulations used for the different waves indicated in different colours. The wave 1 ensemble is presented in grey, the wave 2 ensemble is presented in blue, wave 3 ensemble is presented in yellow, and the reference LES in thick red.

space $\mathcal{X}$, 56.99% of the space is ruled out here. The NROY space after wave 2 contains 28.40% of the initial parameter space $\mathcal{X}$, and around half of Wave 1's NROY space, $\mathcal{X}^1$.

6.5.3 Wave 3 NROY space

Given the new ensemble (plotted in Appendix C.5), a new wave of KHM can be performed. We plot the cloud fraction for the later hour (time=68) of the simulation with the spread of the ensemble of simulations used for the different waves in Figure 6.10. The wave 2 ensemble is presented in blue and the wave 3 ensemble is presented in yellow. By comparing the wave 3 training data with the reference LES, we can see that the wave 3 simulator performs better than wave 2. Most runs with no pattern were ruled out in wave 2.

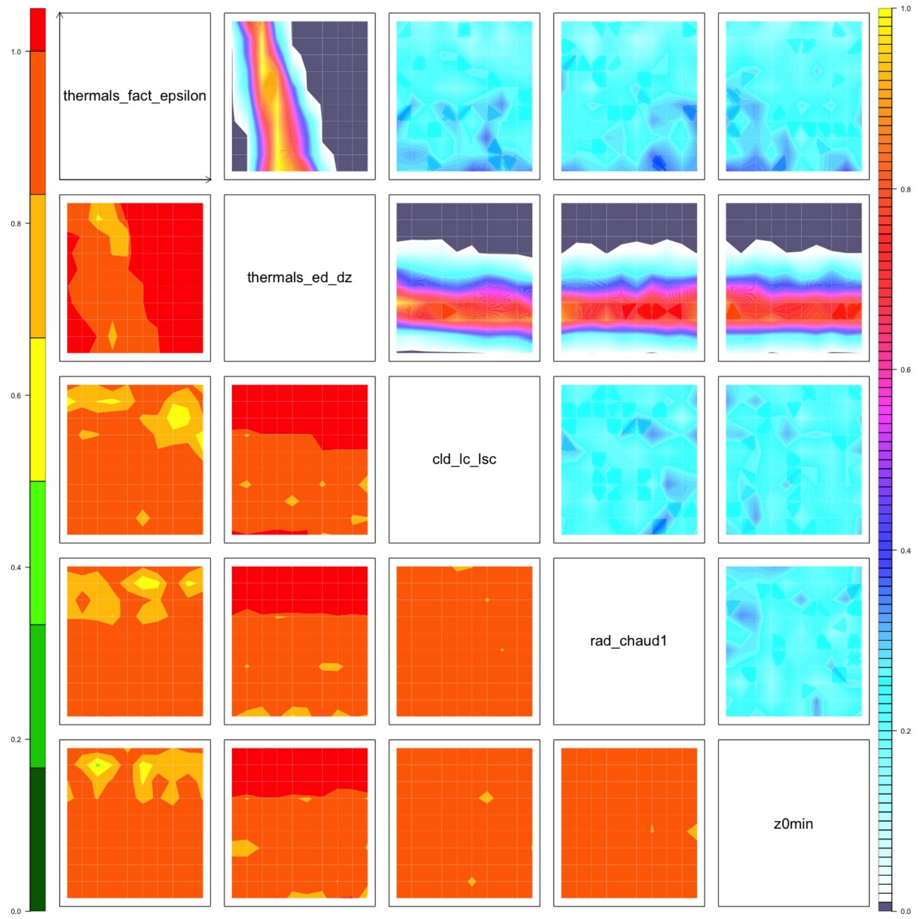


Fig. 6.11 *Upper triangle:* wave 3 NROY density plots for each pair of parameters. *Lower triangle:* minimum implausibility plots for each pair of parameters.

As in previous waves, given the expert's selection, we perform KHM with a specified kernel function determined by the kernel selection algorithm. The requirements for the kernel selection algorithm are consistent with wave 2, and we find that $g = 10412.98$, $\omega = 0.321491060$, $\sigma = 0.002003871$ and $T = 1415.613$ is suggested by the algorithm. Using our emulators, and the implausibility bounds, an NROY space for wave 3 can be defined. We plot the NROY density plots for each pair of parameters in Figure 6.11. Performing wave 3 does not provide us with a significant reduction in NROY space: the NROY space after wave 3 is 23.99% of the original parameter space. The strong relationship between `thermals fact epsilon` and `thermals ed dz` still exists.

6.6 Discussion

In this chapter, we demonstrated that our method scales to important real-world examples. We applied KHM to the French climate model, IPSL-CM, with large output fields that are typically seen in climate model calibration. We presented an interactive app to perform the expert's selection of acceptable runs. The application introduces complexity not considered in our toy example: clouds fractions have various patterns, the expert's selections are incoherent across the three waves, and the expert judgement are different for the same training data on different days. This brings up an important question for our method: what does the true NROY space mean for KHM? In Chapter 5, we defined the true NROY space as the expert's NROY space, but in this application, the same expert has made two classifications for our wave 1 ensemble members.

To complement the definition of true NROY space for KHM, the philosophy behind our approach needs to be discussed. Seen from our practical situation, we have asked Dr. Hourdin to make selections twice for wave 1, at different times (different days), with the ensemble members plotted in a different order in the `Shiny` app, and his acceptable set different between exercises. He selected 8 runs in both sessions as acceptable runs, 4 runs are only chosen as acceptable in the first session and 2 acceptable runs are only chosen in the second session. Based on reality, if true NROY exists and is fixed for expert, an obvious question needs to be answered: how could the expert be wrong when the true NROY space belongs to the expert?

There are several possible explanations for this question. The first possibility is that the expert was wrong, the expert does have a true NROY, but there is some "error" in their selections. This "human error" could be using our new app, within a time limited exercise the expert is only able to identify members their NROY with error. Also as we introduced, during the two sessions, the order of the ensemble runs were different. Since there are 90 runs, the expert need to do 90 comparisons and it is easy to get tired for later runs, leading to error. Moreover, the error can be caused by psychological reasons. How a person approaches the testing being performed is highly important to the results, and differences in accuracy can be caused by differences in model. Overall, we do not think these kinds of errors can be eliminated, but they might be reduced by improving the design of the app.

Another possible explanation is that the expert's true NROY space could be different at different days and at different times. This phenomenon is common in the psychology field and also in life: people's opinion can change as time goes by. For the same wave, there are many plausible mechanisms for these evolving opinions, e.g prior experience, with the tool leads the classification to become more accounted, or something may have occurred to the expert believes session leading them to value/penalise certain features in a different way. We believe this could be the reason why expert's selections are incoherent across the three waves. As we claimed before, the expert might have a strict requirement for acceptable runs in later waves when more of the "good" runs appear, but a loose requirement might be used in the initial wave when no "good" runs or very few exist in the ensemble. In fact, this is a strength of the multi wave history matching approach. There is no exact standard definition of "acceptable", it is a subjective judgement. The aim of our methodology is to find an NROY space that is close to the expert's

NROY, and we believe that our method delivers this. The question raised above are important and would be an interesting area of future works.

6.7 Conclusion

By performing KHM with the French climate model, we show that our method can be applied in important practical problem. It can be seen that our method approaches the experts true NROY space by comparing the last wave ensemble with acceptable runs selected at the beginning. If it were possible, dividing the available runs into more waves may have provided better results.

Some important questions remain open. The first one is about the expert's judgement and true NROY space, as discussed in the last section. With the different acceptable runs for the same training data, the value of the influence factor, α, is important to the kernel optimization algorithm. KHM performance is sensitive to α, with a big value of α, the accuracy of KHM will be more important than efficiency. In the application, we set the goal to retain all of the expert's choices in the NROY space, an arbitrary choice, $\alpha = 0.8$, was adopted. However, our choice may not be optimal. Keeping all of the expert's acceptable runs in NROY space may not be best when we know there are errors in the expert's classification. Moreover, we find that a high value (bigger than 0.9) of the objective function, $\mathcal{P}(\mathcal{K}_{par}, T)$ can be reached when we set a super small α (e.g $\alpha = 0.05$). The kernel can pass the selection step by retaining only parts of acceptable runs in such a situation. However, we did not find this situation with a reasonable choice, such as 0.7 and 0.8. Besides α, there are many other sensitive parameters in the selection step, such as the number of the basis, r, the maximum time, and the maximum performance evaluation we are willing to accept as a stopping criterion. More investigation and comparison of these parameters would be worth studying in further work. In future, we plan to update the interaction app by considering the subjective assessment error and adding more visual control. In particular, we find that our optimization algorithm is not efficient enough, taking a long time in the climate model application, meaning that we should prioritise developing a more efficiently optimised kernel selection algorithm.

Chapter 7

Conclusion

In this thesis, we identify a range of previously unexplored flaws relating to standard history matching for calibrating computer models. These flaws can lead to biased parameter inference and we hence developed methods to robustify history matching in order to overcome these limitations.

The limitations we have addressed were identified through collaboration with HIGH-TUNE project. The first limitation of standard history matching was addressed in Chapter 3. We observed the inadequacy of history matching for retaining good parameter choices in the NROY space, when the emulator does not simulate the target NROY space accurately enough. This can happen even if the emulator passes standard diagnostic checks on the whole parameter space. There was no existing diagnostic to indicate whether this situation could be occurring. We developed a two-step approach; a detect step is introduced based on standard diagnostics. We then presented a robust history matching method to identify the region where the emulator is failing, but is close to the target NROY space and isolate it so that the rest of the input space can be calibrated by history matching with an emulator as normal, without the need for bespoke analysis. This last point is important to users such as HIGH-TUNE project, where many emulators are fit automatically by the modellers themselves, and bespoke emulators are not feasible without statistician involvement. We demonstrated the accuracy of our approach using two illustrative examples and through the output of climate models from HIGH-TUNE. When comparing our method with standard history matching, we find that our method can detect the limitation in the cases efficiently and the NROY space found by our method is more accurate than standard history matching.

In Chapter 4, we identify a further limitation of standard history matching for tuning computer models with high-dimensional output. We demonstrated that if the position of the key signal/current is not fixed in the model output space, then standard history matching (PCA-based history matching) is not able to calibrate these features, which then leads to incorrect parameter inferences. To overcome this limitation, we introduced kernel methods into history matching. A kernel function is adopted to project the model output into a higher-dimensional feature space, where the features that the modeller wants to calibrate can be compared to the reality, even if the locations of the feature are not fixed in output space. To perform history matching in a feature space, uncertainties that belong to model output space (observation error and model structural error) need to be accounted for. We introduced three different models to project uncertainties into the feature space. By trialing our proposed methods on a toy example using a comparison study, we argued that kernel-based history matching (KHM) is the most natural and most credible way to perform history matching in a feature space, as uncertainties are quantified through the kernel. Unlike standard history matching where the judgement of a model run being ‘close’ to the data or ‘far away’ is part of the implausibility function, KHM captures these judgements in the kernel, and only the emulator variance is computed in our new defined feature space implausibility function (or the cut-off threshold function).

In Chapter 5, we developed an automatic kernel optimization procedure for KHM to provide tailored kernels for different applications. We introduced a mixture kernel function, as the combination of a linear kernel and a non-linear kernel. Some unknown parameters are introduced in this mixture kernel, e.g. the influence factor. A suitable kernel can be determined by optimising these parameters. In particular, we use a weight matrix in the mixture kernel to carry uncertainties, which is set as the sum of observation error variance and an unknown term, Σ_η. Giving a potential model for Σ_η (we used the Gaussian covariance function), unknown parameters within it can be optimised away with the other kernel parameters through our kernel optimization procedure for any given application. In Section 5.3, we introduced a new history matching performance evaluation function as the objective function for our kernel optimization procedure. History matching accuracy and efficiency were considered in this objective function, and a weight factor, α, is introduced to provide a compromise between accuracy and efficiency. In our optimization algorithm, we use the judgement of the expert/modeller to evaluate KHM's performance, which allows the modeller to provide their feedback on the key features that determine whether model is close or not. When the performance evaluation function is maximised, KHM would provide the expert's exact (true) NROY space (without simulation outputs). Hence, the aim of kernel optimization is to enable KHM to produce the same NROY space as the expert's NROY, to as close a degree as possible.

Combining the methods that were introduced in Chapter 4 and Chapter 5, we now have a complete calibration method. KHM is a generalisation of standard history matching that reduces the simulator's input space by identifying and discarding input space that is unlikely to provide good model outputs relative to the expectations of an expert. History matching for spatial–temporal fields considers the distance between model outputs in their own space. The generalised approach is to define this distance via a bespoke kernel-based inner product. This inner product represents the dot product in a feature space so effectively KHM is projecting outputs and data into this space for comparison. The generalisation to history matching is to compare output with data in the most relevant space and to have the expert help with defining that space, for example, we achieve that with the kernel selection algorithm.

Standard history matching considers implausibility to be the distance between model outputs of a model at $\mathbf{x}$ and observations, with all sources of uncertainties accounted for (discrepancy variance, observation error variance and emulation uncertainty). Our generalised approach put the uncertainties that belong to model output space into the kernel function, model outputs and data are scaled by these uncertainties through the kernel projection. We defined the implausibility for KHM as the L_2 distance between mapped model outputs and mapped observations in feature space. As the emulators are constructed in feature coefficient space to represent the mapped model outputs, we explored two interpretations of the implausibility with the coefficient emulator prediction: $\mathcal{I}_{F1}(\mathbf{x}) = (\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right])$, and $\mathcal{I}_{F2}(\mathbf{x}) = (\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right])^T(\mathbf{1}_D + \mathrm{Var}\left[\phi(f(\mathbf{x}))\right])^{-1}(\phi(z) - \mathrm{E}\left[\phi(f(\mathbf{x}))\right])$. In particular, KHM with $\mathcal{I}_{F1}(\mathbf{x})$ uses a different approach to standard history matching that defines the implausibility relative to emulator uncertainty with a fixed threshold for cutting space. We defined the implausibility, $\mathcal{I}_{F1}(\mathbf{x})$, as the difference between emulator prediction and observations without the emulator uncertainty, and the threshold for $\mathcal{I}_{F1}(\mathbf{x})$, as a function of $\mathbf{x}$, $T(\mathbf{x})$, that accounts for the emulator uncertainty. For the second approach, we defined a new notion of distance for implausibility, the difference between the emulator prediction and the observations is scaled by the emulator uncertainty. Because there are no obvious statistical properties for $\mathcal{I}_{F2}(\mathbf{x})$, the threshold cannot be set via a statistical heuristic as with standard history matching (e.g. the 3-sigma rule). We choose the threshold T as a level of implausibility that considers what "too far" means based on expert judgement. We demonstrate the efficiency of both methods.

Standard history matching attempts to identify target NROY space based on a best input assumption, the model output at the best input, $\mathbf{x}^*$, with the observation error, e, and model discrepancy, η, is consistent with the real-world observation, z. Particularly, η is a judgement of how different the model outputs are allowed to be from each other in the model output space, which represents the elements which experts want to see in the model outputs. If the belief of model structural error is given, then the discrepancy can be specified, as a 'weighting' on different grid cells of the model output space. However, the difference between the computer model and reality is not meaningful in certain applications, e.g. the moving patterns. Therefore, KHM does not make a best input assumption, instead, it tries to find the regions of input space that corresponding to the model expert's acceptable matches given an uncertainty specification. The true NROY for KHM, a region of good parameter settings, is defined as the modeller's NROY space. The beliefs about model discrepancy (or tolerance to it) are encoded in the expert's judgement. When the linear kernel is selected in our kernel optimization procedure with the given expert's judgement, we showed that Σ_η will correspond to what we normally think of as model discrepancy variance, following the definition given by Kennedy and O'Hagan (2001) and KHM would be the same as standard history matching in this special case. Otherwise, Σ_η is only a part of the weight matrix in the kernel. We claim that the difference between model outputs and the real world processes are only comparable in a feature space.

There are several possible extensions to the methodology we have developed. Firstly, the choice of basis vectors in the current KHM is based on a requirement that the ensemble variance is well-explained by the basis vectors. This method was commonly used for PCA based approaches; a high value of the explained variability ensures that we do not lose the signal in calibration. However, our optimization algorithm ensures that the chosen subspace, defined by the selected basis vectors, can extract the key signal for which we are calibrating, so that the variability explained by the basis vectors is no longer as important in KHM. Moreover, for the kernel PCA based method, the explained variability is computed for the mapped ensemble in the feature space, rather than the initial ensemble. A significant number of basis vectors could be required to achieve a high percentage of the explained variability in the mapped ensemble. While we follow the suggestion given by Higdon et al. (2008), to not take more than 5 basis vectors in practice, due to the fact that accurate emulation for later coefficients is difficult to achieve, this suggestion is not necessarily the most effective method. A possible extension to the research is to consider emulation performance as a factor in the kernel selection algorithm. The optimization algorithm would then guarantee, not only calibration performance, but also accurate emulation.

In Chapter 6, we applied KHM to IPSL-CM, a French climate model with a large spatio-temporal output. We developed a new R `Shiny` app to collect the expert's judgements, so that our kernel selection algorithm could be applied to these judgements. We then performed three iterations of KHM for this climate model. The results show that our method identifies the expert's true NROY space by comparing the last wave ensemble with the acceptable runs selected at the beginning. However, there are two unexpected problems which occurred in this application. First, unlike the numerical examples, the cloud fraction element was more complex, which introduced 'human error' into the expert judgement. We have provided a detailed discussion on Section 6.6, as to how a significant human error in the expert judgement can mislead our kernel optimization algorithm. A possible solution is to increase the visual control in the R `Shiny` app to enhance the expert's experience and to improve the users' interaction. As discussed in Chapter 6, we believe the level of this human error can be reduced, however it may remain non-negligible, meaning that estimating the value of the error is also important. This can be achieved by improving our experimental design, adding a quantitative step for human error. For example, we could randomly select 20% to 40% of runs that are considered twice in the same expert selection session. This would enable possible human error to be estimated by the selection results for these runs. A suitable experiment of design can be developed to increase the rigor, predictability, and efficiency of our app development process.

Once the human error can be captured, the setting of the influence factor of the KHM performance evaluation, α, could also be developed. In the application, an arbitrary choice, $\alpha = 0.8$, was adopted to give more weight to the accuracy of KHM, rather than efficiency. However, there is no evidence that our choice is sufficient. If the value of human error in the expert judgement can be estimated, we could use it to compute the maximum possible value for the accuracy function. A more plausible α can then be determined.

This climate model is part of our collaboration with the HIGH-TUNE project. Our collaboration involves providing methods to both emulate and history match to a large number of process-based metrics, rapidly and automatically, so that the modellers can use the tools independently. However, the current optimization algorithm is not efficient enough, taking a long period of time in the climate model application due to the high-dimensional (3600 dimensional) computation. For each iteration of the optimization algorithm, an inverse of a 3600×3600 matrix is computed, which is time-consuming. We currently do not have solutions to solve this high dimensional matrix inverse problem, however it is a worthwhile area of research to be explored in the future. In addition to this, KHM works similarly to standard history matching, apart from the kernel optimization step. Given a high value of the performance evaluation, selecting a suitable kernel also costs the most computational time. The kernel selection step usually takes hours (1-2 hours for numerical examples, 3-6 hours for the climate model) to find an appropriate kernel function. The actual application takes longer than the numerical studies because of the higher-dimensional outputs. From our experience, simulated annealing is time costing. There are a number of optimization algorithms that could be explored to reduce the computation time in the future, such as the sparrow search algorithm (SSA) (Xue and Shen, 2020).

Overall, I have explored only a selection of areas relating to the current calibration approach, and I believe there remains an extensive field of available potential research. For example, within this thesis, we only identified flaws with the standard history matching, however these limitations could also exist for the Bayesian calibration, and applying the Bayesian calibration on feature spaces could be a further direction of study. The success of our applications assures me that our proposed method provides a way to robustly and automatically calibrate computer models.

References

MILTON Abramowitz. l. a. stegun, 1972: Handbook of mathematical functions. *National Bureau of Standards Applied Mathematics Series*, 55:589–626, 1985.

Md Ashad Alam and Kenji Fukumizu. Hyperparameter selection in kernel principal component analysis. 2014.

Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

Ioannis Andrianakis and Peter G Challenor. The effect of the nugget on gaussian process emulators of computer models. *Computational Statistics & Data Analysis*, 56(12):4215–4228, 2012.

Ioannis Andrianakis, Ian R Vernon, Nicky McCreesh, Trevelyan J McKinley, Jeremy E , Rebecca N Nsubuga, Michael Goldstein, and Richard G White. Bayesian history matching of complex infectious disease models using emulation: a tutorial and a case study on hiv in uganda. *PLoS computational biology*, 11 (1):e1003968, 2015.

Ioannis Andrianakis, Nicky McCreesh, Ian Vernon, Trevelyan J McKinley, Jeremy E Oakley, Rebecca N Nsubuga, Michael Goldstein, and Richard G White. Efficient history matching of a high dimensional individual-based hiv transmission model. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):694–719, 2017.

Bruce Ankenman, Barry L Nelson, and Jeremy Staum. Stochastic kriging for simulation metamodeling. *Operations research*, 58(2):371–382, 2010.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Michael J Asher, Barry FW Croke, Anthony J Jakeman, and Luk JM Peeters. A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51(8):5957–5973, 2015.

Nedjem-Eddine Ayat, Mohamed Cheriet, and Ching Y Suen. Automatic model selection for the optimization of svm kernels. *Pattern Recognition*, 38(10):1733–1745, 2005.

Kenneth D Bailey. *Typologies and taxonomies: An introduction to classification techniques*. Number 102. Sage, 1994.

Leonardo S Bastos and Anthony O'Hagan. Diagnostics for gaussian process emulators. *Technometrics*, 51(4):425–438, 2009.

MJ Bayarri, JO Berger, John Cafeo, G Garcia-Donato, F Liu, J Palomo, RJ Parthasarathy, R Paulo, Jerry Sacks, D Walsh, et al. Computer model validation with functional output. *The Annals of Statistics*, 35(5):1874–1906, 2007.

Omar Bellprat, Sven Kotlarski, Daniel Lüthi, and Christoph Schär. Objective calibration of regional climate models. *Journal of Geophysical Research: Atmospheres*, 117(D23), 2012.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Sandrine Bony and Jean-Louis Dufresne. Marine boundary layer clouds at the heart of tropical cloud feedback uncertainties in climate models. *Geophysical Research Letters*, 32(20), 2005.

Richard G Bower, I Vernon, Michael Goldstein, AJ Benson, Cedric G Lacey, Carlton M Baugh, Shaun Cole, and CS Frenk. The parameter space of galaxy formation. *Monthly Notices of the Royal Astronomical Society*, 407(4):2017–2045, 2010.

AR Brown, RT Cederwall, A Chlond, PG Duynkerke, J-C Golaz, M Khairoutdinov, DC Lewellen, AP Lock, MK MacVean, C-H Moeng, et al. Large-eddy simulation of the diurnal cycle of shallow cumulus convection over land. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 128(582):1075–1093, 2002.

Jennỳ Brynjarsdóttir and Anthony O'Hagan. Learning about physical parameters: The importance of model discrepancy. *Inverse problems*, 30(11):114007, 2014.

Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

Gustavo Camps-Valls, Luis Gómez-Chova, Javier Calpe-Maravilla, José David Martín-Guerrero, Emilio Soria-Olivas, Luis Alonso-Chordá, and José Moreno. Robust support vector method for hyperspectral data classification and knowledge discovery. *IEEE Transactions on Geoscience and Remote sensing*, 42(7):1530–1542, 2004.

George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

Qun Chang, Qingcai Chen, and Xiaolong Wang. Scaling gaussian rbf kernel width to improve svm classification. In *2005 International Conference on Neural Networks and Brain*, volume 1, pages 19–22. IEEE, 2005.

Won Chang, Murali Haran, Roman Olson, Klaus Keller, et al. Fast dimension-reduced climate model calibration and the effect of data aggregation. *The Annals of Applied Statistics*, 8(2):649–673, 2014.

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

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Harold Trevor Clifford, William Stephenson, H Clifford, and W Stephenson. *An introduction to numerical classification*, volume 240. Academic Press New York, 1975.

Stefano Conti and Anthony O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3):640–651, 2010.

Stefano Conti, John Paul Gosling, Jeremy E Oakley, and Anthony O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

F Couvreux, F Guichard, J-L Redelsperger, C Kiemle, V Masson, J-P Lafore, and Cyrille Flamant. Water-vapour variability within a convective boundary-layer assessed by large-eddy simulations and ihop_2002 observations. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(611):2665–2693, 2005.

Fleur Couvreux, Frédéric Hourdin, Daniel Williamson, Romain Roehrig, Victoria Volodina, Najda Villefranque, Catherine Rio, Olivier Audouin, James Salter, Eric Bazile1, Florent Brient, Florence Favot, Rachel Honnert, Marie-Pierre Lefebvre, Jean-Baptiste Madeleine, Quentin Rodier, and Wenzhe Xu. Process-based climate model development harnessing machine learning: I. a calibration tool for parameterization improvement. *Earth and Space Science Open Archive*, 2020. URL https://doi.org/10.1002/essoar.10503597.1.

Peter S Craig, Michael Goldstein, Allan H Seheult, and James A Smith. Pressure matching for hydrocarbon reservoirs: a case study in the use of bayes linear strategies for large computer experiments. In *Case studies in Bayesian statistics*, pages 37–93. Springer, 1997.

PS Craig, Michael Goldstein, AH Seheult, and JA Smith. Bayes linear strategies for matching hydrocarbon reservoir history. *Bayesian statistics*, 5:69–95, 1996.

Jonathan A Cumming and Michael Goldstein. Bayes linear uncertainty analysis for oil reservoirs based on multiscale computer experiments. *O'Hagan, West, AM (eds.) The Oxford Handbook of Applied Bayesian Analysis*, pages 241–270, 2010.

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

Rameswar Debnath and Haruhisa Takahashi. Kernel selection for the support vector machine. *IEICE transactions on information and systems*, 87(12):2903–2904, 2004.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Neil R Edwards, David Cameron, and Jonathan Rougier. Precalibrating an intermediate complexity climate model. *Climate dynamics*, 37(7-8):1469–1482, 2011.

Paul N Edwards. Representing the global atmosphere: Computer models, data, and knowledge about climate change. *Changing the atmosphere: Expert knowledge and environmental governance*, 31:33, 2001.

Jean Gallier. Notes on convex sets, polytopes, polyhedra, combinatorial topology, voronoi diagrams and delaunay triangulations. *arXiv preprint arXiv:0805.0292*, 2008.

Peter Vincent Gehler. *Kernel learning approaches for image classification*. PhD thesis, Citeseer, 2009.

Marc G Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, 2(Dec):299–312, 2001.

Andrew Gettelman and Richard B Rood. *Demystifying climate models: a users guide to earth system models*. Springer Nature, 2016.

Rupert M Gladstone, Victoria Lee, Jonathan Rougier, Antony J Payne, Hartmut Hellmer, Anne Le Brocq, Andrew Shepherd, Tamsin L Edwards, Jonathan Gregory, and Stephen L Cornford. Calibrated prediction of pine island glacier retreat during the 21st and 22nd centuries with a coupled flowline model. *Earth and Planetary Science Letters*, 333:191–199, 2012.

Michael Goldstein and Jonathan Rougier. Probabilistic formulations for transferring inferences from mathematical models to physical systems. *SIAM journal on scientific computing*, 26(2):467–487, 2004.

Michael Goldstein and Jonathan Rougier. Reified bayesian modelling and inference for physical systems. *Journal of Statistical Planning and Inference*, 139(3):1221–1239, 2009.

John C Gower. Adding a point to vector diagrams in multivariate analysis. *Biometrika*, 55(3):582–585, 1968.

Robert B Gramacy and Herbert KH Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3):713–722, 2012.

Robert B Gramacy, Derek Bingham, James Paul Holloway, Michael J Grosskopf, Carolyn C Kuranz, Erica Rutter, Matt Trantham, R Paul Drake, et al. Calibrating a large computer experiment simulating radiative shock hydrodynamics. *The Annals of Applied Statistics*, 9(3):1141–1168, 2015.

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

Mengyang Gu, Jesus Palomo, and James O Berger. Robustgasp: Robust gaussian stochastic process emulation in r. *arXiv preprint arXiv:1801.01874*, 2018.

Françoise Guichard and Fleur Couvreux. A short review of numerical cloud-resolving models. *Tellus A: Dynamic Meteorology and Oceanography*, 69(1):1373578, 2017.

Gang Han, Thomas J Santner, and Jeremy J Rawlinson. Simultaneous determination of tuning and calibration parameters for computer experiments. *Technometrics*, 51(4):464–474, 2009.

RG Haylock and A O'Hagan. On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. *Bayesian statistics*, 5:629–637, 1996.

Michiel Hazewinkel. Chebyshev inequality in probability theory. In *Encyclopedia of mathematics*. Springer New York, NY, 2001.

Dave Higdon, Marc Kennedy, James C Cavendish, John A Cafeo, and Robert D Ryne. Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466, 2004.

Dave Higdon, James Gattiker, Brian Williams, and Maria Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002.

Richard G Hills and Timothy G Trucano. Statistical validation of engineering and scientific models: Background. *Sandia National Laboratories, Albuquerque, NM, Report No. SAND99-1256*, 1999.

Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3): 863–874, 2007.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

AAM Holtslag, Gunilla Svensson, P Baas, S Basu, B Beare, ACM Beljaars, FC Bosveld, J Cuxart, Jenny Lindvall, GJ Steeneveld, et al. Stable atmospheric boundary layers and diurnal cycles: challenges for weather and climate models. *Bulletin of the American Meteorological Society*, 94(11):1691–1706, 2013.

David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

Frédéric Hourdin, Fleur Couvreux, and Laurent Menut. Parameterization of the dry convective boundary layer based on a mass flux representation of thermals. *Journal of the atmospheric sciences*, 59(6):1105–1123, 2002.

Frédéric Hourdin, Ionela Musat, Sandrine Bony, Pascale Braconnot, Francis Codron, Jean-Louis Dufresne, Laurent Fairhead, Marie-Angèle Filiberti, Pierre Friedlingstein, Jean-Yves Grandpeix, et al. The lmdz4 general circulation model: climate performance and sensitivity to parametrized physics with emphasis on tropical convection. *Climate Dynamics*, 27(7-8):787–813, 2006.

Frédéric Hourdin, Thorsten Mauritsen, Andrew Gettelman, Jean-Christophe Golaz, Venkatramani Balaji, Qingyun Duan, Doris Folini, Duoying Ji, Daniel Klocke, Yun Qian, et al. The art and science of climate model tuning. *Bulletin of the American Meteorological Society*, 98(3):589–602, 2017.

Frédéric Hourdin, Daniel Williamson, Catherine Rio, Fleur Couvreux, Najda Villefranque Romain Roehrig, Ionela Musat, F.Binta Diallo Laurent Fairhead, and Victoria Volodina. Process-based climate model development harnessing 2 machine learning: Ii. model calibration from single 3 column to global. 2020. URL https://www.lmd.jussieu.fr/~hourdin/TMP/ITUNE/ItuneII.pdf.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

Thorsten Joachims. *Learning to classify text using support vector machines*, volume 668. Springer Science & Business Media, 2002.

Ian Jolliffe. *Principal component analysis*. Springer, 2011.

Cari G Kaufman, Derek Bingham, Salman Habib, Katrin Heitmann, Joshua A Frieman, et al. Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics*, 5(4):2470–2492, 2011.

Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

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

Seung-Jean Kim, Alessandro Magnani, and Stephen Boyd. Optimal kernel selection in kernel fisher discriminant analysis. In *Proceedings of the 23rd international conference on Machine learning*, pages 465–472, 2006.

Steven W Kirkpatrick. Development and validation of high fidelity vehicle crash simulation models. Technical report, SAE Technical Paper, 2000.

David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

JT-Y Kwok and IW-H Tsang. The pre-image problem in kernel methods. *IEEE transactions on neural networks*, 15(6):1517–1525, 2004.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

LA Lee, KS Carslaw, KJ Pringle, and GW Mann. Mapping the uncertainty in global ccn using emulation. *Atmospheric Chemistry and Physics*, 12(20):9739–9751, 2012.

LA Lee, KJ Pringle, CL Reddington, GW Mann, P Stier, DV Spracklen, JR Pierce, and KS Carslaw. The magnitude and causes of uncertainty in global model simulations of cloud condensation nuclei. *Atmospheric Chemistry & Physics Discussions*, 13(3), 2013.

Simon Lehuger, Benoit Gabrielle, Marcel Van Oijen, David Makowski, J-C Germon, Thierry Morvan, and Catherine Hénault. Bayesian calibration of the nitrous oxide emission module of an agro-ecosystem model. *Agriculture, Ecosystems & Environment*, 133(3-4):208–222, 2009.

Fei Liu, Mike West, et al. A dynamic modelling strategy for bayesian computer model emulation. *Bayesian Analysis*, 4(2):393–411, 2009.

Xiang Ma and Nicholas Zabaras. Kernel principal component analysis for stochastic input model generation. *Journal of Computational Physics*, 230(19):7311–7331, 2011.

Thorsten Mauritsen, Bjorn Stevens, Erich Roeckner, Traute Crueger, Monika Esch, Marco Giorgetta, Helmuth Haak, Johann Jungclaus, Daniel Klocke, Daniela Matei, et al. Tuning the climate of a global model. *Journal of advances in modeling Earth systems*, 4(3), 2012.

DJ McNeall, Peter G Challenor, JR Gattiker, and EJ Stone. The potential of an observational data set for calibration of a computationally expensive computer model. 2013.

Erik HW Meijering, Karel J Zuiderveld, and Max A Viergever. Image reconstruction by convolution with symmetrical piecewise nth-order polynomial kernels. *IEEE transactions on image processing*, 8(2):192–201, 1999.

Scott Menard. *Applied logistic regression analysis*, volume 106. Sage, 2002.

James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209 (441-458):415–446, 1909.

Sebastian Mika, Bernhard Schölkopf, Alex J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems*, pages 536–542, 1999.

Hossein Mohammadi, Peter Challenor, and Marc Goodfellow. Emulating dynamic non-linear simulators using gaussian processes. *Computational Statistics & Data Analysis*, 139:178–196, 2019.

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

Owen J Murphy. Nearest neighbor pattern classification perceptrons. *Proceedings of the IEEE*, 78(10):1595–1598, 1990.

Christine Nam, Sandrine Bony, J-L Dufresne, and H Chepfer. The 'too few, too bright'tropical low-cloud problem in cmip5 models. *Geophysical Research Letters*, 39(21), 2012.

Radford M Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*, 1997.

Roel AJ Neggers, Martin Köhler, and Anton CM Beljaars. A dual mass flux framework for boundary layer convection. part i: Transport. *Journal of the Atmospheric Sciences*, 66(6):1465–1487, 2009.

Douglas Nychka, Christopher Wikle, and J Andrew Royle. Multiresolution models for nonstationary spatial covariance functions. *Statistical Modelling*, 2(4):315–331, 2002.

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

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

Antony M Overstall and David C Woods. Multivariate emulation of computer simulators: model selection and diagnostics with application to a humanitarian relief model. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 65 (4):483–505, 2016.

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

TN Palmer, FJ Doblas-Reyes, R Hagedorn, and A Weisheimer. Probabilistic prediction of climate using multi-model ensembles: from basics to applications. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1463):1991–1998, 2005.

Abani K Patra, Andrew C Bauer, CC Nichita, E Bruce Pitman, Michael F Sheridan, M Bursik, Byron Rupp, A Webber, AJ Stinton, LM Namikawa, et al. Parallel adaptive numerical simulation of dry avalanches over natural terrain. *Journal of Volcanology and Geothermal Research*, 139(1-2):1–21, 2005.

Matthew Plumlee. Bayesian calibration of inexact computer models. *Journal of the American Statistical Association*, 112(519):1274–1285, 2017.

Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

Catherine Rio and Frédéric Hourdin. A thermal plume model for the convective boundary layer: Representation of cumulus clouds. *Journal of the atmospheric sciences*, 65(2):407–425, 2008.

Luiz Felippe S Rodrigues, Ian Vernon, and Richard G Bower. Constraints on galaxy formation models from the galaxy stellar mass function and its evolution. *Monthly Notices of the Royal Astronomical Society*, 466(2):2418–2435, 2017.

Jonathan Rougier. Probabilistic inference for future climate using an ensemble of climate model evaluations. *Climatic Change*, 81(3-4):247–264, 2007.

Jonathan Rougier. Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843, 2008.

Jonathan Rougier, David MH Sexton, James M Murphy, and David Stainforth. Analyzing the climate sensitivity of the hadsm3 climate model using ensembles from different but related experiments. *Journal of Climate*, 22(13):3540–3557, 2009.

Olivier Roustant, David Ginsbourger, and Yves Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. 2012.

Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

Andrea Saltelli, Karen Chan, M Scott, et al. Sensitivity analysis. probability and statistics series. *John and Wiley & Sons, New York*, 2000.

Andrea Saltelli, Marco Ratto, Stefano Tarantola, and Francesca Campolongo. Sensitivity analysis for chemical models. *Chemical reviews*, 105(7):2811–2828, 2005.

James M Salter and Daniel Williamson. A comparison of statistical emulation methodologies for multi-wave calibration of environmental models. *Environmetrics*, 27(8):507–523, 2016.

James M Salter and Daniel B Williamson. Efficient calibration for high-dimensional computer model output using basis methods. *arXiv preprint arXiv:1906.05758*, 2019.

James M Salter, Daniel B Williamson, Lauren J Gregoire, and Tamsin L Edwards. Quantifying spatio-temporal boundary condition uncertainty for the north american deglaciation. *arXiv preprint arXiv:1808.09322*, 2018.

James M Salter, Daniel B Williamson, John Scinocca, and Viatcheslav Kharin. Uncertainty quantification for spatio-temporal computer models with calibration-optimal bases. *arXiv preprint arXiv:1801.08184*, 2019.

Irina Sandu and Bjorn Stevens. On the factors modulating the stratocumulus to cumulus transitions. *Journal of the Atmospheric Sciences*, 68(9):1865–1881, 2011.

Thomas J Santner, Brian J Williams, William Notz, and Brain J Williams. *The design and analysis of computer experiments*, volume 1. Springer, 2003.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

Bernhard Schölkopf, Sebastian Mika, Alex Smola, Gunnar Rätsch, and Klaus-Robert Müller. Kernel pca pattern reconstruction via approximate pre-images. In *International Conference on Artificial Neural Networks*, pages 147–152. Springer, 1998a.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5): 1299–1319, 1998b.

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

David MH Sexton, James M Murphy, Mat Collins, and Mark J Webb. Multivariate probabilistic projections using imperfect climate models part i: outline of methodology. *Climate dynamics*, 38(11-12):2513–2542, 2012.

John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Huaitao Shi, Jianchang Liu, and Yingwei Zhang. An optimized kernel principal component analysis algorithm for fault detection. *IFAC Proceedings Volumes*, 42 (8):846–851, 2009.

Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*, volume 12. Siam, 2013.

Guido F Smits and Elizabeth M Jordaan. Improved svm regression using mixtures of kernels. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pages 2785–2790. IEEE, 2002.

Rosanna Soentpiet et al. *Advances in kernel methods: support vector learning*. MIT press, 1999.

César R Souza. Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, 3:29, 2010.

Ron Sun et al. *Cognition and multi-agent interaction: From cognitive modeling to social simulation*. Cambridge University Press, 2006.

Karl E Taylor, Ronald J Stouffer, and Gerald A Meehl. An overview of cmip5 and the experiment design. *Bulletin of the American Meteorological Society*, 93(4): 485–498, 2012.

Claudia Tebaldi and Reto Knutti. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 365(1857):2053–2075, 2007.

Rui Tuo and CF Wu. A theoretical framework for calibration in computer models: parametrization, estimation and convergence properties. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):767–795, 2016.

Rui Tuo, CF Jeff Wu, et al. Efficient calibration for imperfect computer models. *The Annals of Statistics*, 43(6):2331–2352, 2015.

Tony Van Gestel, JAK Suykens, Bart De Moor, and Joos Vandewalle. Automatic relevance determination for least squares support vector machine regression. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 4, pages 2416–2421. IEEE, 2001.

Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

Ian Vernon, Michael Goldstein, Richard G Bower, et al. Galaxy formation: a bayesian uncertainty analysis. *Bayesian analysis*, 5(4):619–669, 2010.

Tomas F Yago Vicente, Minh Hoai, and Dimitris Samaras. Leave-one-out kernel optimization for shadow detection and removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):682–695, 2017.

Aurore Voldoire, E Sanchez-Gomez, D Salas y Mélia, B Decharme, Christophe Cassou, S Sénési, Sophie Valcke, I Beau, A Alias, M Chevallier, et al. The cnrm-cm5. 1 global climate model: description and basic evaluation. *Climate Dynamics*, 40(9-10):2091–2121, 2013.

Victoria Volodina and Daniel Williamson. Diagnostics-driven nonstationary emulators using kernel mixtures. *SIAM/ASA Journal on Uncertainty Quantification*, 8 (1):1–26, 2020.

Tinghua Wang, Houkuan Huang, Shengfeng Tian, and Jianfeng Xu. Feature selection for svm via optimization of kernel polarization with gaussian ard kernels. *Expert Systems with Applications*, 37(9):6663–6668, 2010.

Richard D Wilkinson. Bayesian calibration of expensive multivariate computer experiments, 2010.

Christopher KI Williams. On a connection between kernel pca and metric multidimensional scaling. In *Advances in neural information processing systems*, pages 675–681, 2001.

Daniel Williamson. Exploratory ensemble designs for environmental models using k-extended latin hypercubes. *Environmetrics*, 26(4):268–283, 2015.

Daniel Williamson and Adam T Blaker. Evolving bayesian emulators for structured chaotic time series, with application to large climate models. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):1–28, 2014.

Daniel Williamson, Michael Goldstein, and Adam Blaker. Fast linked analyses for scenario-based hierarchies. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(5):665–691, 2012.

Daniel Williamson, Michael Goldstein, Lesley Allison, Adam Blaker, Peter Challenor, Laura Jackson, and Kuniko Yamazaki. History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. *Climate dynamics*, 41(7-8):1703–1729, 2013.

Daniel Williamson, Adam T Blaker, Charlotte Hampton, and James Salter. Identifying and removing structural biases in climate models with history matching. *Climate dynamics*, 45(5-6):1299–1324, 2015.

Daniel B Williamson, Adam T Blaker, and Bablu Sinha. Tuning without overtuning: parametric uncertainty quantification for the nemo ocean model. *Geoscientific Model Development*, 10(4):1789, 2017.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Max A Woodbury and M Woodbury. Inverting modified matrices. 1950.

WW Xing, V Triantafyllidis, AA Shah, PB Nair, and Nicholas Zabaras. Manifold learning for the emulation of spatial fields from computational models. *Journal of Computational Physics*, 326:666–690, 2016.

Ying Xiong, Wei Chen, Daniel Apley, and Xuru Ding. A non-stationary covariance-based kriging method for metamodelling in engineering design. *International Journal for Numerical Methods in Engineering*, 71(6):733–756, 2007.

Wenzhe Xu, Daniel Williamson, and Peter Challenor. Local voronoi tessellations for robust multi-wave calibration of computer models. *International Journal for Uncertainty Quantification*, 2021.

Jiankai Xue and Bo Shen. A novel swarm intelligence optimization approach: sparrow search algorithm. *Systems Science & Control Engineering*, 8(1):22–34, 2020.

Ru Zhang, Chunfang Devon Lin, and Pritam Ranjan. A sequential design approach for calibrating a dynamic population growth model. *arXiv preprint arXiv:1811.00153*, 2018.

Mark H Zweig and Gregory Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39 (4):561–577, 1993.

Appendix A

Mathematical proofs for Chapter 4

A.1 Proof of Equation (4.15)

By replacing **W** with on W_k $(D \times 1)$ equation (4.13) (only consider one eigenvector), and multiplying $\tilde{\phi}(f(\mathbf{x}_j))^T$ $(1 \times D)$ on both side of equation (4.13), we have that

$$\tilde{\phi}(f(\mathbf{x}_j))GW_k = \lambda\tilde{\phi}(f(\mathbf{x}_j))W_k, \tag{A.1}$$

for all $j = 1 \ldots n$.

The left side can be computed as:

$$\begin{aligned}
\mathrm{L} &= \tilde{\phi}(f(\mathbf{x}_j))^T GW_k \\
&= \tilde{\phi}(f(\mathbf{x}_j))^T \frac{1}{n}\sum_{i=1}^{n}\tilde{\phi}(f(\mathbf{x}_i))\tilde{\phi}(f(\mathbf{x}_i))^T \sum_{l=1}^{n}\alpha_{kl}\tilde{\phi}(f(\mathbf{x}_l)) \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{n}\alpha_{kl}\tilde{\phi}(f(\mathbf{x}_j))^T\tilde{\phi}(f(\mathbf{x}_i))\tilde{\phi}(f(\mathbf{x}_i))^T\tilde{\phi}(f(\mathbf{x}_l)) \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{n}\alpha_{kl}\tilde{k}(f(\mathbf{x}_j), f(\mathbf{x}_i))\tilde{k}(f(\mathbf{x}_i), f(\mathbf{x}_l)) \\
&= \frac{1}{n}(\tilde{\mathbf{K}}^2\alpha_k)_j,
\end{aligned}$$

and the right side is

$$\begin{aligned}
\mathrm{R} &= \tilde{\phi}(f(\mathbf{x}_j))^T \lambda_k W_k \\
&= \lambda_k \sum_{i=1}^{n} \alpha_{ki} \tilde{\phi}(f(\mathbf{x}_j))^T \tilde{\phi}(f(\mathbf{x}_i)) \\
&= \lambda_k \sum_{i=1}^{n} \alpha_{ki} \tilde{k}(f(\mathbf{x}_j), f(\mathbf{x}_i)) \\
&= \lambda_k (\tilde{\mathbf{K}} \alpha_k)_j .
\end{aligned}$$

Therefore, we have that

$$\frac{1}{n}(\tilde{\mathbf{K}}^2 \alpha_k)_j = \lambda_k (\tilde{\mathbf{K}} \alpha_k)_j ,$$

which holds for any $j = 1, \ldots, n$. The eigenvalue problem is then equivalent to

$$\begin{aligned}
&\frac{1}{n}\tilde{\mathbf{K}}^2 \alpha_k = \lambda_k \tilde{\mathbf{K}} \alpha_k , \\
&\Leftrightarrow \tilde{\mathbf{K}} \alpha_k = n \lambda_k \alpha_k , \\
&\Leftrightarrow \tilde{\mathbf{K}} \alpha_k = \tilde{\lambda}_k \alpha_k .
\end{aligned} \tag{A.2}$$

Equation (A.2) shows that α_k is an eigenvector of $\tilde{\mathbf{K}}$, such that the eigenvectors W of covariance matrix G can be represented by the eigenvectors of $\tilde{\mathbf{K}}$. Also, given the eigenvalues $\tilde{\lambda}$ of $\tilde{\mathbf{K}}$, the eigenvalues λ of covariance matrix G are $\tilde{\lambda} = n\lambda$.

A.2 Proof of Equation (4.64)

$||\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]||^2$ can be written as:

$$\begin{aligned}
||\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]||^2 &= (\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)])^T (\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]) \\
&= (\mathbf{W}^{rT} \tilde{\phi}(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)])^T (\mathbf{W}^{rT} \tilde{\phi}(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]) \\
&= (\mathbf{W}^{rT} \tilde{\phi}(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)])^T \mathbf{W}^{rT} \mathbf{W}^r (\mathbf{W}^{rT} \tilde{\phi}(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]) \\
&= (\mathbf{W}^r \mathbf{W}^{rT} \tilde{\phi}(\mathbf{x}^*) - \mathbf{W}^r \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)])^T (\mathbf{W}^r \mathbf{W}^{rT} \tilde{\phi}(\mathbf{x}^*) - \mathbf{W}^r \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]) \\
&= (\tilde{\phi}_r(\mathbf{x}^*) - \mathrm{E}\left[\tilde{\phi}_r(f(\mathbf{x}^*))\right])^T (\tilde{\phi}_r(\mathbf{x}^*) - \mathrm{E}\left[\tilde{\phi}_r(f(\mathbf{x}^*))\right]) \\
&= (\phi_r(\mathbf{x}^*) - \mathrm{E}[\phi_r(f(\mathbf{x}^*))])^T (\phi_r(\mathbf{x}^*) - \mathrm{E}[\phi_r(f(\mathbf{x}^*))]).
\end{aligned} \tag{A.3}$$

To proof the equation (4.64), we compute the difference between $||\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]||^2$ and $||\ \phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))]\ ||^2$:

$$\begin{aligned}
&||\phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))]\ ||^2 - ||\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]\ ||^2 \\
&\quad = (\phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi_r(f(\mathbf{x}))])^T(\phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi_r(f(\mathbf{x}))]) - \\
&\quad (\phi_r(f(\mathbf{x}^*)) - \mathrm{E}[\phi_r(f(\mathbf{x}))])^T(\phi_r(f(\mathbf{x}^*)) - \mathrm{E}[\phi_r(f(\mathbf{x}))]) \\
&\quad = \tilde{k}(f(\mathbf{x}^*), f(\mathbf{x}^*)) + \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]^T\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)] - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]^T\mathbf{A}\tilde{\mathbf{K}}_{f(\mathbf{x}^*)} - \\
&\quad \Big((\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*)))^T(\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*))) + \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]^T\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)] - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]^T(\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*)))\Big) \\
&\quad = \tilde{k}(f(\mathbf{x}^*), f(\mathbf{x}^*)) - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]^T\mathbf{A}\tilde{\mathbf{K}}_{f(\mathbf{x}^*)} - \Big((\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*)))^T(\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*))) - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]^T\mathbf{A}\tilde{\mathbf{K}}\Big) \\
&\quad = \tilde{k}(f(\mathbf{x}^*), f(\mathbf{x}^*)) - (\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*)))^T(\mathbf{W}^{rT}\tilde{\phi}(f(\mathbf{x}^*))) \\
&\quad = ||\varepsilon_f||^2,
\end{aligned} \tag{A.4}$$

where ε_f is the reconstruction error of model output and $||.||^2$ is the euclidean distance function. We show that $\phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))]\ ||^2$ can be calculated from $||\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]\ ||^2$ and $||\varepsilon_f||^2$,

$$||\phi(f(\mathbf{x}^*)) - \mathrm{E}[\phi(f(\mathbf{x}^*))]\ ||^2 = ||\mathbf{C}_r(\mathbf{x}^*) - \mathrm{E}[\mathbf{C}_r(\mathbf{x}^*)]\ ||^2 + ||\varepsilon_f||^2.$$

A.3 The expectation and variance of $d^2_{\phi(z),\phi(f(x^*))}$

To make the mathematics clear, let $\Delta = z - f(x^*)$, Δ is then a vector with length m. The expectation of $d^2_{\phi(z),\phi(f(x^*))}$ is

$$
\begin{aligned}
\mathrm{E}\left[d^2_{\phi(z),\phi(f(x^*))}\right] &= 2 - 2\mathrm{E}\left[\exp(-\sigma \parallel \Delta \parallel^2)\right] \\
&= 2 - 2\int_{-\infty}^{\infty} \exp(-\sigma\Delta^T\Delta)p(\Delta)d\Delta \\
&= 2 - 2\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_e+\Sigma_\eta|^{\frac{1}{2}}}\int_{-\infty}^{\infty} \exp(-\sigma\Delta^T\Delta)exp(-\frac{1}{2}\Delta^T(\Sigma_e+\Sigma_\eta)^{-1}\Delta)d\Delta \\
&= 2 - 2\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_e+\Sigma_\eta|^{\frac{1}{2}}}\int_{-\infty}^{\infty} \exp(-\frac{1}{2}\Delta^T(2\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})\Delta)d\Delta \\
&= 2 - 2\frac{(2\pi)^{\frac{n}{2}}|(2\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{(2\pi)^{\frac{n}{2}}|\Sigma_e+\Sigma_\eta|^{\frac{1}{2}}}\times \\
&\int_{-\infty}^{\infty}\frac{1}{(2\pi)^{\frac{n}{2}}|(2\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}\exp(-\frac{1}{2}\Delta^T(2\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})\Delta)d\Delta \\
&= 2 - 2\frac{|(2\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{|\Sigma_e+\Sigma_\eta|^{\frac{1}{2}}},
\end{aligned}
$$

where $\mathbf{I}_{m\times m}$ is the $m\times m$ identity matrix. The variance of $d^2_{\phi(z),\phi(f(x^*))}$ is

$$
\begin{aligned}
\mathrm{Var}\left[d^2_{\phi(z),\phi(f(x^*))}\right] &= \mathrm{E}\left[(d^2_{\phi(z),\phi(f(x^*))})^2\right] - \mathrm{E}\left[d^2_{\phi(z),\phi(f(x^*))}\right]^2 \\
&= \mathrm{E}\left[(2-2\exp(-\sigma \parallel \Delta \parallel^2))^2\right] - \mathrm{E}\left[2-2\exp(-\sigma \parallel \Delta \parallel^2)\right]^2 \\
&= \mathrm{E}\left[4+4(\exp(-\sigma \parallel \Delta \parallel^2))^2 - 8\exp(-\sigma \parallel \Delta \parallel^2)\right] - \left(2-\mathrm{E}\left[2\exp(-\sigma \parallel \Delta \parallel^2)\right]\right)^2 \\
&= 4+\mathrm{E}\left[4(\exp(-\sigma \parallel \Delta \parallel^2))^2\right] - 8\mathrm{E}\left[\exp(-\sigma \parallel \Delta \parallel^2)\right] - \\
&\left(4+\mathrm{E}\left[2\exp(-\sigma \parallel \Delta \parallel^2)\right]^2 - 8\mathrm{E}\left[2\exp(-\sigma \parallel \Delta \parallel^2)\right]\right) \\
&= 4\mathrm{E}\left[(\exp(-\sigma \parallel \Delta \parallel^2))^2\right] - \mathrm{E}\left[2\exp(-\sigma \parallel \Delta \parallel^2)\right]^2 \\
&= 4\frac{|(4\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{|\Sigma_e+\Sigma_\eta|^{\frac{1}{2}}} - 4\left(\frac{|(2\sigma\mathbf{I}_{m\times m}+(\Sigma_e+\Sigma_\eta)^{-1})^{-1}|^{\frac{1}{2}}}{|\Sigma_e+\Sigma_\eta|^{\frac{1}{2}}}\right)^2.
\end{aligned}
$$

A.4 Proof of Equation (4.84)

To prove equation (4.84), we first compute $||\mathbf{C}_r(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]||^2$,

$$\begin{aligned}
||\mathbf{C}_r(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]||^2 &= (\mathbf{C}_r(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{C}_r(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]) \\
&= (\mathbf{W}^{rT}\tilde{\phi}(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{W}^{rT}\tilde{\phi}(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]) \\
&= (\mathbf{W}^{rT}\tilde{\phi}(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T\mathbf{W}^{rT}\mathbf{W}^r(\mathbf{W}^{rT}\tilde{\phi}(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]) \\
&= (\mathbf{W}^r\mathbf{W}^{rT}\tilde{\phi}(z) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})])^T(\mathbf{W}^r\mathbf{W}^{rT}\tilde{\phi}(z) - \mathbf{W}^r\mathrm{E}[\mathbf{C}_r(\mathbf{x})]) \\
&= (\tilde{\phi}_r(z) - \mathrm{E}\left[\tilde{\phi}_r(f(\mathbf{x}))\right])^T(\tilde{\phi}_r(z) - \mathrm{E}\left[\tilde{\phi}_r(f(\mathbf{x}))\right]) \\
&= (\phi_r(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))])^T(\phi_r(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))]).
\end{aligned} \tag{A.5}$$

We compute the difference between $||\phi(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))]||^2$ and $||\mathbf{C}_r(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]||^2$:

$$\begin{aligned}
&||\phi(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))]||^2 - ||\mathbf{C}_r(z) - \mathrm{E}[\mathbf{C}_r(\mathbf{x})]||^2 \\
&= (\phi(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))])^T(\phi(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))]) - (\phi_r(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))])^T(\phi_r(z) - \mathrm{E}[\phi_r(f(\mathbf{x}))]) \\
&= \tilde{k}(z,z) + \mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathrm{E}[\mathbf{C}_r(\mathbf{x})] - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathbf{A}\tilde{\mathbf{K}}_z - \\
&\left((\mathbf{W}^{rT}\tilde{\phi}(z))^T(\mathbf{W}^{rT}\tilde{\phi}(z)) + \mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathrm{E}[\mathbf{C}_r(\mathbf{x})] - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T(\mathbf{W}^{rT}\tilde{\phi}(z))\right) \\
&= \tilde{k}(z,z) - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathbf{A}\tilde{\mathbf{K}}_z - \left((\mathbf{W}^{rT}\tilde{\phi}(z))^T(\mathbf{W}^{rT}\tilde{\phi}(z)) - 2\mathrm{E}[\mathbf{C}_r(\mathbf{x})]^T\mathbf{A}\tilde{\mathbf{K}}\right) \\
&= \tilde{k}(z,z) - (\mathbf{W}^{rT}\tilde{\phi}(z))^T(\mathbf{W}^{rT}\tilde{\phi}(z)) \\
&= ||\varepsilon_z||^2,
\end{aligned} \tag{A.6}$$

where ε_z is the observation reconstruction error and $||.||^2$ is the euclidean distance function. We show that $\mathcal{I}_{F1}(\mathbf{x})$ can be calculated from $\mathcal{I}_{C1}(\mathbf{x})$, where

$$\mathcal{I}_{F1}(\mathbf{x}) = \mathcal{I}_{C1}(\mathbf{x}) + ||\varepsilon_z||^2.$$

A.5 Proof of Equation (5.11)

Given $\mathbf{W}$ as $P^T\mathbf{W}_{PCA}^T$, $\mathrm{E}[\phi(f(\mathbf{x}))] = P^T\mathrm{E}[f(\mathbf{x})]$, and $\mathrm{Var}[\phi(f(\mathbf{x}))] = P\mathrm{Var}[f(\mathbf{x})]P^T$, we can write the implausibility $\mathcal{I}_{F0}(\mathbf{x})$ as

$$\begin{aligned}\mathcal{I}_{F0}(\mathbf{x}) &= (\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))])^T(\mathbf{1}_D + \mathrm{Var}[\phi(f(\mathbf{x}))])^{-1}(\phi(z) - \mathrm{E}[\phi(f(\mathbf{x}))]) \\ &= (P^Tz - P^T\mathrm{E}[f(\mathbf{x})])^T(\mathbf{1}_D + P\mathrm{Var}[f(\mathbf{x})]P^T)^{-1}(P^Tz - P^T\mathrm{E}[f(\mathbf{x})]) \qquad \text{(A.7)} \\ &= (z - \mathrm{E}[f(\mathbf{x})])^TP(\mathbf{1}_D + P\mathrm{Var}[f(\mathbf{x})]P^T)^{-1}P^T(z - \mathrm{E}[f(\mathbf{x})]).\end{aligned}$$

To expand $\mathcal{I}_{F0}(\mathbf{x})$, we apply the Woodbury formula to $(\mathbf{1}_D + P\mathrm{Var}[f(\mathbf{x})]P^T)^{-1}$, so that:

$$\begin{aligned}(\mathbf{1}_D + P\mathrm{Var}[f(\mathbf{x})]P^T)^{-1} &= \mathbf{1}_D - P^T(\mathrm{Var}[f(\mathbf{x})])^{-1} + PP^T)^{-1}P \\ &= \mathbf{1}_D - P^T\left((PP^T)^{-1} - (PP^T)^{-1}\left(\mathrm{Var}[f(\mathbf{x})] + (PP^T)^{-1}\right)^{-1}(PP^T)^{-1}\right)P.\end{aligned} \qquad \text{(A.8)}$$

Therefore, $\mathcal{I}_{F0}(\mathbf{x})$ can be written as:

$$\begin{aligned}&\mathcal{I}_{F0}(\mathbf{x}) \\ &= (z - \mathrm{E}[f(\mathbf{x})])^TP\Bigg(\mathbf{1}_D - P^T\Bigg((PP^T)^{-1} - (PP^T)^{-1}\Big(\mathrm{Var}[f(\mathbf{x})] + (PP^T)^{-1}\Big)^{-1}(PP^T)^{-1}\Bigg)P\Bigg)P^T(z - \mathrm{E}[f(\mathbf{x})] \\ &= (z - \mathrm{E}[f(\mathbf{x})])^T\Bigg(PP^T - PP^T\Bigg((PP^T)^{-1} - (PP^T)^{-1}\Big(\mathrm{Var}[f(\mathbf{x})] + (PP^T)^{-1}\Big)^{-1}(PP^T)^{-1}\Bigg)PP^T\Bigg)(z - \mathrm{E}[f(\mathbf{x} \\ &= (z - \mathrm{E}[f(\mathbf{x})])^T\left(\Upsilon^{-1} - \Upsilon^{-1}\left(\Upsilon - \Upsilon(\mathrm{Var}[f(\mathbf{x})] + \Upsilon)^{-1}\Upsilon\right)\Upsilon^{-1}\right)(z - \mathrm{E}[f(\mathbf{x})]) \\ &= (z - \mathrm{E}[f(\mathbf{x})])^T\left(\Upsilon^{-1} - \Upsilon^{-1}\Upsilon\Upsilon^{-1} - \Upsilon^{-1}\Upsilon(\mathrm{Var}[f(\mathbf{x})] + \Upsilon)^{-1}\Upsilon\Upsilon^{-1}\right)(z - \mathrm{E}[f(\mathbf{x})]) \\ &= (z - \mathrm{E}[f(\mathbf{x})])^T\left(\Upsilon^{-1} - \Upsilon^{-1} - (\mathrm{Var}[f(\mathbf{x})] + \Upsilon)^{-1}\right)(z - \mathrm{E}[f(\mathbf{x})]) \\ &= (z - \mathrm{E}[f(\mathbf{x})])^T(\mathrm{Var}[f(\mathbf{x})] + \Upsilon)^{-1}(z - \mathrm{E}[f(\mathbf{x})]).\end{aligned} \qquad \text{(A.9)}$$

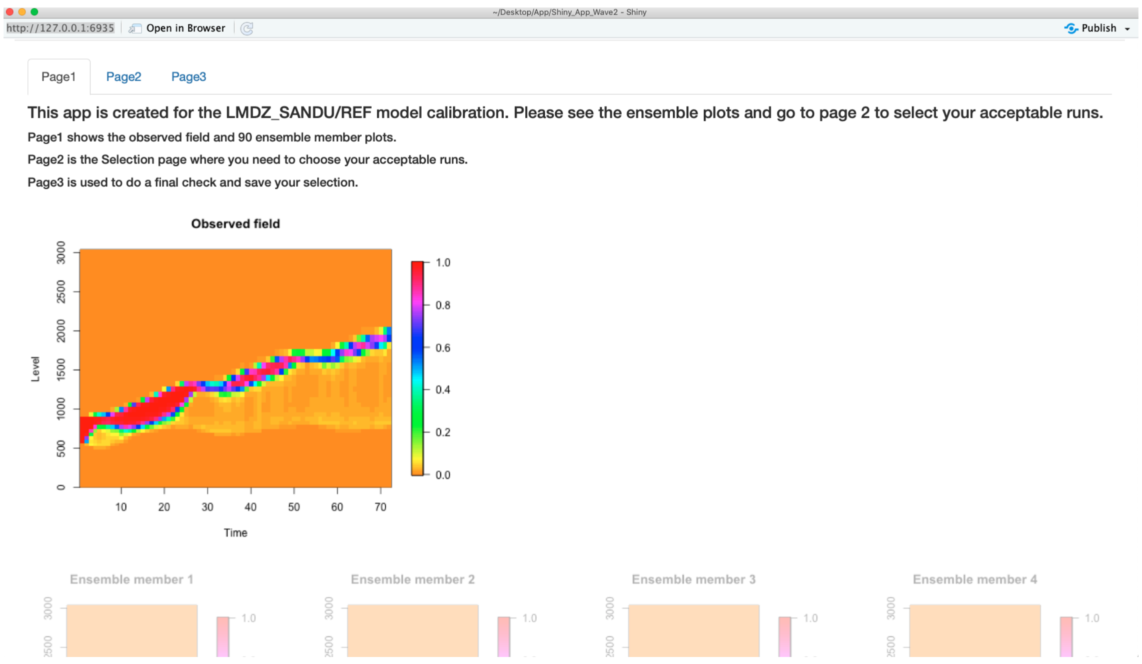
Hence, we prove that

$$\mathcal{I}(\mathbf{x}) = \mathcal{I}_{F0}(\mathbf{x}).$$

Appendix B

R `Shiny`

R `Shiny` app is created for the LMDZ model calibration, aiming to consider the modeller's information in the calibration process. In this appendix, we present the contents of the app. For interaction purpose, experts will use the app to look at the model output and accept or reject. Hence, the `Shiny` app we created includes three pages, page 1 shows the observed field and 90 ensemble member plots, page 2 is the selection page where the experts need to choose their acceptable runs, and page 3 is used to do a final check and save the experts selection.



This app is created for the LMDZ_SANDU/REF model calibration. Please see the ensemble plots and go to page 2 to select your acceptable runs.

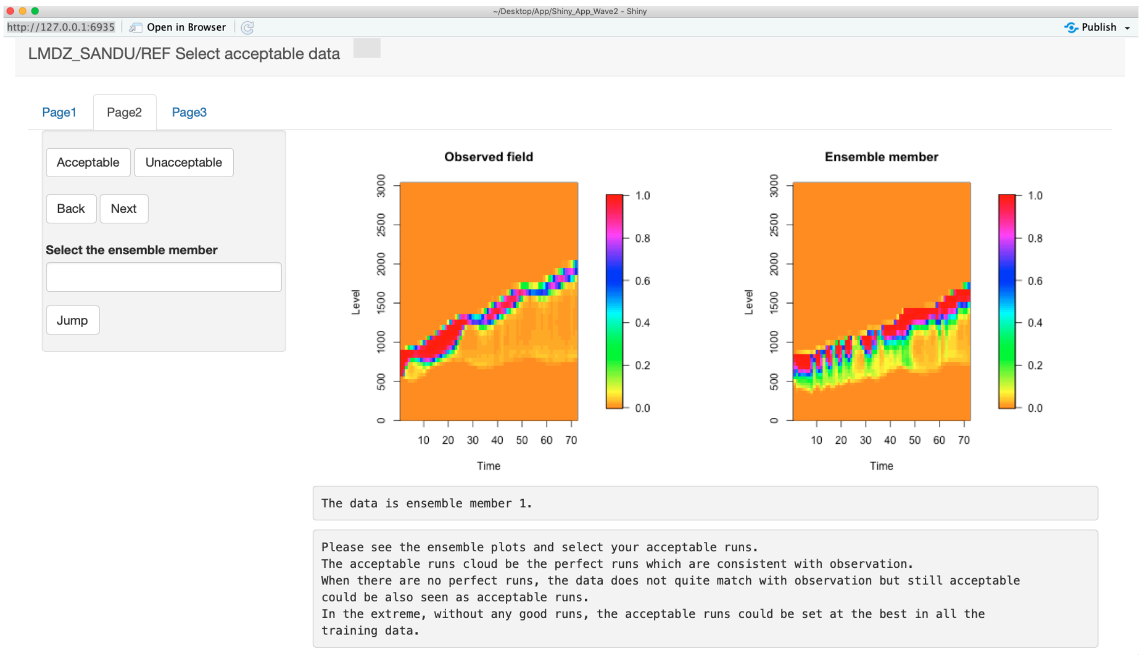
Page1 shows the observed field and 90 ensemble member plots.

Page2 is the Selection page where you need to choose your acceptable runs.

Page3 is used to do a final check and save your selection.

Fig. B.1 Page 1: Overall of the ensemble.

The first page of our app is presented in figure B.1. On the top of this page, there are three buttons with page numbers that can be used to switch the pages. Under the brief overview, the observation is fist presented, and the 90 model runs are plotted (the full page 1 is too long, we only paste part of it here, the rest can be seen by using the app). By looking at all ensemble members on page 1, experts can get an idea of which runs look best before moving to the accept reject page.



LMDZ_SANDU/REF Select acceptable data

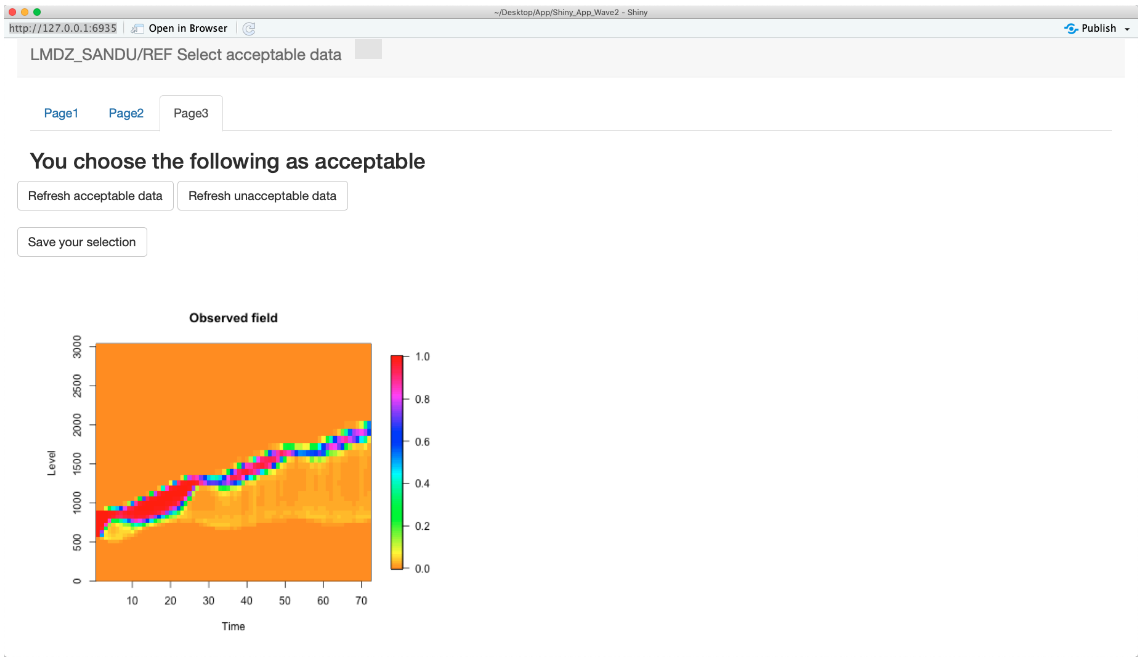
```
The data is ensemble member 1.
```

```
Please see the ensemble plots and select your acceptable runs.
The acceptable runs cloud be the perfect runs which are consistent with observation.
When there are no perfect runs, the data does not quite match with observation but still acceptable
could be also seen as acceptable runs.
In the extreme, without any good runs, the acceptable runs could be set at the best in all the
training data.
```

Fig. B.2 Page 2: Selection page.

The page 2 is presented in figure B.2. On the right panel, the left figure shows the observation, and the right figure shows the ensemble member (from the first one to ninetieth). Once the experts click the acceptable/unacceptable button on the left panel, then ensemble member will change to the next one. The “Jump”, “Back” and “Next” buttons could be used when the experts want to correct their decisions. The app will only save their final decision.

The page 3 is presented in figure B.3 to show all of the experts selections that made in page 2. The selection will be saved by clicking the ‘save your selection’ button. If the experts are not sure about their selection, they can go back to page 2 and type the unsure ensemble member, then they can easily compare the observed field with this ensemble member and correct the choice.



LMDZ_SANDU/REF Select acceptable data

You choose the following as acceptable

Fig. B.3 Page 3: Final check and save the data.

After all the steps introduced above, a csv file called ‘Acceptable.csv’ will be automatically generated. This csv file will save expert’s acceptable runs as 1, expert’s unacceptable runs as 2 and 0 means experts did not make any selection for this ensemble member.

Appendix C

Addition remarks for examples

This appendix gives additional information for toy models and applications used to illustrate methodology throughout Chapter 4, Chapter 5 and Chapter 6.

C.1 Chapter 4 toy model

The spatial toy function that was introduced in Chapter 4, giving output over a 10×10 field, with 5 input parameters each taking values in $[-1, 1]$, is defined as $f(\mathbf{x}) = \text{Signal}(\mathbf{x}) + e$, where e is the error that generates from a Normal distribution mean 0 (to ensure the output is positive) and variance 0.05, independently for each box in a 10×10 grid, and $\text{Signal}(\mathbf{x})$ is the cross marks pattern specified over the grid. Whether the signal, $\text{Signal}(\mathbf{x})$, exist depends on parameters, $\mathbf{x}$, $14 + 8x_1 + 3x_2 + x_3$ controls the location on the horizontal axis, and $3x_4 + 5x_5 + 8$ controls the location on the vertical axis. Only if these two coordinate values are less than 10, model outputs contains the key pattern. We sample 50 parameter settings, $\mathbf{X}$, using a Latin Hyper cube from the 5-dimensional parameter space $\mathcal{X}$, giving an ensemble, $\mathbf{F}$, with dimension 100×50. We plot the ensemble, $\mathbf{F}$ (100×50), from the 1st run to the 50th in Figure C.1.

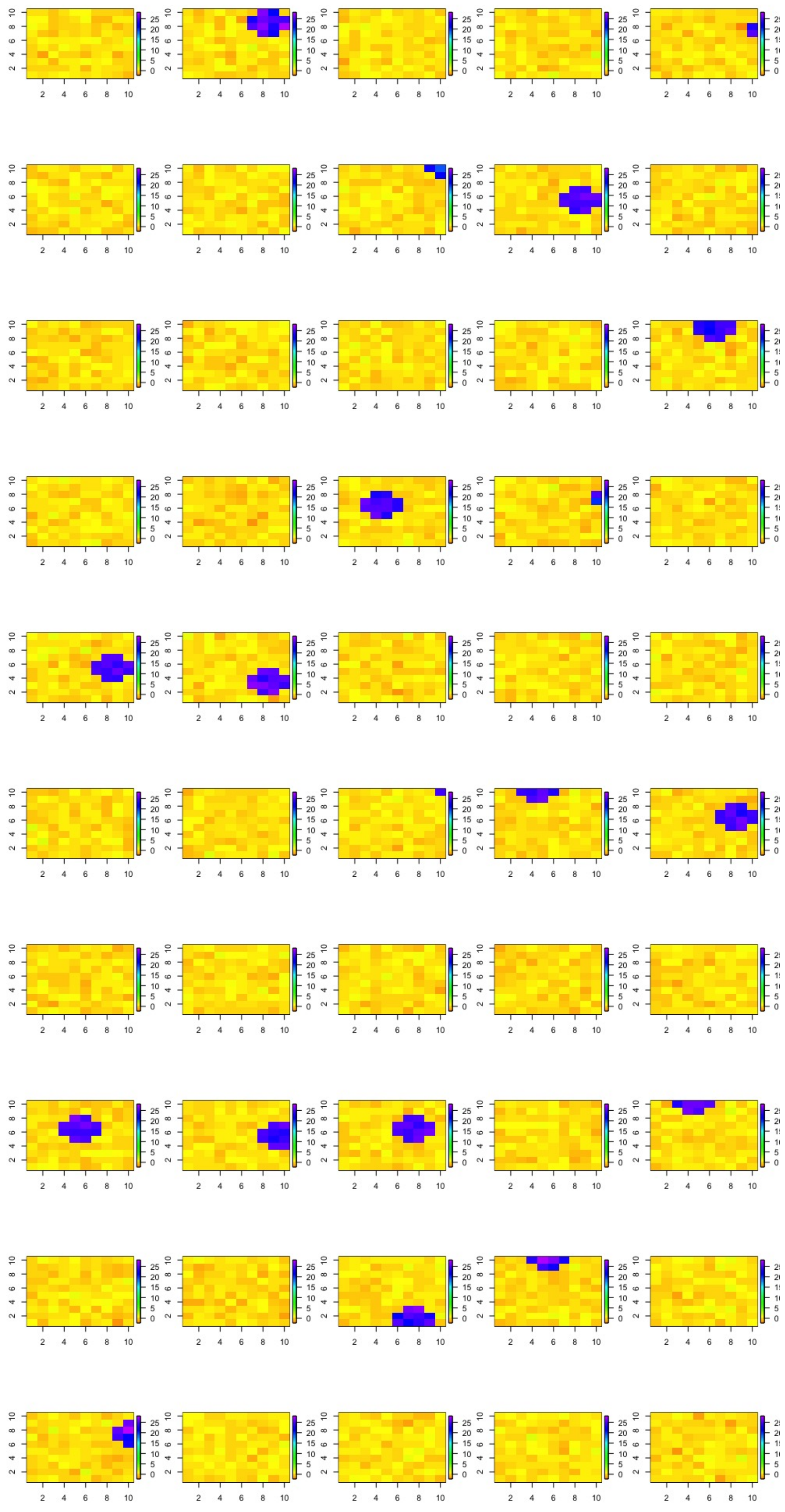


Fig. C.1 The ensemble plots for Chapter 6 toy model.

C.1.1 Emulator diagnostic

Before using an emulator with other approaches, diagnostics must be used to validate and assess the adequacy of a Gaussian process emulator for representing the simulator. We perform ‘leave one out’ validation on the training data to assess the fit of emulators. For the following leave one out diagnostic plots, each plot represents one left-out emulator predicted, black points and error bar are from the emulator posterior mean and two standard deviation prediction intervals. The true function values are in blue if they lie within two standard deviation prediction intervals, or red otherwise.

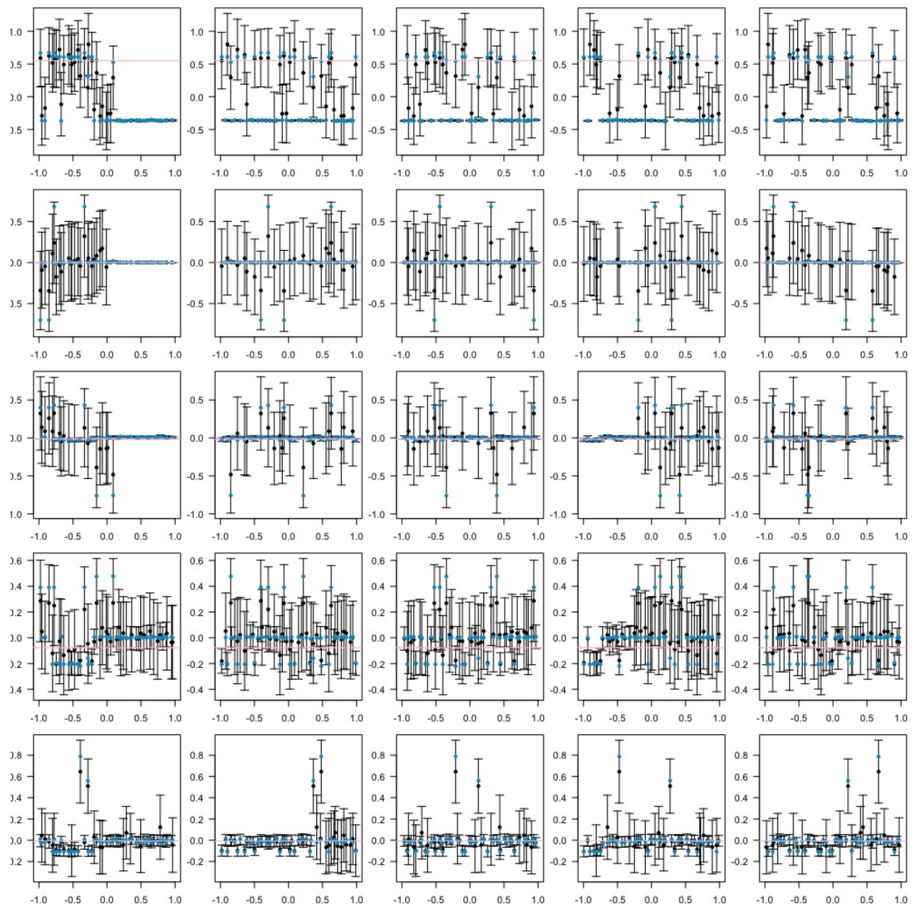


Fig. C.2 Leave-one-out cross-validation plots for the emulators for the coefficients on the first 5 basis vectors.

C.2 Refocusing of Chapter 5 numerical example 1

We continue the refocusing steps for the first numerical example introduced in Chapter 5. To perform a second wave, a new ensemble is required. For consistency with the first wave ensemble, we select 60 members for the new ensemble. The new ensemble design, $\mathbf{X}_2$, is randomly generated from $\mathcal{X}^1$. The new ensemble members are presented in Figure C.3. By comparison with the ensemble members from the first wave, we can see that more than half of the ensemble members do contain the key patterns, which indicates that the NROY space $\mathcal{X}^1$ is closer to true NROY space. Hence, more acceptable runs are selected in wave 2. As well as using the new ensemble, the acceptable runs in the fist wave ensemble are also used to build emulators.

We now follow the same methodology as in the first wave, where the best 16 members are selected as the acceptable runs, and our kernel optimising algorithm is applied to select a kernel function. As before, the linear kernel is selected as the best kernel function, but only first two basis vectors are required to reach 95% of the ensemble variability. We calculate the projections for the ensemble of model runs with the selected kernel, and fit Gaussian process emulators to the coefficients for each of the first two basis vectors. The validation plots for each of the GP emulator is given in Figure C.4. Given the acceptable runs and the emulators for the coefficients, we perform the second wave of KHM. The density plot and minimum implausibility plots of the second wave NROY space are shown in Figure C.6. We also perform a third wave, following the same procedures. Wave 3 ensemble members are presented in Figure C.7, the validation plots for each of the GP emulator is given in Figure C.5 and wave 3 NROY space are shown in Figure C.6.

The NROY space after wave 2, $\mathcal{X}^2$, contains 13.28% of the initial parameter space $\mathcal{X}$, and more than half of $\mathcal{X}^1$ is ruled out in the second wave. Performing wave 3 does not reduce $\mathcal{X}^2$ significantly, which contains 8.84% of the initial parameter space. In the first wave, only parameter $\mathbf{x}_6$ had clear visual signs of having

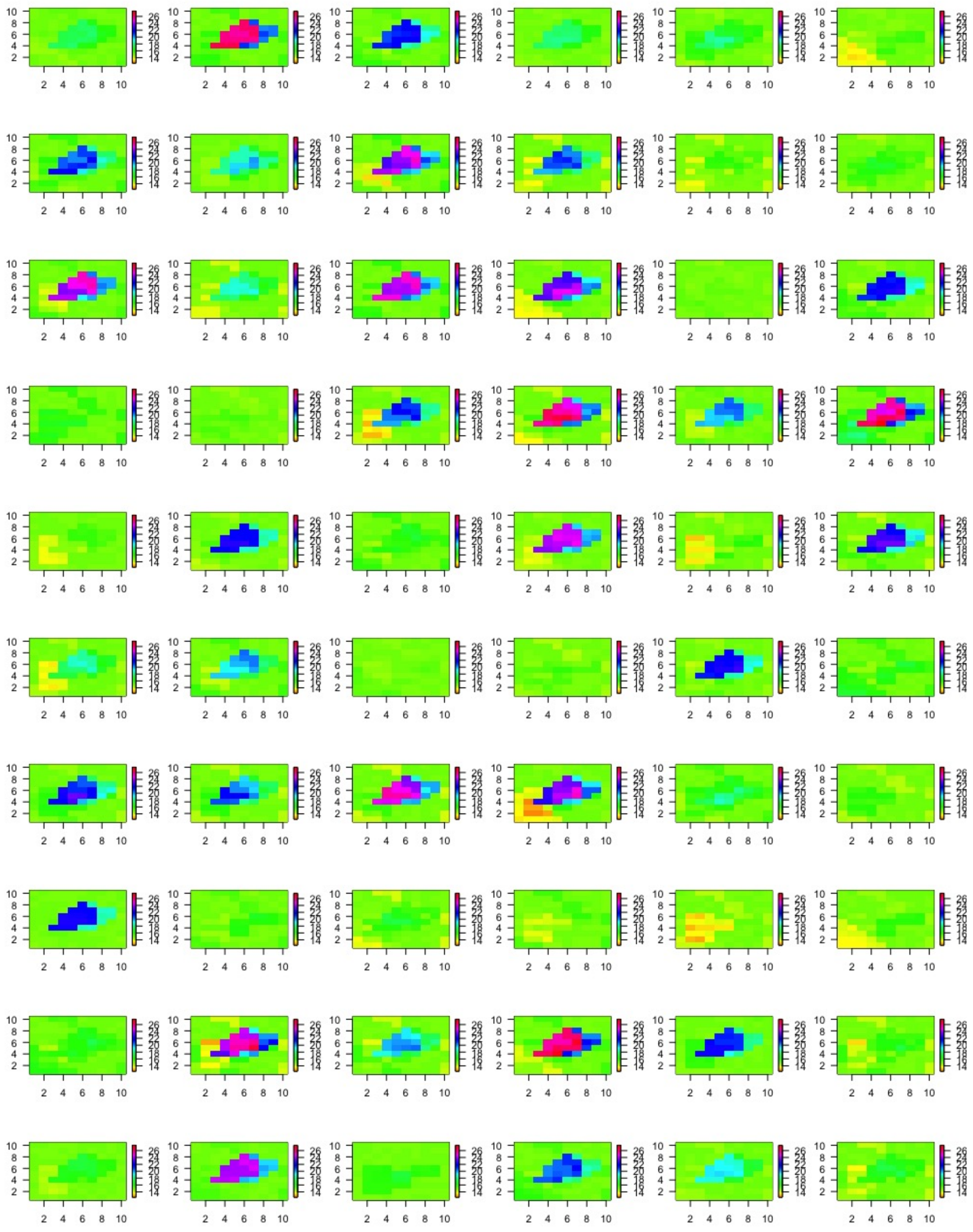


Fig. C.3 The 60 ensemble members for wave 2.

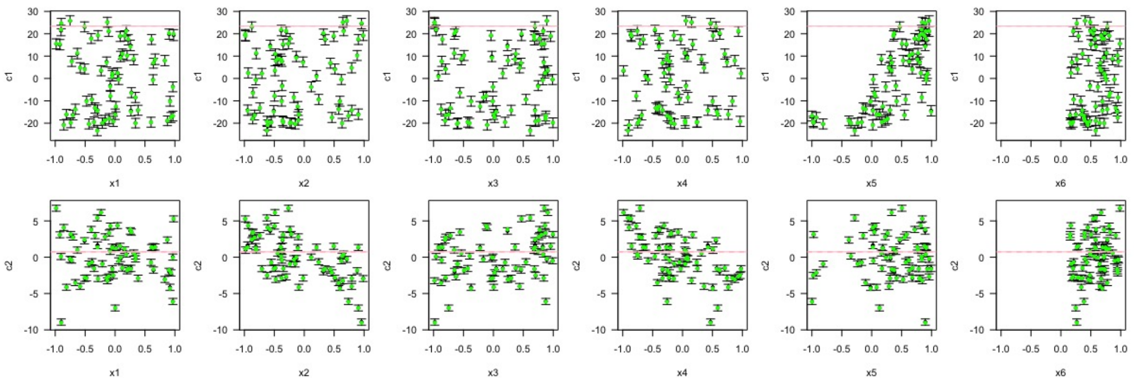


Fig. C.4 Wave 2 Leave-one-out cross-validation plots for the emulators for the coefficients on the first two basis vectors.

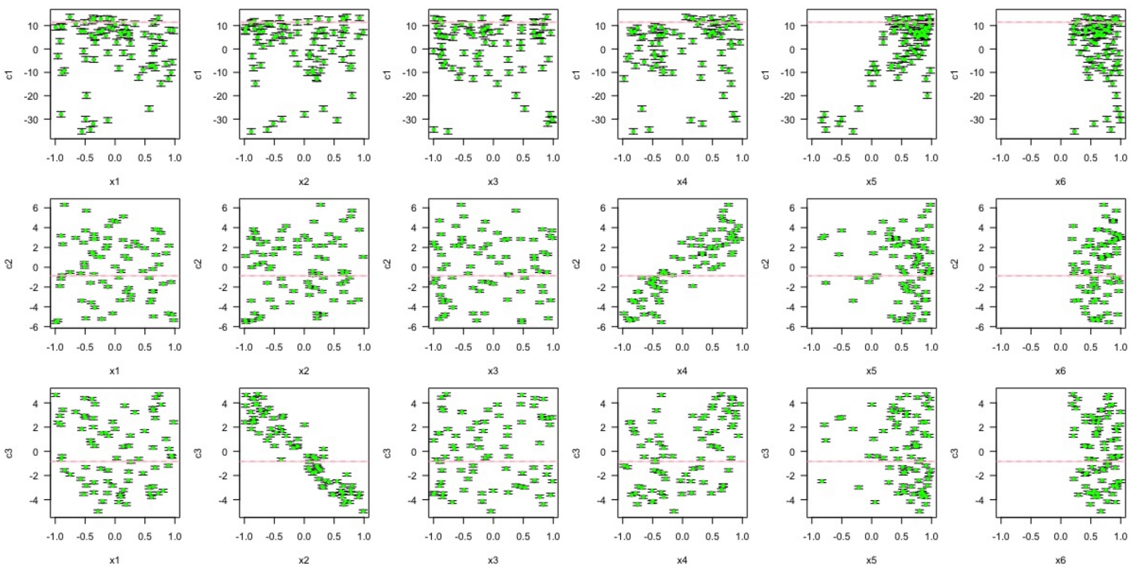


Fig. C.5 Wave 3 Leave-one-out cross-validation plots for the emulators for the coefficients on the first three basis vectors.

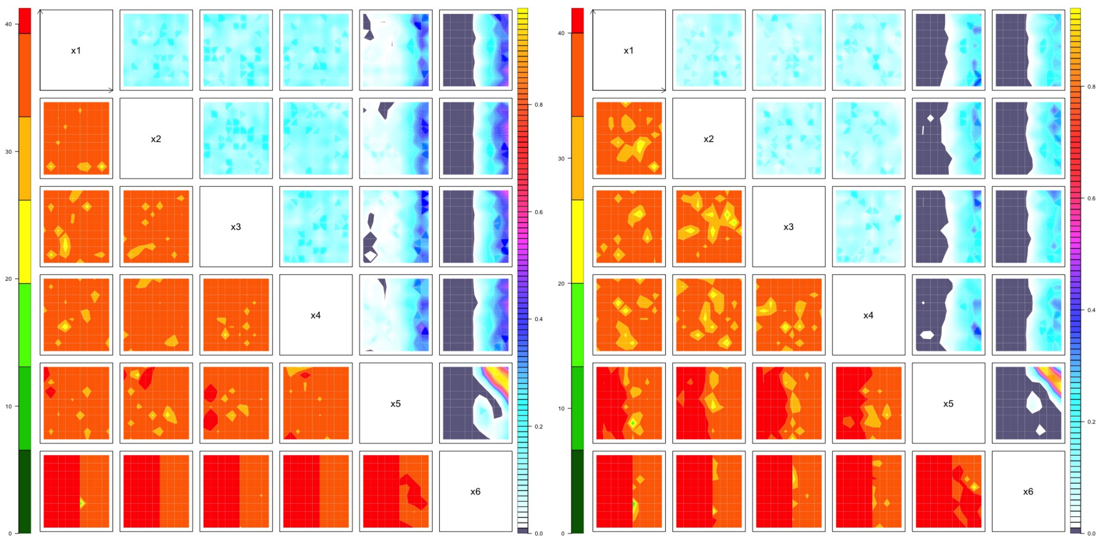


Fig. C.6 *Left*: Wave 3 NROY space. *Right*: Wave 3 NROY space.

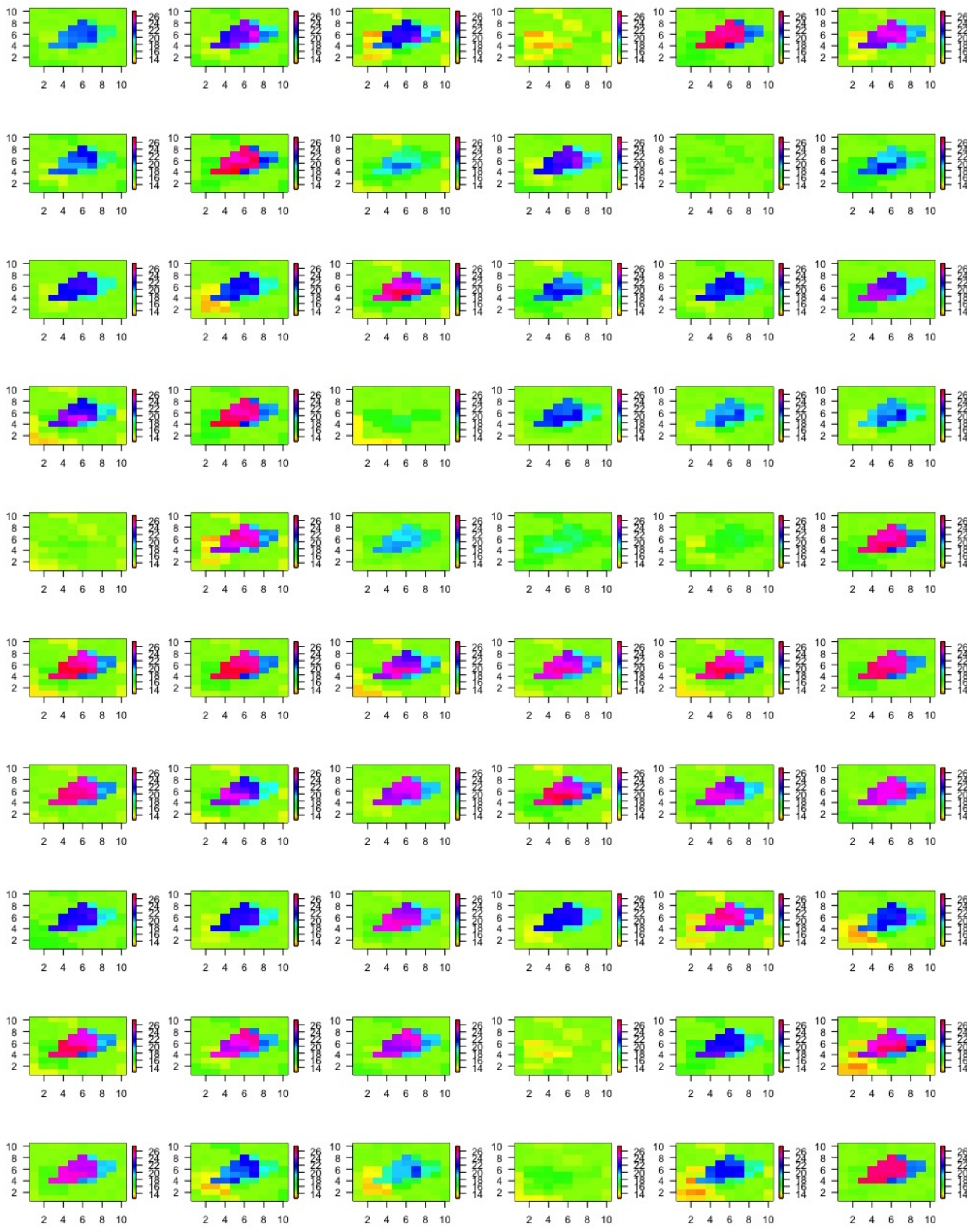


Fig. C.7 The 60 ensemble members for wave 3.

been constrained. In Wave 3, we find a similar reduction of the true NROY space for $\mathbf{x}_5$. The NROY space after wave 3 cuts us down to the positive corner of values of $\mathbf{x}_5$ and $\mathbf{x}_6$, which has the same structure as true NROY space. In total, 96.25% of the true NROY are retained in the final NROY space. The performance of KHM shows that our kernel optimization method works with this numerical example, and it suggests that using the linear kernel when the standard history matching is suitable for the application (from wave 1 to wave 3). But that does not means that the linear kernel is always the best choice for all the applications, in order to prove the feasibility of KHM and kernel optimization methods, we also gives another applied instance in Section 5.5.

C.3 Chapter 5 numerical example 2

For KHM the numerical example 2 in Section 5.5, we fitted two waves of Gaussian process emulators. Wave 2 ensemble members are presented in Figure C.8, and the validation plots for wave 2 GP emulators is given in Figure C.8.

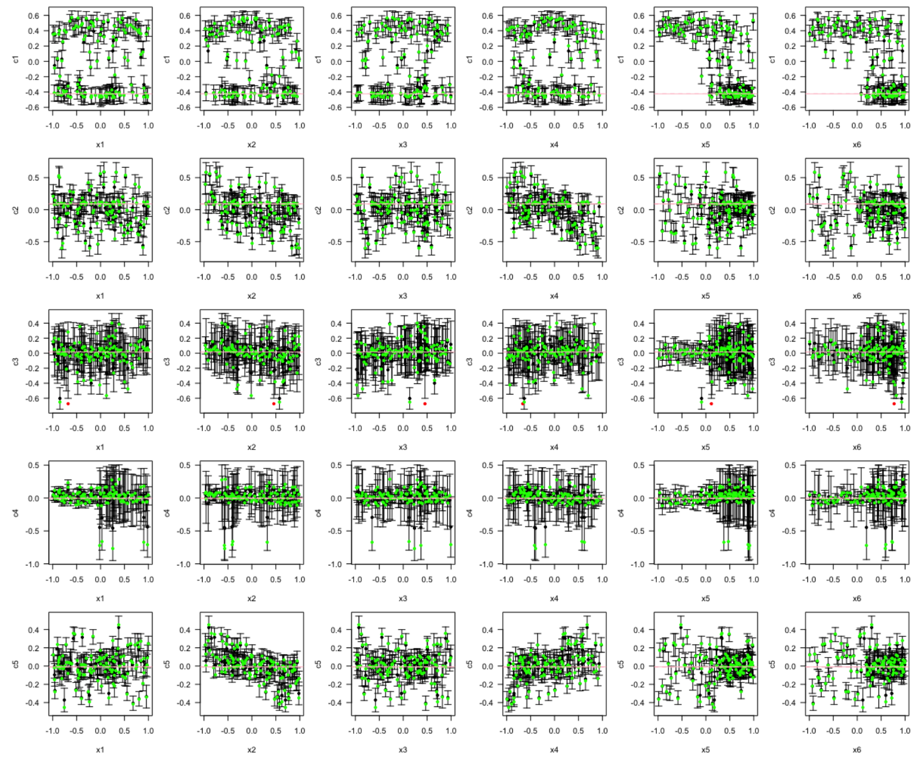


Fig. C.8 Wave 2 Leave-one-out cross-validation plots for the emulators for the coefficients on the first five basis vectors.

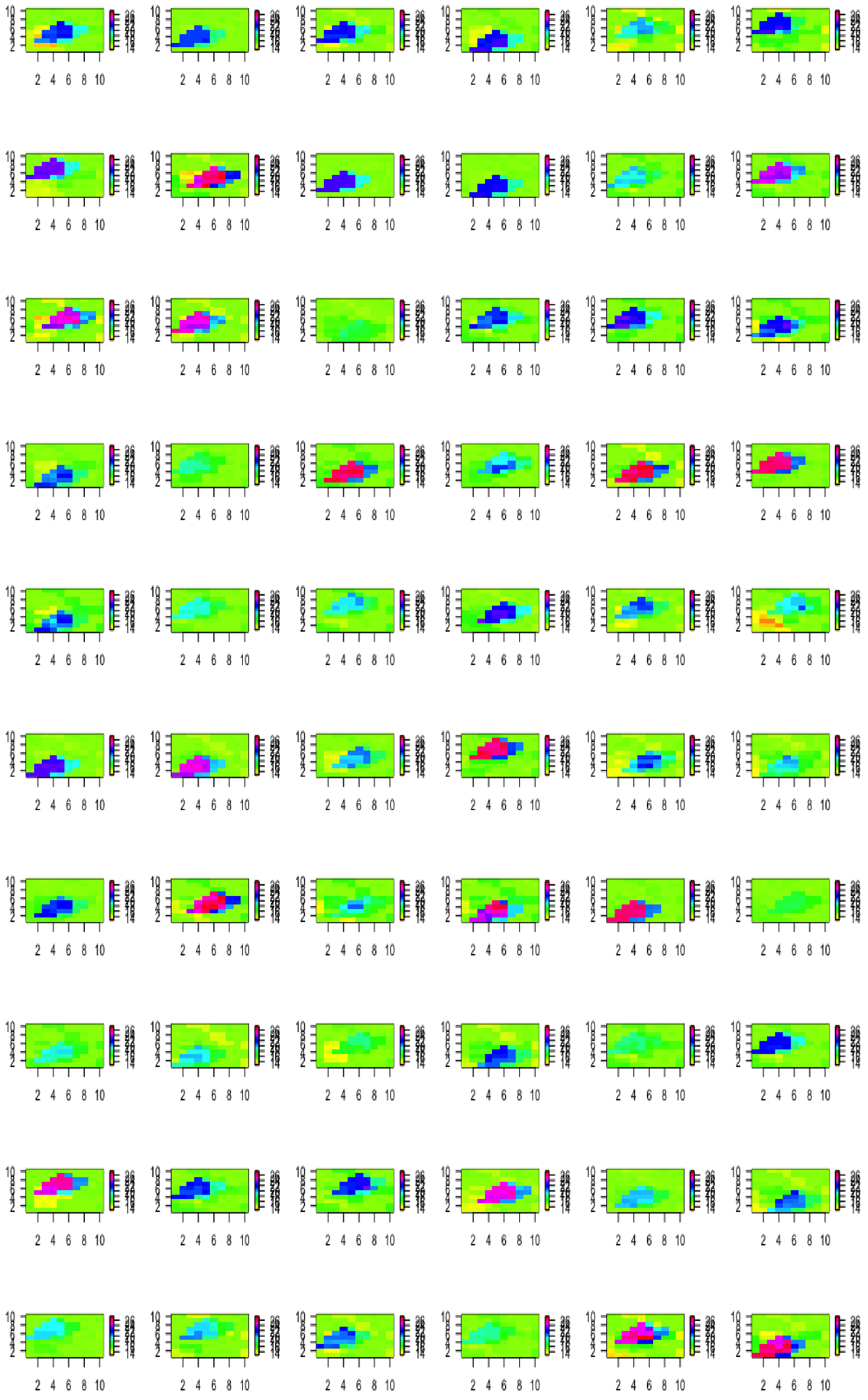


Fig. C.9 The 60 ensemble members for wave 2.

C.4 Chapter 5: loading vector plots

We illustrate KHM to Chapter 5 numerical studies. In the first example, our optimization algorithm suggests that the optimal situation, $\mathcal{P}(\mathcal{K}_{par}) = 1$, can be achieved when $T^{**}(\mathcal{K}_{par}) = \min(\mathcal{I}(\mathbf{x})^U)$, $\delta_1 = \delta_2 = 1$ and $\omega = 1$ (since the weight of the Gaussian kernel is zero, the choice of Gaussian kernel parameters is irrelevant). The optimal kernel function for the toy example is

$$k(f(\mathbf{x}), f(\mathbf{x}')) = f(\mathbf{x})^T(\Sigma_e + \Sigma_\eta)^{-1} f(\mathbf{x}').$$

KHM with the above kernel represents standard PCA-based history matching (Salter et al. (2019)). It is usual when doing PCA to plot the loading vectors. We plot the loading vectors for the first example with the selected kernel in Figure C.11. To make a comparison between kernel PCA and standard PCA, we also plot PCA loading vectors using the same data in Figure C.11. The structure of these two plots are the same, but the directions are different due to the different transfer of the data. In these plots, the black dots are unacceptable runs, and the green points are acceptable runs. Given the same components, standard history matching and KHM would give the same results.

To show the kernel PCA components with mixture kernel, we also plot the kernel PCA loading plots for numerical study 2. The wave 1 kernel PCA loading plot is shown in Figure C.12, and the wave 2 kernel PCA loading plot is shown in Figure C.12. With the difference ensemble, these two waves give different plots. However, we could see that the components of the acceptable runs are pretty similar in these two plots, except three outliers in wave 2.

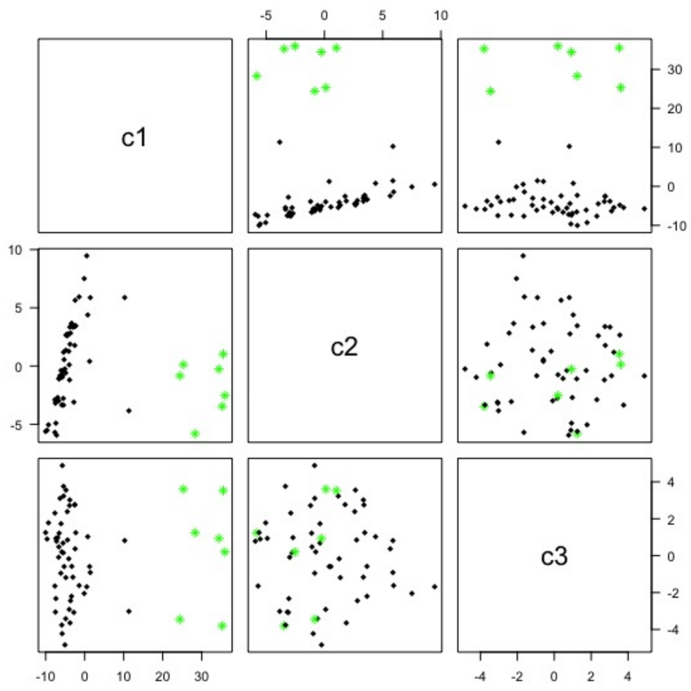


Fig. C.10 Plot of the first three kernel principal component (PC) loading vectors.

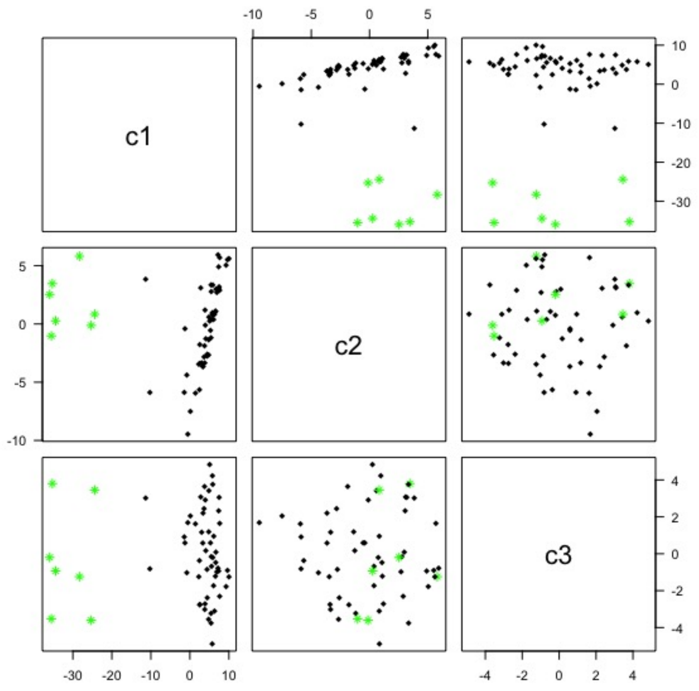


Fig. C.11 Plot of the first three principal component (PC) loading vectors.

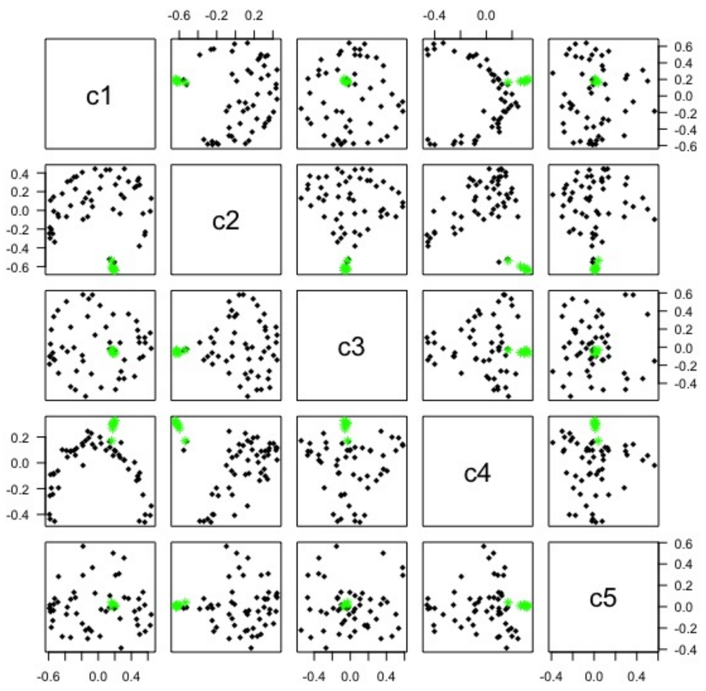


Fig. C.12 Plot of the first 5 principal component (PC) loading vectors in wave 1.

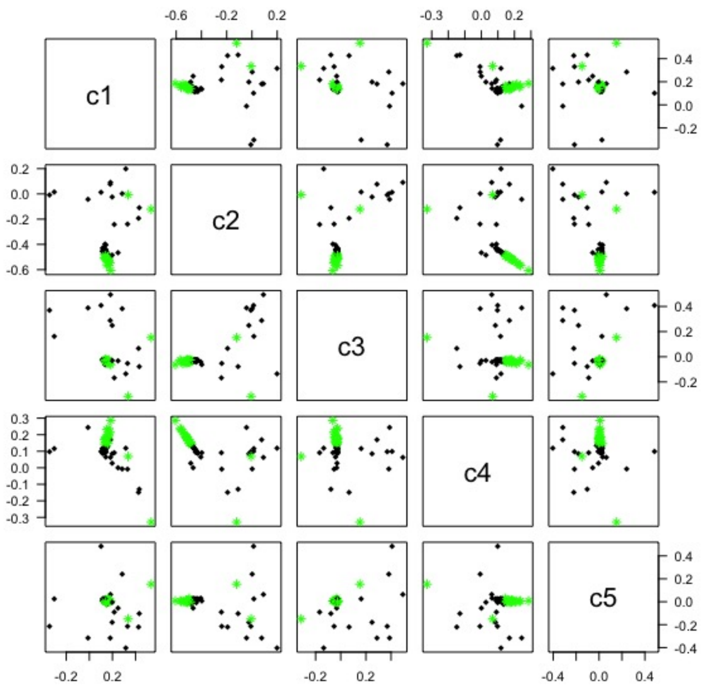


Fig. C.13 Plot of the first 5 principal component (PC) loading vectors in wave 2.

C.5 Boundary-layer cloud Model

C.5.1 Wave 1

We perform KHM for the climate model with a different set for Σ_η. The mixture kernel used here is same as Chapter 6

$$k(f(\mathbf{x}), f(\mathbf{x}')) = \omega f(\mathbf{x})^T \Upsilon^{-1} f(\mathbf{x}') + (1-\omega) g \exp(-(f(\mathbf{x}) - f(\mathbf{x}'))^T \Upsilon^{-1} (f(\mathbf{x}) - f(\mathbf{x}'))/\sigma), \tag{C.1}$$

where ω is a weight parameter, $\omega \in [0,1]$, σ is a Gaussian kernel parameter, g is scale parameter, and Υ is a $l \times l$ positive definite weight matrix defined as the sum of the observation error (LES reference error) variance ,Σ_e, and another variance term, Σ_η, $\Upsilon = \Sigma_e + \Sigma_\eta$, where $e \sim N(0, \Sigma_e)$ and Σ_e is given in Chapter 6. We set Σ_η as the Gaussian covariance function, with two unknown correction length parameters. We follow the same performance evaluation function in Section 6.4.1, the arbitrary influence factor is set as $\alpha = 0.8$. The optimization algorithm finds that $\omega = 0.08011$ is the best choice for the weight parameter, $\sigma = 0.4898$ for Gaussian kernel parameter, $g = 428.6988$, the two correction length parameters in Σ_η are suggested as 0.3943 and 0.2550. Given the same training data and same expert judgement, our algorithm suggest two different kernels because of the different initial setting of the kernel structure. This interesting results show that there is plenty of scope for potential kernel structure, more complex mixture kernel function can be developed in the future.

Given this kernel function, we calculate the ensemble projections by applying the kernel PCA algorithm. Given the ensemble of the 5-dimensional input parameter space, and the coefficient projection for each $\mathbf{x}$ for each output in feature space, $\mathbf{C}_i(\mathbf{x})$, we build five univariate Gaussian process emulators for the first five basis vectors. Leave-one-out cross-validation plots are shown in Figure C.15. We use KHM with $\mathcal{I}_{F1}(\mathbf{x})$ to rule out of regions of parameter space. The wave 1 NROY density plots and the minimum implausibility plots for each pair of parameters is shown in Figure C.16. We achieve an NROY space $\mathcal{X}^1$ of size 56.78% of $\mathcal{X}$.

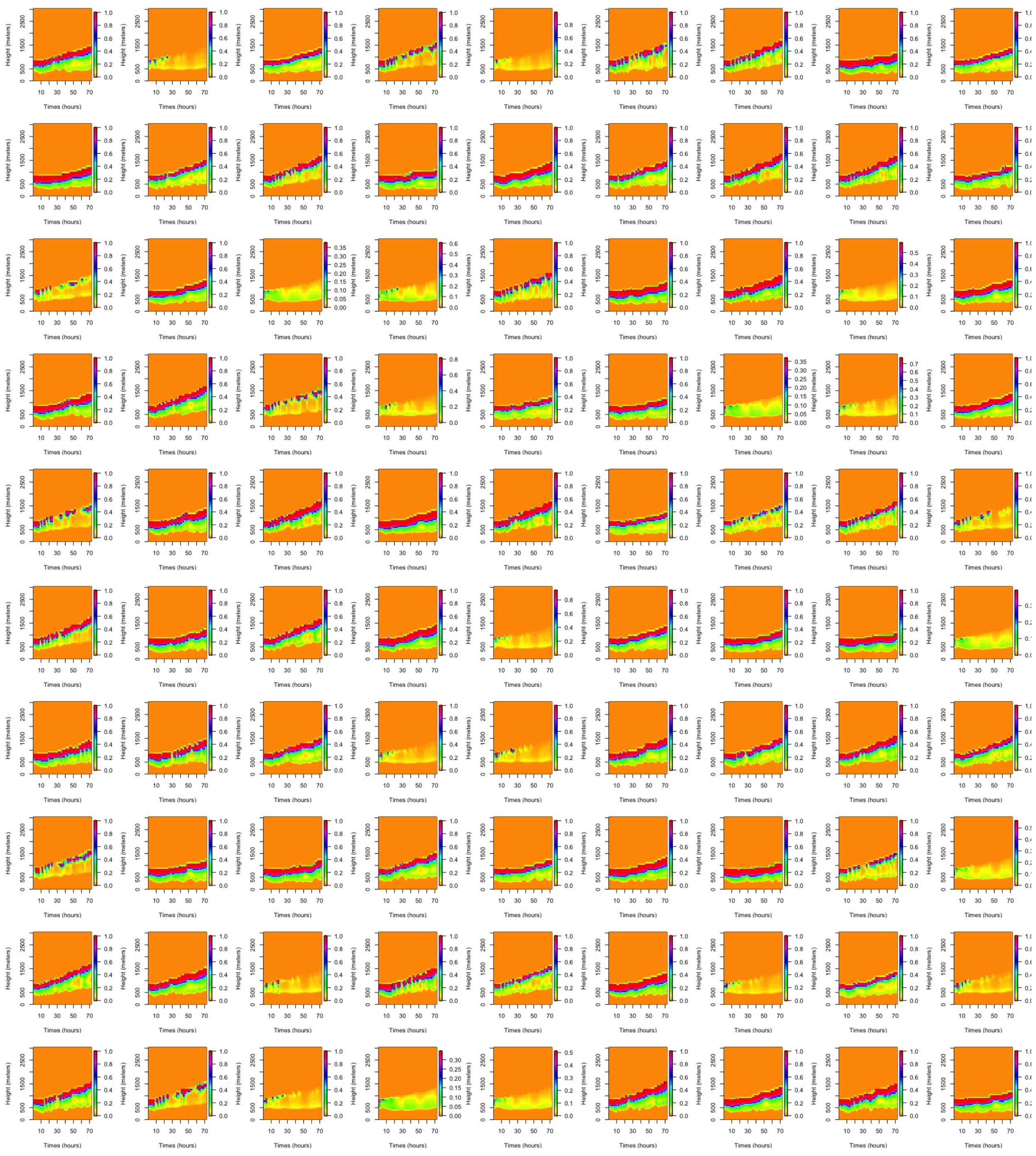


Fig. C.14 Wave 1 Ensemble runs from SCM simulators: the ensemble outputs are plotted ordinarily from the 1st run to the 90th. For each plot, it shows the hourly averages of the cloud fraction profiles during 72 hours of SCM simulation.

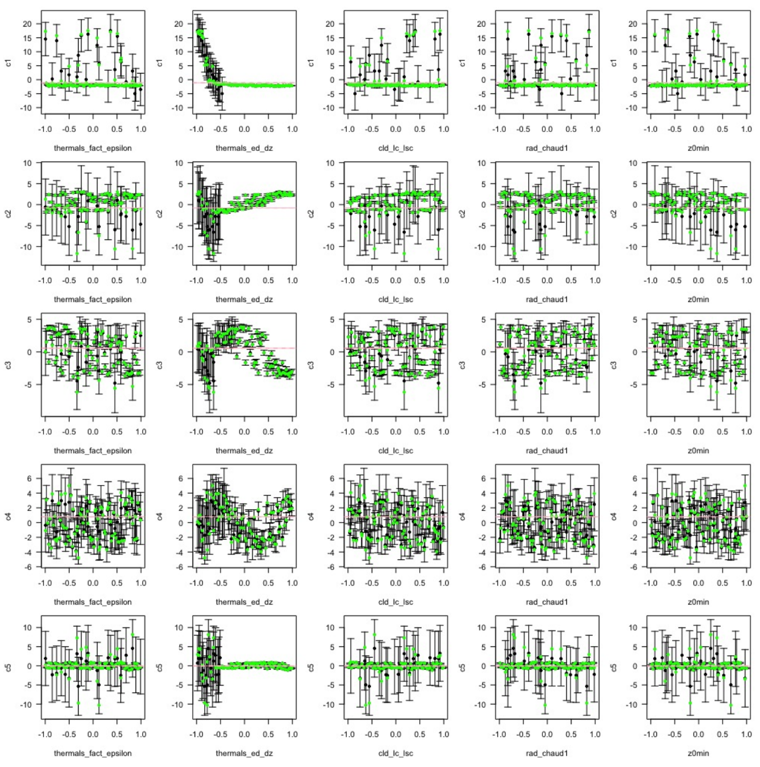


Fig. C.15 Leave-one-out cross-validation plots: wave 1 Gaussian process emulators for $\mathbf{C}(\mathbf{X})$.

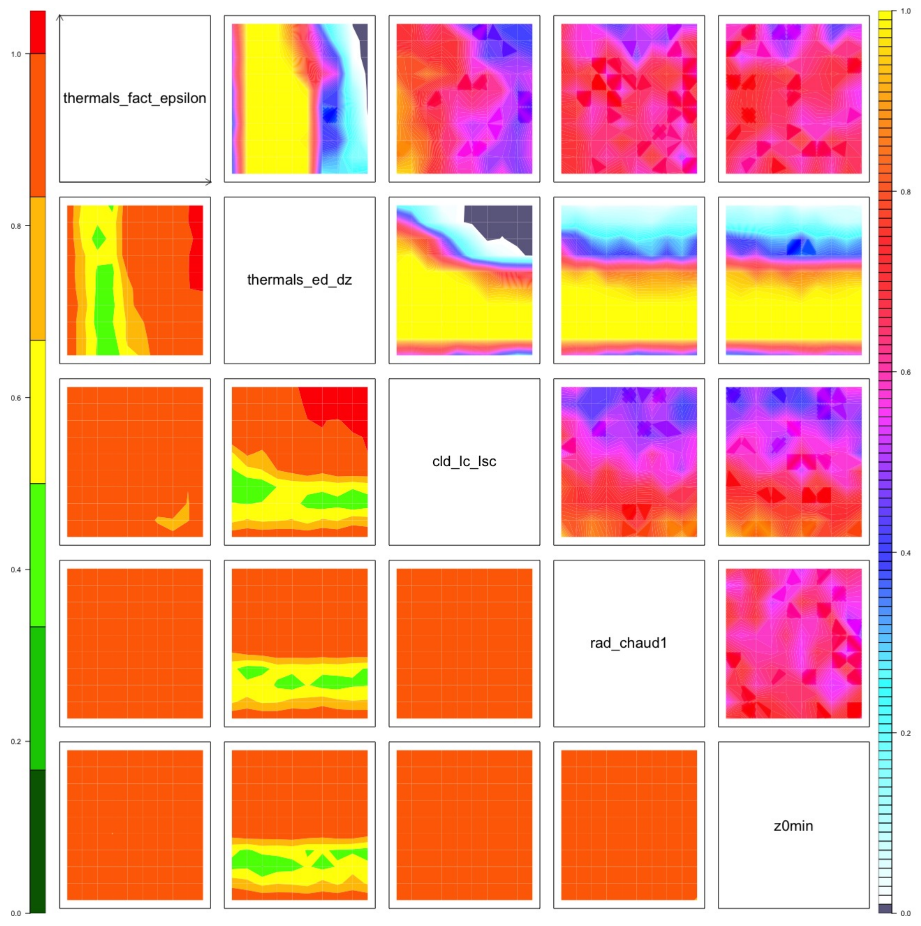


Fig. C.16 *Upper triangle:* wave 1 NROY density plots for each pair of parameters. *Lower triangle:* minimum implausibility plots for each pair of parameters.

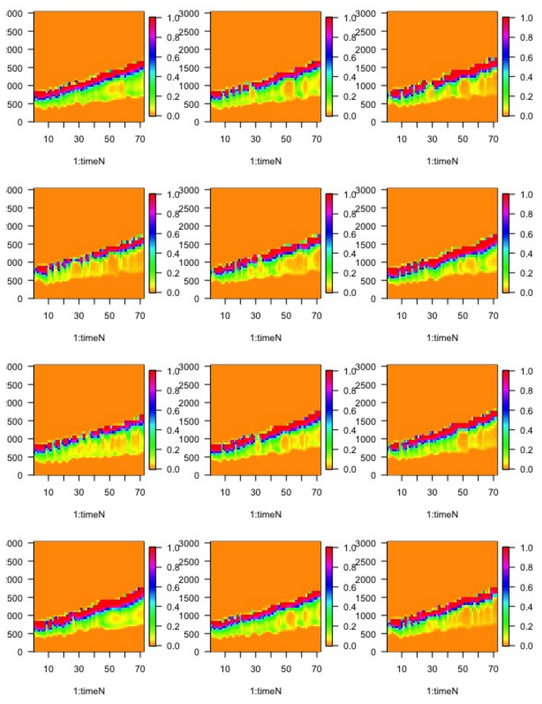


Fig. C.17 The acceptable runs by expert's selection for wave 3.

By comparing $\mathcal{X}^1$ with the NROY space shown in Figure 6.6, we can observe a similar structure of NROY density plot, which indicates that KHM would produce a similar NROY space (near true NROY) once the selected kernel returns a high score of the performance evaluation function.

C.5.2 Refocusing: wave 2 & wave 3

In wave 2 and wave 3 we have selected new runs for KHM. The new ensemble design, $\mathbf{X}_2$ and $\mathbf{X}_3$ are presented in Figure C.18 and C.19, wave 3 experts selection is presented in Figure **??**, and the Gaussian process validation plots for wave 2 and wave 3 GP emulators are given in FigureC.20 and C.21.

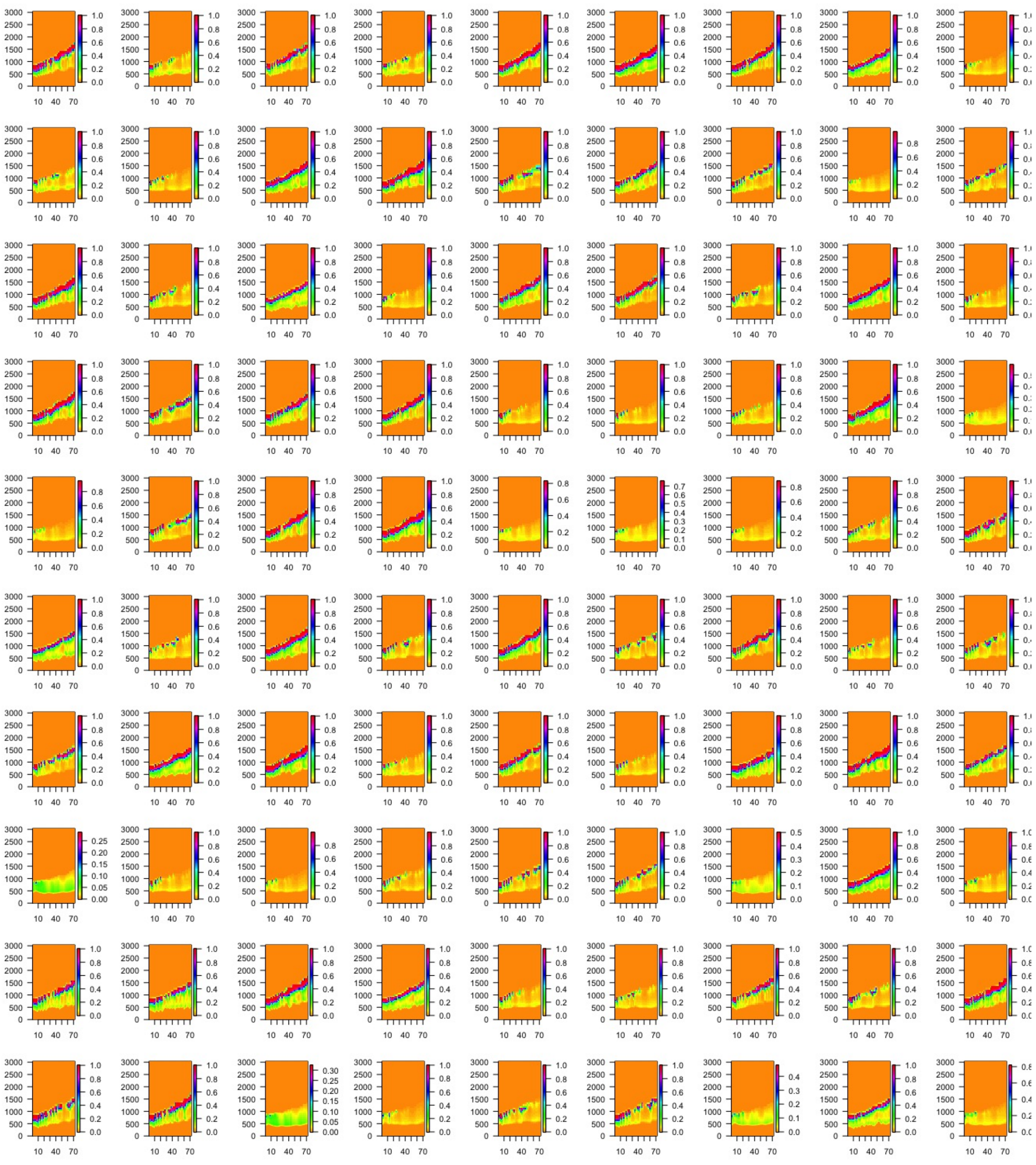


Fig. C.18 Wave 1 Ensemble runs from SCM simulators: the ensemble outputs are plotted ordinarily from the 1st run to the 90th. For each plot, it shows the hourly averages of the cloud fraction profiles during 72 hours of SCM simulation.

Fig. C.19 Wave 2 Ensemble runs from SCM simulators.

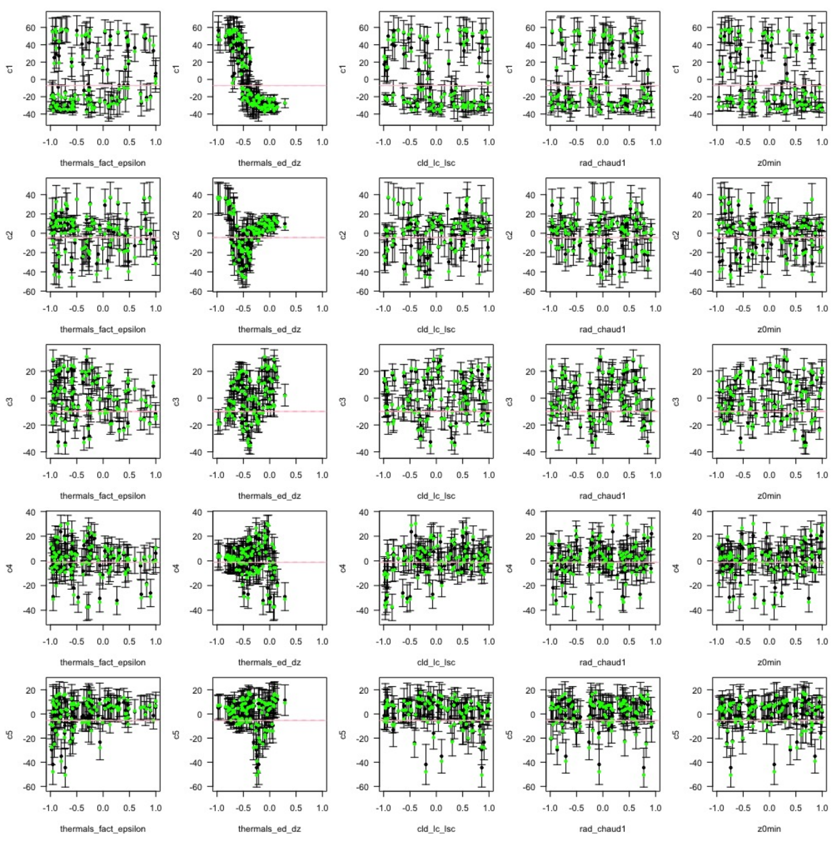


Fig. C.20 Wave 2 Leave-one-out cross-validation plots for the emulators for the coefficients on the first five basis vectors.

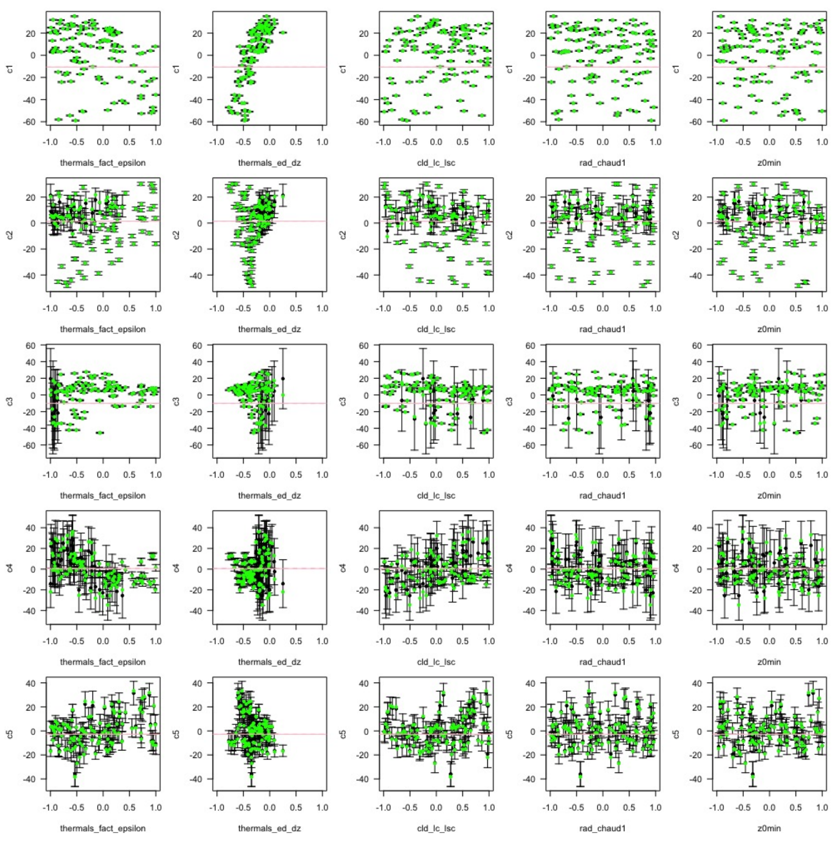


Fig. C.21 Wave 3 Leave-one-out cross-validation plots for the emulators for the coefficients on the first five basis vectors.