University of Exeter

Department of Mathematics

Uncertainty quantification for spatial field data using expensive computer models: refocussed Bayesian calibration with optimal projection

James Martin Salter

March 2017

Supervised by

Daniel Williamson

Peter Challenor

Submitted by James Martin Salter, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Mathematics , March 2017.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) .....

Abstract

In this thesis, we present novel methodology for emulating and calibrating computer models with high-dimensional output.

Computer models for complex physical systems, such as climate, are typically expensive and time-consuming to run. Due to this inability to run computer models efficiently, statistical models ('emulators') are used as fast approximations of the computer model, fitted based on a small number of runs of the expensive model, allowing more of the input parameter space to be explored. Common choices for emulators are regressions and Gaussian processes.

The input parameters of the computer model that lead to output most consistent with the observations of the real-world system are generally unknown, hence computer models require careful tuning. Bayesian calibration and history matching are two methods that can be combined with emulators to search for the best input parameter setting of the computer model (calibration), or remove regions of parameter space unlikely to give output consistent with the observations, if the computer model were to be run at these settings (history matching). When calibrating computer models, it has been argued that fitting regression emulators is sufficient, due to the large, sparsely-sampled input space. We examine this for a range of examples with different features and input dimensions, and find that fitting a correlated residual term in the emulator is beneficial, in terms of more accurately removing regions of the input space, and identifying parameter settings that give output consistent with the observations. We demonstrate and advocate for multi-wave history matching followed by calibration for tuning.

In order to emulate computer models with large spatial output, projection onto a low-dimensional basis is commonly used. The standard accepted method for selecting a basis is to use n runs of the computer model to compute principal components via the singular

value decomposition (the SVD basis), with the coefficients given by this projection emulated. We show that when the n runs used to define the basis do not contain important patterns found in the real-world observations of the spatial field, linear combinations of the SVD basis vectors will not generally be able to represent these observations. Therefore, the results of a calibration exercise are meaningless, as we converge to incorrect parameter settings, likely assigning zero posterior probability to the correct region of input space. We show that the inadequacy of the SVD basis is very common and present in every climate model field we looked at.

We develop a method for combining important patterns from the observations with signal from the model runs, developing a calibration-optimal rotation of the SVD basis that allows a search of the output space for fields consistent with the observations. We illustrate this method by performing two iterations of history matching on a climate model, CanAM4. We develop a method for beginning to assess model discrepancy for climate models, where modellers would first like to see whether the model can achieve certain accuracy, before allowing specific model structural errors to be accounted for.

We show that calibrating using the basis coefficients often leads to poor results, with fields consistent with the observations ruled out in history matching. We develop a method for adjusting for basis projection when history matching, so that an efficient and more accurate implausibility bound can be derived that is consistent with history matching using the computationally prohibitive spatial field.

Thanks to Danny Williamson for constant guidance and support throughout, and for convincing me to do a PhD in the first place! Thanks also to Peter Challenor for many interesting discussions.

Thanks to those at CCCMA that allowed us to run an ensemble on their supercomputer.

Finally, a huge thank you to my family and friends, without whom none of this would have been possible.

Contents

List of Tables

List of Figures

Publications

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

Salter, James M., and Daniel Williamson. "A comparison of statistical emulation methodologies for multiwave calibration of environmental models." Environmetrics 27.8 (2016): 507-523.

1. Introduction

Determining the input parameter settings for computer models, so that the output is consistent with real-world observations, is an important and challenging problem (Hourdin et al., 2016). Careful parameter estimation (equivalently, solving of the inverse problem, or 'tuning' in climate) of the inputs of computer models is required, so that parameter settings that lead to accurate representations of the real world can be found, allowing computer models to be used for tasks such as forecasting. This is commonly performed for climate models, so that, for example, forecasts based on different future carbon dioxide scenarios can be assessed (e.g. by the UKCP09 project (Murphy et al., 2009)). Many computer models feature large input spaces and take a long time to run, so that it is not possible to simply run the computer model at any input parameter choices of interest. Instead, exploration of the input space is achieved using a small ensemble of model output at different parameter settings.

The uncertainty quantification literature approaches this problem by building 'emulators' using the available runs of the computer model (Sacks et al., 1989b). An emulator is a fast approximation of the computer model output at input parameters $\mathbf{x}$. As this is only an approximation to the model, there is a measure of uncertainty given with emulator predictions. Within the statistics literature, the use of Gaussian process emulators is common, with a correlated residual term fitted. In a number of applications to computer experiments, regression is used instead, due to the ability to evaluate predictions more efficiently. Emulators can be used as a proxy for complicated computer codes, and can be used to identify input parameter settings that give output consistent with the observations. Within the uncertainty quantification literature, there are two main methodologies used to solve the tuning problem: Bayesian calibration (Kennedy and O'Hagan, 2001a) and history matching (Craig et al., 1996).

Bayesian calibration assumes that a 'best input' setting, $\mathbf{x}^*$, of the input parameters exists, such that the model output when run at $\mathbf{x}^*$ is consistent with the observations, $\mathbf{z}$, up to observation error and model discrepancy (the difference between the real world and the best setting of the model). Conditional on the observed runs of the model, and emulators built for the model output, a posterior distribution for $\mathbf{x}^*$ is found. A limitation of this method is that because the result is a distribution, the posterior density must always integrate to 1. Therefore, it is not possible for the result to be that there are no parameter settings that lead to output consistent with $\mathbf{z}$, whereas this may be the case in applications.

History matching requires no distributional assumptions, and makes no assumption about the existence of $\mathbf{x}^*$, instead ruling out regions of the input parameter space that are considered unlikely to give model output consistent with $\mathbf{z}$. The resulting not ruled out space contains parameter settings that are 'not implausible', i.e. it is not yet known whether the model output here will match $\mathbf{z}$, but we cannot rule out these parameter settings, based on the current knowledge about the model, from the known model runs and the emulators. History matching is well suited to an iterative or 'refocussed' approach, with new ensembles designed in the current not ruled out space (Vernon et al., 2010). This allows more accurate emulation in the regions of parameter space that are of interest, i.e. the regions that may lead to output consistent with $\mathbf{z}$.

Bayesian calibration and history matching are both methods that can be applied to determine input settings for computer models with high-dimensional output, for example spatial or temporal data. This requires the emulation of multiple values. The most efficient method for achieving this is via projection of the output onto a low-dimensional basis representation, to reduce the complexity of calculations, such as variance matrix inversions. For reducing the dimensionality of spatial fields, a common basis choice is the SVD basis, given by calculating the principal components of the ensemble, with emulators built for the coefficients given by projection onto this basis (Higdon et al., 2008a). This basis is the default choice in calibration exercises, particularly for climate models (Sexton et al., 2011, Holden et al., 2013, Chang et al., 2014a).

Defining the low-dimensional basis using the ensemble of model runs, which is likely to be small for expensive computer models, and performing a successful calibration using this basis, relies on the assumption that the ensemble contains the main modes of variability from the observations, $\mathbf{z}$. Whilst it is generally assumed that the computer model is a

reasonable approximation of the real-world system that it represents, there is no reason that using a basis derived from a small ensemble of model runs will be completely suited to representing $\mathbf{z}$, unless there is little discrepancy in the model, and we have by chance explored the correct region of the large input parameter space in the small ensemble. The assumption that this basis choice requires has not been explored adequately in the literature, with the SVD basis becoming an automatic choice in many applications of high-dimensional Bayesian calibration.

1.1. Aims

In this thesis, we explore perceived wisdoms in emulation and calibration. Regression-only emulators are often used in place of Gaussian process emulators in calibration and history matching exercises, when the input space is large. When tuning high-dimensional spatial fields, the output is commonly projected onto the SVD basis, with emulators built, and Bayesian calibration performed using the coefficients on this basis. We investigate whether these are sensible choices, and develop alternative methodology.

More specifically, we aim to answer the following questions:

- Is there any benefit in fitting Gaussian process emulators in high-dimensional input parameter spaces, or are regression emulators sufficient?
- Does performing Bayesian calibration after a multi-wave refocussed history match lead to a greater accuracy in results, compared to calibrating using the initial ensemble?
- When the observations for a spatial field are not similar to observed model runs, is projecting onto the SVD basis an appropriate choice if we wish to learn about whether there are parameter settings of the computer model that represent the observations?
- How can we define more appropriate basis choices, based on the observations, model runs and physical knowledge?
- How can we efficiently and accurately history match a large spatial field?

- When a specification for the discrepancy variance is not available, how can we quantify this?

- When the output of the spatial field is too large for history matching to be performed over the field, how can we accurately history match using the basis coefficients?

1.2. Structure of the thesis

In Chapter 2, we outline the current literature in uncertainty quantification, with a focus on Gaussian process emulation, Bayesian calibration, and history matching, both for models with scalar output and multivariate output. We briefly discuss other methods for tuning climate models.

Chapter 3 provides a comparison of two types of emulators, regressions and Gaussian processes, in the context of a multi-wave history matching experiment, for functions with a single output. We investigate whether the correlated residual term is necessary when faced with sparse samples from large parameter spaces, and compare history matching and Bayesian calibration results for each emulator type, across multiple waves, for four examples with features commonly found in computer models. We then assess the performance of the emulator types for an environmental model.

In Chapter 4, we explore emulation and calibration for models with large spatial output fields. We discuss the drawbacks of using the SVD basis for projection, when the observations do not lie in the low-dimensional subspace spanned by the ensemble. We illustrate the problem with an idealised example and for two climate models. We then develop a method for combining important elicited patterns with basis vectors derived from the ensemble, in order to find a basis that gives a more accurate representation of the observations.

In Chapter 5, we extend the previous basis selection methodology into an automated iterative method, based on a rotation of the SVD basis. This method identifies the optimal basis for the goal of building emulators and searching for parameter settings that can reproduce the observations. We investigate extensions required in order to find optimal rotations for applications where there is a known structure to the observation error and

discrepancy variances.

In Chapter 6, we extend the earlier methodology for history matching a spatial field to overcome the challenges faced when the spatial field is large. We develop a Bayesian hierarchical model for linking the implausibility over the field with the implausibility given by the basis coefficients. We design a new ensemble of the climate model CanAM4, by selecting optimal bases for three output fields, and history matching using our model for the implausibilities to select the appropriate bound. We provide methods for defining a spatial discrepancy, based on selecting patterns deemed to be a structural error, setting the values so that the observations will not be ruled out when history matching.

2. Literature review

2.1. Computer models

A computer model is a collection of code (often, thousands of lines) representing a real system that acts as a function, taking a set of input parameters $\mathbf{x}$ and returning output $f(\mathbf{x})$. Computer models are used to simulate real-world systems, to either help understand or predict the system under various scenarios (for example, studying the effect of different emissions scenarios on climate change (Johns et al., 2003)), or to accurately represent historical events (for example, matching historical output of a hydrocarbon reservoir (Craig et al., 1997)).

Examples of areas that computer models have been used in include climate (Kiehl et al., 1998, Gordon et al., 2000, Pope et al., 2000, Meehl et al., 2007), oil reservoir modelling (Tavassoli et al., 2004), cosmology (Vernon et al., 2010, Gómez et al., 2012), simulating geomagnetic storms (Heaton et al., 2015) as well as in biological applications such as epidemic modelling (Farah et al., 2014), DNA modelling (Henderson et al., 2012) and heart modelling (Harrild and Henriquez, 2000).

The output of a computer model can take many different forms, for example a single value, a time series, a spatial field, or it may give multiple different outputs simultaneously. For climate models, the output is given for each grid box over the entire world (for global climate models, e.g. CanAM4 (von Salzen et al., 2013), over a horizontal and vertical grid) or a region (e.g. over the western United States, as in Dickinson et al. (1989)), with several different output fields, e.g. temperature, precipitation, and other aspects of the climate.

The input parameters may range from inputs directly linked to physical processes, to

parameters that control numerical integrations or other parametrisations of physical processes. Models may have only a small number of input parameters, for example the 3 input parameters in the Lyon-Fedder-Mobarry model in Heaton et al. (2015) or the IC fault model (Tavassoli et al., 2004), to tens or more parameters in climate models (e.g. 27 parameters for the atmosphere and ocean in HadCM3 (Williamson et al., 2015, Gordon et al., 2000)).

Due to the complexity of modelling some real-world systems, especially the global climate, computer models require a large amount of computing resources (Williamson et al., 2015). Therefore, if the goal is to better explore the input parameter space and tune the model, or perform other inference, statistical methods are required.

2.2. Uncertainty quantification

Uncertainty quantification is the general term for a group of methods that are used to analyse and make inferences about computer model output.

Kennedy and O'Hagan (2001a) discuss the types of uncertainty that can be introduced when seeking to model or learn about a real-world system, and how they may be accounted for. One of the common goals of uncertainty quantification is to tune or calibrate the input parameters of the model (see Section 2.5). Uncertainty is introduced here as the values of these parameters are unknown. Although they may represent real processes, parametrisations often contain simplifications, so that it may not be possible to set these parameters based purely on physical knowledge. The uncertainty associated with not knowing the true parameter values is known as 'parameter uncertainty'.

Another source of uncertainty arises from the fact that it is often not possible to run the computer model as often as may be required ('code uncertainty'). Therefore, an alternative statistical representation of the model may be necessary (emulators, see Section 2.3). This is not the same as running the true model, and hence any output given by the statistical model will not be known with certainty (unless the computer model is deterministic, and we have observed the output at this input setting). Emulators are used to quantify this uncertainty, by giving uncertainty bounds on any predictions.

More uncertainty is introduced due to the fact that computer models are generally not perfect. Physical systems represented by computer models can be extremely complex (e.g. global climate), and hence it is not possible to perfectly parametrise every single atmospheric or oceanic process. Some processes may occur on a more local level than the resolution of the global model ('sub-grid scale modelling' (Chaboureau and Bechtold, 2005)), and approximations are required to represent these (von Salzen et al., 2013). Therefore, it is likely that there is some inherent discrepancy between the real world and the model, also referred to as model inadequacy. In addition to this, real world observations of a process may be imperfect, for a variety of reasons (for example, a limitation of instruments or human error). This error must also be considered.

Sensitivity analysis is another common tool in the uncertainty quantification literature (Saltelli et al., 1999, 2000). The goal here is to identify how the input parameters affect the model output. There are two main types of sensitivity analysis that are performed: local, where the inputs are varied by small amounts, and global, where the entire input parameter space can be considered.

In local sensitivity analysis, derivatives are calculated with respect to each input parameter in turn, with every other parameter fixed at chosen values (Turányi, 1990). This identifies how the output of the model changes with respect to each parameter in turn. To investigate the effect that potential interactions have on the output, global sensitivity analysis instead varies all parameters simultaneously. Saltelli et al. (1999) calculate the contribution of each input variable on the variability in the output, accounting for both individual effects of the inputs and any joint effects between parameters.

2.3. Emulation

An emulator is a statistical model that is used in lieu of being able to run a computer model as often or efficiently as is required (Sacks et al., 1989b, Currin et al., 1991, Haylock and O'Hagan, 1996, Santner et al., 2003). Emulators can be used as a surrogate for the computer model in future analyses, such as Bayesian calibration (Kennedy and O'Hagan, 2001a) and history matching (Craig et al., 1996). Statistical emulators account for the code uncertainty introduced when replacing the computer model with an approximation

through the provision of uncertainty on any predictions from the emulator.

Given a computer model $f(\cdot)$, a vector-valued function taking inputs $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is the p-dimensional space of possible model inputs, and an ensemble of runs $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$, an emulator for $f(\cdot)$ can be constructed. The input space $\mathcal{X}$ may be continuous, with input parameters x_i allowed to take on any value in a finite, continuous interval, but may also contain discrete 'switch' parameters (or 'factors'), as is the case for the climate model HadCM3 (Gordon et al., 2000). Switch parameters have a finite number of allowable values. Hereafter, we assume that all parameters are continuous.

For output i of the computer model, the general form of an emulator is (Sacks et al., 1989b)

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

where $h_j(\mathbf{x})$ are chosen functions of the parameters (e.g. linear terms, interactions between the parameters, and higher powers thereof), β_{ij} are unknown coefficients to be estimated, and ϵ_i is the residual. It is assumed that the β_{ij}s and ϵ_i are independent.

In (2.1), if the residual term $\epsilon_i(\mathbf{x})$ is assumed to have mean zero and a constant variance σ^2 for all $\mathbf{x}$, and is assumed to be uncorrelated, then the emulator is a linear regression model, where a polynomial surface is fitted to the model output (Rougier et al., 2009, Sexton et al., 2011, Williamson et al., 2013, Holden et al., 2013, Williamson et al., 2015). An argument used in favour of this approach is that it is computationally efficient (in the sense that it is quick to evaluate predictions for large quantities of input points), as well as being simple to fit, with only the coefficients needing to be estimated.

2.3.1. Gaussian processes

A Gaussian process is a type of emulator that allows correlations between the output at different inputs to be incorporated into the emulator. A Gaussian process is a stochastic process where the joint distribution of a finite number of random variables from this process is Gaussian (Rasmussen and Williams, 2006), and is defined by a mean function and a correlation function:

$$f_i(\cdot)|\boldsymbol{\beta}_i, \boldsymbol{\delta}, \sigma^2 \sim GP(m_i(\cdot), \sigma^2 C_i(\cdot, \cdot)) \tag{2.2}$$

where $\boldsymbol{\beta}_i$ is the vector of parameters in the mean function $m_i(\cdot)$, $\boldsymbol{\delta}$ is a vector of parameters used to define the correlation function $C_i(\cdot,\cdot)$, and σ^2 is a variance parameter to be estimated. $C_i(\mathbf{x},\mathbf{x}')$ gives the correlation between the response at points $\mathbf{x}$ and $\mathbf{x}'$ in the input space.

In the formulation of (2.1), this is equivalent to setting

$$m_i(\mathbf{x}) = \sum_{j=1}^{k} \beta_{ij} h_j(\mathbf{x})$$

and

$$\mathrm{Cov}(\epsilon_i(\mathbf{x}), \epsilon_i(\mathbf{x}')) = \sigma^2 C_i(\mathbf{x},\mathbf{x}')$$

so that $\epsilon_i(\cdot)$ is the systematic departure from the linear model, represented by a Gaussian process with mean zero and the above covariance function (Sacks et al., 1989a).

Correlation functions

Correlation functions are generally defined so that as the distance between points $\mathbf{x}$ and $\mathbf{x}'$ in the input space $\mathcal{X}$ increases, the correlation $C(\mathbf{x},\mathbf{x}')$ decreases to zero. Santner et al. (2003) discuss various choices of $C(\cdot,\cdot)$, with different choices of the correlation function giving varying degrees of smoothness. For ease of computation, correlation functions that are weakly stationary (i.e. the function is only dependent on the distance between points, regardless of location in $\mathcal{X}$) are typically used.

A common choice of correlation function is the Gaussian or squared exponential (Kennedy and O'Hagan, 2000, 2001a, Vernon et al., 2010, Wilkinson, 2010):

$$C(\mathbf{x},\mathbf{x}') = \exp\left\{ -\sum_i \left(\frac{x_i - x_i'}{\delta_i}\right)^2 \right\} \tag{2.3}$$

where $\boldsymbol{\delta}$ is a vector of non-negative correlation length parameters. In each input dimension, the difference between the two input parameters is squared and then scaled by the square of the corresponding entry of the correlation length parameter vector. As the distance between inputs increases, the correlation between them tends to zero.

A more general form of the squared exponential correlation function is the power exponential, where instead of the second power there is a new power parameter, α_i, to be estimated for each input dimension (Higdon et al., 2004, 2008a):

$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\sum_i \left(\frac{x_i - x_i'}{\delta_i}\right)^{\alpha_i} \right\}$$

where decreasing the power from 2 reduces the smoothness of the function.

Another regularly-used correlation function is the Matérn function, defined by non-negative correlation length parameters δ and smoothness parameter α (Golchi et al., 2015, Tripathy et al., 2016):

$$C(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\alpha}}{\Gamma(\alpha)} \left(\frac{|\mathbf{x} - \mathbf{x}'|}{\delta} \right)^{\alpha} \mathcal{K}_\alpha \left(\frac{|\mathbf{x} - \mathbf{x}'|}{\delta} \right)$$

where $|\mathbf{x} - \mathbf{x}'|$ is the distance between two points in the input space, $\Gamma(\cdot)$ is the gamma function, and $\mathcal{K}_\alpha(\cdot)$ is the modified Bessel function of the second kind (Watson, 1995). Rather than a single parameter δ scaling the distance, this can be scaled in each input dimension as in the squared exponential.

The Matérn correlation function does not give the assumption of infinite differentiability, whereas the squared exponential does make this assumption. As α increases to infinity, the Matérn converges to the squared exponential (Nychka et al., 2002), and hence the smoothness of the correlation increases with α. A potential downside of the Matérn (and power exponential) correlation function is that it has an extra parameter(s) to be estimated compared to the squared exponential.

If there are a large number of ensemble runs, then calculating the full $n \times n$ ensemble correlation matrix and inverting it may be challenging. There are alternative methods for defining a correlation function for the situation of large n. Kaufman et al. (2011), for example, impose sparsity in the correlation matrix, so that if the distance between two points is above a chosen threshold, the correlation between these two points is set to be zero. This simplification allows the appropriate calculations involved in emulation to be performed. However, as the ensemble size is likely to be relatively small in any applications used in this thesis, due to the complexity of climate models, the above correlation functions, such as the squared exponential, are suitable.

The nugget parameter

A Gaussian process as described in Section 2.3.1 will interpolate the data exactly, with zero variance at observed points. This may not always be a useful property, hence Craig et al. (1996) extend the emulator in (2.1) by adding a nugget term $\nu_i(\cdot)$, independent to β_{ij} and $\epsilon_i(\cdot)$:

$$f_i(\mathbf{x}) = \sum_{j=1}^{k} \beta_{ij} h_j(\mathbf{x}) + \epsilon_i(\mathbf{x}) + \nu_i(\mathbf{x}) \tag{2.4}$$

This has the effect of removing the condition that the model exactly interpolates the data points from $\mathbf{F}$, i.e. there is now a non-zero variance at observed points.

There are a number of reasons why it may be desirable to include a nugget in the emulator. Craig et al. (1996) and Craig et al. (2001) divide the input variables in $\mathbf{x}$ into active and inactive variables, where the active variables are those that are included in $h_j(\mathbf{x})$. The nugget is then included to account for the variation in $f_i(\mathbf{x})$ that is due to the inactive variables. The nugget is modelled with a zero mean and the same variance σ_ν^2 for all $\mathbf{x}$, and is assumed to be uncorrelated with itself at different inputs, i.e.

$$\mathrm{Cov}(\nu_i(\mathbf{x}), \nu_i(\mathbf{x}')) = \begin{cases} \sigma_\nu^2 & \text{if } \mathbf{x} = \mathbf{x}' \\ 0 & \text{otherwise} \end{cases}$$

In the case of emulating climate model output, it is necessary to include a nugget due to the internal variability in these models: varying the initial conditions can lead to different model output for the same $\mathbf{x}$ (Hawkins and Sutton, 2009, Williamson and Blaker, 2014). In this setting, an emulator that exactly interpolates model runs is inappropriate.

Andrianakis and Challenor (2012) discuss other reasons for including a nugget term in an emulator, and investigate the effects of varying the nugget. One such reason is to overcome numerical problems with the correlation matrix of the ensemble, which must be inverted when fitting an emulator (Kennedy and O'Hagan, 2001a). Gramacy and Lee (2012) argue that including a nugget term can lead to a better emulator, in terms of predictive accuracy.

Fitting a Gaussian process emulator

A Bayesian approach is typically used to fit a Gaussian process emulator to data, so that prior knowledge can be incorporated (Currin et al., 1991), and then updated to give a posterior having observed an ensemble, $\mathbf{F}$, of runs from the computer model. In this section, it is assumed that there is only one scalar output given by $f(\cdot)$, so that the subscript i can be dropped in order to give greater clarity. Furthermore, let $\boldsymbol{\phi}$ be the vector containing all of the parameters for the correlation function.

Haylock and O'Hagan (1996) assume a priori that $f(\cdot)$ is (or, more likely, can be represented by) a Gaussian process , i.e. that

$$f(\cdot)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi} \sim \mathrm{N}(\mathbf{h}(\cdot)^T\boldsymbol{\beta}, \sigma^2 C(\cdot,\cdot)) \tag{2.5}$$

where $\mathbf{h}(\cdot) = (h_1(\cdot), \ldots, h_k(\cdot))^T$ is the vector containing the functions of the parameters that are used in the mean function.

Prior distributions are required for the unknown parameters $\boldsymbol{\beta}$ and σ^2. In this paper, a 'non-informative' prior is assumed:

$$P(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2} \tag{2.6}$$

as the modeller may not have any prior beliefs about these parameters before running $f(\cdot)$. A benefit of this prior is that the posterior analysis is tractable: it is possible to write down the posterior conditioned on the ensemble.

Using these descriptions of the prior distributions, a posterior distribution for $f(\cdot)$ can be found, conditioned on the ensemble $\mathbf{F}$:

$$f(\cdot)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\phi}, \mathbf{F} \sim \mathrm{N}(m^*(\cdot), \sigma^2 C^*(\cdot,\cdot))$$

where

$$\begin{aligned} m^*(\mathbf{x}) &= \mathbf{h}(\mathbf{x})^T\boldsymbol{\beta} + \mathbf{t}(\mathbf{x})^T\mathbf{A}^{-1}(\mathbf{F} - \mathbf{H}\boldsymbol{\beta}) \\ C^*(\mathbf{x}, \mathbf{x}') &= C(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T\mathbf{A}^{-1}\mathbf{t}(\mathbf{x}') \end{aligned} \tag{2.7}$$

where $\mathbf{H}$ is the $n \times k$ design matrix with i, j^{th} entry $h_j(\mathbf{x}_i)$, $\mathbf{A}$ is the $n \times n$ correlation

matrix with i, j^{th} entry $C(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{t}(\cdot)$ is a vector of length n with i^{th} entry $C(\cdot, \mathbf{x}_i)$.

By integrating out $\boldsymbol{\beta}$, as the true value of this is not known, the posterior distribution becomes the Gaussian process

$$f(\cdot)|\sigma^2, \boldsymbol{\phi}, \mathbf{F} \sim \mathrm{N}(m^{**}(\cdot), \sigma^2 C^{**}(\cdot, \cdot)) \tag{2.8}$$

where

$$m^{**}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})^T \mathbf{A}^{-1}(\mathbf{F} - \mathbf{H}\hat{\boldsymbol{\beta}}) \tag{2.9}$$

and

$$C^{**}(\mathbf{x}, \mathbf{x}') = C(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T \mathbf{A}^{-1} \mathbf{t}(\mathbf{x}') + (\mathbf{h}(\mathbf{x})^T - \mathbf{t}(\mathbf{x})^T \mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^T \mathbf{A}^{-1} \mathbf{H})^{-1}(\mathbf{h}(\mathbf{x}')^T - \mathbf{t}(\mathbf{x}')^T \mathbf{A}^{-1}\mathbf{H})^T \tag{2.10}$$

for $\hat{\boldsymbol{\beta}} = (\mathbf{H}^T \mathbf{A}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{A}^{-1} \mathbf{F}$. Finally, by integrating out the variance σ^2, the result is a t-distribution for $f(\mathbf{x})|\boldsymbol{\phi}, \mathbf{F}$:

$$\frac{f(\mathbf{x}) - m^{**}(\mathbf{x})}{\sqrt{\frac{\mathbf{F}^T(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}(\mathbf{H}^T\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{A}^{-1})\mathbf{F} C^{**}(\mathbf{x}, \mathbf{x})}{n - q - 2}}} \sim t_{n-q} \tag{2.11}$$

where q is the rank of the design matrix $\mathbf{H}$. From this distribution, a prediction of the model output at parameter setting $\mathbf{x}$ can be evaluated, along with the uncertainty on this prediction, given $\mathbf{F}$ and values for the correlation parameters $\boldsymbol{\phi}$.

This formulation does not account for the inclusion of a nugget in the emulator. However, the method of Haylock and O'Hagan (1996) can be adapted to include a nugget by replacing the original correlation function $C(\cdot, \cdot)$ with

$$\tilde{C}(\mathbf{x}, \mathbf{x}') = \nu I_{x=x'} + (1 - \nu) C(\mathbf{x}, \mathbf{x}')$$

where the nugget parameter ν represents the proportion of residual variability that is not explainable by the correlation function. Replacing $C(\cdot, \cdot)$ with $\tilde{C}(\cdot, \cdot)$ everywhere it appears (for example, in the definitions of $\mathbf{A}$ and $\mathbf{t}(\cdot)$) yields the same posterior as in (2.11), with the nugget included as an extra parameter in the correlation function.

Other choices can be made for the form of the prior distribution for the model parameters.

Oakley and O'Hagan (2002) use a normal inverse gamma prior in place of (2.6), i.e.

$$P(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-\frac{1}{2}(r+q+2)} \exp[-\{(\boldsymbol{\beta}-\boldsymbol{\mu})^T \boldsymbol{V}^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}) + a\}/(2\sigma^2)] \tag{2.12}$$

where $r, \boldsymbol{\mu}, V$ and a are the parameters of this distribution, and must be specified. The inclusion of these parameters changes the form of the posterior distribution, yielding:

$$\frac{f(\mathbf{x}) - m^{**}(\mathbf{x})}{\hat{\sigma}\sqrt{C^{**}(\mathbf{x}, \mathbf{x})}} \sim t_{r+n}$$

where the equation for $m^{**}(\mathbf{x})$ is as in (2.9), and

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

for

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \mathbf{V}^*(\mathbf{V}^{-1}\boldsymbol{\mu} + \mathbf{H}^T\mathbf{A}^{-1}\mathbf{F}) \\
\hat{\sigma^2} &= \frac{a + \boldsymbol{\mu}^T\mathbf{V}^{-1}\boldsymbol{\mu} + \mathbf{F}^T\mathbf{A}^{-1}\mathbf{F} - \hat{\boldsymbol{\beta}}^T(\mathbf{V}^*)^{-1}\hat{\boldsymbol{\beta}}}{n + r - 2} \\
\mathbf{V}^* &= (\mathbf{V}^{-1} + \mathbf{H}^T A^{-1}\mathbf{H})^{-1}
\end{aligned}$$

Oakley and O'Hagan (2002) note that the weak form of their prior is simply the prior used by Haylock and O'Hagan (1996) (equation (2.6)), which is equivalent to assigning an infinite prior variance to $f(\mathbf{x})$, but that the full version of the prior should be used if prior knowledge about $f(\cdot)$ is available. The parameters relating to prior distributions are often referred to as hyperparameters. Oakley (2002) discusses how the hyperparameters of the prior in (2.12) may be elicited, by first asking experts general questions about properties of the model output, and then translating these into probabilistic judgements.

Estimating parameters

When fitting an emulator, there are a number of parameters that must be estimated in order to complete the posterior distribution. The coefficients $\boldsymbol{\beta}$ for the mean function and the variance σ^2 are typically integrated out, removing the dependency on these quantities, as in the previous section. However, the posterior distribution of the emulator still depends on the parameters $\boldsymbol{\phi}$ of the correlation function, which need to be considered before

predictions can be evaluated from the emulator.

Sacks et al. (1989b) find the minimum of the following expression, which is only dependent on the ensemble and the correlation lengths:

$$(det\mathbf{A})^{\frac{1}{n}}\hat{\sigma}^2$$

where

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

After minimising this function, the resulting value for $\boldsymbol{\phi}$ is 'plugged-in' to the emulator, with these estimates treated as fixed, rather than uncertain, values. This completes the specification of the emulator.

Kennedy and O'Hagan (2001a) take a similar approach, and fix the hyperparameters at estimated values after conditioning on the ensemble. Fixed values are used in order to reduce the computational burden required to calculate or sample from the intractable posterior distribution. Fixing the hyperparameters of the model to simplify future calculations, usually via using the maximum likelihood estimates for them, is a common technique (Currin et al., 1991, Oakley and O'Hagan, 2002, Bayarri et al., 2007, Conti et al., 2009, Bhat et al., 2010), and is justified by arguing that the uncertainty due to the estimation of these parameters is small compared to the other sources of uncertainty, hence the increased computational time required is unnecessary.

In order to fully account for uncertainty in the model parameters, $\mathbf{x}$ (as these have unknown values), a full Bayesian analysis can be carried out (Higdon et al., 2004). Prior distributions are specified for all of the unknown parameters, and a joint posterior distribution, conditional on the observations and the model runs, can be written down. This distribution is difficult to write analytically, and hence is sampled from numerically using MCMC: the posterior for the output of $f(\cdot)$ at a value of the input parameters is calculated, with all uncertainty accounted for.

2.3.2. Bayes linear methods

An alternative to the usual Bayesian approach to fitting emulators comes through the use of Bayes linear methods (Goldstein and Wooff, 2007). Instead of specifying full probability distributions for the model parameters as in the full Bayesian approach, the Bayes linear approach requires only 2nd-order specifications. This relaxation of the need for distributional assumptions can simplify the calculation of the posterior distribution of the emulator.

Given observed data D, the prior expectation and variance of a quantity B can be updated to give the adjusted expectation and variance of B given D:

$$\mathrm{E}_D(B) = \mathrm{E}(B) + \mathrm{Cov}(B, D)[\mathrm{Var}(D)]^{-1}(D - \mathrm{E}(D))$$
$$\mathrm{Var}_D(B) = \mathrm{Var}(B) - \mathrm{Cov}(B, D)[\mathrm{Var}(D)]^{-1}\mathrm{Cov}(D, B)$$

Craig et al. (1996) describe fitting an emulator using Bayes linear methods. Prior specifications are required for each of the quantities in the statistical model from (2.4). Each coefficient β_{ij} in the linear component of the model requires a prior mean and variance. The residual term is given a correlated structure (as in Gaussian process emulation), with a zero prior expectation and a chosen covariance function relating the output at different input values. Finally, a nugget term is also included, and this is given a prior mean equal to zero, and some prior variance. The different terms in the emulator are assumed to be independent. This simplifies the prior specification, as the covariances between them are all set equal to zero.

Given an ensemble of model runs $\mathbf{F}$, the emulator for $f(\cdot)$ can be updated using the equations for the adjusted expectation and variance (Craig et al., 2001):

$$\mathrm{E}_{\mathbf{F}}(f(\mathbf{x})) = \mathrm{E}(f(\mathbf{x})) + \mathrm{Cov}(f(\mathbf{x}), \mathbf{F})[\mathrm{Var}(\mathbf{F})]^{-1}(\mathbf{F} - \mathrm{E}(\mathbf{F}))$$
$$\mathrm{Var}_{\mathbf{F}}(f(\mathbf{x})) = \mathrm{Var}(f(\mathbf{x})) - \mathrm{Cov}(f(\mathbf{x}), \mathbf{F})[\mathrm{Var}(\mathbf{F})]^{-1}\mathrm{Cov}(\mathbf{F}, f(\mathbf{x}))$$

This expectation and variance has a similar form to the mean and covariance where the emulator has been conditioned on the ensemble in the full Bayesian case (2.7). However, due to the distributional assumptions required for the full Bayesian case, these are not the same. Using this conditioning, predictions can be made for unseen $\mathbf{x}$ similarly as in the

full Bayesian case.

Not requiring distributional assumptions is a powerful simplification. This approach clearly lends itself to tuning a model via history matching (see Section 2.7), where again only expectations, variances and covariances are required for the extra statistical model parameters introduced. However, an emulator of this form is unsuitable for other goals of uncertainty quantification, for example Bayesian calibration (Section 2.5), as full distributions are required, and given as the results of the calibration.

In this thesis, history matching and Bayesian calibration are compared and combined, and as such adopting a Bayes linear approach to emulation may not be suitable. In order to perform Bayesian calibration, emulators admitting full probability distributions are required, but these fully Bayesian emulators can however still be used in history matching. Therefore, full Bayesian emulators will be used hereafter to provide a fair comparison between both tuning methods, and to avoid the need to construct separate emulators. We note that the additional distributional specification of full Bayesian emulators implies that they should pass more stringent diagnostics than their Bayes linear counterparts.

2.4. Multivariate emulation

The above equations for univariate computer model output can be extended to cases where the output is multivariate. Computer models give various types of multivariate output, including time series, spatial fields, spatio-temporal fields, and different attributes of a physical system (e.g. salinity and sea surface temperature in an ocean model). Multivariate outputs can require different emulation approaches due to the nature of relationships and correlations between, for example, local grid boxes in a spatial model, and adjacent time points.

One approach for emulating multivariate output is to build univariate emulators using the methodology of Section 2.3 for each of the responses separately, as in Lee et al. (2013). In this paper, the output is the monthly mean at every spatial location (equivalently, every grid box) for a time series of months, and a Gaussian process emulator is fitted independently to the scalar output for each month and grid box, ignoring any spatial or temporal correlations over the output space. A drawback of this method is that it

may be important to include the information from correlations across outputs in order to build more accurate emulators. Furthermore, there will be significant computational time needed to fit thousands of emulators, as well as the time required to evaluate predictions from these thousands of emulators, for every parameter choice $\mathbf{x}$.

Gu et al. (2016) employ a similar approach to emulating high-dimensional output, by fitting an independent Gaussian process emulator to every individual spatial and temporal output. The difference here is that the terms in the mean function are constrained to be the same. Furthermore, the correlation length parameters for the Gaussian process are set at common values for every emulator. This considerably reduces the computation time required to fit the emulators, as correlation parameters only need to be estimated once, rather than for every emulator.

These simplifications allow predictions at a new parameter setting $\mathbf{x}$ to be calculated efficiently, giving further savings over the method in Lee et al. (2013). However, although Gu et al. (2016) claim that their method is more accurate than building thousands of completely separate emulators, the Lee et al. (2013) method should give emulators at least as accurate if fitted carefully, due to the increased flexibility allowed. The question then becomes what trade-off of accuracy, against the time required to fit the large number of emulators, is acceptable in a problem.

A method for emulating time series output is given by Liu and West (2009), where multivariate time-varying autoregressive models are used to model $f(\cdot)$. An autoregressive model is fitted for the response at each time t, for an input $\mathbf{x}$ and specified lag p:

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

where correlation over the input space $\mathcal{X}$ is included via $\epsilon_t(\mathbf{x})$ being modelled as a zero-mean Gaussian process with $\mathrm{Cov}(\epsilon_t(\mathbf{x}), \epsilon_t(\mathbf{x}')) = \sigma_t^2 C(\mathbf{x}, \mathbf{x}')$. This assumes that the correlation function is the same for all time t, although the variance can change. The autoregressive parameters $\phi_t = (\phi_{t,1}, \ldots, \phi_{t,p})^T$ are also allowed to vary over time, and are modelled as a random walk:

$$\phi_t = \phi_{t-1} + w_t, \quad w_t \sim \mathrm{N}(0, \sigma^2 W_t) \tag{2.13}$$

for a matrix W_t.

An extension of this method for emulating time series in climate models, or any computer model that exhibits 'structured chaos', is outlined in Williamson and Blaker (2014). Structured chaos means that a small change to the input parameters may result in short-term fluctuations in the output, although longer trends should still be similar, so a nugget term is required so that the emulator does not interpolate the ensemble exactly. Similarly to Liu and West (2009), the temporal relationships are modelled using a time-varying autoregressive model, with a Gaussian process for the parameter-dependent relationships. A dynamic regression is added to capture additional variation over the input parameters, as well as a temporal nugget term ν_t, giving the following emulator:

$$f_t(\mathbf{x}) = \sum_{j=1}^{p} \phi_{t,j} f_{t-j}(\mathbf{x}) + \sum_{i=1}^{k} \beta_{t,i} h_i(\mathbf{x}_t) + \epsilon_t(\mathbf{x}) + \nu_t$$

where $\epsilon_t(\mathbf{x})$ is now a zero-mean Gaussian process with $\text{Cov}(\epsilon_t(\mathbf{x}), \epsilon_t(\mathbf{x}')) = \tau\sigma_t^2 C(\mathbf{x}, \mathbf{x}')$, for parameter $\tau \in [0, 1]$, and $\nu_t \sim N(0, (1-\tau)\sigma^2)$. The autoregressive parameters are modelled as in (2.13).

When a computer model has several outputs of the same type (for example, time series output), it is possible to use the index of the outputs as an additional input, as done by Rougier (2008). In this paper, given a space $\mathcal{S}$ of l different outputs $s_1, \ldots, s_l$ of the computer model $f(\cdot)$, an emulator with a similar form to that in (2.1) is fitted, with the inclusion of a dependence on the index of the output in the linear functions and the residual term:

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

where the linear functions $h_j(\cdot, \cdot)$ should be the same for each of the l outputs for tractability. A normal inverse gamma prior is placed on the hyperparameters as in (2.12), with a common variance multiplier used across the outputs. The difficulty here comes in the specification of the correlation function $C(\cdot, \cdot)$ for the residual, as it is defined on $\mathcal{X} \times \mathcal{S}$, across the space of the inputs and outputs.

Rougier (2008) assumes a separable structure for this correlation function, i.e. the joint correlation function can be written as the product of a correlation function over the inputs

multiplied by a correlation function across the outputs:

$$C((\mathbf{x}, s), (\mathbf{x}', s')) = C_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')C_{\mathcal{S}}(s, s')$$

where $C_{\mathcal{X}}(\cdot, \cdot)$ and $C_{\mathcal{S}}(\cdot, \cdot)$ are the correlation functions over the input parameter space and the output space respectively. This assumption allows the use of Kronecker products, simplifying the required matrix inversions when predicting output using the emulator. Using a common variance multiplier across the outputs may not be a suitable assumption, because the variance may, for example, increase over time, as the model output may diverge for different input parameter settings as it is run for longer.

Conti et al. (2009) and Conti and O'Hagan (2010) remove the need for a constant variance multiplier across all of the l different outputs, and do not require a correlation function to be specified across both the inputs and outputs, instead representing the vector output of $f(\cdot)$ as an l-dimensional Gaussian process:

$$f(\cdot)|\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\phi} \sim \mathrm{N}_l(\mathbf{m}(\cdot), C(\cdot, \cdot)\boldsymbol{\Sigma})$$

where $\boldsymbol{\Sigma}$ is an $l \times l$ matrix representing the covariance between the l outputs at an input, $C(\cdot, \cdot)$ is the correlation over $\mathcal{X}$, dependent on parameters $\boldsymbol{\phi}$, and $\boldsymbol{\beta}$ is a $k \times l$ matrix of coefficients for the functions of input parameters.

Using the non-informative prior function for the coefficients $\boldsymbol{\beta}$ and the covariance matrix,

$$P(\boldsymbol{\beta}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-(l+1)/2}$$

conditioning on the ensemble

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

now a matrix with dimension $l \times n$ containing the output at $\mathbf{x}_i$ in the i^{th} column, proceeds similarly as in the univariate case with a non-informative prior, as in Haylock and O'Hagan (1996). The multivariate generalisation of equation (2.8), the output of the computer model given the ensemble, is

$$f(\cdot)|\boldsymbol{\Sigma}, \mathbf{F}, \boldsymbol{\phi} \sim \mathrm{N}_l(\mathbf{m}^{**}(\cdot), C^{**}(\cdot, \cdot)\boldsymbol{\Sigma})$$

where $\mathbf{m}^{**}(\cdot)$ and $C^{**}(\cdot,\cdot)$ are the multivariate generalisations of (2.9) and (2.10) respectively, so that:

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

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

where

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

is a $k \times l$ matrix of coefficients, $\mathbf{h}(\cdot) = (h_1(\cdot), \ldots, h_k(\cdot))^T$ contains the functions of the parameters from the mean function, $\mathbf{H}$ is the $n \times k$ design matrix with i, j^{th} entry $h_j(\mathbf{x}_i)$, $\mathbf{A}$ is the $n \times n$ correlation matrix with i, j^{th} entry $C(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{t}(\cdot)$ is a vector of length n with i^{th} entry $C(\cdot, \mathbf{x}_i)$.

As in the 1-dimensional case, the dependency on $\boldsymbol{\Sigma}$ can be removed by integrating out this parameter, giving the following t-distribution:

$$f(\cdot)|\mathbf{F}, \boldsymbol{\phi} \sim \mathcal{T}_{n-k}(\mathbf{m}^{**}(\cdot), C^{**}(\cdot,\cdot)\hat{\boldsymbol{\Sigma}})$$

where

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

Conti and O'Hagan (2010) give an example showing that this multi-output emulator (with $\boldsymbol{\phi}$ fixed at an estimate) gives superior predictions than the method using time as an input when applied to time series output from a computer model.

The methodology of Conti and O'Hagan (2010) has the advantage of tractability due to specifying the same spatial correlation function $C(\cdot,\cdot)$ for each of the l outputs. However, this may not be a reasonable assumption, especially if the outputs represent a variety of different quantities, as is assumed to be the case by Fricker et al. (2013). In this paper, non-separable correlation functions are considered, allowing spatial correlations to be different across outputs.

The Gaussian process for the l-dimensional residual $\boldsymbol{\epsilon}$ can be defined via process convolutions (Higdon, 2002):

$$\epsilon_i(\mathbf{x}) = \int_{\mathcal{X}} \kappa_i(\mathbf{u} - \mathbf{x})w(\mathbf{u})d\mathbf{u} \quad i = 1, \ldots, l \tag{2.14}$$

where $\kappa_i(\cdot)$ is a smoothing kernel chosen for output i, and $w(\mathbf{u})$ is a Gaussian white noise process. If this white noise process has mean zero and variance equal to one, then an individual element of the correlation matrix can be written as

$$C_{ij}(\mathbf{x}, \mathbf{x}') = \int_{\mathcal{X}} \kappa_i(\mathbf{u} - \mathbf{x})\kappa_j(\mathbf{u} - \mathbf{x}')d\mathbf{u}$$

This method doesn't allow for any direct specification of between-output correlations, as this is completely defined by the smoothing kernels for each individual output. This is overcome by instead allowing a multivariate white noise process (Fricker et al., 2013), so that $w(\mathbf{u})$ in (2.14) is $w_i(\mathbf{u})$, with

$$\text{Cov}(w_i(\mathbf{x}), w_j(\mathbf{x}')) = p_{ij}\delta(x - x')$$

where p_{ij} are the entries of some correlation matrix, used to control the between-output correlations. This generalisation to multivariate white noise leads to the following entries of the covariance matrix:

$$C_{ij}(\mathbf{x}, \mathbf{x}') = p_{ij} \int_{\mathcal{X}} \kappa_i(\mathbf{u} - \mathbf{x} + \mathbf{x}')\kappa_j(\mathbf{u})d\mathbf{u}$$

For large l, this method, and the previously described technique of Conti et al. (2009), may not be practical due to the difficulty in inverting the $l \times l$ correlation matrix, as will be required in calibration (Section 2.5). Therefore, methods that allow for dimension reduction are also considered.

2.4.1. Basis methods

The previous methods are suited for modelling time series output, but other methods may be required for spatial or spatio-temporal data, as instead of correlations between time points, there are correlations between locations, which may need to be treated differently. Furthermore, the problem of large l referred to previously needs to be taken into account.

The most common approach to emulating spatial output is by projecting the data onto a low-dimensional basis, using, for example, principal components (Higdon et al., 2008a), and then emulating the coefficients on this basis.

Principal component analysis (PCA) (Jolliffe, 2002) is used to extract the main modes or directions of variability within a data set. Variability is used in a climate setting to describe the total variance in a data set, and is used as such hereafter. In the setting of multivariate emulation, PCA is applied over the output fields, $\mathbf{F}$, of $f(\cdot)$, so that the patterns and directions of variability across the ensemble are explained. The majority of variability in an ensemble may be explainable by a small number of orthogonal directions, hence dimension reduction can be achieved without a large loss of information. The decomposition of the data using PCA is equivalent to the singular value decomposition (SVD) (Golub and Reinsch, 1970) and finding empirical orthogonal functions (EOFs). In the future, this basis will be referred to as the SVD basis.

To calculate this basis, the singular value decomposition for the ensemble $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$, the $l \times n$ matrix of multivariate computer model output, is written as

$$\mathbf{F}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \tag{2.15}$$

where $\mathbf{U}$ is an $n \times n$ matrix of the left singular vectors, $\boldsymbol{\Sigma}$ is an $n \times l$ matrix containing the singular values on the leading diagonal (with zeros elsewhere), ordered so that the largest is first, and $\mathbf{V}^T$ is an $l \times l$ matrix containing the right singular vectors (Golub and Reinsch, 1970). In the majority of applications in the computer model literature where dimension reduction is considered, $n << l$ (i.e. the ensemble size is smaller than, and often considerably so, the number of outputs). Therefore, only the first n rows of $\mathbf{V}^T$ correspond to the singular values in $\boldsymbol{\Sigma}$, and the basis required to represent the ensemble with n orthogonal directions is given by the first n columns of $\mathbf{V}$.

In Higdon et al. (2008a), the computer model gives output over a 20×26 grid, giving $l = 520$ values for each setting of the parameters. The spatial output is vectorised by 'stacking' the columns, i.e. the vector is filled by first taking the first column of the grid, then the second, and so on. From this point onwards, it will be assumed that output over a spatial grid is always converted into a vector in this manner. In order to reduce the dimensionality of the problem, first the ensemble mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_l)^T$ is calculated, where

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

i.e. μ_i is the ensemble mean of the i^{th} output. This is subtracted from each column of the

ensemble $\mathbf{F}$, giving the centred ensemble $\mathbf{F}_{\boldsymbol{\mu}}$, from which the principal component basis is calculated, as in (2.15). Let this basis be denoted by $\boldsymbol{\Gamma}$, where

$$\boldsymbol{\Gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_{n-1})$$

are the first $n-1$ columns of $\mathbf{V}$. Each individual basis vector $\boldsymbol{\gamma}_i$ has length l, the dimension of the output, and there are $n-1$ basis vectors, one less than the size of the ensemble $\mathbf{F}$ as the ensemble mean has been removed. These vectors are orthogonal by construction.

Given this basis, $f(\cdot)$ can be written as a linear combination of the basis vectors:

$$f(\mathbf{x}) - \boldsymbol{\mu} = \sum_{i=1}^{n-1} \boldsymbol{\gamma}_i c_i(\mathbf{x}) + \boldsymbol{\epsilon} \tag{2.16}$$

where $c_i(\mathbf{x})$ is the coefficient for basis vector $\boldsymbol{\gamma}_i$ and parameter setting $\mathbf{x}$, and $\boldsymbol{\epsilon}$ is a residual vector of length l. A residual is included as the basis is not of full rank, hence it may not be possible to accurately represent any general output $f(\cdot)$ with linear combinations of the vectors in the basis $\boldsymbol{\Gamma}$, because $n << l$. The residual is orthogonal to each $\boldsymbol{\gamma}_i$.

The previous equation can also be written in matrix form:

$$f(\mathbf{x}) - \boldsymbol{\mu} = \boldsymbol{\Gamma}\mathbf{c}(\mathbf{x}) + \boldsymbol{\epsilon}$$

where $\mathbf{c}(\mathbf{x})$ is the vector of coefficients for input parameter $\mathbf{x}$ given this basis, and has length $n-1$. From this, an expression for $\mathbf{c}(\mathbf{x})$ can be derived:

$$\begin{aligned} & f(\mathbf{x}) - \boldsymbol{\mu} = \boldsymbol{\Gamma}\mathbf{c}(\mathbf{x}) + \boldsymbol{\epsilon} \\ \implies & \boldsymbol{\Gamma}^T(f(\mathbf{x}) - \boldsymbol{\mu}) = \boldsymbol{\Gamma}^T\boldsymbol{\Gamma}\mathbf{c}(\mathbf{x}) + \boldsymbol{\Gamma}^T\boldsymbol{\epsilon} \\ \implies & \mathbf{c}(\mathbf{x}) = (\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}^T(f(\mathbf{x}) - \boldsymbol{\mu}) \end{aligned} \tag{2.17}$$

By performing this calculation for each member of the ensemble, each member of the ensemble is now associated with $n-1$ coefficients, rather than the original l outputs, and hence the dimensionality of the data has been reduced, perhaps significantly. This process is referred to as projecting the ensemble onto a basis, and (2.17) is the projection equation. The projected ensemble can be written

$$\mathbf{F}_{\mathbf{c}} = (\mathbf{c}(\mathbf{x}_1), \ldots, \mathbf{c}(\mathbf{x}_n))$$

Exploiting the fact that the basis vectors of $\mathbf{\Gamma}$ are orthogonal, Higdon et al. (2008a) fit univariate Gaussian process emulators for the coefficients $c_i(\cdot)$ for each basis vector separately:

$$c_i(\cdot)|\mathbf{F_c}, \boldsymbol{\phi} \sim \text{GP}(0, \lambda_i^{-1} C_i(\cdot, \cdot)) \tag{2.18}$$

This paper uses a zero-mean Gaussian process, with a power exponential correlation function $C(\cdot, \cdot)$ and precision λ_i, although any univariate emulation approach from Section 2.3 may be used in general. For example, Wilkinson (2010) fits Gaussian processes with a mean function as in (2.2). It is possible to fit emulators with different mean functions, correlation function forms and correlation function parameters for each set of coefficients, overcoming the issue of constant correlations across different outputs required by previously described methods.

Using this method for reducing the dimensionality of the model output, Higdon et al. (2008a) reduce it further by only considering the first three basis vectors, as these are all that are required to explain over 99% of the variability in the ensemble. The justification for this is that it is difficult to fit emulators with any predictive ability to the coefficients for later basis vectors, and that using only the first few still allows for the model output to be represented accurately. This truncated basis of the first q vectors can be written as

$$\mathbf{\Gamma}_q = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q)$$

From the set of univariate emulators on the first q basis coefficients (equation (2.18)), a model can be written for the vector output of $f(\cdot)$:

$$f(\cdot)|\mathbf{F}, \boldsymbol{\phi} \sim \text{N}(\mathbf{0}_l, \mathbf{\Gamma}_q \mathbf{\Sigma}_c \mathbf{\Gamma}_q^T + \lambda_\epsilon^{-1} \mathbf{I}_l)$$

where $\mathbf{0}_l$ is a zero vector of length l;

$$\mathbf{\Sigma}_c = \text{diag}(\lambda_1^{-1} C_1(\cdot, \cdot), \ldots, \lambda_q^{-1} C_q(\cdot, \cdot))$$

i.e. $\mathbf{\Sigma_c}$ is a $q \times q$ matrix with the variances from the individual coefficient emulators on the diagonal, with zeros elsewhere as it is assumed that the emulators for different coefficients are independent; λ_ϵ is the error precision, included as it is assumed that the error vector from (2.16) is Normal with this precision; and $\mathbf{I}_l$ is the $l \times l$ identity matrix.

To complete the model specification, priors are specified for all of the unknown parameters ($\lambda_1, \cdots, \lambda_q, \lambda_\epsilon$, and the Gaussian process correlation lengths).

Projecting spatial data onto the SVD (principal component) basis prior to emulating the coefficients has been applied in many papers, including Wilkinson (2010), Holden and Edwards (2010), Sexton et al. (2011), Holden et al. (2013), Chang et al. (2014a,b), Bounceur et al. (2015), Holden et al. (2015), Oyebamiji et al. (2015), Chang et al. (2016), with slight differences or extensions to the methodology used by Higdon et al. (2008a).

Wilkinson (2010) extends the previous methodology with an addition to the variance of the emulator. As only the first q basis vectors are used for emulation, some ensemble variability has been ignored. In order to account for this, multiples of the discarded basis vectors are added to the variance term. These are modelled using a zero-mean Normal distribution. Therefore, the reconstruction $\mathbf{r}(\cdot)$ of a particular field can be written as

$$\mathbf{r}(\cdot) = \mathbf{\Gamma}_q \mathbf{c}(\cdot) + \mathbf{\Gamma}_{-q} \mathbf{\Psi}$$

so that $\mathbf{r}(\cdot)$ is a vector of length l. The matrix $\mathbf{\Gamma}_{-q}$ is $\mathbf{\Gamma}$ with the first q columns removed so that

$$\mathbf{\Gamma} = (\mathbf{\Gamma}_q, \mathbf{\Gamma}_{-q})$$

and $\mathbf{\Psi}$ is a vector of multiples with length $(n-1-q)$, where the i^{th} entry of the vector is sampled from the distribution

$$\Psi_i \sim \mathrm{N}(0, \Sigma_{q+i,q+i})$$

$\Sigma_{q+i,q+i}$ denotes the eigenvalue associated with the $(q+i)^{th}$ basis vector of $\mathbf{\Gamma}$ (from (2.15)).

To emulate the coefficients for the first q basis vectors, Wilkinson (2010) then follows the method of Haylock and O'Hagan (1996). Univariate Gaussian process emulators are fitted for each of the first q basis vectors:

$$c_i(\cdot)|\boldsymbol{\beta}_i, \sigma_i^2, \boldsymbol{\phi}_i \sim \mathrm{N}(\mathbf{h}_i(\cdot)^T \boldsymbol{\beta}_i, \sigma_i^2 C_i(\cdot,\cdot)) \quad i = 1, \ldots, q$$

with all quantities defined as in (2.5), with the addition of i subscripts as these quantities can all vary for each coefficient emulator. Conditioning on the (projected) ensemble proceeds as in the univariate case described earlier, leading to separate posterior means and

covariances (as in (2.9) and (2.10)) for each of the q emulators. Let $m_i^{**}(\cdot)$ and $C_i^{**}(\cdot,\cdot)$ denote the posterior mean and covariance for the emulator of the coefficients on basis vector i, and

$$\boldsymbol{\mu}_{\mathbf{c}}(\mathbf{x}) = (m_1^{**}(\mathbf{x}), \ldots, m_q^{**}(\mathbf{x}))^T$$
$$\boldsymbol{\Sigma}_{\mathbf{c}}(\mathbf{x},\mathbf{x}) = \text{diag}(\sigma_1^2 C_1^{**}(\mathbf{x},\mathbf{x}), \ldots, \sigma_q^2 C_q^{**}(\mathbf{x},\mathbf{x}))$$

Using this notation, the posterior distribution for the reconstruction of $f(\cdot)$ can be written as

$$f(\mathbf{x})|\mathbf{F}_{\mathbf{c}}, \boldsymbol{\sigma}^2, \boldsymbol{\phi} \sim \text{N}(\boldsymbol{\Gamma}_q \boldsymbol{\mu}_{\mathbf{c}}(\mathbf{x}), \boldsymbol{\Gamma}_q \boldsymbol{\Sigma}_{\mathbf{c}}(\mathbf{x},\mathbf{x}) \boldsymbol{\Gamma}_q^T + \boldsymbol{\Gamma}_{-q} \boldsymbol{\Sigma}^* \boldsymbol{\Gamma}_{-q}^T) \tag{2.19}$$

where

$$\boldsymbol{\Sigma}^* = \text{diag}(\Sigma_{q+1,q+1}, \ldots, \Sigma_{n-1,n-1})$$

and $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_q^2)$.

Holden and Edwards (2010) and Holden et al. (2013) apply the technique of Higdon et al. (2008a) to spatio-temporal climate model output, emulating the decadal averages for temperature anomalies and precipitation in the former, and surface air temperature variability in the latter. The spatio-temporal model output is converted into a single vector for each parameter choice by first turning the spatial grid for each time point into a vector as described previously, before the vectors for each time point are concatenated into a single vector for $f(\mathbf{x})$. Linear regression is used for each of the coefficient emulators in both of these papers, with the first 5 and 10 SVD basis vectors used respectively.

Similar methods have been used by Sexton et al. (2011), where the coefficients for the eigenvectors are emulated univariately using linear regression, before a covariance term is added to account for dependencies between the emulators. Chang et al. (2014a) calculate the average across rows of spatial output to give a vector response at each parameter choice, and then build independent Gaussian process emulators for the first 10 basis vectors.

An alternative basis method is given by Bayarri et al. (2007). This paper considers emulating time series output using basis functions known as the wavelet decomposition (Vidakovic, 2009). The computer model output is represented as

$$f(\mathbf{x}, t) = \sum_i w_i(\mathbf{x}) \psi_i(t)$$

where $\psi_i(t)$ are the wavelet basis functions, defined over the range of the model time series, and $w_i(\mathbf{x})$ are coefficients, dependent on the parameter setting, and are assumed to be independent across i. The basis elements are chosen in order to be able to accurately reconstruct the original time series output, with default R choices used here due to the nature of the output. As in the SVD projection case, the coefficients $w_i(\mathbf{x})$ are then emulated using Gaussian processes.

Another representation of time series output using basis functions has been used by Williamson et al. (2012). Here, instead of using the wavelet decomposition, the model output is represented using a spline basis (Fahrmeir and Tutz, 2001). Splines are polynomials defined over time (in this case), used to smoothly represent the output. Given these basis functions, coefficients that minimise a chosen criteria are fitted for the ensemble runs, and these coefficients are then emulated. Therefore, the representation of the computer model output is

$$f(\mathbf{x}, t) = \sum_i c_i(\mathbf{x}) B_i(t) + \eta(\mathbf{x}, t)$$

for coefficients $c_i(\mathbf{x})$ dependent on $\mathbf{x}$, and spline basis function $B_i(t)$ that varies with time, where $\eta(\mathbf{x}, t)$ is the difference between the smooth spline representation and the true output. A benefit of this basis is that the splines can be defined so that their coefficients have physical meaning.

For spatial climate model output, Furrer et al. (2007) use a basis constructed using spherical harmonics (Jones, 1963) to represent the main variability in the output, with the coefficients on this basis modelled. However, in this model there is an assumption that the ensemble coefficients are centred around the 'true' value for these coefficients i.e. it assumes that there are not systematic biases away from the true climate. This may not always be a valid assumption, as there will often be common biases observed for any choice of input parameter.

2.5. Bayesian calibration

Given emulators for the output of a computer model $f(\cdot)$, inferences can be made about the model. Emulators give efficient predictions, with associated uncertainty, for any chosen $\mathbf{x}$, and hence can be used as a tool for exploring the output space of $f(\cdot)$ as the input

parameters vary. Of particular interest is finding settings for the parameters that lead to model output that is consistent with observed real-world values for the system represented by $f(\cdot)$ (Kennedy and O'Hagan, 2001a, Annan et al., 2005, Rougier, 2007, Bellprat et al., 2012). These parameter settings are generally not known, due to the complexity of computer models. Only a small sample from the often large parameter space $\mathcal{X}$ is usually available, so that a vanishingly small percentage of this space has been explored, and it is generally unlikely that a perfect representation of the real-world is contained in this small sample.

It is important to find parameter settings that give output consistent with the underlying system represented by the model for a number of reasons. If the computer model output is similar to the real-world, then it should be more suitable for tasks such as prediction, if the computer model is run into the future. Using climate model output to assess long-term climate change scenarios (for example, under different carbon dioxide forcings) is a critical feature of, and provides key evidence for, Intergovernmental Panel on Climate Change reports (Bindoff et al., 2013). These future predictions (or forecasts) ought to be more accurate if they start from the present-day state of the climate. However, rigorous model validation, for example ensuring we have not overfitted to the available data, is important: it is possible for the climate model to make poor predictions, even if it matches current observations.

Bayesian calibration is a statistical method that can be used to find a probability distribution over the unknown input parameters of a computer model, conditional on an ensemble of runs and real-world observations (Kennedy and O'Hagan, 2001a, Higdon et al., 2004, Rougier, 2007). Generally, the computer model takes a long time to run, and hence emulators are used in place of running the true model. Calibration can be performed for a single output or for multiple outputs jointly (see Section 2.6 for methods for calibrating a large number of outputs).

In Bayesian calibration, a statistical model linking the computer model, $f(\cdot)$, to the real-world system that it represents is used, as the computer model may not be able to exactly reproduce this system for any $\mathbf{x}$ (the model discrepancy). Observations of the real-world system are also required. These are likely to be subject to some error (for example, measurement error from instruments), termed the observation error. Given a statistical model that links the computer model (equivalently, its emulators), the discrepancy, and

the observation error, calibration gives a posterior distribution over $\mathcal{X}$, with more posterior density at points more likely to give output consistent with the observations.

Calibration assumes that a 'best input' setting, $\mathbf{x}^*$, of the parameters exists (Kennedy and O'Hagan, 2001a). That is, if the computer model is run at $\mathbf{x}^*$, then the output will be the true value for the system, up to the statistical model linking this to model discrepancy and observation error. This assumption may not always be valid, particularly if the discrepancy is not correctly specified. For example, if the discrepancy between the model and reality is assumed to be zero, whereas in fact there is a non-zero discrepancy, then there will be no parameter settings that satisfy the best input requirement. However, calibration always returns a probability distribution over the inputs, so by definition must have posterior density for some $\mathbf{x}^*$. In the scenario of the best input not existing, the posterior distribution will not necessarily highlight suitable parameter settings.

Kennedy and O'Hagan (2001a) perform Bayesian calibration as follows. The input parameters are first divided into two groups: the calibration parameters, $\mathbf{x}_{cal}$, and the control or decision inputs, $\mathbf{x}_{con}$, so that $\mathbf{x} = (\mathbf{x}_{cal}, \mathbf{x}_{con})$. The calibration parameters have unknown values, and are the parameters for which the best input $\mathbf{x}^*$ is desired. The control parameters have known values, and are fixed at these throughout the calibration process. These parameters relate to systems where it is possible to obtain multiple observations of the real system for different values of $\mathbf{x}_{con}$. The multiple observations are written $\mathbf{z} = (z_1, \ldots, z_N)^T$, where z_i is the observation for known control input $\mathbf{x}_{con}^i$.

A statistical model is defined that describes the relationship between $f(\cdot)$ and the underlying system that it represents, $y(\cdot)$:

$$y(\mathbf{x}_{con}) = \rho f(\mathbf{x}^*, \mathbf{x}_{con}) \oplus \eta(\mathbf{x}_{con}) \tag{2.20}$$

where $\mathbf{x}^*$ is the best input for the calibration parameters, ρ is an unknown regression parameter, and $\eta(\mathbf{x}_{con})$ is the discrepancy between the real world, $y(\cdot)$, and the model, $f(\cdot)$, at control parameters $\mathbf{x}_{con}$. $\oplus$ indicates the addition of independent terms, i.e. the discrepancy is independent of the best parameter setting, and of the model $f(\mathbf{x})$.

Since it is not possible to observe the real-world system exactly, a statistical model linking

the observations with the true value of the system $y(\cdot)$ is required:

$$z_i = z(\mathbf{x}^i_{con}) = y(\mathbf{x}^i_{con}) \oplus e_i \tag{2.21}$$

where e_i is the error when observing the system at $\mathbf{x}^i_{con}$. Combining (2.20) and (2.21) gives the following calibration model:

$$z_i = \rho f(\mathbf{x}^*, \mathbf{x}^i_{con}) \oplus \eta(\mathbf{x}^i_{con}) \oplus e_i \tag{2.22}$$

Modelling assumptions are required for the unknown quantities in this equation. The observation error for each i is assumed to be independent of the value of the computer model and discrepancy, and to have a Normal distribution with a common variance:

$$e_i \sim \mathrm{N}(0, \sigma^2_{\mathbf{e}})$$

Gaussian processes are chosen as the prior models for both $f(\cdot)$ and $\eta(\cdot)$:

$$f(\cdot,\cdot) \sim \mathrm{N}(m_1(\cdot,\cdot), \sigma^2_1 C_1[(\cdot,\cdot),(\cdot,\cdot)]) \tag{2.23}$$

$$\eta(\cdot) \sim \mathrm{N}(m_2(\cdot), \sigma^2_2 C_2(\cdot,\cdot)) \tag{2.24}$$

where the mean functions are modelled using unknown coefficients $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ of functions of parameters $\mathbf{h}_1(\cdot)$ and $\mathbf{h}_2(\cdot)$ as in Section 2.3.1, and the correlation functions are defined by hyperparameters $\boldsymbol{\phi}$. Prior distributions are also selected for these parameters:

$$\pi(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) \propto 1$$
$$\pi(\mathbf{x}^*, \boldsymbol{\beta}, \boldsymbol{\psi}) \propto \pi(\mathbf{x}^*)\pi(\boldsymbol{\psi})$$

with prior information about $\mathbf{x}^*$ assumed to be independent of the other hyperparameters, $\boldsymbol{\psi} = (\rho, \sigma^2_{\mathbf{e}}, \boldsymbol{\phi})$.

Similarly as for emulation in Section 2.3, the prior models are updated by conditioning on the available data to find a posterior distribution (Kennedy and O'Hagan, 2001b). In this case, the data is the vector of observations $\mathbf{z}$ and an ensemble $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$ of n runs of $f(\cdot)$, at parameter settings $\mathbf{D}_1 = \{\mathbf{x}_i = (\mathbf{x}^i_{cal}, \mathbf{x}^i_{con})\}^n_{i=1}$. These are combined into a single data vector $\mathbf{d} = (\mathbf{F}^T, \mathbf{z}^T)^T$ of length $n + N$ (when the computer model output is

a scalar). Conditioning on the data and using the above priors, the posterior distribution for the unknown parameters can be written as:

$$\begin{aligned}\pi(\mathbf{x}^*, \boldsymbol{\beta}, \boldsymbol{\psi}|\mathbf{d}) &\propto \pi(\mathbf{x}^*, \boldsymbol{\beta}, \boldsymbol{\psi})\pi(\mathbf{d}|\mathbf{x}^*, \boldsymbol{\beta}, \boldsymbol{\psi}) \\ &\propto \pi(\mathbf{x}^*)\pi(\boldsymbol{\psi})\mathrm{N}(\mathbf{m}_d(\mathbf{x}^*), \mathbf{V}_d(\mathbf{x}^*)) \\ &\propto \pi(\mathbf{x}^*)\pi(\boldsymbol{\psi})|\mathbf{V}_d(\mathbf{x}^*)|^{-\frac{1}{2}} \exp[-\frac{1}{2}(\mathbf{d} - \mathbf{m}_d(\mathbf{x}^*))^T\mathbf{V}_d(\mathbf{x}^*)^{-1}(\mathbf{d} - \mathbf{m}_d(\mathbf{x}^*))]\end{aligned}$$

using that $\mathbf{d}$ has a Normal distribution given $\mathbf{x}^*$, $\boldsymbol{\beta}$ and $\mathbf{V}_d(\mathbf{x}^*)$, with mean $\mathbf{m}_d(\mathbf{x}^*)$ and variance $\mathbf{V}_d(\mathbf{x}^*)$, where

$$\mathbf{m}_d(\mathbf{x}^*) = \mathbf{H}(\mathbf{x}^*)\boldsymbol{\beta}, \quad \mathbf{H}(\mathbf{x}^*) = \begin{pmatrix} \mathbf{H}_1(\mathbf{D}_1) & 0 \\ \rho\mathbf{H}_1(\mathbf{D}_2(\mathbf{x}^*)) & \mathbf{H}_2(\mathbf{D}_2) \end{pmatrix}$$

$\mathbf{H}_1(\cdot)$ is the design matrix for the Gaussian process model for $f(\cdot)$, $\mathbf{H}_2(\cdot)$ is the design matrix for $\eta(\cdot)$, and $\mathbf{D}_2(\mathbf{x}^*)$ is the design concatenating the control parameters for each observation i with $\mathbf{x}^*$. Additionally:

$$\mathbf{V}_d(\mathbf{x}^*) = \begin{pmatrix} \mathbf{V}_1(\mathbf{D}_1) & \rho\tilde{C}_1(\mathbf{D}_1, \mathbf{D}_2(\mathbf{x}^*))^T \\ \rho\tilde{C}_1(\mathbf{D}_1, \mathbf{D}_2(\mathbf{x}^*)) & \sigma^2_{\mathbf{e}}\mathbf{I}_N + \rho^2\mathbf{V}_1(\mathbf{D}_2(\mathbf{x}^*)) + \mathbf{V}_2(\mathbf{D}_2) \end{pmatrix}$$

where the i, j^{th} entry of the various required matrices are defined as:

$$\begin{aligned}\mathbf{V}_1(\mathbf{D}_1)_{ij} &= C_1(\mathbf{x}_i, \mathbf{x}_j) \\ \mathbf{V}_1(\mathbf{D}_2(\mathbf{x}^*))_{ij} &= C_1((\mathbf{x}^*, \mathbf{x}^i_{con}), (\mathbf{x}^*, \mathbf{x}^j_{con})) \\ \mathbf{V}_2(\mathbf{D}_2) &= C_2(\mathbf{x}^i_{con}, \mathbf{x}^j_{con}) \\ \tilde{C}_1(\mathbf{D}_1, \mathbf{D}_2(\mathbf{x}^*)) &= C_1(\mathbf{x}_i, (\mathbf{x}^*, \mathbf{x}^j_{con}))\end{aligned}$$

By integrating out $\boldsymbol{\beta}$, and by estimating $\boldsymbol{\psi}$, a posterior distribution for $\mathbf{x}^*$, conditioned on the data and hyperparameters, can be written as

$$\pi(\mathbf{x}^*|\boldsymbol{\psi}, \mathbf{d}) \propto \pi(\mathbf{x}^*)|\mathbf{V}_d(\mathbf{x}^*)|^{-\frac{1}{2}}|\mathbf{W}(\mathbf{x}^*)|^{\frac{1}{2}} \exp[-\frac{1}{2}(\mathbf{d} - \hat{\mathbf{m}}_d(\mathbf{x}^*))^T\mathbf{V}_d(\mathbf{x}^*)^{-1}(\mathbf{d} - \hat{\mathbf{m}}_d(\mathbf{x}^*))] \tag{2.25}$$

where

$$\begin{aligned}\mathbf{W}(\mathbf{x}^*) &= (\mathbf{H}(\mathbf{x}^*)^T\mathbf{V}_d(\mathbf{x}^*)^{-1}\mathbf{H}(\mathbf{x}^*))^{-1} \\ \hat{\mathbf{m}}_d(\mathbf{x}^*) &= \mathbf{H}(\mathbf{x}^*)\hat{\boldsymbol{\beta}}\end{aligned}$$

To determine this posterior distribution, MCMC sampling is required. Prior to this, the correlation length parameters for the Gaussian process for $f(\cdot)$ are estimated, using the ensemble $\mathbf{F}$. The remaining hyperparameters (namely, those for the Gaussian process of the discrepancy term, the regression parameter ρ, and the variance for the observation error) are found via the MCMC.

The posterior distribution for the observations at control parameters $\mathbf{x}_{con}$ given $\mathbf{x}^*$ is also calculated, and is a Gaussian process with posterior mean and variance analogous to a multivariate extension of the posteriors in (2.9) and (2.10):

$$\begin{aligned}
\mathrm{E}[z(\mathbf{x}_{con})|\mathbf{x}^*, \boldsymbol{\psi}, \mathbf{d}] &= \mathbf{h}(\mathbf{x}^*, \mathbf{x}_{con})^T\hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x}^*, \mathbf{x}_{con})^T\mathbf{V}_d(\mathbf{x}^*)^{-1}(\mathbf{d} - \hat{\mathbf{m}}_d(\mathbf{x}^*)) \\
\mathrm{Cov}[y(\mathbf{x}_{con}), y(\mathbf{x}'_{con})|\mathbf{x}^*, \boldsymbol{\psi}, \mathbf{d}] &= \rho^2 C_1((\mathbf{x}^*, \mathbf{x}_{con}), (\mathbf{x}^*, \mathbf{x}'_{con})) + C_2(\mathbf{x}_{con}, \mathbf{x}'_{con}) \\
&\quad - \mathbf{t}(\mathbf{x}^*, \mathbf{x}_{con})^T\mathbf{V}_d(\mathbf{x}^*)^{-1}\mathbf{t}(\mathbf{x}^*, \mathbf{x}'_{con}) \\
&\quad + (\mathbf{h}(\mathbf{x}^*, \mathbf{x}_{con}) - \mathbf{H}(\mathbf{x}^*)^T\mathbf{V}_d(\mathbf{x}^*)^{-1}\mathbf{t}(\mathbf{x}^*, \mathbf{x}_{con}))^T\mathbf{W}(\mathbf{x}^*) \\
&\quad \times (\mathbf{h}(\mathbf{x}^*, \mathbf{x}'_{con}) - \mathbf{H}(\mathbf{x}^*)^T\mathbf{V}_d(\mathbf{x}^*)^{-1}\mathbf{t}(\mathbf{x}^*, \mathbf{x}'_{con}))
\end{aligned} \tag{2.26}$$

for

$$\begin{aligned}
\mathbf{h}(\mathbf{x}^*, \mathbf{x}_{con}) &= (\rho\mathbf{h}_1(\mathbf{x}^*, \mathbf{x}_{con}), \mathbf{h}_2(\mathbf{x}_{con}))^T \\
\mathbf{t}(\mathbf{x}^*, \mathbf{x}_{con}) &= (\rho\mathbf{V}_1((\mathbf{x}^*, \mathbf{x}_{con}), \mathbf{D}_1), \rho^2\mathbf{V}_1((\mathbf{x}^*, \mathbf{x}_{con}), \mathbf{D}_2(\mathbf{x}^*)) + \mathbf{V}_2(\mathbf{x}_{con}, \mathbf{D}_2))^T
\end{aligned}$$

This distribution can be used to make inferences about the value of the system.

This framework is a flexible way to make inferences about $\mathbf{x}^*$ for a computer model given observations, as all of the priors may be changed depending on the problem. As long as the likelihood for $\mathbf{d}$ can be written analytically, the posterior can be derived, and then sampled from numerically. The case with univariate output has been outlined above. This can be adapted for models with multivariate output so that observations of multiple different outputs can be calibrated to at the same time. The l multiple outputs can be combined into the data vector $\mathbf{d}$, so that this now has $(n+N)l$ entries, with the corresponding means and variances in the posterior distributions updated so that the emulator predictions relate to the correct outputs.

The calibration method from this paper has been applied to computer models for a wide variety of systems, including in the study of the effects of radiation (McFarland et al.,

2007), sending shock waves through materials (Williams et al., 2006), the energy efficiency of buildings (Heo et al., 2011), to a general circulation model (Guillas et al., 2009), and to a flood inundation model (Hall et al., 2011). In the latter four of these, the simplification of setting $\rho = 1$ is applied.

The methodology from Kennedy and O'Hagan (2001a) is adjusted slightly for a specific example by Higdon et al. (2004). Here, the model linking the real system with the computer model has $\rho = 1$, as the simpler additive model is judged to be suitable for their example. This reduces the number of uncertain parameters in the model, and the amount of prior distributions that need to be specified. This also reduces the problem of identifiability: the discrepancy parameters, observation error variance and ρ are all determined at the same time using MCMC, and the separate contributions from each of these can be difficult to identify.

Furthermore, the data vector $\mathbf{d}$ is transformed to have mean zero and variance one, so that the parametrisation in the prior for $f(\cdot)$ can be simplified, with the prior mean set as either zero or a constant. The discrepancy $\eta(\cdot)$ is treated similarly, with the prior mean set at zero. The prior correlation function for both $f(\cdot)$ and $\eta(\cdot)$ is chosen to have a power exponential form, with priors also set for the correlation lengths.

Rougier (2007) discusses calibration in the context of climate models. In this setting, there is only one available set of historical observations of the climate, hence there is only one possible setting of $\mathbf{x}_{con}$. Therefore, the dependence on control parameters, and the existence of multiple observations z_i, can be removed from (2.22), to give the following calibration model:

$$\mathbf{z} = f(\mathbf{x}^*) \oplus \boldsymbol{\eta} \oplus \mathbf{e} \tag{2.27}$$

The observations $\mathbf{z}$ may still be a vector, for example of different aspects of the climate at different spatial and temporal points, but it no longer contains observations of the same system at different control parameters. The following distributions are assumed:

$$\boldsymbol{\eta} \sim \mathrm{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\eta}}) \tag{2.28}$$

$$\mathbf{e} \sim \mathrm{N}(0, \boldsymbol{\Sigma}_{\mathbf{e}}) \tag{2.29}$$

These variances are a single value in the 1-dimensional case, and a covariance matrix

across the different outputs in the higher-dimensional case. Rougier (2007) does not use emulators, so that $f(\cdot)$ is the climate model output here.

In Sexton et al. (2011), calibration is performed simultaneously for multiple independent responses, given observational data. The observations are projected onto six eigenvectors (Section 2.4.1), and the coefficients for $\mathbf{z}$ on these vectors are calibrated to. This uses the multivariate version of the Kennedy and O'Hagan (2001a) method, where there is one observation of the real world (as in Rougier (2007)), but six different coefficients (derived from high-dimensional observation vector $\mathbf{z}$) to be calibrated to. Therefore, the data vector $\mathbf{d}$ contains the projection of the ensemble onto these six basis vectors, followed by the observation coefficient vector.

2.5.1. Forecasting

Forecasting can be performed following calibration, as in Rougier (2007). The true system $\mathbf{y}$ is divided into past climate $\mathbf{y}_h$ and future climate $\mathbf{y}_f$, the quantity that forecasting is desired for, and therefore the model output can also be divided into past (simulated) climate $f_h(\cdot)$ and future (simulated) climate $f_f(\cdot)$. Using the model for calibration in (2.27), a probability distribution over possible future climates $\mathbf{y}_f$ is calculated via

$$\pi(\mathbf{y}_f|\mathbf{z}) = \int \pi(\mathbf{y}_f|\mathbf{x}^*, \mathbf{z})\pi(\mathbf{x}^*|\mathbf{z})d\mathbf{x}^*$$

where $\pi(\mathbf{x}^*|\mathbf{z})$ is the posterior of $\mathbf{x}^*$ given the data, as in (2.25) for the emulation case, and, in Rougier (2007):

$$\pi(\mathbf{x}^*|\mathbf{z}) = \pi(\mathbf{z})^{-1}\pi(\mathbf{x}^*)|\Sigma_\mathbf{e} + \Sigma_{\boldsymbol{\eta}}|^{-\frac{1}{2}} \exp[-\frac{1}{2}(\mathbf{z} - f(\mathbf{x}^*))^T(\Sigma_\mathbf{e} + \Sigma_{\boldsymbol{\eta}})^{-1}(\mathbf{z} - f(\mathbf{x}^*))]$$

and $\mathbf{y}_f$ has a prior distribution with some mean $\mathbf{m}_f$ and variance $\boldsymbol{\Sigma}_f$. If the computer model can be run exactly, as in Rougier (2007), then the posterior $\pi(\mathbf{y}_f|\mathbf{x}^*, \mathbf{z})$ has updated mean and variance

$$\mathbf{m}_{f|\mathbf{z}}(\cdot) = f_f(\cdot) + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{fh}(\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{hh} + \boldsymbol{\Sigma}_\mathbf{e})^{-1}(\mathbf{z} - f_h(\cdot))$$
$$\boldsymbol{\Sigma}_{f|\mathbf{z}} = \boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{ff} - \boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{fh}(\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{hh} + \boldsymbol{\Sigma}_\mathbf{e})^{-1}\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{hf}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{fh}$ refers to selecting the rows of the discrepancy variance relating to the future climate, and the columns relating to the historical climate, $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{hh}$ selects the rows and columns relating to the historical climate, and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{ff}$ selects the rows and columns relating to the future climate.

More commonly, an emulator will be required to represent the computer model, and the posterior $\pi(\mathbf{y}_f|\mathbf{x}^*, \mathbf{z})$ instead has mean and covariance as in (2.26).

2.6. Calibration in higher dimensions

The calibration methodology of Kennedy and O'Hagan (2001a) allows for a number of outputs to be calibrated simultaneously. However, as the number of outputs increases, it may become too computationally intensive to perform all necessary steps, such as the inversion of the variance matrix $\mathbf{V}_d(\mathbf{x}^*)$, which has dimension $((n+N)l) \times ((n+N)l)$ if there are l different outputs. Instead, multivariate calibration approaches have been developed, extending the univariate methodology.

Bhat et al. (2010) considers the problem of calibrating a computer model that has multiple spatial fields as the output, using several of these fields simultaneously. If there are k different output fields being used for calibration, each with observations $\mathbf{z}_i$ and model output given over a spatial field of size l, then the combined observational vector is written as $\mathbf{z} = (\mathbf{z}_1^T, \ldots, \mathbf{z}_k^T)^T$. The computer model output for each field is also combined into a single vector: if each of the k output fields has computer model output

$$\mathbf{F}_i = (f_i(\mathbf{x}_1)^T, \ldots, f_i(\mathbf{x}_n)^T)^T$$

then the full model output vector can be written

$$\mathbf{F}_s = (\mathbf{F}_{1,1}, \mathbf{F}_{2,1}, \ldots, \mathbf{F}_{k,1}, \ldots, \mathbf{F}_{1,nl}, \ldots, \mathbf{F}_{k,nl})^T$$

where $\mathbf{F}_{i,j}$ denotes the j^{th} entry of $\mathbf{F}_i$, and $\mathbf{F}_s$ is a vector of length knl.

The extension from the previous calibration methodology is that this structure allows for spatial dependence in the output space, and correlations between the different output fields, to be accounted for via a statistical model. A Gaussian process emulator is fitted

to the data:

$$f(\cdot)|\boldsymbol{\beta}, \mathbf{x}, \boldsymbol{\phi} \sim \mathrm{N}(\mathbf{m}_s(\mathbf{x}), \boldsymbol{\Sigma}_s)$$

where the mean function is a combination of the mean vectors for each output field individually:

$$\mathbf{m}_s(\mathbf{x}) = (\mathbf{m}_1(\mathbf{x})^T, \ldots, \mathbf{m}_k(\mathbf{x})^T)^T$$

and the covariance function is written as a combination of a term for the nugget $\boldsymbol{\nu}$, a term for the correlation across the spatial grid and parameter space $C(\cdot, \cdot)$, and a term for the relationships between the k output fields $T(\kappa, \rho)$:

$$\boldsymbol{\Sigma}_s = \boldsymbol{\nu} + C(\cdot, \cdot) \otimes T(\kappa, \rho)$$

where the correlation function C is defined across $\mathcal{X} \times \mathcal{S}$ and is separable.

The same model for linking the model output with the observations as in (2.27) is then used:

$$\mathbf{z} = f(\mathbf{x}^*) \oplus \boldsymbol{\eta} \oplus \mathbf{e}$$

Here, $f(\cdot)$ is the simulator output for each of the spatial fields simultaneously, i.e. all observations are combined into a single vector. The observation error $\mathbf{e}$ is given a Normal distribution:

$$\mathbf{e} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{e})$$

where $\boldsymbol{\Sigma}_\mathbf{e}$ is a diagonal matrix with k parameters ψ_i to be estimated, referring to the observation error variance for field i. This assumes that the observation error variance across an individual output field is constant. The discrepancy is modelled separately for each of the output fields, with an independent zero mean Gaussian process fitted for each:

$$\boldsymbol{\eta}_i \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\eta}_i})$$

where again there are hyperparameters to be estimated.

Given this statistical model for the output (equation (2.27)), and the emulators fitted to the ensemble, the posterior distribution for $\mathbf{x}^*$ is found similarly to the earlier cases via MCMC. This distribution is dependent on the observation error parameters, the parameters for the discrepancy Gaussian process, and the emulator correlation length parameters, if these are

not estimated and fixed via maximum likelihood.

Bhat et al. (2012) adds an extra step at the emulation stage. When there are two different output fields, with output $\mathbf{F}_1$ and $\mathbf{F}_2$, the modelling is done hierarchically instead of jointly. A Gaussian process model is fitted for $\mathbf{F}_2$, then for $\mathbf{F}_1|\mathbf{F}_2$. This is done instead of the previous approach to allow more inter-dependence between output fields, and to allow different spatial correlation patterns for fields, a more realistic assumption to make. Hierarchical emulation has been performed in other contexts to link computer models rather than output fields (Oughton and Craig, 2016).

However, using this technique may be problematic when the dimension of the spatial field or number of ensemble members is large. In either of these situations, the ability to emulate the model output as in Bhat et al. (2010) decreases due to the inversion of the variance matrix $\mathbf{\Sigma}_s$, with dimension $knl \times knl$, required every time the emulator is run at a new parameter setting. In this paper, the illustrative example has an ensemble with $n = 10$ members, giving output at $l = 13$ spatial locations, for $k = 3$ different outputs, so that this matrix is 390×390. However, situations where the number of spatial locations is in the hundreds or thousands are commonplace, and alternative methods are required for calibrating in this context, using some of the multivariate emulation techniques discussed in Section 2.4.1.

Wilkinson (2010) fits emulators to multivariate output using the basis approach outlined in Section 2.4.1, leading to the posterior for the output on the original field, given the ensemble $\mathbf{F}$ and other parameters, as in (2.19). This distribution is then combined with the prior on $\mathbf{x}^*$ to give the calibration distribution

$$\pi(\mathbf{x}^*|\mathbf{z}, \mathbf{F}) \propto \pi(f(\mathbf{x}^*)|\mathbf{F}, \mathbf{x}^*)\pi(\mathbf{x}^*)$$

which is sampled from using MCMC to give posterior probabilities for $\mathbf{x}^*$.

In Chang et al. (2014b), a methodology for emulating and calibrating larger spatial fields is discussed. Their example involves an ensemble of 250 runs of $f(\cdot)$, giving output at 61,051 locations, hence the previously discussed calibration methods of Kennedy and O'Hagan (2001a) and Bhat et al. (2010) are not suitable. Emulation of this data set is carried out by projecting the output onto a principal component basis as in equation (2.16), and

building independent Gaussian process emulators for each of the first $q < n$ basis vectors (equation (2.18), with the addition of a nugget term). The observations $\mathbf{z}$ are modelled as

$$\mathbf{z} = \boldsymbol{\Gamma}_q \mathbf{c}(\mathbf{x}^*) \oplus \boldsymbol{\eta} \oplus \mathbf{e}$$

with $\mathbf{e} \sim \mathrm{N}(0, \sigma^2 \boldsymbol{\mathcal{I}}_l)$, and the discrepancy $\boldsymbol{\eta}$ modelled using a kernel convolution representation (Higdon, 1998). For some kernel basis $\mathbf{K}$ with dimension $l \times \kappa$, where $\kappa < l$, the discrepancy is written as

$$\boldsymbol{\eta} = \mathbf{K}\boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathrm{N}(0, \psi_K \boldsymbol{\mathcal{I}}_\kappa)$$

i.e. in this model, the discrepancy does not need to be specified over the original field, or on the subspace defined by $\boldsymbol{\Gamma}$, and ψ_K is a variance parameter.

To reduce the computational cost, this model is reformulated on the $(q + \kappa)$-dimensional space:

$$(\boldsymbol{\Gamma}_K^T \boldsymbol{\Gamma}_K)^{-1} \boldsymbol{\Gamma}_K^T \mathbf{z} = \begin{pmatrix} \mathbf{c}(\mathbf{x}^*) \\ \boldsymbol{\xi} \end{pmatrix} + (\boldsymbol{\Gamma}_K^T \boldsymbol{\Gamma}_K)^{-1} \boldsymbol{\Gamma}_K^T \mathbf{e}$$

where $\boldsymbol{\Gamma}_K = (\boldsymbol{\Gamma}_q, \mathbf{K})$. Then, the distribution for the projection of the observations can be written as

$$(\boldsymbol{\Gamma}_K^T \boldsymbol{\Gamma}_K)^{-1} \boldsymbol{\Gamma}_K^T \mathbf{z} \sim \mathrm{N}\left(\begin{pmatrix} \mathrm{E}(\mathbf{c}(\cdot)) \\ 0 \end{pmatrix}, \begin{pmatrix} \mathrm{Var}(\mathbf{c}(\cdot)) & 0 \\ 0 & \psi_K \boldsymbol{\mathcal{I}}_\kappa \end{pmatrix} + \sigma^2 (\boldsymbol{\Gamma}_K^T \boldsymbol{\Gamma}_K)^{-1} \right)$$

where $\mathrm{E}(\mathbf{c}(\cdot))$ is a vector of the individual emulator means, and $\mathrm{Var}(\mathbf{c}(\cdot))$ is a diagonal matrix containing the corresponding emulator variances. Prior distributions for the model parameters are selected, with inverse gamma priors for σ^2 and ψ_K, and the posterior distribution for $\mathbf{x}^*$ is found using MCMC. The explicit prior choices are detailed in the supplementary information for Chang et al. (2014a), where the method is applied to an ice sheet model.

Whether this method is suitable for calibrating high-dimensional model output depends on the basis used for emulation of the ensemble, $\boldsymbol{\Gamma}_q$. When sampling from the posterior distribution, values for both the input parameters and the discrepancy parameters are obtained. If the basis is not suitable for representing the observations $\mathbf{z}$, for example if it induces biases in the field $\boldsymbol{\Gamma}_q \mathbf{c}(\mathbf{x}^*)$, then this will be accounted for by the discrepancy term, with these parameters and $\mathbf{x}^*$ jointly tuned. However, if the basis used for projection is

poor in some sense, then these would only be the best input parameters given this discrepancy, and the computer model may in fact have a superior $\mathbf{x}^*$, with 'less' discrepancy. Minimising discrepancy is preferred as this is a statistical model, in comparison with the complicated computer model code containing physical equations, so if forecasts are to be made, it is desired that the discrepancy has as little effect as possible. For more discussion about what constitutes a suitable basis, see Section 2.6.1 and Chapter 4.

Chang et al. (2016) also follow this method for calibration, instead finding a lower-dimensional representation using logistic PCA as the spatial model output is binary.

Determining the discrepancy in this manner (using a separate basis) may only be necessary when there is reason to believe that it should be written in this way. Otherwise, defining the discrepancy over the original spatial field, or on the basis $\mathbf{\Gamma}_q$, may be acceptable, and will simplify the calibration.

2.6.1. Choice of $\mathbf{\Gamma}_q$

In general, the basis used for projecting the ensemble onto a low-dimensional space may contain biases. For expensive computer models, the ensemble is likely to be small (< 100), exploring an extremely limited region of $\mathcal{X}$. It is possible that all of the ensemble runs contain output that differs in a possibly systematic way from the observed field $\mathbf{z}$, and hence a basis calculated from the ensemble using SVD is likely to also contain these biases. Any linear combinations of these basis vectors then may also lead to reconstructed fields that contain these same biases.

However, it may not be accurate to then conclude that this implies that these biases are a structural error in $f(\cdot)$; rather, it may only suggest that given the current ensemble and choice of basis, it is not possible to find fields similar to $\mathbf{z}$. The chosen basis restricts emulated outputs to a subspace of the full output space if $n < l$, and there is no guarantee that this subspace should include $\mathbf{z}$.

If the basis choice implies that $\mathbf{z}$ cannot be found for any choice of $\mathbf{x}$, this may lead to problems in calibration: if there is a setting of the parameters $\mathbf{x}^*$ that leads to $f(\mathbf{x}^*) = \mathbf{z}$ (possibly up to observation error), the basis choice may prevent this $\mathbf{x}^*$ from being identified.

For more in-depth discussion on basis selection, see Chapter 4.

2.7. History matching

History matching is another technique in the uncertainty quantification literature that can be used for searching for appropriate input values for $f(\cdot)$. Instead of explicitly calculating a distribution for the best input setting for the model, history matching rules out regions of parameter space that are unlikely to give output similar to historical observations of the physical system (Craig et al., 1996, Vernon et al., 2010, Edwards et al., 2011, Gladstone et al., 2012, Williamson et al., 2013, McNeall et al., 2013, Williamson et al., 2015, Vernon et al., 2016, Rodrigues et al., 2016). History matching can be used to explore what $f(\cdot)$ is unable to do, by removing the parts of $\mathcal{X}$ that lead to model output that is inconsistent with the observational data under a given uncertainty description.

Unlike calibration, history matching does not require the specification of full probability distributions for all terms involved, with a Bayes linear philosophy applied instead so that only prior means and variances are needed (Craig et al., 1996).

The statistical model used to link $f(\cdot)$ with the real system is the same as in the version of calibration with only one historical set of observations to match to, as in (2.27):

$$z_i = f_i(\mathbf{x}^*) \oplus \eta_i \oplus e_i$$

where each term has a subscript i as it is possible to simultaneously history match using multiple model outputs. Prior means and variances are required for each of the uncertain quantities involved in this equation.

In order to rule out regions of the input parameter space $\mathcal{X}$ that are unlikely to lead to model output consistent with the observations z_i, under the given error specification, the 'implausibility' can be calculated (Craig et al., 1996, Vernon et al., 2010, Williamson et al., 2013), giving a measure of how far away the output of the model at $\mathbf{x}$ is from the observations:

$$\mathcal{I}_i(\mathbf{x}) = \frac{|z_i - \mathrm{E}[f_i(\mathbf{x})]|}{\sqrt{\mathrm{Var}[z_i - \mathrm{E}[f_i(\mathbf{x})]]}} \tag{2.30}$$

where, given the statistical model in (2.27):

$$\begin{aligned}
\mathrm{Var}[z_i - \mathrm{E}[f_i(\mathbf{x}^*)]] &= \mathrm{Var}[z_i - y_i + y_i - \mathrm{E}[f_i(\mathbf{x}^*)]] \\
&= \mathrm{Var}[e_i + y_i - f_i(\mathbf{x}^*) + f_i(\mathbf{x}^*) - \mathrm{E}[f_i(\mathbf{x}^*)]] \\
&= \mathrm{Var}[e_i + \eta_i + f_i(\mathbf{x}^*) - \mathrm{E}[f_i(\mathbf{x}^*)]] \\
&= \mathrm{Var}[e_i] + \mathrm{Var}[\eta_i] + \mathrm{Var}[f_i(\mathbf{x}^*) - \mathrm{E}[f_i(\mathbf{x}^*)]] \\
&= \mathrm{Var}[f_i(\mathbf{x}^*)] + \mathrm{Var}[e_i] + \mathrm{Var}[\eta_i]
\end{aligned} \tag{2.31}$$

Therefore, in order to perform history matching for a scalar output of a computer model, all that is required is an emulator for $f_i(\cdot)$, a value for the observations z_i, and a specification of the expectation and variance for the observation error e_i and the discrepancy η_i.

The implausibility is a scaled version of the difference between the observations and the emulator mean at $\mathbf{x}$, scaled by the emulator variance at $\mathbf{x}$, and the other sources of uncertainty from (2.27), the observation error variance and the discrepancy variance. As the value of $\mathcal{I}(\mathbf{x})$ increases, it is less likely that $f(\mathbf{x})$ is an accurate representation of z under this model. A low value of $\mathcal{I}(\mathbf{x})$ indicates that either $f(\mathbf{x})$ is a good match for z given the current emulator, or that there is currently too much uncertainty (for example, from the emulator variance at this point) to be able to confidently rule out $\mathbf{x}$.

Using this definition of the implausibility, the following subset of parameter space can be defined (Williamson et al., 2013):

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}(\mathbf{x}) < a\} \tag{2.32}$$

where a is some selected tolerance to error. In the univariate case, this cut-off is commonly set as 3, based on Pukelsheim's Three Sigma Rule (Pukelsheim, 1994), which states that for a unimodal variable X, the probability that X lies more than 3 standard deviations from its mean is less than 0.05. In terms of the implausibility, this means that the probability that a value of $\mathbf{x}$ consistent with z gives an implausibility of greater than 3 is less than 5%.

'NROY' refers to the 'Not Ruled Out Yet' part of parameter space, the space of 'not implausible' points. That is, they have not been definitively ruled out so far, given the current emulator and tolerances to error, and the set of outputs currently considered. This

space may contain parameter settings that give model output inconsistent with z: there may be large uncertainty on emulator predictions in parts of parameter space, and hence it would not be possible to confidently rule out $\mathbf{x}$ here yet.

There may not only be one output of the computer model that is of interest. If it is assumed that there is no relationship between k different outputs, then emulators can be built independently for each output. Indeed, even if the outputs are related, if accurate univariate emulators can be built for the outputs, then this may be preferable to multivariate emulation. If some assumption can also be made about the observation error and discrepancy for each of the outputs, then k implausibilities $\mathcal{I}_i$ can be calculated for a single parameter choice $\mathbf{x}$. In order to define an NROY space in this situation, Craig et al. (1997) introduces the 'maximum implausibility measure':

$$\mathcal{I}_M(\mathbf{x}) = \max_i \mathcal{I}_i(\mathbf{x}) \tag{2.33}$$

Similarly, the 'second maximum implausibility measure' is given by

$$\mathcal{I}_{2M}(\mathbf{x}) = \max_i(\{\mathcal{I}_i(\mathbf{x})\}\backslash\mathcal{I}_M(\mathbf{x})) \tag{2.34}$$

In general, the 'j^{th} maximum implausibility measure' is

$$\mathcal{I}_{jM}(\mathbf{x}) = \max_i(\{\mathcal{I}_i(\mathbf{x})\}\backslash\{\mathcal{I}_M(\mathbf{x}), \mathcal{I}_{2M}(\mathbf{x}), \ldots, \mathcal{I}_{(j-1)M}(\mathbf{x})\}) \tag{2.35}$$

The NROY space using this measure is then

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_{jM}(\mathbf{x}) < a\} \tag{2.36}$$

Setting $j > 1$ gives a more conservative rule for removing regions of space: if the maximum is simply used, then there is the danger that important regions of $\mathcal{X}$ will be ruled out. Even with accurate emulators, it is expected that 5% of the model outputs would rule out a parameter setting that is perfectly consistent with $\mathbf{z}$. Allowing some of the outputs to have an implausibility greater than the chosen cut-off reduces the risk of this possibility.

2.7.1. Refocussing

History matching can be done in several 'waves', as in Vernon et al. (2010). At the first wave, an ensemble is designed to cover the whole input space $\mathcal{X}$, and an emulator is constructed based on these runs of the computer model. History matching is then carried out as above. Assuming that NROY space is non-empty, a new ensemble of runs can be designed in this NROY space, and a second wave of history matching can be performed after building an emulator using these new runs. This method is called 'refocussing'. Subsequent waves all rely on an ensemble of points in the current NROY space, and an emulator being built for the outputs of the model that are being matched on.

Formally, refocussing proceeds as follows. First, a sample of n_1 points is taken from the complete parameter space $\mathcal{X}$, and $f(\cdot)$ is run at these points, giving an initial ensemble $\mathbf{F}^{(1)}$:

$$\mathbf{F}^{(1)} = (f(\mathbf{x}_{1,1}), \ldots, f(\mathbf{x}_{1,n_1}))^T$$

Using this computer model output, an emulator $f^{(1)}(\mathbf{x})$ is constructed, valid on $\mathcal{X}$. Given the expectation and variance from this emulator for a parameter choice $\mathbf{x}$, and observations $\mathbf{z}$, the implausibility $\mathcal{I}^{(1)}(\mathbf{x})$ can be calculated, and an NROY space is defined as (assuming the standard choice of $a = 3$):

$$\mathcal{X}^{(1)} = \{\mathbf{x} \in \mathcal{X} \mid \mathcal{I}^{(1)}(\mathbf{x}) < 3\}$$

For the second wave of history matching, sampling is now carried out only within the existing NROY space, $\mathcal{X}^{(1)}$, with a new ensemble defined as:

$$\mathbf{F}^{(2)} = (f(\mathbf{x}_{2,1}), \ldots, f(\mathbf{x}_{2,n_2}))^T$$

Similarly as for the previous wave, an emulator $f^{(2)}(\mathbf{x})$ using $\mathbf{F}^{(2)}$ is built. Any runs from $\mathbf{F}^{(1)}$ that are in $\mathcal{X}^{(1)}$ may be incorporated into this emulator, for example as a validation data set. The emulator $f^{(2)}(\mathbf{x})$ is valid only in $\mathcal{X}^{(1)}$. Again, this emulator can be used to define a new NROY space:

$$\mathcal{X}^{(2)} = \{\mathbf{x} \in \mathcal{X}^{(1)} \mid \mathcal{I}^{(2)}(\mathbf{x}) < 3\}$$

In general, at wave m, a sample of size n_m is drawn from $\mathcal{X}^{(m-1)}$ to create an ensemble

$$\mathbf{F}^{(m)} = (f(\mathbf{x}_{m,1}), \ldots, f(\mathbf{x}_{m,n_m}))^T$$

of model runs in the current NROY space. From this, an emulator $f^{(m)}(\mathbf{x})$ is built, valid in $\mathcal{X}^{(m-1)}$, with which $\mathcal{I}^{(m)}(\mathbf{x})$ is calculated. Using this, the wave m NROY space is:

$$\mathcal{X}^{(m)} = \{\mathbf{x} \in \mathcal{X}^{(m-1)} \mid \mathcal{I}^{(m)}(\mathbf{x}) < 3\}$$

A benefit of this method is that as later waves are reached, the density of points that the computer model has been run at is greater in the reduced space than in the original space. The emulator built at this wave only needs to be accurate over the current NROY space, and hence should be a better proxy for $f(\cdot)$ than the wave 1 emulator. As waves proceed and the size of NROY space decreases, the quality of the emulator will continue to improve, and the variance shrink. After reducing the size of parameter space and having an accurate emulator, calibration may also be used to find a more accurate probability distribution for $\mathbf{x}^*$ than a calibration performed over $\mathcal{X}$. This would then lead to better results if forecasting is the intention (see Chapter 5 for an application of this).

A further benefit of performing waves in this manner is that initially, some model outputs may be hard to emulate accurately, potentially due to extremely non-physical behaviour observed in regions of space. By first constraining $\mathcal{X}$ using outputs that can be emulated, it may become more straight-forward to emulate other outputs of interest. This is also an argument in favour of history matching over calibration. Calibration is always performed over the complete input space, hence emulators are required for all outputs that are being calibrated to.

Another way that history matching differs from calibration is that in calibration, the result is always a probability distribution over the input space. Therefore, even if $f(\cdot)$ is an extremely poor model for the system y, calibration will give a distribution for $\mathbf{x}^*$, potentially with a peak of density somewhere in the input space. If $f(\cdot)$ cannot represent the system, then this result is meaningless, and the calibration distribution is unfit to be used in any further applications, e.g. forecasting.

In the scenario described here, history matching would rule out the entire parameter

space, which intuitively makes sense if the computer model is not representative of the true system. Finding an empty NROY space suggests that there are no parameter settings that give output 'close-enough' to z, under the current specification of the discrepancy. Ruling out all of $\mathcal{X}$ in this manner may suggest structural errors are present in the model, and suggests that the specification of the discrepancy variance may need to be re-visited. This may lead to decreased implausibilities, and potentially result in less space being ruled out.

2.7.2. Multivariate history matching

The univariate definition of history matching can be generalised so that vectors of observations (whether this is a time series, observations over a spatial field, or observations of multiple different attributes of a model) can be matched to. Craig et al. (1997) give the multivariate definition of the implausibility as

$$\mathcal{I}(\mathbf{x}) = (\mathbf{z} - \mathrm{E}[f(\mathbf{x})])^T (\mathrm{Var}(\mathbf{z} - \mathrm{E}[f(\mathbf{x})]))^{-1} (\mathbf{z} - \mathrm{E}[f(\mathbf{x})]) \tag{2.37}$$

The same statistical model linking the real system with the computer model (equation (2.27)) is still used for the multivariate case, with the same definitions for the observation error and discrepancy as before:

$$\mathbf{z} = f(\mathbf{x}^*) \oplus \boldsymbol{\eta} \oplus \mathbf{e}$$

so that

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

This is the same as for the univariate case in (2.31), with the exception that each of the variances are now $l \times l$ matrices rather than scalars, where l is the dimension of the observations, $\mathbf{z}$. Specifying the error and discrepancy variances can now be a more challenging task, because covariances between the outputs may need to be taken into account to achieve the greatest accuracy.

The observations being matched to are now of a higher dimension, hence the cut-off used to define NROY space changes. Vernon and Goldstein (2009) and Vernon et al. (2010) use the 99.5% value of a chi-squared distribution, with the number of degrees of freedom

equal to the dimension of the observations, l:

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}(\mathbf{x}) < \chi^2_{l,0.995}\}$$

The justification for this comes from Goldstein and Wooff (2007). The multivariate implausibility in (2.37) is analogous to the definition of the discrepancy of an observed vector in Bayes linear methods:

$$\text{Dis}(\mathbf{d}) = (\mathbf{d} - \text{E}(\mathbf{D}))^T \text{Var}(\mathbf{D})^{-1} (\mathbf{d} - \text{E}(\mathbf{D}))$$

where $\mathbf{d}$ is the observed value of $\mathbf{D}$, an l-dimensional quantity with prior expectation $\text{E}(\mathbf{D})$ and prior variance $\text{Var}(\mathbf{D})$. Assuming that the data has a multivariate Normal distribution implies that the discrepancy (and hence the multivariate implausibility) can be approximated by a chi-squared distribution with l degrees of freedom.

History matching can be used for spatial or other high-dimensional problems in a number of ways, in combination with the various types of emulators described previously. If a summary of the output is used to match to observations, for example the mean global temperature, then emulators can be built for this statistic, and the univariate version of the implausibility used to find NROY space. Vernon et al. (2010) fit multivariate emulators to multiple outputs and then history match across these outputs using the multivariate implausibility, accounting for correlations between outputs.

For high-dimensional problems, the basis projection method for emulation is commonly used, and history matching using emulators for basis coefficients will be the focus later (Chapters 4 and 5). After building emulators for the coefficients, there are three potential ways to proceed with history matching.

Firstly, the univariate implausibility can be calculated for each set of coefficients, and space ruled out using the maximum implausibility measure in (2.36), after the observations, and error and discrepancy variances are projected onto the same basis. More formally, the implausibility for the coefficients on basis vector i is calculated as

$$\mathcal{I}_i(\mathbf{x}) = \frac{|c_i(\mathbf{z}) - \text{E}(c_i(\mathbf{x}))|}{\sqrt{\text{Var}(c_i(\mathbf{z}) - \text{E}(c_i(\mathbf{x})))}}$$

where $\mathbf{c}(\mathbf{z})$ is defined similarly to the projection of $f(\mathbf{x})$ in (2.17):

$$\mathbf{c}(\mathbf{z}) = (\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T(\mathbf{z} - \boldsymbol{\mu})$$

so that $c_i(\mathbf{z})$ is the coefficient for the projection of $\mathbf{z}$ onto the i^{th} column of $\mathbf{\Gamma}$. The variance term on the denominator can again be written in terms of the observation error and discrepancy:

$$\begin{aligned}
\text{Var}[c_i(\mathbf{z}) - \text{E}[c_i(\mathbf{x}^*)]] &= \text{Var}[c_i(\mathbf{z}) - c_i(\mathbf{y}) + c_i(\mathbf{y}) - \text{E}[c_i(\mathbf{x}^*)]] \\
&= \text{Var}[c_i(\mathbf{e}) + c_i(\mathbf{y}) - c_i(\mathbf{x}^*) + c_i(\mathbf{x}^*) - \text{E}[c_i(\mathbf{x}^*)]] \\
&= \text{Var}[c_i(\mathbf{e}) + c_i(\boldsymbol{\eta}) + c_i(\mathbf{x}^*) - \text{E}[c_i(\mathbf{x}^*)]] \\
&= \text{Var}[c_i(\mathbf{e})] + \text{Var}[c_i(\boldsymbol{\eta})] + \text{Var}[c_i(\mathbf{x}^*) - \text{E}[c_i(\mathbf{x}^*)]] \\
&= \text{Var}[c_i(\mathbf{x}^*)] + \text{Var}[c_i(\mathbf{e})] + \text{Var}[c_i(\boldsymbol{\eta})]
\end{aligned}$$

The first term in this sum is the variance from the emulator for the coefficients for basis vector i. The variance for the projection of the error and discrepancy variances is calculated as follows (assuming that these quantities are specified over the original field):

$$\begin{aligned}
\text{Var}[\mathbf{c}(\mathbf{e})] &= \text{Var}[(\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T\mathbf{e}] \\
&= (\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T\text{Var}[\mathbf{e}]\mathbf{\Gamma}(\mathbf{\Gamma}^T\mathbf{\Gamma})^{-T}
\end{aligned}$$

This is an $n \times n$ matrix. For univariately history matching the coefficients on each basis vector, the variance $\text{Var}[c_i(\mathbf{e})]$ would be set as the i, i^{th} entry of $\text{Var}[\mathbf{c}(\mathbf{e})]$, ignoring any correlations. In the case of orthonormal basis vectors, this is the same as

$$\text{Var}[c_i(\mathbf{e})] = \text{Var}[\mathbf{c}(\mathbf{e})]_{ii} = \boldsymbol{\gamma}_i^T\text{Var}[\mathbf{e}]\boldsymbol{\gamma}_i$$

Another option is to calculate the multivariate implausibility for all of the emulated basis vectors, by stacking the expectation from each emulator into a vector, and setting the variance as a diagonal matrix of the individual emulator variances (as in Higdon et al. (2008a)), as the emulators are assumed to be independent:

$$\begin{aligned}
\text{E}[\mathbf{c}(\mathbf{x})] &= (\text{E}[c_1(\mathbf{x})], \ldots, \text{E}[c_n(\mathbf{x})])^T \\
\text{Var}[\mathbf{c}(\mathbf{x})] &= \text{diag}(\text{Var}(c_1(\mathbf{x})), \ldots, \text{Var}(c_n(\mathbf{x})))
\end{aligned}$$

The multivariate implausibility in (2.37) then becomes

$$\mathcal{I}(\mathbf{x}) = (\mathbf{c}(\mathbf{z}) - \mathrm{E}[\mathbf{c}(\mathbf{x})])^T(\mathrm{Var}(\mathbf{c}(\mathbf{x})) + \mathrm{Var}(\mathbf{c}(\mathbf{e})) + \mathrm{Var}(\mathbf{c}(\boldsymbol{\eta})))^{-1}(\mathbf{c}(\mathbf{z}) - \mathrm{E}[\mathbf{c}(\mathbf{x})]) \quad (2.38)$$

This has the advantage that covariances in the (projections of the) error and discrepancy variance matrices are included, as well as only requiring an inversion of an $n \times n$ matrix for every $\mathbf{x}$ (or, more likely, a $q \times q$ matrix as the basis will be truncated after $q < n$ vectors).

The final option is analogous to the multivariate calibration in Wilkinson (2010) (equation (2.19)), where the expectations and variances for the emulators on the reduced space are projected back to the original space. The drawback of this method is that it requires an inversion of the $l \times l$ variance matrix every time the implausibility is evaluated at a new parameter setting $\mathbf{x}$, and for a reasonable number of parameters, millions of evaluations can be required to build an adequate picture of NROY space. Therefore, for large spatial fields, history matching (or calibrating) after projecting back to the l-dimensional field may not be practical.

2.7.3. Discrepancy

In both calibration and history matching, it is important to have an accurate representation of the discrepancy between the computer model and the real-world system. As an example, in univariate history matching, simply decreasing the discrepancy variance could lead to a change from the whole space being classified as NROY, to considering all of $\mathcal{X}$ to be implausible.

The importance of the discrepancy is illustrated by Brynjarsdóttir and O'Hagan (2014). In this paper, the authors find that ignoring discrepancy can lead to incorrect inferences about the true parameter values when applying Bayesian calibration in a problem. This can lead to biased predictions if the posterior distribution is then used for forecasting. Therefore, including an accurate discrepancy is critical, and how this should be modelled is an important question.

Kennedy and O'Hagan (2001a) assign a Gaussian process prior to the discrepancy term (equation (2.23)), specifying a mean $m_2(\cdot)$, and covariance $\sigma_2^2 C_2(\cdot, \cdot)$. This adds extra hyperparameters that must be determined. These hyperparameters are estimated and given

a fixed value here, while they are treated as uncertain by Higdon et al. (2004). Given that the ensemble is the only data available, it is involved in identifying the hyperparameters for both the emulator of $f(\cdot)$ and of $\boldsymbol{\eta}(\cdot)$.

Kennedy and O'Hagan (2001a) use the ensemble of model runs to estimate the hyperparameters for the model for $f(\cdot)$. Given these fixed values, the values for the discrepancy hyperparameters are then estimated using $\mathbf{z}$. Therefore, if the prior beliefs about the discrepancy are weak, then the posterior will rely heavily on the ensemble, and the difference between this and $\mathbf{z}$. If there is something in the model output that appears to be structural error, and hence should be accounted for by the discrepancy term, is it possible to be sure that this is not truly a structural error, and instead $f(\cdot)$ has not yet been observed in the correct parts of $\mathcal{X}$? This method will potentially overestimate the discrepancy, dependent on the ensemble.

Chang et al. (2016) use an alternative method for setting the discrepancy. However, it also makes use of the limited ensemble runs. The example here is an ice sheet model with binary output. For an individual grid box i, a non-zero contribution is given to the discrepancy term if the proportion of ensemble runs where the output is different from the observations is greater than a chosen threshold (here, 0.5), so that common ensemble deviations away from the parameters are highlighted.

More specifically, the discrepancy is represented by a single basis vector $\gamma_{\boldsymbol{\eta}}$ of length l, with i^{th} entry η_i defined as

$$\eta_i = \begin{cases} log(\frac{1+r_i}{1-r_i}) & \text{if} \quad |r_i| > c \\ 0 \qquad \text{otherwise} \end{cases}$$

for a chosen threshold c, where

$$r_i = \frac{1}{n} \sum_{j=1}^{n} sgn(f_i(\mathbf{x}_j) - z_i) I(f_i(\mathbf{x}_j) \neq z_i)$$

An issue with this method may be that if the ensemble contains biases away from the observations, then not only is this ensemble used to select a basis to be used for emulation, but also for defining the discrepancy, 'double-counting' the errors. In this example, it is possible that for output i, only zeros have been observed in the small number of model

runs n, whereas the observation at this location is equal to 1. This method would specify a non-zero discrepancy for this location.

Not observing a certain output in the ensemble does not necessarily imply that this is a structural error and should be treated as discrepancy. A situation may exist where $f(\cdot)$ has not yet been run at a suitable parameter setting, but such a setting does exist. Using this method will assume that it does not, as any deviations in the ensemble will automatically be passed into the discrepancy term, which may make it impossible to find the true $\mathbf{x}^*$.

2.8. Tuning climate models

Aside from calibration and history matching, other methods for tuning climate models, from outside of the uncertainty quantification literature, are also used. Hourdin et al. (2016) provides an overview of current practices in the climate model tuning community.

Another common approach is to define a cost function that relates the climate model output to observations, and to minimise this (Bellprat et al., 2012, Zou et al., 2014, Zhang et al., 2015), either by fitting a simple, fast-to-evaluate model to the chosen cost function, or by iteratively selecting a new parameter setting and then re-running the climate model at this input.

Bellprat et al. (2012) objectively tune a regional climate model by fitting a second order polynomial metamodel (Simpson et al., 2001) to a 'performance score' used to relate the model output to the observations. The fitted model measures the change in a particular output value when the input parameter vector is varied away from a default setting $\mathbf{x}_{def}$. The metamodel has the following form:

$$f_i(\mathbf{x}_p) = f_i(\mathbf{x}_{def}) + \mathbf{x}_p^T \beta_0 + \mathbf{x}_p^T \boldsymbol{\beta} \mathbf{x}_p$$

where $\mathbf{x}_p = \mathbf{x} - \mathbf{x}_{def}$ is the difference between a new parameter setting $\mathbf{x}$ and the default setting, β_0 is a linear term, and $\boldsymbol{\beta}$ contains coefficients for the quadratic and interaction terms for the parameters. f_i is a single output from the regional climate model, i.e. a separate metamodel is fitted for every region, monthly average, and each of the three output variables being used to optimise the parameters.

Using this set of metamodels, the 'performance score' (PS) is used to measure how well the model output fits the observations:

$$PS = \exp(-0.5PI^2)$$
$$PI = \left\langle \frac{f_i(\mathbf{x}) - \mathbf{z}_i}{\sigma_o + \sigma_{iv} + \mathbf{e}_i} \right\rangle$$

where $\langle \cdot \rangle$ denotes averaging over each individual model output value. The denominator contains sources of uncertainty: σ_o, the interannual variability, σ_{iv}, the internal variability, and $\mathbf{e}_i$, the observation error for output i.

Despite the large number of individual metamodels that need to be fitted, as it is simply a quadratic regression, it is quick to evaluate predictions. A Latin hypercube (Morris and Mitchell, 1995) sample is taken over $\mathcal{X}$, and the regions where PS is maximised can be identified.

Other functions can be chosen for optimisation, some based on the spatial standard deviation and correlations between output variables. Zou et al. (2014) and Zhang et al. (2015) select a cost function to optimise and a starting value for the parameters, and then use an optimisation technique to select new parameter settings one at a time, seeking to move towards the best regions of $\mathcal{X}$ according to the cost function. This relies on it being computationally possible to perform a new run of the computer model when the new parameter setting has been selected, which may not always be the case. Furthermore, in Zhang et al. (2015) each climate model run takes around 6 hours, so that optimisation in this sequential manner is a lengthy process.

These methods do not explicitly incorporate any knowledge about the discrepancy of the model, or any spatial patterns in the output fields. They rely on the fact that the cost functions used will vary smoothly as the parameters vary.

Compared to the uncertainty quantification techniques for tuning a climate model, there are various advantages and disadvantages. The simplicity of fitting a meta-model rather than a Gaussian process may be an attractive option. However, although making predictions with a Gaussian process emulator requires more computational time than a single regression, this benefit is negated by the fact that an individual regression model is required for every single timepoint, grid box and model output. The multivariate approaches

in the UQ literature provide flexible ways of modelling this type of output, incorporating spatial and temporal correlations, as well as correlations across the input space. Furthermore, uncertainty bounds being provided, and then accounted for in calibration or history matching, give increased robustness to any conclusions.

2.8.1. Data assimilation

Data assimilation is another popular method for constraining the parameters of a climate model using observational data (Annan et al., 2005). Prior distributions are selected for the input parameters $\mathbf{x}^*$, and these are then updated over time using runs of the computer model, and the observations $\mathbf{z}$.

Annan et al. (2005) apply data assimilation (following the method of Keppenne (2000)) to the ocean component of a climate model, with output given over a 36×36 grid. Data assimilation uses the Kalman filter equations (Kalman, 1960), used to update the model over time. The equation used for updating the model output and parameter settings is

$$\mathbf{x}^a = \mathbf{x}^f + \mathbf{K}(\mathbf{z} - \mathbf{H}\mathbf{x}^f)$$

for

$$\mathbf{K} = \mathrm{Var}(\mathbf{x}^f)\mathbf{H}^T(\mathbf{H}\mathrm{Var}(\mathbf{x}^f)\mathbf{H}^T + \mathrm{Var}(\mathbf{e}))^{-1}$$

The uncertainty on this update is given by the covariance matrix

$$\mathrm{Var}(\mathbf{x}^a) = (\boldsymbol{\mathcal{I}} - \mathbf{K}\mathbf{H})\mathrm{Var}(\mathbf{x}^f)$$

$\mathbf{x}^f = (f(\mathbf{x}_i)^T, \mathbf{x}_i)^T$ is a vector consisting of the model output at $\mathbf{x}_i$, with these input parameters added at the end. $\mathbf{x}^a$ is then an adjusted version of these, updated by using the Kalman term $\mathbf{K}$. $\mathbf{H}$ is a matrix that relates the model output with the observations.

An inversion of an $l \times l$ matrix is required at every time step. Therefore, for computational efficiency, the output field in this example is divided into 54 regions, and the output and parameters are updated simultaneously for each.

Applications of this and similar data assimilation methods are commonly used (Aksoy et al., 2006, Schirber et al., 2013, Ruiz et al., 2013, Juan Jose et al., 2013, Li et al.,

2016). Other cost functions can be minimised, but generally the model output and model parameters are combined into one vector, and then $\mathbf{x}^f$ is updated iteratively over time using the observations, possibly averaging results spatially at each step when the output has a high dimension.

A drawback of this method of tuning a climate model is that it allows the parameter values to vary over time. Whilst it may be true that there are different 'best' parameter values for different time periods, this does not allow projections to be made: if the final distribution for $\mathbf{x}$ only matches the last part of the observed data, why should this parameter setting necessarily be accurate in the future? (Williamson et al., 2015)

Another potential issue is that, when there is a large amount of data, for example when there are observations over a grid for the whole world as in global climate models, the output field must be split into regions. In each of these, the new parameter vector is updated locally at a time point, and these are then smoothed, to give one global set of parameters, before the updating at the next time point occurs. Optimising separately over regions risks an averaged parameter vector that is not a fit for the observational data anywhere if very different vectors are required to match the observations in different regions. Additionally, there may be regions of model output nothing like the observations that parameter values are being estimated for.

The UQ literature has the benefit that methods have been developed to search for settings of the parameters that are consistent with the observations, up to the model discrepancy as well as observation error, where high dimensionality can be overcome, for example by using the basis projection methods prior to calibration or history matching. Additionally, these methods do not necessarily require new runs of the computer model, instead relying on an initial ensemble of runs. In data assimilation, the model needs to be run at new parameter settings at each step, a large computational cost for complex models.

In this thesis, we will explore the use of uncertainty quantification techniques for tuning expensive computer models with high-dimensional spatial output.

3. Multi-wave emulation and calibration

3.1. Introduction

The main results of this chapter have appeared in Salter and Williamson (2016).

When designing and then analysing the results of a computer experiment, there are a number of decisions that a statistician may be faced with:

1. At which parameter settings should the computer model $f(\cdot)$ be run to create the ensemble $\mathbf{F}$?

2. What type of emulators should be built? Namely, is a regression sufficient, or should a correlated residual term be fitted?

3. If the goal is to find optimal parameter settings, how should this be achieved?

Question 1 may not be one that the statistician is able to dictate the answer to; especially in the case of complex computer models such as climate models, an ensemble that has already been run on a supercomputer for some other purpose may be provided. However, given an ensemble $\mathbf{F}$, there will always be a decision to be made about how to emulate the given data, and whether to use calibration, history matching, or some other tuning method, in order to find answers about where the model should be run so that it gives output similar to the observations.

If it is possible for the statistician to have a say in the design of the computer experiment, they must also decide how best to allocate these runs. For example, if n runs of the model can be performed, should these all be spent on a space-filling design over the full parameter space, or should the resources be split so that an initial space-filling design can

later be supplemented with more points in important regions of parameter space (which are unlikely to be known a priori)?

In this chapter, it is assumed that the model can be run at the desired settings efficiently, so that a comparison between performing a single wave and multiple waves can be carried out, while having the ability to test the accuracy of the results. History matching lends itself to being used over multiple waves iteratively (refocussing, as described in Section 2.7.1), therefore this will be the main focus of this study, although calibration is also considered alongside this.

Section 3.2 discusses the motivations for comparing regression and Gaussian process emulators, and why it can be attractive to use simpler emulators in some circumstances. Section 3.3 outlines the experiment to be performed on several toy examples, and Section 3.4 gives the results of this. Section 3.5 discusses an unexpected result found in the study. Finally, Section 3.6 compares the performance of these emulation techniques on a geological model.

3.2. Regression vs Gaussian process emulators

Although Gaussian processes are well-studied and used in the statistics literature (Higdon et al., 2008a,b, Vernon et al., 2010, Lee et al., 2013, McNeall et al., 2013, Vernon et al., 2016), in applications, particularly in climate science, regression-only emulators are often used (Rougier et al., 2009, Sexton et al., 2011, Holden et al., 2013, Williamson et al., 2013, 2015). Reasons given for this include the simplicity in fitting regressions and explaining these models to non-statisticians, the speed to evaluate predictions with a regression emulator, and that the correlated residual term makes a negligible difference in high-dimensional input space, in terms of reducing uncertainties around the sparse design points. In climate science in particular, it is argued that regression emulators can be used to represent the signal in the model output well enough, so that the (uncorrelated) residual can be interpreted as representing the internal variability of the model.

3.2.1. Tractability

As the ensemble size n increases, and as the number of waves increases, performing the number of emulator evaluations required to give a picture of NROY space increases significantly, and becomes increasingly intractable, due to the inversions of covariance matrices required. On a standard desktop with 4 cores, it takes 1 second to evaluate the output of a regression emulator for a sample of 1 million points, if $n = 200$. For a Gaussian process emulator for the same data, this process (if performed in parallel) takes 90 seconds. Similarly, if $n = 400$, the regression emulator requires 2 seconds, compared to 188 seconds for the Gaussian process.

If there is only one emulator required, then this difference in the computational time is perhaps trivial. However, in climate applications it could be problematic. For example, Lee et al. (2013) build 8192 emulators, one for each individual grid box for the output of a global aerosol model. If 1 million parameter choices are to be evaluated for each emulator, so that the full output field for each parameter setting can be calculated, then this calculation will take 8192 seconds (2 hours 15 minutes) if each emulator is a regression. If Gaussian process emulators are used for every grid box, 8 and a half days are needed; a faster computer or faster emulator is clearly required here so that predictions can be made in a reasonable time frame.

Even when only one output is being emulated, computational problems can still arise when there are multiple waves. Andrianakis et al. (2015) perform nine waves of history matching, and find an NROY space that is 10^{11} times smaller than the original parameter space. Therefore, in order to sample from the resulting NROY space, many millions of emulator evaluations may be required: to check whether a parameter choice is in NROY space, the emulator at each wave may need to be evaluated (although an efficient ordering of the waves could be chosen, based on which are most important for determining whether a point is ruled out or not, to reduce the number of evaluations required).

These examples highlight the difficulties with a Gaussian process emulator, and hence the attractiveness of the quick-to-evaluate alternative of regression. For problems with large n, Gramacy et al. (2015) fit Gaussian processes locally to reduce the computational time required by calibration. However, in their problem, $n = 26{,}458$, whereas our ensembles are generally small compared to this, and hence the benefits of this local fitting may not

be realised.

3.2.2. Sparse sampling of the input space

Another argument against using Gaussian process emulators is that in applications, the input space typically has a high dimension. Therefore, if the ensemble size n is small, the points in $\mathcal{X}$ are too spread out for a correlated residual term to have any effect on the emulator, because the correlations between points are essentially zero. Using a Gaussian process emulator will reduce the uncertainty around the design points, but when the sample is sparse, an important question is whether this can affect the calibration or history matching enough to negate the additional expertise and time needed to build the emulator, as well as the computational issue described in the previous section.

To illustrate the potential impact of fitting a Gaussian process emulator instead of a regression emulator when there is a high-dimensional input space, a toy function $f_1(\cdot)$ (defined in Appendix A) with 10 input parameters is considered. 200 design points are selected, and $f_1(\cdot)$ is run at each of these to give an ensemble $\mathbf{F}$, using which two emulators are built: one using only regression, and one with an added correlated residual term.

Figure 3.1 shows the prediction from each of these emulators, with the associated 99% uncertainty bounds, along a line in 10-dimensional space defined between design points $\mathbf{x}_1$ and $\mathbf{x}_2$. This plot demonstrates that the Gaussian process emulator is a better approximation of $f_1(\cdot)$ in this part of parameter space. As well as providing a better mean prediction, the uncertainty on this prediction is also smaller for the Gaussian process: the regression emulator has constant error bounds everywhere along this line, whereas for the Gaussian process, the fitted values of the correlation lengths $\boldsymbol{\delta}$ have reduced the uncertainty everywhere on this line. However, the uncertainty does not shrink to zero at the design points, due to the presence of a nugget (and this term appears to be large for this Gaussian process emulator).

The solid black line in this plot represents an assumed observed or 'true' value for the function. In this example, $z = 0$. The dotted lines around this value represent observation error. Although the true function is never within the margin of error for the observation in this region of $\mathcal{X}$, when history matching is carried out it may not be possible to rule

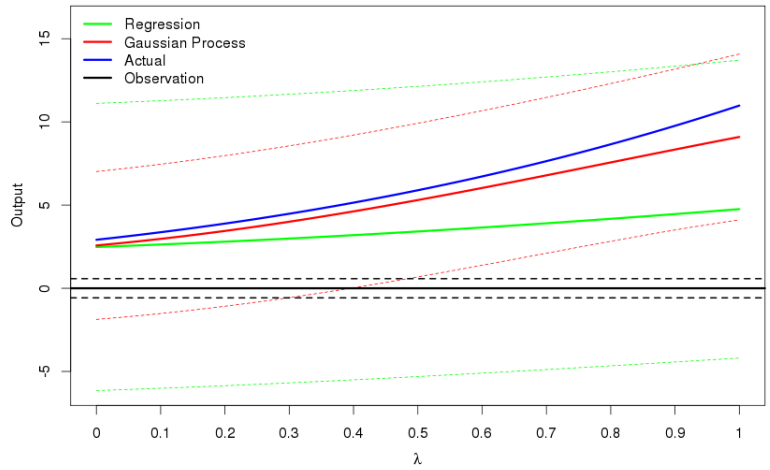

Figure 3.1. How the prediction and 99% uncertainty bounds change for a regression emulator (green) and a Gaussian process emulator (red) for a line between two design points x_1, x_2 in 10-dimensional space, with this line given by $\lambda x_2 + (1-\lambda)x_1$. The blue line shows the toy function. The observation is taken to be 0, observed with an observation error given by the dotted black lines. The nugget for the Gaussian process emulator is relatively large in this example.

out much or any of the parameter choices along this line, due to the uncertainty on the emulator predictions. Using the definition of the implausibility of a parameter setting $\mathbf{x}$ (Craig et al., 1996) from (2.30), i.e.

$$\mathcal{I}(\mathbf{x}) = \frac{|z - \mathrm{E}[f(\mathbf{x})]|}{\sqrt{\mathrm{Var}[z - \mathrm{E}[f(\mathbf{x})]]}} \tag{3.1}$$

we evaluate $\mathcal{I}(\mathbf{x})$ along this line for each of the two emulators. Figure 3.2 shows this implausibility.

From this plot, assuming a cutoff of $a = 3$ to define NROY space, i.e.

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}(\mathbf{x}) < 3\}$$

it is clear that no space can be ruled out using the regression emulator. This was evident from Figure 3.1, because the uncertainty bounds for the regression emulator's prediction included the observation and observation error within them. However, for the Gaussian process emulator, it is possible to rule out space for $\lambda > 0.52$, as the implausibility increases above 3 after this point: it is possible to confidently rule out this part of space based on the uncertainty on the emulator's prediction, and (correctly) say that this part of space leads to model output that is not consistent with $z = 0$ and the error on this observation.

This demonstrates that the Gaussian process emulator is superior to the regression em-

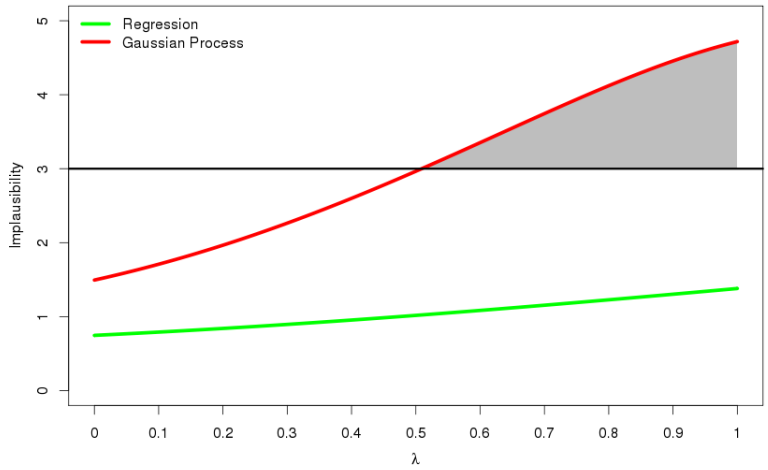

Figure 3.2. The implausibility $\mathcal{I}(x)$ for the above two emulators. With 3 chosen as the threshold for ruling out points, the regression emulator cannot rule out anything in this part of space, while the Gaussian process emulator can for $\lambda > 0.52$.

ulator, but only along this line through 10-dimensional space, of which infinitely many could be considered. This is also dependent on fitting suitable correlation lengths, and here the smoothness of the function allows the uncertainty to be reduced. However, it does motivate the comparison of these two emulator types across the full space: will the percentage of space that can be ruled out be significantly different when the Gaussian process emulator is used instead of regression?

3.3. Simulation study design

The goal is to investigate whether it is important to use a Gaussian process emulator rather than a regression emulator when the number of design points in high-dimensional space is small. Alongside this goal, the tractability issue of the multiple wave scenario will also be considered. Specifically, whether or not it is acceptable to initially use a regression, followed by a Gaussian process at a later wave.

Therefore, to compare regression and Gaussian process emulators in this context of multi-wave history matching, we perform the following experiment for a given function or computer model $f(\cdot)$:

1. An initial sample $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of size n is taken from parameter space $\mathcal{X}$, using a Latin hypercube maximin design (Morris and Mitchell, 1995).

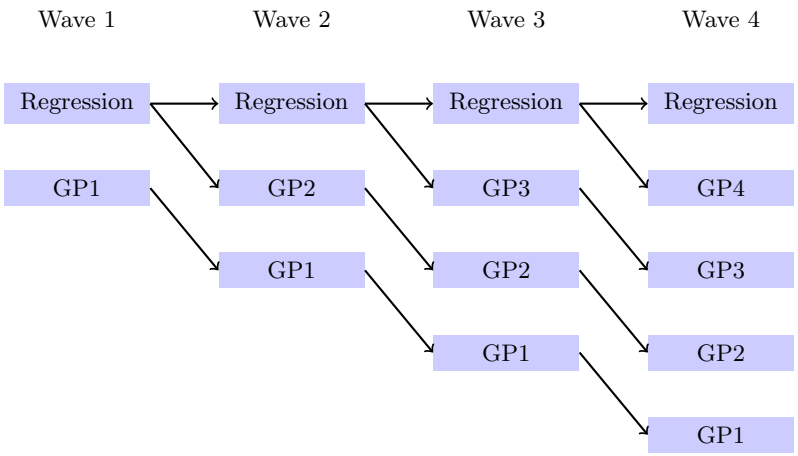

Figure 3.3. Flow chart showing the emulators built for a comparison between the regression-only case and the Gaussian process case. GP1 denotes that a Gaussian process is used from wave 1 onwards.

2. The ensemble $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$ is found by evaluating $f(\cdot)$ at each design point.

3. The ensemble is divided into a training and validation set, with a regression and Gaussian process emulator fitted to the training data. Validation checks are carried out before continuing.

4. By taking each of the validated emulators in turn, the size of NROY space is estimated by successively sampling 10,000 points from $\mathcal{X}$, continuing until 1,000 points are not ruled out according to the implausibility.

5. Steps 1-4 are repeated until four waves of history matching have been completed, for each path in Figure 3.3, with new ensembles created by sampling from the current NROY space.

Figure 3.3 shows the complete experiment that is carried out for an individual function $f(\cdot)$. There is an alternative multi-wave comparison generated at each wave between regression and Gaussian process emulators, as the result of taking an experiment that has thus far had only regression emulators, and adding a Gaussian process for the current NROY space. This allows a direct comparison to be made between regressions and Gaussian processes at every wave. We assume that once a Gaussian process emulator has been used, this type of emulator is always used thereafter, so that the NROY spaces defined using Gaussian process emulators then have new Gaussian process emulators fitted in them at subsequent waves.

For greater clarity: at wave 1, we fit a regression emulator and a Gaussian process emulator using the same ensemble. Each of these emulators defines a different NROY space. Therefore, at wave 2, there are two new ensembles generated: $\mathbf{F}_{reg}^{(2)}$ and $\mathbf{F}_{GP}^{(2)}$, sampled from the regression emulator's NROY space and the Gaussian process emulator's NROY space respectively. Using $\mathbf{F}_{reg}^{(2)}$, we fit new regression and Gaussian process emulators, comparable as they are fitted using the same design. Additionally, we fit a new Gaussian process emulator using $\mathbf{F}_{GP}^{(2)}$.

Therefore, at wave m, there are $m+1$ emulators fitted, defining $m+1$ different NROY spaces: one regression emulator, valid in an NROY space defined by only regression emulators, and m Gaussian processes, defined in NROY spaces where Gaussian processes started to be used from each of waves $1, \ldots, m$. Overall, there is one multi-wave history match carried out using only regression emulators, and this is compared to history matching when Gaussian processes start to be used at different waves.

This experiment allows for a regression emulator to be compared to a Gaussian process at every wave, and will also illustrate the combined benefits or otherwise of always using a Gaussian process. It may also allow conclusions to be drawn regarding whether it is sensible to initially use regression emulators, followed by Gaussian process emulators at later waves, when the density of design points has increased.

Since the uncertainty decreases around design points, we expect that more space will be ruled out when a Gaussian process is used, although how much more is the question that this study intends to answer. Intuitively, one would expect that the more waves a Gaussian process is used at, the smaller the resulting NROY space will be after four waves: using a Gaussian process at every wave should lead to the smallest NROY space, followed by starting to use a Gaussian process from wave 2, with the 'always regression' case ending with the largest NROY space after wave 4. However, the literature argues that the differences in the results are negligible (Rougier et al., 2009, Williamson et al., 2013).

3.3.1. Toy examples

We apply the above methodology for comparing emulator types, as well as single and multi-wave experiments, to four toy examples, designed to exhibit features and problems regularly encountered when calibrating or history matching climate models. These toy functions are defined in Appendix A.1. Each of the input parameters for these functions has been defined or scaled so that it takes values in $[-1, 1]$.

Function 1 ($f_1(\cdot)$, equation (A.1)) represents a smooth 10-dimensional unknown function, that should be well approximated by a fitted polynomial surface, so that both the regression and Gaussian process emulators ought to perform well due to the smoothness of the output. Function 2 ($f_2(\cdot)$, (A.2)) is a more complex 10-dimensional function containing periodic functions, that should favour the added flexibility that the Gaussian process emulator provides. Function 3 ($f_3(\cdot)$, (A.3)) has been defined to have more input dimensions (20), so that whether or not local variability around sparse points in higher-dimensional space affects history matching can be studied. This is also a typical situation in climate applications: this input space has the same dimension as the NEMO ocean model in Williamson et al. (2016).

Each of these three functions also contains a random term: each time the function is evaluated, a sample is drawn from a zero-mean Normal distribution with specified variance to add noise to the function output. This ensures that if the function is run more than once at the same input $\mathbf{x}$, the output will not necessarily be the same. This noise has been added to correspond to the internal variability of climate models.

The final function that we investigate is the borehole function, a standard test function (Worley, 1987, Morris et al., 1993), given in (A.4) as $f_4(\cdot)$.

Table 3.1 gives some additional information for each of the toy examples. The range of possible output values for the function are given, along with the output that is assumed to be the observed value of the function, z. In each case, this value is observed up to some error $\mathrm{Var}[e]$, and the variance of the random noise is also noted. The ensemble size n used for each function is given, where $f_3(\cdot)$ has a higher value for n due to the larger input space for this function.

Function	**Range**	z	**Var**$[e]$	**Var**$[\eta]$	n	**NROY size**
f_1	[-42, 59]	15	10^{-5}	0.05^2	200	0.17%
f_2	[-3, 12.5]	9	10^{-3}	0.15^2	200	0.24%
f_3	[-145, 136]	50	10^{-2}	0.5^2	400	0.28%
f_4	[0, 300]	100	10^{-3}	0	200	0.11%

Table 3.1. Information about the toy functions for history matching. Range denotes the spread of possible outputs for the function, and NROY size denotes the theoretical size of NROY space, given this error structure.

'NROY size' denotes the percentage of $\mathcal{X}$ (equivalently, the volume of space) that cannot be ruled out given this setting for the observations and these variances, if the function is known perfectly, i.e. an emulator is not required, and the function is simply evaluated so that there is no code uncertainty. To estimate this volume, a large (100,000 or more) Latin hypercube sample is taken from $\mathcal{X}$, and the percentage of the points that are not ruled out is calculated. This is theoretically the true NROY space that multi-wave history matching will eventually be able to find. These have all been designed to be small regions of parameter space, as is typically encountered in environmental applications.

3.3.2. Modelling choices

Within this experimental framework, there are a number of choices to be made to ensure a consistent and fair comparison.

We use a Bayesian regression approach (Gelman et al., 2014), as this accounts for the uncertainty in the regression coefficients $\boldsymbol{\beta}$, for which the true values are unknown. The regression emulator will be fitted using a stepwise approach, with a random noise variable included as one of the predictors. The noise variable is a random sample from a zero-mean Normal distribution, with a specified variance. Once the noise variable is selected by the stepwise algorithm, no further terms are added. The maximum number of predictors allowed in the model is a tenth of the sample size (which may be a too conservative choice, restricting the flexibility of the linear model that can be fitted), with powers of individual parameters and interactions between several parameters allowed, if the required lower order terms are also included. For the Gaussian processes, the mean function will be used to explain as much of the response as possible (Vernon et al., 2010), and this will be fitted using the same technique. Where the regression and Gaussian process emulators are directly comparable, i.e. the sample being used to fit each of them is the same, then

the mean function for the Gaussian process will simply be the regression.

For the Gaussian process emulators, we follow the prior choice and subsequent posterior analysis of Haylock and O'Hagan (1996), given by equations (2.6), (2.9), (2.10) and (2.11). This matches with the Bayesian regression technique of accounting for the uncertainty on the mean function coefficients.

The correlation function used in all of the emulators fitted in this study is the Gaussian correlation function, with the nugget expressed as a percentage of the residual variability that is unexplained by correlations between parameters:

$$C(x, x') = \nu I_{x=x'} + (1-\nu)\exp\left\{-\sum_i \left(\frac{x_i - x_i'}{\delta_i}\right)^2\right\}$$

where $I_{x=x'}$ is 1 if $x = x'$, and 0 otherwise. The nugget term is included here due to the random noise in the toy functions, hence the emulators should not interpolate the data.

Not all of the input parameters for the toy examples have much effect on the output. Therefore, the correlated residual term is only fitted for those variables deemed to be 'active' (Craig et al., 2001). These active variables can change over the waves: initially, some variables may be responsible for large-scale variation in the output, before other variables are more important once the size of NROY space decreases and **F** highlights more local variability. The active variables for a given Gaussian process emulator are chosen by selecting the variables that were at some point included during the selection of the mean function (i.e. for a variable to be considered active, it need not be in the final mean function, but it must have been deemed to have at least some effect on the output, prior to the number of predictors being reduced to meet the maximum criteria). This likely benefits the Gaussian process emulators compared to the regressions, as they are potentially fitted to more of the variables.

We perform history matching using the univariate definition of the implausibility (3.1), because only functions with a scalar output are considered in this study. The cutoff in the definition of NROY space will be set at 3, as this is the commonly-used setting based on Pukelsheim's Three Sigma Rule (Pukelsheim, 1994).

3.3.3. Parameter estimation

Rather than treating the emulator in a fully Bayesian way, we estimate the parameters of the correlation function for the Gaussian process, and fix their values at this estimate, as in Kennedy and O'Hagan (2001a). This is because it is assumed that the uncertainty due to estimation of these parameters is small or negligible compared to other sources of uncertainty, and because of the large number of emulators required in this study, needing to sample values for these parameters every time the emulator is evaluated will add to the computation time substantially.

Estimating the correlation parameters using maximum likelihood did not always give satisfactory results, so a different automatic approach based on cross-validation is used here, to ensure that all of the emulators validate well.

First, we choose an initial setting for the values of $\boldsymbol{\delta}$ and the nugget ν. This can be done using a heuristic such as the one outlined by Vernon et al. (2010), where the initial values for the correlation lengths are based on the order of each term in the regression model. At later waves, when there has been a Gaussian process emulator fitted previously, the starting values are the parameter values from the previous wave. An emulator is fitted for a choice of $\boldsymbol{\delta}$ and ν to the training data, and this choice is validated by checking the fit of the emulator to the training data, as well as checking whether this emulator leads to adequate predictions for the design points in the validation set.

More specifically, we validate the fit of the emulator on the training set by performing leave-one-out cross-validation for this data. That is, each point $\mathbf{x}_i$ in the training set is left out in turn, and the emulator is re-fitted without this point, and then used to predict $f(\mathbf{x}_i)$. To ensure that the emulator fits, the average standard error on these predictions is minimised, with the condition that roughly 95% of the true values lie within these error bars. Fitting the correlation lengths based on leave-one-out diagnostics is common (e.g. the DiceKriging R package (Roustant et al., 2012), Williamson et al. (2012)).

Alongside this cross-validation check, the emulator is used to predict the output at each of the points in the validation set, and again it is required that around 95% of the predictions are within 95% error bars.

Minimising the average cross-validation standard error, while simultaneously requiring that there are not too many or too few predictions outside of the uncertainty bounds, for both the training data and the validation data, leads to an emulator that fits the training data well, but also avoids the problem of overfitting. Furthermore, it can be confidently used for prediction.

Using these validation checks, we search for the values of the correlation function parameters using the following algorithm:

1. Select initial values for $\boldsymbol{\delta}$, ν.

2. Set $i = 1$.

3. Change the value of δ_i by 0.1.

4. Fit an emulator using the current values of the parameters.

5. Calculate the mean cross-validation standard error as above, and count the number of predictions outside 95% error bars for a) cross-validation of the training data, denoted by ϵ_1, and b) prediction of the validation data, ϵ_2.

6. If this mean is lower than the previous best, and ϵ_1 and ϵ_2 are both acceptable, return to step 3, and continue to perturb δ_i. Else, set $i = i + 1$, and return to step 3.

7. When $i = p_a + 1$, where p_a is the number of active variables, instead vary the value of ν by 0.0005, and proceed as for the δ_is.

8. Set the final parameter values as those that have the smallest mean cross-validation standard error, while also passing both of the validation checks.

9. Re-fit the emulator with these parameter values, including both the training and validation data sets.

0.1 has been chosen as the amount that parameters are varied by at each step as, from experience of fitting these types of emulators, it was found that this achieves a reasonable compromise between the ability to find good choices of the parameters and the amount of time required to fit emulators for every different parameter choice.

This method may not lead to parameter settings that are strictly optimal, due to the fact that each individual parameter is varied in turn, but it does allow values that are consistent with the sample data to be found, and that give an emulator with predictive ability. Furthermore, it does so automatically in short enough of a time frame so that the large number of emulators required for this study can be built.

An alternative to varying parameters one by one would be to use simulated annealing or a similar optimiser in place of this step, with the goal still to minimise the mean standard error while the validation checks are satisfied. To find strictly optimal values using this may take more computing time, hence the above algorithm is used for this study, as it has the desirable combination of finding acceptable parameters while not taking an excessively long time to do so.

In later chapters, we fit correlation parameters using simulated annealing, rather than the approach outlined here.

3.3.4. Validating emulators

Figure 3.4 gives an example of the validation plots that are generated when checking the fit of an emulator. The left panels show cross-validation plots for the training data, and the right panels show the predictions for the points in the validation set. The emulator predictions have 99% error bars, and the true function values are coloured green should they lie within these bars, and red otherwise (Williamson et al., 2015).

3.3.5. Sampling from NROY space

After wave 1, the new ensemble of size n is created by sampling from the current NROY space. This is done by creating Latin hypercube samples and evaluating the current emulators at these points to determine which are in NROY space. Sampling is continued until n runs in NROY space have been found. The aim here is that the selected runs will be spread out over NROY space, although there is no guarantee that different Latin hypercube samples will be spaced appropriately. However, this automatic approach should be sufficient for this experiment. If the problem was to history match one function over multiple waves, then more time could be allocated to creating the new ensemble.

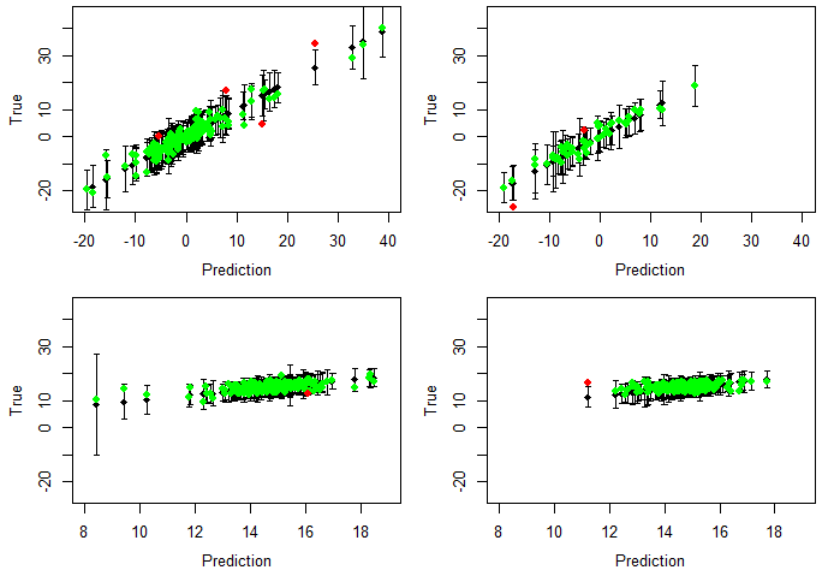

Figure 3.4. Leave-one-out cross validation plots (left) and prediction for the validation set (right), for the Gaussian process emulators for function 1, after wave 1 (top) and wave 4 (bottom). The black points indicate the prediction given by the emulator, with 95% error bars. The green and red points are the actual function values, coloured green if they lie within the 99% error bars around the prediction. Emulators are deemed to validate well if there are not too many or too few of the true values outside of these error bars.

It is likely that not all of the ensemble from the previous wave will have been ruled out. Therefore, after wave 1, the training set is the new ensemble (size n), with the emulator then validated using the runs from the previous wave that have not yet been ruled out.

3.4. Multi-wave history matching results

We now perform the experiment described in Section 3.3 for each of the toy functions in turn. The values for the observation, z, and the observation error variance used to calculate the implausibility are as given in Table 3.1. For example, for $f_1(\cdot)$, the true output of the function that parameter settings are required for is assumed to be $z = 15$. The error on the measurement of this observation is taken to have variance 10^{-5}. The noise that is added every time the function is evaluated is sampled from a Normal distribution with variance 0.05^2. The ensemble size sampled at each wave is $n = 200$ (the number of parameters multiplied by 20 here). Finally, if instead of building emulators, we evaluate $f_1(\cdot)$ instead, the size of the NROY space that fits with the above observations and error specification is 0.17% of the original space (for the same design, this size may vary slightly due to the internal variability, however we find this difference to be negligible). Therefore, this should be a lower bound on the size of the NROY space it is possible to find when

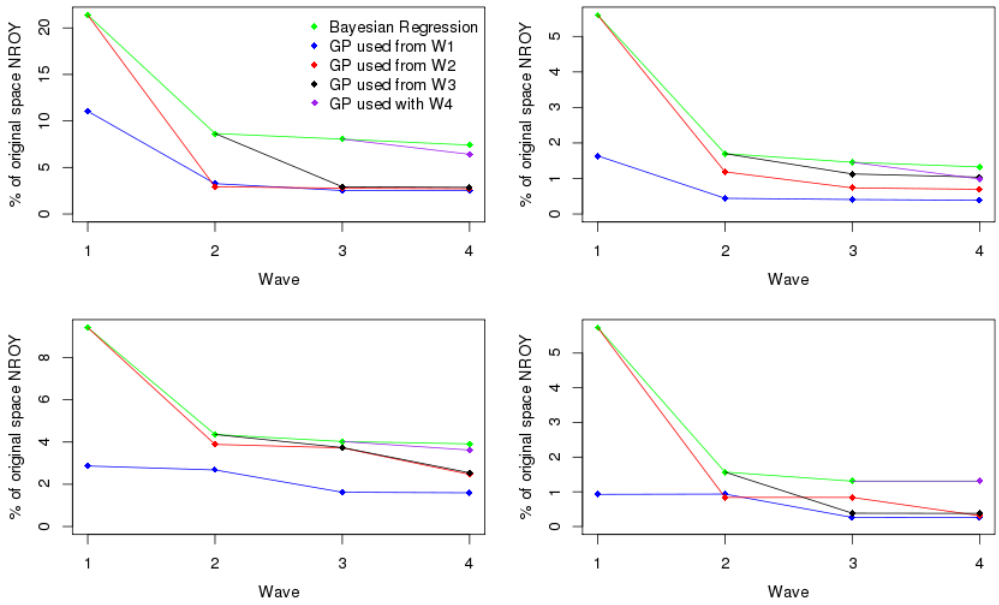

Figure 3.5. Top left function 1, top right function 2, bottom left function 3, bottom right borehole function. This picture shows the sizes of NROY space at each wave when history matching each of the toy functions with regression emulators, and when a Gaussian process emulator starts to be used at different waves.

emulators are used. If smaller spaces than this are found, then this suggests that the emulators are not accurate.

The information for each of the other functions can be read from the table in a similar manner. The magnitude of Var[e] has been varied slightly, as has the noise variance. However, these have been kept relatively small so that the true NROY space is a low percentage of $\mathcal{X}$, in order to represent real problems where it is difficult to find accurate parameter settings.

3.4.1. Size of NROY space

The resulting sizes (volumes) of the NROY spaces when we history match each of the four functions, for the various combinations of regression and Gaussian process emulators, are shown in Figure 3.5. The green line represents when a regression emulator is always used, and a blue line represents always using a Gaussian process. A new line is added at each wave to represent the new Gaussian process emulator starting from the regression NROY space at the previous wave. Some of the percentages associated with these plots are provided in Table 3.2.

For each function, it is clear that using Gaussian process emulators provides a large im-

Function	Regression	Gaussian process
f_1	7.410%	2.525%
f_2	1.326%	0.387%
f_3	3.913%	1.596%
f_4	1.308%	0.258%

Table 3.2. The size of NROY space (as a percentage of the original space) after wave 4, when only regression emulators have been used, and when a Gaussian process has always been used.

provement over only using regression emulators, in the majority of cases. At wave 1 for each function (i.e. the emulators are being fitted in the full high-dimensional parameter space), using a Gaussian process emulator rules out much more space than the regression, fitted using the same ensemble. Often in applications only a single wave will be possible, so this suggests that even in the full parameter space with a relatively sparse sample, a Gaussian process should be preferred. For example, for function 1, the regression emulator at wave 1 gives an NROY space that is 21.4% of $\mathcal{X}$. By instead fitting a Gaussian process, 10% more space can be ruled out, resulting in an NROY space consisting of 11.1% of $\mathcal{X}$. This is an important result, as it was expected that a polynomial surface would do well here; even for this type of function, using a Gaussian process is beneficial.

The results at wave 1 for the other functions give a similar story: for function 2, the NROY spaces have size 5.6% for the regression compared to 1.6% for the Gaussian process. For function 3, with the larger input space believed to not be as conducive to using Gaussian processes, the more complicated emulator reduced NROY space to 2.9% compared to 9.4% for the regression, suggesting that a Gaussian process emulator should be used even when there are 20 input parameters. For the borehole function ($f_4(\cdot)$), the wave 1 regression resulted in an NROY space that is 5.7% of $\mathcal{X}$, compared to 0.9% for the Gaussian process.

Table 3.2 shows the sizes of NROY space after wave 4 for each function, for the cases when only a regression is used, and for when only Gaussian processes are used. Similarly as for after one wave, the Gaussian process emulators have ruled out more space, and have managed to find spaces relatively close to the true NROY spaces. For example, after four waves of Gaussian process emulators for $f_2(\cdot)$, the NROY space is 0.39% of $\mathcal{X}$, compared to the true NROY space of 0.24%. Using only regressions in this case results in an NROY space that is still more than five times larger than the true space after four waves. For the borehole function, a similar improvement is observed: the wave 4 Gaussian process NROY space is slightly more than twice the size of the true space, whereas the wave 4 regression NROY space is more than 10 times larger than the true space.

While in these examples it is possible to efficiently evaluate predictions for four Gaussian process emulators, and find runs that are in NROY space, as discussed in Section 3.2.1, this may not always be computationally manageable. However, the results in Figure 3.5 show that favourable results can still be achieved if a regression is initially used. For these examples, it has not always been possible to find an NROY space that is extremely close to the 'always Gaussian process' case after four waves if a regression has been used at wave 1. For example, for functions 2 and 3, there is a clear separation between the blue line and any of the other NROY spaces at wave 4 (although for these, using a Gaussian process from wave 2 or 3 onwards still gives a reasonable improvement over only using regressions).

For function 1 and the borehole function, however, there is evidence of convergence between the blue, red and black lines at wave 4. That is, after four waves of history matching, the resulting size of NROY space is similar, regardless of whether the correlated residual term has been fitted at wave 1 or 2. This is an extremely beneficial property, as it allows computational savings to be made. Although this convergence has not been observed after four waves for each of the functions, the NROY space defined by using regressions for two or three waves, followed by Gaussian processes thereafter, is still substantially smaller than for the 'always regression' case. Therefore, if computing time is an issue, using regression emulators for the first couple of waves, then fitting Gaussian process emulators for the later waves, may be a reasonable compromise, in terms of minimising computation time while finding a reasonable NROY space.

In this setting, the regression is essentially used to capture global variation in the function output. Having removed parameter settings based on this, a Gaussian process emulator is then fitted in order to model the local variability in the (possibly small) NROY space. Intuitively, this makes sense: being able to focus an ensemble in a smaller NROY space increases the accuracy of the Gaussian process emulator due to the greater density of points. Therefore, the shrinking of the variance around the design points that the Gaussian process causes will have a more profound effect than in the full space.

3.4.2. Highlighting unusual results

In Figure 3.5, there is a discrepancy between the results and our expectation of what the results should be. Namely, using a Gaussian process emulator more often does not always result in a smaller NROY space.

For function 1, using a regression at wave 1 followed by a Gaussian process at wave 2 results in a smaller wave 2 NROY space than when a Gaussian process is used at both waves, although this difference is reasonably small so may be simply sampling error. Starting to use a Gaussian process emulator at wave 4 does not have as large an effect as might have been expected in this example, compared to the impact adding a Gaussian process had at the previous waves. However, more space is ruled out than by the regression emulator.

The results for function 3 also exhibit some unexpected behaviour. When a Gaussian process starts to be used from wave 2 onwards (red line), little improvement is made over the regression at wave 2. Then, at wave 3, only an additional 0.15% of $\mathcal{X}$ is ruled out, before a larger improvement is found after wave 4. There is a similar levelling off, followed by more space being ruled out, in the always Gaussian process case.

For the borehole function, fitting a Gaussian process emulator at wave 2, having also fitted one at wave 1, fails to rule out any extra space. Similarly, the red line (Gaussian process from wave 2 onwards) is flat between waves 2 and 3, before a later improvement. Adding a Gaussian process at wave 4 after three waves of regression emulators also has no effect, and neither does the regression emulator at this wave.

For function 3 and the borehole function, not being able to rule out more space in certain circumstances cannot be because all points in the current NROY space are deemed close enough to the observation: the sizes of these NROY spaces are not equal or close to the true NROY percentage. Furthermore, improvements have been made after the size of NROY space seemingly converged in the experiment. This premature convergence is a problem, because in applications the true NROY space is unlikely to be known, therefore results such as the lack of improvement made by the Gaussian process emulator at wave 2 for the borehole function may lead to the incorrect conclusion that space cannot be reduced any further (although if the average emulator variance is larger than the other variances, then we can conclude that it should be possible to remove more space). These

unexpected results are explored further in Section 3.5.

3.4.3. Composition of NROY space

Simply ruling out more space may not be desirable if the emulators are incorrectly ruling out points that are in fact close to the observations, or are leaving regions of space that give output far from the observations. Therefore, we also consider the composition of the various NROY spaces from the above experiments.

To do this, we sample from the NROY spaces as defined above. The true output of the function is calculated for each sampled point $\mathbf{x}$, and a weighted density of the function output is calculated, with $e^{-\mathcal{I}(\mathbf{x})}$ used as the weight. This weighting is used as it ensures that as the implausibility decreases (i.e. it is more likely that this point leads to output consistent with z), the weight attributed to this point increases. If the emulators above have resulted in low implausibilities for points that in fact give output far from z, this will be reflected by greater weight being assigned to this output in the weighted density, and the density will be skewed away from z, leading to the conclusion that the emulators are not suitable.

Another reason for using this weighting is that this is analogous to Bayesian calibration. If a uniform prior is assumed for the best input $\mathbf{x}^*$, and using the Normality assumptions of Kennedy and O'Hagan (2001a), then the likelihood of z is $e^{-\mathcal{I}(\mathbf{x})}$ in NROY space. Assigning zero likelihood to points that have been ruled out at previous waves, the re-weighted sample is then a sample from the posterior distribution $\pi(\mathbf{x}^*|z, \mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \mathbf{F}^{(3)}, \mathbf{F}^{(4)})$.

The weighted densities for some of the NROY spaces for the toy functions are shown in Figure 3.6. The left-hand panels give these densities for the wave 1 NROY spaces defined by the regression emulator and the Gaussian process emulator, for each function in turn. The right-hand panels give the equivalent wave 4 NROY spaces, for the always regression and always Gaussian process cases. In every case, using a Gaussian process emulator has reduced the parameter uncertainty: the spread of outputs in the resulting NROY space has decreased compared to when regressions are used. Using a Gaussian process emulator has allowed more extreme values, far from z (indicated by the red line), to be ruled out for each of the four toy functions, whether only one or four waves have been carried out.

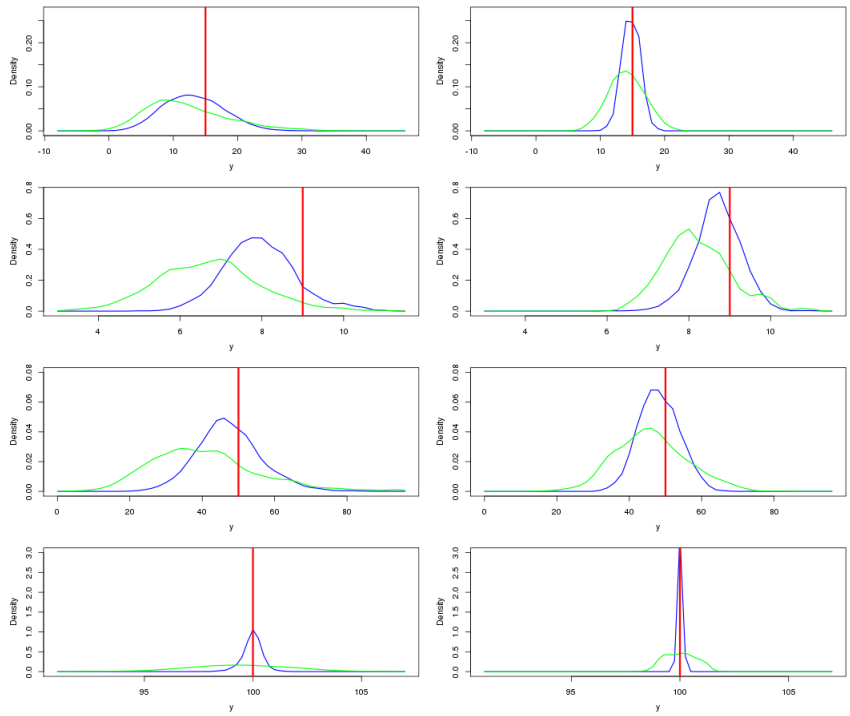

Figure 3.6. The weighted densities for the function output at points in NROY space after wave 1 (left) and wave 4 (right) for each of the four functions, for the 'always Gaussian process' case (blue) and the regression emulators (green). The observation for each function is given by the red line.

Performing more waves of history matching should lead to a reduction in the spread of outputs in NROY space, as the emulators should be improving at each wave, and this is shown to be the case here. However, a more interesting result from these densities is that for each function except $f_1(\cdot)$, there is less parameter uncertainty after one wave when a Gaussian process is used, compared to when four waves have been performed using regression emulators. This provides a further reason for preferring Gaussian processes, even if only one wave of calibration or history matching is possible.

To more formally quantify this difference in the spread of outputs for each emulator, we consider the variance. For example, for function 1, after wave 1, the variance of outputs in NROY space is 43.45 for the regression emulator, compared to 26.98 for the Gaussian process. After four waves of history matching, the regression approach has reduced this variance to 11.19. However, using Gaussian processes results in a variance of 2.38. This reduction is observed for each of the other functions, although the weighted densities in Figure 3.6 are perhaps a better illustration of the changes.

In addition to being superior at ruling out extreme values, using Gaussian process emulators results in another desirable property: in every plot in Figure 3.6, there is more weight

around the true observation in the Gaussian process NROY space, whether after one or four waves. For function 1, after the first wave, the weighted density for the regression is skewed away from z. The density for the Gaussian process has a similar shape, but is much closer to z, although the peak of this density is slightly less than z. After four waves, the density for the regression is almost centred on z. The Gaussian process NROY space, however, assigns far more weight around the observation. For this function, it is clear that using a Gaussian process is superior, as the density assigns more weight to z, and contains fewer extreme values, at waves 1 and 4.

We expected that the regression emulators would perform best for function 1. Given that the Gaussian process emulators have been found to be superior in this case, it is not surprising that similar conclusions are reached for each of the other toy functions. For function 2, both densities exhibit some bias away from z for each wave. However, the Gaussian process NROY space is much closer to z for each wave, and performing more waves moves the peak of the density closer to z, so that the wave 4 Gaussian process NROY space assigns a large weight to z (although this value does not receive the greatest weight).

For the 20-dimensional function, $f_3(\cdot)$, the wave 1 regression density is fairly smooth, containing a wide range of values, with the mode of the distribution reasonably far from z. The Gaussian process improves this, with more weight around z at both wave 1 and wave 4 (although there is a slight skew in both cases).

For the borehole function, the difference between the two emulator types is most pronounced. After the first wave, the majority of the density for the Gaussian process NROY space is around the observation, with the peak of this distribution at z. The regression emulator leads to a much smoother density, with similar weights assigned for a far greater range of outputs. By wave 4, the regression NROY space has more weight at z (although this is still less than the wave 1 Gaussian process has), but the Gaussian process NROY space has a more concentrated peak at z than before, with the majority of the density falling within 0.25 of z.

In general, by fitting a Gaussian process emulator at the first wave, it has not only been possible to remove extreme outputs more efficiently, but more weight has been assigned to the regions of parameter space that in fact lead to output that is closer to the true

output. If this were a real application, both of these would have an effect on inferences made about the real system: if new samples were required in the existing NROY space, if the space of possible outputs is smaller, and more focussed around z, then it is more likely that runs of the computer model that are similar to z can be found. This in turn leads to more accurate emulation of the model behaviour in this important region of $\mathcal{X}$.

3.5. Sensitivity to sample design

In the previous section, whilst fitting a Gaussian process generally leads to a smaller NROY space, and one with superior properties to that defined by regression-only emulators, Figure 3.5 also highlighted some unexpected results, involving premature convergence of the size of NROY space, as highlighted in Section 3.4.2. This is not only an issue for Gaussian process emulators, as this is also observed in some of the regression-only cases.

We focus on the experiment for the borehole function here. One of the problems observed for this experiment is that when a Gaussian process is used from wave 2 onwards, there is little difference between the wave 2 and wave 3 NROY spaces, before an improvement is made at wave 4, showing that the levelling off after wave 2 does not imply that it is no longer possible to rule out any further parts of parameter space. Related to this, using a regression for two waves, followed by a Gaussian process at wave 3, resulted in a smaller NROY space than one regression followed by two Gaussian processes. Furthermore, having used regressions for three waves, neither emulator ruled out any more space at wave 4.

There are a number of possible reasons for no additional space being ruled out, given that it is clear that the true NROY space has not yet been identified, so that more space could theoretically be ruled out using history matching. The mean function that is selected for the regression or Gaussian process emulator, and hence the estimates of these coefficients, may not be a good fit. Although each emulator has passed validation checks, there may be a lot of uncertainty on some of these emulators, resulting in the majority of predictions lying within 95% uncertainty bounds, but in reality it not being possible to rule out any parameter settings because the uncertainty bounds also include z. This may be difficult to fix, as the mean function is selected using a stepwise regression approach, and choosing which linear terms and interactions to include by hand is not possible for large numbers

of parameters. Additionally, because the true functions are known, it would be possible to simply select the interactions that are in the function definitions, but this would not give an experiment that is representative of applications.

Another potential reason could be due to the fitting of the correlation length and nugget parameters, in the case of Gaussian process emulators. As before, although certain settings of these may validate well, they may have large uncertainty associated with predictions. Selecting these parameters differently may result in superior emulators.

The emulator that is fitted is also dependent on the ensemble that it is fitted with. The choice of the ensemble directly affects the subsequent choices of the mean function and correlation function parameters, and there may be situations where the ensemble does not allow a suitable, representative emulator to be fitted, whether regression or a Gaussian process is used. Whilst it is generally assumed that the model output can be represented by a polynomial surface or a Gaussian process, only having a sparse sample of $\mathcal{X}$ available may lead to a different emulator than if the available ensemble size is much larger. For example, there may be a region of $\mathcal{X}$ that has different behaviour from the rest of space, or where a certain input parameter is active (and inactive elsewhere), and hence having a point in this region is important if an accurate emulator for all of $\mathcal{X}$ is to be built. If this region is small, then not every Latin hypercube sample of $\mathcal{X}$ will contain a point in this region, and emulation may not be accurate there.

3.5.1. Refitting emulators

In order to improve the results of history matching for the borehole function, we now refit the emulators that gave little or no improvement to the size of NROY space, with a focus on the wave 3 Gaussian process, for the case where a regression is used only at wave 1.

First, using the current ensemble from this experiment, we refit the correlation lengths and nugget, by applying the algorithm in Section 3.3.3 multiple times from different starting points. Using these refitted emulators, we perform history matching again, and compare the size of NROY space to that previously defined. However, by simply re-estimating the correlation parameters, no extra input parameter settings can be ruled out. Ignoring the algorithm and attempting to fit the parameters using other methods, for example

maximum likelihood, or by tuning these by hand, was also unable to offer any improvement on the originally fitted emulator.

To investigate the problem, we take additional samples from the current NROY space, giving a range of ensembles, from which new emulators are fitted, and a new history match is carried out, giving a range of sizes of potential NROY spaces. Some of these new emulators do allow a smaller NROY space to be found at wave 3, suggesting that it may be the original ensemble that is the problem. The 'best' emulator (i.e. the one that leads to the smallest NROY space), with newly estimated mean function, correlation lengths and nugget, is fitted to the ensemble from the original experiment.

Fitting this new emulator, and using this for history matching, leads to a smaller NROY space than originally found, and one that is similar in size to the NROY space found using the 'best' emulator. This suggests that the problem is in fitting the emulator: it is important to be able to select the correct mean function and correlation parameters, although this was not possible with the original ensemble, and these could only be identified using a completely different sample. Since with a poor mean function, it was not possible to find correlation lengths that resulted in a smaller NROY space, this suggests that if the mean function is chosen well, then reasonable correlation lengths can also be found: if the mean function is a good fit, then there is less variability to be explained by the residual, and it is generally more straight-forward to fit a Gaussian process.

Given that it has now been shown that it is possible to find a smaller NROY space using the original ensemble, we now consider an alternative method for fitting the mean function based on this ensemble. In the automatic approach for fitting emulators in this experiment, the mean function was selected using a stepwise approach with a noise variable added, to guard against overfitting. However, there is a possibility that this variable could be correlated with other input variables, leading to premature selection of the noise variable, and hence termination of the process. Using different samples from the Normal distribution, different mean functions can be fitted, for the same data.

To account for this potential problem, we take 10 separate samples of the noise, and fit a new mean function for each. The best mean function from these is then selected, in the sense that the most variation in the ensemble is explained. However, all of these resulted in poor fits, with the same inadequate mean functions selected commonly. The best fitting

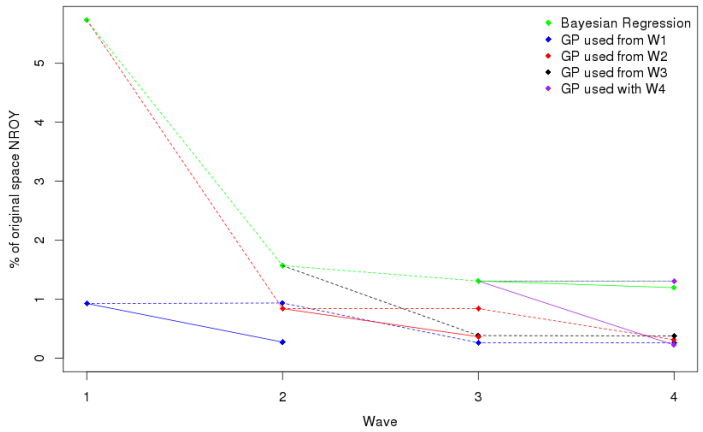

Figure 3.7. The progression of the sizes of NROY space for the borehole function. The dotted lines indicate the original NROY spaces found, as in Figure 3.5, with the solid lines showing improvements achieved through either fitting a new mean function (in the case of GP2 (red line)), or by taking a new sample in the existing NROY space.

mean function from this sample contained only seven predictor terms, with only three of the eight input parameters highlighted as active, leading to an emulator with high variance and little ability to rule out parameter settings.

By taking further samples of the noise, and fitting new mean functions for the same ensemble, we eventually select a better mean function, although the majority of new mean functions we found gave poor fits. This better function contains more of the borehole function parameters, and is similar to the one selected by using a different ensemble. Fitting new correlation parameters for the Gaussian process with this mean function, the new emulator leads to a wave 3 NROY space that is 0.36% of $\mathcal{X}$, compared to 0.84% in the original experiment. This is much more in line with what would be expected, and is very similar to the NROY space found at wave 4 in the original experiment: by refitting the emulator, it has been possible to achieve the same results after three waves that originally required four.

Figure 3.7 shows the original results for the borehole function, with this new emulator added. However, it has not been possible to fix the other unexpected results highlighted previously by fitting alternative mean functions to the original ensembles. For the wave 2 emulator for the 'always Gaussian process' case, and for when a regression is used for the first three waves, re-sampling the noise variable and fitting new mean functions did not offer any improvement over the original emulators. By taking a new sample in the current NROY space, and fitting new emulators using this ensemble, both of these results can in fact be improved, and these are also shown in Figure 3.7. The results here are now more

in line with the other functions.

For the always Gaussian process case, instead of there being little change between waves 1 and 2, a better emulator allows a reduction in the size of NROY space to be made at wave 2. Now, the size of NROY space after two waves is similar to the size after four waves in the original experiment, therefore allowing computational savings if an accurate emulator can be fitted. Similarly, for the wave 3 regression NROY space, the new regression is unable to rule out much more space, but the Gaussian process emulator is far more accurate, and converges to a similar-sized space as for the other experiments, reinforcing the conclusion that originally fitting regressions is acceptable.

These results demonstrate that selection of the mean function is important, and that it is sometimes possible to improve this by taking multiple samples of the noise variable, and fitting different mean functions based on this. The automatic method used in the experiment above can be simply adjusted to allow multiple mean functions to be fitted, and the best of these can be used to fit the final emulator. This fixes some of the unusual results.

However, fitting the mean function well is completely dependent on the ensemble available, as no other prior information is assumed, given that the toy functions are treated as unknown. Therefore, the problem becomes one of sample design: when a different sample is selected from NROY space, and used to create the ensemble, it is possible to fit a superior mean function, and hence emulator, which appears to be more accurate across the whole NROY space. In this experiment, it is possible to simply discard the original ensemble and take a new sample from NROY space as the toy functions can be run instantaneously. This is not possible when model runs are expensive, hence the sample design problem is an important one.

3.6. Application to an environmental model

Having established these ideas using toy examples, we now verify whether they hold when an actual physical model is used for history matching.

The IC fault model is a cross-sectional model of a reservoir, with three unknown parame-

ters: h, the fault throw; k_g, the good-quality sand permeability; and k_h, the poor-quality sand permeability (Tavassoli et al., 2004, 2005). This model is known to be difficult to calibrate accurately, hence investigating whether history matching is suitable, alongside performing the same experiment as for the toy functions, is an interesting question. This also provides a good example of the types of environmental models that history matching is used for, as there are multiple outputs for each parameter choice: the model returns a monthly time series for 36 months of the oil production rate o, the water injection rate w, and the water cut (or production) rate c.

Parameter space having only 3 dimensions, and the output being deterministic (no internal variability) does not correspond as well with typical climate applications. However, the IC fault model is used for this study because it is a calibration problem that has been solved by 'brute force', with a database of 159,661 runs at different parameter choices. Therefore, it is now a test bed for efficient methods, as the solution space has interesting features.

Using this model, and the large database of known runs, removes the need to run a time-consuming model, which would be prohibitively expensive to implement for the same experiment performed for the toy examples. Instead, samples from this database can be drawn to create ensembles. Given that there are a large number of model runs, emulation and history matching would be more accurate if more of these runs were used. However, in typical applications, there will not be access to a database of runs such as this, therefore small samples are taken here. The methodology of Gramacy et al. (2015), where Gaussian process emulators are fitted locally to a large ensemble of model runs, may in reality lead to a more accurate calibration exercise, if this were the only goal here.

The observations that we will match to are given in Figure 3.8. The water injection and oil production values remain relatively constant over the 36 months, with some fluctuations, and a drop-off in the oil production from month 27 onwards. Around the same time as this, the water production rate begins to increase from zero.

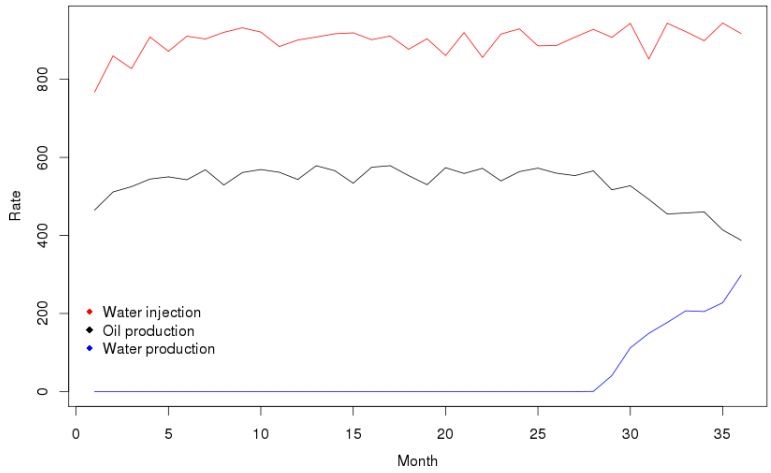

Figure 3.8. The observations for the IC fault model.

3.6.1. Summary statistics

Given that the model output here is multivariate, in the form of three separate time series, it would be possible to emulate the entire series using one of the multivariate emulation techniques from Section 2.4. However, in order for a comparison to be made with the results obtained for the toy examples (Section 3.4), we define summary statistics of the output, and emulate these instead of the full output. There are strong correlations between certain outputs, so that history matching using a limited number of these may give similar results to using all of the outputs.

It would be possible to, for example, average across each of the time series and build emulators for this summary. However, this ignores any trends, and runs in NROY space may not appear similar to the observations when plotted. Therefore, we select the following three summary statistics:

- o_{24}, the oil production rate in month 24.
- o_{36}, the oil production rate in month 36.
- w_{36}, the water injection rate in month 36.

The first two of these are designed to capture the decline in the oil production rate over the final 12 months of the observations. w_{36} is included so that runs with the correct magnitude of the water injection rate after 36 months can be found. The water production rate is ignored in this experiment. In the ensemble of model runs, the time where the increase

from zero starts varies. Hence, this time may be difficult to emulate using standard emulation techniques, and the results here may not be directly comparable to the toy example conclusions.

The observed values for these statistics are given by the following vector:

$$\mathbf{z} = (563.6, 387.5, 917.2)^T$$

In order to history match, the variance of the observation error is also required. Neither this nor the discrepancy variance are known, and hence they are combined in this problem. This combined variance is set equal to 1 for each output.

Previously, we were only matching to one observation, therefore using the univariate definition of the implausibility, and ruling out points where this implausibility was less than 3, was possible. Here, because we build independent emulators for each of the three statistics, there will be three implausibility values for every $\mathbf{x}$. Therefore, the second maximum implausibility measure (Craig et al., 1997) is used to define NROY space:

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

where

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

with parameter settings not ruled out if $\mathcal{I}_{2M}(\mathbf{x}) < 3$.

3.6.2. Sampling from the database

The other change that is required from the toy example experiment is that it is not possible to run the IC fault model at any parameter setting. However, using the database of 159,661 runs, ensembles can be created, with the ensemble size n set equal to 60.

To create the ensemble used to fit the wave 1 emulators, we divide parameter space along each dimension into 60 sections of equal length, so that $\mathcal{X}$ is divided into 60^3 cubes. Then, similarly to Latin hypercube sampling, we select 60 of these regions, so that when projected onto each individual dimension, each of the 60 sections contains exactly one point. For each of the 60 selected regions in 3-dimensional space, a run is sampled from

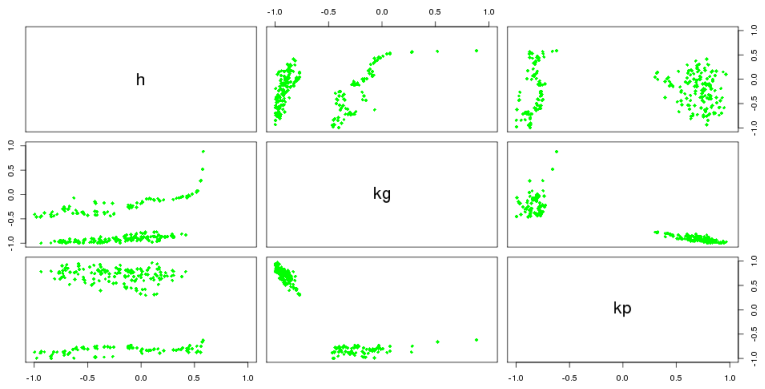

Figure 3.9. The parameter settings for the runs in the database that are in NROY space according to the three chosen statistics.

the database. This gives an original sample that is a Latin hypercube, using the constraint that only certain parameter settings are available.

At later waves, to create new ensembles, we draw samples from the database of runs, and evaluate the relevant emulators at each point to determine whether that run is in the current NROY space. If it lies in NROY space, then this run is added to the new ensemble.

3.6.3. True NROY space

Since there is a large database of model runs, using the observations and variance to be used in history matching, we calculate the implausibility for each of the runs in the database, with no emulator variance. This can be used to identify whether the model can actually reproduce the observations, and the size of the true NROY space can be estimated.

The runs that are in NROY space for the chosen statistics, using the second maximum implausibility measure and a cutoff equal to 3, are shown in Figure 3.9. From this picture, it is clear that the true NROY space is split into two disjoint regions of $\mathcal{X}$. For each pairwise parameter plot, there is a separation between two groups of possible optimal values. This property of the true NROY space may make it challenging to find the exact space with the standard emulation techniques used in this study, although this may not be a problem until after several waves have been performed and space has already been constrained substantially. This true NROY space consists of 0.14% of $\mathcal{X}$.

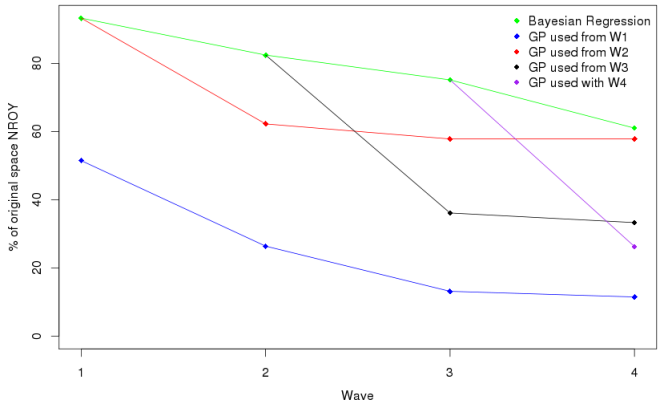

Figure 3.10. The progression of the size of NROY space when history matching the IC fault model with only regression emulators, and when starting to use a Gaussian process emulator at different waves.

3.6.4. History matching results

Similarly as for the toy functions, we now perform a four wave history matching experiment for the IC fault model, with the required adjustments in the method as described above. The results of this experiment are shown in Figure 3.10.

The main conclusion from these results is the same as before: the Gaussian process cases all outperform the regression-only case, generally by a large percentage. When a regression emulator is used at every wave, the resulting NROY space is still large, with 61% of $\mathcal{X}$ in this space. At the other extreme, when a Gaussian process emulator is always used, an NROY space containing 11.5% of the original parameter space is found. The other three cases give results between these two NROY spaces. Using a regression for the first three waves followed by a Gaussian process leads to an NROY space with 26% of $\mathcal{X}$, while using a regression for two waves followed by two waves of Gaussian process emulators results in 33% of $\mathcal{X}$ not being ruled out.

However, the same problem observed previously is shown again here: for the case where a Gaussian process is used from wave 2 onwards, it has not been possible to rule out much additional space after wave 2. After 2 waves of history matching here, NROY space is 62% of $\mathcal{X}$, while after 4 waves it is still 58%. This suggests the importance of the sample design once again. Despite this problem, the resulting NROY space after 4 waves is still smaller than that when regression emulators are always used. As well as for the reasons discussed previously, the sensitivity to sampling for this model may be due to the disjoint nature of the true NROY space, perhaps leading to a difficulty in building accurate emulators,

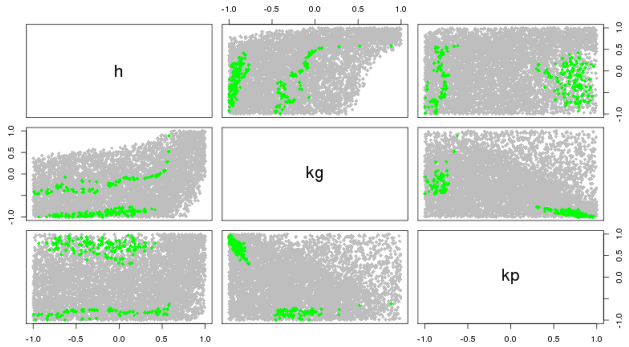

Figure 3.11. A parameter plot showing the true NROY space (green) and points classified as being in NROY space after 4 waves when regression emulators are used at each wave.

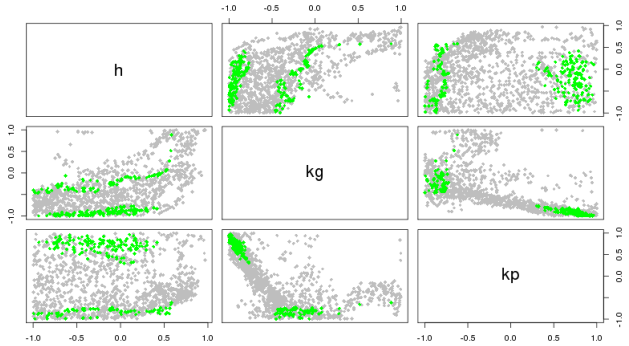

Figure 3.12. A parameter plot showing the true NROY space (green) and points classified as being in NROY space after 4 waves when Gaussian process emulators are used at each wave.

unless points in certain informative regions of space are observed.

As before, the size of the resulting NROY space is not the only important result. Pairwise plots of the parameters for points in the wave 4 NROY space where only regression emulators have been used are shown in Figure 3.11, with the equivalent plot for when Gaussian process emulators are always used in Figure 3.12. The runs in the true NROY space are overlaid in green for each of these.

For the regression emulators, only 39% of parameter space has been ruled out after four waves, and this is highlighted by the pairwise parameter plot, with few regions of space where there are no runs. The majority of runs that have been ruled out are on the edges of parameter space. For example, when k_g is high (greater than 0.5) and h is less than 0.6, all runs have been ruled out. The sections of space where the true NROY runs are located have not been ruled out, however it has not been possible to rule out any of the space between the two disjoint regions of the true NROY space. If there was no access to the large database of runs, and conclusions were to be made from only the four waves of history matching using regression emulators, it would not be possible to infer much about the true shape of NROY space.

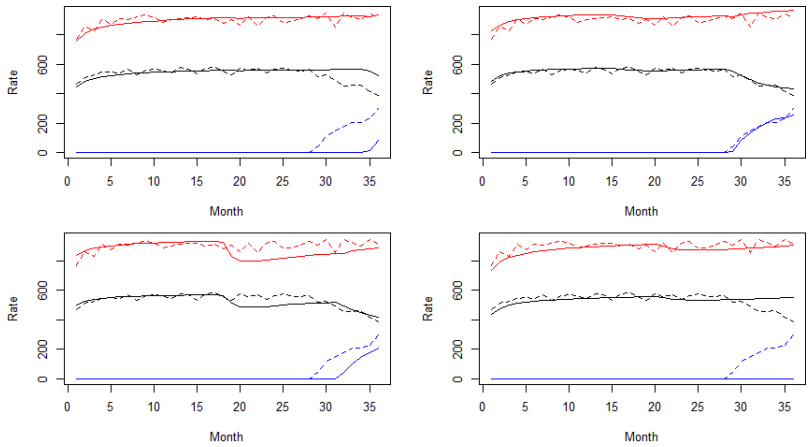

Figure 3.13. The output for four runs in NROY space, when a Gaussian process emulator is always used, with the solid lines giving the model output for a particular parameter choice, and the dotted lines showing the observations. Red is the water injection rate, black is the oil production rate, and blue is the water cut.

Figure 3.12 illustrates how Gaussian process emulation has improved the final NROY space. Although the separation between the two regions of the true NROY space has not completely been identified, a lot more of the structure of this space can be seen in the pairwise plots. For example, there is a clear relationship between k_g and k_p, with a linear section of space linking the two true NROY regions. For the other plots, the true runs lie towards the edge of this wave 4 NROY space, with some runs lying in between. As for the regression case, all of the true NROY runs lie in this NROY space.

Comparing these two cases shows that always using a Gaussian process emulator has ruled out significantly more space than the regression-only history matching, and has done so as accurately. The true structure of NROY space has also started to be highlighted by the Gaussian process NROY space, whereas using regressions has been unable to rule out much space.

Some of the runs in the wave 4 NROY space defined by always using a Gaussian process are shown in Figure 3.13, selected by random sampling from this NROY space. The first run (top left) matches the water injection rate well for the whole 36 months, although the oil production rate stays relatively constant over the last year, a departure from the observations, where this rate decreases. This run is however in the current NROY space because of the decision to conservatively rule out runs with the second maximum implausibility measure, so that o_{36} does not necessarily need to match the observed value. This run also does not have a good match for the water production rate, as it only begins to increase from zero around month 35.

The second sampled run (top right) is a far superior match for $\mathbf{z}$, and despite the fact that the water production rate was not explicitly involved in history matching, this run is close to $\mathbf{z}$ for each time series. The third run (bottom left) slightly underestimates the water injection rate, but does capture the decrease in oil production adequately, and has a similar magnitude to the observed increase in water production, albeit a couple of months too late. Finally, the fourth run looks similar to the first, with the difference that the water production never increases above zero.

3.6.5. Calibration

As for the toy functions (Section 3.4.3), we calibrate based on the NROY spaces defined by the regression and Gaussian process emulators after waves 1 and 4, assigning a uniform prior in NROY space for $\mathbf{x}^*$. Figure 3.14 shows the densities for each of the outputs that have been history matched to, with the vertical red lines indicating the observed value of each. These were calculated by sampling 1000 points from the database for each NROY space.

For o_{24}, the density for the wave 1 NROY space, defined by either the regression or Gaussian process emulator, is centred at the true value of this output. The Gaussian process emulator assigns more density to this true value. After four waves of history matching, always using regression emulators has not offered a large improvement; the density here is similar to the wave 1 Gaussian process density. Using Gaussian process emulators for four waves has narrowed the density, with a larger peak at the true value.

The results for w_{36} (bottom plots) are similar, with the wave 1 densities centred at the observed value, with a higher peak for the Gaussian process NROY space. After wave 4, the regression has improved, but the Gaussian process NROY space is again far superior, with more extreme values removed, and more density assigned to the truth.

For o_{36}, the results are less conclusive. There is not a large improvement between waves, and it is not clear that the Gaussian process is superior. A potential cause of this is that NROY space has been defined using the second maximum implausibility measure, so that o_{36} is allowed to be inaccurate if the other two outputs match the observations. For example, half of the 1000 runs in NROY space used for the wave 4 Gaussian process plot

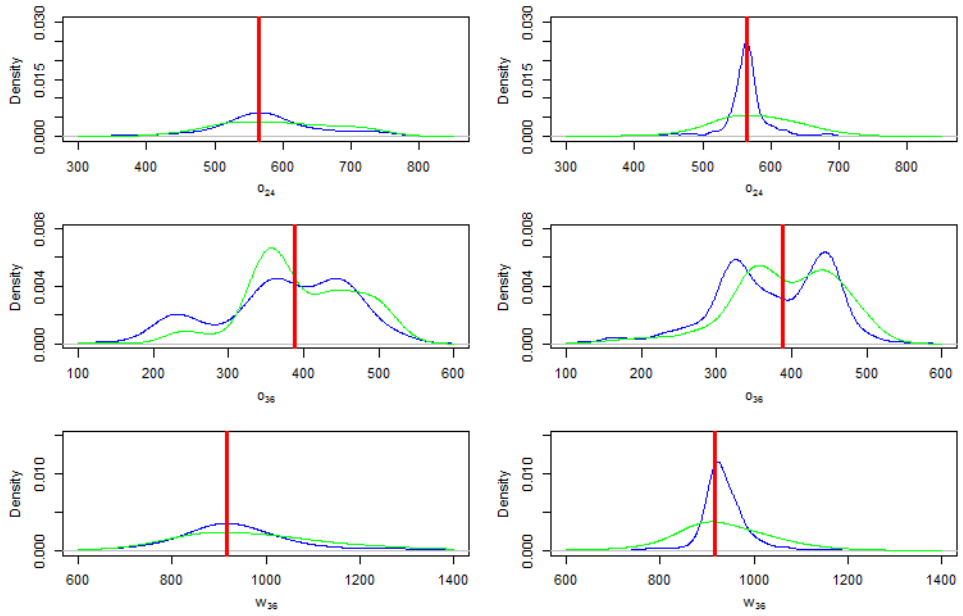

Figure 3.14. The weighted densities for the function output at points in NROY space after wave 1 (left) and wave 4 (right) for the output statistics, o_{24}, o_{36} and w_{36}, for the 'always Gaussian process' case (blue) and the regression emulators (green). The observation for each of the outputs is given by the red line.

have implausibilities greater than 3 for o_{36}, so that these would be ruled out if we only matched for this output. These runs have less density assigned to them, however. This result may be highlighting that, generally, accurate values for o_{24} and w_{36} can be found, but doing so has an adverse effect on o_{36}. A further wave could focus on ruling out runs based on o_{36} only.

Overall, the conclusions made from these densities are similar to those found from the toy functions: the Gaussian process emulator outperforms the regression emulators, and in fact the wave 1 calibration given by the Gaussian process emulator is similar or superior to the calibration given by the regression emulators after four waves of history matching. Performing additional waves results in a greater accuracy of calibration, particularly for Gaussian process emulators.

3.7. Discussion

In this chapter, the statistical emulation techniques of regression and Gaussian processes have been compared for multi-wave history matching. Using two 10-dimensional toy functions, a 20-dimensional function, and the standard test problem, the borehole function, various combinations of emulators have been used to history match the functions. The

same experiment was then performed for an environmental model, with the caveat that there were only three input parameters, resulting in a substantially smaller input space than for the earlier examples.

The results above suggest that it is beneficial to fit a Gaussian process emulator when history matching, either for a single wave or multi-wave experiment. When performing a single wave, we found that using a Gaussian process offers a significant improvement over a regression emulator, both in terms of the size and the composition of the resulting NROY space. In every example, a smaller NROY space was found using a Gaussian process emulator, and sampling these NROY spaces showed that more density was distributed around the observation than for the NROY spaces defined by regressions. This remained the case after multiple waves of history matching were performed, with a single wave of Gaussian process history matching commonly outperforming four waves of regression emulators.

Finding a smaller, and more accurate, NROY space, will have a substantial impact on any inference made about the real system that the model represents. Performing a single wave has always been the practice in Bayesian calibration-only analyses (Kennedy and O'Hagan, 2001a, Rougier, 2007, Higdon et al., 2008b, Sexton et al., 2011), so using a Gaussian process in these applications (as advocated by Kennedy and O'Hagan (2001a)) rather than regression (e.g. in Sexton et al. (2011)) should allow the calibration to be improved.

The results in this chapter also suggest that history matching should be performed over multiple waves, where resources allow this. How to allocate resources is an interesting question: if, for example, a total of n runs of the model can be obtained, how best to allocate these accordingly across waves is not obvious. Here it has been assumed that a fixed number of new runs can be sampled at every wave, but this may not be the most efficient way to rule out space.

In many applications, particularly in climate, regression-only emulators are used (Rougier et al., 2009, Sexton et al., 2011, Holden et al., 2013, Williamson et al., 2013), with the expectation that fitting a Gaussian process will make little difference to the calibration or history matching to be performed, due to the huge parameter spaces and small sample sizes. However, the results in this chapter suggest that even in this setting, the cumula-

tive effect of the variance shrinkage around these sparse points can be enough to have a significant, and lasting, effect on the analysis. Over multiple waves, using a Gaussian process often resulted in a large improvement over the case where only regressions were used, and in some of the examples, performing more waves of regression did not overcome the difference made by fitting Gaussian processes at the first couple of waves. However, due to the longer computational times associated with Gaussian process emulators, and the large number of emulators or waves that can be required (Section 3.2.1), there is perhaps some trade-off to be found between the two emulator types (Andrianakis et al., 2017).

When it is possible to carry out history matching in multiple waves, it may be acceptable to fit a regression emulator at wave 1. This allows space to be ruled out based on global behaviour initially, before fitting a Gaussian process at later waves allows local behaviour to be modelled more accurately. This methodology is shown to be a reasonable approach by the above experiments, as some convergence is observed between the size and composition of NROY spaces when regression emulators are used for a couple of waves, and when Gaussian processes are always used. This would allow computational time to be saved thanks to fewer Gaussian process evaluations being required to determine whether millions of points in $\mathcal{X}$ are in NROY space. This would also reduce the time required to fit the emulator initially, as no correlation parameters would need to be estimated, which can be difficult and time-consuming.

For the IC fault model, additional outputs were added so that the experiment more accurately reflected a typical history matching exercise in this field. In general, the experiment performed for this model agreed with the conclusion from the toy experiment: if multiple waves are to be performed, then using a regression initially followed by Gaussian processes later on is acceptable. Both the toy experiment and IC fault experiment suggest that fitting emulators, and hence the sample design in NROY space, is important.

It may be expected that adding a correlated residual term can make up for a poor choice of mean function, and indeed, this is part of the reason that many papers advocate the use of a constant mean for the Gaussian process (Sacks et al., 1989b, Chen et al., 2016). The fact that this was not found to be the case in numerous applications of the method should be troubling. A possible reason for this may be that the assumption of a weakly stationary process being a good approximation for the functions was not valid, and different emulator types should be used (although the emulators all passed the standard validation tests, and

the majority of results were as expected).

By taking more time to fit the emulators, some of the unexpected results relating to a premature convergence in the size of NROY space are improved. However, it appears possible to have an ensemble where it is not possible to fit a good emulator, while this can be overcome if an alternative ensemble is sampled. This suggests that a good sample design is crucial, as this directly affects the accuracy of the emulator that can be fitted. When new ensembles were sampled, it was often much more straight-forward to fit an accurate emulator, and the first mean function selected using the stepwise approach would allow accurate correlation lengths to be estimated, and result in a reduction in the size of NROY space.

It is not clear what constitutes a better sample design in this case. In this experiment, NROY spaces were sampled from by successively taking Latin hypercube samples until enough NROY runs were found. The aim of this was to ensure a reasonable coverage of the NROY space, however the fact that repeating this procedure can lead to extremely different emulators and results suggests that simply spreading out the available design points is not enough. It may be that there are (possibly small) regions of the current NROY space where it is more important to have design points, perhaps because the output varies more in that region. Therefore, samples where this region was highlighted allowed more space to be ruled out, compared to ensembles that missed this part of space, because emulation was more accurate there. We provide a different method for designing an ensemble in NROY space in Section 5.6.1.

The location of these important regions of space may be difficult to detect without any prior knowledge. It may simply be enough to change the design methodology, for example after wave 1 by placing more points where there is larger uncertainty in the current emulator. However, although the wave 1 emulators in the toy function experiment all performed well, there is no reason why a general wave 1 space-filling sample guarantees the ability to fit the correct mean function. This is troubling, as often there will be no prior knowledge as to which regions of high-dimensional space the small number of design points should be placed in, so a space-filling design is generally preferred.

It is not inconceivable that sometimes it will not be possible to improve perceived anomalous results, regardless of the sample, perhaps due to the limited number of runs in high-

dimensional space, and this may be a problem for both regression and Gaussian process emulators. However, it is hoped that this will very rarely be the case, and that when a single computer model is being history matched, and only one emulator needs to be fitted, more time can be allocated to ensuring an informative ensemble design, and careful fitting of the emulator.

Poor emulator performance may also be observed if the NROY space that the emulator is built to be accurate in is not connected. In general, it may be extremely challenging to identify that the true space consists of multiple separate regions of space, due to the high-dimensional nature of the space, and may lead to one poor emulator being fitted (e.g. because high leverage points were used when fitting the mean function), whereas a separate emulator for each disjoint part of NROY space may be more appropriate.

This has perhaps caused an issue with the IC fault model. For this model, it is known that the true space where the model is consistent with the observations is disjoint. In order to perform a more accurate history matching that may be able to rule out more space, fitting separate emulators for the two different parts of parameter space may be necessary. Careful consideration of the design in this complicated NROY space would be required. For example, how these two sets should be defined needs to be considered: in the final Gaussian process NROY space in Figure 3.12, parts of the space between the two disjoint sets have not been ruled out. Even given the knowledge that the resulting space will be disjoint, it is not clear over which subregions of NROY space the emulators should be defined. For points between the two disjoint NROY spaces for the IC fault model, should classification be simply done using Euclidean distance, or using a more complicated measure, e.g. by placing points in the most important parts of NROY space, and identifying which parts result in similar model behaviour, by some metric.

Another way to improve the IC fault model history matching would be to emulate and match using more of the outputs, or to emulate the entire time series using any multivariate emulation approach. This should improve the match between the observations and runs in the final NROY space, as rather than runs only needing to be within the tolerance to error at (at least two of) three outputs, this would be required everywhere. However, due to the correlations between outputs, the runs in NROY space match the observed time series reasonably well, except for the water cut. If a further wave was to be performed, emulating a statistic of this time series, such as output at month 36, or where non-zero

values start to be observed, would allow more space to be removed.

Additionally, at later waves the emulators are more representative of the computer model, as there are more points in a smaller space. Therefore, rather than using the more conservative second maximum implausibility measure, runs could simply be discarded if any of the individual implausibilities are greater than 3.

3.8. Conclusion

The comparison of regression and Gaussian process emulators in the context of multi-wave history matching in this chapter suggests that, in general, Gaussian process emulators should be favoured over regressions, regardless of the dimensionality of the input space, and that history matching should be an iterative process. The only exception is if there are either a large number of outputs being emulated, or several waves to be performed. In this case, it is acceptable to fit regression emulators initially if the computational burden is too great. For single wave experiments, as is usually the case in Bayesian calibration, a Gaussian process emulator should be used. The examples considered here were all relatively smooth, and of moderate dimension, so that these conclusions may not be completely general.

It is more common, especially in the case of climate models, that the model output is high-dimensional, for example a spatial field over the globe. While this output can be compressed using summary statistics, as for the IC fault model here, and univariate history matching can be performed, as the dimensionality of the output field increases it may be more informative to use the entire field to inform the history matching, allowing correlations between outputs to be accounted for. By selecting only some outputs of the model, history matching will find an NROY space based on the observations for these outputs, but offers no guarantee that the other outputs will return acceptable values. Higher-dimensional emulation and history matching will be explored in Chapter 4.

4. Looking the wrong way: the problem with the SVD basis

4.1. Introduction

There are a number of different methodologies that have been applied for the emulation and calibration of computer models with large spatial output, as discussed in Section 2.4. As the size, l, of the spatial output increases, the most computationally feasible option is to project this output onto a basis, usually derived from an ensemble of computer model runs (Higdon et al., 2008a, Wilkinson, 2010, Bayarri et al., 2007). This significantly reduces the dimensionality of the problem, and results in more straight-forward and efficient emulation.

The most commonly used basis in applications is the basis derived from the singular value decomposition of the ensemble, henceforth described as the 'SVD basis' (Higdon et al., 2008a). The ensemble is projected onto a truncated version of this basis, with emulators built for the coefficients, either univariately or multivariately, with history matching or calibration performed either on these coefficients, or on reconstructions of the original field (Foley et al., 2016, Chang et al., 2017) (see Section 2.4.1 for more details).

When the size of the ensemble, n, is small, relative to the dimension of the spatial field, l, the SVD basis is not of full rank. This is very common, particularly for climate models (e.g. Sexton et al. (2011) uses an SVD basis for 280 ensemble members and 175,000 outputs). Therefore, it is not possible to represent a general spatial field with this basis, and reconstructions using coefficients on this basis are restricted to an n-dimensional (or $n-1$, if the ensemble mean has been removed) surface in l-dimensional space. When calibrating or history matching, it is important that reconstructions that are 'similar' to the observations $\mathbf{z}$ are found if they exist. However, as we will show, when the basis is

purely ensemble-derived (as the SVD basis is), we may never find good reconstructions, regardless of whether they exist over the spatial field, as we are restricted to the (at most) n-dimensional subspace.

This chapter investigates the properties of the SVD basis method for calibration. In it, we highlight a serious deficiency with the method that, as far as we are aware, has not appeared in the literature on high-dimensional calibration in uncertainty quantification.

Section 4.2 introduces an idealised spatial example that will be used for illustrative purposes. Section 4.3 discusses the issues that using the SVD basis for projection may cause, and Section 4.4 investigates whether these perceived issues are evident when calibrating and history matching the spatial toy function. Section 4.5 introduces two climate models to show that the highlighted problems are seen in important applications. Section 4.6 proposes an alternative method for selecting a basis for this problem, and Section 4.7 applies this method, identifying benefits and drawbacks of our approach.

4.2. A spatial toy example

In order to illustrate and investigate the benefits and drawbacks of the SVD basis method, we design an idealised example, which we will hereafter refer to as the 'toy example'. There are several important features that this function requires so that it is useful for providing insight into current practices in the literature.

The main reason for using a toy function is that it is quick to evaluate for any setting of the parameters, but also that the dimensionality of the output is not so great that this causes computational difficulties. When calculating the multivariate implausibility for a parameter setting $\mathbf{x}$ in history matching (Craig et al., 1997), or the likelihood for $\mathbf{x}$ in calibration (Wilkinson, 2010), an inversion of an $l \times l$ variance matrix is required. This varies with $\mathbf{x}$, so that to evaluate the implausibility or likelihood as often as required in calibration or history matching, thousands or millions of matrix inversions may be required. To allow for this, the toy example developed here will have 100 outputs, arranged in a 10×10 field when displayed, and treated as a 100-dimensional vector in calculations.

To study the quality of the SVD basis when patterns similar to the observations are not

available from the ensemble, the majority of function evaluations from a Latin hypercube sample of parameter space should lead to model output that is dissimilar to the observations. Therefore, the region of parameter space that contains output similar to the observations should be small, so that it is unlikely that an initial ensemble of model runs will capture the required signal in the output.

We use a 'perfect model' approach, so that the experiment is one where it is known that an $\mathbf{x}^*$ exists with $f(\mathbf{x}^*)$ that matches the observations, up to observation error. The question then becomes one of whether or not this $\mathbf{x}^*$ can be found using the SVD basis and calibration or history matching.

Given all of these requirements, we design a toy function $f(\mathbf{x})$, with six input parameters $x_1, \ldots, x_6$ that each take on values in $[-1, 1]$. The function is defined using a set of orthogonal basis vectors $(\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_8)$ specified over a 10×10 grid, shown by Figure A.1. Combinations of these basis vectors are summed to give an output field $f(\mathbf{x})$, given by (A.5). The most important of these basis vectors are $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$, as they have been defined to represent the main signal for the 'true pattern' found in the observations, and the 'biased pattern' found in the majority of the ensemble. To represent internal variability, every time the function is run, 100 samples are taken from a zero mean Normal distribution, and added to the deterministic portion of the output to give $f(\mathbf{x})$.

The least important parameter is perhaps x_1, as it does not provide any contribution to either $\boldsymbol{\varphi}_1$ or $\boldsymbol{\varphi}_2$. x_2, and to a lesser extent x_3, affect $\boldsymbol{\varphi}_2$, with values towards the boundaries of the interval $[-1, 1]$ providing a larger effect. x_4 gives the value of the Normal density with mean 0.2 and variance 0.1^2, so that x_4 would need to be close to 0.2 to have a large impact on $\boldsymbol{\varphi}_1$. x_5 and x_6 in combination are also important for driving the pattern in $\boldsymbol{\varphi}_1$.

The value of $\mathbf{x}^*$ is chosen as

$$\mathbf{x}^* = (0.7, 0.01, 0.01, 0.25, 0.8, -0.9)$$

However, because for real-world systems the observations are only accurate up to some observation error, a value from the distribution of this error, $\mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{e})$ is sampled, and added to the model output at $\mathbf{x}^*$ to give the observed field, $\mathbf{z}$. Here, $\boldsymbol{\Sigma}_\mathbf{e}$ is defined using the Gaussian correlation function, with the input parameters treated as the spatial

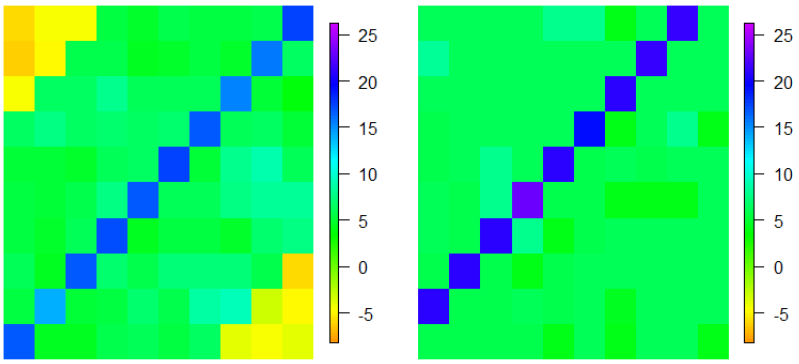

Figure 4.1. The observations, $\mathbf{z}$, for the toy function (left), and the mean of the ensemble $\mathbf{F}$.

coordinates of the 10×10 grid, $s_i = (a_i, b_i)$. The correlation length for each dimension is set equal to 1, the nugget parameter set at 0, and the common variance multiplier set equal to 1, so that the i, j^{th} entry of $\mathbf{\Sigma_e}$ is

$$\mathbf{\Sigma}_{\mathbf{e}}^{ij} = \exp\{-(a_i - a_j)^2 - (b_i - b_j)^2\} \tag{4.1}$$

The observations, $\mathbf{z}$, are shown in the left panel of Figure 4.1. A key feature of this spatial field is that the largest output values appear on the diagonal going from the bottom left to the top right (for simplicity, this diagonal pattern will be described as the 'main diagonal' in future). In the other corners of the output are the smallest output values, with values less than zero towards these corners. Between these two distinct features there are no clear patterns. The location of the largest output values will be of most interest.

A Latin hypercube sample of size 60 is taken from the 6-dimensional parameter space $\mathcal{X}$, giving an ensemble $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{60}))$ with dimension 100×60. The gridded model output is converted into a vector by 'stacking' the columns: the first column of the output matrix is entered into the 100-dimensional vector first, followed by the second column, and so on until the 100-dimensional vector contains all of the gridded output. The right panel of Figure 4.1 shows the mean output field of this ensemble (and this is representative of the individual model runs). Here, the largest values are no longer on the main diagonal, and instead lie one grid box above the main diagonal. Essentially, the most common output of the model is this biased version of $\mathbf{z}$, with the strongest signal shifted away from the main diagonal, and increased in intensity. This successfully sets up a problem with the desired qualities.

The output of the toy function could be thought of in a climate context. For example, the strong signal in the spatial field could be the Gulf Stream, with this observed in the incorrect location in the model output. The problem is then to identify whether or not the model can fix this observed bias.

4.3. The SVD basis

Given an ensemble $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$, an $(l \times n)$-dimensional matrix containing the vectorised spatial fields in the n columns, define the ensemble mean $\boldsymbol{\mu}$ as a vector of length l, with i^{th} entry

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

where $f_i(\mathbf{x})$ is the i^{th} output of the model at $\mathbf{x}$. In all that follows, the ensemble mean will first be removed from the model output, to remove common patterns that explain a large percentage of the ensemble variability. The centred ensemble is defined as

$$\mathbf{F}_{\boldsymbol{\mu}} = (f(\mathbf{x}_1) - \boldsymbol{\mu}, \ldots, f(\mathbf{x}_n) - \boldsymbol{\mu}) \tag{4.2}$$

The SVD (or principal component) basis is found by calculating the singular value decomposition of the centred ensemble (Golub and Reinsch, 1970):

$$\mathbf{F}_{\boldsymbol{\mu}}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \tag{4.3}$$

The SVD basis is defined as the first $n-1$ columns of $\mathbf{V}$, and is denoted $\boldsymbol{\Gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_{n-1})$. The first q columns of this basis are referred to as $\boldsymbol{\Gamma}_q$, and this basis is named the 'truncated basis'. The discarded basis vectors are denoted by $\boldsymbol{\Gamma}_{-q}$.

To project a spatial field $f(\mathbf{x})$ with l outputs onto a basis $\boldsymbol{\Gamma}$ with $n-1$ vectors, we calculate

$$\mathbf{c}(\mathbf{x}) = (\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}^T(f(\mathbf{x}) - \boldsymbol{\mu}) \tag{4.4}$$

This reduces the dimensionality of the output at $\mathbf{x}$ from l to $n-1$: instead of having l outputs associated with $\mathbf{x}$, there are now $n-1$ coefficients for each $\mathbf{x}$.

The coefficients can be used to 'reconstruct' or 'back-project' to the original l-dimensional

output space, giving a reconstruction $\mathbf{r}(\mathbf{x})$ of $f(\mathbf{x})$:

$$\begin{aligned}\mathbf{r}(\mathbf{x}) &= \boldsymbol{\Gamma}\mathbf{c}(\mathbf{x}) + \boldsymbol{\mu} \\ &= \boldsymbol{\Gamma}(\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}^T(f(\mathbf{x}) - \boldsymbol{\mu}) + \boldsymbol{\mu}\end{aligned} \tag{4.5}$$

This may not be identical to the original field: the basis is not full rank because $n < l$, and hence any general l-dimensional vector may not be represented exactly using $\boldsymbol{\Gamma}$. If however $\mathbf{x} \in (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, those inputs used in the ensemble $\mathbf{F}$, then the reconstruction is exact. Jolliffe (2002) shows that this is the case generally when principal components of data are taken, and this holds in the computer experiment application.

In the case of the SVD basis, $\boldsymbol{\Gamma}^T\boldsymbol{\Gamma}$ is the identity matrix, and the expressions in (4.4) and (4.5) can be simplified to remove the matrix inversion.

Combining the methodology of Higdon et al. (2008a) for emulating spatial fields using an orthogonal basis, with the univariate Gaussian process emulator methodology of Haylock and O'Hagan (1996), we build emulators for the coefficients of the first q SVD basis vectors:

$$c_i(\mathbf{x}) \sim \mathrm{GP}(m_i(\mathbf{x}), \sigma_i^2 C_i(\mathbf{x}, \mathbf{x})) \tag{4.6}$$

where $m_i(\cdot)$ and $C_i(\cdot, \cdot)$ are chosen mean and correlation functions, which may be different for $i = 1, \ldots, q$. Only the first q basis vectors have emulators built for the coefficients because projection onto these vectors explains the majority of ensemble variability (generally, 90% or 95% is deemed to be sufficient). The later basis vectors are difficult to emulate due to explaining a small percentage of variability, so are not included directly at the emulation stage. In this chapter, the uncertainty due to ignoring these vectors is incorporated into the variance of the emulated reconstruction, as in Wilkinson (2010), giving the following posterior distribution for the emulator for $f(\mathbf{x})$:

$$f(\mathbf{x})|\mathbf{F}, \boldsymbol{\sigma}^2, \boldsymbol{\phi} \sim \mathrm{N}(\boldsymbol{\Gamma}_q\mathbf{m}^{**}(\mathbf{x}), \boldsymbol{\Gamma}_q\mathbf{C}^{**}(\mathbf{x}, \mathbf{x})\boldsymbol{\Gamma}_q^T + \boldsymbol{\Gamma}_{-q}\boldsymbol{\Sigma}^*\boldsymbol{\Gamma}_{-q}^T) \tag{4.7}$$

where

$$\begin{aligned}
\mathbf{m}^{**}(\mathbf{x}) &= (m_1^{**}(\mathbf{x}), \ldots, m_q^{**}(\mathbf{x}))^T \\
\mathbf{C}^{**}(\mathbf{x}, \mathbf{x}) &= \text{diag}(\sigma_1^2 C_1^{**}(\mathbf{x}, \mathbf{x}), \ldots, \sigma_q^2 C_q^{**}(\mathbf{x}, \mathbf{x})) \\
\boldsymbol{\sigma}^2 &= (\sigma_1^2, \ldots, \sigma_q^2) \\
\boldsymbol{\Sigma}^* &= \text{diag}(\Sigma_{q+1,q+1}, \ldots, \Sigma_{n-1,n-1})
\end{aligned}$$

$m_i^{**}(\mathbf{x})$ and $C_i^{**}(\mathbf{x}, \mathbf{x})$ are the posterior mean and variance from the Haylock and O'Hagan (1996) univariate emulators (equations (2.9) and (2.10)) for the coefficients on basis vector i, and $\Sigma_{q+i,q+i}$ is the eigenvalue for the $(q+i)^{th}$ basis vector, given by $\boldsymbol{\Sigma}$ in (4.3).

Therefore, given emulators for the ensemble coefficients on the first q basis vectors, equation (4.7) gives a probability distribution for the field. This distribution can be used for calibration, or an expectation and variance can be extracted for history matching the spatial field.

4.3.1. For the toy example

Using the toy example defined in Section 4.2, and the previously described ensemble $\mathbf{F}$ with $n = 60$ members, we calculate the SVD basis. First, the ensemble mean is subtracted from each individual model run, before this centred output is decomposed via singular value decomposition, giving 100×59 basis $\boldsymbol{\Gamma}$.

The first 8 SVD basis vectors are shown in Figure 4.2. Between them, these 8 vectors explain over 99.9% of the variability in the ensemble. Even after the ensemble mean has been subtracted, the basis vector that explains the most variability in the centred ensemble highlights the off-diagonal pattern, with a strong signal here and little difference from zero elsewhere. This basis vector explains 64% of the variability in the centred ensemble, and is generally similar to $\boldsymbol{\varphi}_2$ from the definition of $f(\cdot)$.

The second SVD basis vector mainly captures the pattern from $\boldsymbol{\varphi}_4$, and explains 16% of the variability across the ensemble. The third SVD basis vector mostly corresponds with $\boldsymbol{\varphi}_3$, explaining 11% of variability. These first three SVD basis vectors explain 91% of the ensemble variability between them, and adding in the fourth SVD vector (with a similar

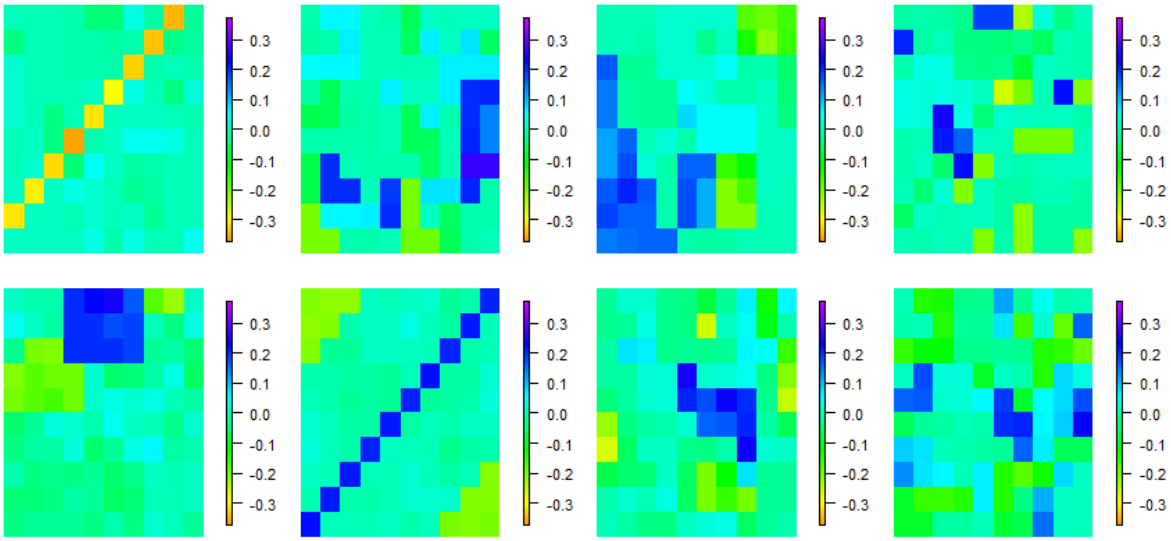

Figure 4.2. The first 8 SVD basis vectors of the centred ensemble.

pattern to $\boldsymbol{\varphi}_8$) increases the total percentage explained to 96.6%. The basis would be truncated here in applications, because the vast majority of ensemble variability ($> 95\%$) has been captured with these four basis vectors. The remaining basis vectors individually explain very small proportions of the ensemble variability, and hence do not contain much signal when the ensemble is projected onto them. This may be a problem if these basis coefficients need to be emulated.

However, these next SVD basis vectors are shown here in Figure 4.2 for illustrative purposes. The sixth SVD basis vector looks extremely similar to $\mathbf{z}$, highlighting the main diagonal as well as having negative values in the corners. Although this basis vector only explains 1% of the ensemble variability, it is perhaps important to include this in any analysis because of the desirable pattern it exhibits.

It is slightly fortunate, and due to the construction of the toy function using orthogonal vectors, that this pattern, similar to the chosen observations, is clearly visible in the SVD basis. In most applications, this pattern, if it is represented by the ensemble variability at all, will not (or is unlikely to) appear in this manner. For example, if there were more orthogonal directions in the function definition, then this pattern, if present, would likely be more hidden. It may be a low-eigenvalue basis vector, and if n was higher, this may never be checked, or, even worse, it may be disguised as a linear combination of several different eigenvectors, and be impossible to identify by eye.

Even if we observe that an important vector, in terms of being similar to $\mathbf{z}$, is found later

in the SVD basis, simply extending the truncated basis to include this will not usually be a suitable fix. It is likely that this basis vector explains $< 1\%$ of the variability in $\mathbf{F}_{\boldsymbol{\mu}}$, and hence building emulators for the coefficients will likely lead to predictions with a large amount of uncertainty for all $\mathbf{x}$, so that they are not informative.

To illustrate this problem, we fit an emulator to the coefficients on the sixth SVD basis vector for our toy function. The cross-validation plot for this emulator is shown in Figure 4.3. There is a large uncertainty on all predictions, relative to the spread of coefficients. The majority of the ensemble members give projections close to zero, due to the lack of ensemble signal explained by projection onto this basis vector, with a couple of extreme values. Due to these outlying values, the emulator that has been fitted gives predictions away from zero for some of the ensemble members with coefficients similar to zero. Furthermore, the emulator has failed to predict the two most extreme ensemble coefficients. Given that the projection of the observations onto this basis vector gives a coefficient of 51, it is clear that this emulator is not of practical use, as it will not be able to give this as a prediction, even with uncertainty included.

For this reason, it is realistic to truncate the SVD basis after 95% of ensemble variability has been explained, as this would normally occur in applications. Throughout this thesis, we assume that a large proportion of variance explained is a reasonable proxy for ease of emulation. This truncated SVD basis is hereafter referred to as $\boldsymbol{\Gamma}_4$. Truncating the basis in this manner allows a comparison to be made between bases with and without the pattern similar to $\mathbf{z}$ in them. If the basis does not contain this pattern, how does this affect the calibration or history matching, and can accurate conclusions be made, given that the true value of $\mathbf{x}^*$ is known?

4.3.2. Reconstructing z

In calibration or history matching, the goal is to find parameter settings that are either consistent, or not inconsistent in the case of history matching, with $\mathbf{z}$. For spatial problems, general output fields cannot be found using the projection and reconstruction method, as the rank of the basis is $(n-1) << l$. Even if the number of ensemble members is greater than the number of outputs, because the basis is truncated so that emulators can be built, the basis used for emulation and then calibration is not of full rank. Therefore, the choice

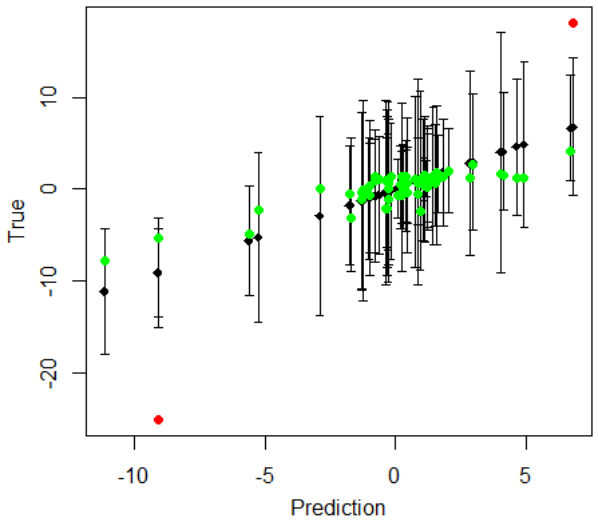

Figure 4.3. The leave-one-out cross-validation plot for the emulator for the coefficients given by projecting onto the sixth vector of the SVD basis, with the error bars representing 99% uncertainty bounds.

of basis constrains the original l-dimensional output space so that only a subspace of it can be found using reconstructions with this basis.

This perhaps identifies a fundamental problem with the SVD basis. Since it is only defined using the ensemble, it relies on the fact that ensemble members and the observations are similar enough, i.e. they contain the same main modes of variability, so that the SVD basis is an acceptable representation of the model output, both seen and unseen. Any ensemble member can be represented perfectly using the SVD basis by definition, and because the model is representing a physical system, it is hoped that the physics of the model are adequate for reproducing the observed values of the system, to within some tolerance to error. If $\mathbf{z}$ is similar to runs in $\mathbf{F}$, the SVD basis may be adequate for reconstructing $\mathbf{z}$.

In most applications, only a vanishingly small region of parameter space has been explored with the ensemble, and it may not be representative of, or be able to capture, all important or feasible model behaviours, especially if l is large. When the model represents a complex physical system, there are likely to be differences between $\mathbf{z}$ and $\mathbf{F}$ in certain regions of the output. Due to the limited number of directions that can be defined using the ensemble, these differences will not be accounted for by the SVD basis. If a certain pattern or bias is found in every member of the ensemble, this bias will be contained in the SVD basis, and hence also in any reconstructions of model output given by this basis. However, this does not necessarily imply that this bias is a structural error, as it may be that the correct part of parameter space has not yet been explored.

To illustrate the above scenario, consider an extreme example where every member of the ensemble is a multiple of the off-diagonal basis vector, $\boldsymbol{\varphi}_2$. Then, there is only one orthogonal direction of variability in the ensemble, and the SVD basis for this ensemble will consist of a normalised $\boldsymbol{\varphi}_2$, denoted $\tilde{\boldsymbol{\varphi}}_2$, followed by some arbitrary patterns orthogonal to this. 100% of the ensemble variability would be explained by the first SVD basis vector, and each ensemble member could be represented perfectly just using this vector. However, this basis vector can clearly not represent any other variability allowed by the true model $f(\cdot)$. A general field with non-zeros away from the off-diagonal projected onto the first SVD basis vector would give a coefficient c. When used for reconstruction, the resulting field would be $c\tilde{\boldsymbol{\varphi}}_2$, with any patterns away from the off-diagonal lost.

Signal in a general l-dimensional vector may be lost when projected onto q or n basis vectors, for $q < n << l$. If the ensemble, and hence the SVD basis, does not contain patterns similar to $\mathbf{z}$, this signal may be lost through projection and reconstruction: for any given coefficients, reconstructions of fields given by the SVD basis may contain any strong signals that are present in the ensemble.

We now illustrate this using $f(\cdot)$. The ensemble, and hence the truncated SVD basis $\boldsymbol{\Gamma}_4$, contains a strong signal on the off-diagonal, whereas when calibrating, we are more interested in fields where the strongest signal is observed on the main diagonal, as for $\mathbf{z}$. If $\mathbf{z}$ is projected onto this basis, and is then mapped back to the original l-dimensional output space, the resulting field is given by

$$\mathbf{r}(\mathbf{z}) = \boldsymbol{\Gamma}_4(\boldsymbol{\Gamma}_4^T\boldsymbol{\Gamma}_4)^{-1}\boldsymbol{\Gamma}_4^T(\mathbf{z} - \boldsymbol{\mu}) + \boldsymbol{\mu}$$

The left panel of Figure 4.4 shows this field, and we see that it is completely different from $\mathbf{z}$. Projecting onto $\boldsymbol{\Gamma}_4$, and then reconstructing using these coefficients, has removed the strong signal on the main diagonal, and given a field that is relatively constant everywhere.

Note that there are no emulators involved here: the exact coefficients found when $\mathbf{z}$ is projected onto $\boldsymbol{\Gamma}_4$ have been used for this reconstruction. Therefore, once emulators are added, even if these emulators are perfect representations of the coefficients on each basis vector, with no uncertainty on the emulator predictions, a field resembling $\mathbf{z}$ cannot be found.

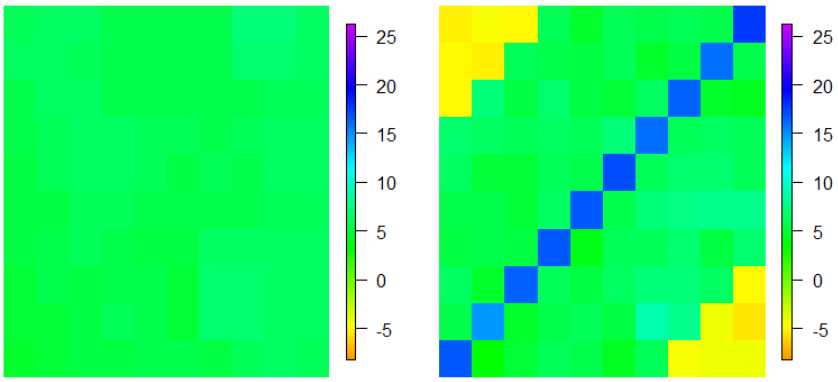

Figure 4.4. The reconstruction of $\mathbf{z}$ after it has been projected onto the first four SVD basis vectors $\mathbf{\Gamma}_4$, and the full SVD basis $\mathbf{\Gamma}$.

If, instead, $\mathbf{z}$ is projected onto the full SVD basis $\mathbf{\Gamma}$, the reconstructed field is extremely similar to $\mathbf{z}$, as shown in the right panel of Figure 4.4. The reconstruction using this basis has captured the important patterns of $\mathbf{z}$, and although the individual grid box values are not exactly the same as for $\mathbf{z}$, this is due to the noise in the ensemble.

Rather than this being an argument for using the full SVD basis in place of the truncated version in general, the excellent reconstruction of $\mathbf{z}$ with the full basis in our example is due to including the sixth SVD vector, i.e. the one with a pattern similar to $\mathbf{z}$. Once this vector is included in the basis, the reconstruction of the observations improves dramatically. That is, this is a property of the construction of the toy function and ensemble, instead of a general property of an SVD basis. Even if the answer were as simple as using the full SVD basis, emulation becomes a problem due to lack of signal on lower-order basis vectors (as demonstrated in Figure 4.3).

The property of being unable to reconstruct $\mathbf{z}$ with $\mathbf{\Gamma}_4$ suggests that if the desire is to find parameter settings that reproduce $\mathbf{z}$, the truncated SVD basis is not suitable: even before emulators are involved, it is impossible to produce output fields similar to $\mathbf{z}$. The basis choice has restricted the space of possible reconstructions to a 4-dimensional subspace in 100-dimensional space, that does not contain the important patterns of $\mathbf{z}$.

The implication that this has on a potential calibration or history matching exercise is problematic. Despite the fact that the true $\mathbf{x}^*$ is known for the toy example, being unable to find fields in the output space that are similar to $\mathbf{z}$, or even contain the main diagonal signal from $\mathbf{z}$, is likely to lead to all of parameter space being ruled out when history

matching. When all of parameter space is ruled out, if the emulators are accurate (which is likely to be the case for the first four SVD basis vectors), then the conclusion is that the computer model does not have parameter settings that lead to $\mathbf{z}$, under the current tolerance to error, and any biases may be deemed structural errors. This would be an incorrect conclusion for this example, given that $\mathbf{x}^*$ exists.

Bayesian calibration by definition must return a probability distribution over the input parameters, regardless of the suitability of the computer model for representing $\mathbf{z}$. In the situation that the difference between $\mathbf{z}$ and emulated outputs is large (given the various uncertainties required), calibration will return a spike of density somewhere in $\mathcal{X}$. There is no guarantee that this will correspond to $\mathbf{x}^*$, leading to a likely-incorrect region of parameter space being highlighted, rendering the calibration distribution useless if forecasting is the intended goal.

Both calibration and history matching may be unable to overcome a poor basis, and ruling out all of parameter space or highlighting the wrong region of $\mathcal{X}$ may not imply that $\mathbf{x}^*$ does not exist. Instead, the basis, and hence emulator, may not be allowing a search in the correct directions of the output space. This will be explored further with the toy example in Section 4.4.

4.3.3. Quantifying the reconstruction error

Although in our example it is clear that $\mathbf{\Gamma}_4$ poorly reconstructs $\mathbf{z}$, the difference between $\mathbf{z}$ and reconstructions of $\mathbf{z}$ using a basis needs to be quantified. For a general basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_q)$ with q vectors, which need not be orthogonal, we define this difference as the 'reconstruction error':

$$\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) = \|\mathbf{z} - \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{z}\|_{\mathbf{W}} \tag{4.8}$$

where

$$\|\mathbf{z} - \mathbf{r}(\mathbf{z})\|_{\mathbf{W}} = (\mathbf{z} - \mathbf{r}(\mathbf{z}))^T\mathbf{W}^{-1}(\mathbf{z} - \mathbf{r}(\mathbf{z})) \tag{4.9}$$

(see Section 5.7 for a refined version of (4.8)). It is desired that $\|\mathbf{z} - \mathbf{r}(\mathbf{z})\|_{\mathbf{W}}$ is less than ϵ, for some chosen tolerance to error $\epsilon > 0$ and an $l \times l$ weight matrix $\mathbf{W}$. Rather than $\mathbf{z}$, this will instead generally be the centred version, $\mathbf{z} - \boldsymbol{\mu}$, because the basis will be defined

using the centred ensemble. We will continue to refer to $\mathbf{z}$, as it will be clear from context whether this has been centred by the ensemble mean.

The choice of the weight matrix should be problem dependent. If all outputs are to be treated equally, for example if there is no prior knowledge about the structure of any errors, then $\mathbf{W}$ may be set equal to the $l \times l$ identity matrix $\boldsymbol{\mathcal{I}}_l$. Then, if the expression in (4.8) is divided by l, averaging the reconstruction error across the l grid boxes, this is the mean squared error for the difference between $\mathbf{z}$ and its reconstruction. By introducing this division by l, (4.8) can be described as the 'weighted mean squared error', given a weight matrix $\mathbf{W}$. The tolerance to error ϵ may then be set based on this. If the output field is temperature, for example, then ϵ could be set equal to 1, i.e. it is desired that reconstructions are within, on average, 1°C of $\mathbf{z}$.

If the reason for emulation is to calibrate or history match $f(\cdot)$, then a natural alternative choice for $\mathbf{W}$ is the sum of the observation error and discrepancy variances, as in the definition of implausibility. If history matching is to be performed, then these quantities should already be available, and can therefore be used to assess the quality of the basis prior to emulation. This would allow for any known structural errors to be accounted for in the reconstruction: if it is known that accurate values cannot be found for some grid boxes when $f(\cdot)$ is run, then differences between $\mathbf{z}$ and the reconstruction for these outputs should not be weighted as highly as outputs for which accuracy may be possible. Similarly, if there are known observation errors, and these vary across the outputs rather than being constant, then including these errors in $\mathbf{W}$ may be sensible.

By setting $\mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$, equations (4.8) and (4.9) have a similar form as the multivariate implausibility:

$$\mathcal{I}(\mathbf{x}) = (\mathbf{z} - \mathrm{E}[f(\mathbf{x})])^T (\mathrm{Var}(f(\mathbf{x})) + \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}})^{-1} (\mathbf{z} - \mathrm{E}[f(\mathbf{x})]) \tag{4.10}$$

Rather than the emulator expectation, equation (4.8) contains the reconstruction of $\mathbf{z}$ with the basis $\mathbf{B}$. Since this is an exact calculation with no emulation involved, there is no associated variance term for this, and hence the matrix to be inverted is $\mathbf{W}$. Instead of comparing the expected difference between the observations and the output at parameter setting $\mathbf{x}$, (4.8) replaces the latter with the reconstructed field. This is analogous to the implausibility because it is also comparing $\mathbf{z}$ to another output field, and therefore the

history matching bound for NROY space can be used to select ϵ.

For multivariate history matching, the bound T for ruling out parameter settings is generally based on a chi-squared distribution with l degrees of freedom (Vernon and Goldstein (2009), see Section 2.7.2):

$$T = \chi^2_{l,0.995} \tag{4.11}$$

If the multivariate implausibility at $\mathbf{x}$, as given by (4.10), is greater than the 99.5% value of this chi-squared distribution, then $\mathbf{x}$ is unlikely to give output close to $\mathbf{z}$, with respect to $\boldsymbol{\Sigma}_\mathbf{e}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$. This can be extended to say that if the reconstruction of $\mathbf{z}$ with basis $\mathbf{B}$ gives an implausibility greater than this value, then this run would be ruled out when history matching is performed: even with perfect emulation, the basis representation of $\mathbf{z}$ would be ruled out because it is too dissimilar to $\mathbf{z}$.

Therefore, with perfect emulators, all of space would be ruled out using this basis. This would likely lead to the conclusion that there is structural error in $f(\cdot)$, and it is not possible to find parameters that give $\mathbf{z}$. However, the choice of basis has guaranteed that this would be the conclusion, independently of the quality of the model. If it is not possible to accurately represent the observations with the basis, then this will always be the conclusion (unless there is large enough emulator uncertainty causing less space to be ruled out), regardless of whether $\mathbf{x}^*$ does exist.

This issue can be illustrated with the spatial toy function. Setting the weight as the identity matrix, so that each grid box is treated equally, the value of the reconstruction error can be calculated for each truncated basis derived from the SVD basis: $\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_{59}$, where $\boldsymbol{\Gamma}_q$ denotes the first q vectors of $\boldsymbol{\Gamma}$. These are shown by the red lines in Figure 4.5, where the reconstruction error in equation (4.8) is scaled by $l = 100$ (i.e. this is the mean squared error in this case). If the bound is set as the 99.5% quantile of the chi-squared distribution with 100 degrees of freedom, T, and this is divided by 100, this can also be plotted and compared to the reconstruction error. This is added as a horizontal dotted line. Whether it is possible to find reconstructions of $\mathbf{z}$ using $\boldsymbol{\Gamma}$, up to this tolerance of error, can then be identified visually: where the red line is below the bound, reconstructions of $\mathbf{z}$ would not be ruled out if history matching was performed using this basis, with $\mathbf{W}$ as the variance.

The blue line in Figure 4.5 shows the percentage of ensemble variability explained by

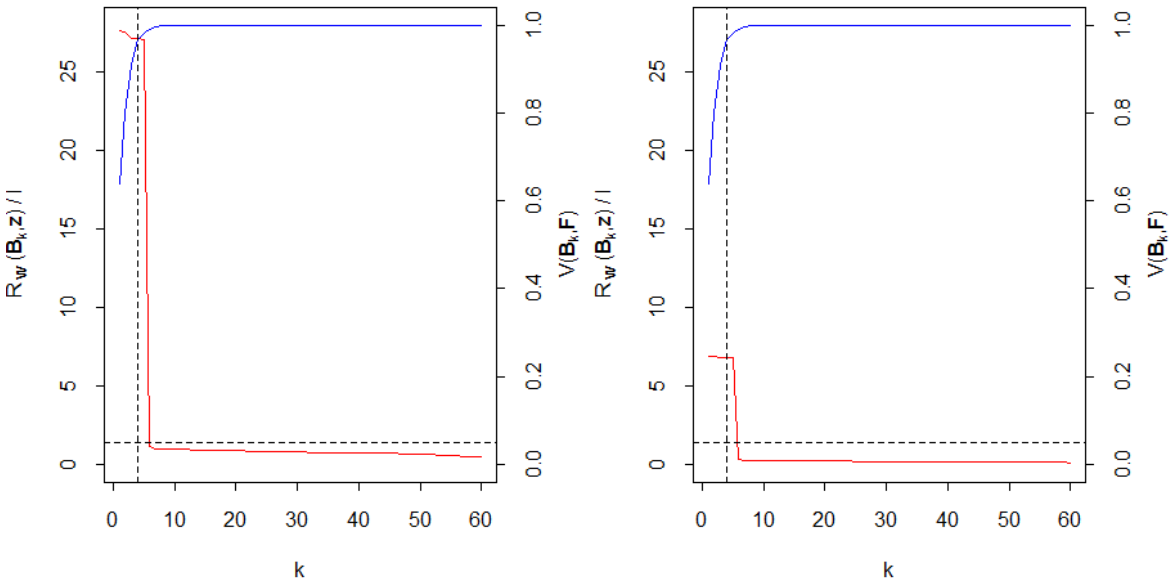

Figure 4.5. A plot showing how the reconstruction error, scaled by the field size $l = 100$ (red), and percentage of ensemble variability explained (blue) change as the SVD basis is increased in size, for $\mathbf{W} = \boldsymbol{\mathcal{I}}_l$ (left) and $\mathbf{W} = 4\boldsymbol{\mathcal{I}}_l$. The horizontal dotted line gives $\epsilon = T/100$. The vertical dotted line shows how many basis vectors are required to explain at least 95% of ensemble variability.

projection onto the first k vectors of $\boldsymbol{\Gamma}$, $\mathcal{V}(\boldsymbol{\Gamma}_k, \mathbf{F}_{\boldsymbol{\mu}})$. In the case of the SVD basis, this is defined as

$$\mathcal{V}(\boldsymbol{\Gamma}_k, \mathbf{F}_{\boldsymbol{\mu}}) = \frac{\sum_{i=1}^{k} \Sigma_{ii}^2}{\sum_{i=1}^{n} \Sigma_{ii}^2} \tag{4.12}$$

i.e. the squares of the first k eigenvalues are summed over, and divided by the total sum of squares. A more general definition for the percentage of variance explained by a basis will be given in Section 4.6.4.

The type of plot in Figure 4.5 will be referred to in future as a VarMSE plot, and the reconstruction error will always be scaled by the number of dimensions, l.

From this plot, as suggested by comparing reconstructions of $\mathbf{z}$, the first six SVD vectors are required to achieve an adequate reconstruction of $\mathbf{z}$. When only the first four are used ($\boldsymbol{\Gamma}_4$) so that 95% of the ensemble variability is accounted for, as shown by the vertical dotted line, the reconstruction error is substantially larger than the bound, and the observations would be ruled out. In reality, history matching will not be performed using the identity matrix to represent the observation error and discrepancy variances. However, if these are defined as a scalar multiplied by the identity matrix, the (left) y-axis is simply divided by this scalar. This is demonstrated by the right plot in Figure 4.5, where a constant combined error and discrepancy variance equal to $4\boldsymbol{\mathcal{I}}_l$ is assumed. In

this case, this does not change the conclusion that at least the first six SVD basis vectors are required.

These types of plots can offer a simple diagnostic check for whether a basis is suitable. Although only the identity and constant variances have been used in Figure 4.5, it highlights the types of conclusions that may be made. In this example, the SVD basis clearly contains the important patterns that allow accurate representations of $\mathbf{z}$, however they would not be included in a standard application of the SVD methodology for emulation due to explaining low percentages of the ensemble variability.

This setup allows the potential flaws of the SVD basis as discussed earlier to be studied. It is known that this toy example has settings of the parameters that lead to output close (up to the assumed errors) to the observations, but that the dominant signal in the ensemble, and indeed in the majority of $\mathcal{X}$, is biased away from this. Naively, by looking at the ensemble of runs, it may be tempting to conclude that the model is not able to find the observations, and this perceived model inadequacy may be incorporated immediately into the discrepancy term.

If the error and discrepancy variances are known in a problem, then the reconstruction error never going below the bound T suggests one of two things: either the specification of the discrepancy is incorrect, as the model can never get as close to $\mathbf{z}$ as desired, or the ensemble does not yet contain directions that allow this to be properly assessed. This advocates a multi-wave approach to exploring the output space.

4.4. Calibrating the toy function

The previous section suggests that $\mathbf{\Gamma}_4$ may not be useful for calibrating or history matching $f(\cdot)$. In order to perform either, a specification for the discrepancy variance $\mathbf{\Sigma}_{\boldsymbol{\eta}}$ is required, in addition to the earlier described variance for the observation uncertainty $\mathbf{\Sigma}_{\mathbf{e}}$ (4.1). Given both $\mathbf{\Sigma}_{\mathbf{e}}$ and $\mathbf{\Sigma}_{\boldsymbol{\eta}}$, the SVD basis's suitability for reconstructing the observations can be properly assessed via the VarMSE plot. By setting $\mathbf{W} = \mathbf{\Sigma}_{\mathbf{e}} + \mathbf{\Sigma}_{\boldsymbol{\eta}}$ so that the reconstruction error is analogous to the multivariate implausibility, if

$$\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_4, \mathbf{z}) > T$$

then the representation of $\mathbf{z}$ on this basis would be ruled out by history matching, i.e. the basis is inadequate for searching for $\mathbf{z}$.

4.4.1. Specifying the discrepancy variance

Although it is known that the toy function can reproduce the observations $\mathbf{z}$, at least up to the observation error $\mathbf{\Sigma_e}$, a non-zero discrepancy variance may still be included when history matching or calibration is performed.

Adding a discrepancy variance allows for the importance of regions of the output field to be accounted for. For example, it may be considered most important to achieve the correct magnitude for the output on the main diagonal of the field. If this is correct, one may not be as interested in the patterns elsewhere. A discrepancy can be constructed to reflect this desire, resulting in a true NROY space with less tolerance to error on the main diagonal than elsewhere.

Rather than introducing additional parameters to be estimated (as in Kennedy and O'Hagan (2001a)), the discrepancy will instead be given fixed values for this problem. This has the benefit of not introducing potential identifiability problems due to the estimation of parameters for the emulator and discrepancy components of the overall model using the same ensemble (see Section 2.7.3 for more discussion of discrepancy).

The discrepancy in our example could be specified using the random noise representing internal variability. However, this will not usually be known, and here it is assumed to be small compared to $\mathbf{\Sigma_e}$. Therefore, the discrepancy here will be used to represent the importance of certain patterns in the output.

The discrepancy for the toy function is modelled as a zero mean l-dimensional Gaussian process:

$$\boldsymbol{\eta} \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma_\eta})$$

For the discrepancy variance matrix, variances v_i for the discrepancies for each grid box are defined. Combining these with a spatial correlation function gives the $(i,j)^{th}$ entry of the variance matrix as

$$\mathbf{\Sigma}_{\boldsymbol{\eta}}^{ij} = v_i v_j C(s_i, s_j)$$

where $C(\cdot, \cdot)$ is a correlation function that gives the correlation between locations s_i and s_j. Therefore, the parameters that need to be defined to complete the specification are $v_1, \ldots, v_l$ and the correlation length parameters of $C(\cdot, \cdot)$.

For this example, the most important aspect of the output is the signal on the main diagonal, and spatial fields where this is accurate, but perhaps aren't exactly correct elsewhere, should not be ruled out. Let the indices for the grid boxes on the diagonal be contained in the set S. Then the discrepancy variances v_i are specified via

$$v_i = \begin{cases} w_1 & \text{if } i \in S \\ w_2 & \text{otherwise} \end{cases}$$

w_1 is set equal to 0.1, and w_2 to 1, so that the discrepancy allows the output to be 'more wrong' outside of S. The specification of the discrepancy is completed by using the squared exponential as the correlation function, with the correlation length parameters both set equal to 1, as in the definition of $\mathbf{\Sigma_e}$.

This method for defining the discrepancy could be extended to k sets of indices, if there were a reason to further divide the grid boxes. For the toy function, a separate multiplier could have been included for the patterns in the corners of the observations. However, these are not deemed to be as important, and hence are treated in the same fashion as the other grid boxes that are away from the main diagonal, and grouped with them.

With this discrepancy, the reconstruction error for each truncated SVD basis, $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q, \mathbf{z})$, can be calculated using $\mathbf{W} = \mathbf{\Sigma_e} + \mathbf{\Sigma_\eta}$, giving the VarMSE plot in Figure 4.6. The reconstruction error now corresponds directly to the implausibility due to the choice of $\mathbf{W}$. This makes little difference to the previously observed plots (Figure 4.5), and again the first 6 SVD basis vectors are required for the reconstruction error to be below the history matching tolerance to error, T.

This check on the suitability of a basis has never, as far as we're aware, been performed in the calibration or history matching literature. The standard practice is to truncate the basis once a chosen threshold of ensemble variability is passed, or when basis coefficients become difficult to emulate. Whether or not this gives a basis that represents $\mathbf{z}$ is not considered.

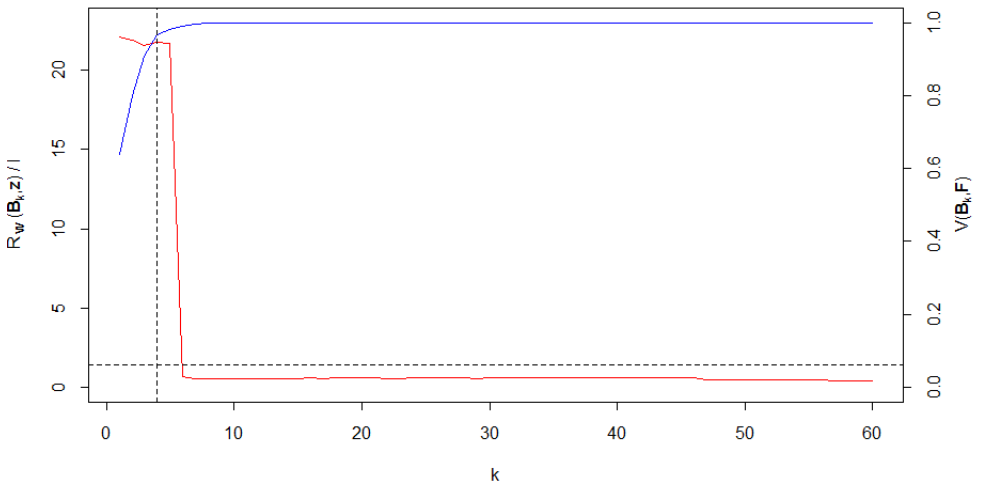

Figure 4.6. A plot showing how the reconstruction error (red) and percentage of ensemble variability explained (blue) change as the SVD basis is increased in size, for $\mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$, with the dotted lines defined as in Figure 4.5. As before, the left y-axis is scaled by $l = 100$.

4.4.2. True NROY space

Using the error and discrepancy variances, the theoretical 'true' NROY space can be identified. Since the toy function is quick to evaluate, the spatial output for any parameters $\mathbf{x}$ can easily be evaluated. Using this, the 'true' implausibility of this field can be calculated, with no emulator uncertainty involved, to estimate the percentage of $\mathcal{X}$ that leads to output close to $\mathbf{z}$, given $\boldsymbol{\Sigma}_\mathbf{e}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$.

Figure 4.7 gives a representation of this true NROY space, both as pairwise densities and points. The densities are plotted by calculating the proportion of each pairwise combination of parameters that is not ruled out, when averaged across all the other parameters (an 'optical depth plot' (Vernon et al., 2010)). The orientation of the axes for the lower left plots has been changed to allow easier comparison between the two halves. The green point in each pairwise plot shows $\mathbf{x}^*$, with the true NROY space in a region around this point.

Across the whole parameter space $\mathcal{X}$, the true NROY space consists of around 0.01% of parameter settings (note that without the added discrepancy term, the true NROY space is roughly 20% of this already small space). The locations of these points are shown here, and appear to be in a connected subspace of $\mathcal{X}$. From this, it is fairly clear that the value of x_1 is not particularly important. However, finding the correct settings of each of the other parameters is crucial.

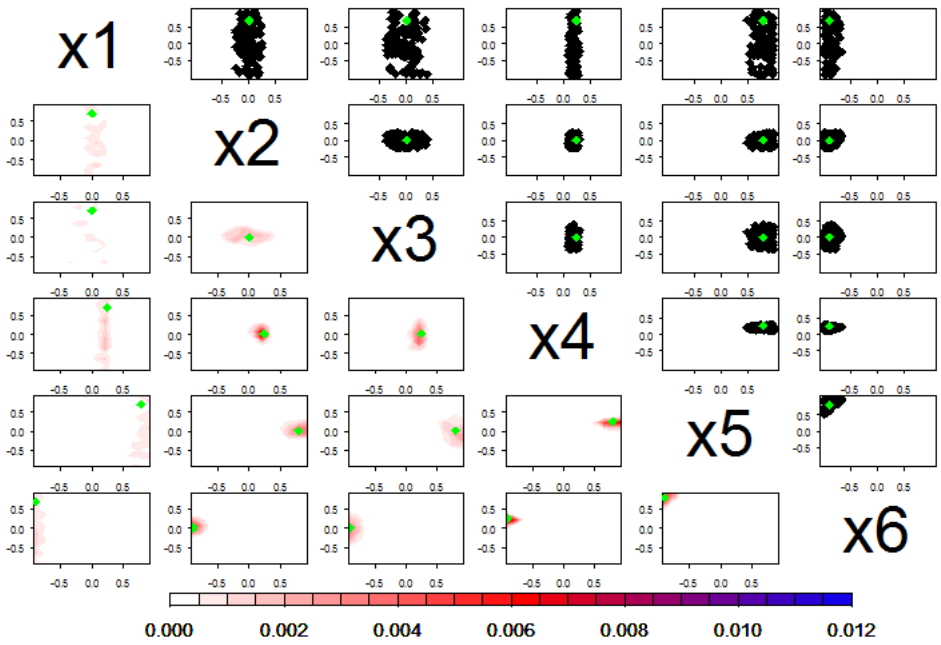

Figure 4.7. A picture showing the true NROY space for the spatial toy function. The density plots on the bottom left half show the proportion of space that is in the true NROY space for each pairwise combination of the parameters, averaged over the remaining parameters. The plots on the top right show the corresponding points in the true NROY space. The axes have been switched for the plots in the lower half (i.e. the higher parameter is always on the x-axis) so that these correspond to the top half. The green point corresponds to $\mathbf{x}^*$.

When calibration is performed for this function, the result should be that (the most likely) $\mathbf{x}^*$ is one of the runs in this true space. This is also the space that multi-wave history matching should be able to find, although the number of waves required may be high due to the small size of this space.

4.4.3. Calibration

We perform history matching and calibration using the truncated SVD basis $\mathbf{\Gamma}_4$. The reconstructions of the previous section have suggested that it is not possible to find the observations with this basis. However, this may change when emulator uncertainty is included. Given that calibration always returns a probability distribution over the input parameters, it is possible that there will be non-zero posterior density at $\mathbf{x}^*$ if this run is one of the closest representations of $\mathbf{z}$ with the chosen basis. Whether this is the case is investigated.

We apply the spatial calibration methodology of Wilkinson (2010) here. The ensemble runs are projected onto the basis, and Gaussian process emulators are constructed for the coefficients on the first four basis vectors (4.6), using the method from Section 3.3.3, with

simulated annealing in place of the previous step-by-step approach (see Appendix B.3 for validation plots for these emulators). The posterior distribution is given by mapping the emulator expectations and variances back to the original field of size l, with the discarded basis vectors accounted for in the variance (4.7).

Using this definition, the calibration distribution for the inputs is found via

$$\pi(\mathbf{x}^*|\mathbf{z}, \mathbf{F}) \propto \pi(f(\mathbf{x}^*)|\mathbf{F}, \mathbf{x}^*)\pi(\mathbf{x}^*)$$

There is assumed to be no prior information about the input parameters, therefore $\pi(\mathbf{x}^*)$ is taken to be a uniform distribution on $[-1, 1]$ for each input dimension.

We use MCMC to sample from this posterior distribution. A Metropolis-Hastings algorithm is used in this instance (Metropolis et al., 1953, Hastings, 1970). At each step, a sample is drawn from the proposal distribution for $\mathbf{x}^*$, and the likelihood of $f(\mathbf{x}^*)$ is calculated. This new parameter draw is then either accepted or rejected based on the Metropolis acceptance ratio. For this example, the initial value for the chain was set as $\mathbf{x}^*$, so that it is possible for the MCMC to return this result.

The proposal distribution $q(\mathbf{x}'|\mathbf{x})$ is given a Normal distribution, so that it is symmetric:

$$q(\mathbf{x}'|\mathbf{x}) \sim \mathbf{N}(\mathbf{x}, \mathbf{\Psi})$$

so that the proposed value $\mathbf{x}'$ is sampled from a Normal distribution, with mean equal to the current value $\mathbf{x}$, and a specified variance matrix $\mathbf{\Psi}$.

The acceptance probability is set as

$$\alpha = \min\{1, r(\mathbf{x}, \mathbf{x}')\}$$

for

$$r(\mathbf{x}, \mathbf{x}') = \frac{q(\mathbf{x}|\mathbf{x}')l(\mathbf{x}')}{q(\mathbf{x}'|\mathbf{x})l(\mathbf{x})}$$

where $l(\cdot)$ is the likelihood:

$$l(\mathbf{x}) \propto \pi(\mathbf{x}^*)|\mathbf{V}(\mathbf{x})|^{-\frac{1}{2}}\exp\{-\frac{1}{2}(\mathbf{z} - \mathrm{E}(f(\mathbf{x})))^T(\mathbf{V}(\mathbf{x}))^{-1}(\mathbf{z} - \mathrm{E}(f(\mathbf{x})))\}$$

for

$$\mathbf{V}(\mathbf{x}) = \text{Var}(f(\mathbf{x})) + \mathbf{\Sigma_e} + \mathbf{\Sigma_\eta}$$

with a uniform prior assumed for $\pi(\mathbf{x}^*)$, and $\text{E}(f(\mathbf{x}))$ and $\text{Var}(f(\mathbf{x}))$ given by equation (4.7). To determine whether the proposed value is accepted, a value p is sampled from the uniform distribution $U(0,1)$, resulting in

$$\mathbf{x}_{new} = \mathbf{x}' \quad \text{if} \quad p < \alpha$$

$$\mathbf{x}_{new} = \mathbf{x} \quad \text{else}$$

The chains are run multiple times with different choices of the proposal variance $\mathbf{\Psi}$, resulting in similar converged chains each time. The posterior distributions for each parameter also exhibit the same general results with any choice of thinning. Figure 4.8 shows the posterior distributions with every 100th sample of the chains included. The converged MCMC chains for this calibration are given in Figure C.1.

In Figure 4.8, the values of $\mathbf{x}^*$ have been added as vertical red lines so that a comparison can be made between the calibration results and what a perfect calibration would hope to return. Here, it is clear that the calibration has not been accurate. From the earlier plot of the true NROY space, x_1 was the least constrained parameter, with comparatively narrow ranges required to return the best runs for each of the other parameters. In this posterior, x_1 has been restricted, and the peak of this distribution is reasonably close to the true value of x_1, denoted by x_1^*. Similarly, x_2 is close. However, there is little posterior density assigned to x_2^*, with a sharp peak of density slightly above this value. This may not necessarily be a problem by itself, given that the runs in the true NROY space can take on this value, so that while this may not give exactly the observations, it may still be suggesting a run that is within the tolerance to error of $\mathbf{z}$.

For x_3, there is a clear deviance from $\mathbf{x}^*$. Rather than being centred around $x_3^* = 0$, all of the posterior density is at the lower end of the potential values. x_4 is possibly the most important parameter for $\mathbf{z}$ as it controls the strength of the signal on the main diagonal. The mode of the posterior for this parameter does lie around x_4^*, but there remains a wide range of possible values: the calibration has been unable to give much insight about this important parameter. The posteriors for x_5 and x_6 both contain a spike. For x_5, this is very close to the true value, but for x_6, calibration has led to overestimation of this input.

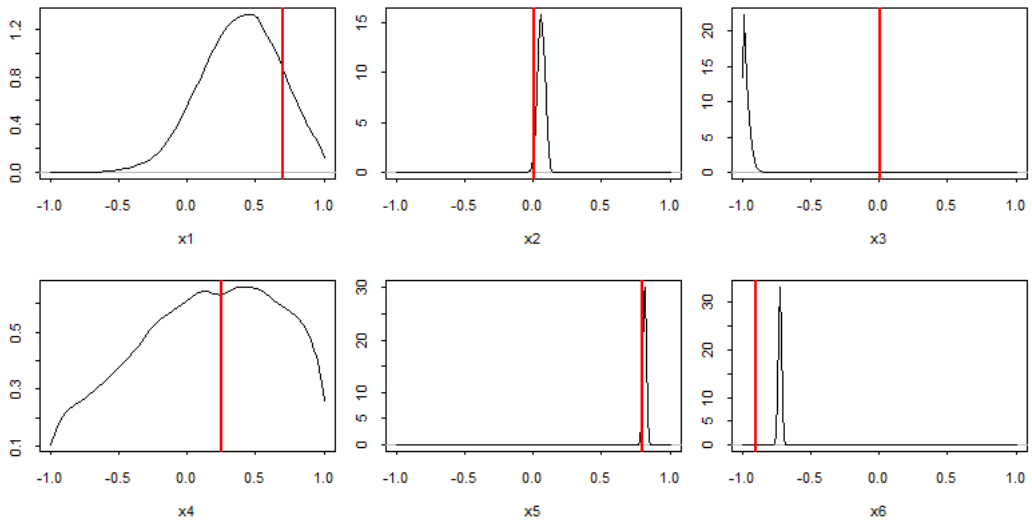

Figure 4.8. The posterior distributions for each of the parameters, when calibration is performed using the SVD basis $\mathbf{\Gamma}_4$. The red vertical lines indicate the true value of $\mathbf{x}^*$.

Taking the posterior distributions individually suggests that calibration has not managed to find the best, or possibly even good, settings of $\mathbf{x}$. There is a possibility that when taken jointly, samples of parameters from these distributions may result in output fields that are at least similar to the observations, even if they do not exactly reproduce them. We draw samples from these posterior distributions, and run the toy function at these sampled values.

The output at 16 samples from the posterior are given in Figure 4.9. Every sample contains the biased version of the observations, with the largest values lying on the off-diagonal. Some of these runs do have slightly higher values on the main diagonal than the ensemble mean (Figure 4.1), with some light blue values observed here. However, it has not generally been possible to improve upon the runs that were used for the selection of the basis.

This shows that if the basis is chosen purely using the ensemble, and does not contain patterns that are similar to the observed field, then the results of a calibration may be incorrect, and the posterior may not identify regions of parameter space that give output similar to the observations, even in scenarios where these settings are known to exist. The biases from the ensemble, and hence the inability to reconstruct $\mathbf{z}$ using $\mathbf{\Gamma}_4$, have carried through to the calibration and remain evident in the results.

This is a problematic result, as it leads to the incorrect conclusion that a parameter setting $\mathbf{x}^*$ does not exist so that $f(\mathbf{x}^*)$ is equal to $\mathbf{z}$, up to observation error. This suggests that there is a discrepancy between the model and the observations, and the conclusion of this

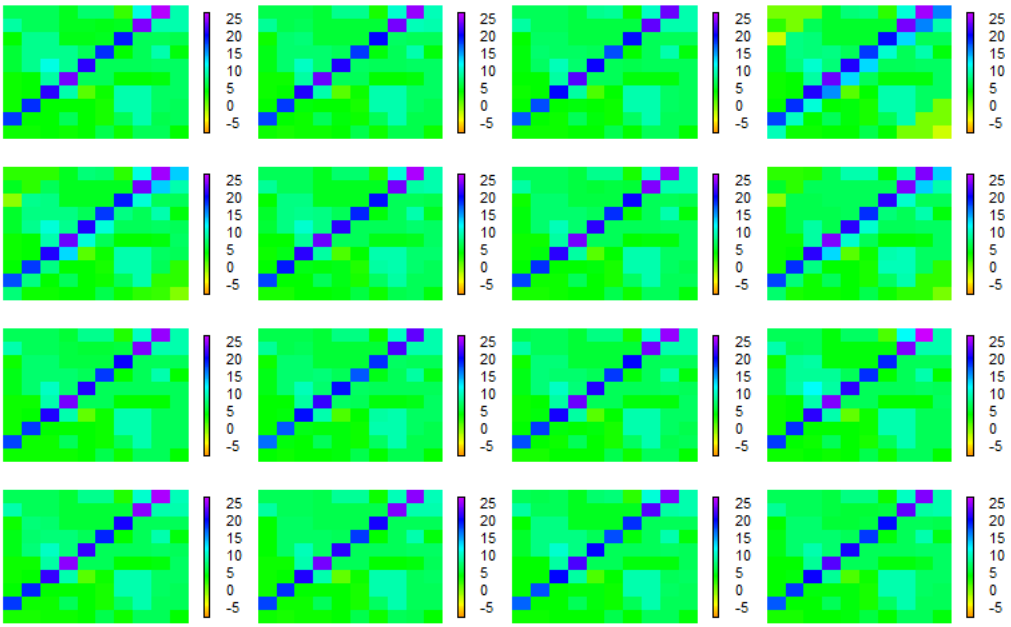

Figure 4.9. $f(\mathbf{x})$ at 16 samples of $\mathbf{x}$ from the calibration posterior distribution, using the SVD basis $\mathbf{\Gamma}_4$ for projection and emulation.

exercise would be that there is structural error in the model. If forecasting was the goal, then the discrepancy term may be changed to reflect this. In applications, the computer model will represent something physical, based on known equations of physical processes. The discrepancy, on the other hand, is a statistical model. If forecasting is performed using a large discrepancy to account for a perceived structural error that has been identified, then rather than using the physical component of the model to forecast, the statistical discrepancy will dominate the forecast. In scenarios such as for the toy function, where the perceived structural error does not in fact exist, the physical equations of the model would play less of a role in predicting future output than they could, potentially reducing the meaningfulness of and confidence in forecasts.

4.4.4. History matching on the field

Instead of calibrating, due to the clear failure of this method using the chosen basis, we perform history matching using the same ensemble, definitions of the observation error and discrepancy variances, basis, and emulators for the coefficients.

To ensure a fair comparison with calibration, history matching is carried out using the multivariate implausibility in (4.10), with the emulator predictions and variances mapped back to the original 100-dimensional output as in (4.7). The uncertainty from the discarded

basis vectors is therefore accounted for here as well. The implausibility for each point $\mathbf{x}$ is then calculated from this expectation and variance. An inversion of a 100×100 matrix is required to calculate the implausibility for each $\mathbf{x}$, a small enough dimension that it is computationally possible to evaluate the implausibility at millions of points quickly. The bound used to define NROY space is the 99.5% value of the chi-squared distribution with 100 degrees of freedom (Vernon et al., 2010). This is equal to 140.2.

Calculating the implausibility for Latin hypercube samples of $\mathcal{X}$, it is found that there are no parameter settings with implausibilities below 140.2, and hence all of parameter space is ruled out. The implausibilities range from 172 to 1464, so that all of parameter space is ruled out even if the 99.99% value of the chi-squared distribution is used to define NROY space. The runs that were earlier found to lie in the true NROY space have implausibilities between 174 and 207. When emulator uncertainty is added, it is expected that the implausibility decreases because the output is not known as accurately. The opposite has happened for these runs due to the poor reconstructions with the SVD basis $\mathbf{\Gamma}_4$.

Similar to the calibration in the previous section, this result implies that under this specification of the discrepancy and observation error, there are no parameter settings that give fields similar to the observations, and that there may be structural error in the computer model. Again, the conclusion of the chosen tuning method is incorrect.

The wrong conclusions are reached by both calibration and history matching due to the choice of basis. The ensemble containing biases away from the observations, and hence the SVD basis vectors required to explain at least 95% of the ensemble showing these biases also, resulting in poor reconstructions of $\mathbf{z}$, is the cause of this. Despite the fact that Figure 4.6 suggests that including the sixth basis vector may rectify this problem, this basis vector only explains 1% of the ensemble variability, and it is not possible to fit an emulator with much or any predictive ability to these coefficients. An alternative approach to basis selection may be required to accurately solve this calibration problem, and will be explored in Section 4.6.

4.5. Climate models

In this section, two climate models are introduced, that each give spatial output over a large field. The same problem of being unable to reconstruct the observed field using the SVD basis is illustrated here, implying that these results have implications for important applications.

4.5.1. ORCA2

ORCA2 (Rodgers et al., 2003, Madec, 2008) is the version of the ORCA model, a configuration of the NEMO ocean model, with output over a grid over the global oceans with 2° resolution. NEMO is the ocean model used in the majority of the world's climate models, and ORCA is the configuration used by the UK climate model, UKESM, at different resolutions. Other countries using NEMO include France, Germany, Canada, USA, and many more. We consider 21 input parameters of the model, controlling various aspects of physical processes (see Williamson et al. (2016) for additional details). 20 of these are scaled so that they take on values in the interval $[-1, 1]$, with the final parameter being a switch parameter, constrained to be equal to either 2 or 3.

Williamson (2015) designed an ensemble with 400 members for ORCA2 using a 25-extended 16 point Latin hypercube, and this ensemble is used here. The model output is given over a 360×180 grid on the entire globe, with values given for ocean temperature and salinity for 31 levels of the ocean. A large percentage of the 64,800 grid boxes are over land, and hence do not return values for these ocean outputs, and therefore the dimensionality of the output is $l = 39,547$. This output is given as a time series, with values at monthly intervals for each grid box for 180 years. The model output over the last 10 years is averaged, so that this becomes a spatial problem rather than a spatio-temporal one. The focus here is on the ocean temperature at the highest level, i.e. the sea surface temperature (SST). The observational data for the SST is given by the EN3 dataset (Ingleby and Huddleston, 2007).

Figure 4.10 shows the SST anomaly between the mean of the ORCA2 ensemble, and the EN3 observations. In the anomaly field there is a dipole in the North Atlantic Ocean, at the location of the Gulf Stream, with a band of strong positive bias alongside one of

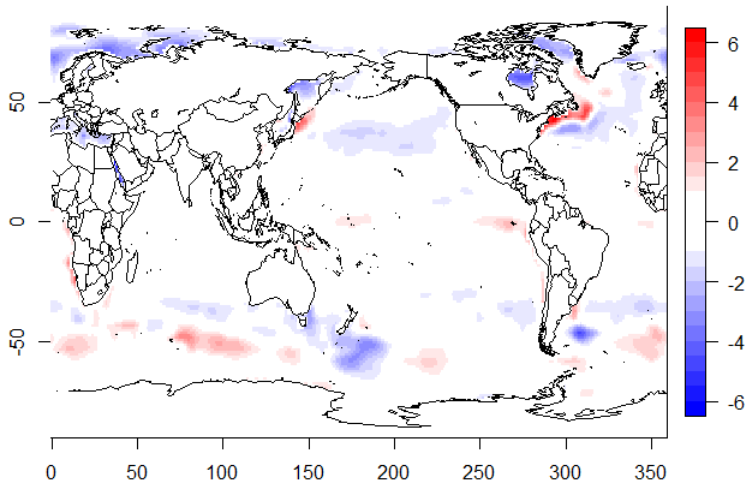

Figure 4.10. The anomaly between the ensemble mean and the SST observations, in °C.

negative bias. This appears to indicate that in the model output, the Gulf Stream is in the wrong location. This situation mirrors the toy function, with a biased version of the observations being more prevalent than the truth for sampled values from parameter space, although whether the model can in fact remove this bias or whether this is a structural error is unknown.

Other large biases between the ensemble and the observations are found in the Pacific Ocean near to the coast of Japan, with a warm bias in the model here. The Southern Ocean also contains large anomalies, with strong cold biases near New Zealand and Argentina. The anomalies between the ensemble and observations are as large as 6.5°C.

As with the toy function, the mean of the ensemble is subtracted from each individual ensemble member so that common trends are removed. The singular value decomposition of this centred ensemble is calculated, giving the SVD basis (Figure 4.11). The first basis vector explains 54% of the variability in the (centred) ensemble, and highlights the anomaly pattern observed in the Gulf Stream. In fact, patterns in the North Atlantic are observed in the majority of the leading basis vectors. The other major patterns from the average anomaly (Figure 4.10) can be observed in one of the first nine basis vectors.

In order to explain greater than 90% of the ensemble variability, the first 10 vectors of the SVD basis are required. To explain 95%, a further 10 are needed, and as these next 10 combined only explain an extra 5%, emulation may be difficult, and hence emulators would only be built for the first 10 SVD basis vectors in an application. To assess the suitability of this basis truncation, the observations are projected onto this basis, and then these

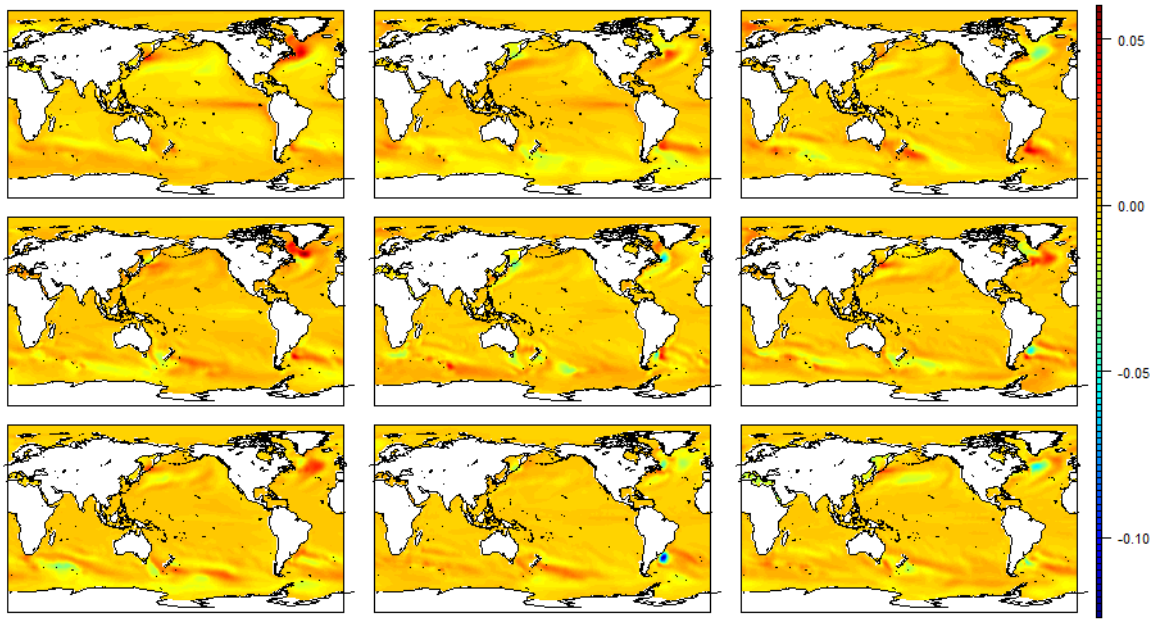

Figure 4.11. The first nine basis vectors of the SVD basis for the centred ORCA ensemble.

coefficients are used to reconstruct the original field. The anomaly of this reconstruction is shown in Figure 4.12.

This reconstruction consists of similar patterns to the anomaly for the ensemble mean, with slightly weaker biases in some places. For example, there is less of a warm bias in the Gulf Stream, and some of the warm biases previously observed in the Southern Ocean are not present. This reinforces the fact that as only the ensemble informs the choice of the SVD basis, patterns observed in the ensemble are still present in the representation of $\mathbf{z}$ (or any other general field) on this basis.

To produce the VarMSE plot, we need to specify $\mathbf{W}$. In lieu of knowledge about the true observation error or discrepancy variances, each grid box will be weighted equally. As regions coloured white (anomalies of between -1°C and 1°C) in the plots are generally deemed to be 'close enough' to the truth by the modellers ('white is alright'), we set this range of ± 1°C as equal to 3 standard deviations, so that the majority of grid boxes will have an error less than this if a field is not ruled out. Therefore, the identity matrix is multiplied by $\frac{1}{9}$ (the variance that gives a range of 2°C as 3 standard deviations) to give the weight matrix (equivalently, this can be thought of as $\mathbf{\Sigma_e}$ here).

The VarMSE plot in Figure 4.12 shows the reconstruction error for $\mathbf{z}$ when using the SVD basis, with this specification of the error variance. It shows that the reconstruction error never falls below T, even when all 400 basis vectors are used. Adding more basis vectors

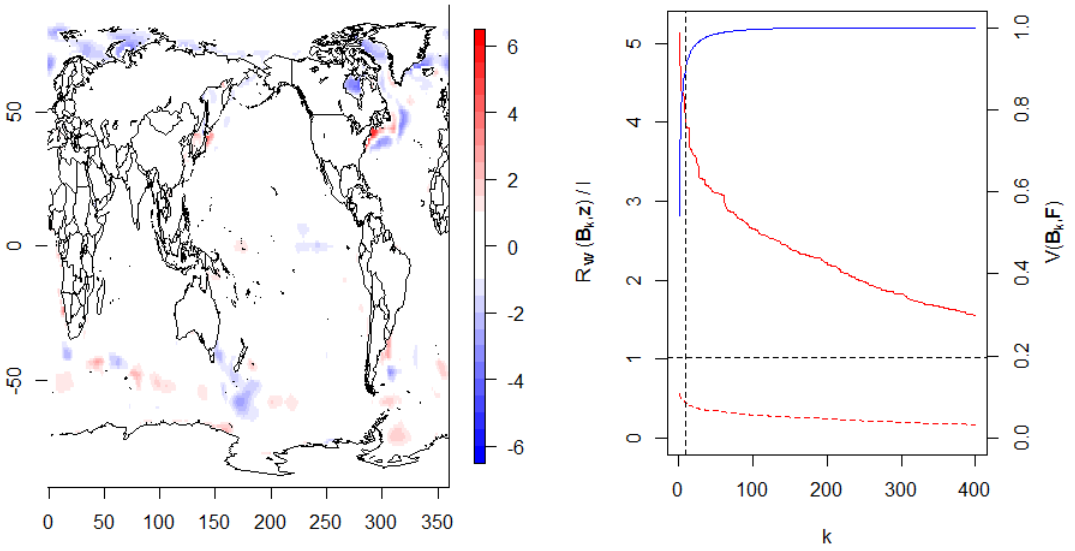

Figure 4.12. The anomaly between the reconstruction of the SST observations and the observations, using the first 10 SVD basis vectors for projection and back-projection. The right panel shows the VarMSE plot with $\mathbf{W} = \frac{1}{9}\boldsymbol{\mathcal{I}}_l$ (solid line) and $\mathbf{W} = \boldsymbol{\mathcal{I}}_l$ (dotted line).

improves reconstructions greatly, so that the truncated SVD basis theoretically performs worse than the full one, but regardless of where the basis is truncated, the representation of $\mathbf{z}$ on this basis would be ruled out. Although the true observation error and discrepancy variances are likely to be more complicated, with non-zero covariances, the magnitude of the error used here is reasonable, and clearly suggests that this basis is not suitable.

If instead we choose to set 1°C as 1 standard deviation, we have $\mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} = \boldsymbol{\mathcal{I}}_l$. This is shown by the dotted line in Figure 4.12, and suggests that the truncated SVD basis is suitable. However, this may be allowing too much error, as the reconstruction of $\mathbf{z}$ contains large anomalies, particularly in the North Atlantic, so perhaps should be ruled out.

Specifying a discrepancy term for this spatial field is important, but challenging. History matching of this spatial field is beyond the scope of this section, so a discrepancy will not be defined properly. Instead, this climate model is intended to serve as an illustration that reconstructions of $\mathbf{z}$ contain similar biases to runs in the ensemble, but also that these biases may decrease slightly as more of the SVD basis vectors are included. This suggests that there may be important patterns contained in low-eigenvalue basis vectors, or in combinations of many of these. Due to the lack of ensemble signal for these vectors, they are not practically useful for calibration or history matching because emulation is required.

4.5.2. CCCMA model

The CCCMA model, CanAM4, is an Atmospheric General Circulation Model (AGCM) (von Salzen et al., 2013). This differs from ORCA2 in that it simulates atmospheric processes, rather than the oceanic processes of an OGCM. CanAM4 is a part of the CMIP5 project (Taylor et al., 2012). von Salzen et al. (2013) describe the parametrisations of atmospheric physical processes employed by this model. The input parameter space $\mathcal{X}$ for the model has 13 dimensions, and these are scaled to have inputs in $[-1, 1]$. There are several different outputs, including precipitation (PR), sea level pressure (PSL), the net downward radiative flux at the top of the atmosphere (RTMT or TOA), the total vertical cloud overlap percentage (CLTO), air temperature (TA), the outgoing longwave radiation (RLUT), and the total sky albedo (ALBS). The output is given over a 128×64 grid of longitude-latitude locations for most of the outputs, with this grid given for 37 vertical pressure levels for certain outputs, e.g. TA. The output is given as monthly averages. To avoid the need to account for seasonal differences, the output for June, July and August (JJA) for the final 5 years is averaged, and considered to be the model output here.

Figures 4.13, 4.14 and 4.15 show the anomaly fields for the standard parameter choice of the model, for CLTO, RTMT and TA. The observational data for CLTO and RTMT are from CERES (Wielicki et al., 1996), and are given over the longitude-latitude grid. For TA, the ERA-Interim reanalysis data is treated as the observations (Dee et al., 2011). To produce an anomaly plot for TA, the longitudinal values are averaged, and pressure is plotted against latitude. In each of these plots, the areas coloured white are generally considered to be acceptable, with darker red and blue regions further away from the truth. Red indicates that the model output is greater than the observations (i.e. a warm bias in the model for temperature fields). Each of these outputs shows that there are large anomalies between the standard run of the model and the observations.

For the total cloud overlap, there are large positive biases in the North Atlantic Ocean, the northern Pacific, and near to Australia. Negative biases are more prevalent over the land masses, particularly over the Americas and Asia. The top of atmosphere balance has large positive anomalies over eastern Asia and the western Pacific, as well as regions of positive bias along the western coasts of North and South America. There is also positive bias along the eastern coast of North America and in the Gulf of Mexico. The largest

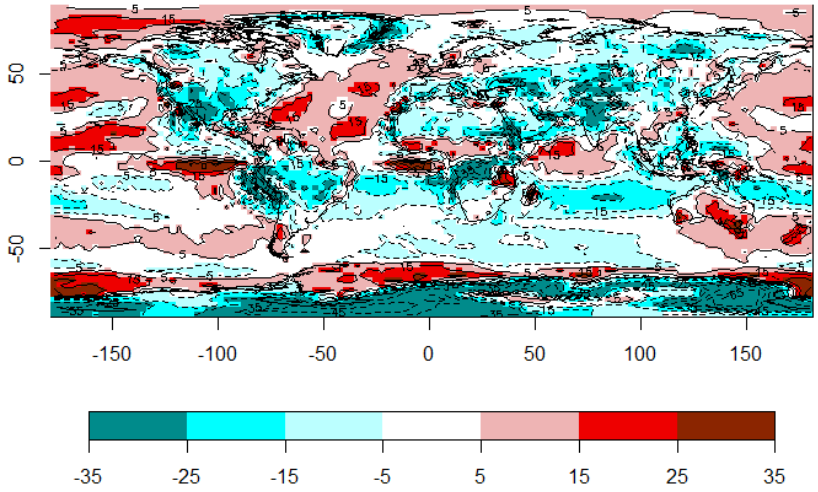

Figure 4.13. The total cloud overlap percentage (CLTO) anomaly for the standard run of CanAM4.

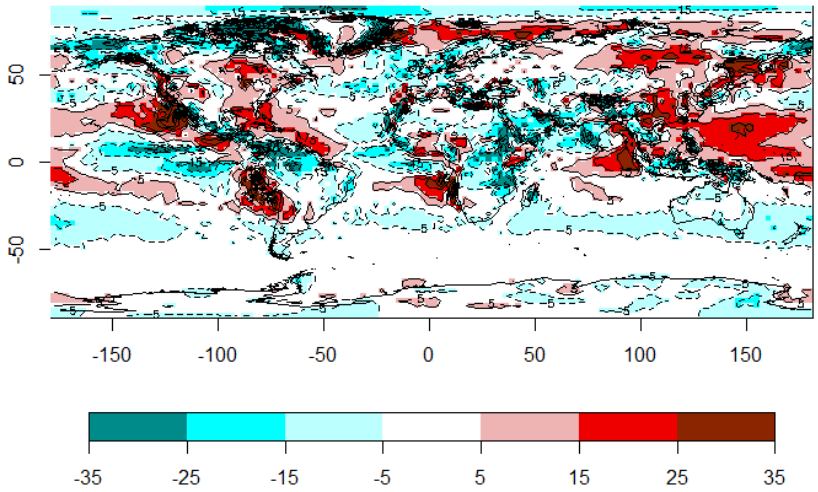

Figure 4.14. The top of atmosphere (TOA) balance anomaly for the standard run of CanAM4, in W/M^2.

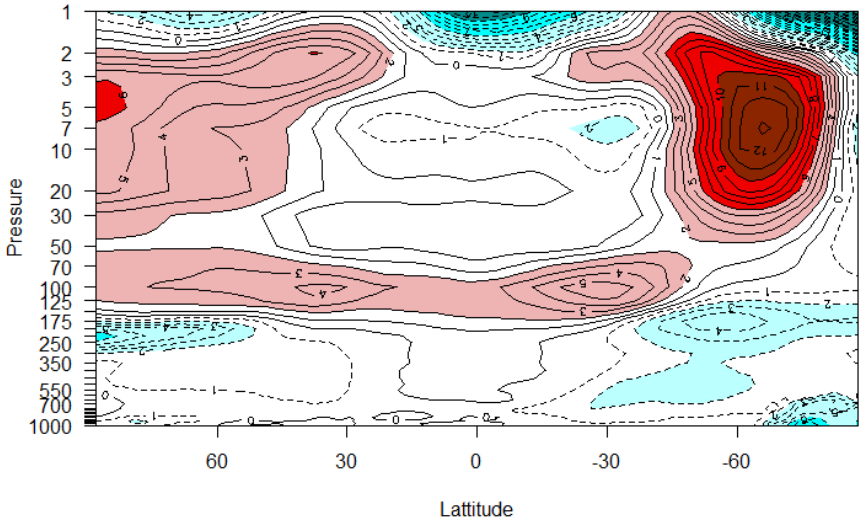

Figure 4.15. The vertical air temperature anomaly for the standard run of CanAM4, in Kelvin.

regions with strong negative biases lie close to the Equator, in the Pacific, South America and eastern Africa.

The anomaly for TA shows a clearer pattern. There are strong warm biases at high latitudes in both hemispheres, although this bias is much larger and spread across more pressure levels in the Southern Hemisphere. This warm bias is found at high altitudes. Lower in the atmosphere, there are some cold biases, again towards the poles. Around the Equator, the model is generally closest to the observations: there is a small warm bias around 100hPa, with nothing else until a large cold bias at 1hPa.

These types of biases are also observed for the other output fields of this model, and for most global climate models. By not accurately representing the present-day observations, projections made using these models may not necessarily be meaningful, hence careful tuning, with the aim of reducing model biases across all output fields simultaneously, of the input parameters is required, and is an active area of research amongst climate modellers (Hourdin et al., 2016).

4.5.3. Reconstructing the CanAM4 observations

As with the toy function, we calculate the SVD basis for each of the output fields of CanAM4. For certain outputs, including RTMT and TA, there are 62 ensemble members, from a wave 1 space-filling design of $\mathcal{X}$. For CLTO, there are additional runs from another wave of runs, so that there are a total of 119 model runs for this output. Using the truncated SVD basis, the reconstruction of the observations for each of these fields can be found, to demonstrate that the inability of accurately reconstructing the observations is not solely a feature of the chosen toy function.

It is possible to study the VarMSE plot for each of the reconstructions. At this stage, there are not known observation errors or discrepancies for any of the output fields. Therefore, the only available choice for $\mathbf{W}$ is a multiple of the identity matrix, so that the error in each grid box is weighted equally. The multiple is set in the same way as for ORCA2, with the cases where the white coloured areas are treated as 1 standard deviation and 3 standard deviations both considered. For both CLTO and RTMT, errors of ± 5 are deemed acceptable, and hence for each of these, $\mathbf{W} = \boldsymbol{\Sigma}_{\mathbf{e}} = \frac{25}{9}\boldsymbol{\mathcal{I}}_l$ (3 standard deviations)

and $\mathbf{W} = \mathbf{\Sigma_e} = 25\boldsymbol{\mathcal{I}}_l$ (1 standard deviation) to reflect this. For TA, anomalies of $\pm 2°$C are coloured white, and hence we set $\mathbf{W} = \mathbf{\Sigma_e} = \frac{4}{9}\boldsymbol{\mathcal{I}}_l$ (3 standard deviations) and $\mathbf{W} = \mathbf{\Sigma_e} = 4\boldsymbol{\mathcal{I}}_l$ (1 standard deviation). Given these specifications for the error, and assuming a zero discrepancy due to a lack of knowledge about this term, the VarMSE plots again show whether the SVD basis will be suitable for searching for $\mathbf{z}$. They also identify whether there is a large reduction in the error between the truncated basis and the full basis, showing whether there are important patterns contained within the lower-eigenvalue basis vectors.

The anomalies for the reconstructions of the observations for TA, CLTO and RTMT are shown in the left half of Figures 4.16, 4.17 and 4.18 respectively. On the right are the associated VarMSE plots for the SVD basis with $\mathbf{W} = \mathbf{\Sigma_e}$, for both the 3 standard deviation case (solid lines) and the 1 standard deviation case (dotted lines).

For the reconstruction of the TA observations (Figure 4.16), the truncated basis consists of the first 7 SVD basis vectors, explaining 95% of the ensemble variability. The remainder of the SVD basis offers an improvement in the reconstruction error, as seen by the VarMSE plot. For $\mathbf{W} = \frac{4}{9}\boldsymbol{\mathcal{I}}_l$, the reconstruction with the full SVD basis is ruled out. When the larger specification of the error variance is used, the truncated SVD basis still rules out $\mathbf{z}$. However, by including more basis vectors, the line representing the reconstruction error goes below the history matching bound, and the reconstruction of $\mathbf{z}$ would not be ruled out. This is not particularly useful, as 35 basis vectors are required, and it is not practical to emulate the coefficients for all of these vectors. However, under this error variance specification, it demonstrates that the SVD basis does contain patterns that allow us to not rule out $\mathbf{z}$, albeit in low-eigenvalue basis vectors.

In the reconstruction with the truncated basis (Figure 4.16), the same warm bias observed in the southern hemisphere for the standard model run is seen here. The general patterns here are similar, with a decrease in altitude for the location of one region of warm bias at the Equator. The warm bias at the North Pole is less extreme here than for the standard run, but there are large biases when this basis is used for projection.

For CLTO, the first 11 SVD basis vectors are required to explain at least 90% of the ensemble variability. To explain 95% here, 31 basis vectors would be needed, therefore the basis is truncated after 11. This ensures that emulation is not overly time consuming,

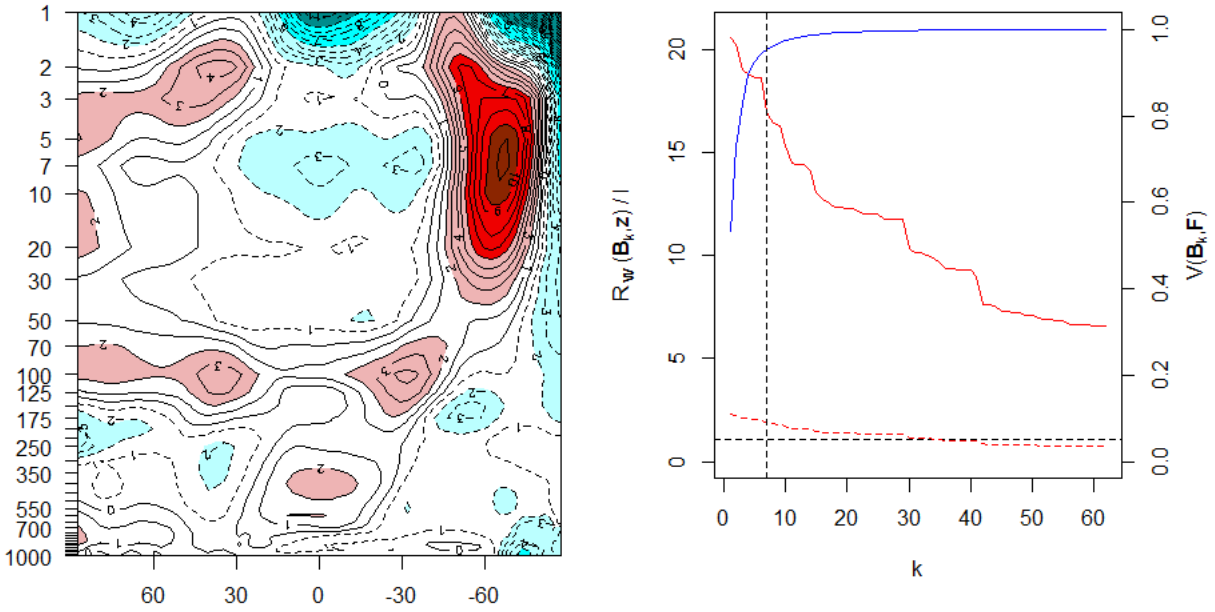

Figure 4.16. The anomaly between the reconstruction of the TA observations and the observations themselves, when the first 7 SVD basis vectors are used for projection, and the VarMSE plot for this field with $\mathbf{W} = \frac{4}{9}\boldsymbol{\mathcal{I}}_l$ (solid line) and $\mathbf{W} = 4\boldsymbol{\mathcal{I}}_l$ (dotted line).

and also reflects the difficulty in emulating the coefficients for basis vectors explaining low percentages of the ensemble. Regardless of where the truncation occurs, $\mathbf{z}$ is ruled out, for either choice of $\mathbf{W}$, although the VarMSE plot shows that it is possible to improve reconstructions by including later SVD basis vectors. The reconstruction of the observations with the first 11 basis vectors has some features that are superior to the standard run, with a decreased positive bias in the North Atlantic Ocean. However, this is compensated for by larger biases around the Equator, and over Australia, and the reconstruction is generally completely different from the observations themselves.

For RTMT (Figure 4.18), the anomaly field for the reconstruction is generally better than that for the standard model run, with the positive biases in the western Pacific reduced substantially. There are also improvements over the standard run observed in the Southern Ocean. Despite these improvements, there is a large difference between the reconstruction and the observations. For this output field, the first 11 SVD basis vectors were used for the reconstruction. This basis explains 90% of the ensemble, with the truncation occurring at this level again because 23 basis vectors are required to explain 95%. The VarMSE plot shows that after the truncation, there is a slight improvement made by adding more basis vectors, but that $\mathbf{z}$ would still be ruled out by the full SVD basis, again for either specification of $\mathbf{W}$.

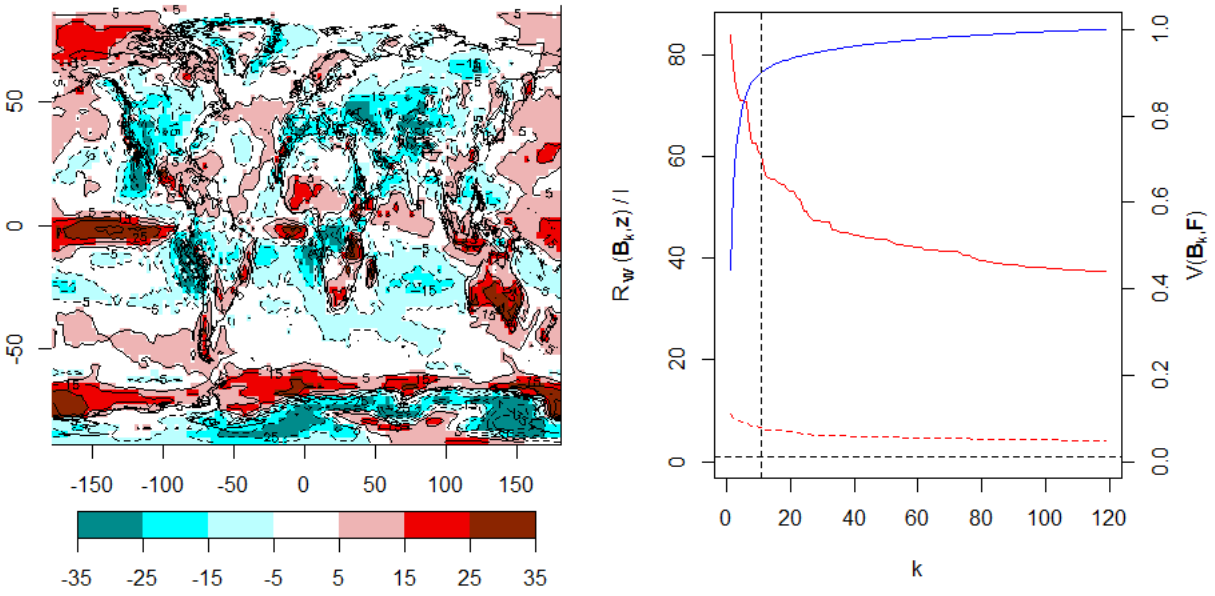

Figure 4.17. The anomaly between the reconstruction of the CLTO observations and the observations themselves, when the first 11 SVD basis vectors are used for projection, and the VarMSE plot for this field with $\mathbf{W} = \frac{25}{9}\boldsymbol{\mathcal{I}}_l$ (solid line) and $\mathbf{W} = 25\boldsymbol{\mathcal{I}}_l$ (dotted line).

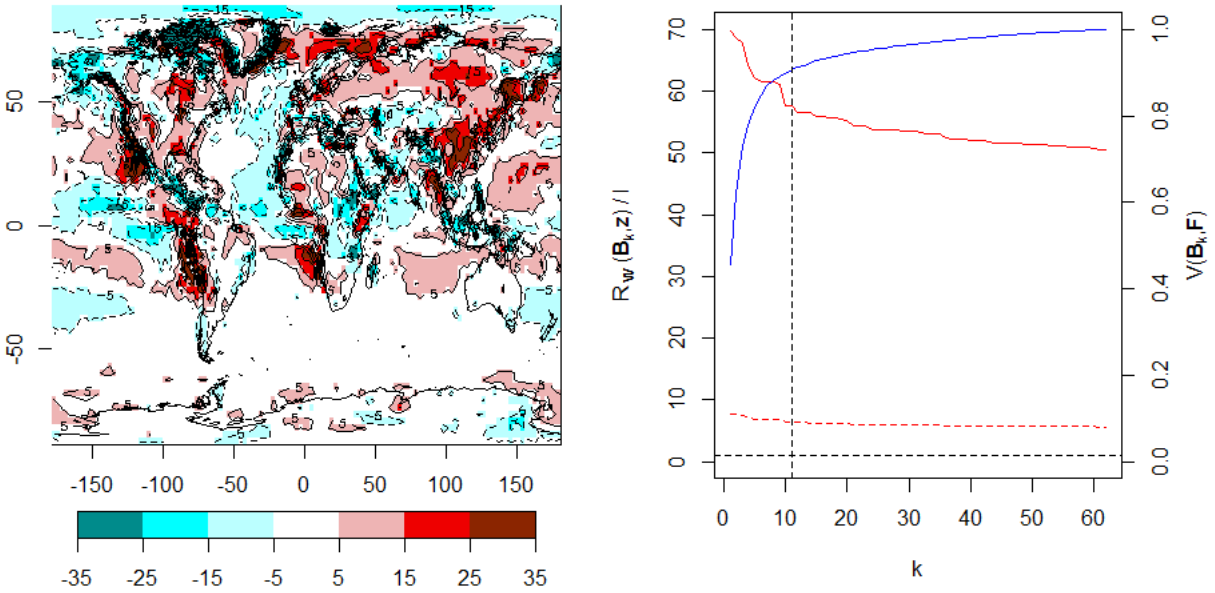

Figure 4.18. The anomaly between the reconstruction of the RTMT observations and the observations themselves, when the first 11 SVD basis vectors are used for projection, and the VarMSE plot for this field with $\mathbf{W} = \frac{25}{9}\boldsymbol{\mathcal{I}}_l$ (solid line) and $\mathbf{W} = 25\boldsymbol{\mathcal{I}}_l$ (dotted line).

For each of the outputs considered here, the above plots have shown the inability to reconstruct $\mathbf{z}$ using the truncated SVD basis, despite this basis capturing the majority of the ensemble variability. This demonstrates that the inability to accurately reconstruct the observations is not only an issue for toy functions, and the same problems can be expected to arise in history matching and calibration for climate models. Ideally, these exercises would be carried out for this model, given some appropriate specification of the

variances. However, it would not be possible to verify the accuracy of results without running the model again. Due to the extremely limited ability to run new ensembles, and the expense of this, it would not be prudent to run the model when it is expected that the results will not be improvements on the current ensemble, and given the conclusions made from calibrating the toy function with the SVD basis, this is exactly what would be expected here. Once an alternative method for selecting a suitable basis has been described, history matching for CanAM4 will be revisited (see Section 6.3).

The inability to reconstruct the observations using the SVD basis, and the effect this has on the outcome of calibration or history matching, motivates answering the question of how the choice of basis can be improved to overcome the problems outlined here.

4.6. Constructing a physical basis

For the toy example, it is fortunate that the clearly important pattern is observed as one of the low-eigenvalue SVD basis vectors, so that it is easy to observe that adding this basis vector improves reconstructions of $\mathbf{z}$ dramatically. This is due to the construction of the example rather than a feature of the SVD basis, and is unlikely to be the case in applications. This pattern may be hidden as a linear combination of several low-eigenvalue basis vectors, and hence be impossible to identify from the basis by eye, or may not appear in the SVD basis in any form.

Therefore, rather than solely considering the SVD basis, and hence the ensemble data, for projection, basis vectors may instead be selected differently. The computer model is likely to represent a physical system, hence there may be important physical patterns that are known, and it may be useful to include these in any basis. The problem of basis selection then becomes one of expert elicitation. Alternatively, given that $\mathbf{z}$ is already known, this vector could be chosen as one of the basis vectors.

Using only a set of chosen patterns as the basis may lead to a basis that is not representative of the variability in the ensemble. Even if it is known that the ensemble is very different from the observations, as has been demonstrated with the toy function, it is important to incorporate the ensemble information into the basis: maintaining ensemble signal is required so that when the basis is projected onto, informative emulators can be built for

the coefficients.

Therefore, given a set of chosen patterns $\mathbf{B}_p = (\mathbf{b}_1, \ldots, \mathbf{b}_p)$, the basis may need to be 'completed', so that enough of the ensemble can be explained. This is done here by defining the 'ensemble residual' and the 'residual basis'.

4.6.1. The residual basis

To define the ensemble residual $\mathbf{F}_\epsilon$ for a given basis $\mathbf{B}_p$, the ensemble is projected onto the basis, and then back-projected to the original field:

$$\mathbf{F}_\epsilon = \mathbf{F}_{\boldsymbol{\mu}} - \mathbf{B}_p(\mathbf{B}_p^T\mathbf{B}_p)^{-1}\mathbf{B}_p^T\mathbf{F}_{\boldsymbol{\mu}} \tag{4.13}$$

This gives the residual for each ensemble member in the respective columns of $\mathbf{F}_\epsilon$, and represents the variability that is not explained by the chosen basis $\mathbf{B}_p$. Using the ensemble residual, this remaining variability may be accounted for with the right singular vectors from the SVD of this residual. The basis from this calculation is defined as the 'residual basis', denoted $\mathbf{B}_\epsilon$. The dimension of this basis is $l \times (n-1)$, and it has orthogonal columns by construction. However, the final p columns explain none of the variation in the residual ensemble (if the chosen basis vectors lie in the subspace defined by the ensemble), and hence have eigenvalues equal to zero, so that these basis vectors can be discarded, leaving the first $(n-1-p)$ vectors of the residual SVD basis.

Combining the residual basis with the defined patterns gives a basis with $n-1$ vectors, if the final p non-informative basis vectors from $\mathbf{B}_\epsilon$ are removed:

$$\mathbf{B} = (\mathbf{B}_p, [\mathbf{B}_\epsilon]_{n-1-p})$$

Defining the basis using this method satisfies two important goals of basis selection: all of the ensemble variability is explained, and patterns that are considered to be important, whether physically or otherwise, are included. If the initial patterns have been selected carefully, then this basis should offer an improvement over the SVD basis in terms of the ability to reconstruct the observations.

For emulation, this basis is then truncated in the usual manner, with the first q basis

vectors of $\mathbf{B}$ selected, giving the basis $\mathbf{B}_q$ where $\mathcal{V}(\mathbf{B}_q, \mathbf{F}_{\boldsymbol{\mu}}) > v_{tot}$, where v_{tot} is the proportion of ensemble variability to be explained (often $v_{tot} = 0.95$). Unless the user-defined basis vectors contain patterns similar to the ensemble, this truncated basis will consist of all of the defined patterns, with the first few residual basis vectors added so that enough variability is explained.

4.6.2. Orthogonality of the basis

By construction, the basis selected through performing SVD on the residual ensemble, $\mathbf{B}_{\boldsymbol{\epsilon}}$, is orthogonal. Regardless of the orthogonality of the selected vectors in $\mathbf{B}_p$, it can be shown that the residual basis is orthogonal to $\mathbf{B}_p$.

Result. *The residual basis $\boldsymbol{B}_{\boldsymbol{\epsilon}}$ is orthogonal to $\boldsymbol{B}_p = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_p)$.*

Proof. First, we show that the columns of the residual ensemble are each orthogonal to the columns of $\mathbf{B}_p$:

$$\begin{aligned}
\mathbf{B}_p^T \mathbf{F}_{\boldsymbol{\epsilon}} &= \mathbf{B}_p^T (\mathbf{F}_{\boldsymbol{\mu}} - \mathbf{B}_p (\mathbf{B}_p^T \mathbf{B}_p)^{-1} \mathbf{B}_p^T \mathbf{F}_{\boldsymbol{\mu}}) \\
&= \mathbf{B}_p^T \mathbf{F}_{\boldsymbol{\mu}} - (\mathbf{B}_p^T \mathbf{B}_p)(\mathbf{B}_p^T \mathbf{B}_p)^{-1} \mathbf{B}_p^T \mathbf{F}_{\boldsymbol{\mu}} \\
&= \mathbf{B}_p^T \mathbf{F}_{\boldsymbol{\mu}} - \mathbf{B}_p^T \mathbf{F}_{\boldsymbol{\mu}} \\
&= \mathbf{0}
\end{aligned}$$

The result is a $p \times n$ zero matrix, so that each of the vectors of $\mathbf{B}_p$ are orthogonal with the vectors of $\mathbf{F}_{\boldsymbol{\epsilon}}$. Using this result, and the definition of the singular value decomposition of $\mathbf{F}_{\boldsymbol{\epsilon}}$, we show that the residual basis, $\mathbf{B}_{\boldsymbol{\epsilon}}$, is orthogonal to the chosen basis, $\mathbf{B}_p$:

$$\begin{aligned}
\mathbf{F}_{\boldsymbol{\epsilon}}^T &= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \\
\implies \mathbf{F}_{\boldsymbol{\epsilon}}^T &= \mathbf{U}\boldsymbol{\Sigma}\mathbf{B}_{\boldsymbol{\epsilon}}^T
\end{aligned}$$

where $\mathbf{U}$ is an orthonormal $n \times n$ matrix, $\boldsymbol{\Sigma}$ is a diagonal $n \times n$ matrix, and $\mathbf{V} = \mathbf{B}_{\boldsymbol{\epsilon}}$ is an $l \times n$ matrix with orthonormal columns, so that $\mathbf{B}_{\boldsymbol{\epsilon}}^T \mathbf{B}_{\boldsymbol{\epsilon}} = \boldsymbol{\mathcal{I}}_n$. Taking the transpose and

multiplying on the right by $\mathbf{U}$ gives

$$\implies \mathbf{F}_{\boldsymbol{\epsilon}} = \mathbf{B}_{\boldsymbol{\epsilon}}\boldsymbol{\Sigma}^T\mathbf{U}^T$$

$$\implies \mathbf{F}_{\boldsymbol{\epsilon}}\mathbf{U} = \mathbf{B}_{\boldsymbol{\epsilon}}\boldsymbol{\Sigma}^T\mathbf{U}^T\mathbf{U}$$

$$\implies \mathbf{B}_p^T\mathbf{F}_{\boldsymbol{\epsilon}}\mathbf{U} = \mathbf{B}_p^T\mathbf{B}_{\boldsymbol{\epsilon}}\boldsymbol{\Sigma}^T$$

where the orthonormality of $\mathbf{U}$ has been used, and the equation has been multiplied on the left by $\mathbf{B}_p^T$. From above, $\mathbf{B}_p^T\mathbf{F}_{\boldsymbol{\epsilon}} = \mathbf{0}$, and hence

$$\implies \mathbf{B}_p^T\mathbf{B}_{\boldsymbol{\epsilon}}\boldsymbol{\Sigma}^T = \mathbf{0}$$

$\mathbf{B}_{\boldsymbol{\epsilon}}$ has dimension $l \times n$, and $\boldsymbol{\Sigma}$ has dimension $n \times n$, due to the dimension of the residual ensemble. However, the final $(p+1)$ eigenvalues contained on the diagonal of $\boldsymbol{\Sigma}$ are zero (due to the degrees of freedom removed by the ensemble mean and the p chosen patterns), and hence only the first $(n-p-1)$ columns of $\mathbf{B}_{\boldsymbol{\epsilon}}$ are of interest, as no ensemble variability is explained by the final $(p+1)$ columns. Discarding columns associated with zero eigenvalues (as the basis would always be truncated before zero-eigenvalue vectors are included), and using that $\boldsymbol{\Sigma}$ is diagonal, the final step is

$$\implies \mathbf{B}_p^T\mathbf{B}_{\boldsymbol{\epsilon}} = \mathbf{0}$$

and hence the vectors of the residual basis are orthogonal to the chosen basis vectors, as expected. □

This is an attractive feature of defining the residual basis using this method. The first few basis vectors from the residual basis can be combined with the selected patterns so that enough of the variability in the ensemble is explained, giving a truncated orthogonal basis dependent on some important patterns and the ensemble. With careful selection of $\mathbf{B}_p$, this should be an improvement over the SVD basis, derived only from the ensemble.

Hereafter, the residual basis $\mathbf{B}_\epsilon$ is defined as the first $(n-p-1)$ vectors from the singular value decomposition of the residual ensemble, $\mathbf{F}_{\boldsymbol{\epsilon}}$ (given in (4.13)).

4.6.3. Imposing orthogonality on a basis

It may not be desirable that the chosen basis vectors are orthogonal with each other if they are based on physical, real-life patterns, for interpretability. Orthogonality may be imposed on a general set of basis vectors using the Gram-Schmidt process (Björck, 1967). Orthogonality is a desirable property for a basis, as if the basis is non-orthogonal, the coefficients for fields may vary dependent on the number of basis vectors used to calculate them. For an orthogonal basis, the coefficients for the projection of a field onto a basis remain fixed for each basis vector, regardless of whether $\mathbf{B}_q$ or $\mathbf{B}_n$ was used to calculate it.

Given a set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_q$, an orthogonal basis is calculated as follows:

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{b}_1 \\ \mathbf{v}_2 &= \mathbf{b}_2 - \frac{\langle \mathbf{v}_1, \mathbf{b}_2 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle} \mathbf{v}_1 \\ &\vdots \\ \mathbf{v}_k &= \mathbf{b}_k - \sum_{j=1}^{k-1} \frac{\langle \mathbf{v}_j, \mathbf{b}_k \rangle}{\langle \mathbf{v}_j, \mathbf{v}_j \rangle} \mathbf{v}_j \end{aligned} \tag{4.14}$$

where the inner product $\langle \cdot, \cdot \rangle$ is defined as

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T \mathbf{b} \tag{4.15}$$

These vectors are then normalised to give an orthonormal basis, $\mathbf{\Gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q)$, where

$$\boldsymbol{\gamma}_k = \frac{\mathbf{v}_k}{\sqrt{\langle \mathbf{v}_k, \mathbf{v}_k \rangle}}$$

As it has already been shown that there is always orthogonality between the initial basis vectors and the residual basis, this operation need only be performed on the initial selected basis. Gram-Schmidt can be computationally expensive in high dimensions if there are a large number of vectors to be orthogonalised. There are unlikely to be many chosen basis vectors in this setting, as we only have a limited number of degrees of freedom (ensemble members n), and also because eliciting even a small number of important patterns may be difficult, especially when the output dimension is large. Therefore, this operation is reasonably fast.

The above recursive formula may be rewritten using matrices (Björck, 1994):

$$\mathbf{B} = \mathbf{\Gamma R}$$

where $\mathbf{B}$ contains the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_q$, $\mathbf{\Gamma}$ is the $l \times q$ basis containing the normalised, orthogonal vectors $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_q$, and $\mathbf{R}$ is a $q \times q$ upper-triangular matrix relating the two bases. This shows that the j^{th} new basis vector is a linear combination of the first j basis vectors of the original basis.

Although the projections of a general l-dimensional vector $f(\cdot)$ will change due to the imposed orthogonality, the reconstructions of the vector are identical, regardless of whether the first k vectors of the original basis, $\mathbf{B}$, or the first k vectors of the orthogonal basis, $\mathbf{\Gamma}$, are used. This is shown here for $k = q$, but is true for any $k < q$, with bases $\mathbf{B}_k$, $\mathbf{\Gamma}_k$ and upper-triangular matrix $\mathbf{R}_{kk}$, where kk denotes taking the upper $k \times k$ section of the matrix.

Using $\mathbf{\Gamma}^T\mathbf{\Gamma} = \boldsymbol{\mathcal{I}}_k$ because $\mathbf{\Gamma}$ consists of orthonormal columns by construction, the reconstruction of $f(\cdot)$ with the original basis $\mathbf{B}$ is

$$\begin{aligned}
\mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T f(\mathbf{x}) &= \mathbf{\Gamma R}((\mathbf{\Gamma R})^T\mathbf{\Gamma R})^{-1}(\mathbf{\Gamma R})^T f(\mathbf{x}) \\
&= \mathbf{\Gamma R}(\mathbf{R}^T\mathbf{\Gamma}^T\mathbf{\Gamma R})^{-1}\mathbf{R}^T\mathbf{\Gamma}^T f(\mathbf{x}) \\
&= \mathbf{\Gamma R}(\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{\Gamma}^T f(\mathbf{x})
\end{aligned}$$

Then, using that $\mathbf{R}$ is invertible and the identity $(\mathbf{CD})^{-1} = \mathbf{D}^{-1}\mathbf{C}^{-1}$ for square matrices $\mathbf{C}$ and $\mathbf{D}$,

$$\begin{aligned}
&= \mathbf{\Gamma R R}^{-1}\mathbf{R}^{-T}\mathbf{R}^T\mathbf{\Gamma}^T f(\mathbf{x}) \\
&= \mathbf{\Gamma\Gamma}^T f(\mathbf{x}) \\
&= \mathbf{\Gamma}(\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T f(\mathbf{x})
\end{aligned} \tag{4.16}$$

i.e. the reconstruction using the new basis $\mathbf{\Gamma}$. This can alternatively be proved by showing that both bases span the same k-dimensional subspace, and using that a basis allows a general field to be written as a unique linear combination of the basis vectors (Kuttler, 2012).

Given that the reconstructions are the same, we also have equality for the reconstruction

errors: for $q = 1, \ldots, (n-1)$,

$$\mathcal{R}_{\mathbf{W}}(\mathbf{B}_q, \mathbf{z}) = \mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q, \mathbf{z})$$

Therefore, the reconstruction and reconstruction error of $\mathbf{z}$, or of any general field, are the same, regardless of whether the original basis, or the orthogonalised version is used. This means that orthogonalisation occurs without having any effect on the quality of reconstructions, and Gram-Schmidt can hence be applied as a tool to allow for the convenience of fixed coefficients for use in emulation.

4.6.4. Proportion of ensemble variability explained

If we use a basis with a set of chosen patterns included, there is no longer a matrix of eigenvalues directly associated with the basis vectors. However, the formulation of the proportion of variance explained using the eigenvalues (in (4.12)) is equivalent by definition to the sum of squares explained by the basis, divided by the total sum of squares. Therefore, the proportion of ensemble variability explained by a basis $\mathbf{B}_q$ can be written as

$$\mathcal{V}(\mathbf{B}_q, \mathbf{F}) = \frac{\sum_{i=1}^{l}\sum_{j=1}^{n}[\mathbf{B}_q(\mathbf{B}_q^T\mathbf{B}_q)^{-1}\mathbf{B}_q^T\mathbf{F}]_{ij}^2}{\sum_{i=1}^{l}\sum_{j=1}^{n}\mathbf{F}_{ij}^2} \tag{4.17}$$

The numerator gives the reconstruction of the ensemble using the basis $\mathbf{B}_q$, and then squares every individual element of this $l \times n$ matrix, and sums these squares. The denominator squares every entry of $\mathbf{F}$ and sums these.

We similarly define the variance explained by projection onto a single basis vector $\mathbf{b}_k$ ($k \leq q$) as

$$\mathcal{V}_k(\mathbf{B}_q, \mathbf{F}) = \frac{\sum_{i=1}^{l}\sum_{j=1}^{n}[\mathbf{b}_k(\mathbf{b}_k^T\mathbf{b}_k)^{-1}\mathbf{b}_k^T\mathbf{F}]_{ij}^2}{\sum_{i=1}^{l}\sum_{j=1}^{n}\mathbf{F}_{ij}^2} \tag{4.18}$$

When $\mathbf{B}$ is orthogonal, we have

$$\mathcal{V}(\mathbf{B}_q, \mathbf{F}) = \sum_{k=1}^{q}\mathcal{V}_k(\mathbf{B}_q, \mathbf{F})$$

To show this, we ignore the common denominator, and prove that the matrix $\mathbf{B}_q(\mathbf{B}_q^T\mathbf{B}_q)^{-1}\mathbf{B}_q^T\mathbf{F}$ can be written in terms of the columns of $\mathbf{F}$. Using that $\mathbf{B}$ is orthogonal, we can immedi-

ately simplify this expression:

$$\mathbf{B}_q(\mathbf{B}_q^T\mathbf{B}_q)^{-1}\mathbf{B}_q^T\mathbf{F} = \mathbf{B}_q\mathbf{B}_q^T\mathbf{F}$$

and by writing $\mathbf{B}$ in terms of its columns $\mathbf{b}_k$, the j^{th} column of this matrix is

$$[\mathbf{B}_q\mathbf{B}_q^T\mathbf{F}]_{\cdot j} = \sum_{k=1}^{q} \mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j)$$

That is, it can be decomposed into a sum of terms dependent on the individual basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_q$ (Jolliffe, 2002). Squaring and summing the elements of this j^{th} column gives

$$(\sum_{k=1}^{q} \mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j))^T(\sum_{m=1}^{q} \mathbf{b}_m\mathbf{b}_m^T f(\mathbf{x}_j)) = \sum_{k=1}^{q} f(\mathbf{x}_j)^T\mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j)$$

because when $k \neq m$,

$$\begin{aligned}(\mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j))^T(\mathbf{b}_m\mathbf{b}_m^T f(\mathbf{x}_j)) &= f(\mathbf{x}_j)^T\mathbf{b}_k\mathbf{b}_k^T\mathbf{b}_m\mathbf{b}_m^T f(\mathbf{x}_j)\\ &= 0\end{aligned}$$

by the orthogonality of the columns of $\mathbf{B}$. By rewriting the numerator of (4.18), this is the same as in the definition of $\mathcal{V}_k(\mathbf{B}_q, \mathbf{F})$ if the columns of $\mathbf{F}$ are summed over:

$$\begin{aligned}\sum_{j=1}^{n}(\sum_{k=1}^{q} f(\mathbf{x}_j)^T\mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j)) &= \sum_{j=1}^{n}(\sum_{k=1}^{q}(\mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j))^T(\mathbf{b}_k\mathbf{b}_k^T f(\mathbf{x}_j))\\ &= \sum_{k=1}^{q}(\sum_{i=1}^{l}\sum_{j=1}^{n}[\mathbf{b}_k\mathbf{b}_k^T\mathbf{F}]_{ij}^2)\end{aligned}$$

This does not hold when $\mathbf{B}$ is not orthogonal, and hence defining the variance explained by a single basis vector requires an extra step. If $\mathbf{\Gamma}$ is the orthogonal basis obtained by performing Gram-Schmidt on non-orthogonal $\mathbf{B}$, then, for any $q = 1, \ldots, (n-1)$:

$$\mathcal{V}(\mathbf{B}_q, \mathbf{F}) = \mathcal{V}(\mathbf{\Gamma}_q, \mathbf{F}) = \sum_{k=1}^{q} \mathcal{V}_k(\mathbf{\Gamma}_q, \mathbf{F})$$

because the reconstruction with $\mathbf{B}_q$ is the same as the reconstruction with $\mathbf{\Gamma}_q$ by (4.16). However, the reconstruction with an individual basis vector (other than the first) is not necessarily the same due to the interaction between the non-orthogonal basis vectors, and the variance in $\mathbf{F}$ can no longer be written as a sum dependent on each basis vector

individually.

4.7. Applications of the physical basis

To attempt to improve the calibration of the toy function, we now apply our methodology for defining a basis using a set of selected patterns and the residual basis. The same ensemble and definitions of the observation error and discrepancy variances as for the application with the SVD basis are used in this section.

Given the knowledge that is available about the toy function, natural choices for $\mathbf{B}_p$ would be the observations themselves, or $\boldsymbol{\varphi}_1$, the basis function most similar to $\mathbf{z}$. This is also extremely similar to the sixth basis vector of the SVD basis, which has already been found to greatly reduce the reconstruction error when it is included in the basis. Both of these basis vectors contain the pattern on the main diagonal, so that reconstructions of $\mathbf{z}$ will be greatly improved compared to the SVD basis $\boldsymbol{\Gamma}_4$. There is however no guarantee that there is enough signal on either of these basis vectors, so that informative emulators can be constructed, and indeed it is the case that emulation on $\boldsymbol{\varphi}_1$ is difficult, so that the observations will be selected as the physical pattern here.

Setting the observations as a basis vector raises the question of whether this constitutes 'double counting': because the aim is to find $\mathbf{z}$ using the basis, it is perhaps not reasonable to include $\mathbf{z}$ exactly in the basis. There are very few directions included in the basis, compared to the dimension of the output, and therefore fields that may not in reality look similar to $\mathbf{z}$ may be forced to appear more similar to $\mathbf{z}$ due to the limited flexibility in this emulation method, and the incorrect regions of $\mathcal{X}$ may be highlighted. Therefore, a pattern that contains certain desirable aspects of $\mathbf{z}$, while not being exactly $\mathbf{z}$, would generally be preferred.

This requires elicitation, and even for the toy function, this is challenging. Various combinations of the main patterns in the output have been set as $\mathbf{B}_p$, with little success in choosing a pattern that has both accurate reconstructions of $\mathbf{z}$ and emulatable coefficients. Without any judgement about what a better, emulatable, pattern might be, the observations are used here to illustrate the method.

4.7.1. Using the observations in the physical basis

The chosen pattern $\mathbf{B}_p$ in this section is the (centred, normalised version of the) observations:

$$\mathbf{B}_p = \frac{\mathbf{z} - \boldsymbol{\mu}}{\sqrt{\langle (\mathbf{z} - \boldsymbol{\mu}), (\mathbf{z} - \boldsymbol{\mu}) \rangle}}$$

using the inner product as defined in (4.15).

To project $\mathbf{z}$ onto this basis, the ensemble mean $\boldsymbol{\mu}$ is first subtracted, and the difference $\mathbf{z} - \boldsymbol{\mu}$ is projected onto $\mathbf{B}_p$. Mapping this coefficient back to the original field gives an exact reconstruction:

$$\begin{aligned}
\mathbf{r}(\mathbf{z}) &= \mathbf{B}_p(\mathbf{B}_p^T\mathbf{B}_p)^{-1}\mathbf{B}_p^T(\mathbf{z} - \boldsymbol{\mu}) + \boldsymbol{\mu} \\
&= \frac{\mathbf{z} - \boldsymbol{\mu}}{\sqrt{\langle (\mathbf{z} - \boldsymbol{\mu}), (\mathbf{z} - \boldsymbol{\mu}) \rangle}} \frac{(\mathbf{z} - \boldsymbol{\mu})^T}{\sqrt{\langle (\mathbf{z} - \boldsymbol{\mu}), (\mathbf{z} - \boldsymbol{\mu}) \rangle}} (\mathbf{z} - \boldsymbol{\mu}) + \boldsymbol{\mu} \\
&= \mathbf{z} - \boldsymbol{\mu} + \boldsymbol{\mu} \\
&= \mathbf{z}
\end{aligned}$$

due to the fact that $\mathbf{B}_p^T\mathbf{B}_p$ is the identity matrix.

While this single basis vector itself can reconstruct $\mathbf{z}$ perfectly, it only explains 28% of the ensemble variability, i.e. $\mathcal{V}_1(\mathbf{B}_p, \mathbf{F}_{\boldsymbol{\mu}}) = 0.28$. Therefore, the ensemble residual and the residual basis are calculated. The first four basis vectors in the residual basis need to be combined with $\mathbf{B}_p$ so that $\mathcal{V}(\mathbf{B}_q, \mathbf{F}_{\boldsymbol{\mu}}) > 0.95$, where $\mathbf{B} = (\mathbf{B}_p, \mathbf{B}_\epsilon)$. The pattern $\mathbf{B}_p$, along with the first three residual basis vectors, are given in Figure 4.19, along with the VarMSE plot for this basis.

The centred and normalised version of the observations, $\mathbf{B}_p$, highlights the difference between the main diagonal and the biased ensemble, with entries with opposite signs in these locations. The leading basis vector of the residual basis shows a similar pattern, although the signs here are the same, with differing magnitudes. The VarMSE plot shows what has already been proved to be true: the chosen physical basis gives zero reconstruction error for $\mathbf{z}$ when used for projection and reconstruction.

Theoretically, this basis should be suitable for calibration and history matching, as it is able to give reconstructions exactly like the observations. Whether or not it is practical is dependent on the quality of emulators that can be built for the coefficients on these basis

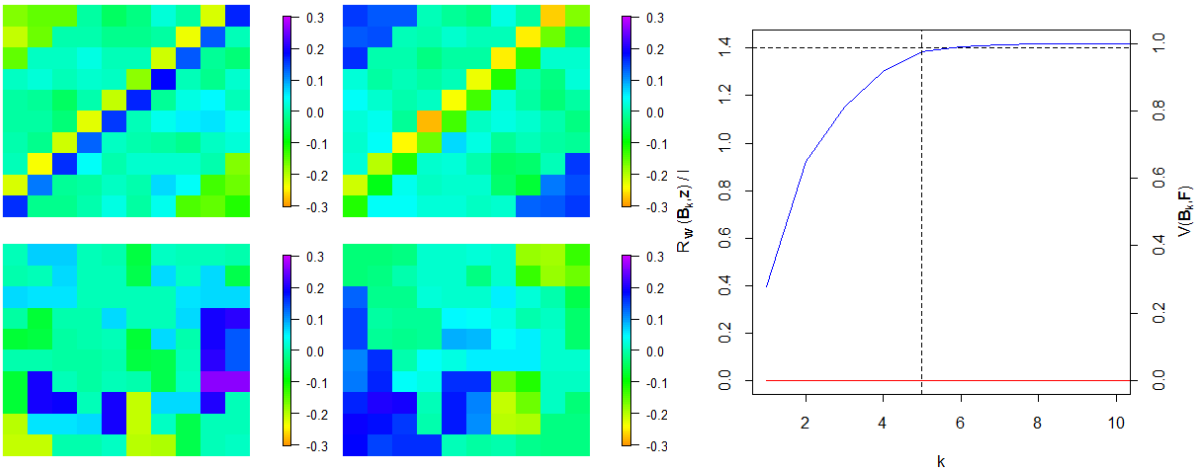

Figure 4.19. The first four basis vectors when a basis is constructed using $\mathbf{z}$ as the first basis vector, and the associated VarMSE plot.

vectors: if accurate emulators cannot be built, then it is likely that the same result as earlier will be found (i.e. no parameter settings exist that give output similar to $\mathbf{z}$). The first basis vector, $\mathbf{B}_p$, in the selected basis has $\mathcal{V}_1(\mathbf{B}_p, \mathbf{F}_{\boldsymbol{\mu}}) = 0.28$, and hence the coefficients on this basis should be emulatable, as there is enough ensemble signal in this direction. A potential problem may lie in the fact that while the spread of ensemble coefficients on $\mathbf{B}_p$ is from -51 to 33, the projection of $\mathbf{z}$ has a coefficient of 69 on this vector. The emulators may be accurate for the observed regions of parameter space, but may struggle for predictive ability towards the true NROY space, either giving large uncertainties or underestimating the coefficients here. The latter of these will lead to emulated fields that are not consistent with $\mathbf{z}$, but due to poor emulation, rather than the inability of the basis to find $\mathbf{z}$, as was the case with $\boldsymbol{\Gamma}_4$ previously.

It is important to note that if emulators are not involved, and history matching is performed theoretically, with the implausibility calculated from the exact known coefficients found by running the toy function, then the resulting space is extremely similar to the true NROY space. We fit Gaussian process emulators to the coefficients on the first five basis vectors in the usual manner. The emulator for each basis vector passes all validation checks, fitting the ensemble well and predicting left-out runs accurately (Appendix B.3).

Using the emulators for the first five basis vectors leads to an NROY space consisting of 41% of the original parameter space, and containing all of the runs that lie in the true NROY space. This is a far superior result to that with the SVD basis, with the best parameter settings correctly not ruled out. This NROY space is far from the size of the true NROY space, suggesting that there is too much emulator uncertainty on predictions.

This may be due to the fact that there is not enough signal in the direction of $\mathbf{z}$, so that while emulated fields that show some signal in this direction can be found, the magnitude of the pattern in $\mathbf{z}$ cannot be reproduced by the emulators. The emulators do, however, include enough uncertainty so that the true coefficients in this direction are not ruled out.

This can be illustrated by considering the emulated field at $\mathbf{x}^*$, shown in Figure 4.20. The signal on the off-diagonal is predicted to be too strong. There is some evidence of the desired pattern here, with the corner pattern faintly observed, although these values are too high. The values on the main diagonal are predicted to be too low.

There is uncertainty on this prediction, however, and assuming that this uncertainty is Normally distributed around the mean prediction, samples can be drawn from this distribution. This does not lead to runs that look exactly like $\mathbf{z}$, but does contain fields with distinctive main diagonals. Compared to the samples from the calibration distribution for $\mathbf{x}^*$ using the SVD basis $\mathbf{\Gamma}_4$, this is at least an improvement, as the main diagonal could not be found in any form within this distribution.

This example suggests that while theoretically this basis works, and emulators can be built as the selected pattern explains enough variability, there is not enough signal in the direction of $\mathbf{z}$ to be able to reproduce the desired output field well. However, the fact that the coefficient in this direction is being emulated, and included within the uncertainty, allows runs that may have more signal in this direction to be not ruled out.

Cutting down $\mathcal{X}$ to this NROY space, keeping runs that may have stronger signal in the direction of $\mathbf{z}$, and hence will be more similar to $\mathbf{z}$, strongly advocates a second wave of history matching. Sampling from this NROY space should allow runs that contain more signal in the correct direction to be included in the wave 2 ensemble. This would then have the effect that there should be more signal from the main diagonal, so that when the ensemble is projected onto this basis, there should be coefficients closer to the coefficient for $\mathbf{x}^*$, allowing more accurate emulators to be built. Refocussed history matching and calibration for the toy function will be carried out in Section 5.6, after a superior method for selecting the basis has been developed.

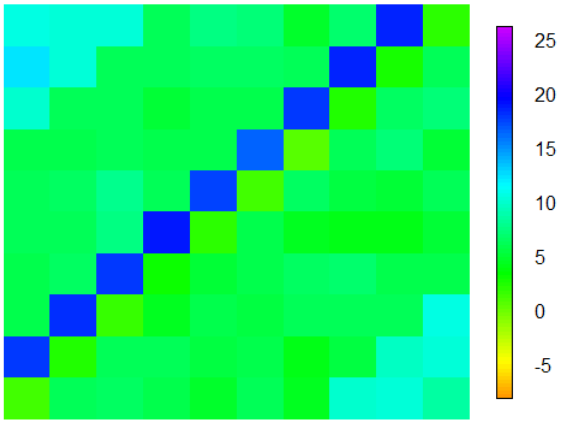

Figure 4.20. The predicted field at $\mathbf{x}^*$, using emulators for the first five basis vectors of the physical basis with the centred and scaled version of $\mathbf{z}$ as the initial pattern.

4.7.2. Selecting a basis for CanAM4

In addition to the problem of double counting, a further problem of setting $\mathbf{z}$ as a basis vector is found using the CanAM4 model.

Setting $\mathbf{z}$ as a pattern in the basis, and then using the residual basis to explain the remaining variability, may be attractive as this information is already available and because this guarantees accurate reconstructions with the basis. This is not practical if the observations are substantially different from the ensemble, as they are in the situation where an alternative basis choice is required. It therefore follows that it is unlikely that there is much ensemble signal to be projected onto $\mathbf{z}$ as a basis vector, rendering emulation challenging or meaningless.

To illustrate this, consider setting the observations for TA as a basis vector. Normalising this vector so that the mean coefficient is zero, and projecting $\mathbf{z}$ onto the basis gives a coefficient of 150.2, which gives a perfect reconstruction of $\mathbf{z}$ if this basis vector is used for reconstruction. This basis vector explains 3.2% of the ensemble variability by itself, and in this case it is possible to build emulators for the coefficient. However, projecting the ensemble onto the normalised observation vector gives coefficients ranging from -37 to 21. Finding the true coefficient for $\mathbf{z}$ is unlikely to be possible due to the large extrapolation required away from the spread in the ensemble. Hence, although in theory the observations can be reconstructed using this basis vector, it is not possible to actually observe the required coefficients. In this scenario, it is not clear whether this implies that it is not

possible to find a coefficient of 150 for any $\mathbf{x}$, or whether there is simply not enough signal in this direction to be able to answer this question.

This suggests that while this basis choice is perfect in terms of reconstructing $\mathbf{z}$, it is not useful for making inferences about $\mathbf{x}^*$. Alternative, automatic methods for selecting a basis may be required, to avoid basis elicitation.

4.8. Discussion

In this chapter, a spatial toy function was introduced, and used to illustrate the potential flaws of using the SVD basis for emulation, calibration and history matching. The SVD basis is an attractive choice, given that it extracts the main modes of variability in the ensemble, and returns an orthogonal basis. Typically, only a small number of vectors are required to explain the majority of the variability, allowing the basis to be truncated, simplifying emulation.

For the defined toy function, it is known that it is possible to find output fields that are close to $\mathbf{z}$, so that a best parameter choice $\mathbf{x}^*$ does exist. However, when performing calibration and history matching using the first four SVD basis vectors, so that over 95% of the variability in the ensemble $\mathbf{F}$ is explained, the results suggest either that there is no $\mathbf{x}^*$, or highlight an incorrect $\mathbf{x}^*$. This suggests that it is possible for this conclusion to be misguided in applications, if the literature default basis has been used for projection and reconstruction.

The fatal flaw for the truncated SVD basis in the case of the toy function was that when $\mathbf{z}$ was projected onto this basis, and then these exact coefficients were used to reconstruct the spatial field, the result appeared to look nothing like $\mathbf{z}$. The choice of basis had removed the distinctive signal from the main diagonal in $\mathbf{z}$, and guaranteed that fields similar to $\mathbf{z}$ could not be reconstructed, even if emulation was perfect (no uncertainty).

The inability to produce accurate reconstructions of $\mathbf{z}$ was explored and quantified further in Section 4.3.3, via VarMSE plots and the reconstruction error, $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}, \mathbf{z})$ (4.8). These quantified the difference between the true $\mathbf{z}$ and the reconstructions with a given basis, and showed that even in the case of perfect emulators, the reconstruction of $\mathbf{z}$ would be

ruled out if the first four vectors of the SVD basis were used for history matching.

Performing calibration and history matching using this basis gave results consistent with the expectations. Calibration was not able to highlight the correct region of parameter space containing runs similar to $\mathbf{z}$, and instead the majority of samples from the posterior gave model output similar to runs in the ensemble, with a strong signal on the off-diagonal. This is unsurprising, given that this is the main pattern that the basis contained. Observing results such as these would likely lead to the incorrect conclusion that there is no $\mathbf{x}^*$.

History matching with this basis ruled out all of parameter space, suggesting that no parameter settings give output consistent with $\mathbf{z}$. Again, this is an incorrect conclusion, as it was showed that regions of parameter space exist where output similar to $\mathbf{z}$ is observed. Neither calibration or history matching was able to overcome the poor basis choice in this example, and suggests that even with excellent emulators, flawed conclusions can be made using the default literature choice (Higdon et al., 2008a, Sexton et al., 2011, Chang et al., 2014a).

For the toy example, this problem could be somewhat mitigated by including the sixth SVD vector, as the reconstruction once this vector was added was found to be within the history matching tolerance to error. However, simply adding more SVD basis vectors is not a general solution to the reconstruction problem. In this example, due to the construction of the toy function with orthogonal basis vectors, there are not many directions of variability in the function output, and so the SVD basis perhaps inevitably highlighted the main diagonal in a vector. This is unlikely to be the case in a general example, and in fact the climate model examples show that while adding in more SVD basis vectors improves reconstructions to some extent, perfect reconstructions are not found. The lack of signal from the ensemble on the sixth SVD basis vector, with only 1% of ensemble variability explained by it, is an argument against including more SVD basis vectors, and we showed that emulation of the coefficients on this basis vector is poor. In the climate examples, important patterns may be in lower-order basis vectors, explaining even less ensemble variability, and may be 'hidden' in linear combinations of many vectors.

Therefore, rather than adding in extra vectors from the SVD basis, a method for defining a basis given a chosen pattern or set of patterns was developed. Given one or more patterns,

the 'residual basis' can be calculated, giving a full orthogonal basis that combines the patterns that are perceived to be important, whether for physical or other reasons, but also that explains the variability in the ensemble. Using this, it is hoped that a basis can be engineered that allows 'good' reconstructions of $\mathbf{z}$, in the sense that the reconstructed field would not be ruled out in the perfect emulator case, while each basis vector also contains enough ensemble signal so that emulators can be built. Fixing a pattern in the basis that gives good reconstructions, but cannot be emulated at all, is not of any use practically.

For the toy example, the centred observations are a good choice for the initial, selected basis. This pattern explains enough ensemble variability so that emulators can be built, and the coefficient of the observations on this basis vector is not 'too far' from the spread of ensemble coefficients, so that it is possible for the emulator to at least contain this true coefficient in prediction intervals. It may be possible to build an accurate emulator for the ensemble coefficients, but if these are all far away from the $\mathbf{z}$ coefficient, then accurate reconstructions may not be possible with the emulators, giving a different problem from before: the basis choice now allows accurate reconstructions, but the emulator does not as there is not enough signal in the direction of $\mathbf{z}$.

In this case, setting the centred observations as the initial basis vector, and combining this with the first few vectors of the residual basis, achieves both of these goals: accurate reconstructions of $\mathbf{z}$, and emulatable coefficients on each basis vector. Using this basis for history matching does not cut down space enough so that it only contains parameter settings that lead to output similar to $\mathbf{z}$, but is able to remove over half of parameter space. This is a substantial improvement over the example with the truncated SVD basis, with all runs that lie in the true NROY space contained within the NROY space found with this new basis.

The fact that a lot of parameter space that leads to output dissimilar to $\mathbf{z}$ has not been ruled out suggests that there is not enough signal in the correct direction to perfectly emulate the coefficients. The emulators for a couple of the coefficients have a great deal of uncertainty on predictions, causing fewer runs to be ruled out than if the coefficients in these directions could be emulated more accurately. Although the key pattern has been included in the basis to guarantee accurate reconstructions theoretically, emulation relies entirely on the ensemble.

This is a clear advocate for a multi-wave or refocussed approach to history matching and calibration. At this wave, some runs that clearly have no signal in the direction of $\mathbf{z}$ have been removed. It is hoped that if a second wave were to be performed, the ensemble sampled from the current NROY space will contain more signal relating to the main diagonal pattern, so that less uncertain emulators can be built for the important directions, and space can be further constrained.

Setting $\mathbf{z}$, or the centred observations, as the initial basis vector, was successful for the toy example, but is not likely to be a generally viable method in applications, as demonstrated by the CanAM4 example. For the TA field, the basis vector $\mathbf{z}$ explained a much smaller percentage of ensemble variability than for the toy example, and the spread of ensemble coefficients was far enough away from the coefficient for $\mathbf{z}$ that emulators were unable to include this coefficient in 99% prediction intervals. Therefore, although this choice of basis leads to perfect reconstructions, the poor emulation leads to fields that would be ruled out, giving the same result as if a poor basis was used. The issue of how this might be double counting was also discussed.

This suggests that in general, setting $\mathbf{B}_p = \mathbf{z}$ does not solve the problem of basis selection, and perhaps should not be allowed regardless. Eliciting other potentially important patterns that could be set as $\mathbf{B}_p$ is challenging in the case of high-dimensional climate model output, and may not be possible. Therefore, rather than selecting a pattern based on a physical argument, an automatic approach for selecting a basis that allows a) accurate reconstructions of $\mathbf{z}$ and b) accurate emulators for the coefficients, would be a superior method, and will be the focus of Chapter 5.

4.9. Conclusion

Even in a perfect model scenario, where the emulators predict the model output with no uncertainty, if the observations do not lie in the subspace of possible reconstructed fields, then any calibration exercise is doomed to fail before it begins. In this chapter, we provided a method for quantifying the reconstruction error of the observations for a given basis, and also developed a method for combining chosen important patterns with ensemble variability, to provide a basis that should be more suited to representing

$\mathbf{z}$ than the truncated SVD basis. We showed that defining a basis in this manner has desirable properties, such as orthogonality, and that imposing orthogonality on a set of non-orthogonal patterns does not affect the reconstruction error.

Before emulators are built using a basis, the reconstruction of $\mathbf{z}$ should be assessed. If this does not lie within the desired tolerance to error on the VarMSE plot, then this basis choice is not suitable, given the current discrepancy. If it does fall below the line, then the basis can be used for emulation and calibration. This is a simple check that can be performed to mitigate against some incorrect conclusions.

The SVD basis generally provides emulatable coefficients, because it is based on the ensemble. Setting a chosen pattern such as the observations in the basis guarantees accurate reconstructions, theoretically. To avoid difficult elicitation of patterns in high-dimensional space, an automatic procedure for selecting a basis is preferred. Finding a basis that minimises the reconstruction error while using signal from the ensemble to ensure emulatable coefficients is explored in the next chapter by considering rotations of the SVD basis.

5. Optimal rotation of a basis

5.1. Introduction

In this chapter, we extend the basis selection ideas from the previous chapter to a new automatic method for finding a suitable basis.

Previously, eliciting a set of patterns $\mathbf{B}_p = (\mathbf{b}_1, \ldots, \mathbf{b}_p)$ to be used in the initial basis, possibly based on physical knowledge, was required to overcome the issue with the SVD basis. This was then combined with the 'residual basis' (Section 4.6.1) to give a basis explaining all of the ensemble variability, while reducing (in comparison to the SVD basis) the reconstruction error for the truncated basis. Selecting patterns may not be suitable for high-dimensional output, especially if there is no prior knowledge about which patterns are important. Even if selecting important patterns is possible, these patterns may contain little signal from the ensemble, and hence building emulators will not be possible, so that the time taken to elicit a pattern would have been wasted.

Setting the observations as a basis vector satisfies the goal of being able to accurately reconstruct the observations using the basis, but often fails at the emulation stage. There are also questions regarding double-counting. Regardless, if it is possible to emulate the coefficients given by projection onto the observations, it is likely that the SVD basis will already contain this signal, and a better basis choice may not be required. A suitable basis for a calibration exercise is one that combines the two goals of representing the observations (through minimising the reconstruction error), while explaining enough of the variability in the ensemble for emulating each truncated basis vector coefficient.

We develop an automatic approach based on rotating the SVD basis, in order to minimise the reconstruction error for the observations on this basis, while selecting basis vectors

that can be emulated using the signal in the ensemble by blending important patterns from the observations with ensemble signal. In general, this could be applied to any given basis, but the focus here is on the SVD (principal component) basis.

Section 5.2 introduces general high-dimensional rotations, and discusses how the SVD basis might be rotated. Section 5.3 outlines key properties of an optimisation criteria, and Section 5.4 defines the final iterative algorithm. Sections 5.5 and 5.6 apply this method to the toy function from the previous chapter. Section 5.7 improves the method for situations where the weight matrix is known, and Section 5.8 continues the iterative history matching of the toy example using this extension.

5.2. Basis rotation

There are a number of existing methods for rotating a principal component or SVD basis $\mathbf{\Gamma}$ of dimension $l \times n$ (or $l \times (n-1)$ if the ensemble mean has been removed, with n used in this section for convenience), as described by Richman (1986) and Jolliffe (2002). These methods can be divided into orthogonal and non-orthogonal (or 'oblique') rotations, and aim to rotate the SVD basis for reasons such as simplicity, or physical interpretability. In general, this rotation is given by

$$\mathbf{\Gamma\Lambda}$$

where $\mathbf{\Lambda}$ is an $n \times n$ rotation matrix (Jolliffe, 2002).

The rotation matrix can be defined using many different criteria, and the choice of how to define $\mathbf{\Lambda}$ is problem dependent. Varimax is a commonly-used orthogonal rotation criteria, that maximises the variance of the coefficients for the ensemble members (Kaiser, 1958), which can be written as (Abdi, 2003)

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{c}_i(\mathbf{x}_j)^2 - \bar{\mathbf{c}}(\mathbf{x})^2)^2$$

where $\mathbf{c}_i(\mathbf{x}_j)$ is the projection of $f(\mathbf{x}_j)$ onto the i^{th} basis vector, and $\bar{\mathbf{c}}(\mathbf{x})^2$ is the mean of the squared coefficients. The rationale behind this rotation is that it is easier to interpret the effects that each basis vector has on each ensemble member if the ensemble member is represented by a small number of large coefficients, with zeros for the other basis vectors.

Other orthogonal rotation methods use similar criteria, such as quartimax rotation, which seeks to explain each ensemble member with as few basis vectors as possible (Neuhaus and Wrigley, 1954), and equamax, which combines varimax and quartimax (Kaiser, 1974). None of these are suitable for the purposes of reconstructing the observations, as they all involve a criteria based solely on explaining the variability in the ensemble.

Instead of requiring an orthogonal rotation, Procrustes target rotation allows any general rotation (Hurley and Cattell, 1962). Given a 'target' basis $\mathbf{B}$, the following equation is solved for $\mathbf{\Lambda}$:

$$\mathbf{B} = \mathbf{\Gamma\Lambda} + \boldsymbol{\epsilon}$$

where the trace of $\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$ is minimised, giving

$$\mathbf{\Lambda} = (\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T\mathbf{B}$$

This method is only suitable when the target basis $\mathbf{B}$ is known. This is similar to the problem outlined in the previous chapter, as $\mathbf{z}$ is essentially the target. However, in this setting, this would only give a single basis vector to represent $\mathbf{z}$ if this was set as the target. Furthermore, even if there were a true underlying basis based on physical effects, this would not necessarily be useful in the application of emulation and calibration, as it is also important for the rotated basis to contain signal from the ensemble.

None of the above rotation methods are suitable for the goal of rotation here, namely to be able to select a basis that minimises the reconstruction error for $\mathbf{z}$, while retaining signal from the ensemble. We develop a novel criteria.

5.2.1. Rotation in n dimensions

In n dimensions, an $n \times n$ rotation matrix $\mathbf{\Lambda}_{ab}$ is defined as (Murnaghan, 1962, Duffin and Barrett, 1994):

$$\mathbf{\Lambda}_{ab}(\theta) = [\lambda_{ij}] = \begin{cases} \lambda_{ii} = 1, \quad i \neq a, i \neq b \\ \lambda_{aa} = cos(\theta) \\ \lambda_{bb} = cos(\theta) \\ \lambda_{ab} = -sin(\theta) \\ \lambda_{ba} = sin(\theta) \\ \lambda_{ij} = 0, \quad \text{otherwise} \end{cases} \tag{5.1}$$

where $\mathbf{\Lambda}_{ab}$ defines a rotation around the plane defined by coordinate axes x_a and x_b by angle θ. That is, all other dimensions remain fixed, with only directions x_a and x_b being rotated.

To achieve a general rotation in n dimensions, rotation matrices around any pair of coordinate axes can be defined in this manner. These individual rotation matrices are then multiplied together to give a general rotation matrix. Murnaghan (1962) shows that any n-dimensional rotation matrix $\mathbf{\Lambda}$ can be found by multiplying $\frac{n(n-1)}{2}$ rotation matrices as defined in (5.1), where a and b take on each possible combination of the dimensions $1, \ldots, n$, to give a rotation matrix $\mathbf{\Lambda}$:

$$\mathbf{\Lambda} = \prod_{a=1}^{n-1} \prod_{b=a+1}^{n} \mathbf{\Lambda}_{ab}$$

The above formulation can be adapted to give a rotation around any simplex defined with coordinates, rather than just the coordinate axes, as in Aguilera and Pérez-Aguila (2004). Here, an algorithm is given for rotation around a simplex, leading to a rotation by angle θ around an $(n-2)$-dimensional subspace. Any number of these rotation matrices can then be composed to give a new rotation matrix as before.

Therefore, it is possible to rotate a basis by defining a set of angles and pairs of coordinate axes. Given the $(l \times n)$-dimensional SVD basis $\mathbf{\Gamma}$, there are two potential ways to proceed with rotation: by multiplying $\mathbf{\Gamma}$ on the left with a rotation matrix, or on the right. Each of these rotations results in a new matrix with the same dimension as $\mathbf{\Gamma}$.

In the case of rotating $\mathbf{\Gamma}$ by multiplying on the left by $\mathbf{\Lambda}$, this rotation matrix needs to have dimension $l \times l$ so that $\mathbf{\Lambda\Gamma}$ still consists of n vectors of length l, for n ensemble members and l-dimensional model output. By defining $\mathbf{\Lambda}$ as an $l \times l$ matrix, the coordinate axes a, b from above are the different grid boxes in the l-dimensional output. To define this matrix, up to $\frac{l(l-1)}{2}$ angles are required to give the rotation for each pair of grid boxes.

Given that l is often large for computer model output, and that the angles that define the rotation are unlikely to have known values, their values must be optimised, based on a chosen criteria for giving the 'best' basis. This results in an extremely challenging optimisation problem: for the CanAM4 output over the longitude-latitude grid in Section 4.5.2, $l = 8192$, so that if a rotation matrix were to be defined for a basis $\mathbf{\Gamma}$ for this model output using this method, 33,550,336 angles would need to be given, assuming that there is no prior knowledge about which dimensions should be rotated around.

If instead $\mathbf{\Lambda}$ is defined as an $n \times n$ matrix, the rotated basis $\mathbf{\Gamma\Lambda}$ has dimension $l \times n$, but instead of the number of angles increasing with the size of the output, it is dependent on the number of ensemble members. This is generally small for computer models with large output, and for the $n = 62$ ensemble members of CanAM4, 1891 angles are needed to give a rotation matrix. While this is still a large number of quantities to be specified, it is orders of magnitude more reasonable than needing to know over 33 million.

Not only is multiplying by a rotation matrix on the right more computationally attractive, the interpretation of this rotation is clearer. Let the i, j^{th} entry of $\mathbf{\Lambda}$ be written as λ_{ij}, and the i, j^{th} entry of $\mathbf{\Gamma}$ be written as γ_{ij}, where $\mathbf{\Gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n)$ contains n basis vectors as its columns. Then, multiplying on the left with a rotation matrix gives a new matrix with

$$(\mathbf{\Lambda\Gamma})_{ij} = \sum_{k=1}^{l} \lambda_{ik}\gamma_{kj}$$

Therefore, the first vector in the rotated basis is

$$\begin{pmatrix} \sum_{k=1}^{l} \lambda_{1k}\gamma_{k1} \\ \vdots \\ \sum_{k=1}^{l} \lambda_{lk}\gamma_{k1} \end{pmatrix}$$

i.e. the new first vector only depends on the first vector of the original basis $\mathbf{\Gamma}$, but requires the entirety of $\mathbf{\Lambda}$. The entries of the first vector of $\mathbf{\Gamma}$ are being re-weighted, and

re-distributed over the l-dimensional vector, dependent on $\mathbf{\Lambda}$. However, this does not make sense. For example, in the ensemble there may be grid boxes that have only zeros in them. Therefore, the basis vectors in $\mathbf{\Gamma}$ will contain zeros for these grid boxes, but rotation in this manner would allow non-zero entries.

If rotation is instead carried out through multiplication on the right, a general entry of the new matrix is given by

$$(\mathbf{\Gamma\Lambda})_{ij} = \sum_{k=1}^{n} \lambda_{kj}\gamma_{ik} \tag{5.2}$$

so that the first vector in the rotated matrix is

$$\begin{pmatrix} \sum_{k=1}^{n} \lambda_{k1}\gamma_{1k} \\ \vdots \\ \sum_{k=1}^{n} \lambda_{k1}\gamma_{lk} \end{pmatrix} \tag{5.3}$$

Now, the j^{th} vector of the rotated basis depends on only the j^{th} column of $\mathbf{\Lambda}$. Furthermore, rather than re-weighting across the first basis vector, this rotation treats each grid box separately, so that the new value for grid box i only depends on the values for grid box i in the original basis. Essentially, instead of rotating around each grid box, this rotation is carried out using the original basis vectors as the coordinate axes.

This is similar to how SVD works, with this being a rotation into a direction that explains variability the best. Rotating the SVD vectors by multiplying by $\mathbf{\Lambda}$ will give new vectors that are no longer optimal in the sense that the most ensemble variability is explained, but may allow a basis to be found that has other important properties, such as the ability to reconstruct $\mathbf{z}$.

Therefore, only rotation via multiplying the basis on the right will be considered. The large saving in terms of the number of angles that are required to define this matrix, combined with the more logical interpretation of the rotation and the fact that it only rotates the SVD basis vectors, make this the preferred method for further exploration.

5.3. Optimising for $\mathbf{\Lambda}$

As introduced in Chapter 4, a natural way of quantifying the suitability of a basis $\mathbf{\Gamma}_q$ for representing $\mathbf{z}$ is by using the reconstruction error, $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q, \mathbf{z})$, for a chosen weight matrix $\mathbf{W}$ (equation (4.8)). The parallel between this norm and the multivariate implausibility was shown for $\mathbf{W} = \mathbf{\Sigma_e} + \mathbf{\Sigma_\eta}$, and hence with this choice of weight matrix, the reconstruction error can be used to identify whether the projection of $\mathbf{z}$ would be ruled out, even if the coefficients were known exactly. Therefore, requiring that

$$\mathcal{R}_{\mathbf{W}}((\mathbf{\Gamma\Lambda})_q, \mathbf{z}) < T = \chi^2_{l,0.995}$$

for the chi-squared history matching bound T from (4.11), is a key feature for the first q vectors of the rotated basis $\mathbf{\Gamma\Lambda}$. If this is not satisfied, $\mathbf{x}^* : f(\mathbf{x}^*) = \mathbf{z}$ would be ruled out using this basis.

Having basis vectors that accurately reconstruct $\mathbf{z}$ is not by itself practically useful if emulators cannot be built for the coefficients given by the projection of the ensemble onto the basis. A second criteria for $(\mathbf{\Gamma\Lambda})_q$ is therefore

$$\mathcal{V}_j((\mathbf{\Gamma\Lambda})_q, \mathbf{F_\mu}) > v_j, \quad j = 1, \ldots, q$$

for proportions $(v_1, \ldots, v_q)$, so that enough signal is found on each of the vectors of the truncated basis. This is problem dependent, but from experience, setting 0.1 for the first few basis vectors gives the best results. If n is larger, this value would likely need to be lower. The first few vectors of the rotated basis are likely to reduce the reconstruction error the most (i.e. they will involve patterns that are informative for $\mathbf{z}$), so that if we are to rule out runs that are dissimilar from $\mathbf{z}$, we want emulators that have predictive ability, and a low uncertainty, in comparison to the spread of possible outputs. Using a lower value than 0.1 can result in large, constant uncertainties across $\mathcal{X}$, not allowing emulators with predictive power to be built, or not allowing any runs to be ruled out based on this basis vector, and hence we would be unable to achieve the goal of ruling out poor model runs. The truncation after q basis vectors can be set in the normal manner, requiring the majority (v_{tot}) of the ensemble variability to be explained by projection onto the basis:

$$\mathcal{V}((\mathbf{\Gamma\Lambda})_q, \mathbf{F_\mu}) > v_{tot}$$

(see Section 4.6.4 for the definitions of $\mathcal{V}(\cdot,\cdot)$ and $\mathcal{V}_j(\cdot,\cdot)$). If a basis can be found satisfying each of these constraints, this basis should be suitable for use in emulation and calibration.

Even when the ensemble size n is relatively small, the problem of optimising for the angles of the rotation matrix is challenging. For example, with the 62 ensemble members for the RTMT and TA fields of CanAM4, $\frac{62\times 61}{2} = 1891$ angles must be defined to find $\mathbf{\Lambda}$. However, simplifications to the problem can be made to overcome this computational issue.

5.3.1. Invariance of $\mathcal{R}_{\mathbf{W}}(\mathbf{B},\cdot)$ to rotation

It can be shown that the reconstruction error using the full basis of size n is invariant to rotation. Rotations are linear operators: applying a rotation does not affect the subspace represented by $\mathbf{B}$ (Rudin, 1976). We prove in our context that if we define the new basis by multiplication on the right, it is not possible for the reconstruction error to be less than the reconstruction error for the full original basis.

Result. *For a basis $\boldsymbol{B}$, which need not be orthogonal, a general $n \times n$ rotation matrix $\mathbf{\Lambda}$, a weight matrix $\boldsymbol{W}$, and a vector of output $f(\boldsymbol{x})$, we have:*

$$\mathcal{R}_{\boldsymbol{W}}(\boldsymbol{B}\mathbf{\Lambda}, f(\boldsymbol{x})) = \mathcal{R}_{\boldsymbol{W}}(\boldsymbol{B}, f(\boldsymbol{x}))$$

Proof. The reconstruction error for the rotated basis can be written as

$$\begin{aligned}\mathcal{R}_{\mathbf{W}}(\mathbf{B\Lambda}, f(\mathbf{x})) &= \|f(\mathbf{x}) - \mathbf{B\Lambda}((\mathbf{B\Lambda})^T\mathbf{B\Lambda})^{-1}(\mathbf{B\Lambda})^T f(\mathbf{x})\|_{\mathbf{W}} \\ &= \|f(\mathbf{x}) - \mathbf{B\Lambda}(\mathbf{\Lambda}^T\mathbf{B}^T\mathbf{B\Lambda})^{-1}\mathbf{\Lambda}^T\mathbf{B}^T f(\mathbf{x})\|_{\mathbf{W}}\end{aligned}$$

By the definition of rotation matrices, $\mathbf{\Lambda}$ is invertible, with $\mathbf{\Lambda}^T = \mathbf{\Lambda}^{-1}$. Furthermore, $\mathbf{\Lambda}^T\mathbf{B}^T\mathbf{B}$ is a square matrix, and is invertible ($\mathbf{B}^T\mathbf{B}$ is non-singular as $\mathbf{B}$ is a basis, so none of the columns are linear combinations of each other, i.e. $\mathbf{B}$ has rank n), so that the

identity $(\mathbf{CD})^{-1} = \mathbf{D}^{-1}\mathbf{C}^{-1}$ for $\mathbf{C}, \mathbf{D}$ both $n \times n$ invertible matrices can be applied:

$$\begin{aligned}
&= \|f(\mathbf{x}) - \mathbf{B}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-1}(\boldsymbol{\Lambda}^T\mathbf{B}^T\mathbf{B})^{-1}\boldsymbol{\Lambda}^T\mathbf{B}^T f(\mathbf{x})\|_{\mathbf{W}} \\
&= \|f(\mathbf{x}) - \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}(\boldsymbol{\Lambda}^T)^{-1}\boldsymbol{\Lambda}^T\mathbf{B}^T f(\mathbf{x})\|_{\mathbf{W}} \\
&= \|f(\mathbf{x}) - \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T f(\mathbf{x})\|_{\mathbf{W}} \\
&= \mathcal{R}_{\mathbf{W}}(\mathbf{B}, f(\mathbf{x}))
\end{aligned}$$

where the same identity has been applied a second time, with $\mathbf{C} = \boldsymbol{\Lambda}^T$ and $\mathbf{D} = \mathbf{B}^T\mathbf{B}$, to give the result. □

The reconstructed field with the full rotated basis is the same as the reconstruction with the original basis, regardless of the rotation that has been applied. By taking linear combinations of the original basis, only basis vectors that define the same n-dimensional subspace can be found using this method.

This at first appears to be a negative, as it shows that this method cannot find a lower reconstruction error than the full SVD basis gives. However, if any general subspace of l-dimensional space was a potential basis, i.e. the multiplication was performed on the left instead, then not only would the optimisation problem be much more difficult, with many more parameters, but also directions of space where there is little or no signal from the ensemble could be explored. Here, the rotation restricts the new basis to the n-dimensional subspace defined by SVD: the subspace where ensemble signal is found.

Therefore, although a basis that reduces the reconstruction error more, or gets 'closer' to $\mathbf{z}$, than the full SVD basis $\boldsymbol{\Gamma}$ cannot be found, in practice the majority of these SVD basis vectors are discarded. Rotating the basis allows important directions, if any, from these discarded, low-eigenvalue basis vectors to be combined in such a way that they are included in the first few new basis vectors. Combining the important directions with patterns that contain more signal allows emulators to be built, and hence gives a trade-off between emulatability and minimising the reconstruction error. This will improve reconstructions compared to the truncated SVD basis.

5.3.2. Rotation as re-weighting

From the equations showing the new basis vectors in terms of the original basis vectors (equations (5.2) and (5.3)) we identify a potential simplification in defining a rotation matrix.

The form of (5.2) shows that in order to define the first column $j = 1$ of the new matrix, i.e. the first new basis vector, only the first column of $\mathbf{\Lambda}$ is used, and applied to the entire SVD basis $\mathbf{\Gamma}$. More generally, in order to define the j^{th} vector of the new basis, n parameters or weights are required, as given by the j^{th} column of $\mathbf{\Lambda}$, and these n parameters are applied to the entire original basis $\mathbf{\Gamma}$. This can be demonstrated using (5.2) and (5.3), where $\mathbf{M}_{\cdot j}$ denotes the j^{th} column of a matrix $\mathbf{M}$:

$$\begin{aligned}(\mathbf{\Gamma\Lambda})_{\cdot j} &= \begin{pmatrix} \sum_{k=1}^{n} \lambda_{kj}\gamma_{1k} \\ \vdots \\ \sum_{k=1}^{n} \lambda_{kj}\gamma_{lk} \end{pmatrix} \\ &= \lambda_{1j} \begin{pmatrix} \gamma_{11} \\ \vdots \\ \gamma_{l1} \end{pmatrix} + \ldots + \lambda_{nj} \begin{pmatrix} \gamma_{1n} \\ \vdots \\ \gamma_{ln} \end{pmatrix} \\ &= \lambda_{1j}\boldsymbol{\gamma}_1 + \ldots + \lambda_{nj}\boldsymbol{\gamma}_n \end{aligned} \tag{5.4}$$

Each column in the rotated basis $\mathbf{\Gamma\Lambda}$ is a linear combination of the columns of $\mathbf{\Gamma}$.

Instead of defining $\mathbf{\Lambda}$ strictly as a rotation matrix as in (5.1), the problem may be treated as one of selecting appropriate values to be contained within $\mathbf{\Lambda}$, rather than setting angles for every combination of basis vectors. This does not immediately appear to be a simplification. In fact, this would require the n^2 entries of $\mathbf{\Lambda}$ to all be set individually, whereas if $\mathbf{\Lambda}$ is defined strictly as a rotation matrix, then 'only' $\frac{n(n-1)}{2}$ angles are required.

However, consider that when a basis is used in applications with the goal of building emulators for the coefficients, as in Chapter 4, the basis is truncated, and only the coefficients on the first q basis vectors are emulated, with q selected so that this basis explains a large percentage of ensemble variability. The form of each vector in the rotated basis, as written in (5.4), shows that to find q new vectors, only the first q columns of $\mathbf{\Lambda}$ are required; the remaining columns do not contribute. Let the first q columns of $\mathbf{\Lambda}$ be denoted by $\mathbf{\Lambda}_q$, and

referred to as the 'truncated rotation matrix'.

As we are only interested in the truncated rotation matrix for emulation, and consider the entries of $\mathbf{\Lambda}_q$ as sets of multiples for the columns of $\mathbf{\Gamma}$, rather than requiring angles, then only nq parameters need to be estimated. As q is generally taken to be much smaller than n, this will usually be fewer than the $\frac{n(n-1)}{2}$ for the full rotation matrix.

However, if we define $\mathbf{\Lambda}_q$ using general multiples, is this a valid method for defining a new basis, and indeed is it a rotation of the original basis? If the full rotation matrix method is used, and the initial basis is orthonormal, as the SVD basis is, then the rotated basis is also orthonormal. If multiples are optimised instead, there is no guarantee of this. In fact, if no restrictions are placed upon the multiples, then it is unlikely that the new vectors will be orthogonal. Orthogonality can however be imposed after selecting general values for $\mathbf{\Lambda}_q$ through the application of Gram-Schmidt (Section 4.6.3). This does not affect the resulting reconstructions of the output, as shown in (4.16).

5.3.3. Combining Gram-Schmidt and re-weighted basis vectors

By definition, a rotation matrix $\mathbf{\Lambda}$ does not affect the inner product of two vectors, so that if the original matrix $\mathbf{B}$ is orthonormal, then $\mathbf{\Gamma} = \mathbf{B\Lambda}$ also is. If instead we apply a general multiplication to the original basis, and then perform Gram-Schmidt to give a new orthonormal basis, we show that this is equivalent to specifying a full rotation matrix, for the case when the original basis is orthogonal.

The requirement that the original $\mathbf{B}$ contains orthonormal columns is not restrictive here, as if this is not true, Gram-Schmidt can be applied to the non-orthogonal basis as a first step, without affecting reconstructions with the basis (4.16).

Result. *Let $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ be an orthonormal basis, and let $\boldsymbol{M}$ be an $n \times n$ matrix containing entries m_{ij} that will be used to multiply $\boldsymbol{B}$ on the right, giving a re-weighting of the columns of $\boldsymbol{B}$. We assume that the columns of $\boldsymbol{M}$ are not linearly dependent, as otherwise Gram-Schmidt cannot be applied. Then $\boldsymbol{R}$ is an upper-triangular matrix given by Gram-Schmidt (with entries r_{ij}), so that $\mathbf{\Gamma}$, given by*

$$\mathbf{\Gamma} = \boldsymbol{BMR}$$

is also an orthonormal basis. Regardless of the choice of entries for $\boldsymbol{M}$ *(given that the columns are not linearly dependent),* $\boldsymbol{MR}$ *gives a rotation of the original matrix* $\boldsymbol{B}$*. That is,*

$$(\boldsymbol{MR})^T = (\boldsymbol{MR})^{-1}$$

i.e. $\boldsymbol{MR}$ *is orthogonal, and hence has determinant equal to* ± 1*, and is a rotation matrix.*

Proof. First, we write a general column of $\boldsymbol{\Gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n)$ in terms of $\mathbf{B}, \mathbf{M}$ and $\mathbf{R}$:

$$\begin{aligned} \boldsymbol{\gamma}_j &= \sum_{p=1}^{n} \sum_{k=1}^{n} \mathbf{b}_k m_{kp} r_{pj} \\ &= \sum_{p=1}^{j} \sum_{k=1}^{n} \mathbf{b}_k m_{kp} r_{pj} \end{aligned} \tag{5.5}$$

using that $\mathbf{R}$ is upper-triangular, so that $r_{pj} = 0$ if $p > j$. From the properties of orthonormal matrices, we have that

$$\boldsymbol{\gamma}_i^T \boldsymbol{\gamma}_j = \mathbf{b}_i^T \mathbf{b}_j = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}$$

Using (5.5), we can write

$$\boldsymbol{\gamma}_i^T \boldsymbol{\gamma}_j = (\sum_{p=1}^{i} \sum_{k=1}^{n} \mathbf{b}_k m_{kp} r_{pi})^T (\sum_{q=1}^{j} \sum_{s=1}^{n} \mathbf{b}_s m_{sq} r_{qj})$$

By the orthonormality of the columns of $\mathbf{B}$, the only non-zero elements of this sum occur when $s = k$, allowing a simplification to

$$\begin{aligned} \boldsymbol{\gamma}_i^T \boldsymbol{\gamma}_j &= (\sum_{p=1}^{i} \sum_{k=1}^{n} \mathbf{b}_k m_{kp} r_{pi})^T (\sum_{q=1}^{j} \mathbf{b}_k m_{kq} r_{qj}) \\ &= \sum_{p=1}^{i} \sum_{k=1}^{n} \sum_{q=1}^{j} m_{kp} r_{pi} m_{kq} r_{qj} \mathbf{b}_k^T \mathbf{b}_k \\ &= \sum_{k=1}^{n} \sum_{p=1}^{i} m_{kp} r_{pi} \sum_{q=1}^{j} m_{kq} r_{qj} \end{aligned} \tag{5.6}$$

By writing $\boldsymbol{\Lambda} = \mathbf{MR}$, and again using that $\mathbf{R}$ is upper triangular, a general element λ_{ij} is

$$\lambda_{ij} = \sum_{a=1}^{j} m_{ia} r_{aj}$$

Hence we can write the columns of $\mathbf{\Gamma}$ in terms of the columns of $\mathbf{\Lambda}$, and (5.6) becomes

$$\begin{aligned}\boldsymbol{\gamma}_i^T\boldsymbol{\gamma}_j &= \sum_{k=1}^{n}(\lambda_{ki})(\lambda_{kj}) \\ &= \boldsymbol{\lambda}_i^T\boldsymbol{\lambda}_j\end{aligned}$$

where $\boldsymbol{\lambda}_i$ is the i^{th} column of $\mathbf{\Lambda}$. Using the orthonormality of $\mathbf{\Gamma}$, we have

$$\boldsymbol{\lambda}_i^T\boldsymbol{\lambda}_j = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}$$

and $\mathbf{\Lambda} = \mathbf{MR}$ is orthogonal, and has determinant equal to ± 1, and is a rotation matrix, regardless of the values of $\mathbf{M}$. (Strictly, if the determinant is -1, then there is also a reflection, and the rotation is called 'improper'. This distinction is not important in our context). □

If the columns of $\mathbf{M}$ are linearly dependent, then the columns of $\mathbf{BM}$ are also linearly dependent, and Gram-Schmidt cannot be applied. This constraint would need to be set when the values of $\mathbf{M}$ are being found.

We now generalise the result that $\mathbf{MR}$ is a rotation matrix to show that we only need to define nq values to find a rotation. Let

$$\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_q, \mathbf{e}_{q+1}, \ldots, \mathbf{e}_n)$$

where $(\mathbf{m}_1, \ldots, \mathbf{m}_q)$ are chosen vectors of multiples for the columns of $\mathbf{B}$, for example chosen to minimise the reconstruction error for the new basis, and $\mathbf{e}_k$ is the vector with k^{th} entry 1, and zeros elsewhere. Then, the previous result shows that by applying Gram-Schmidt to $\mathbf{BM}$, giving the Gram-Schmidt matrix $\mathbf{R}$, gives a new orthogonal basis $\mathbf{\Gamma} = \mathbf{BMR}$, and hence $\mathbf{MR}$ is a rotation matrix.

By the result that the application of Gram-Schmidt does not affect reconstructions, because the new k^{th} basis vector is a linear combination of the first k basis vectors of the non-orthogonal $\mathbf{BM}$ (by equation (4.16)), to find a truncated rotation matrix $\mathbf{\Lambda}_q$, we only need to define nq values for $\mathbf{M}$, and perform Gram-Schmidt. This is an extremely useful result, as it shows that to find a rotation of our original basis $\mathbf{B}$, instead of needing to

define $\frac{n(n-1)}{2}$ angles, we can set $q = 1$ and optimise for n values that minimise $\mathcal{R}_{\mathbf{W}}(\cdot, \mathbf{z})$, subject to constraints on the variability explained, $\mathcal{V}(\cdot, \mathbf{F}_{\boldsymbol{\mu}})$. By combining this with the residual basis, we can iteratively select basis vectors.

5.4. The iterative optimal rotation algorithm

Using the invariance result (Section 5.3.1) and the fact that a rotation matrix can be defined by combining a set of values and Gram-Schmidt, we now present an iterative process for selecting an optimal basis for searching for $\mathbf{z}$.

Given an orthogonal basis $\mathbf{B}$ with dimension $l \times (n-1)$ (as we generally remove the ensemble mean initially), a positive definite $l \times l$ weight matrix $\mathbf{W}$, a vector $\mathbf{v}$ containing values for the minimum proportion of the ensemble variability to be explained by a single basis vector, the total proportion of ensemble variability to be explained by the basis v_{tot}, and a bound T (usually that implied by history matching, $T = \mathcal{X}^2_{0.995,l}$), we find an optimal basis for performing calibration or history matching as follows:

1. If $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) > T$, stop and revisit the specification of $\mathbf{W}$. Else set $k = 1$.

2. Let $\boldsymbol{\Gamma}^*_k = (\boldsymbol{\gamma}^*_1, \ldots, \boldsymbol{\gamma}^*_{k-1}, \mathbf{B}\boldsymbol{\lambda}_k)$ and set

$$\boldsymbol{\lambda}^*_k = argmin_{\boldsymbol{\lambda}_k} \|\mathbf{z} - \boldsymbol{\Gamma}^*_k (\boldsymbol{\Gamma}^{*T}_k \boldsymbol{\Gamma}^*_k)^{-1} \boldsymbol{\Gamma}^{*T}_k \mathbf{z}\|_{\mathbf{W}}$$

   with the constraint that $\mathbf{B}\boldsymbol{\lambda}^*_k$ explains at least v_k of the ensemble variability:

$$\mathcal{V}_k(\boldsymbol{\Gamma}^*_k, \mathbf{F}_{\boldsymbol{\mu}}) \geq v_k$$

3. If $k > 1$, perform Gram-Schmidt so that $\mathbf{B}\boldsymbol{\lambda}^*_k$ is orthogonal to each of the previously selected vectors. Denote this new orthogonal vector as $\boldsymbol{\gamma}^*_k$, and the current basis as $\boldsymbol{\Gamma}^*_k = (\boldsymbol{\gamma}^*_1, \ldots, \boldsymbol{\gamma}^*_k)$. The reconstruction error $\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}^*_k, \mathbf{z})$ is invariant to the application of Gram-Schmidt, as we showed in Section 4.6.3.

4. Calculate the ensemble residual $\mathbf{F}_\epsilon$ using the current basis $\boldsymbol{\Gamma}^*_k$:

$$\mathbf{F}_\epsilon = \mathbf{F}_{\boldsymbol{\mu}} - \boldsymbol{\Gamma}^*_k (\boldsymbol{\Gamma}^{*T}_k \boldsymbol{\Gamma}^*_k)^{-1} \boldsymbol{\Gamma}^{*T}_k \mathbf{F}_{\boldsymbol{\mu}}$$

Calculate the SVD of $\mathbf{F}_\epsilon$ to give the residual basis $\mathbf{B}_\epsilon$

$$\mathbf{F}_\epsilon^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

$$\mathbf{B}_\epsilon = \mathbf{V}_{n-1-k}$$

for $\mathbf{U}$, $\boldsymbol{\Sigma}$ and $\mathbf{V}$ as in (2.15), and for $\mathbf{V}_{n-1-k}$ denoting the first $(n-1-k)$ columns of $\mathbf{V}$. This gives an orthogonal rank $(n-1)$ basis (by Section 4.6.2)

$$\boldsymbol{\Gamma}^* = (\boldsymbol{\Gamma}_k^*, \mathbf{B}_\epsilon)$$

5. Define $q \geq k$ as the minimum value satisfying

$$\mathcal{V}(\boldsymbol{\Gamma}_q^*, \mathbf{F}_{\boldsymbol{\mu}}) \geq v_{tot}$$

where $\boldsymbol{\Gamma}_q^*$ is the first q columns of $\boldsymbol{\Gamma}^*$. If

$$\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}_q^*, \mathbf{z}) < T$$

then stop, and return $\boldsymbol{\Gamma}_q^*$ as the truncated basis. Else, set $k = k+1$, and return to step 2 to select a new vector.

To optimise for $\boldsymbol{\lambda}_k$, we use simulated annealing (Yang Xiang et al., 2013).

5.4.1. Discussion of the algorithm

Prior to applying the algorithm and rotating $\mathbf{B}$, a few choices need to be made. As suggested in the previous chapter, a natural choice for $\mathbf{W}$ is

$$\mathbf{W} = \boldsymbol{\Sigma}_{\mathbf{e}} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$$

giving the direct parallel to history matching and the chi-squared bound T, making this tolerance to error a sensible check. The total amount of variability to be explained, v_{tot}, will generally be set equal to 0.95, although for ensembles exhibiting large variability between members, this may require a large number of basis vectors, making emulation more time-consuming. In a problem of this nature, this value could be decreased.

Setting the vector of proportions, $\mathbf{v}$, is again problem dependent, but from experience, setting at least the first few values as 0.1 produces satisfactory results. This helps to ensure emulation is possible for the vectors that are likely to be minimising the reconstruction error the most, as these are selected first.

The initial check $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) > T$ comes from the fact that $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) = \mathcal{R}_{\mathbf{W}}(\mathbf{B}\boldsymbol{\Lambda}, \mathbf{z})$ for any rotation matrix $\boldsymbol{\Lambda}$, as proved in Section 5.3.1. It is therefore known a priori whether a rotation of this form will exist such that

$$\mathcal{R}_{\mathbf{W}}((\mathbf{B}\boldsymbol{\Lambda})_q, \mathbf{z}) < T$$

under the current specification of the weight $\mathbf{W}$. If this does not hold, then $\mathbf{z}$ will be ruled out, regardless of the rotation applied.

Terminating the algorithm at this step has a few possible implications. Firstly, the specification of the variances $\boldsymbol{\Sigma}_{\mathbf{e}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$, or any other matrix used as the weight, may need revisiting. Given the current ensemble, and these variances, the observations cannot be found. This may suggest that more model runs are required: given those that are available, nothing close enough to the observations has been found, and hence the SVD basis does not contain important patterns.

This suggests a way that the discrepancy could be defined for a problem. Increasing the discrepancy variance will reduce the reconstruction error for a given basis, and the initial criteria may then be satisfied. This need not be a permanent change in the specification of $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$, but could be used as a means of obtaining a basis that won't rule out $\mathbf{z}$, so that a search can be performed in this direction of the output space. Ruling out all of space may not be useful, especially if the ensemble is small and more runs can be obtained. The idea of increasing the discrepancy and hence the weight $\mathbf{W}$ so that the reconstruction of the observations does fall below T will be developed further in Section 6.3.2.

An iterative approach has been adopted for selecting the rotation, due to the fact that a full rotation is implied by combining a vector of $(n-1)$ values with Gram-Schmidt. This has reduced the optimisation from one of selecting $(n-1)^2$ values or $\frac{(n-1)(n-2)}{2}$ angles, to one where $(n-1)$ values are optimised for at each iteration.

At each iteration, a full (rank $(n-1)$) basis is calculated at step 4 given the k chosen

basis vectors, by combining these with the residual basis from Section 4.6.1. This gives a basis that explains all of the ensemble variability, with the residual basis orthogonal to the basis vectors that have previously been selected (see Section 4.6.2). Combining the optimised basis vectors with the residual basis essentially completes the rotation matrix so that it has dimension $(n-1) \times (n-1)$, via an extremely efficient calculation, in a manner that orders the remaining basis vectors by variance explained. This is a rotation because the new basis explains 100% of the variability in $\mathbf{F}_{\boldsymbol{\mu}}$ with $(n-1)$ vectors (by the construction of the residual basis) as in $\mathbf{B}$, so that the new basis represents the same $(n-1)$-dimensional subspace as $\mathbf{B}$.

This full basis is then truncated after q basis vectors in step 5, so that at least v_{tot} of ensemble variability has been explained. Using this truncated basis, the reconstruction error is calculated, and if this is now less than T, then the algorithm terminates, as a suitable basis has been found. In practice, it has often been found that one iteration is sufficient, with the first basis vector rotated in such a way that combining it with the residual basis is far superior to the original basis $\mathbf{B}$, so that only $(n-1)$ values need to be optimised overall, a much more straight-forward and efficient optimisation problem than that requiring $(n-1)^2$ or $\frac{(n-1)(n-2)}{2}$ values to be optimised.

If the current truncated basis does not satisfy $\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}_q^*, \mathbf{z}) < T$, a new basis vector $\mathbf{B}\boldsymbol{\lambda}_k$ is selected, given any basis vectors from previous steps of the rotation, so that the combination of the previously selected vectors and this new one minimises the reconstruction error, subject to satisfying the variability constraint. Utilising an iterative method simplifies the application of the variability constraint, with only $\mathcal{V}_k(\boldsymbol{\Gamma}_k, \mathbf{F}_{\boldsymbol{\mu}}) \geq v_k$ needing to be evaluated and satisfied for a fixed k, rather than for all $k = 1, \ldots, q$ simultaneously.

If there are a large number of ensemble members, and hence basis vectors, there are more values, $(n-1)$, to be optimised at each iteration. If the optimisation becomes too computationally intensive, then rather than considering the full SVD basis, the SVD basis vectors that have the largest individual effect on reducing the reconstruction error could be solely considered. This would reduce the number of values to be optimised, and should not affect the ability to identify an optimal basis too greatly, as the reconstruction error would still be reduced, and then the residual basis ensures that enough overall variability is explained by the final truncated basis.

5.4.2. Increasing efficiency

Increased computational efficiency can be achieved via a slight modification of the algorithm. This has the effect of removing the Gram-Schmidt step completely, and reducing the number of values or parameters to be optimised at each iteration from $(n-1)$ to $(n-k)$. Instead of directly rotating the original basis $\mathbf{B}$ at iteration $k > 1$, multiples of the vectors of the residual basis $\mathbf{B}_\epsilon$ from iteration $k-1$ can be considered.

Due to the fact that the residual basis, combined with the previously selected patterns, gives a rank $(n-1)$ basis, and hence defines the same subspace as $\mathbf{B}$, the residual basis can be written as

$$\mathbf{B}_\epsilon = \mathbf{B}\mathbf{\Lambda}_\epsilon$$

for an $(n-1) \times (n-k)$ matrix $\mathbf{\Lambda}_\epsilon$, which can be written as

$$\mathbf{\Lambda}_\epsilon = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{B}_\epsilon = \mathbf{B}^T\mathbf{B}_\epsilon$$

Exploiting this formulation of the residual basis as dependent on the original basis, the optimisation at the following iteration can be performed for a vector $\boldsymbol{\lambda}_k$ with $(n-k)$ entries:

$$\mathbf{B}_\epsilon\boldsymbol{\lambda}_k = \mathbf{B}\mathbf{\Lambda}_\epsilon\boldsymbol{\lambda}_k$$

Defining $\tilde{\boldsymbol{\lambda}}_k := \mathbf{\Lambda}_\epsilon\boldsymbol{\lambda}_k$ then gives a vector of length $(n-1)$ that gives a linear combination of the original basis $\mathbf{B}$. Applying a rotation to the residual basis still gives a rotation of $\mathbf{B}$, with fewer values to be optimised for, simplifying the optimisation problem.

The second simplification that the change to the residual basis allows is the removal of the Gram-Schmidt step. By definition, the vector selected by rotating the residual basis is orthogonal to the previously selected vectors. Recall from Section 4.6.2 that the residual basis is orthogonal to each of the basis vectors used to calculate the ensemble basis, i.e.

$$\mathbf{\Gamma}_{k-1}^{*T}\mathbf{B}_\epsilon = \mathbf{0}$$

Hence the first $k-1$ vectors are orthogonal to the rotation of the residual basis at step k:

$$\mathbf{\Gamma}_{k-1}^{*T}(\mathbf{B}_\epsilon\boldsymbol{\lambda}_k) = (\mathbf{\Gamma}_{k-1}^{*T}\mathbf{B}_\epsilon)\boldsymbol{\lambda}_k = \mathbf{0}$$

The residual basis automatically constrains any chosen vectors to be orthogonal to those basis vectors previously selected (shown in Section 4.6.2). The new basis vector still needs to be normalised so that it has length 1, but the Gram-Schmidt step is now redundant. These additions lead to the following updated algorithm:

1. If $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) > T$, stop and revisit the specification of $\mathbf{W}$. Else set $k = 1$.

2. Let $\mathbf{\Gamma}_k^* = (\boldsymbol{\gamma}_1^*, \ldots, \boldsymbol{\gamma}_{k-1}^*, \mathbf{B}\boldsymbol{\lambda}_k)$ and set

$$\boldsymbol{\lambda}_k^* = argmin_{\boldsymbol{\lambda}_k} \|\mathbf{z} - \mathbf{\Gamma}_k^*(\mathbf{\Gamma}_k^{*T}\mathbf{\Gamma}_k^*)^{-1}\mathbf{\Gamma}_k^{*T}\mathbf{z}\|_{\mathbf{W}}$$

   with the constraint that $\mathbf{B}\boldsymbol{\lambda}_k^*$ explains at least v_k of the ensemble variability:

$$\mathcal{V}_k(\mathbf{\Gamma}_k^*, \mathbf{F}_{\boldsymbol{\mu}}) \geq v_k$$

   Define the new normalised vector as

$$\boldsymbol{\gamma}_k^* = \frac{\mathbf{B}\boldsymbol{\lambda}_k^*}{\|\mathbf{B}\boldsymbol{\lambda}_k^*\|}$$

   and set $\mathbf{\Gamma}_k^* = (\boldsymbol{\gamma}_1^*, \ldots, \boldsymbol{\gamma}_{k-1}^*, \boldsymbol{\gamma}_k^*)$.

3. Calculate the ensemble residual $\mathbf{F}_\epsilon$ using the current basis $\mathbf{\Gamma}_k^*$:

$$\mathbf{F}_\epsilon = \mathbf{F}_{\boldsymbol{\mu}} - \mathbf{\Gamma}_k^*(\mathbf{\Gamma}_k^{*T}\mathbf{\Gamma}_k^*)^{-1}\mathbf{\Gamma}_k^{*T}\mathbf{F}_{\boldsymbol{\mu}}$$

   Calculate the SVD of $\mathbf{F}_\epsilon$ to give the residual basis $\mathbf{B}_\epsilon$

$$\mathbf{F}_\epsilon^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$
$$\mathbf{B}_\epsilon = \mathbf{V}_{n-1-k}$$

   for $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ as in (2.15), and for $\mathbf{V}_{n-1-k}$ denoting the first $(n-1-k)$ columns of $\mathbf{V}$. This gives an orthogonal rank $(n-1)$ basis (by Section 4.6.2)

$$\mathbf{\Gamma}^* = (\mathbf{\Gamma}_k^*, \mathbf{B}_\epsilon)$$

4. Define $q \geq k$ as the minimum value satisfying

$$\mathcal{V}(\mathbf{\Gamma}_q^*, \mathbf{F}_{\boldsymbol{\mu}}) \geq v_{tot}$$

where $\mathbf{\Gamma}_q^*$ is the first q columns of $\mathbf{\Gamma}^*$. If

$$\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z}) < T$$

then stop, and return $\mathbf{\Gamma}_q^*$ as the truncated basis. Else, set $k = k + 1$ and $\mathbf{B} = \mathbf{B}_\epsilon$, and return to step 2 to select a new vector.

5.5. Basis rotation for the toy function

A first wave of calibration and history matching for the toy function was carried out in Section 4.4, using the SVD basis. The conclusion of this application was that the wrong region of parameter space was highlighted (by calibration), with samples from the posterior distribution for $\mathbf{x}^*$ not resembling the observed field $\mathbf{z}$. History matching ruled out all of parameter space. History matching was improved by selecting a pattern in Section 4.7. Instead of selecting a pattern, we now find an optimal rotation by applying the method from Section 5.4.2 to this problem.

As discussed above, there are a number of choices to be made when applying this method. Firstly, the basis that will be rotated will be the SVD basis, $\mathbf{\Gamma}$, of the toy function ensemble. The full rank 60 basis will be considered in the optimisation stage, as optimising for 60 parameters at each step is not prohibitively time-consuming. The reconstruction error will be minimised with respect to the weight matrix $\mathbf{W} = \mathbf{\Sigma_e} + \mathbf{\Sigma}_{\boldsymbol{\eta}}$, so that the direct comparison with history matching is suitable. The minimum proportion of variability to be explained by the initial basis vector is set as $v_1 = 0.4$, with all other entries of $\mathbf{v}$ set equal to 0.1.

Given these choices, we find a new basis $\mathbf{\Gamma}^* = \mathbf{\Gamma\Lambda}$ by applying the algorithm, the first four vectors of which are shown in Figure 5.1. To find a basis that has $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z}) < T$, only one iteration of the algorithm was required: the basis vector given by the rotation of $\mathbf{\Gamma}$ at the first iteration is by itself sufficient to give a reconstruction error below the threshold

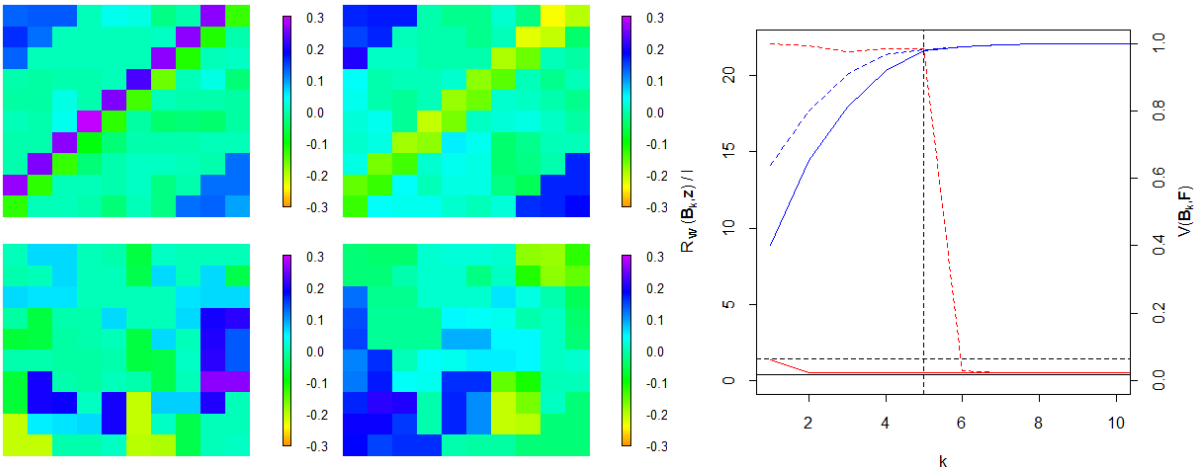

Figure 5.1. The first four basis vectors for the basis selected by rotating the SVD basis, alongside the VarMSE plot for this basis (solid lines) and the SVD basis (dotted lines), with $\mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$. The dotted horizontal line represents the history matching bound, and the solid horizontal line represents the reconstruction error given when the full SVD basis is used. The dotted vertical line shows the truncation for the rotated basis.

T. The new full basis was then given by combining this first vector with its residual basis.

The first five basis vectors of this new basis are required in order to explain more than 95% of the ensemble variability. It is not surprising that an extra basis vector is needed compared to the SVD basis: the proportion explained by the first basis vector, $\mathcal{V}_1(\boldsymbol{\Gamma}^*, \mathbf{F}_{\boldsymbol{\mu}})$, has been reduced by forcing it to contain more patterns similar to $\mathbf{z}$, which does not explain much of $\mathbf{F}_{\boldsymbol{\mu}}$. This new first basis vector explains 40.01% of the variability in $\mathbf{F}_{\boldsymbol{\mu}}$. In repeated applications of this algorithm, we have found that this percentage will often be a small amount above the set threshold: if the projection onto this basis vector explains more of the variability in $\mathbf{F}_{\boldsymbol{\mu}}$, then it is likely to be worse at representing $\mathbf{z}$.

The VarMSE plot in Figure 5.1 shows that only the first basis vector is required for the reconstruction error, $\mathcal{R}_\mathbf{W}(\boldsymbol{\Gamma}^*, \mathbf{z})$, to be below the history matching bound. Prior to rotation, this was not the case for the first basis vector, or even the truncated SVD basis (shown by the dotted red line). Adding the second vector improves this further, and takes the error towards its theoretical minimum for the SVD basis, $\mathcal{R}_\mathbf{W}(\boldsymbol{\Gamma}, \mathbf{z})$. This, combined with the fact that projection onto these first basis vectors explains a reasonable amount of the variability in the ensemble, suggests that this basis is suitable for calibration and history matching, and it should be possible to build informative emulators for the ensemble coefficients on these vectors.

We fit Gaussian process emulators for these coefficients as described in Section 4.3. Cross-

validation plots are shown in Appendix B.3, suggesting that our emulators are suitable for predicting the coefficients at input parameters $\mathbf{x}$.

5.5.1. Calibration with the rotated basis

We use the emulators built for the coefficients of the optimally rotated SVD basis to perform Bayesian calibration for the toy model. We use the same method as in Section 4.4, with the emulator expectation and variance mapped back to the original l-dimensional space, following the method of Wilkinson (2010). Sampling from the posterior distribution is performed using the same Metropolis-Hastings algorithm as described in Section 4.4.3, with the initial value for the chain again set at the true input setting, $\mathbf{x}^*$.

Figure 5.2 shows the posterior distributions for each of the input parameters for the rotated basis with solid lines, with the posteriors given by the SVD basis represented by dotted lines. As in the previous calibration, we apply a thinning, with every 100^{th} value taken, although the results are the same with a different choice of thinning. Traceplots for the MCMC chains are shown in Appendix C.

The posterior for x_1 shows a clear bias away from the true parameter value, given by the red line. However, this parameter is not particularly important, as it has no impact on the main diagonal or the off-diagonal pattern. Therefore, while having differences between the sampled values and the true value for this parameter is not ideal, because it does not affect the parts of the output that are of interest, it may have no further impact. When calibrating using the SVD basis, the posterior for x_1 contained a peak closer to the true value.

For x_2, the parameter that is the main driver of the strength of the off-diagonal, the posterior is close to the true value, with the majority of the density slightly above 0. This is similar to the result given by the SVD basis, although the rotated basis has done better: the mean value in this posterior is 0.039, compared to 0.054 for the SVD posterior. This will have the effect of reducing the strength of the off-diagonal, as required.

An accurate value for x_3 has not been found here, although again this is an improvement over the SVD calibration. This is another parameter that affects the off-diagonal strength, so it is important that its value is correct. All of the posterior density lies away from the

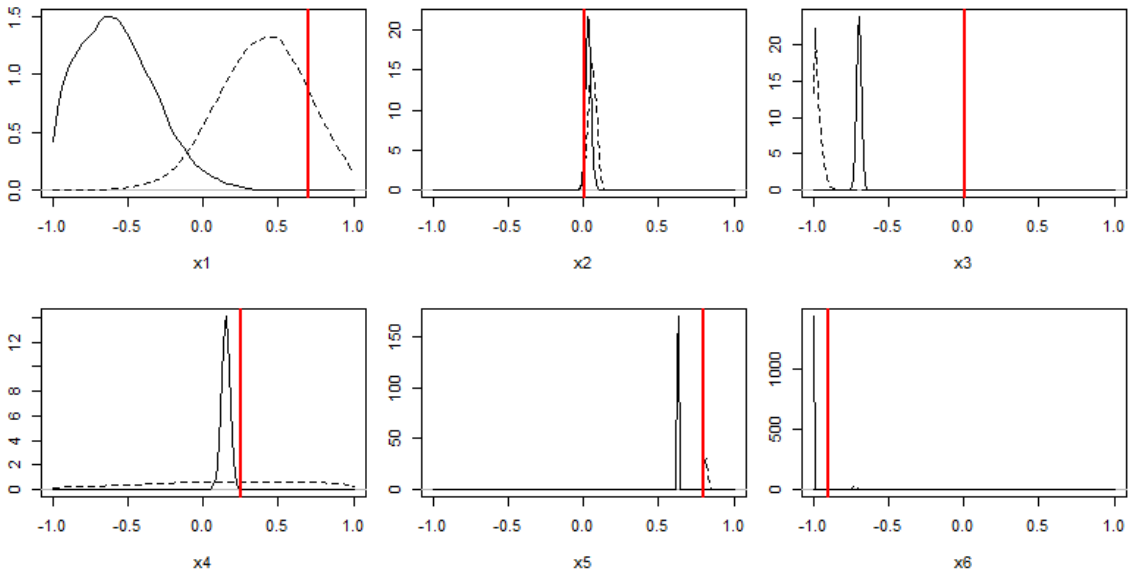

Figure 5.2. The posterior distributions for each of the parameters, when calibration is performed using the first five vectors of the rotated basis (solid lines), with the dotted lines showing the posteriors from when the SVD basis was used, as in Figure 4.8. The red vertical lines indicate the true value of $\mathbf{x}^*$.

true value, with the mean of this distribution at $x_3 = -0.699$. For the SVD posterior, this mean was equal to -0.909, so that while the posterior for the rotated basis is clearly not correct, it is at least suggesting a more accurate value than the SVD calibration was able to.

The key parameter for controlling the strength of the main diagonal is x_4, and so identifying the correct setting of this parameter is critically important if fields resembling the observations are to be found. The SVD calibration was not able to constrain this parameter particularly well, with the posterior density spread across the whole range of x_4. Although the peak of this density contained x_4^*, this maximum was uniformly spread from around 0 to 0.5. The calibration with the rotated basis has much more strongly restricted this parameter. The mean of this posterior is at 0.147, although there is a small amount of density at the true value of 0.25.

Although the SVD calibration assigned more posterior density to x_4^*, this may not be a better result, as it also assigns density to any value of this parameter. The rotated basis calibration misses the true value, but restricts the posterior to a value that is at least close to the true value. Which of these is preferable will be studied by sampling from the posterior distributions later.

The final two parameters are linked together in the function definition by the ratio

$$r = \frac{x_5}{1.3 + x_6}$$

Therefore, parameter values that are not strictly equal to x_5^* and x_6^* can lead to output the same as the observations (up to the observation error). Using these true parameter values, this ratio is equal to 2. Rather than comparing the posteriors for these parameters to x_5^* and x_6^*, considering the above ratio will be more informative.

The posteriors for each of these parameters when the rotated basis is used both show narrow peaks, with $x_5 = 0.627$ and $x_6 = -0.995$, both of which are away from $\mathbf{x}^*$. Calculating the above ratio gives 2.055. The SVD posteriors for these parameters also exhibited narrow peaks, with $x_5 = 0.812$ close to being correct, and $x_6 = -0.731$. Combining these values into the above ratio gives 1.427. Figure 5.3 shows the posterior distribution for this ratio, with the rotated basis assigning density closer to the truth than the SVD basis did previously (as shown by the dotted line). Although the SVD posterior perhaps appears better at first, with the density for x_5 in the correct place, when combined with x_6 it is clear that the rotated version of calibration has outperformed SVD here.

From the individual posterior distributions, it appears that the rotated basis has given more accurate results. This can be further illustrated by sampling from these distributions, and running the toy function at these sampled values, as in Figure 4.9 for the SVD basis. These sampled fields all contained a strong signal on the off-diagonal, with the patterns from $\mathbf{z}$ not displayed. Samples from the posterior distribution found with the rotated basis are shown in Figure 5.4.

These runs are all closer to the observations than those sampled from the SVD posterior. The key patterns from the observations, namely large positive values on the main diagonal, and negative values in the corners, are found in each of these sampled fields, a significant improvement over the SVD case. In the majority of these runs, the values on the main diagonal are larger than those on the off-diagonal, for at least some sections of the diagonal. The main problem with these samples is that the values on the off-diagonal are too strong; the main diagonal should be much more distinctive.

Although these samples do not perfectly reproduce fields that are similar to $\mathbf{z}$, this is a

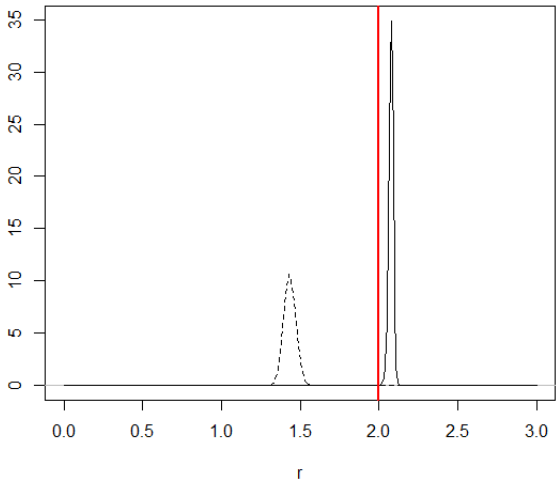

Figure 5.3. The posterior distribution for r, for the rotated basis (solid line) and the SVD basis (dotted line).

much better result than for the SVD basis. In that case, there was no suggestion that it was possible to find runs with a stronger main diagonal, whereas here it is clear that the values on the main diagonal can be affected by the input parameters. This result suggests that an extra wave of calibration should be performed: with little signal in the direction of the observations, it has been possible to identify runs containing some signal in this direction, by incorporating this faint signal into the basis via the rotation. The rotated basis satisfied the requirement that enough signal was contained in the first few basis vectors so that the coefficients were emulatable. If a new sample were to be taken, and runs similar to those in Figure 5.4 were to be used in the ensemble, then there will be more signal in the direction of $\mathbf{z}$ in the ensemble, and so a more suitable basis for searching in this direction may be found. It should be possible to build more accurate emulators for basis vectors in the direction of $\mathbf{z}$, and may be possible to find $\mathbf{x}^*$, and other runs where there are larger values on the main diagonal, and lower values on the off-diagonal.

In order to perform a second wave of calibration, we first carry out history matching for wave 1 using the current basis, to define an NROY space from which a new ensemble will be sampled.

5.5.2. History matching

When the SVD basis was used for history matching in Section 4.4.4, the entire parameter space $\mathcal{X}$ was ruled out. This result was unsurprising, given that the samples from the

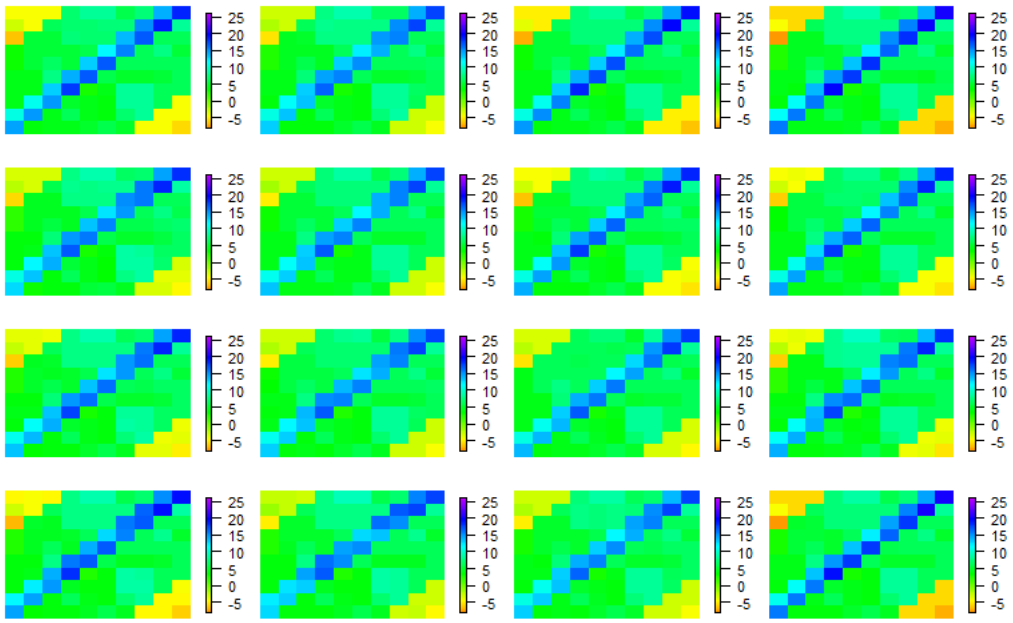

Figure 5.4. $f(\mathbf{x})$ at 16 samples of $\mathbf{x}$ from the calibration posterior distribution, using the rotated basis for projection and emulation.

calibration posterior were all dissimilar from $\mathbf{z}$. Given the improved calibration results with the rotated basis, history matching may result in a non-empty NROY space here. It is clearly important that this NROY space contains $\mathbf{x}^*$ and the other parameter settings from the true NROY space.

The same method for history matching the field is used as previously (Section 4.4.4): the emulator means and variances for the coefficients are mapped back to the original field, and the implausibilities are calculated over the field, with the same chi-squared value, T, used to rule points out. Using the rotated basis and its emulators to calculate the implausibilities, a non-empty NROY space is now found: 31.49% of parameter space is classified as not ruled out. This does not necessarily mean that this basis has improved upon using the SVD basis for history matching, as it is possible that parameter settings that lead to runs consistent with $\mathbf{z}$ have been ruled out incorrectly. However, as the toy function is quick to run at any parameter setting, this can be assessed by calculating the implausibility for parameter choices in the true NROY space. All of these choices are found to lie in NROY space, and hence the rotated basis has improved history matching.

The composition of this NROY space is illustrated in Figure 5.5, with proportions of runs not ruled out shown for each pair of parameters, averaged over the other four parameters (with the axes reversed for the true NROY space plots as before). This shows that the parameter that has been most strongly been constrained is x_2, with extremely high and

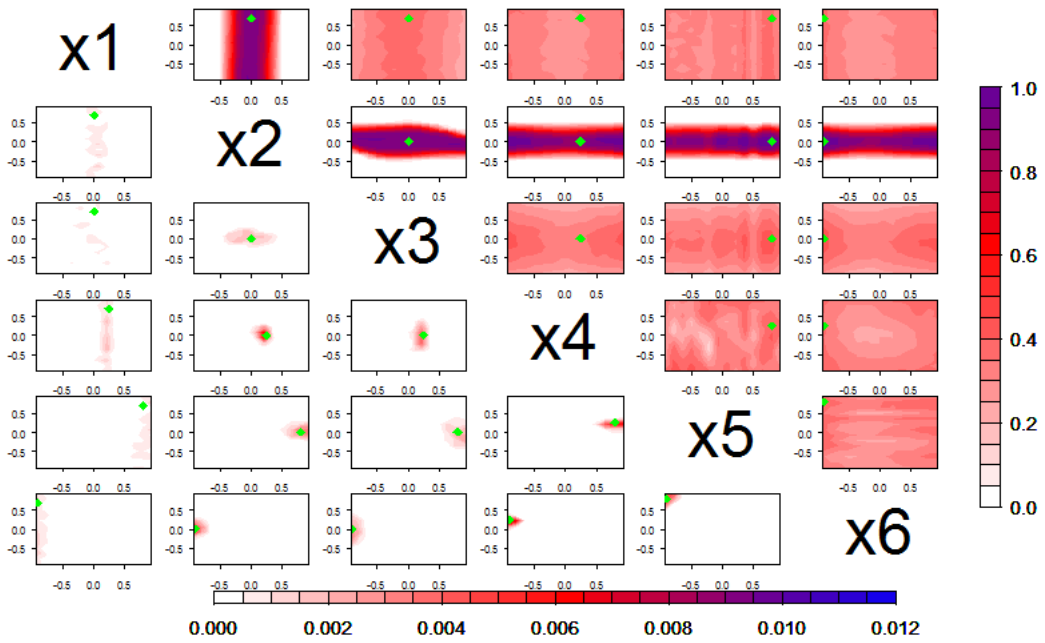

Figure 5.5. Density plot for the wave 1 NROY space defined using the rotated basis (upper right), and the true NROY space (lower left), for each pair of parameters. The remaining parameters are averaged over for each plot, and the proportion in NROY space for each pair is plotted. The axes are reversed for the lower left plots to allow a comparison with the top half. The green point corresponds to $\mathbf{x}^*$.

low values both completely removed. This is a good result: the larger the absolute value of this parameter, the higher the values on the off-diagonal. Ruling out the extreme values of x_2 immediately restricts the magnitude of this pattern, and suggests that if an ensemble were to be sampled from this NROY space, then it is unlikely that runs containing high values on the off-diagonal, and little pattern elsewhere (as in the majority of the wave 1 ensemble) would be selected.

From these pairwise plots, there are no other parameter choices that have been completely ruled out. However, the presence of red colours rather than blue shows that space has been cut down, although not in a way that is immediately identifiable in two dimensions.

5.6. Combining history matching and calibration

In the previous section, a substantial improvement over calibrating or history matching using the SVD basis has been achieved using our rotation method to select an optimal basis. However, the rotation basis has not given perfect results yet: there is still a lot of parameter space leading to biased fields that has not been ruled out, and the true $\mathbf{x}^*$ has not yet been identified.

This is not surprising. In the initial ensemble, there was little signal in the direction of $\mathbf{z}$, and the rotated basis had to account for this by selecting patterns that could be emulated, while explaining some variability in the direction of $\mathbf{z}$, but perhaps not all of it. However, by defining an NROY space using the basis containing some signal from $\mathbf{z}$, if a new ensemble were to be sampled from this space, then it is likely that there will be more signal in the desired direction than in the original ensemble. This assumes that the emulators for this direction were accurate, but the emulatability constraint should have helped to ensure this.

Therefore, a new ensemble is selected, and further waves of history matching and calibration are performed.

5.6.1. Ensemble design

In this example, the new ensemble is designed to have 60 members, for consistency with the ensemble for the first wave. The new design points are selected from the NROY space defined using the wave 1 rotated basis and emulators.

We present the following method for multi-wave design. Due to the efficiency with which the implausibility can be calculated for a parameter setting $\mathbf{x}$, we first take a large sample $\mathbf{S}$ (10,000 members here) of points in NROY space. From these points, we select the 60 ensemble members as follows:

1. Let m be the minimum implausibility given by points in $\mathbf{S}$:

$$m = \min_{\mathbf{x} \in \mathbf{S}} \mathcal{I}(\mathbf{x})$$

2. Divide $\mathbf{S}$ into i groups of width $\omega = \frac{T-m}{i}$ based on the value of $\mathcal{I}(\mathbf{x})$, so that group k is given by

$$s_k = \{\mathbf{x} \in \mathbf{S} | m + (k-1)\omega \leq \mathcal{I}(\mathbf{x}) < m + k\omega\}$$

   where T is the bound used to define $\mathcal{X}_{NROY}$.

3. Sample j points from each of the i groups s_k, so that $ij = n$ is the ensemble size.

As the size of $\mathbf{S}$ tends to infinity, $\mathbf{S} = \mathcal{X}_{NROY}$, so that m is the minimum implausibility for any $\mathbf{x}$.

Designing the ensemble in this way is similar to taking a Latin hypercube in implausibility space, rather than across the input space as is commonly done. For example, we could divide the implausibility space into $i = n$ groups, and, from each of these n intervals, sample $j = 1$ value of $\mathbf{x}$, so that each implausibility interval is represented, as for Latin hypercubes.

Sampling in this manner, rather than searching for a design that is space-filling for the current NROY space, ensures that some runs with the lowest implausibilities are included. If a purely space-filling design were used, it would be possible that all of these runs would have implausibilities close to the bound used to define NROY space. There are likely to be fewer runs in the interval containing the lowest implausibilities, so that these runs may not be selected if space-filling was the sole requirement. Including some runs with lower implausibilities is sensible as these are the runs that are currently deemed to be most likely to give output consistent with the observations. This will hopefully lead to an ensemble that is more informative for $\mathbf{x}^*$.

By sampling using the implausibilities, there is no guarantee that a representative sample, in terms of the input parameters, of the current NROY space is selected: it is possible that the majority of points would be clustered in one region of the space. To ensure this is not the case, having selected n points by sampling from the implausibility intervals, we calculate the minimum distance between each point and another point in the design. This is compared to the spread of these distances if n runs are instead chosen at random from NROY space. If the minimum distances from the implausibility design are similar or larger than randomly-selected samples, we deem this design to be an acceptable representation of NROY space.

For the wave 2 design, 60 design points are selected by dividing the implausibility into $i = 10$ equal intervals, and sampling $j = 6$ points at random from the points that have implausibility in each of these intervals. The minimum implausibility in the 10,000 samples from NROY space is 53.9, and the maximum is 140.2. The resulting ensemble contains a spread of these implausibilities, whilst also spacing points out in the wave 1 NROY space.

The wave 2 ensemble is defined as:

$$\mathbf{F}^{(2)} = (f(\mathbf{x}_1^{(2)}), \ldots, f(\mathbf{x}_{60}^{(2)})) \quad \text{such that } \mathcal{I}^{(1)}(\mathbf{x}_i^{(2)}) < \chi^2_{0.995,100} \text{ for } i = 1, \ldots, 60$$

where $\mathbf{x}_i^{(j)}$ denotes the i^{th} design point at wave j.

5.6.2. Wave 2

Using this new ensemble, we can perform basis rotation, emulation, history matching, and calibration for wave 2. Prior to this, we check whether any points from the wave 1 ensemble lie in NROY space. Given that these are known runs of the model, these should also be included in the wave 2 analysis, to improve emulation. These wave 1 points can be used as validation points to assess the wave 2 emulators, before being added to the emulator if validation checks are satisfied. The emulators built at this wave are only defined to be valid in NROY space, hence the ruled out runs from the previous wave are not included.

Let the validation set at wave k be given by

$$\mathbf{F}_v^{(k)} = \{f(\mathbf{x}) \in \mathbf{F} = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(k-1)}) | \mathcal{I}^{(i)}(\mathbf{x}) < T_i,\ i = 1, \ldots, k-1\}$$

i.e. runs that have been observed in any previous ensemble, and have not been ruled out at any previous waves, where T_i is the threshold used to define NROY space at wave i. For the current wave 2 example, none of the runs from the wave 1 ensemble are in NROY space, hence there are 60 points used in the modelling at this wave. We divide these 60 runs into a training and validation set randomly for fitting emulators.

We now follow the same methodology as at wave 1. We calculate the SVD basis for the centred ensemble of model runs, and again apply the optimal rotation algorithm to ensure that the observations would not be ruled out when history matching is performed. The rotated basis found at this wave is shown in Figure 5.6, alongside a VarMSE plot displaying the reconstruction error for the SVD and rotated bases. The first five vectors of the rotated basis explain greater than 95% of the variability in $\mathbf{F}_{\boldsymbol{\mu}}^{(2)}$, so that the rotated basis is truncated after these vectors. Comparing this basis to the wave 1 rotated basis in Figure 5.1, we observe that the patterns included in the basis are extremely similar:

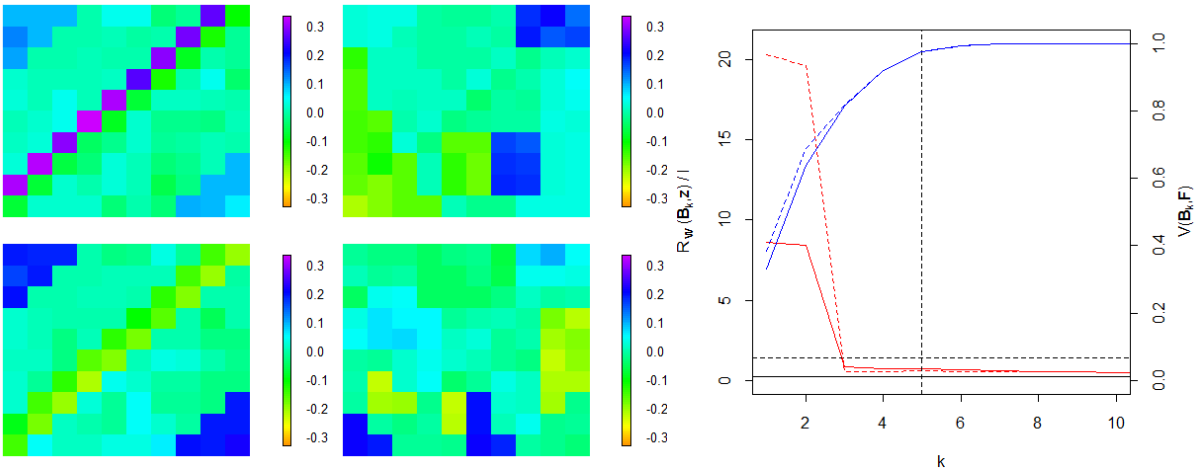

Figure 5.6. The first four basis vectors for the wave 2 rotated basis. The VarMSE plot, with $\mathbf{W} = \mathbf{\Sigma_e} + \mathbf{\Sigma_\eta}$, shows both the SVD basis (dotted red and blue lines) and the rotated basis (solid lines).

there are some differences between the two selected bases, and the vectors are ordered differently, but the general patterns are the same.

Given that there are a limited number of 'true' directions built into the toy function, finding a similar basis is expected. Furthermore, the rotation aims to optimise the basis for representing $\mathbf{z}$, and hence the same patterns are likely to be important for this, and included in the rotated basis, should they be present in the ensemble.

The VarMSE plot in Figure 5.6 shows that with the SVD basis, the reconstruction error is below the dotted line representing the history matching bound when the first five basis vectors are included. Unlike at wave 1, the basis vector that has the greatest impact on reducing the error is included in the truncated SVD basis, as the first five vectors are required in order to explain more than 95% of the ensemble variability. Therefore, it could be questioned why a rotation has been applied in this case. The argument for performing this is that with the SVD basis, the majority of the improvement in reconstructing $\mathbf{z}$ is given by the third SVD vector. This vector explains around 12% of the variability in the wave 2 ensemble, however building an emulator for the coefficients on this basis vector leads to large uncertainties for all $\mathbf{x}$. As this is the only vector in the SVD basis that is important for representing $\mathbf{z}$, having informative emulators for this basis vector is critical, as otherwise it may not be possible to rule out any additional space, or perform an accurate calibration.

We find a rotation so that the important signal from the third SVD basis vector can be

combined into other basis vectors. The third basis vector of the rotated basis still causes a large reduction in the reconstruction error (solid line in Figure 5.6), but it is not as large as for the SVD basis, and now some of this signal is included in the first basis vector. Each of these explains more variability (the first basis vector explains 33%, the third 18%), so that emulation is more informative, and hence this basis should be more suited to ruling out runs not consistent with $\mathbf{z}$. We fit Gaussian process emulators to the coefficients for each of the first five basis vectors, with validation checks performed (Appendix B.3).

Given the rotated basis and the emulators for the coefficients, we perform a second wave of history matching. Starting from the wave 1 NROY space consisting of 31.49% of the full parameter space $\mathcal{X}$, 90.28% of this NROY space is ruled out here. Hence, after two waves of history matching, NROY space has been reduced to 3.06% of $\mathcal{X}$. A density plot of this NROY space is shown in Figure 5.7.

Previously, only x_2 had clearly been constrained. Now, large pairwise sections of parameter space have been completely ruled out. The interaction between x_2 and x_3, as observed in the true NROY space, is clear in the wave 2 NROY space. The main difference between the current NROY space and the true NROY space is in the values of x_4 allowed. In the wave 2 NROY space, a large range of x_4 values are still possible, whereas a narrow range around $x_4 = 0.25$ is all that in fact leads to output consistent with $\mathbf{z}$. As this parameter is one of the main drivers of the main diagonal, the fact that in this NROY space a large range of values are possible suggests that if calibration is performed here, runs sampled from this posterior may still not be completely consistent with $\mathbf{z}$.

The composition of this NROY space suggests that further waves are necessary. Although space has been reduced significantly, x_4 has not yet been restricted enough. This may be because this parameter has not had much effect on the first two ensembles. Now that parameter space has been reduced, and the other parameters strongly constrained, it may be possible to more accurately identify the effect of this parameter (x_4 may now become a more active variable).

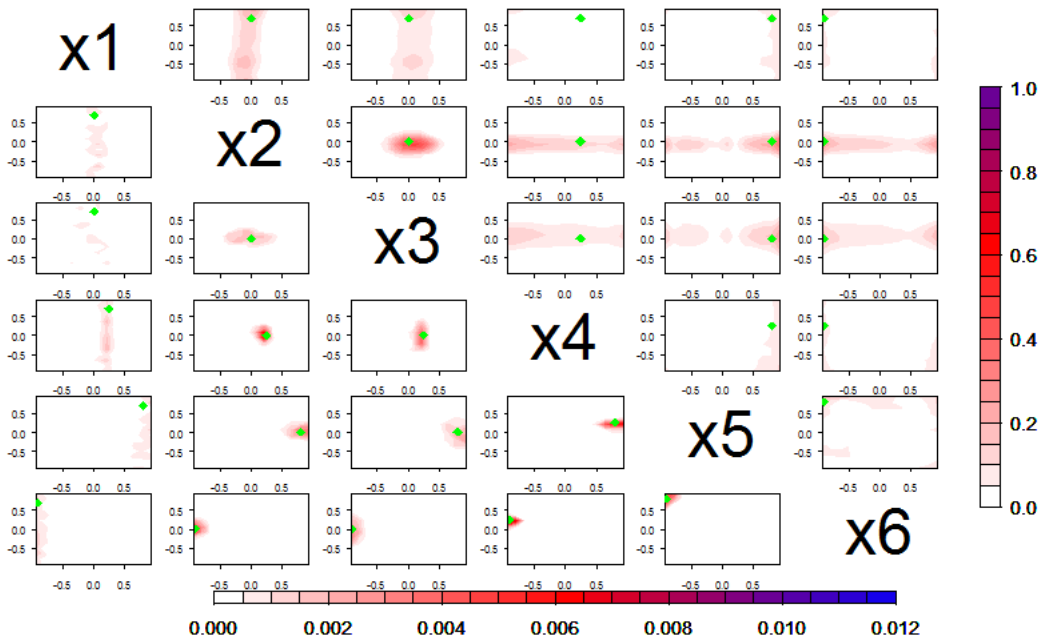

Figure 5.7. Density plot for the wave 2 NROY space defined using the rotated basis (upper right), and the true NROY space (lower left), for each pair of parameters.

5.6.3. Wave 2 calibration

We also perform Bayesian calibration using the wave 2 basis and emulators, to check whether the posterior distribution is now more accurate than after the first wave. We apply the same method as at wave 1, with the prior for $\mathbf{x}^*$ changed to reflect the constraint that zero probability should be assigned to parameter settings outside of the wave 1 NROY space:

$$\pi(\mathbf{x}^*) \propto \begin{cases} 1 & \text{if} \quad \mathbf{x}^* \in \mathcal{X}_{NROY} \\ 0 & \text{otherwise} \end{cases}$$

The traceplots from the MCMC are shown in Figure C.3.

Figure 5.8 shows the posterior densities for each parameter, compared to the true parameter setting $\mathbf{x}^*$ (given by the red lines). Each parameter has the majority of posterior density over a small range of input values, whereas previously x_1 at least had a wide range of values with non-zero posterior density. x_1 and x_2 both have posterior density concentrated around $\mathbf{x}^*$. This is particularly important in the case of x_2, as this parameter is most important for controlling the strength of the off-diagonal. x_3 is slightly underestimated by the posterior at this wave. However, this is substantially closer than for the wave 1 calibration.

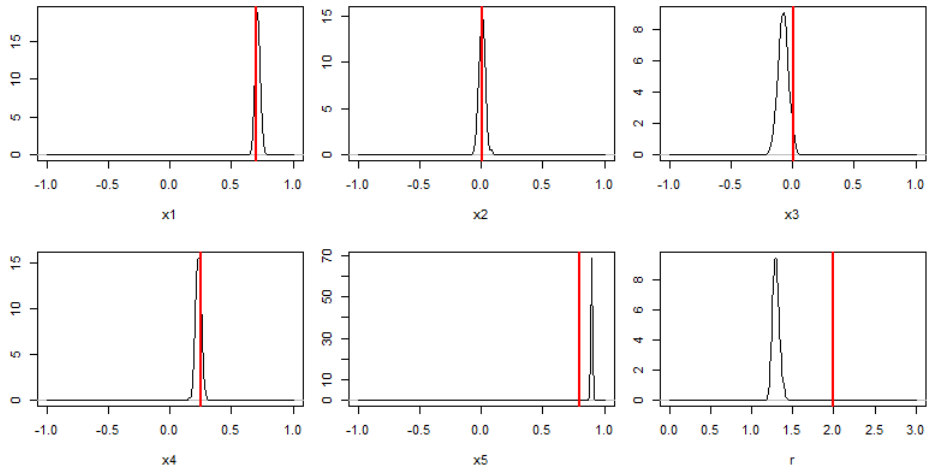

Figure 5.8. The posterior distributions for $x_1, \ldots, x_5$ and the ratio r, when calibration is performed using the wave 2 rotated basis and emulators.

The peak for x_4 is close to the true value of 0.25, although slightly underestimates this. Again, this is an improvement over the wave 1 calibration, with the posterior density now moved closer to the truth. The densities for x_5 and x_6 are away from their true values, however as discussed previously, the ratio $r = \frac{x_5}{1.3+x_6}$ being equal to 2 is more important. The peak for the posterior density of r is located at 1.3, with no density at 2. Although the calibration has improved for each of the other parameters, this is not an improvement on the wave 1 calibration, where this ratio was distributed around 2.06 when the rotated basis was used.

Sampling from the posterior distribution as before and running the toy function at these sampled parameter values gives output shown by Figure 5.9. This is a clear improvement over the wave 1 calibration with the rotated basis (Figure 5.4), where there were large values on the off-diagonal in each sampled field. Here, the key patterns from $\mathbf{z}$ have been identified (i.e. the main diagonal and the corners), with fairly constant values elsewhere. The most important change from the previous wave is that it has now been possible to identify runs where the values on the off-diagonal have been reduced. These fields are not perfect matches for $\mathbf{z}$. The values on the main diagonal are on average 3 less than the truth, due to the underestimation of the x_5, x_6 ratio, r. Similarly, the values in the two corner patterns are around 3 too high in the sampled fields.

In summary, although it has been possible to find more accurate fields, it has not been possible to find fields with values as extreme as in $\mathbf{z}$ on the main diagonal, or in the corners. This is perhaps not a surprising result, as runs exhibiting these values have not

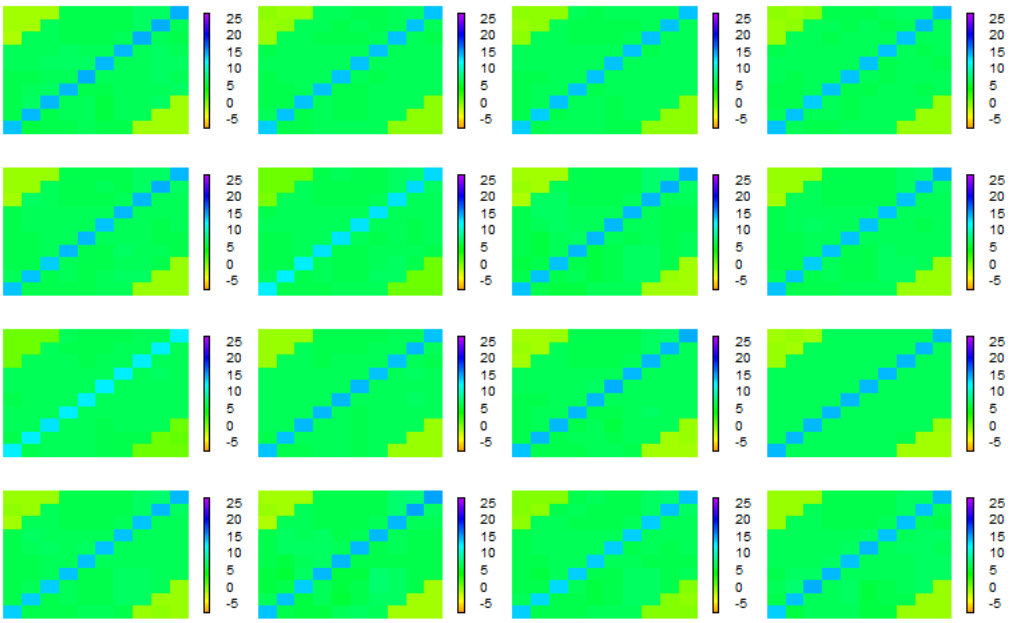

Figure 5.9. $f(\mathbf{x})$ at 16 samples of $\mathbf{x}$ from the wave 2 calibration posterior distribution.

been included in either ensemble, as they are produced by an extremely small region of $\mathcal{X}$. However, rotating the basis to include the important patterns from $\mathbf{z}$ has again successfully highlighted runs in the correct direction.

This again clearly demonstrates the importance of performing multiple waves of history matching and calibration. The wave 2 ensemble has allowed runs in the direction of $\mathbf{z}$ to be found, but if a further ensemble were to be sampled from the new, wave 2 NROY space, it may now be possible to choose runs that contain more signal similar to $\mathbf{z}$, for example with stronger main diagonals. The wave 2 calibration is a huge improvement over wave 1, but the results are not yet as accurate as they could be.

Before continuing with this example, we must further develop our methodology. At the first two waves, the differences between the ensemble runs and the observations were generally large compared to the values of $\boldsymbol{\Sigma}_{\mathbf{e}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$. Now that we have identified a small subset of parameter space containing the true NROY space, better runs and hence superior reconstructions of $\mathbf{z}$ are now possible, and more local patterns can be identified. Subtle changes in the observation error and discrepancy variances can now have larger effects on $\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}, \mathbf{z})$.

5.7. Weighted projection

By the definition of SVD, the projection onto the SVD basis is the projection that minimises the reconstruction error of the ensemble, with respect to the norm $\|\cdot\|_2$, with every dimension of the output weighted equally. If $\mathbf{W} \propto \boldsymbol{\mathcal{I}}_l$, then $\mathcal{R}_{\mathbf{W}}(\cdot,\cdot)$ is the same as $\|\cdot\|_2$. However, if $\mathbf{W}$ is a general positive definite matrix, then an adjustment in the method of projection is required to achieve optimal reconstructions with respect to the weighted norm.

5.7.1. Non-increasing reconstruction error

It can be shown that for a general basis $\mathbf{B}$, the reconstruction error $\mathcal{R}_{\mathbf{W}}(\mathbf{B}_k, \mathbf{z})$ is non-increasing as k increases.

First, define the reconstruction error of the ensemble $\mathbf{F} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ as

$$\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{F}) = \sum_{j=1}^{n} \mathcal{R}_{\mathbf{W}}(\mathbf{B}, f(\mathbf{x}_j))$$

The Eckart-Young theorem states that the best rank q approximation, in terms of minimising the reconstruction error $\mathcal{R}_{\mathbf{W}}(\mathbf{B}_q, \mathbf{A})$ in the norm

$$\|\mathbf{A}\|^2 = \sum_{ij} \mathbf{A}_{ij}^2$$

for a matrix $\mathbf{A}$, is given by the first q vectors of the SVD basis calculated from $\mathbf{A}$ (Eckart and Young, 1936). In our setting, this norm is equivalent to setting $\mathbf{W}$ as a multiple of the identity matrix, so that each output is treated equally. Therefore, in the case of $\mathbf{B} = \boldsymbol{\Gamma}_q$, where $\boldsymbol{\Gamma}_q$ is the first q columns of the SVD basis for the ensemble $\mathbf{F}^T$, and if each output dimension is treated equally ($\mathbf{W} \propto \boldsymbol{\mathcal{I}}_l$), it is not possible to find a basis of rank q that better represents $\mathbf{F}^T$, and hence $\mathcal{R}_{\mathbf{W}}(\mathbf{B}_q, \mathbf{F})$ is minimised by taking $\mathbf{B}_q = \boldsymbol{\Gamma}_q$. It follows from this that the usual SVD projection gives the optimal projection of a general field $f(\mathbf{x})$ onto the basis $\mathbf{B}_q$ for any q, with respect to minimising the reconstruction error for $\mathbf{W} \propto \boldsymbol{\mathcal{I}}_l$.

However, if a different norm is used in the reconstruction error, e.g. the norm given by any

weight matrix that is not a multiple of the identity matrix, then the coefficients given by the usual projection do not give optimal reconstructions with respect to $\mathbf{W}$. A different projection is required to minimise the reconstruction error in $\|\cdot\|_{\mathbf{W}}$, as now grid boxes may not be weighted equally, and there may be correlations between outputs. Improvements in the reconstruction error for some regions of the output space may be weighted more heavily, and increasing q may lead to an increase in $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q, \mathbf{z})$ when using the usual projection method. A reconstruction error that fluctuates as additional basis vectors are included is not useful, as when considering the VarMSE plot, it may lead to conclusions that a basis explaining, say, 80% of the ensemble is superior to one explaining 95% (in terms of best representing $\mathbf{z}$).

For example, it may be the case that the reconstruction in only one grid box is deemed to be important. To do this, $\mathbf{W}$ could be defined as a diagonal matrix as follows:

$$\mathbf{W}_{ij} = \begin{cases} 0.001 & \text{if} \quad i = j = 1 \\ 1000 & \text{if} \quad i = j \neq 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

When $\mathbf{W}$ is inverted, as in $\mathcal{R}_{\mathbf{W}}(\cdot, \cdot)$, these values are inverted, so that in the weight norm, the error in the first dimension is weighted heavily through multiplication by 1000, with all others values given extremely low weights. Essentially, the reconstruction error is dominated by errors in the reconstruction of the first grid box, with patterns elsewhere unimportant.

For the normal SVD projection method, the projection is such that when the next orthogonal basis vector is added, the squared error across the entire reconstructed field is minimised. However, it is possible for the reconstruction of an individual grid box to have an increased error, despite an extra basis vector being included. This is because there may be a coefficient that generally improves the reconstruction across the whole output space by more, trading-off a decrease in accuracy in some parts of the reconstruction for greater general accuracy.

In the constructed scenario above, it is possible that the first grid box does not improve every time a new basis vector is added. Since this is the only grid box of interest according to the defined weight $\mathbf{W}$, a small decrease in accuracy for this grid box results in a

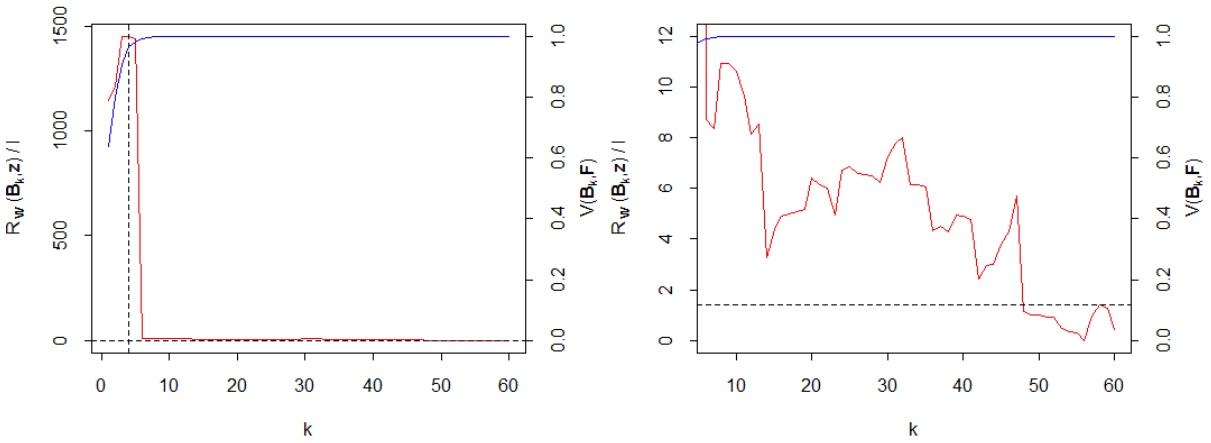

Figure 5.10. The VarMSE plot for the toy function using the SVD basis for the wave 1 ensemble, with the alternative specification for $\mathbf{W}$, with the right plot zooming in on the later basis vectors.

large increase in the weighted error here, and as improvements in any other grid box are comparatively meaningless, the value of the weighted norm could increase significantly, despite the basis now having a higher rank.

This is illustrated by the spatial toy function. Instead of using $\mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ as before, which contains values with a similar magnitude on the diagonal, this could be changed to the version of $\mathbf{W}$ with 0.001 as the first diagonal entry, followed with 1000s as every other diagonal value (equation (5.7)), meaning that it is desired that the value in the first grid box must be accurate, with little interest in values elsewhere. Setting this as the weight for the reconstruction error produces the VarMSE plot in Figure 5.10, with the left plot showing the full plot, and the right plot zooming in to show the behaviour for later basis vectors.

This exhibits the described problems. The reconstruction error after one basis vector has been used for projection is not the maximum error, and the error in this norm increases when the next few basis vectors are added, before a large drop after six basis vectors (as in the standard norm for the SVD basis). Looking more closely at the later basis vectors shows again that the error fluctuates as more basis vectors are added.

This is not a desirable property if the goal is to minimise the reconstruction error. The result here suggests that using only one SVD basis vector is superior to using five in order to achieve more accurate reconstructions, when this ought not to be the case. Fluctuations in the reconstruction error may lead to odd decisions regarding truncation. For example, the SVD basis explaining 80% of the ensemble may reconstruct $\mathbf{z}$ better than the SVD

basis explaining 90% of the ensemble, so therefore the former would be preferred by an optimisation, ignoring the fact that discarded basis vectors increase the uncertainty in reconstructed fields.

As the basis size increases, it would be preferable for the reconstruction error to not increase, for optimisation purposes. To achieve this for a general $\mathbf{W}$, we develop an alternative projection.

5.7.2. Weighted projection

Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a general basis, which need not be orthogonal, with dimension $l \times n$, and $\mathbf{W}$ be an $l \times l$ positive definite matrix. Then, the reconstruction $\mathbf{r} = (r_1, \ldots, r_l)^T$ of $\mathbf{f} = (f_1, \ldots, f_l)^T$ has reconstruction error

$$\|\mathbf{f} - \mathbf{r}\|_{\mathbf{W}} = (\mathbf{f} - \mathbf{r})^T \mathbf{W}^{-1} (\mathbf{f} - \mathbf{r})$$

in the $\mathbf{W}$ norm, where the reconstruction is given by unknown coefficients $\mathbf{c} = (c_1, \ldots, c_n)^T$ such that

$$\mathbf{r} = \sum_{k=1}^{n} \mathbf{b}_k c_k = \mathbf{B}\mathbf{c}$$

The problem is to find $\mathbf{c}$ such that the reconstruction error is minimised for a general vector in l-dimensional space, with respect to the $\mathbf{W}$ norm. Similarly as for deriving least squares estimates, the expression to be minimised is differentiated and set equal to zero. First, expand the expression for the weighted reconstruction error in terms of $\mathbf{c}$:

$$\begin{aligned}
(\mathbf{f} - \mathbf{r})^T \mathbf{W}^{-1} (\mathbf{f} - \mathbf{r}) &= (\mathbf{f} - \mathbf{B}\mathbf{c})^T \mathbf{W}^{-1} (\mathbf{f} - \mathbf{B}\mathbf{c}) \\
&= (\mathbf{f}^T - \mathbf{c}^T \mathbf{B}^T) \mathbf{W}^{-1} (\mathbf{f} - \mathbf{B}\mathbf{c}) \\
&= \mathbf{f}^T \mathbf{W}^{-1} \mathbf{f} - \mathbf{c}^T \mathbf{B}^T \mathbf{W}^{-1} \mathbf{f} - \mathbf{f}^T \mathbf{W}^{-1} \mathbf{B}\mathbf{c} + \mathbf{c}^T \mathbf{B}^T \mathbf{W}^{-1} \mathbf{B}\mathbf{c}
\end{aligned}$$

Next, we differentiate this expression with respect to $\mathbf{c}$, using the following results for differentiating a scalar by a vector $\mathbf{x}$, for a symmetric matrix $\mathbf{A}$:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{y} &= \frac{\partial}{\partial \mathbf{x}} \mathbf{y}^T \mathbf{x} = \mathbf{y}^T \\
\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{x} &= 2\mathbf{x}^T \mathbf{A}
\end{aligned}$$

Applying these to the previous expression, and using that $\mathbf{W}$ is positive definite, and hence symmetric, so that $\mathbf{W}^{-1}$ is also positive definite and symmetric:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{c}}\|\mathbf{f}-\mathbf{r}\|_{\mathbf{W}} &= 0-(\mathbf{B}^T\mathbf{W}^{-1}\mathbf{f})^T-\mathbf{f}^T\mathbf{W}^{-1}\mathbf{B}+2\mathbf{c}^T\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}\\ &= -\mathbf{f}^T\mathbf{W}^{-1}\mathbf{B}-\mathbf{f}^T\mathbf{W}^{-1}\mathbf{B}+\mathbf{c}^T\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}+\mathbf{c}^T\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}\\ &= -2\mathbf{f}^T\mathbf{W}^{-1}\mathbf{B}+2\mathbf{c}^T\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}\end{aligned}$$

Setting this equal to zero, and solving for $\mathbf{c}$ gives:

$$\begin{aligned}0 &= -2\mathbf{f}^T\mathbf{W}^{-1}\mathbf{B}+2\hat{\mathbf{c}}^T\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}\\ &\implies \hat{\mathbf{c}}^T\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}=\mathbf{f}^T\mathbf{W}^{-1}\mathbf{B}\\ &\implies \mathbf{B}^T\mathbf{W}^{-1}\mathbf{B}\hat{\mathbf{c}}=\mathbf{B}^T\mathbf{W}^{-1}\mathbf{f}\\ &\implies \hat{\mathbf{c}}=(\mathbf{B}^T\mathbf{W}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}^{-1}\mathbf{f}\end{aligned}$$

If the weight is the identity matrix, this expression simplifies to give the usual projection equation:

$$\hat{\mathbf{c}}=(\mathbf{B}^T\boldsymbol{\mathcal{I}}_l^{-1}\mathbf{B})^{-1}\mathbf{B}^T\boldsymbol{\mathcal{I}}_l^{-1}\mathbf{f}=(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{f}$$

For a given weight, this definition of the coefficients on the basis gives a superior projection to the standard SVD projection, and therefore should be used when attempting to minimise the reconstruction error of a basis when grid boxes are not equally weighted. This projection should have been used for the first two waves of our toy example, as $\mathbf{W}$ was not proportional to the identity matrix. As this projection is optimal for a given basis, the reconstruction error will not increase as orthogonal basis vectors are added.

As an alternative to this, it is possible to weight the ensemble rather than using a different projection. Weighting the ensemble prior to performing SVD then allows the standard projection method and norm to be used in the subsequent basis selection process (see generalised SVD/PCA, an overview of which is given in Jolliffe (2002)). This may be a suitable approach to take if a weight matrix is known and fixed. Then, SVD need only be performed once with this weighted ensemble.

A benefit of applying the weighting via the projection is that an SVD need only be calculated once. Performing SVD may be time-consuming for high l and n. If $\mathbf{W}$ is not known, then various projections with different values for this can be considered without the need for performing SVD multiple times.

5.8. Refocussing continued: wave 3

We now perform a further wave of history matching and calibration. We continue to use $\mathbf{W} = \mathbf{\Sigma_e} + \mathbf{\Sigma_\eta}$ as at the previous waves, but as we are now interested in smaller deviances between the observations and its reconstructions, this weight has a greater effect. Hence, we use the weighted projection to calculate the basis coefficients.

We sample a new ensemble $\mathbf{F}^{(3)}$ from the NROY space defined at wave 2, using the sampling method that was introduced in Section 5.6.1, with the new ensemble containing a range of wave 2 implausibilities. None of the wave 2 ensemble runs lie in the current NROY space. The SVD basis is calculated for the centred version of $\mathbf{F}^{(3)}$, and is shown in Figure 5.11. The first vector in this basis, explaining 87% of the variability in the wave 3 ensemble, contains the key patterns from the observations. This is expected, given that history matching has been ruling out runs dissimilar to this, intending to leave a space containing runs consistent with $\mathbf{z}$.

The difference that the alternative projection makes to $\mathcal{R}_{\mathbf{W}}(\cdot, \cdot)$ is also shown in Figure 5.11, with the standard projection given by the dotted lines. The error for the standard projection fluctuates slightly as more orthogonal basis vectors are added, whereas the weighted projection consistently decreases (or at least, does not increase), as it should. The fluctuations in the error are not as great as in the example from Figure 5.10 as the values in $\mathbf{\Sigma_e}$ and $\mathbf{\Sigma_\eta}$ do not vary as greatly. The error with the weighted projection is always less than the error with the standard projection for the same basis (as the weighted projection is optimal for this weight).

The reconstruction error, with either projection used, is below the history matching bound for the first three SVD basis vectors (those required to explain more than 95% of the new ensemble). The NROY space that has been sampled from to design the wave 3 ensemble has been restricted substantially, so that runs in the ensemble are more similar to the observations than previously, hence it is not surprising that the SVD basis is now suitable to be used for projection, and no rotation is required here. A rotation could be performed, as there is a potential improvement to be made because the minimum with the full SVD basis is lower than the error with the first three. In this case, emulation is straight-forward on the SVD basis, so we do not apply the rotation algorithm, and instead use the truncated

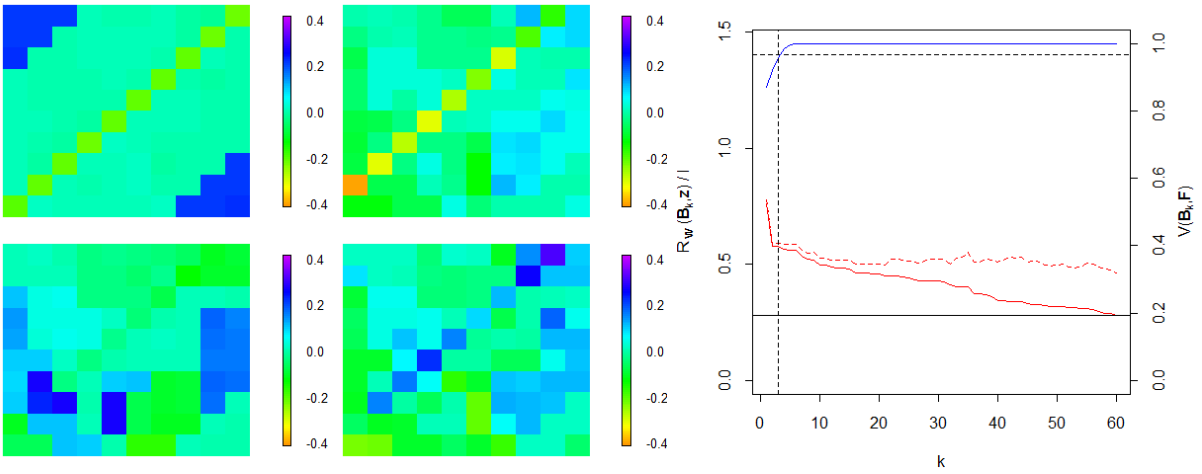

Figure 5.11. The first four basis vectors for the wave 3 SVD basis. The VarMSE plot, with $\mathbf{W} = \boldsymbol{\Sigma}_{\mathbf{e}} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$, shows the error when the weighted projection is used (solid red line) and the standard projection (dotted red line), with the same $\mathbf{W}$.

SVD basis at this wave. We build emulators for the (weighted) coefficients for the first three basis vectors, with diagnostics shown in Appendix B.3.

5.8.1. History matching

Performing history matching using the wave 3 emulators, 66.7% of the existing NROY space is not ruled out at this wave, leaving 2.04% of $\mathcal{X}$ in NROY space after three waves. Figure 5.12 shows this NROY space, with a different scale used than previously so that it is easier to identify which regions of space still contain runs that are not ruled out.

The main difference between the true NROY space and the wave 3 NROY space is that x_4, x_5 and x_6 have not yet been constrained enough, in relation to the x_2 and x_3 values. The former three parameters are those that have most effect on the main diagonal, so that these have generally been harder to restrict as the amount of signal from these was limited initially. A further wave of history matching is likely to rule out some of the more incorrect values of these parameters, as now the effect that they have should be more apparent.

5.8.2. Bayesian calibration

We carry out Bayesian calibration using the wave 3 emulators, with zero prior probability assigned to parameter settings outside of the wave 2 NROY space.

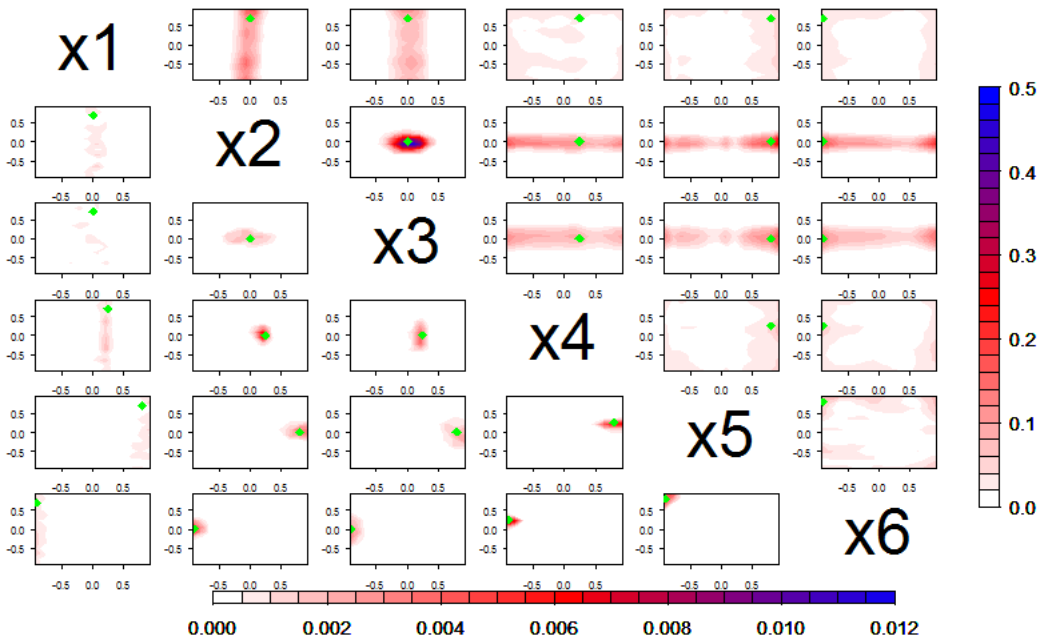

Figure 5.12. Density plot for the wave 3 NROY space defined using the SVD basis (upper right), and the true NROY space (lower left), for each pair of parameters.

The posterior distributions for the input parameters are given in Figure 5.13. The main change from the wave 2 posteriors is that x_1 no longer has a narrow peak of posterior density, with the density now spread across the entire range of this parameter. This is as expected, because the true NROY space contains points for all x_1. The posterior density for x_2 is again concentrated around its true value, and x_3 is slightly underestimated, as at wave 2.

For x_4, the peak of density is slightly further away from the true value than at the previous wave. However, this parameter affects the main diagonal in combination with x_5 and x_6, and the posteriors for these two parameters are an improvement: the ratio r has density at 2, and all of the density is closer to 2 than at the previous wave.

Sampling from the posterior gives output fields as shown by Figure 5.14. From this, it is clear that the calibration has been more accurate than at the previous wave. Now, the values on the main diagonal are the correct strength for several of the sampled runs, whereas previously this pattern was too weak. Furthermore, the values in the corners are much closer to those in $\mathbf{z}$. Not all of the runs from this sample are close matches to $\mathbf{z}$, with some of the runs still exhibiting too low values on the main diagonal, but overall, performing an extra wave of history matching prior to Bayesian calibration has improved the results.

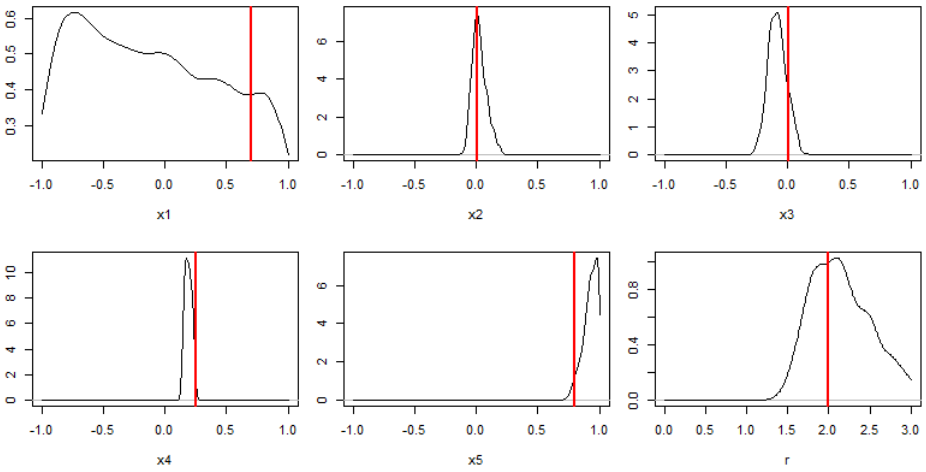

Figure 5.13. The posterior distributions for $x_1, \ldots, x_5$ and r, when calibration is performed using the wave 3 SVD basis and emulators.

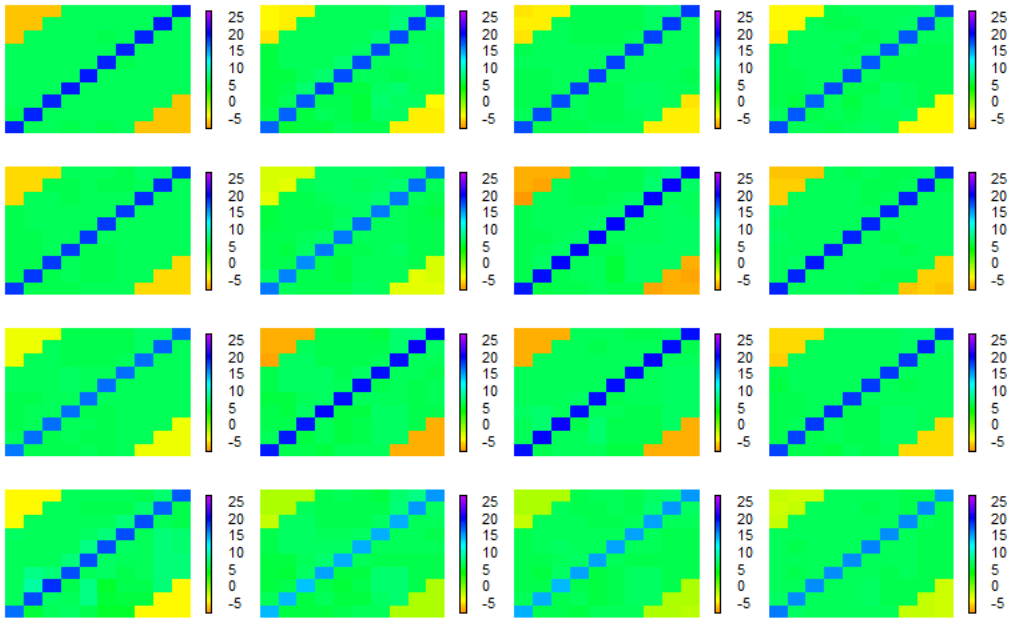

Figure 5.14. $f(\mathbf{x})$ at 16 samples of $\mathbf{x}$ from the wave 3 calibration posterior distribution.

5.9. Discussion

In this chapter, we introduced an automatic procedure for selecting a basis to be used for projecting spatial fields onto, expanding on the ideas and methods introduced in Chapter 4. We developed an optimisation criteria, based on the reconstruction error $\mathcal{R}_{\mathbf{W}}(\cdot, \mathbf{z})$, applying rotations to an existing basis. Here, the basis that was rotated was always the SVD basis, however there is no reason why this method could not be applied to a general basis.

The basis rotations considered here are all performed by multiplying a basis on the right by

a rotation matrix. This restricts the resulting rotated basis to lie in the same n-dimensional subspace of l-dimensional space as the original basis, rather than the complete flexibility that would be allowed by rotation on the left. However, rotation as performed above is preferred as it not only reduces the number of parameters that must be determined, but also retains the ability to explain the variability in the ensemble. While minimising the reconstruction error for the observations with the rotated basis, being able to explain the ensemble is important so that emulators may be built for the basis coefficients.

The optimisation criteria introduced in Section 5.3 provide a trade-off of these two competing goals, with a minimum percentage of ensemble variability set for each basis vector. We showed that the full SVD basis $\mathbf{\Gamma}$ gives the minimum reconstruction error that it is possible to achieve with a rotation of this basis, $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}, \mathbf{z})$. Therefore, this can be used as an initial check for whether rotation is able to overcome any problems with the SVD basis, in terms of representing the observations. If the reconstruction error is large, but decreases little or not at all after the basis is truncated, then a rotation defined as above will not be able to improve the basis choice. However, if the error does decrease for the full SVD basis, then by applying the rotation algorithm, it is possible to incorporate the important parts of low-eigenvalue basis vectors into the truncated basis that can be used for emulation.

The final algorithm introduced in Section 5.4 provides a further dimension reduction in terms of the number of parameters that must be optimised. We showed that defining general multiples for basis vectors, instead of angles, results in a rotation matrix when combined with Gram-Schmidt. Exploiting the fact that the rotated basis contains linear combinations of the original basis vectors, these vectors may instead be selected iteratively, until enough ensemble variability is explained, and the reconstruction error minimised adequately - for example, so that this is 'close' to the minimum found with the full SVD basis. This greatly reduces the complexity of the optimisation problem, and by combining this with the residual basis methodology from Section 4.6.1, it finds a basis that is suitable for searching in the direction of $\mathbf{z}$, while explaining the ensemble.

This method is a clear improvement over the basis selection procedure of the previous chapter, as there is no longer any reliance on eliciting a physically important pattern, and no direct inclusion of the observations in the basis is required. Instead, the rotation exploits the signal that is contained within the ensemble, and rearranges it in a manner

more suited to the problem at hand. The only choices that are required in order to apply the algorithm are the weight, $\mathbf{W}$, to be used (and here, this has always been the sum of the observation error and discrepancy variances, to retain the comparison with history matching), and the minimum percentage of variability for each basis vector. Several rotations may be required, with differing minimums, until a suitable one that can be used for emulation and calibration is found. However, due to the dimension reduction achieved by treating rotation as linear combinations of the original basis vectors, and by performing the rotation iteratively, this is not a great expense: for the 60 member ensembles used in this chapter, setting a maximum time of 100 seconds usually provided a suitable solution for a given variance threshold.

This highlights a useful aspect of this method: the number of rotation parameters to be optimised scales with the ensemble size n, rather than the size of the spatial field. In problems with expensive computer models, n may not be much larger than the 60 in this example, and hence the difficulty of the optimisation may not increase by much. The larger computational burden for higher-dimensional problems is likely to lie in calculating the inverse (or Cholesky decomposition) of the $l \times l$ matrix $\mathbf{W}$, although this is a one-off cost.

Later in the chapter, it was noted that the reconstruction error for a given weight matrix $\mathbf{W}$ does not always decrease as the number of basis vectors increases, when the standard SVD projection method is used. This is due to the norm used in the calculation of the SVD basis, so that projections are optimal if each output is weighted equally. However, in the case where there is known observation error, discrepancy, or any other weighting to represent the importance of certain regions of the output field, this projection does not give optimal projections with respect to this non-constant weight. We derived an alternative projection (Section 5.7) so that this is consistent with the norm used to calculate the reconstruction error, and hence this error always decreases as orthogonal basis vectors are added.

If the observation error and discrepancy variances are unknown in a problem, and it is not possible to elicit any beliefs regarding the relative importance of different regions of output space, then setting the weight proportional to the identity is reasonable. Then, the reconstruction error is simply the mean squared error, and the standard projection can be used. For example, it may be known that the discrepancy of an output is, say, 2°C

in every grid box, with no known correlations. Therefore, the weight can be set to reflect this knowledge, but as every grid box is still treated equally, the usual SVD projection is still optimal, and the reconstruction error is divided by the multiple of the identity.

A possible drawback of the rotation method is that the rotation is restricted to the subspace defined by the ensemble through the SVD basis. Therefore, if reconstructions of $\mathbf{z}$ are poor, even with the full SVD basis, this method will not be able to fix the basis choice. However, the rotated basis will give reconstructions as close as possible to $\mathbf{z}$, while allowing emulators to be built as the ensemble signal is incorporated into the basis.

This feature lends itself to an iterative, refocussed approach to calibration, as demonstrated through the toy example. At the first wave, there was some signal in the direction of $\mathbf{z}$, and the rotated basis reflected this by achieving reconstructions of $\mathbf{z}$ that would not be ruled out in the 'perfect' history matching case (i.e. no emulators involved, simply projection and reconstruction of the field). Although the posterior distribution obtained through Bayesian calibration with the rotated basis gave more accurate runs than when the SVD basis was used, the true best parameter setting was not found.

By sampling from NROY space, and performing a second and third wave of history matching and Bayesian calibration, runs consistent with $\mathbf{z}$ were identified by the posterior distribution. The rotated basis enabled runs with little or no signal in the direction of $\mathbf{z}$ to be removed by history matching, so that the wave 2 and 3 ensembles contained more runs with signal on the main diagonal. This enabled more accurate emulators to be built for this important direction, so that parameter settings consistent with $\mathbf{z}$ could be found. By wave 3, the SVD basis itself was suitable for history matching, as it was defined using ensemble members more similar to $\mathbf{z}$ than at previous waves. The results of the wave 3 calibration were a significant improvement over the wave 1 results, demonstrating the power of reducing space with history matching prior to Bayesian calibration. This gives a greater chance of being able to emulate the patterns of interest - but only if the basis allows for the correct directions to be searched in initially.

5.10. Conclusion

In this chapter, we developed an algorithm for finding a calibration-optimal basis, via a rotation of the SVD basis, demonstrating the improvements this achieved over the SVD basis in calibration and history matching.

Where possible, performing multiple waves of history matching prior to Bayesian calibration should be preferred, allowing regions of space clearly inconsistent with the observations to be removed, so that new runs can be evaluated in low implausibility regions of parameter space. By rotating the SVD basis, this procedure is improved as the goal of finding $\mathbf{z}$ is built into the basis selection, rather than allowing the ensemble to completely control the process.

Rotating the basis with respect to being able to represent the observations with the first few basis vectors, by using the information from the SVD basis, is straight-forward to apply, requiring few choices by the user. It is easy to check whether a rotation is required, and hence this should always be carried out prior to a calibration exercise for a spatial field.

This method has proved successful on the 100-dimensional toy function, but of greater importance is emulating and calibrating computer models with higher-dimensional output. In the following chapter, this method is applied to the climate model CanAM4, to show the scalability of this method, and of calibration methods generally.

6. Iterative history matching of CanAM4

6.1. Introduction

Tuning a climate model is an important and challenging problem. As well as large output fields, for which the issue of high-dimensionality must be overcome, there are often multiple different outputs that need to be considered jointly. History matching can be applied to this type of tuning problem to determine parameter settings not inconsistent with the observations, similarly as in the previous chapters, although extensions may be required due to the inversions of large variance matrices that must be calculated to give the multivariate implausibility.

In this chapter, we discuss methods for history matching large spatial fields, with the goal of performing history matching for the 8192-dimensional output fields of CanAM4, and designing a new ensemble to be run on CanAM4. As in Chapters 4 and 5, we emulate the output fields via projection onto a low-dimensional basis, and hence selecting a suitable basis prior to emulation is once again important.

To select a basis for this problem, we apply the iterative rotation method of Chapter 5, with the reconstruction error of the observations, $\mathcal{R}_{\mathbf{W}}((\mathbf{\Gamma\Lambda})_q, \mathbf{z})$, minimised subject to constraints on the variability. Iteratively selecting new basis vectors gave accurate calibration and history matching results for the 100-dimensional toy function. Being able to apply this method for larger fields, as commonly found for climate models, is important, and hence we apply this method to three of the output fields of CanAM4. The number of rotation parameters that must be optimised to find a new basis is similar as for the toy function in Chapter 5 due to the small ensemble size n for CanAM4, although larger matrices will be involved in every calculation. The number of degrees of freedom in this problem, n, is much smaller than the dimension of the output, l, so that it may be difficult

to represent the observations, $\mathbf{z}$, accurately using linear combinations of the SVD basis.

Section 6.2 discusses how to approach history matching for large output fields, and provides a method for linking the implausibility on the field with the implausibility on the coefficients, giving a more conservative bound for ruling out runs based on the coefficients. Section 6.3 applies the basis rotation and coefficient history matching methodology to CanAM4, with a method for specifying the discrepancy given in Section 6.3.2, and a new ensemble based on NROY space designed in Section 6.3.5. Section 6.4 analyses this new wave, discussing where improvements have been made, and performing a second wave of emulation and history matching for CanAM4, with a new spatial discrepancy developed in Section 6.4.3.

6.2. High-dimensional calibration

In order to history match or calibrate CanAM4, the methods applied in Chapter 4 and 5 require extensions. Projecting onto a rotated basis, building emulators for the coefficients, and using reconstructed fields to history match or calibrate was shown to give accurate results for the 100-dimensional toy example, and hence the question to answer is whether these scale up for output more than 80 times as large, given appropriate extensions.

The Bayesian calibration method used (Wilkinson, 2010) requires the inversion of an $l \times l$ variance matrix, $\mathbf{\Sigma}(\mathbf{x}) = \mathrm{Var}(f(\mathbf{x})) + \mathbf{\Sigma_e} + \mathbf{\Sigma_\eta}$, every time the likelihood is calculated, where l is the dimension of the output field. $\mathbf{\Sigma}(\mathbf{x})$ varies as the input parameters $\mathbf{x}$ vary, as the emulator variance is unlikely to be constant for all $\mathbf{x}$, and therefore performing MCMC on the posterior distribution is an extremely time-consuming process. Therefore, when applied to climate models, Bayesian calibration has generally been carried out on the basis vector coefficients, as in Sexton et al. (2011).

For history matching, scalar summaries of the output have generally been matched to, rather than the entire output field (Williamson et al., 2015). The multivariate implausibility of Craig et al. (1997) was applied in the previous two chapters, analogously to the calibration method of Wilkinson (2010). As for calibration, this is not as practical in high dimensions: for every $\mathbf{x}$, to calculate its implausibility, $\mathbf{\Sigma}(\mathbf{x})$ must be inverted. As history matching requires thousands or millions of these evaluations, this may be too

time-consuming for CanAM4. Calculating the Cholesky decomposition of an 8192×8192 variance matrix, as required to calculate the implausibility or likelihood for a single parameter setting $\mathbf{x}$ for the majority of outputs of CanAM4, takes 78 seconds on a desktop computer. To evaluate this for even 10,000 different parameter settings would take more than 200 hours, and generally, more of these inversions are needed.

Performing history matching for the climate model using the same method as for the toy example is clearly not feasible, particularly in an interactive tuning session, so that modellers do not have to wait over a minute for the fast statistical approximation of the model output.

Instead of history matching using the reconstructed versions of the fields, an implausibility can be calculated using the coefficients, as in (2.38):

$$\mathcal{I}_c(\mathbf{x}) = (\mathbf{c}(\mathbf{z}) - \mathrm{E}[\mathbf{c}(\mathbf{x})])^T(\mathrm{Var}[\mathbf{c}(\mathbf{x})] + \mathrm{Var}[\mathbf{c}(\mathbf{e})] + \mathrm{Var}[\mathbf{c}(\boldsymbol{\eta})])^{-1}(\mathbf{c}(\mathbf{z}) - \mathrm{E}[\mathbf{c}(\mathbf{x})]) \quad (6.1)$$

with a c subscript indicating that this implausibility is for the coefficients. This is a similar measure as that used in Bayesian calibration by Chang et al. (2014b), however we will discuss its deficiencies in this chapter. The projection of $\mathbf{z}$ onto the basis, $\mathbf{c}(\mathbf{z})$, is given by

$$\mathbf{c}(\mathbf{z}) = (\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}^T\mathbf{z}$$

The emulator expectations and variances for the first q basis vectors are arranged so that

$$\mathrm{E}[\mathbf{c}(\mathbf{x})] = (\mathrm{E}[c_1(\mathbf{x})], \ldots, \mathrm{E}[c_q(\mathbf{x})])^T$$
$$\mathrm{Var}[\mathbf{c}(\mathbf{x})] = \mathrm{diag}(\mathrm{Var}(c_1(\mathbf{x})), \ldots, \mathrm{Var}(c_q(\mathbf{x})))$$

The projections of the observation error and discrepancy variances $\mathrm{Var}(\mathbf{c}(\mathbf{e}))$ and $\mathrm{Var}(\mathbf{c}(\boldsymbol{\eta}))$ are given by

$$\mathrm{Var}[\mathbf{c}(\mathbf{e})] = (\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-1}\boldsymbol{\Gamma}^T\boldsymbol{\Sigma}_\mathbf{e}\boldsymbol{\Gamma}(\boldsymbol{\Gamma}^T\boldsymbol{\Gamma})^{-T}$$

By only requiring a $q \times q$ matrix to be inverted to calculate the implausibility at $\mathbf{x}$, history matching a large spatial field becomes tractable. The NROY space using this implausibility is then defined as

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_c(\mathbf{x}) < \chi^2_{q,0.995}\}$$

where q is the number of basis vectors for which emulators have been built, and hence is

the number of coefficients being matched to.

For clarity, the multivariate implausibility that has been applied for the emulator reconstructions of fields in Chapters 4 and 5 will be denoted $\mathcal{I}_f(\mathbf{x})$ hereafter, and termed the 'field implausibility', while $\mathcal{I}_c(\mathbf{x})$ as in (6.1) will be referred to as the 'coefficient implausibility'.

6.2.1. Coefficient implausibilities for the toy function

Due to the speed with which both the coefficient and field implausibilities can be evaluated for the spatial toy function of the previous two chapters, prior to history matching the climate model using the coefficient implausibility, we first use $\mathcal{I}_c(\mathbf{x})$ to history match the toy function, and compare the results to those found with the field implausibility.

A point that should be addressed here is that the true NROY space has been defined with respect to the field implausibility being calculated for the true function output, as it is quick to evaluate for the toy function. A true NROY space could equivalently be defined using the coefficient implausibility for a given basis. However, given that this loses information, defining the true NROY space using the field implausibility, and the true output fields, is more sensible. This compares the true value of the observations to the function output in each grid box, rather than a small number of summaries for the observations and model output.

Therefore, having $\mathbf{x}$ where

$$\mathcal{I}_c(\mathbf{x}) > \chi^2_{q,0.995} \quad \text{for} \quad \mathcal{I}_f(\mathbf{x}) < \chi^2_{l,0.995}$$

so that the spatial field is within our tolerance to error, but is ruled out using the coefficients, would be a problematic result.

At wave 1 with the rotated basis, history matching on the field gave an NROY space consisting of 31.49% of the input space $\mathcal{X}$ (Section 5.5.2). By instead calculating the coefficient implausibility for the same input parameters, an alternative NROY space is defined, based on the chi-squared bound with $q = 5$ (as emulators were built for the first five basis vectors). The result of this history match is a space containing 8.56% of $\mathcal{X}$.

At first glance, this appears to have performed better, given that it is known that the true NROY space is around 0.01% of the parameter space. However, the accuracy with which runs are ruled out is more important than simply removing as much space as possible. Calculating the coefficient implausibility for runs that are known to lie in the true NROY space, it is found that 16% of these are ruled out (compared to none being ruled out when the field implausibility was used). Although the coefficient implausibility has allowed more space to be ruled out, the cost of this is losing some accuracy, and removing some parameter settings that are in fact consistent with the observations on the field.

The left plot of Figure 6.1 plots $\mathcal{I}_f(\mathbf{x})$ and $\mathcal{I}_c(\mathbf{x})$ for 1000 parameter settings (with an additional 100 from the true NROY space), for the wave 1 emulators. This plot highlights that there is a strong positive correlation between the two implausibilities. For the coefficient implausibility to be useful, this would need to be the case: fields that are generally similar to $\mathbf{z}$, and hence give a low field implausibility, should also give a low coefficient implausibility, and not be ruled out if history matching were performed on the coefficients instead.

The dotted lines in Figure 6.1 give the 0.995 chi-squared bounds for the field and coefficient implausibilities, and illustrate the problem highlighted above: fewer runs are in NROY space for the coefficient implausibility, with some of the runs that lie in the true NROY space (coloured green) being greater than the chi-squared bound for the coefficient implausibility, and hence being incorrectly ruled out.

One simple fix in this case is to use a more conservative bound to rule out runs for the coefficient implausibility: instead of using the 0.995 value of the chi-squared distribution with 5 degrees of freedom, a higher quantile could be used. Using the 0.9995 value of this distribution to define NROY space, 14.11% of the original parameter space is not ruled out, including 97.1% of the true NROY space, a more satisfactory result. By increasing the bound further, with the 0.99995 value used, 100% of the true NROY space is kept, with 18.81% of the whole parameter space not ruled out.

For the wave 1 emulators, by increasing the bound, the accuracy of history matching using the coefficient implausibility has been improved, and the resulting NROY space is still smaller than that defined using the field implausibility (31.49%). If using a more conservative value of the chi-squared distribution to rule out runs is found to work generally

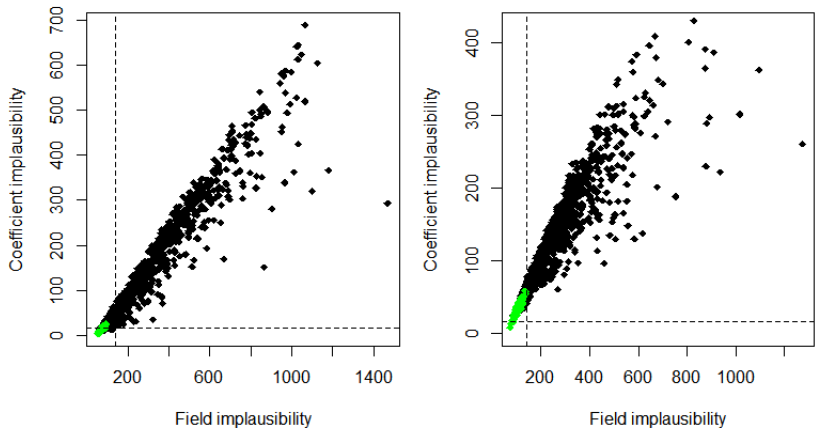

Figure 6.1. Plots comparing the coefficient implausibility $\mathcal{I}_c(\mathbf{x})$ with the field implausibility $\mathcal{I}_f(\mathbf{x})$, for the wave 1 emulators for the rotated basis (left), and the wave 2 emulators, with the implausibilities evaluated at a sample of 1000 points from $\mathcal{X}$, and 100 points from the true NROY space (coloured green). The vertical dotted line represents the history matching bound for $\mathcal{I}_f(\mathbf{x})$, and the horizontal dotted line the bound for $\mathcal{I}_c(\mathbf{x})$, each using the 0.995 value of the corresponding chi-squared distribution.

for the coefficient implausibility, then this could be applied when we history match the climate model.

To test whether increasing the chi-squared bound is generally applicable, we now consider history matching with the coefficient implausibility for the wave 2 emulators for the toy function, compared to the use of the field implausibility in Section 5.6.2. Using the field implausibility led to an NROY space containing 3.06% of parameter space, with none of the true NROY space ruled out (Table 6.1). By calculating the coefficient implausibilities instead, using the standard 0.995 value from the chi-squared distribution with 5 degrees of freedom, the NROY space only consists of 0.009% of $\mathcal{X}$. Increasing the bound to the 0.99995 value still only keeps 0.1% of all parameter settings.

This appears to have ruled out too much space, and by considering only runs that lie in the true NROY space, it is found that only 3.62% are in NROY space (using the 0.995 value), or 28.99% when the 0.99995 value defines whether runs are ruled out. Whilst increasing the bound to that given by the 0.99995 value succeeded in keeping all of the true NROY runs at the first wave, at the second wave too much space has been ruled out, with a large proportion of important parameter values ruled out, despite using the more conservative bound.

The right hand plot of Figure 6.1 illustrates the problem at wave 2. From this, it is seen clearly that parameter settings consistent with $\mathbf{z}$ are ruled out by $\mathcal{I}_c(\mathbf{x})$ and the chi-squared

Wave	$\mathcal{I}_f$	$\mathcal{I}_c$
1	31.49% (100%)	8.56% (84%)
2	3.06% (100%)	0.009% (3.62%)

Table 6.1. The size of the NROY space at waves 1 and 2, when the field and coefficient implausibilities are each used, with the bound set as the 0.995 value of the chi-squared distribution with the associated number of degrees of freedom. The percentages in brackets indicate how much of the true NROY space is not ruled out.

bound: all of the green points lie to the left of the bound for $\mathcal{I}_f(\mathbf{x})$, but it is not true that all, or even the majority of these points, are below the bound for $\mathcal{I}_c(\mathbf{x})$, given by the horizontal dotted line. The two different bounds used to rule out points are not directly related.

This highlights a problem with using the coefficient implausibility to define NROY space for spatial problems: what value should be used to rule out parameter settings? From these two examples, it appears that it is not possible to use the same value from the chi-squared distribution to set the bound. Rather than using a fixed value, a problem-specific bound may be more suitable.

6.2.2. Selecting a conservative bound for history matching

The previous section demonstrated that when history matching using the basis coefficients, selecting the bound to define NROY space is critical, and may not be as straight-forward as using a quantile of the chi-squared distribution. Setting the wrong value can result in parameter settings that lead to output consistent with $\mathbf{z}$ being ruled out incorrectly, as shown in Figure 6.1. Using the chi-squared values as the bound, more space was ruled out than when the field implausibility was used, suggesting that a method for selecting a more conservative bound is necessary.

If possible, it would be preferable to use the field implausibility to rule out runs, due to the fact this takes into account every grid box, and the difference between the observations and emulated fields in each of these. Although it is not possible to calculate this at thousands or millions of points, it could however be evaluated at a small number of points. Then, the positive relationship between the two implausibilities could be exploited, to infer an appropriate bound to be used to rule out parameter settings with the coefficient implausibility.

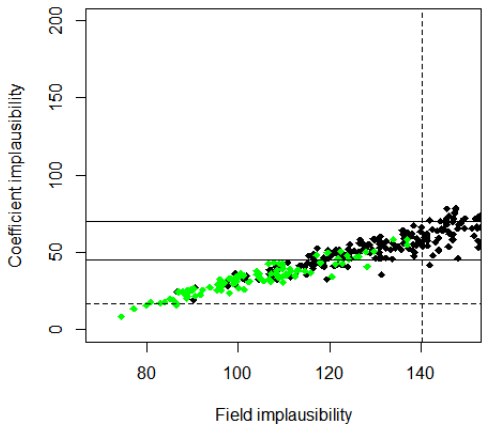

Figure 6.2. The wave 2 implausibilities as in the right hand plot of Figure 6.1, with solid horizontal lines added to show the range of possible $\mathcal{I}_c(\mathbf{x})$ values at $\mathcal{I}_f(\mathbf{x}) = 140.2$.

For example, consider selecting a bound for the wave 2 history matching for the toy function. Almost the entirety of $\mathcal{X}$ is ruled out using the chi-squared bound for $\mathcal{I}_c(\mathbf{x})$, whereas the field implausibility keeps around 3% of space. The chi-squared bound for the field is 140.2, but from Figure 6.2 (a plot showing the lowest implausibilities from the right hand plot in Figure 6.1), it is seen that $\mathcal{I}_f(\mathbf{x}) = 140.2$ can correspond to values of $\mathcal{I}_c(\mathbf{x})$ ranging from 45 to 70 (highlighted by the two horizontal solid black lines). Therefore, selecting a bound for $\mathcal{I}_c(\mathbf{x})$ 'by eye' using this plot, one might decide that 70 is a reasonable bound to define the NROY space for the coefficient implausibility.

Returning to history matching at wave 2 using this new bound, it is found that 5.33% of $\mathcal{X}$ is considered NROY space. More importantly, 100% of the true NROY space is now not ruled out. While this choice of bound rules out less space than the field equivalent, it is more accurate than the chi-squared bound for the coefficients. Having a more conservative bound to guard against incorrectly ruling out parameter settings leading to output consistent with $\mathbf{z}$ is clearly preferable to ruling out these parameters.

In higher dimensions, it will not generally be feasible to evaluate the field implausibility as often as in this example: if the coefficient implausibility is to be used in a problem, this is because it is not computationally efficient to calculate the field implausibility often enough. Therefore, there will not be as large a sample of the two implausibilities to compare, and hence there may not be as clear a relationship, or as many coefficient implausibilities corresponding to the bound for $\mathcal{I}_f(\mathbf{x})$, as in this case. We instead model this relationship.

6.2.3. Modelling $\mathcal{I}_c$ vs $\mathcal{I}_f$

Based on a small space-filling sample of $\mathbf{x}$ values, we define a model to relate the coefficient implausibility to the field implausibility. Given this model, we predict the bound to be used for ruling out space with the coefficient implausibility.

This approach is similar in nature to multi-level or hierarchical emulation (Oughton and Craig, 2016), although the opposite. Only a few evaluations of the more complicated field implausibility are possible, but using these, history matching with the coefficient implausibility can be informed, via the link to the field implausibility (rather than using the coefficient implausibilities to predict the field implausibilities for any $\mathbf{x}$).

From Figure 6.1, it is clear that there is a positive relationship between the two implausibilities. It is also observed that the variance increases with $\mathcal{I}_f$, so that constant variance as in linear regression is not a reasonable assumption. We model the relationship using a set of weights to reflect this non-constant variance, as in weighted least squares. Given a sample of m pairs of implausibilities $\mathbf{I} = \{\mathcal{I}_f(\mathbf{x}_i), \mathcal{I}_c(\mathbf{x}_i)\}_{i=1}^m$, we fit the following model:

$$\begin{aligned} \mathcal{I}_c(\mathbf{x}_i) &= \beta_0 + \beta_1 \mathcal{I}_f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathrm{N}(0, \frac{\sigma_\epsilon^2}{w_i}) \\ \log(w_i) &= \alpha_0 + \alpha_1 \log(\mathcal{I}_f(\mathbf{x}_i)) \end{aligned} \tag{6.2}$$

where $\boldsymbol{\psi} = (\beta_0, \beta_1, \alpha_0, \alpha_1, \sigma_\epsilon^2)$ are all unknown parameters, and w_i are weights. For the variance parameter, we set the prior as

$$\pi(\sigma_\epsilon^2) \propto \sigma_\epsilon^{-2}$$

For the regression coefficients, we know that the coefficient implausibility and field implausibility have a positive relationship, hence we set the prior for β_1 to reflect this, e.g.

$$\pi(\beta_1) \sim Gamma(k_1, \theta_1)$$

or any other distribution that constrains β_1 to be positive. For α_1, we have the opposite relationship, with the weight decreasing with $\mathcal{I}_f$, so that the variance of $\mathcal{I}_c$ increases with $\mathcal{I}_f$. A Gamma prior is appropriate for the negative of this parameter:

$$\pi(-\alpha_1) \sim Gamma(k_2, \theta_2)$$

The values for the intercepts of the two models are not constrained to be positive or negative, so we set Normal priors centred at zero:

$$\pi(\alpha_0) \sim N(0, \xi_1^2)$$
$$\pi(\beta_0) \sim N(0, \xi_2^2)$$

The posterior distribution for $\boldsymbol{\psi} = (\beta_0, \beta_1, \alpha_0, \alpha_1, \sigma_\epsilon^2)$, given the data, is

$$\begin{aligned}\pi(\boldsymbol{\psi}|\mathbf{I}_c, \mathbf{I}_f) &\propto \pi(\mathbf{I}_c|\boldsymbol{\psi}, \mathbf{I}_f)\pi(\boldsymbol{\psi}|\mathbf{I}_f) \\ &\propto \pi(\mathbf{I}_c|\boldsymbol{\psi}, \mathbf{I}_f)\pi(\boldsymbol{\psi})\end{aligned}$$

where $\mathbf{I}_c$ and $\mathbf{I}_f$ are the columns of $\mathbf{I}$ corresponding to the coefficient and field implausibilities respectively. Given this distribution, the distribution for $\mathcal{I}_c$ when $\mathcal{I}_f = T = \mathcal{X}^2_{l,0.995}$ can be found:

$$\begin{aligned}\pi(\mathcal{I}_c|\mathcal{I}_f = T, \mathbf{I}) &= \int \pi(\mathcal{I}_c, \boldsymbol{\psi}|T, \mathbf{I}_f, \mathbf{I}_c)d\boldsymbol{\psi} \\ &= \int \pi(\mathcal{I}_c|\boldsymbol{\psi}, T, \mathbf{I}_f, \mathbf{I}_c)\pi(\boldsymbol{\psi}|\mathbf{I}_c, \mathbf{I}_f)d\boldsymbol{\psi}\end{aligned} \tag{6.3}$$

We find this distribution via MCMC. A conservative bound for ruling out runs using the coefficient implausibility is then selected by taking the 99.5% value of this distribution, $\tilde{T}$.

6.2.4. Bound for the toy function

We now apply the model for selecting a bound to be used for history matching using the coefficient implausibility to the toy function, with the wave 1 rotated basis. We take a Latin hypercube sample with 50 members from $\mathcal{X}$, and calculate the field and coefficient implausibilities for each point to give the data $\mathbf{I}$. Figure 6.3 shows this data.

We treat the hyperparameters for the prior distributions as fixed, setting the following priors for the regression parameters:

$$\pi(\alpha_0) \sim N(0, 5^2)$$
$$\pi(-\alpha_1) \sim Gamma(10, 1)$$
$$\pi(\beta_0) \sim N(0, 10^2)$$
$$\pi(\beta_1) \sim Gamma(10, 1)$$

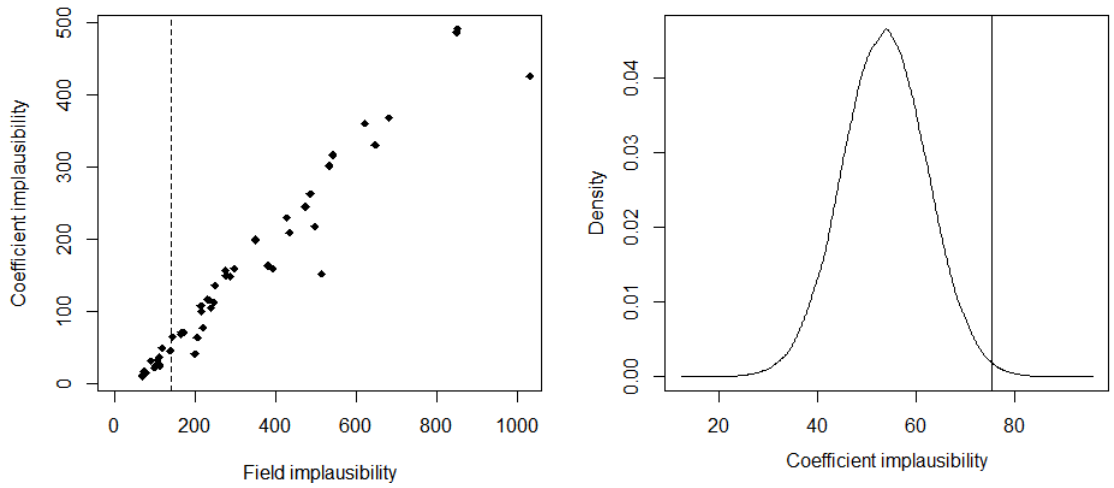

Figure 6.3. Left: The coefficient and field implausibilities for the 50 sampled parameter values, for the wave 1 rotated basis emulators. The dotted line shows the history matching bound for the field implausibility. Right: the posterior density for $\mathcal{I}_c|\mathcal{I}_f = T$, with the vertical line indicating the 99.5% value of this distribution.

This completes the specification of the model in (6.2), and the predictive distribution for $\mathcal{I}_c|\mathcal{I}_f = T$ in (6.3) can be sampled from.

The resulting distribution for the value of the bound for the coefficient implausibility is given in the right panel of Figure 6.3. The 99.5% value of this distribution is equal to 75.56 (as shown by the vertical line in this plot), and hence this is the bound used to define NROY space with $\mathcal{I}_c$:

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_c(\mathbf{x}) < \tilde{T} = 75.56\}$$

The size of this NROY space is 42.27% of the original parameter space $\mathcal{X}$. This is a larger NROY space than the one defined by the field implausibility for the wave 1 rotated basis in Section 5.5.2, which contained 31.49% of $\mathcal{X}$. However, none of the runs that lie in the true NROY space have been ruled out, compared to 16% of these when the chi-squared bound was used for the coefficient implausibility. The modelled bound has given a more conservative, but more accurate, result, than that given by history matching using the chi-squared bound for the coefficient implausibility.

The same model is fitted for the wave 2 implausibilities. This time, a smaller sample (20 runs) of the field implausibility is used to fit the model. The values of the implausibilities at these 20 points are shown in Figure 6.4, again exhibiting a positive linear relationship. Fitting the model leads to the posterior predictive distribution at T, on the right of Figure 6.4, with the bound given by the 99.5% value of the distribution shown by the vertical

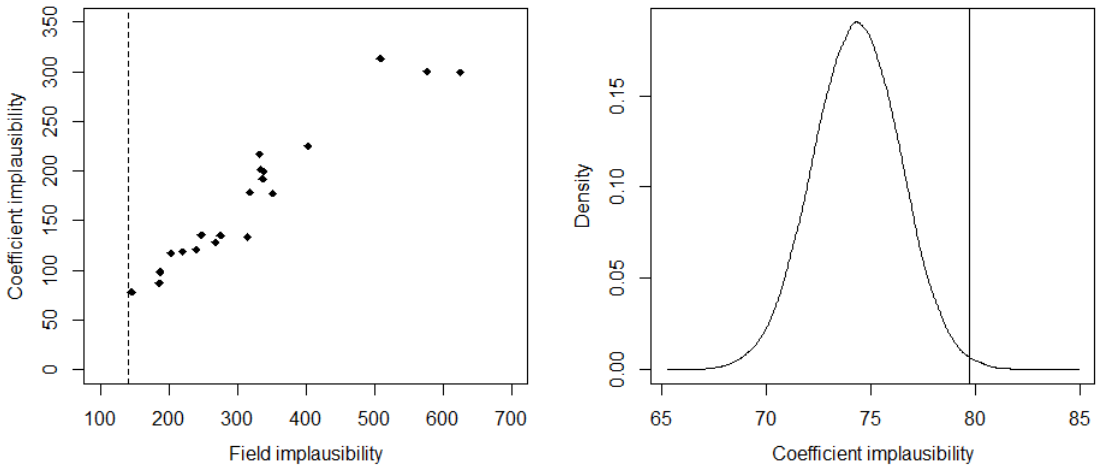

Figure 6.4. Left: The coefficient and field implausibilities for 20 sampled parameter values, for the wave 2 rotated basis emulators. The dotted line shows the history matching bound for the field implausibility. Right: the posterior density for $\mathcal{I}_c|\mathcal{I}_f = T$, with the vertical line indicating the 99.5% value of this distribution.

line. This bound is equal to 79.70. This is higher than the value chosen 'by eye' earlier (70), perhaps due to the smaller sample size being used to fit this model, and the fact that none of the observed field implausibilities are below the chi-squared bound for the field, so the model is extrapolating.

Using this bound to history match using the coefficients, over 75% of the wave 1 NROY space is ruled out, leaving the wave 2 NROY space as 7.51% of the full parameter space. When history matching using the field implausibility, NROY space had been reduced to 3% of $\mathcal{X}$ after two waves. Once again, this bound has resulted in fewer settings of $\mathbf{x}$ being ruled out, but greater accuracy has been achieved, as at wave 1: none of the true NROY space has been ruled out by this bound.

Our model allows the coefficient implausibility to be related to the more accurate (in terms of being a better representation of whether a field is 'close' to $\mathbf{z}$) field implausibility, enabling history matching to be performed for large output fields without the need for vast numbers of high-dimensional matrix inversions. This enables history matching to be performed for the CanAM4 model.

6.2.5. Efficiently calculating the field implausibility

It is in fact possible to rewrite the field implausibility so that only an inversion of a $q \times q$ matrix is required for each new $\mathbf{x}$ (a similar simplification is demonstrated for the

calibration likelihood in Higdon et al. (2008a)).

The Woodbury formula (Woodbury, 1950, Higham, 2002) states that

$$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1}+\mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1} \tag{6.4}$$

where $\mathbf{A}$ is an $l \times l$ matrix, $\mathbf{C}$ is a $q \times q$ matrix, $\mathbf{U}$ is an $l \times q$ matrix, and $\mathbf{V}$ is a $q \times l$ matrix. By setting $\mathbf{A} = \mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$, $\mathbf{C} = \mathrm{Var}[\mathbf{c}(\mathbf{x})]$, $\mathbf{U} = \boldsymbol{\Gamma}_q$ and $\mathbf{V} = \boldsymbol{\Gamma}_q^T$, the matrix inversion in the field implausibility can be rewritten as

$$(\mathrm{Var}[f(\mathbf{x})]+\mathbf{W})^{-1} = \mathbf{W}^{-1} - \mathbf{W}^{-1}\boldsymbol{\Gamma}_q(\mathrm{Var}[\mathbf{c}(\mathbf{x})]^{-1}+\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1}\boldsymbol{\Gamma}_q)^{-1}\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1} \tag{6.5}$$

The observation error and discrepancy variance matrices are fixed, so that the expensive $l \times l$ inversion of their sum only needs to be calculated once. For each new evaluation of $\mathcal{I}_f(\mathbf{x})$, the only inverses that are required are $\mathrm{Var}[\mathbf{c}(\mathbf{x})]^{-1}$ and $(\mathrm{Var}[\mathbf{c}(\mathbf{x})]^{-1}+\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1}\boldsymbol{\Gamma}_q)^{-1}$, both of which are $q \times q$ matrices.

Therefore, the field implausibility can be written as (with $\mathbf{W} = \boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ for clarity):

$$\begin{aligned}
\mathcal{I}_f(\mathbf{x}) &= (\mathbf{z}-\mathrm{E}[f(\mathbf{x})])^T(\mathrm{Var}[f(\mathbf{x})]+\mathbf{W})^{-1}(\mathbf{z}-\mathrm{E}[f(\mathbf{x})]) \\
&= (\mathbf{z}-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})])^T(\boldsymbol{\Gamma}_q\mathrm{Var}[\mathbf{c}(\mathbf{x})]\boldsymbol{\Gamma}_q^T+\mathbf{W})^{-1}(\mathbf{z}-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})]) \\
&= (\mathbf{z}-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})])^T(\mathbf{W}^{-1}-\mathbf{W}^{-1}\boldsymbol{\Gamma}_q(\mathrm{Var}[\mathbf{c}(\mathbf{x})]^{-1}+\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1}\boldsymbol{\Gamma}_q)^{-1}\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1})(\mathbf{z}-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})])
\end{aligned}$$

This can also be written in terms of the reconstruction error:

$$\begin{aligned}
\mathcal{I}_f(\mathbf{x}) = \mathcal{R}_\mathbf{W}(\boldsymbol{\Gamma}_q,\mathbf{z}) &+ (\mathbf{r}(\mathbf{z})-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})])^T\mathbf{W}^{-1}(\mathbf{r}(\mathbf{z})-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})]) - \\
&(\mathbf{z}-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})])^T(\mathbf{W}^{-1}\boldsymbol{\Gamma}_q(\mathrm{Var}[\mathbf{c}(\mathbf{x})]^{-1}+\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1}\boldsymbol{\Gamma}_q)^{-1}\boldsymbol{\Gamma}_q^T\mathbf{W}^{-1})(\mathbf{z}-\boldsymbol{\Gamma}_q\mathrm{E}[\mathbf{c}(\mathbf{x})])
\end{aligned}$$

For $l = 8192$, rather than the 78 seconds it takes to calculate the field implausibility for a single $\mathbf{x}$, the formulation with only $q \times q$ inversions requires just over a second for each evaluation (given that $\mathbf{W}^{-1}$ has already been calculated). This makes the field implausibility viable for larger values of l, although for our example, with $l = 8192$, the hierarchical model is still likely to be required so that the implausibility at millions of parameter settings, as needed in history matching, can be found efficiently enough.

6.3. History matching a climate model

By combining the basis rotation methodology of Chapter 5 and the model relating the field implausibility with the coefficient implausibility, history matching for CanAM4 is now more computationally achievable, and can be performed.

There are a number of different output fields that could be used to history match CanAM4. In order to simplify the problem, a small subset of the outputs will be considered, with basis rotations applied and NROY spaces defined. The three fields that were shown in Section 4.5.2 are chosen for history matching: the air temperature (TA), the top of the atmosphere balance (RTMT), and the cloud overlap (CLTO). TA is given over a 64×37 latitude-pressure grid, averaged over longitude, whereas RTMT and CLTO are over the 128×64 longitude-latitude grid. As in Section 4.5.2, only June, July and August (JJA) are used, with the average taken over five years of model output, so that the problem is spatial, rather than spatio-temporal. These three fields are selected as accurate emulators can be built for them.

The anomaly fields for the standard runs of each of these outputs were given in Figures 4.13, 4.14 and 4.15. The goal of history matching is to identify parameter settings that are more similar to the observed fields than this model run. It is possible to perform 50 additional runs of CanAM4, the choice of which should be guided by history matching.

6.3.1. Basis rotation and emulation

First, we select the basis that will be used for emulating each of the fields, via the optimal rotation algorithm in Section 5.4. In order to select the basis, we first must specify $\mathbf{W}$ for each field. As we have no information about the structure of the observation error or discrepancy variance, each grid box will be treated equally at the rotation step. We set the observation error as in Section 4.5.3, treating white coloured areas of the anomaly plot as acceptable. For this application, we use the lower estimate, setting the white coloured anomalies as equal to 3 standard deviations, i.e.

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{e}}^{CLTO} = \boldsymbol{\Sigma}_{\mathbf{e}}^{RTMT} &= \frac{25}{9}\boldsymbol{\mathcal{I}}_{8192} \\ \boldsymbol{\Sigma}_{\mathbf{e}}^{TA} &= \frac{4}{9}\boldsymbol{\mathcal{I}}_{2368} \end{aligned} \tag{6.6}$$

so that we can investigate whether it is possible to find anomaly fields that are coloured all white. To perform the rotation, we set $\mathbf{W} = \mathbf{\Sigma_e}$ for each field.

Based on the reconstructions and VarMSE plots for the SVD basis for these fields (Figures 4.16, 4.17, and 4.18), the large reconstruction errors suggest that history matching will rule out $\mathbf{z}$ if no discrepancy is specified. This problem will be addressed in Section 6.3.2. For this application, we ignore the first step from the rotation algorithm (stop if $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) > T$), and find an optimal basis, given that the error in each grid box is treated equally, terminating the algorithm when the new truncated basis has an error suitably close to the theoretical minimum given by the SVD basis, $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}, \mathbf{z})$.

CLTO

The SVD basis for CLTO is rotated first. The maximum time for the optimiser to run was set at 300 seconds for each iteration, as experience has suggested that this is sufficiently long enough for the optimiser to converge to reasonable results. By performing the rotation for a variety of different variance thresholds, the final rotation required each of the first three vectors to explain at least 10% of the ensemble variability:

$$v_1 = v_2 = v_3 = 0.1$$

as this gave the best resulting basis, in terms of reducing the reconstruction error while allowing emulation on each basis vector of the truncated basis. From experience, three or fewer iterations are often sufficient to find a basis with reconstruction error similar to the minimum given by the full SVD basis.

The resulting basis is depicted in the VarMSE plot in Figure 6.5, with a comparison to the SVD basis given by the dotted red line. The first 12 basis vectors of the rotated basis are needed to explain 90% of the ensemble variability, i.e. $\mathcal{V}(\mathbf{\Gamma}^*_{12}, \mathbf{F}_{\boldsymbol{\mu}}) > 0.9$, where $\mathbf{\Gamma}^* = \mathbf{\Gamma\Lambda}$ is the rotated basis. A further 19 explain the next 5% of variability ($\mathcal{V}(\mathbf{\Gamma}^*_{31}, \mathbf{F}_{\boldsymbol{\mu}}) > 0.95$) so the basis is truncated after 12, so that emulation does not become too time-consuming. This plot shows that the (scaled) reconstruction error for the truncated rotated basis is $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}^*_{12}, \mathbf{z})/8192 = 42.4$, compared to 60.4 for the truncated SVD basis. The full SVD basis gives reconstructions with an error of 37.5, however in this case it was not possible to

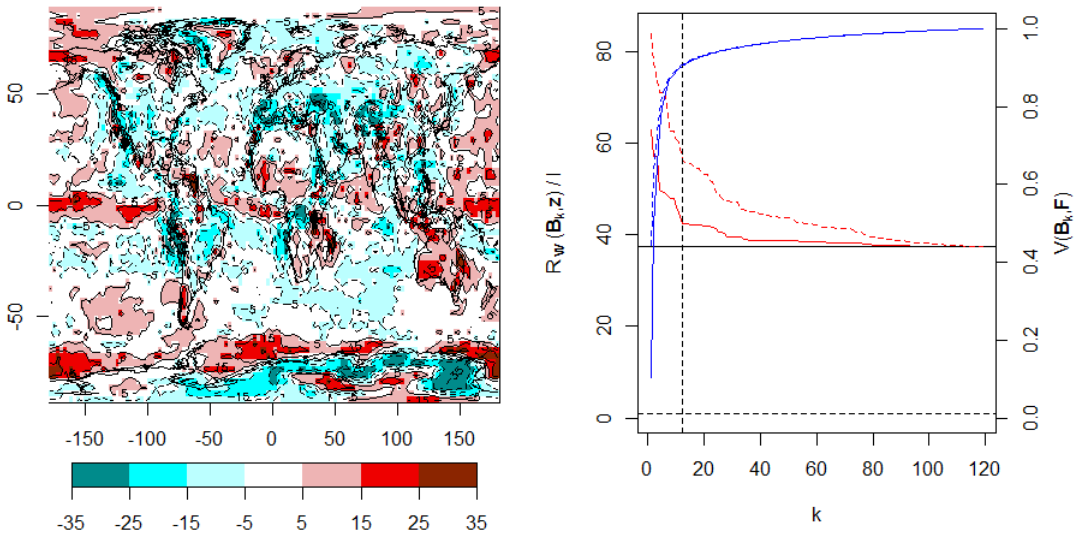

Figure 6.5. The anomaly between the reconstruction of the CLTO observations and the observations, with the truncated rotated basis used for projection. The VarMSE plot compares the rotated basis (solid lines) with the SVD basis (dotted lines), with $\mathbf{W} = \boldsymbol{\Sigma}_{\mathbf{e}}^{CLTO} = \frac{25}{9}\boldsymbol{\mathcal{I}}_{8192}$.

incorporate all of this signal into the rotated basis, and still retain emulatability. Despite this, the rotated basis is a clear improvement over the SVD basis.

The anomaly for the reconstruction of $\mathbf{z}$ using the truncated rotated basis is also given in Figure 6.5. There are large anomalies observed in this plot, shown by the darker colours, unsurprisingly as the VarMSE plot indicates that $\mathbf{z}$ would be ruled out by this basis. However, regions of this reconstruction exhibit improvements over using the SVD basis (Figure 4.17). For example, the positive anomaly near to the Equator in the Pacific has been reduced, as has the anomaly over Australia.

Emulators are built for the first 12 rotated basis vectors, with validation plots provided in Appendix B.4.

RTMT

For RTMT, we again set $v_1 = v_2 = v_3 = 0.1$, as this gave the best combination of minimising $\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}_q^*, \mathbf{z})$ and allowing emulation. Figure 6.6 shows that the truncated rotated basis gives a reconstruction error close to that found with the full SVD basis, and provides an improvement over the truncated SVD basis. For the rotated RTMT basis, we have $\mathcal{V}(\boldsymbol{\Gamma}_{11}^*, \mathbf{F}_{\boldsymbol{\mu}}) > 0.9$ and $\mathcal{V}(\boldsymbol{\Gamma}_{23}^*, \mathbf{F}_{\boldsymbol{\mu}}) > 0.95$. Therefore, the rotated basis is truncated after $q = 11$ vectors, to avoid the need to emulate an extra 12 sets of coefficients

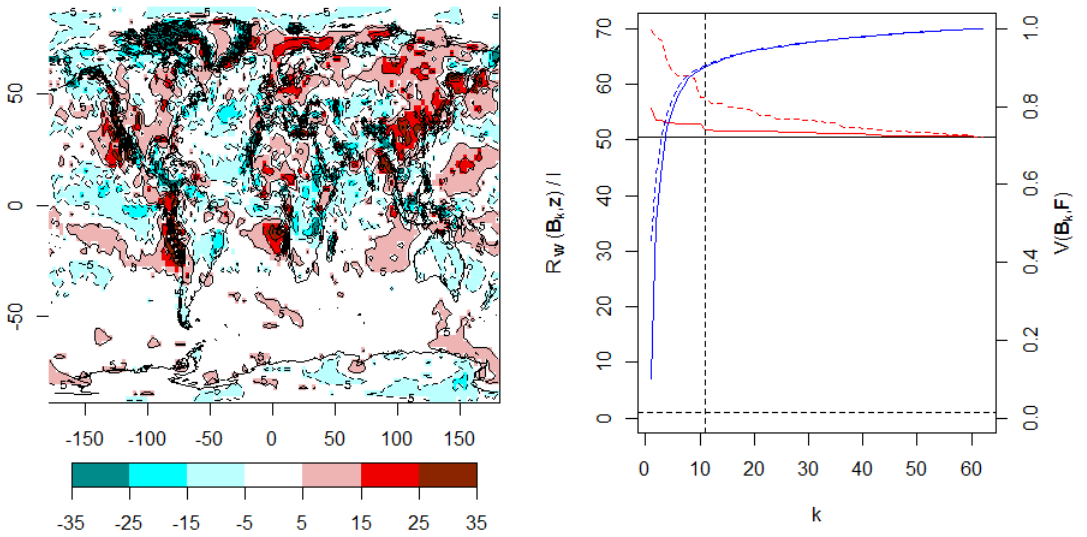

Figure 6.6. The anomaly between the reconstruction of the RTMT observations and the observations, with the truncated rotated basis used for projection. The VarMSE plot compares the rotated basis (solid lines) with the SVD basis (dotted lines), with $\mathbf{W} = \mathbf{\Sigma}_{\mathbf{e}}^{RTMT} = \frac{25}{9}\mathcal{I}_{8192}$.

containing little ensemble signal. The (scaled) reconstruction error with this basis is $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_{11}^{*}, \mathbf{z})/8192 = 51.9$, compared to 57.7 for the truncated SVD basis (the full SVD basis gives 50.6).

Again, without discrepancy properly included, this reconstruction of the observations would be ruled out, as the history matching bound lies a significant distance below the basis reconstruction error. The anomaly plot for the reconstruction with the truncated rotated basis in Figure 6.6 shows why. There are a number of regions with large positive anomalies, especially over East Asia and the west coast of the Americas. These are reduced slightly from when the reconstruction with the truncated SVD basis was considered (Figure 4.18).

Diagnostics for the 11 rotated basis vector emulators are shown in Appendix B.4.

TA

For TA, the rotation algorithm has perhaps performed the best. With the minimum variance for the first basis vector set as $v_1 = 0.35$, and by combining this with its residual basis, we find the basis summarised by the VarMSE plot in Figure 6.7. The reconstruction error for the rotated basis reaches the minimum given by the full SVD basis (up to two decimal places) with the inclusion of six basis vectors, whereas the SVD basis slowly

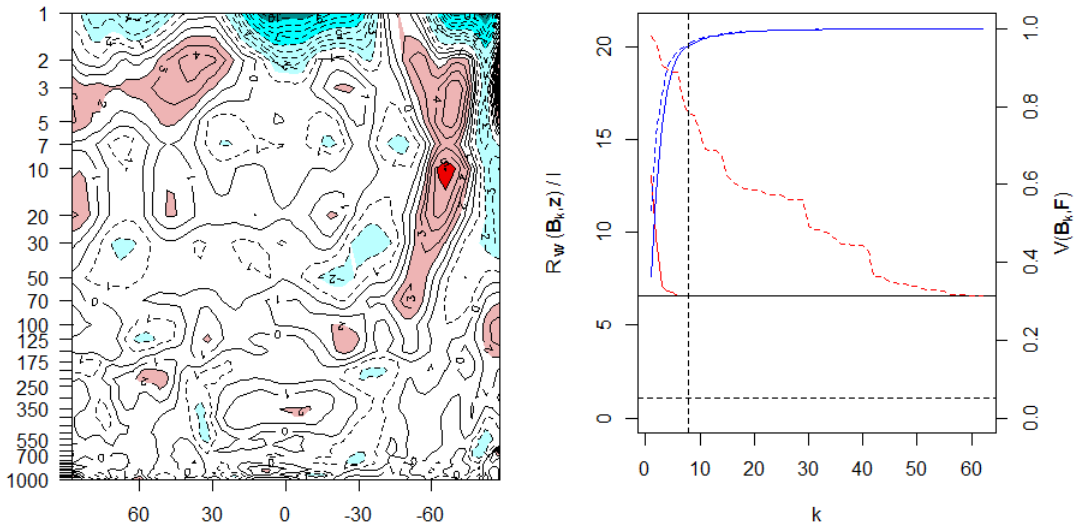

Figure 6.7. The anomaly between the reconstruction of the TA observations and the observations, with the truncated rotated basis used for projection. The VarMSE plot compares the rotated basis (solid lines) with the SVD basis (dotted lines), with with $\boldsymbol{\Sigma}_{\mathbf{e}}^{TA} = \frac{4}{9}\boldsymbol{\mathcal{I}}_{2368}$.

converges to this value. The rotated basis requires $q = 8$ so that $\mathcal{V}(\boldsymbol{\Gamma}_q^*, \mathbf{F}_{\boldsymbol{\mu}}) > 0.95$, and hence the basis is truncated here, with a (scaled) reconstruction error of $\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}_8^*, \mathbf{z})/8192 = 6.6$. The truncated SVD basis has an error of 16.9.

Comparing the reconstructed field given by the truncated rotated basis (Figure 6.7) to that given by the truncated SVD basis (Figure 4.16) highlights this improvement. The large warm anomaly over the South Pole has been greatly reduced here, while smaller gains have also been achieved around the Equator. Emulators are fitted to the first eight rotated basis vectors (Appendix B.4).

6.3.2. Specifying the discrepancy

It has not been possible to obtain a specification for the discrepancy or observation error variances from the climate modellers. The observation error variance is likely to be small in comparison to the discrepancy, and in lieu of any prior information about either, $\boldsymbol{\Sigma}_{\mathbf{e}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ will be combined in this problem, and referred to hereafter as the discrepancy, $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$.

The reconstruction errors shown previously suggest that the basis representations are not capable of finding model runs that replicate reality. However, it is not clear whether this is due to a structural discrepancy between the model and the real world, or whether this

can be attributed to exploring a small fraction of the input space, and the ensemble model runs by chance have not given fields similar to the observations. Moreover, the modellers are unwilling to rule out certain errors as 'structural', preferring to see if their model can overcome some of its perceived deficiencies via tuning.

If history matching were to be performed without a discrepancy term, it is clear that all of parameter space would be ruled out (unless emulator variance is large), as even the true coefficients for $\mathbf{z}$ on each rotated basis give reconstructed fields that would be ruled out. To ensure that the NROY spaces defined have the possibility of being non-empty, a specification for the discrepancy based on the reconstruction error given by the truncated rotated basis is used, with the aim that NROY space will contain runs with some signal in important directions. When a second wave ensemble is designed based on this NROY space, and run with CanAM4, runs containing more of this signal may then be found.

The discrepancy variance is set as

$$\mathbf{\Sigma}_{\boldsymbol{\eta}} = \frac{\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z})}{b}\mathbf{W} \tag{6.7}$$

where $\mathbf{\Gamma}_q^* = \mathbf{\Gamma}\mathbf{\Lambda}_q$ is the truncated rotated basis, and

$$b = \mathcal{X}^2_{l,j}$$

where $j < 0.995$. This takes the value of the reconstruction error for $\mathbf{z}$ given by the truncated rotated basis, using the weight $\mathbf{W}$ involved in the rotation algorithm, and divides it by a value from the chi-squared distribution with l degrees of freedom, and multiplies $\mathbf{W}$ by this constant. Due to the fact that $\mathbf{W}$ is a diagonal matrix in this case, this has the effect of dividing the reconstruction error by this constant, and lowering the red lines on the VarMSE plots. Setting $j = 0.995$ forces the reconstruction error to intersect the horizontal dotted line, representing the history matching bound, where the basis is truncated (q). By selecting a lower value of this distribution, so that the multiple of $\mathbf{W}$ is higher, we have

$$\mathcal{R}_{\mathbf{\Sigma}_{\boldsymbol{\eta}}}(\mathbf{\Gamma}_q^*, \mathbf{z}) < T$$

i.e. the reconstruction of the observations (equivalently, the implausibility for $\mathbf{z}$ here) is not ruled out by the truncated rotated basis, with the discrepancy set as in (6.7).

Output	l	$\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z})/l$	j	b/l	$\mathbf{\Sigma}_{\boldsymbol{\eta}}$
CLTO	8192	42.38	0.68	1.0072	$116.86\mathcal{I}_l$
RTMT	8192	51.89	0.95	1.0258	$140.52\mathcal{I}_l$
TA	2368	6.60	0.68	1.0134	$2.89\mathcal{I}_l$

Table 6.2. A table giving the discrepancy variance for each of the fields, with the values used in their calculation, with $\mathbf{W} = \mathbf{\Sigma}_{\mathbf{e}}$ from (6.6).

This definition reduces the reconstruction error so that the observations should no longer be ruled out. A lower value of the chi-squared distribution is used because if $j = 0.995$, the observations would be ruled out: NROY space is defined as $\mathbf{x}$ with implausibility strictly less than the bound, so that in a perfect emulation scenario, the implausibility for $\mathbf{z}$ would be ruled out.

If we wished to keep the previously defined observation error variances $\mathbf{\Sigma}_{\mathbf{e}}$ (from (6.6)) separate, these could be subtracted from the definition of $\mathbf{\Sigma}_{\boldsymbol{\eta}}$ given in (6.7). As we are assuming the same structure for these two variances in this problem, it does not matter that they are simplified and treated as a single matrix.

Applying this method to define the discrepancy for the three output fields gives the discrepancies in Table 6.2. For example, the reconstruction error (scaled by the field size for consistency with the VarMSE plots) for CLTO is $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z})/8192 = 42.38$. Taking the 0.68 value of the chi-squared distribution with 8192 degrees of freedom then implies that the multiplier required for the identity matrix is 116.86. TA is specified over the pressure-latitude grid, and hence has a lower dimension, as reflected in this table.

Initially, j was set equal to 0.95 for each of the outputs. However, history matching using this variance (and the bound for the coefficient implausibility that this then implied) ruled out all of parameter space for CLTO and TA. Therefore, the value of j was reduced to 0.68, and hence the discrepancy variance increased, reducing implausibilities and giving non-empty NROY spaces for each field individually.

With these discrepancy variances, the VarMSE plots can be revisited, with weight $\mathbf{W} = \mathbf{\Sigma}_{\boldsymbol{\eta}}$ for each field. These are shown for each output in Figure 6.8. For each of these, the reconstruction error, $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z})$, now lies below the history matching bound, due to the construction of the discrepancy, and hence the representation of each output's observations on the rotated basis will not be ruled out, in theory. Essentially, increasing the discrepancy variance uniformly for each grid box has the effect of 'shifting down' the reconstruction

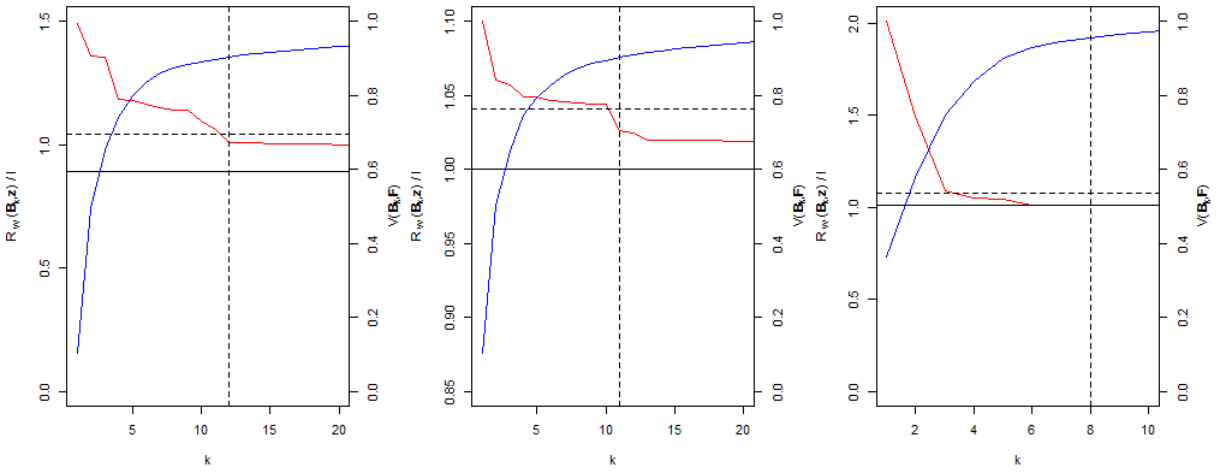

Figure 6.8. The VarMSE plots for CLTO, RTMT and TA respectively, with $\mathbf{W} = \boldsymbol{\Sigma}_{\boldsymbol{\eta}}$.

error, and the minimum given by the full SVD basis, so that it now lies below the bound for the field implausibility (which remains fixed). Now that the observations would not be ruled out if the correct coefficients were given by a choice of $\mathbf{x}$, history matching is possible.

Using this method for defining the discrepancy, it could be argued that the SVD basis may as well be used, if the discrepancy is going to be arbitrarily increased so that $\mathbf{z}$ will not be ruled out. However, this is only used here due to the lack of any prior knowledge, and some discrepancy must be set in order to illustrate history matching for CanAM4. Additionally, as little discrepancy as possible is desired, and as the truncated SVD basis has always been found to give worse reconstructions than the rotated basis, the discrepancy variance would be larger if the SVD basis was used. Finally, calibration and history matching were found to be less accurate when using the SVD basis for the toy function. The rotated basis allows emulators to be built for more important output directions than the SVD basis, so that regardless of the value of the discrepancy, the rotated basis ought to have a greater chance of identifying regions of parameter space exhibiting the most important patterns.

6.3.3. Inferring the coefficient bounds

Using the discrepancy variances set above, and the emulators for the coefficients, we now calculate implausibilities. It is not possible to calculate the field implausibility as often as required, because this takes more than a minute for any parameter setting $\mathbf{x}$. Therefore, prior to history matching, we use the method from Section 6.2.3 to set the bound that

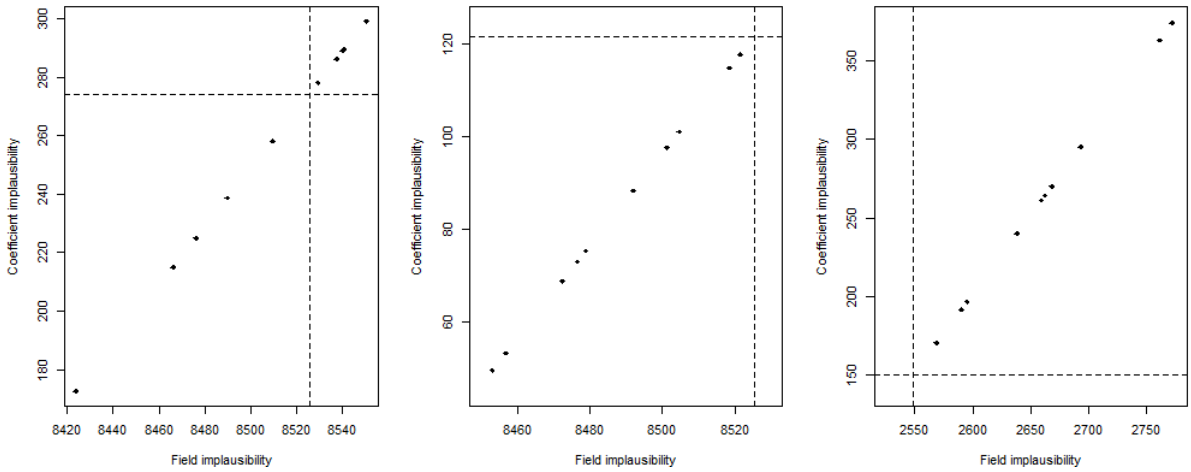

Figure 6.9. Plots comparing the field implausibility and coefficient implausibility for CLTO, RTMT and TA, with the discrepancy as specified in Table 6.2. The vertical line indicates the bound used to history match with the field implausibility, and the horizontal line gives the predicted bound for the coefficients.

will be used to rule out parameter settings using the coefficient implausibility.

In fact, evaluating the field and coefficient implausibilities for each output for a few points immediately suggests that there is a perfect linear relationship between the two (Figure 6.9). While there was a strong correlation between the implausibilities for the toy function, they were not perfectly correlated. The observation error and discrepancy variances matrices were not diagonal for the toy function, but setting them as multiples of the identity matrix still does not give a perfect correlation between the two implausibilities. This is perhaps a function of the size of the field, l, being large in comparison to the number of basis vectors, q.

The benefit of the perfect linear relationship for the climate model implausibilities, given the current specifications of the discrepancy variances, is that the bound for the field implausibility can be used to perfectly infer the bound that should be used for the coefficient implausibility. The bound for history matching on the field, and the coefficient implausibility bound that this implies, is shown in Figure 6.9, and Table 6.3 gives these new bounds for history matching with the coefficient implausibilities, compared to those given by the chi-squared distribution. From this, it is clear that the new bound is much more conservative, with larger implausibilities not ruled out. This is a benefit: the chi-squared bounds here would rule out parts of space that would not be ruled out by the field implausibility, and hence may have ruled out runs that are in fact similar to the observations on the field (as demonstrated for the toy function).

Output	q	T	**New bound**
CLTO	12	28.30	274.12
RTMT	11	26.76	121.78
TA	8	21.95	150.62

Table 6.3. A table giving the standard bound that would be used based on the chi-squared distribution, and the new bound given by the implausibility model.

6.3.4. NROY space

Now that appropriate bounds for the coefficient implausibility have been found for each of the three fields, we perform history matching using these bounds to define NROY space. An NROY space can be defined individually for each of CLTO, RTMT and TA. However, as the aim is to run a new ensemble on CanAM4, it is not useful to spend resources on parameter settings that lie in some, but not all, of the individual NROY spaces, as it is expected that the output here will not be consistent with $\mathbf{z}$, or even an improvement on the current standard output. Therefore, NROY space will be defined as

$$\mathcal{X}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}_{c,CLTO}(\mathbf{x}) < 274.12,\ \mathcal{I}_{c,RTMT}(\mathbf{x}) < 121.78,\ \mathcal{I}_{c,TA}(\mathbf{x}) < 150.62\}$$

i.e. to be in NROY space, $\mathbf{x}$ must be not implausible for each of CLTO, RTMT and TA.

We take a 500,000 member Latin hypercube sample of the 13-dimensional input parameter space, and calculate the coefficient implausibility for each $\mathbf{x}$ for each output, in order to estimate the size of NROY space. Taken individually, 45.5% of parameter settings are not implausible for CLTO, 83.5% are not ruled out for RTMT, and only 2% are not ruled out for TA. Combining the implausibilities to give $\mathcal{X}_{NROY}$, it is found that 0.9% of $\mathcal{X}$ is in NROY space. A representation of this NROY space is given in Figure C.4.

The amount of space that is ruled out is directly related to the choice of discrepancy, so these results are somewhat arbitrary. However, the main goal of this calibration exercise is to highlight regions of input space that give output in the direction of the observations. The lowest implausibilities have the greatest chance of this, and this NROY space should therefore contain the runs most likely to give output consistent with $\mathbf{z}$.

6.3.5. Ensemble design

We now design a 50 member ensemble for CanAM4. The 50 runs should be sampled from the NROY space defined above, but there are a number of key criteria desired for the design.

Firstly, the design should be reasonably space-filling across the input parameters of NROY space. This ensures that an accurate representation of NROY space is obtained, and assists emulation within NROY space at the next wave. Additionally, the selected runs should contain a range of implausibilities (while still not being ruled out, as in Section 5.6.1), for each of the three outputs. For example, parameter settings that minimise (or give one of the lowest) implausibilities for each of the fields, given that they lie in NROY space, should be considered.

The emulated fields should highlight improvements over the standard run, or contain unobserved patterns. As the design is only of 50 runs, and there are only three fields, it is possible to check by hand what it is expected the model output will look like, according to the emulators. Runs that are expected to fix various biases from the standard run are chosen. Related to this criteria, the emulated fields should not be homogeneous: there should be a variety of expected outputs, if there is variability within NROY space, and not just 50 runs all fixing one small aspect of the observed anomalies.

First, a larger sample of runs from NROY space is taken, so that parameter settings with lower implausibilities may be found. Given this sample, the design is constructed via an iterative process. An initial random sample of 50 runs is taken, and the emulated fields are calculated. By studying this ensemble of expected outputs, runs that contain patterns deemed to be important, or at least different to the standard run and original ensemble, are kept. Given these selected runs, new samples are taken to achieve a sample of size 50 again. New runs are then fixed if they exhibit different patterns to those already fixed, or represent a currently unsampled region of NROY space, or of implausibility space. This process continues until 50 satisfactory runs are found, that suitably represent input and implausibility space, and that are predicted to fix various biases from the standard model output. A plot showing the parameters for the new design is given in Figure C.5.

6.4. The new wave

The 50 member design for CanAM4, described in the previous section, is now run on the CCCMA supercomputer. One of the model runs failed, hence the new ensemble has $n = 49$. There are a number of questions that this allows us to answer. Firstly, whether any of the new runs are superior (according to some metric) to the standard run can be answered. Which biases might be fixed through altering the parameters alone, and what effect this has on other regions of the output, are also important questions.

6.4.1. Ensemble summary

To assess the anomalies of the runs in the new ensemble, the root-mean-square error (RMSE) between the model output and the observations can be calculated. The grid boxes in the model output correspond to different areas according to their longitude-latitude location, and this is reflected in this calculation by weighting the differences by area.

Using this measure, there are runs in the new ensemble that have a lower weighted RMSE for each of CLTO, RTMT and TA individually. However, there are none that simultaneously improve upon the standard run across each of the three fields. Figures 6.10, 6.11 and 6.12 show the runs in the new ensemble that minimise the weighted RMSE for each of the three fields, alongside the anomalies for the standard run for comparison.

For CLTO, run 039 improves the weighted RMSE the most, and from the comparison with the standard run, some of these improvements can be identified. Positive biases have been reduced, particularly near to the Equator in the Pacific and North Atlantic oceans. The bias around Australia, and the negative bias observed previously in the Indian Ocean, have been reduced to some extent as well. There are also more white-coloured grid boxes in the new run, particularly in the Southern Ocean. Over the continents, where there tends to be negative anomalies, it is less clear whether any improvement has been made by this run: the patterns are similar, and of a comparable magnitude, as before. Overall, this run appears to at least be an incremental improvement over the standard run.

In Figure 6.11 (RTMT), the changes between the standard run and the new best run

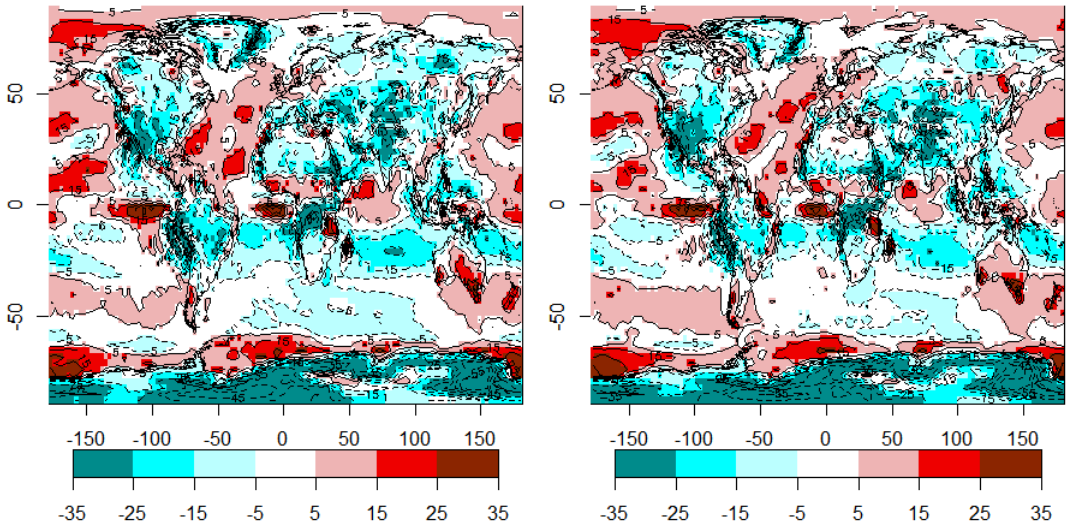

Figure 6.10. The CLTO anomaly for the standard run (left), and for run 039 of the new ensemble.

are much more obvious. The large area of red in the western Pacific Ocean has been mostly removed in run 032, with only small positive biases remaining here. A possible consequence of reducing the anomaly here is that there is now a larger positive bias to the north, over Russia. Whether this is preferable is a question for climate scientists, and may be dependent on what the model is to be used for.

Other changes between the standard run and run 032 include the introduction of a small positive bias in the Southern Ocean, possibly in response to the negative bias to the north of this being reduced. The positive bias previously found near to the Caribbean has been mostly removed, although this may have contributed to an increased anomaly over eastern Canada. Generally, this run has removed some large biases found in the standard run, particularly around the Equator, but doing so may have introduced larger biases towards the poles.

For TA (Figure 6.12), it is not obvious that the new run gives an improvement over the standard run. The magnitude and shape of the warm bias at the South Pole is extremely similar, and all of the other patterns are generally the same as for the standard run. In each of the runs in the new ensemble, the large anomaly at the South Pole has been observed, giving evidence that this error may be structural.

Overall, large improvements have not been found. However, fixing some biases observed in the standard run, and observing the impact that this has on other regions of the output, is useful, and will lead to more accurate emulators, and more accurate history matching,

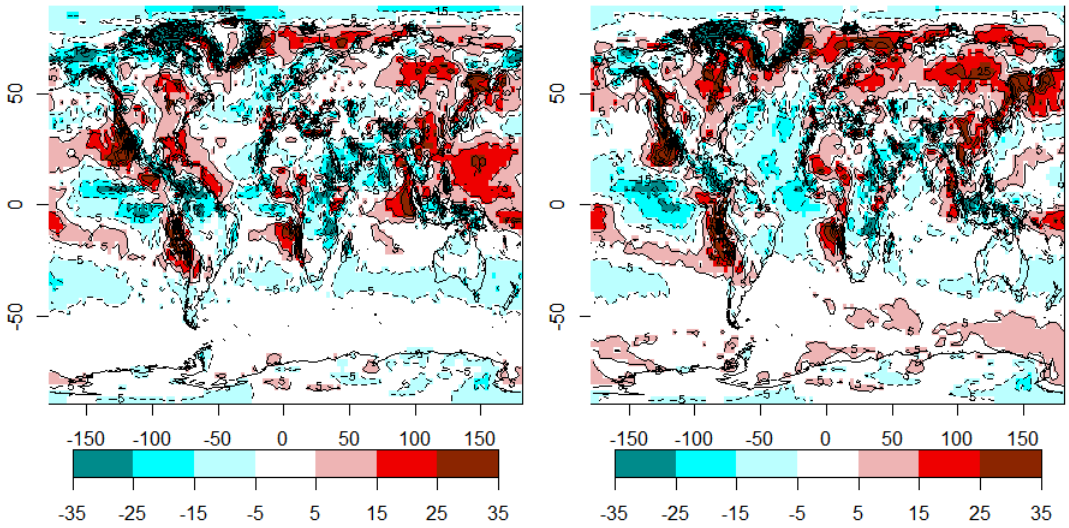

Figure 6.11. The RTMT anomaly for the standard run (left), and for run 032 of the new ensemble.

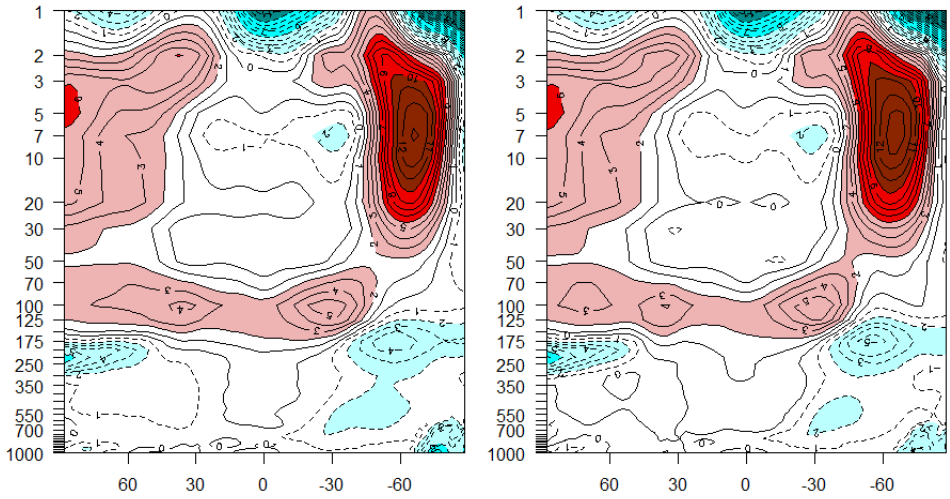

Figure 6.12. The TA anomaly for the standard run (left), and for run 005 of the new ensemble.

when these new runs are used. Previously, the impact of reducing the bias in some regions was unknown from the ensemble, and hence the wave 1 emulators may have struggled to suggest the downsides of such potential improvements. By having these new known correlations included in the basis, modelling should improve. As demonstrated for the much simpler toy function in Chapter 4, history matching should be an iterative process; for a model as complicated as CanAM4, finding far superior model runs after only one wave was perhaps unlikely.

6.4.2. Rotating in NROY space

Using the new ensemble of runs, we perform a second wave of history matching. The methodology is mostly similar to that used at wave 1 (Section 6.3), although a couple of extensions are required.

There is an additional question to be answered prior to rotating the SVD basis, that was not evident when performing history matching at later waves for the toy function. In this application, the new ensemble is smaller than the original ensemble. If the SVD basis was calculated solely using the new ensemble, and rotated, it is possible that the reconstructions of the observations will be worse than at the initial wave. This is because the smaller ensemble size further restricts the number of directions that can be defined. Although the truncation of the basis will lead to a similarly-sized basis that is used in practice, the rotation is restricted by the SVD basis, and hence may not be as accurate as at wave 1.

This problem can be illustrated with the ensembles for CanAM4. Figure 6.13 shows the reconstruction error, $\mathcal{R}_{\mathbf{W}}(\cdot, \mathbf{z})$, for three different SVD bases for RTMT: the basis calculated from the wave 1 ensemble only, the basis calculated from the wave 2 ensemble only, and the basis when all of the wave 1 and wave 2 runs are used. This highlights the issue described above: the full wave 2 SVD basis gives a larger reconstruction error than the wave 1 basis, and in fact the reconstruction error for the full wave 2 basis is greater than the history matching bound. For the full SVD basis at wave 1, the (scaled) reconstruction error is 1, compared to 1.09 for the full wave 2 SVD basis. Combining the two ensembles gives an SVD basis with an error of 0.94 (all with $\mathbf{W} = \boldsymbol{\Sigma}_{\boldsymbol{\eta}} = 140.52\boldsymbol{\mathcal{I}}_{8192}$).

There are a few potential reasons for this. One is that the new ensemble is more dissimilar from the observations than previously, so that patterns resembling $\mathbf{z}$ cannot be extracted from the ensemble. There could be runs in the wave 1 ensemble that, while extremely inconsistent with the observations, contain some signal in the direction of $\mathbf{z}$, or that allows a basis vector to be defined in this direction. For example, there could be a wave 1 model run containing a pattern that appears in one part of the output for $\mathbf{z}$, with the rest of the output extremely dissimilar to $\mathbf{z}$, so that this run was ruled out. Having this pattern present in the definition of the basis would be beneficial.

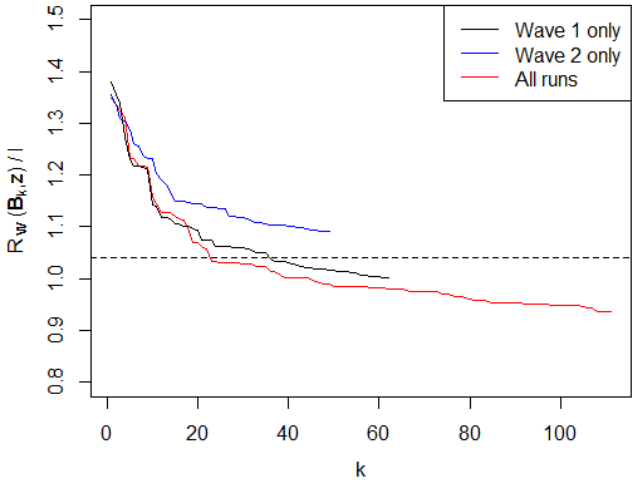

Figure 6.13. The reconstruction error of the RTMT observations given by the SVD basis, when only the wave 1 runs are used to define the basis, when only the wave 2 runs are used, and when all of the known runs are used, with $\mathbf{W} = \boldsymbol{\Sigma}_{\boldsymbol{\eta}} = 140.52\boldsymbol{\mathcal{I}}_{8192}$ in each case. The horizontal dotted line shows the history matching bound.

Finally, it could be the case that there are not enough degrees of freedom in the SVD basis for the new ensemble, so that while there is more information relating to $\mathbf{z}$ contained in the new ensemble, in representing the ensemble with only 49 directions, this gets ignored.

This final case is perhaps shown to be the answer by the line in Figure 6.13 relating to the reconstruction error when the SVD basis is calculated using all of the runs. This line is similar to that for the wave 1 basis until a basis size of around 18, before this combined basis shows an improvement. The extra degrees of freedom allowed here are exploited to give better reconstructions of $\mathbf{z}$.

The basis defined using both ensembles allows a more accurate reconstruction of $\mathbf{z}$ to be found, hence the rotation algorithm should be applied to this. It is generally the case when history matching that at least some of the runs from the previous wave will have been ruled out. Due to this, rotating this basis in the standard manner may lead to basis vectors being selected that do not explain much of the variability in NROY space, the region that emulation is now performed in. An extension to the optimal rotation algorithm is required for this case.

If m ensembles $\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(m)}$ have been generated, let all of the runs of the computer model be denoted by

$$\mathbf{F} = (\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(m)})$$

where $\mathbf{F}^{(i)}$ contains n_i runs of $f(\cdot)$, with $n = n_1 + \ldots + n_m$. From (2.15), the SVD basis

for $\mathbf{F}$ is calculated as

$$\mathbf{F}_{\boldsymbol{\mu}}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

with $\mathbf{F}_{\boldsymbol{\mu}}$ the centred ensemble as defined in (4.2), leading to the usual definition of the SVD basis $\boldsymbol{\Gamma}$ as the first $(n-1)$ columns of $\mathbf{V}$. Define $\mathbf{F}_{NROY}$ as the model runs that lie in the current NROY space:

$$\mathbf{F}_{NROY} = \{(f(\mathbf{x}) - \boldsymbol{\mu}) \in \mathbf{F}_{\boldsymbol{\mu}} | \mathbf{x} \in \mathcal{X}_{NROY}^{(i)} \forall i = 1, \ldots, m-1\}$$

with $\mathbf{F}_{NROY}$ consisting of $\hat{n}$ model runs. If the m ensembles were generated sequentially, i.e. the wave k ensemble $\mathbf{F}^{(k)}$ was given by sampling from within the wave $(k-1)$ NROY space, then $\mathbf{F}_{NROY}$ will be all of the runs from the latest ensemble $\mathbf{F}^{(m)}$, and possibly some runs from the previous ensembles.

The optimal rotation proceeds as in Section 5.4, with the variability conditions edited:

1. If $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) > T$, stop and revisit the specification of $\mathbf{W}$. Else set $k = 1$.

2. Let $\boldsymbol{\Gamma}_k^* = (\boldsymbol{\gamma}_1^*, \ldots, \boldsymbol{\gamma}_{k-1}^*, \mathbf{B}\boldsymbol{\lambda}_k)$ and set

$$\boldsymbol{\lambda}_k^* = argmin_{\boldsymbol{\lambda}_k} \|\mathbf{z} - \boldsymbol{\Gamma}_k^* (\boldsymbol{\Gamma}_k^{*T} \boldsymbol{\Gamma}_k^*)^{-1} (\boldsymbol{\Gamma}_k^*)^T \mathbf{z}\|_{\mathbf{W}}$$

   with the constraint that $\mathbf{B}\boldsymbol{\lambda}_k^*$ explains at least v_k of the variability in NROY space:

$$\mathcal{V}_k(\boldsymbol{\Gamma}_k^*, \mathbf{F}_{NROY}) \geq v_k$$

   Define the new normalised vector as

$$\boldsymbol{\gamma}_k^* = \frac{\mathbf{B}\boldsymbol{\lambda}_k^*}{\|\mathbf{B}\boldsymbol{\lambda}_k^*\|}$$

   and set $\boldsymbol{\Gamma}_k^* = (\boldsymbol{\gamma}_1^*, \ldots, \boldsymbol{\gamma}_{k-1}^*, \boldsymbol{\gamma}_k^*)$

3. Calculate the ensemble residual $\mathbf{F}_\epsilon$ for $\mathbf{F}_{NROY}$ using the current basis $\boldsymbol{\Gamma}_k^*$:

$$\mathbf{F}_\epsilon = \mathbf{F}_{NROY} - \boldsymbol{\Gamma}_k^* (\boldsymbol{\Gamma}_k^*)^T \mathbf{F}_{NROY}$$

Calculate the SVD of $\mathbf{F}_\epsilon$ to give the residual basis $\mathbf{B}_\epsilon$

$$\mathbf{F}_\epsilon^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$
$$\mathbf{B}_\epsilon = \mathbf{V}_{\hat{n}-1-k}$$

for $\mathbf{U}$, $\boldsymbol{\Sigma}$ and $\mathbf{V}$ as in (2.15), and for $\mathbf{V}_{\hat{n}-1-k}$ denoting the first $(\hat{n}-1-k)$ columns of $\mathbf{V}$. This gives an orthogonal rank $(\hat{n}-1)$ basis (by Section 4.6.2)

$$\boldsymbol{\Gamma}^* = (\boldsymbol{\Gamma}_k^*, \mathbf{B}_\epsilon)$$

4. Define $q \geq k$ as the minimum value satisfying

$$\mathcal{V}(\boldsymbol{\Gamma}_q^*, \mathbf{F}_{NROY}) \geq v_{tot}$$

where $\boldsymbol{\Gamma}_q^*$ is the first q columns of $\boldsymbol{\Gamma}^*$. If

$$\mathcal{R}_{\mathbf{W}}(\boldsymbol{\Gamma}_q^*, \mathbf{z}) < T$$

then stop, and return $\boldsymbol{\Gamma}_q^*$ as the truncated basis. Else, proceed to step 5.

5. Define $\tilde{\mathbf{F}}_\epsilon$ and $\tilde{\mathbf{B}}_\epsilon$ as the residual ensemble and residual basis for the full ensemble, $\mathbf{F}_{\boldsymbol{\mu}}$:

$$\tilde{\mathbf{F}}_\epsilon = \mathbf{F}_{\boldsymbol{\mu}} - \boldsymbol{\Gamma}_k^*(\boldsymbol{\Gamma}_k^*)^T\mathbf{F}_{\boldsymbol{\mu}}$$
$$\tilde{\mathbf{F}}_\epsilon^T = \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^T$$
$$\tilde{\mathbf{B}}_\epsilon = \tilde{\mathbf{V}}_{n-1-k}$$

Set $k = k+1$, $\mathbf{B} = \tilde{\mathbf{B}}_\epsilon$, and return to step 2 to select a new vector.

The constraint in step 2 that each selected basis vector explains a minimum proportion of variability,

$$\mathcal{V}_k(\boldsymbol{\Gamma}_k^*, \mathbf{F}_{NROY}) \geq v_k,$$

and the selection of q in step 4, so that

$$\mathcal{V}(\boldsymbol{\Gamma}_q^*, \mathbf{F}_{NROY}) \geq v_{tot}$$

now depend on $\mathbf{F}_{NROY}$, rather than only the current ensemble. We are not interested in explaining the variability across all of the known model runs, because we only need emulators that are accurate in the current NROY space. Therefore, this change in constraints ensures that there is enough signal from NROY space so that the coefficients for each of the leading basis vectors from the rotated basis can be emulated in NROY space, and that at least v_{tot} of the total variability in NROY space is explained by the new basis.

The new step 5 is required because if the current selected basis vectors, combined with the leading basis vectors from the residual ensemble (based on $\mathbf{F}_{NROY}$), do not satisfy $\mathcal{R}_{\mathbf{W}}(\mathbf{\Gamma}_q^*, \mathbf{z}) < T$, then further iterations must be performed. Hence, the residual ensemble and residual basis for the ensemble of all known runs, $\mathbf{F}_{\boldsymbol{\mu}}$, are calculated, so that all of the additional directions can be considered for the next basis vector. Including the ruled out runs in the initial definition of the basis to be rotated, and thereafter in the definition of the residual basis that is rotated at each step, gives the rotation algorithm greater flexibility, as more directions can be considered, and combined to give the minimum reconstruction error, under the variability constraints.

At step 3, the basis is completed by calculating the residual basis for the NROY ensemble $\mathbf{F}_{NROY}$, rather than $\mathbf{F}_{\boldsymbol{\mu}}$, to ensure that the new basis explains as much of the variability in $\mathbf{F}_{NROY}$ as possible. Since variability from ruled out runs may have been incorporated into the first few basis vectors, this rank $(\hat{n} - 1)$ basis for $\mathbf{F}_{NROY}$ (consisting of $\hat{n}$ runs) may not explain 100% of the variability in $\mathbf{F}_{NROY}$. However, as the later basis vectors of $\mathbf{\Gamma}^*$ will almost always be such that $\mathcal{V}_k(\mathbf{\Gamma}^*, \mathbf{F}_{NROY}) \approx 0$, the new basis explains the great majority of $\mathbf{F}_{NROY}$ (and regardless, the basis is always truncated prior to needing the last small proportion of variability). Due to this small amount of information being lost, we no longer have defined a rotation in NROY space (although we still have a rotation in the full space), but the new basis still retains all desired properties for emulation and history matching.

We will apply this new version of our algorithm to find the wave 2 basis for each output in Section 6.4.4.

6.4.3. Adding discrepancy for TA

Using the same method as previously to define the discrepancy (i.e. setting $\mathbf{\Sigma}_{\boldsymbol{\eta}}$ as a multiple of the identity matrix) always resulted in empty NROY spaces for TA at wave 2. Given that every observed run of the model thus far has exhibited a large warm bias at the South Pole, and that we rule out all $\mathbf{x} \in \mathcal{X}$ if a constant error across the output grid is assumed, this is good evidence of there being a structural error in the model.

Using the same discrepancy as specified at wave 1 does not lead to all $\mathbf{x}$ being ruled out at wave 2 for CLTO and RTMT, despite the fact that all model runs contain large biases. However, for these fields, the errors vary more across the ensemble, and wave 2 was able to fix several biases, albeit at the expense of introducing other biases. In contrast, the new ensemble for TA did not fix the South Pole warm bias at all, with the same bias always observed, and hence we treat it differently. Whilst ruling out all of parameter space is a result in itself, it suggests that the discrepancy specification may have been incorrect (likely, due to the simple form we used to allow us to search for the observations). It is again more useful to have a non-empty NROY space so that parameter settings in the direction of $\mathbf{z}$ may be suggested, for possible future runs.

In the previous two chapters, the parallel between the weight matrix $\mathbf{W}$ for the reconstruction error and the multivariate implausibility was drawn and applied on several occasions. The discrepancy variance $\mathbf{\Sigma}_{\boldsymbol{\eta}}$ could therefore be thought of as a weight matrix, defined so that errors in parts of the output space are given less importance. In the case of TA, as it appears that the South Pole bias may be structural, the discrepancy variance could be set so that differences between $\mathbf{z}$ and reconstructions here are given less importance. This would have the effect of decreasing implausibilities, so that NROY space may become non-empty, and may allow other biases in the output to be fixed: currently, the difference between $\mathbf{z}$ and reconstructions is dominated by the South Pole bias. Rather than giving the conclusion that the model cannot reproduce $\mathbf{z}$, it may be more useful for modellers to learn which aspects of the output can be improved by varying $\mathbf{x}$. Therefore, the dependence of the reconstruction error on the South Pole bias will be lessened.

Which grid boxes should be given more discrepancy will be derived from the ensemble. Our idea is similar to the approach of Chang et al. (2016), where the discrepancy is represented by a vector, constructed by considering the differences between the observations and the

ensemble members in each grid box. If these differences are above a chosen threshold, then a non-zero discrepancy is set for that grid box (see Section 2.7.3 for more details).

The discrepancy should represent the difference between the output of the model at its 'best' parameter setting and the observations. Therefore, selecting which grid boxes should have increased discrepancy should not be based on, for example, whether the average anomaly is greater than a chosen threshold. This could be dominated by large anomalies for some ensemble members, and may lead to the discrepancy for a box being increased, even if some ensemble members have a low or zero anomaly for this box. Instead, we order the ensemble anomalies for each grid box, and set D_i equal to the anomaly for grid box i that exceeds a chosen proportion, p, of anomalies for that grid box:

$$D_i = |\mathbf{z}_i - f_i(\mathbf{x}_j)| \quad \text{such that} \quad \frac{1}{n}\sum_{k=1}^{n} I(|\mathbf{z}_i - f_i(\mathbf{x}_k)| \leq |\mathbf{z}_i - f_i(\mathbf{x}_j)|) = p$$

where $I(\cdot)$ is the indicator function. Then, more discrepancy will be assigned to grid boxes if D_i is greater than t°C, where t is a parameter fixed using our judgement. We set the i^{th} diagonal element of the discrepancy variance matrix $\boldsymbol{\Sigma_\eta}$ as

$$[\boldsymbol{\Sigma_\eta}]_{ii} = \begin{cases} d & \text{if} \quad D_i > t \\ 1 & \text{otherwise} \end{cases}$$

where t is the threshold used to select which grid boxes are most biased, and $d >> 1$ is the discrepancy given to these biased grid boxes. This ensures that errors in regions that have large biases observed in them have more discrepancy associated with them. In the reconstruction error with $\mathbf{W} = \boldsymbol{\Sigma_\eta}$, and hence the multivariate implausibility for history matching, errors in these boxes will have less effect on the overall error or implausibility, so that runs may not be ruled out if they match $\mathbf{z}$ elsewhere in the output space, regardless of large biases being observed for the selected grid boxes.

Using this setting of the discrepancy for TA may allow parameter settings to be identified that fix other biases in the output, which may also lead to improvements in the South Pole anomaly. For this problem, we set $p = 0.5$, and $t = 7$, and use all of the wave 1 and 2 runs to calculate D_i. We use a large value of p as we know that there are no ensemble members with anomalies within the tolerance to error for the South Pole, and this combined with our choice of t successfully identifies the main bias here. The grid boxes selected using

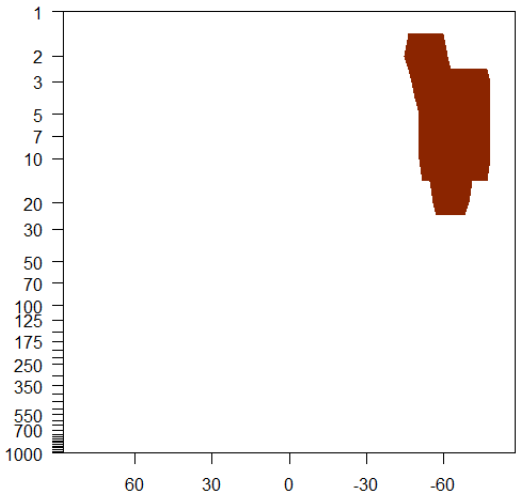

Figure 6.14. The grid boxes where the discrepancy is increased for TA.

this threshold are shown in Figure 6.14 (note: some boxes have cold biases greater than 7°C, but these are ignored in this application, as the South Pole anomaly is deemed to be most important). To complete the discrepancy specification, we set $d = 100$, as this was found to be a value for the discrepancy that lessened the effect of anomalies in the highlighted region enough so that rotation, emulation and history matching would lead to a non-empty NROY space. Let this discrepancy matrix with $d = 100$ be denoted by $\mathbf{W}_{TA}$.

Due to the structure of the discrepancy variance changing, we now revisit the wave 1 NROY space. Based on the wave 2 runs, we are now more confident about the presence of a structural error, and therefore the wave 1 TA emulators with the wave 1 discrepancy should not be used to rule out space: our beliefs about the discrepancy have changed based on observing more data, and it may be more than assumed at wave 1. The wave 1 NROY space is therefore redefined as

$$\mathcal{X}^{(1)}_{NROY} = \{\mathbf{x} \in \mathcal{X} | \mathcal{I}^{(1)}_{c,CLTO}(\mathbf{x}) < 274.12,\ \mathcal{I}^{(1)}_{c,RTMT}(\mathbf{x}) < 121.78\} \tag{6.8}$$

with the superscript (1) indicating that this is the implausibility and NROY space at wave 1. This new definition of the wave 1 NROY space contains 41.41% of the original parameter space; TA was the field that most strongly constrained this NROY space previously.

Redefining the wave 1 NROY space, having already designed and run a new ensemble, means that the wave 2 ensemble does not span $\mathcal{X}^{(1)}_{NROY}$, and instead only represents a subset of this. To overcome this issue, when fitting new emulators for each field, we will

include runs that were not ruled out according to the implausibility for each individual field, adding these not ruled out runs to that field's $\mathbf{F}_{NROY}$. For example, for RTMT, 23 of the wave 1 runs were not ruled out according to the implausibility and bound for RTMT. Despite each of these runs being ruled out previously when combined with the CLTO and TA implausibilities, they will be used in the fitting of the wave 2 emulators, to improve the modelling within the RTMT not implausible space, and to give a more space-filling representation of the redefined $\mathcal{X}^{(1)}_{NROY}$. For CLTO, 14 of the wave 1 runs were not ruled out, and are added to $\mathbf{F}_{NROY}$ for CLTO. None of the wave 1 runs for TA are treated as ruled out due to the structure of the discrepancy changing.

6.4.4. Wave 2 rotation and emulation

Using the new specification of the discrepancy for TA, and the method for selecting a basis using all of the observed runs of the model (Section 6.4.2), we now apply the rotation algorithm to find a wave 2 basis for each output field.

When performing the rotation for CLTO and RTMT, the weight will now be the discrepancy variance:

$$\mathbf{W}_{CLTO} = 116.86\boldsymbol{\mathcal{I}}_{8192}$$

$$\mathbf{W}_{RTMT} = 140.52\boldsymbol{\mathcal{I}}_{8192}$$

The rotation for TA is found using $\mathbf{W}_{TA}$, and therefore we use the weighted projection from Section 5.7 due to the non-constant diagonal of this matrix. A multiple of $\mathbf{W}_{TA}$ may need to be applied so that the reconstruction of $\mathbf{z}$ is not ruled out, as in (6.7). This is because our chosen $\mathbf{W}_{TA}$ represents our beliefs about the structure of the discrepancy, not the magnitude of it, so that this can be adjusted if necessary, based on the reconstruction with the truncated rotated basis.

The reconstruction errors for the new rotated bases are given in Table 6.4, showing an improvement over wave 1 for CLTO and RTMT. A superscript of (i) denotes the wave. The reconstruction of the CLTO observations is now 13.6% better, that for RTMT has been improved by 8.4%, whilst the reconstruction for TA has not improved if each grid box is treated equally. To set the multiple required for the discrepancy for TA, the 0.25 value of the chi-squared distribution is used, as this is the highest value that led to a

Output	$\mathcal{R}_{\boldsymbol{\Sigma}_\mathbf{e}}((\boldsymbol{\Gamma}_q^*)^{(1)}, \mathbf{z})/l$	$\mathcal{R}_{\boldsymbol{\Sigma}_\mathbf{e}}((\boldsymbol{\Gamma}_q^*)^{(2)}, \mathbf{z})/l$	$\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$	Bound
CLTO	42.38	36.60	$116.86\boldsymbol{\mathcal{I}}_l$	274.12
RTMT	51.89	47.54	$140.52\boldsymbol{\mathcal{I}}_l$	121.78
TA	6.60	6.61	$1.62\mathbf{W}_{TA}$	231.00

Table 6.4. A table showing the reconstruction errors for the truncated rotated basis at waves 1 and 2, and the discrepancy variances and history matching bounds for each field.

non-empty NROY space. The coefficient and field implausibilities for TA are no longer perfectly correlated, and hence the new bound for TA is found by taking the 99.5% value of the distribution from (6.3) (shown in Figure C.6).

The VarMSE plots for the rotated basis of each output field, with the discrepancies from Table 6.4 used as $\mathbf{W}$, are given in Figure 6.15. The truncated rotated basis lies below the history matching bound for each of the fields. For TA, this is by construction, due to the new specification of the discrepancy. For CLTO and RTMT, because of the extra degrees of freedom allowed by the new rotation, we would expect to reduce the reconstruction error at least as much as at wave 1. For each field, the rotation has given an improvement over the reconstruction errors given by the SVD bases (indicated by the dotted red lines), and in each case is close to the minimum bound given by the full SVD basis.

None of the truncated SVD bases are below the history matching bound line, showing the importance of rotations in this application. When 30 basis vectors are used for projection, the SVD basis for CLTO and TA would still rule out the observations, although this would not be the case for RTMT.

CLTO proved the most difficult to find an improved basis for, with the reconstruction error decreasing gradually, almost in parallel to the SVD basis. The improvement here has come from incorporating more important patterns into the first basis vector, so that this vector by itself is more useful. 13% of the variability in the not ruled out ensemble is explained by the first rotated basis vector ($\mathcal{V}_1(\boldsymbol{\Gamma}^*, \mathbf{F}_{NROY}) = 0.13$), compared to $\mathcal{V}_1(\boldsymbol{\Gamma}, \mathbf{F}_{NROY}) = 0.17$ for the first SVD basis vector. For CLTO, the first 19 basis vectors, with $\mathcal{V}(\boldsymbol{\Gamma}_{19}^*, \mathbf{F}_{NROY}) > 0.85$, are emulated.

The rotated basis for RTMT only gives a small improvement for the first basis vector, despite projection onto this vector giving $\mathcal{V}_1(\boldsymbol{\Gamma}^*, \mathbf{F}_{NROY}) = 0.21$, compared to 55% of $\mathbf{F}_{NROY}$ being explained by the first vector of the SVD basis $\boldsymbol{\Gamma}$. However, the second rotated vector immediately reduces the reconstruction error below the history matching bound,

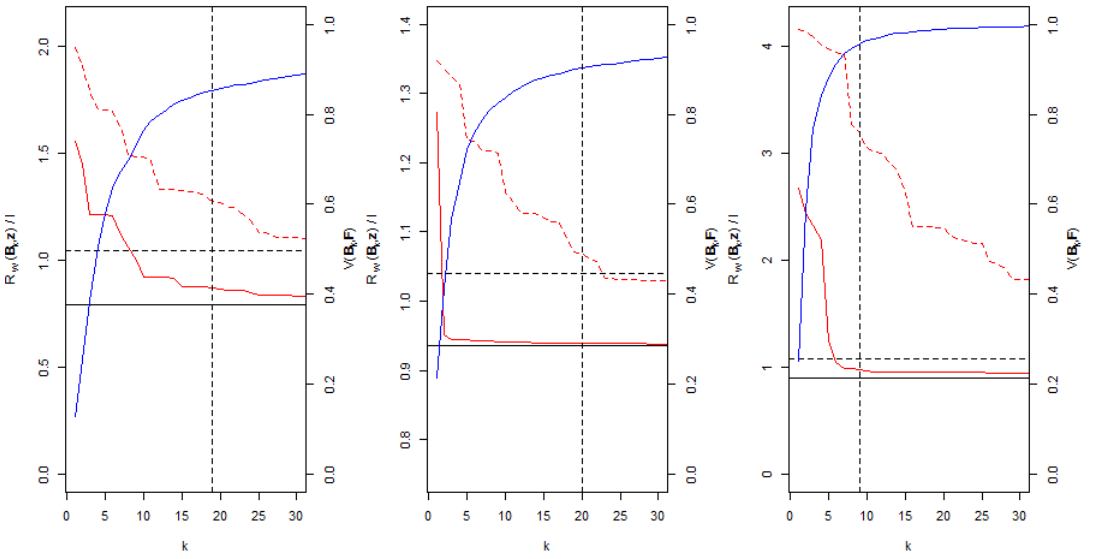

Figure 6.15. The VarMSE plots for CLTO, RTMT and TA respectively, with $\mathbf{W} = \mathbf{\Sigma}_{\boldsymbol{\eta}}$ for each field. The solid red lines show the reconstruction error for the rotated basis, the dotted lines the error for the SVD basis.

whilst the SVD basis decreases the reconstruction error much more slowly. Coefficients for the first 20 basis vectors are emulated ($\mathcal{V}(\mathbf{\Gamma}^*_{20}, \mathbf{F}_{NROY}) > 0.9$).

Similarly, the TA rotated basis gives a reasonable improvement for the first basis vector, with five basis vectors being enough for the reconstruction error to fall below the history matching bound. Again, the SVD basis does not decrease as quickly. Only 9 vectors are required to explain 95% of the ensemble variability for TA, and hence 9 emulators are built for these coefficients.

To assess how these new basis choices have improved compared to wave 1, the anomaly between the observations themselves and the reconstruction of the observations using the truncated rotation basis is given in Figures 6.16, 6.17 and 6.18 for both wave 1 and 2 for each field. Table 6.4 has already indicated that improvements have been achieved, but by considering these plots, which particular anomalies have now been removed can be identified. This may suggest anomalies in the model runs that can now be removed due to the better basis choice.

Figure 6.16 shows this comparison for CLTO. Several of the regions of strong positive bias, given by the darker red colours, have been reduced in size by the new basis. In particular, the positive biases around Australia, in the Pacific Ocean, and in the North Atlantic have been reduced. There is perhaps slightly more map coloured white in the Southern Ocean by the new basis reconstruction, as well as in the Northern Pacific. Over Eurasia, there is

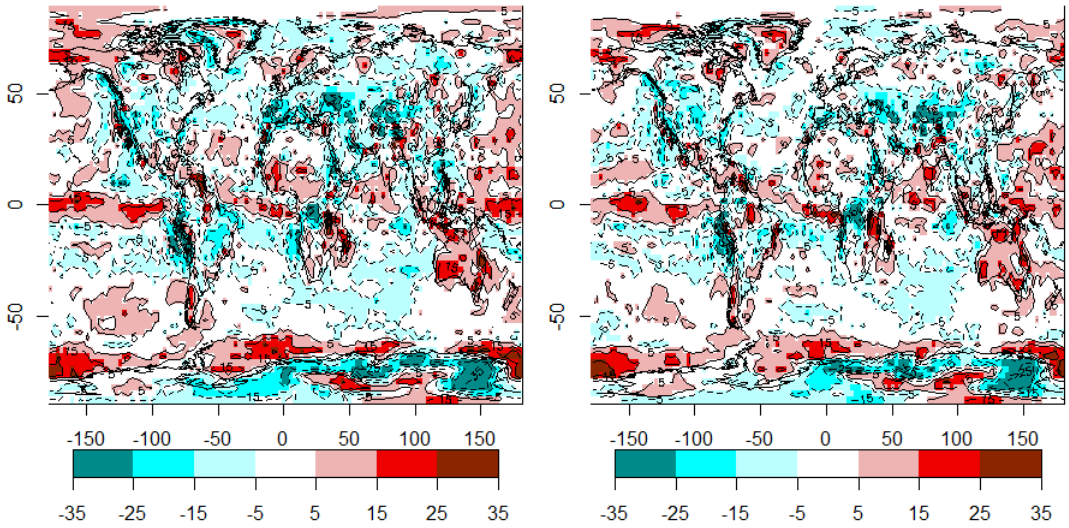

Figure 6.16. The anomaly for the reconstruction of the CLTO observations using the wave 1 truncated rotated basis (left), and the anomaly for the reconstruction with the wave 2 truncated rotated basis.

less negative bias in places, possibly at the expense of slightly more elsewhere.

For RTMT (Figure 6.17), the positive bias in the Western Pacific has been reduced, as has the bias over Northern Europe. Elsewhere, a lot of the stronger bias patterns remain for the new basis, with perhaps slight improvements. There is slightly more white colouring for the new basis reconstruction, in both the Pacific and Southern Oceans.

Figure 6.18 shows the same comparison for TA. Due to the new specification of the discrepancy at wave 2, it is not surprising that the anomaly in the reconstruction for the South Pole region is larger than at wave 1. The region given more discrepancy (Figure 6.14) corresponds to the largest anomalies in this reconstruction. However, ignoring this region, as errors here have little effect on the implausibility, this definition of the discrepancy has allowed biases to be reduced in other regions of the output. The cold biases high in the atmosphere, and at the South Pole, have been reduced at wave 2, as has the warm bias previously found in the northern hemisphere. By assuming that the South Pole anomaly is a structural error, it may now be possible to identify input parameters that improve upon other aspects of the TA output.

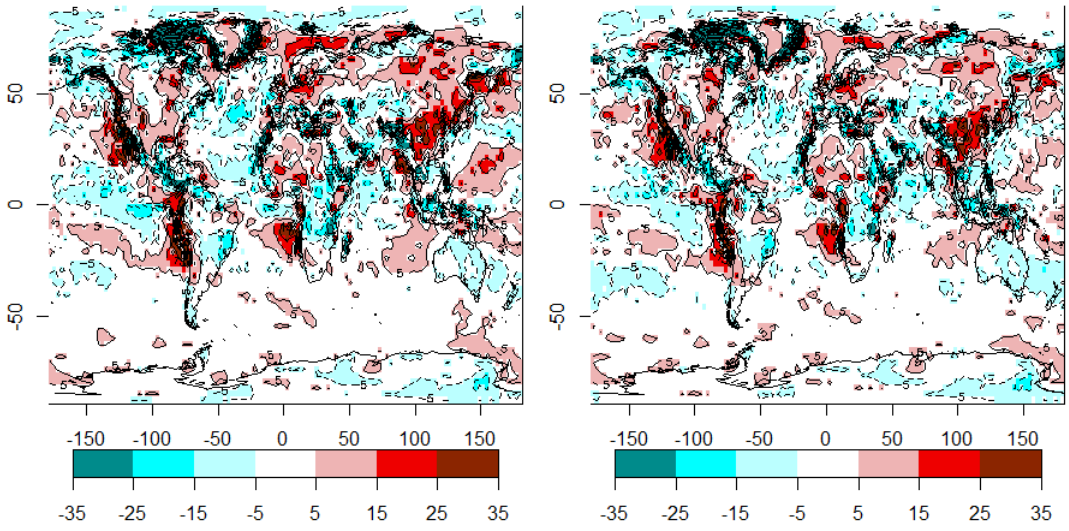

Figure 6.17. The anomaly for the reconstruction of the RTMT observations using the wave 1 truncated rotated basis (left), and the anomaly for the reconstruction with the wave 2 truncated rotated basis.

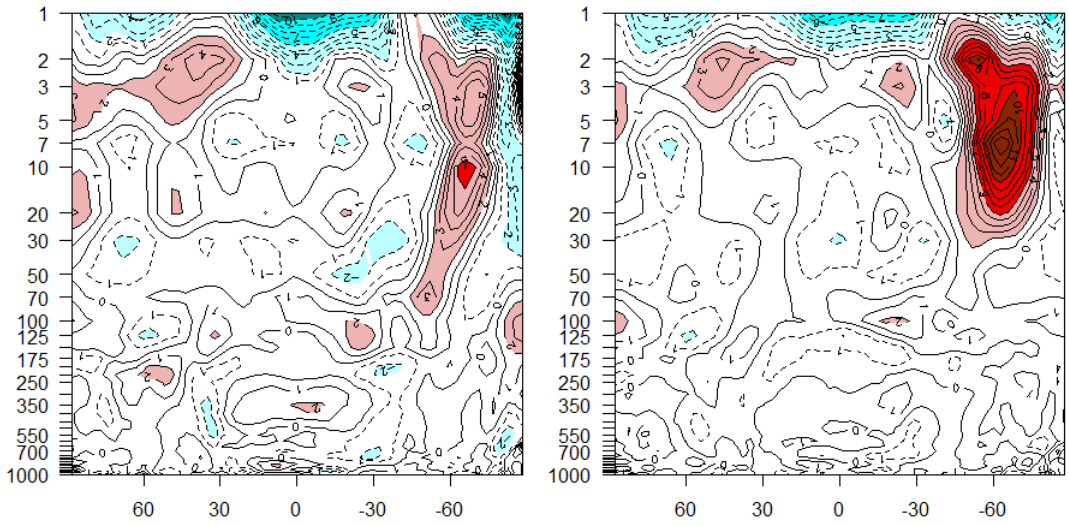

Figure 6.18. The anomaly for the reconstruction of the TA observations using the wave 1 truncated rotated basis (left), and the anomaly for the reconstruction with the wave 2 truncated rotated basis.

6.4.5. Wave 2 history matching

Using our emulators, discrepancies, and the coefficient implausibility bounds, a new NROY space can be defined. The wave 2 NROY space is defined as parameter settings in the wave 1 NROY space that have wave 2 implausibility less than the bound for each of CLTO, RTMT and TA:

$$\mathcal{X}^{(2)}_{NROY} = \{\mathbf{x} \in \mathcal{X}^{(1)}_{NROY} | \mathcal{I}^{(2)}_{c,CLTO}(\mathbf{x}) < 274.12,\ \mathcal{I}^{(2)}_{c,RTMT}(\mathbf{x}) < 121.79,\ \mathcal{I}^{(2)}_{c,TA}(\mathbf{x}) < 231.00\}$$

where $\mathcal{X}^{(1)}_{NROY}$ is the redefined wave 1 NROY space (i.e. without TA, given in (6.8)). Using this, the percentage of parameter space that lies in the wave 2 NROY space can be calculated by sampling from $\mathcal{X}^{(1)}_{NROY}$ and evaluating the coefficient implausibilities for each output field. Given these implausibilities, the wave 2 NROY space contains only 0.03% of $\mathcal{X}$. By taking each of the fields individually, 48.90% of $\mathcal{X}^{(1)}_{NROY}$ is not ruled out by the CLTO implausibility, 49.38% is not ruled out by RTMT, and 0.19% is not ruled out for TA.

Despite changing the discrepancy structure for TA so that the grid boxes with largest anomalies are mostly ignored, the majority of space is again ruled out for TA, suggesting that most of parameter space contains large biases away from the observations. Revisiting the discrepancy so that more grid boxes are treated as possible structural errors would reduce the implausibility, and would result in less space being ruled out. However, this result is only valid given the chosen discrepancies, and hence the bounds used to rule out runs based on the coefficient implausibility. As the discrepancy is not actually known, the percentage of space ruled out is not as important as searching this space for parameter settings where output more consistent with the observations may be found.

6.5. Discussion

In this chapter, we developed our methodology further, tailoring it specifically to suit the multiple large output fields typically seen in climate model tuning. We applied our methodology to the Canadian atmosphere model, CanAM4.

History matching a model with thousands of outputs (e.g. 8192 for the CLTO field of CanAM4) using the field implausibility, as in Chapter 5, is not practical due to the inability to invert the $l \times l$ variance matrix (dependent on $\mathbf{x}$) as often as required. Therefore, we combined our rotation algorithm with history matching using the coefficient implausibility, and found that the standard chi-squared bound used to define NROY space was inaccurate. This standard bound for the coefficients ruled out runs that were deemed to be acceptably close to the observations on the field, and was generally discovered to be too strict a bound.

The field implausibility is naturally more suitable for finding output similar to $\mathbf{z}$, and would be used for history matching if computation time was no issue. In order to be able

to use the coefficient implausibility as a conservative proxy for the field implausibility, we developed a Bayesian hierarchical model linking the coefficient implausibility to the field implausibility. This was based on the observation that there is a strong positive relationship between the two implausibilities. This model was then used to infer the coefficient implausibility associated with the bound for the field implausibility. Using this new bound to history match the toy function on the coefficients was found to give more conservative, but accurate, results than the chi-squared bound.

Due to the structure of the priors for our model linking the implausibilities, only a small number of evaluations of the time-consuming field implausibility are required to update the model, and hence this can significantly reduce the amount of time taken to perform history matching. For CanAM4, the two implausibilities are generally perfectly correlated, and hence inferring the bound is straight-forward and requires very few runs of the field implausibility. For TA at wave 2, more evaluations of the field implausibility were required, due to the different nature of $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$.

Having demonstrated the performance of this extension of history matching for high-dimensional output on the toy function, this was then combined with the basis rotation methodology from Chapter 5 and used to history match the climate model, CanAM4, as the dimension of the output is too high to use the field implausibility for all $\mathbf{x}$. By applying the iterative rotation algorithm, we found rotations that improved upon the SVD basis. This optimisation problem was not significantly more difficult than for the toy function: the ensemble size for each of the problems was similar, and hence a similar number of rotation parameters were optimised at each step. The time required to perform the same number of evaluations of the reconstruction error for new rotated vectors increased slightly due to the higher dimensional matrix multiplications, but suitable bases were generally found with a maximum time allowed of 5 minutes.

This demonstrates that the rotation step is not prohibitively difficult for high-dimensional output. If more ensemble members were available (increased n), then more rotation parameters would be optimised at each step; the problem becomes more complicated with n. If n is too large for the optimiser to converge in a reasonable time, a subset of the n SVD basis vectors could be selected, and only these be given rotation parameters to find optimal values for. This subset could be selected by, for example, selecting each of the first m, where the first m explain the majority of the ensemble variability, combined with

low-eigenvalue vectors that have the greatest impact in reducing the reconstruction error when they are added to the basis.

In order to calculate the implausibility, the discrepancy variance must be specified. Due to no expert judgement regarding the structure of the discrepancy, this was treated as a constant across output space at wave 1, so that the discrepancy variance was a multiple of the identity matrix. To ensure that the observations shouldn't be ruled out, we set this multiple based on the reconstruction error given by the truncated rotated basis at wave 1. This method of setting the discrepancy is not meant to be an accurate representation of the discrepancy. Instead, it aims to ensure that the runs expected to be closest to $\mathbf{z}$ will not be ruled out. If further waves are carried out, the new ensemble may then contain more signal in the important directions highlighted by the rotated basis.

Based on the emulators for the rotated bases, and the bounds for the coefficient implausibilities for each field given by the Bayesian hierarchical model, we defined the wave 1 NROY space. By sampling from this space, we designed a wave 2 ensemble, which was then run on CanAM4. Using the wave 2 ensemble, we performed new rotations, using another extension to the previous methodology.

This extension involved rotating the SVD basis calculated from all available runs of CanAM4, before requiring that the previous variance constraints held for the runs in the current NROY space, rather than across all known runs. This addition was necessary as the wave 2 ensemble only had 49 members, so that a basis defined solely using these runs had fewer degrees of freedom than that at wave 1, giving higher reconstruction errors than previously, due to the complexity of the problem ($l >> n$). This issue was not observed when history matching the toy function, due to the fact that the ensemble size never decreased between waves, and also because of the relatively low dimension of the toy function, simplifying emulation and history matching so that clear improvements were made at each wave.

This extension has the effect of allowing the optimiser to search in more directions, due to the larger SVD basis. Patterns that may only be contained in ruled out runs can then be combined with patterns from runs in NROY space, whilst having a larger n allows much more flexibility: instead of a 62-dimensional subspace of 8192-dimensional space (or 2368 for TA), a 111-dimensional subspace can now be searched. By modifying the variability

constraint so that only runs in NROY space are considered, emulation is still achievable.

Following the method used at wave 1, a non-empty NROY space could not be found for TA. This is perhaps unsurprising, given that every ensemble member contains a large positive anomaly over the South Pole. A different specification of the discrepancy variance was used to incorporate this belief that there may be a structural error in the model, by considering $\mathbf{\Sigma_\eta}$ as the reconstruction error weight matrix $\mathbf{W}$. The grid boxes with the largest anomalies between the observations and ensemble were identified, and then the diagonal values of $\mathbf{\Sigma_\eta}$ associated with these boxes were increased. As the inverse of this matrix is used in the reconstruction error (and implausibility), increasing the discrepancy for certain regions has the effect of allowing the output to be more incorrect in these regions, and hence decreasing the implausibility. Improvements in other regions of the output may then be identified, because runs with large errors in the chosen region may no longer be ruled out, rather than these other improvements being hidden and ruled out due to the dominant anomalies.

For large spatial fields, specifying the discrepancy variance is challenging. Our method allows us to highlight regions of the output where there may be structural errors, and places less importance on anomalies in these regions. We only divided the grid boxes into two sets, but this method could be extended to incorporate additional judgements. Furthermore, we only varied the diagonal entries of the discrepancy. To find a more accurate specification for $\mathbf{\Sigma_\eta}$, non-diagonal values could be similarly defined.

Ideally, further waves would be performed to better explore the input space, although this is not necessarily practical due to the long running time of CanAM4. Based on the wave 2 NROY space, biases that a next wave might expect to remove have been identified for each output field, and a hypothetical wave 3 would be designed in this latest NROY space. At this stage, Bayesian calibration is not likely to yield useful results, given the large anomalies observed, and the unknown specification for the discrepancy. Given these problems, history matching is currently more appropriate. As for the toy function, calibration may become more suitable after more waves of history matching.

6.6. Conclusion

The basis rotation methodology developed in Chapters 4 and 5 has been applied to a climate model with high-dimensional output, with extensions provided so that this is computationally possible. We provided a Bayesian hierarchical model to link the coefficient implausibility with the field implausibility, extended our optimal rotation algorithm to incorporate all model runs, and started to develop methods for specifying spatial discrepancies. In every application of our method, rotation resulted in a basis superior to the SVD basis, in terms of reducing the reconstruction error given by the truncated basis. Some observed biases from the standard model run have been removed, which is of ongoing interest to the modellers at CCCMA.

This chapter has shown that our method scales to important real world examples, whilst also providing a reminder that tuning such complex models is a difficult problem, unlikely to be accurately performed with only two waves (whereas currently, calibration is generally performed without any history matching step). If it were possible, dividing the available runs into more waves may have provided better results. It is perhaps unlikely that a single wave, as is generally the case when Bayesian calibration is used, is enough to give accurate results, even if rotating the basis allows for searching in the correct directions of the output space.

7. Conclusion

This chapter provides a summary of the main results in this thesis, and a discussion of directions for future research.

7.1. Summary

In Chapter 3, we compared the use of regression and Gaussian process emulators for modelling the output of computer models with high-dimensional input spaces and univariate output. Fitting the correlated residual term of the Gaussian process gave a large improvement over the equivalent regression-only emulators, for each toy function and an environmental model, the IC fault model. Combining this comparison with multi-wave history matching experiments, we found that it is often reasonable to initially fit regressions to capture the large-scale variability in the function output, before fitting Gaussian processes at later waves to accurately model more local variation. The results of Bayesian calibration were greatly improved both by the use of Gaussian process emulators instead of regressions, and by performing a few waves of history matching prior to calibrating. We highlighted the sensitivity of the history matching results to the sample design.

Chapter 4 identified a flaw with the standard literature method for selecting a basis for large spatial computer model output, the basis derived by taking principal components of the ensemble, the SVD basis. We showed that if the basis is not able to accurately represent the observations, then any calibration or history matching exercise will likely lead to incorrect conclusions, with the wrong basis choice guaranteeing that fields similar to the observations cannot be found. This problem was evident in the climate model examples given, as well as for our toy example. We introduced a measure for the ability of a basis to represent the observations, the reconstruction error. To overcome the deficiencies of the

SVD basis, we developed a method for combining important patterns, possibly elicited or based on physical knowledge, with the variability from the ensemble, by defining the residual basis. This gives an orthogonal basis intended to be suitable for exploring the desired regions of the output space, while allowing emulation, if the elicited patterns were defined appropriately.

Chapter 5 extended this basis selection method into an automatic, iterative procedure for finding an optimal basis, given an ensemble and observations. The algorithm applies a rotation to the SVD basis, iteratively selecting new basis vectors such that the reconstruction error of the observations given by projection onto this basis is minimised, given constraints to ensure emulation is possible. Our optimal rotation algorithm is able to identify any important signal contained in the ensemble, and combine this with other patterns that explain more ensemble variability, so that this important signal can be emulated. The Bayesian calibration and history matching results given by the rotated basis were superior to those given by the SVD basis, with accurate parameter settings identified. As with the univariate functions, a refocussed approach to history matching, and then an application of Bayesian calibration, provided results with greater accuracy, strongly advocating this approach in calibration problems. We developed a method for designing ensembles in NROY space, to ensure that the new design included low-implausibility regions of parameter space. A weighted projection was also defined for applications where the weight matrix used to calculate the reconstruction error is not a multiple of the identity matrix.

In Chapter 6, we applied the basis rotation algorithm to CanAM4, a climate model with large spatial output. This required an extension for history matching using the coefficient implausibility rather than the field implausibility, due to the computational difficulties of this problem. We developed a Bayesian hierarchical model for predicting the bound for the coefficient implausibility that corresponds to the bound used to history match on the field, as the existing chi-squared bound for the coefficients was shown to rule out output consistent with the observations on the field. We extended our rotation algorithm to allow more directions to be searched at later waves. As a specification for the discrepancy variance was not available, we introduced a method for specifying the discrepancy to ensure that the observations are not ruled out at the first wave, allowing runs to be identified that contain some signal in the direction of the observations. At the following wave, due to the identification of a likely structural bias in one of the climate model outputs, a method for

specifying a more complicated spatial discrepancy was developed, allowing biases in other regions of the output space to be fixed.

7.2. Future work

There are a number of directions that could be explored in future, to expand upon methodology and ideas introduced in this thesis.

Throughout this thesis, it has always been found that the results of Bayesian calibration are superior when history matching has initially been used to reduce the parameter space, with new ensembles being run in the current NROY spaces. Where possible, it is clear that history matching should always be performed prior to Bayesian calibration. However, a problem for future research may be, if there are n runs of the computer model available, how should these best be allocated, to achieve the most accurate calibration results? How these available runs are split between waves is likely to have some effect on the accuracy of results.

It may be sensible to use a small (relative to the total number of available runs, n) space-filling design at wave 1, to capture large-scale variability in the model output. Performing history matching using emulators built for this ensemble would hopefully be able to remove large regions of parameter space that are inconsistent with the observations, allowing the remaining runs to be used to create more dense samples at later waves, focussed in the regions of interest. This would allow more accurate emulation of the parts of space we are interested in, emulating more local variability, rather than wasting resources at wave 1 so that the entire space can be accurately emulated. For applications such as CanAM4, it is not likely that n will be high, and there may not even be an opportunity to divide runs across waves, due to a limited access to the supercomputer resources required.

We restricted ourselves to orthogonal bases in the applications shown. Allowing non-orthogonal bases may allow greater flexibility in the possible representations for the observations. A related improvement could be the inclusion of a correlation between the emulators for each coefficient: although the basis vectors are orthogonal, the coefficients have sometimes been found to be correlated. More accurate emulation could be achieved through the use of a multivariate emulator of the coefficients.

When selecting an optimal basis using our algorithm, the reconstruction error is a natural choice for representing the quality of a basis, due to its relationship with the multivariate implausibility. However, how to define emulatability is a more difficult question. In this thesis, we have used the proportion of ensemble variability explained by projection onto a basis vector as a proxy for whether or not we can build an emulator for the coefficients. For problems where there are a large number of ensemble members, or dominant signals that are explained by several basis vectors, this may not be as suitable.

A potential extension to the general methodology for finding a basis through rotation could be for approaching problems where there are multiple observations $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$ of the same field, and how we would history match in such a scenario. All of the applications in this thesis have been for problems where there is a single observed field $\mathbf{z}$, and hence the rotation is performed to minimise the reconstruction error for $\mathbf{z}$, with history matching then ruling out runs inconsistent with the single observed $\mathbf{z}$. If there are instead N fields, there are a number of options for how to proceed.

The optimal basis for this problem could be selected by minimising a function of the reconstruction errors for each of the N observed fields, for example the maximum reconstruction error across the observations, to ensure that all $\mathbf{z}_i$ are represented adequately. However, this may not lend itself to history matching: as the observations are unlikely to be independent, calculating implausibilities for each field separately may not be appropriate. The multivariate implausibility could be used, but requires the specification of covariances between the observations.

If the N observed fields can be assumed to be samples from the distribution of the 'true' value of the real-world system, $\mathbf{y}$, then it may be possible to derive an implausibility that allows history matching based on this. For example, it may be possible to use the leading orthogonal direction(s) from the set of observations in order to history match.

We used a Bayesian hierarchical model to predict the bound that should be used for history matching using the coefficients, in order to not rule out model output that is consistent with the observations over the field. When calculating the bound for the CanAM4 fields at wave 1, we found that there was a perfect linear relationship between the field and coefficient implausibilities. When the discrepancy structure for TA changed at wave 2, this was no longer the case (although the implausibilities had a very high correlation).

However, it is not true that a diagonal specification for the discrepancy always leads to this. This relationship requires further exploration, with regards to how the structure of the discrepancy affects the implausibilities, and how the relationship changes as l increases. If the implausibilities become more correlated for large l, then fitting the hierarchical model to determine the bound becomes more straight-forward. This would be a beneficial property, as the model is only ever required for large l, i.e. when always calculating $\mathcal{I}_f$ is too time consuming.

For the air temperature (TA) output field, at wave 2 we identified a potential structural bias in the model, and defined the discrepancy variance accordingly, by highlighting the grid boxes at the South Pole with the largest anomalies. We defined two groups of grid boxes (those deemed to contain a structural error, and the remaining boxes), setting discrepancy values so that a rotation that did not rule out all of parameter space could be identified. There is a danger in specifying the discrepancy variance using the ensemble of runs, as the ensemble may suggest possible structural errors that are in fact because the correct region of parameter space has not been explored so far. However, for models with high-dimensional output, it is perhaps unlikely that this variance can be elicited. This is our experience with climate models, hence an approach based on the model runs may be necessary.

There are several possible extensions to our current method for specifying the discrepancy variance. In our application, we divided the output into two regions, but this could be extended to divide the output field into more groups, or defining values for grid boxes individually. The values for each grid box or region could be set based on other metrics, allowing differing levels of perceived bias to be reflected. It is likely that there should be correlations between grid boxes specified for $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$. Defining the off-diagonal elements for the discrepancy variance is more challenging.

The benefit of combining a specification for the structure of the discrepancy variance, with finding a multiple for this structure based on not wanting to rule out the observations when using the truncated rotated basis, is that history matching may be used to sequentially design the discrepancy variance. Our rotation algorithm by design searches for output in the direction of the observations, up to the discrepancy variance. By updating or varying the discrepancy variance, we can add patterns, exploring which biases can be fixed. If we rule out all of parameter space using our current discrepancy specification, when we

theoretically do not rule out the basis representation of the observations, this may suggest that there is not yet enough discrepancy included, and should lead to $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ being updated to incorporate this new belief.

This could be used in combination with an interactive tool that would allow modellers to give any beliefs about structural errors. Patterns could be highlighted, and given these patterns, the rotation algorithm applied to select a suitable basis. Given this rotated basis, the multiple required for this structure of the discrepancy can be calculated (if $\mathcal{R}_{\boldsymbol{\Sigma}_{\boldsymbol{\eta}}}(\boldsymbol{\Gamma}_q^*, \mathbf{z}) > T)$), so that the representation of $\mathbf{z}$ would not be ruled out. Since the basis changes every time the discrepancy does, new emulators would be required prior to history matching, so that this would not be a completely automated procedure for specifying the discrepancy variance. However, given new emulators based on these beliefs about the discrepancy, history matching can be performed to inform the modellers whether or not the defined pattern is likely to be a structural error in the model: if all space is ruled out, it is likely that not enough discrepancy has been specified yet. If there is a non-empty NROY space, then while too much discrepancy may have been assumed for certain regions, biases in other regions of the output may have been improved.

Appendices

A. Toy function definitions

This appendix gives the definitions of the idealised examples (that we commonly refer to as toy functions) used to illustrate methodology throughout this thesis.

A.1. 1-dimensional toy functions

The following four functions are those used in the comparison of regression and Gaussian process emulators in Chapter 3. $\Psi(0, \sigma^2)$ denotes a random draw from a Normal distribution with mean 0 and variance σ^2. The parameters for the first three functions take on values in the interval $[-1, 1]$. The parameters for the borehole function, as defined in (A.4), have different ranges, which are scaled to $[-1, 1]$ for emulation.

Function 1 (10 input parameters):

$$f_1(\mathbf{x}) = 7(x_1 - 1) + 10(x_1 + 1)x_2(x_3 + 0.5)^2 + 5x_4^2 + 5exp(x_5(x_6 + 0.5)) + x_7^2 x_8^3 + 0.5x_9 x_{10} + 0.5x_7 x_{10}^2 + 2x_5 x_8^2 + \Psi(0, 0.05^2) \tag{A.1}$$

Function 2 (10 input parameters):

$$\begin{aligned} f_2(\mathbf{x}) = {} & sin(x_1 x_2) + cos(x_3)sin(x_4)cos(x_5 + x_6) + sin(x_1 x_7)cos(x_7 x_8)exp(x_9 + x_3) \\ & + sin(cos(x_{10} + x_5 + x_8))cos(x_9^2 x_3) + sin(x_7) + cos(sin(x_5 + x_9)) \\ & + exp(sin(x_2 sin(x_{10}))) + x_7^2 cos(x_1)cos(x_3) + exp(x_2^2) + cos(x_1 - x_6) + \Psi(0, 0.15^2) \end{aligned} \tag{A.2}$$

Function 3 (20 input parameters):

$$f_3(\mathbf{x}) = 5x_1(x_2(x_3+0.5)^2x_4 + 5exp(x_3(x_6+0.5)) + x_7^2x_8^3 + 1.5x_9x_{10}) + x_5^2 + 6(x_{11}+x_{12}^3) + 0.5(x_{12} - (x_{13}x_{14})) + x_5exp(x_{15}) - 10exp(x_{16})(x_{17}+x_{18}+x_{19}^2+x_{20}^3) + \Psi(0, 0.5^2) \tag{A.3}$$

The borehole function (8 input parameters):

$$f_4(T_u, H_u, H_l, r, r_w, L, K_w, T_l) = \frac{2\pi T_u(H_u - H_l)}{ln(r/r_w)(1 + \frac{2LT_u}{ln(r/r_w)r_w^2K_w} + \frac{T_u}{T_l})} \tag{A.4}$$

A.2. Spatial toy function

The spatial toy function that was introduced in Chapter 4, giving output over a 10×10 field, with 6 input parameters each taking values in $[-1, 1]$, is defined as

$$f(\mathbf{x}) = 3(10x_2^2\boldsymbol{\varphi}_2 + 5x_3^2\boldsymbol{\varphi}_2 + (x_3 + 1.5x_1x_2)\boldsymbol{\varphi}_3 + 2x_2\boldsymbol{\varphi}_4 + x_3x_1\boldsymbol{\varphi}_5 + (x_2x_1)\boldsymbol{\varphi}_6 + x_2^3\boldsymbol{\varphi}_7 + (x_2+x_3)^2\boldsymbol{\varphi}_8 + 2) + 1.5\pi_N(x_4, 0.2, 0.1^2)\boldsymbol{\varphi}_1\frac{x_5}{1.3+x_6} + \Psi_{10\times10}(0, 0.05^2) \tag{A.5}$$

where $\pi_N(x_4, 0.2, 0.1^2)$ denotes the density function of the Normal distribution with mean 0.2 and variance 0.1^2, and $\Psi_{10\times10}(0, 0.05^2)$ denotes sampling from a Normal distribution with mean zero and variance 0.05^2, independently for each box in a 10×10 grid. The basis vectors used in this definition, $(\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_8)$, are shown in Figure A.1. $\boldsymbol{\varphi}_1$ represents the pattern that is most similar to the observations $\mathbf{z}$, and $\boldsymbol{\varphi}_2$ represents the biased version of the observations, most prevalent in sampled ensembles.

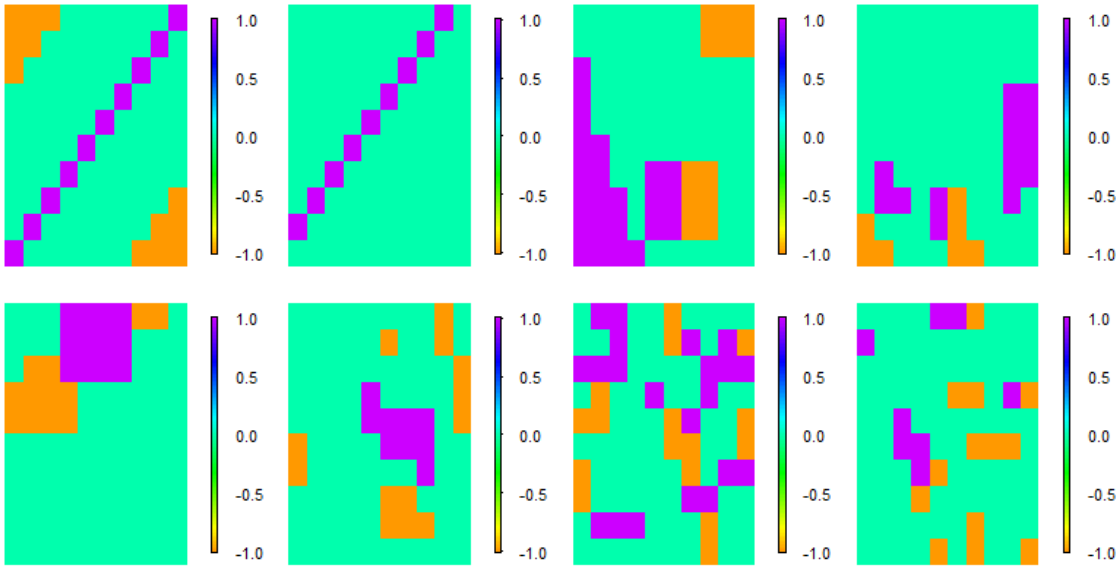

Figure A.1. The 8 orthogonal basis vectors used in the definition of the spatial toy function, with φ_1 the top left plot, φ_2 to the right of this, etc.

B. Emulator diagnostics

As described in the procedure followed for fitting emulators, we perform two main diagnostic checks when assessing the fit of emulators:

1. Leave-one-out cross-validation on the training data.

2. Predicting the output for runs in the validation data, using the emulator fit to the training data.

Before emulators could be used for prediction, and hence calibration or history matching, both of these checks needed to be passed satisfactorily, with the majority of points required to be within a 95% prediction interval in each case. When an emulator passed both of these tests, the points in the validation data were added to the final emulator.

Leave-one-out cross-validation plots are provided here for all runs in given ensembles (i.e. the training and validation data). Each plot gives the emulated values on the x-axis, with the true function output on the y-axis. The error bars give 99% prediction intervals around the mean value. The green and red dots indicate the true function values, with points coloured green if these are within the 99% prediction interval.

B.1. Univariate toy functions

Validation plots for the wave 1 and wave 4 emulators for $f_1(\cdot)$ were given in Figure 3.4. For the remaining functions in Chapter 3, we provide cross-validation plots for the wave 1 and 4 Gaussian process emulators, for the case when a Gaussian process is always fitted.

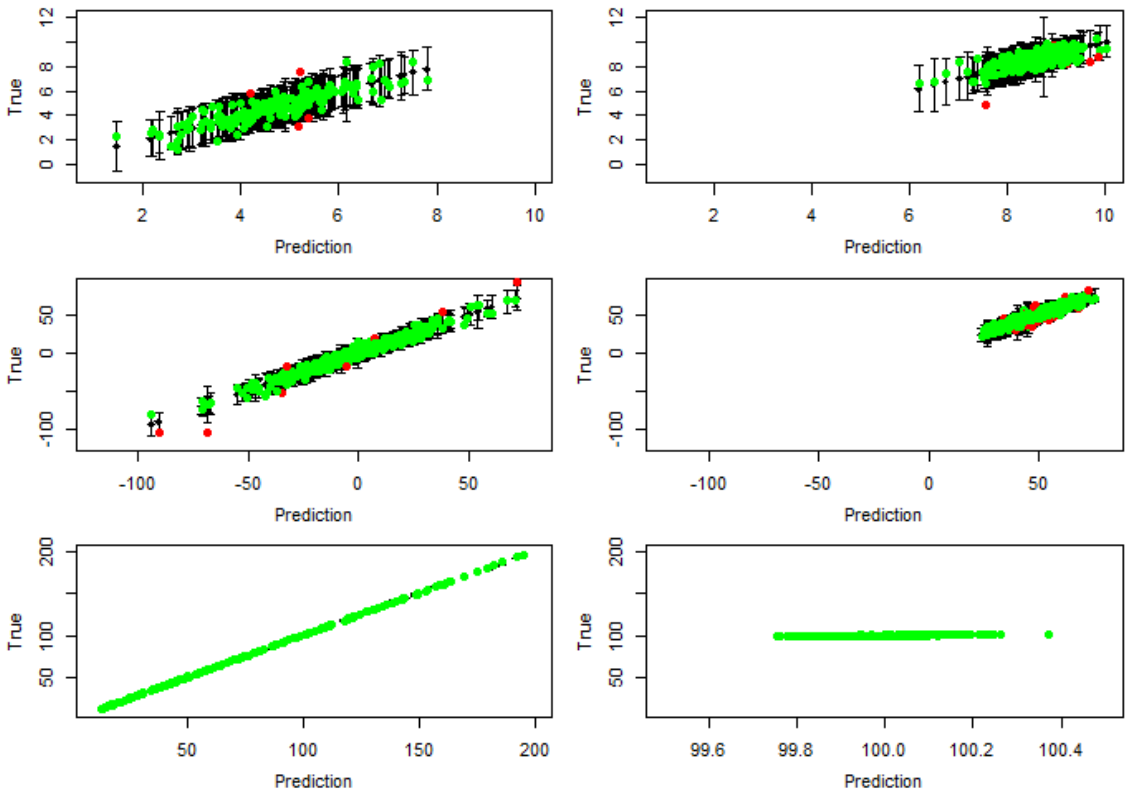

Figure B.1. Leave-one-out cross-validation plots for the toy functions with scalar output. Top: wave 1 and wave 4 Gaussian process emulators for $f_2(\cdot)$. Middle: wave 1 and wave 3 Gaussian processes for $f_3(\cdot)$. Bottom: wave 1 and wave 4 Gaussian processes for the borehole function. The predicted values are plotted on the x-axis, along with error bars. The true values are coloured green if they lie within the 99% error bars, and red otherwise.

B.2. IC fault model

For history matching the IC fault model in Section 3.6, we fitted four waves of regression and Gaussian process emulators for three different outputs, for various combinations of the two emulator types. Figure B.2 provides cross-validation plots for the wave 1 and wave 4 emulators for each output, from the always Gaussian process case. As expected given the calibration results, we see that emulation has been more difficult for o_{36}. The emulators for the other two outputs have clearly improved by the final wave.

B.3. Spatial toy function

For the spatial toy function introduced in Section 4.2, we fitted emulators for the coefficients on several different bases. In this appendix, we provide cross-validation plots for the emulators for the SVD basis (from Section 4.4, shown in Figure B.3), the emulators for when $\mathbf{z}$ was used in the physical pattern $\mathbf{B}_p$ in Section 4.7 (Figure B.4), and the waves

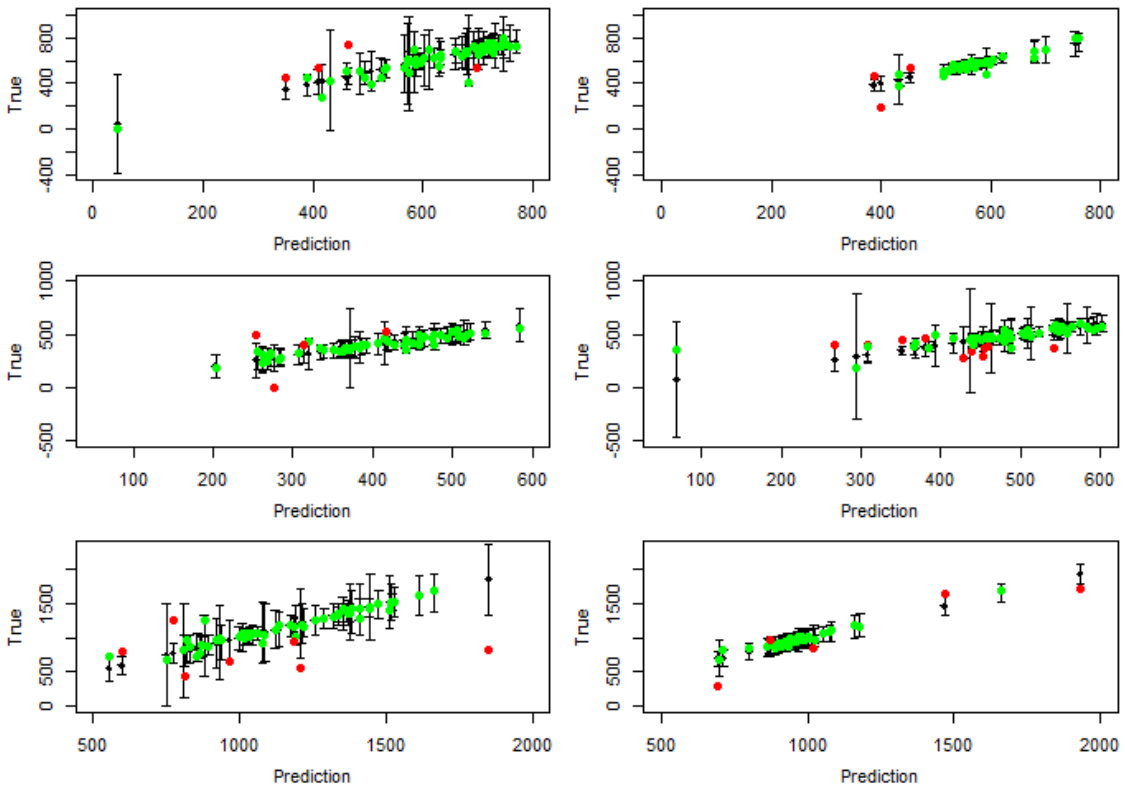

Figure B.2. Leave-one-out cross-validation plots for the IC fault model Gaussian process emulators, at wave 1 (left) and wave 4 (right). The top panel is for o_{24}, the middle is for o_{36}, and the bottom is for w_{36}.

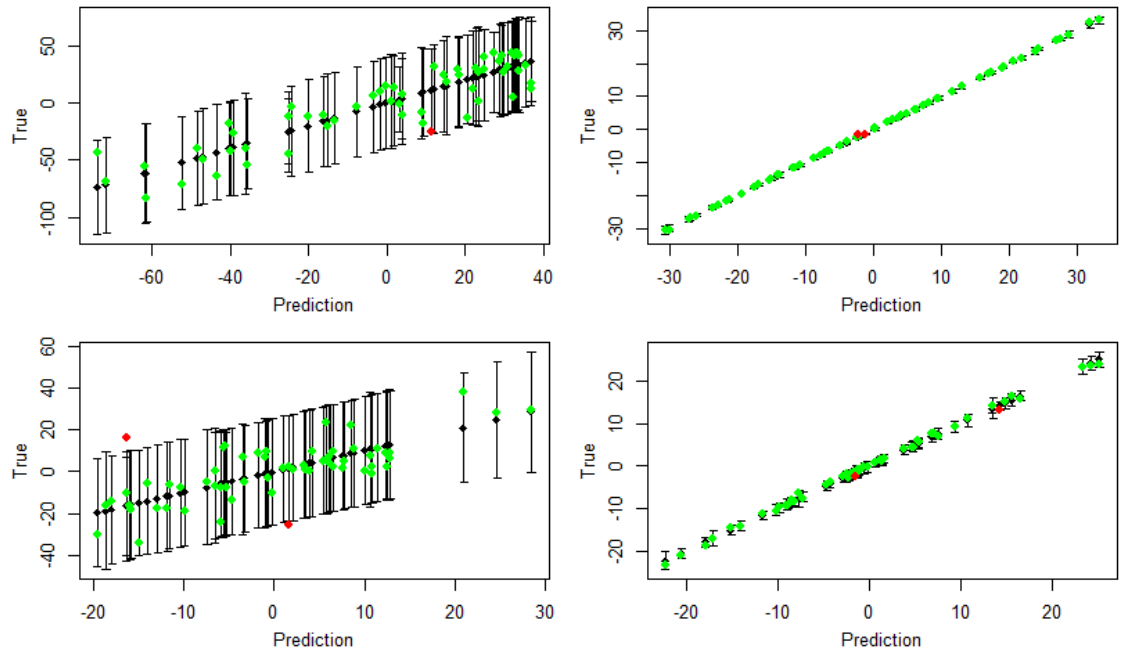

Figure B.3. Leave-one-out cross-validation plots for the emulators for the coefficients on the first four SVD basis vectors for the spatial toy function, with error bars showing 99% prediction intervals.

1 (Section 5.5, Figure B.5) and 2 rotated bases (Section 5.6, Figure B.6), and the wave 3 SVD basis (Section 5.8, Figure B.7), as by this wave the ensemble contained enough important signal so that a rotation was not required.

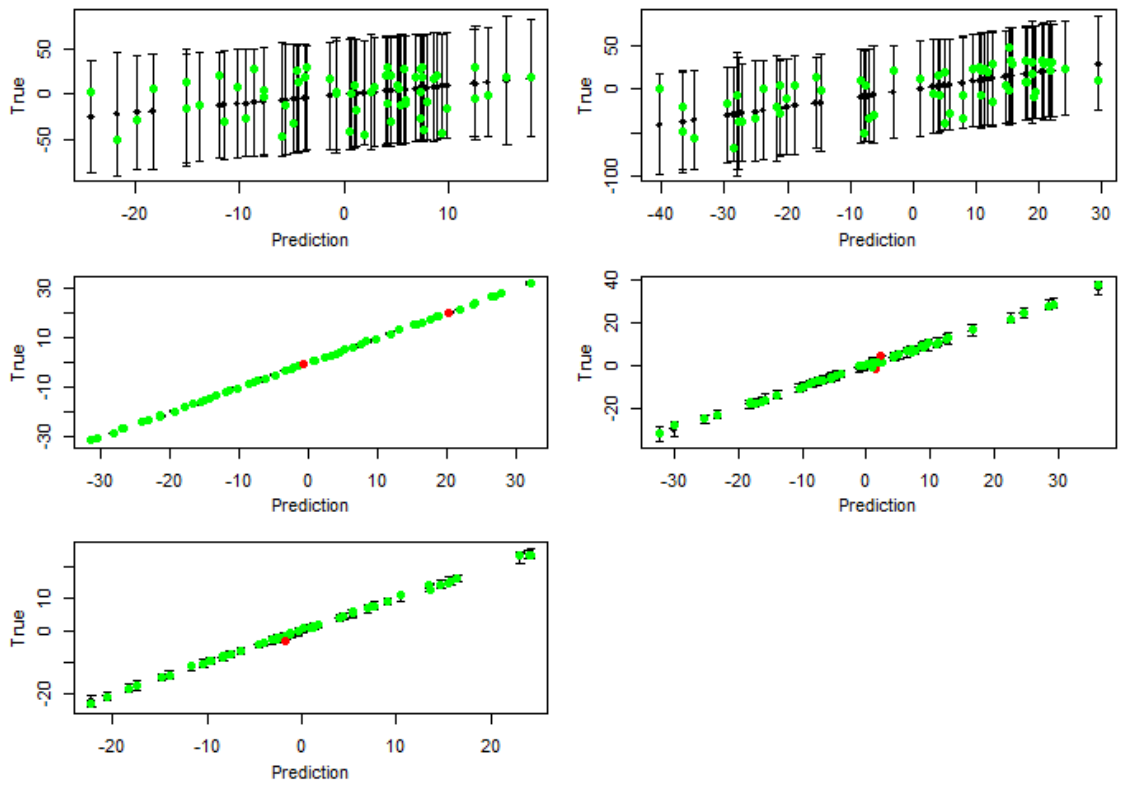

Figure B.4. Leave-one-out cross-validation plots for the emulators for the first five basis vectors when $\mathbf{B}_p$ was set as the observations, with the residual basis used to complete the basis.

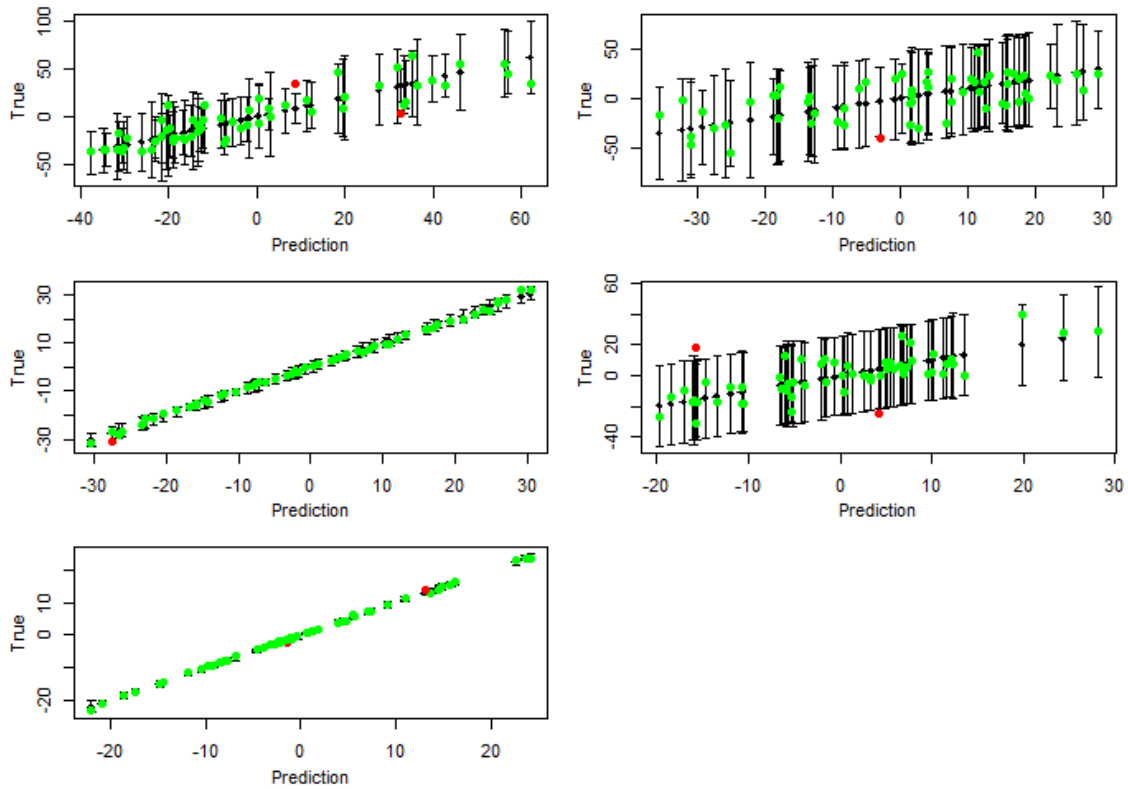

Figure B.5. Leave-one-out cross-validation plots for the emulators for the coefficients on the first five of the rotated basis vectors, for the first wave of emulation of the spatial toy function.

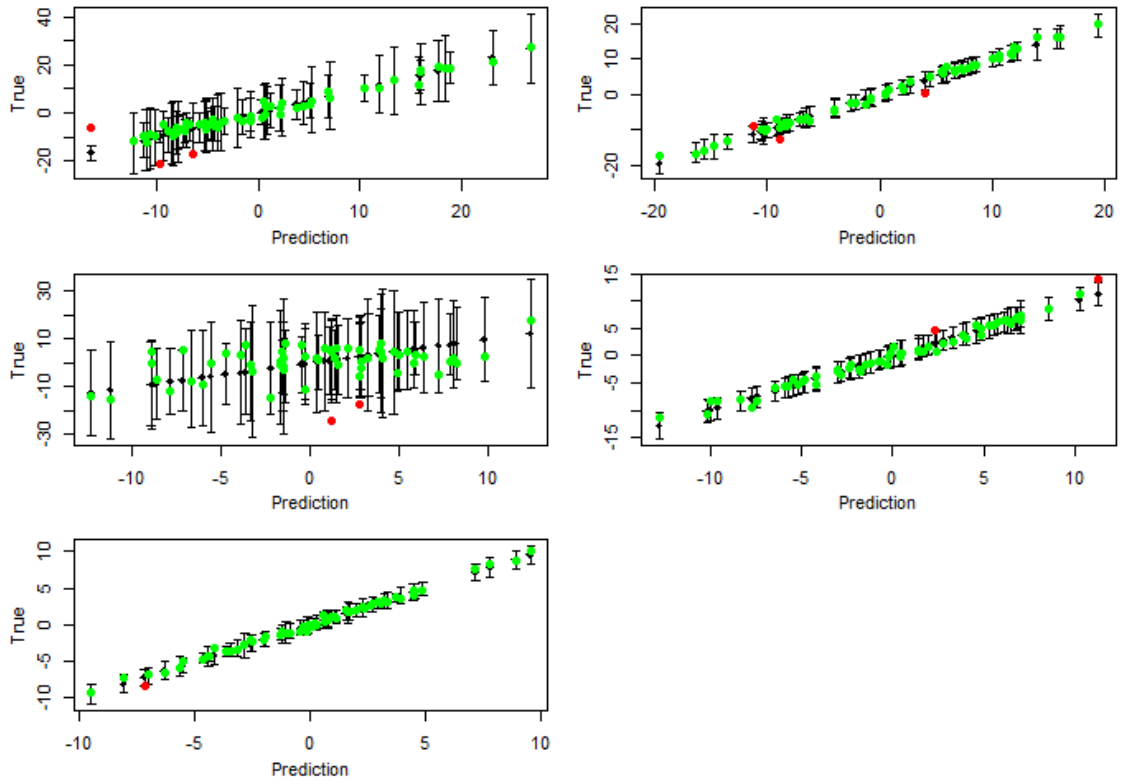

Figure B.6. Leave-one-out cross-validation plots for the emulators for the first five basis vectors of the wave 2 rotated basis.

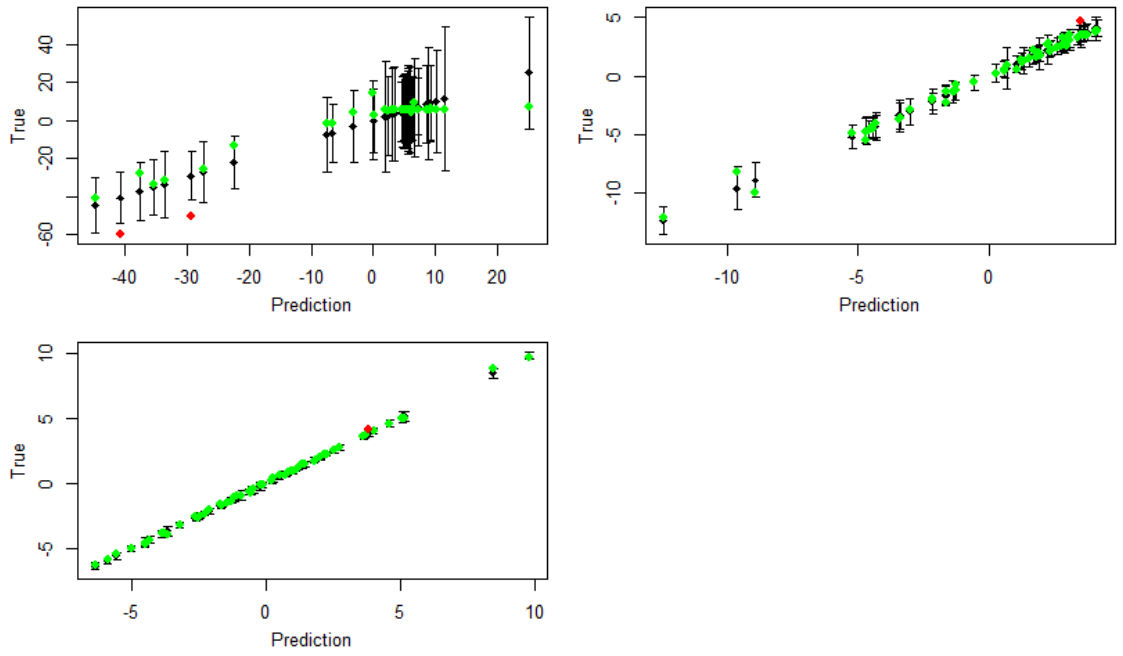

Figure B.7. Leave-one-out cross-validation plots for the emulators for the first three basis vectors of the wave 3 SVD basis (as no rotation was required at this wave).

B.4. CanAM4 emulators

In this section, due to the large number of basis coefficient emulators required for each of CLTO, RTMT and TA at each of the two waves, it is not feasible to show diagnostics for every fitted emulator. We provide cross-validation plots for the first six basis coefficient emulators for each output field at each wave, as after rotation, these are the most important basis vectors for representing the observations. Furthermore, as only one iteration of rotation was often required, the second basis vector onwards is from the residual basis, and hence the first six basis vectors generally explain a large percentage of the ensemble variability.

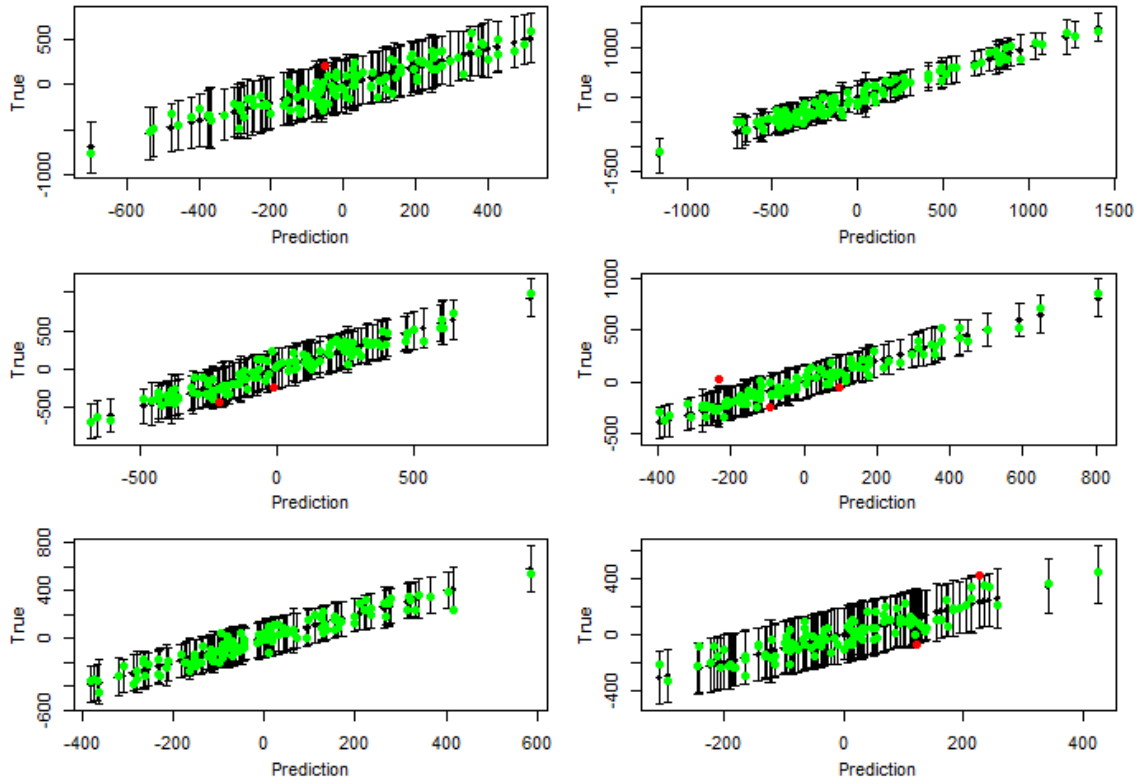

Figure B.8. Leave-one-out cross-validation plots for the emulators for the coefficients given by projection onto the first six basis vectors of the CLTO rotated basis at wave 1.

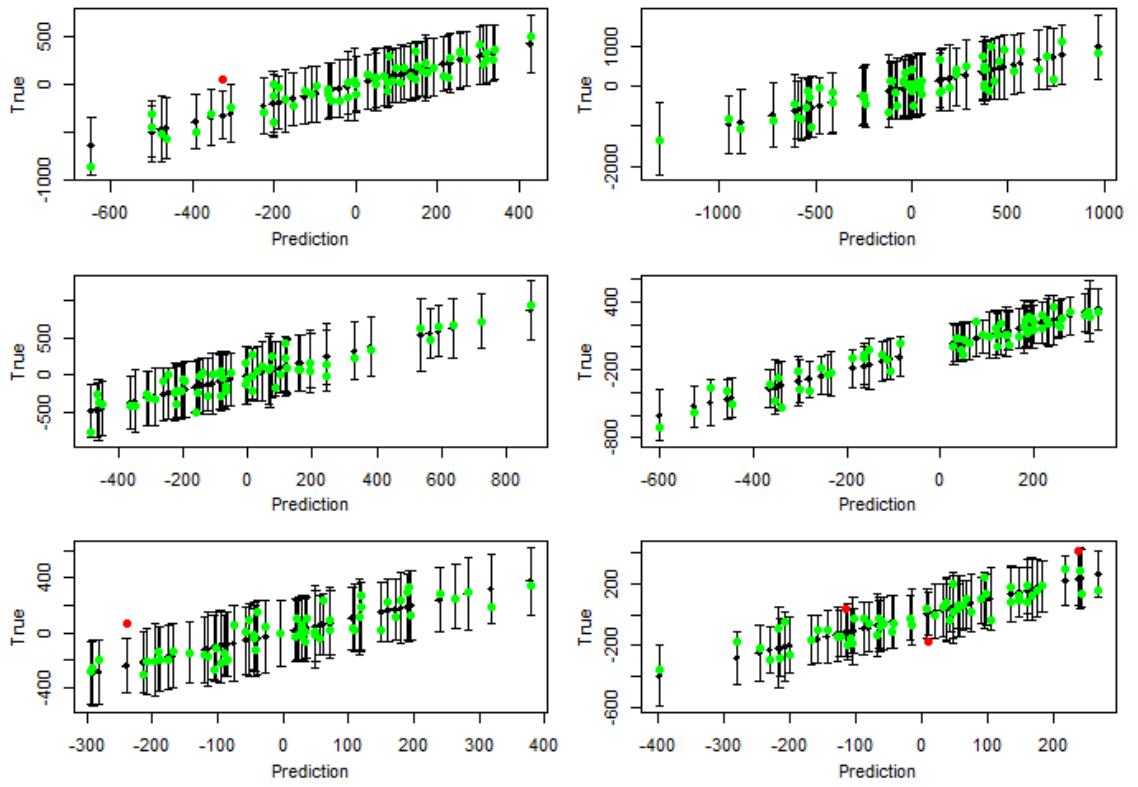

Figure B.9. Leave-one-out cross-validation plots for the emulators for the coefficients on the first six basis vectors of the RTMT rotated basis at wave 1.

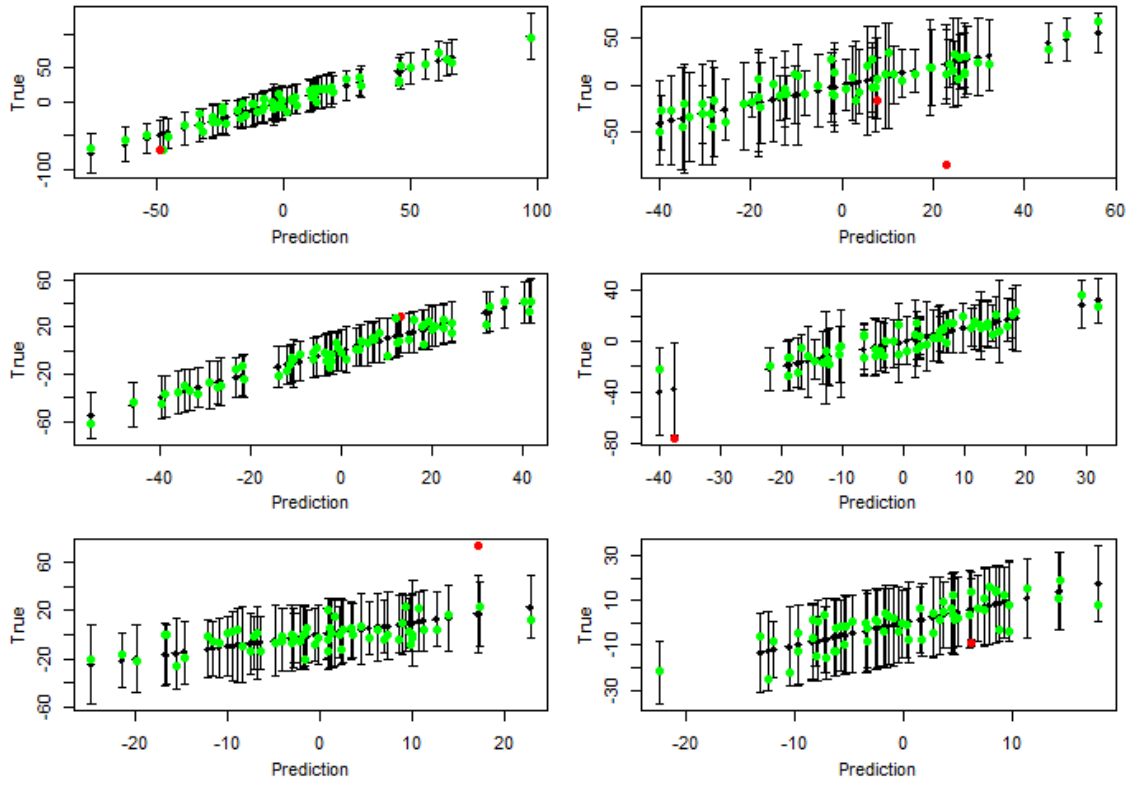

Figure B.10. Leave-one-out cross-validation plots for the emulators for the coefficients on the first six basis vectors of the TA rotated basis at wave 1.

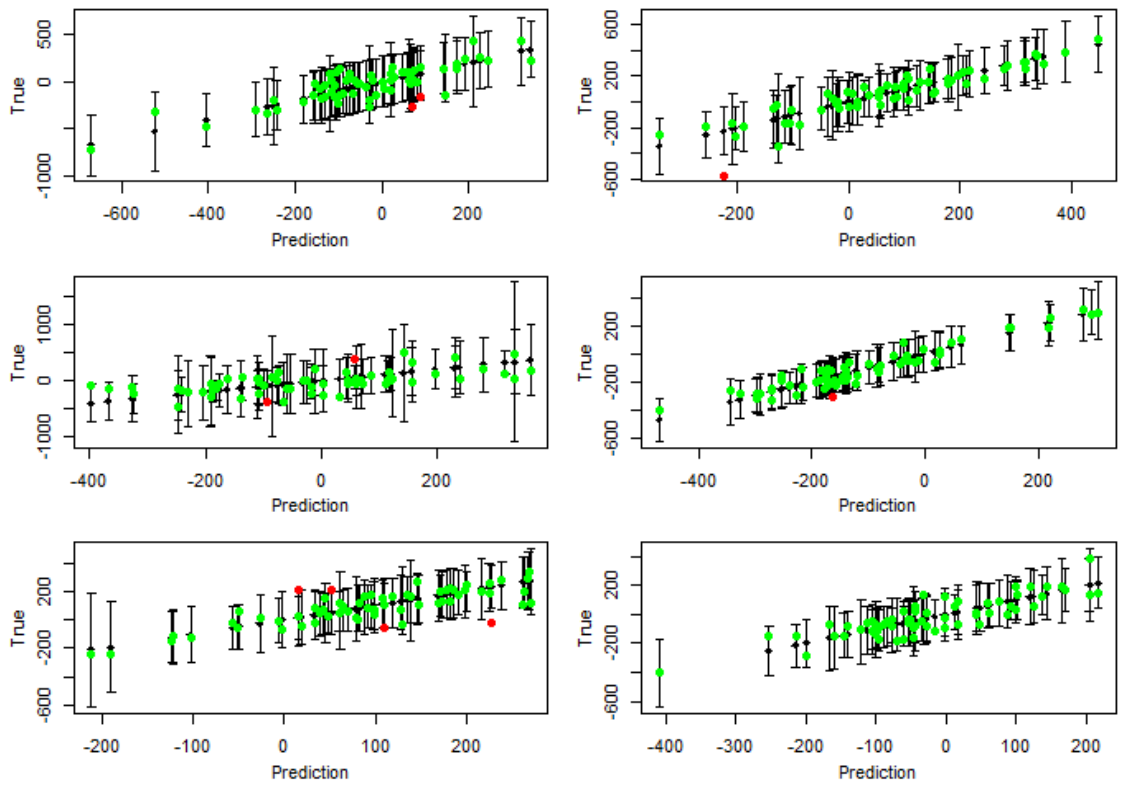

Figure B.11. Leave-one-out cross-validation plots for the emulators for the coefficients on the first six basis vectors of the CLTO rotated basis at wave 2.

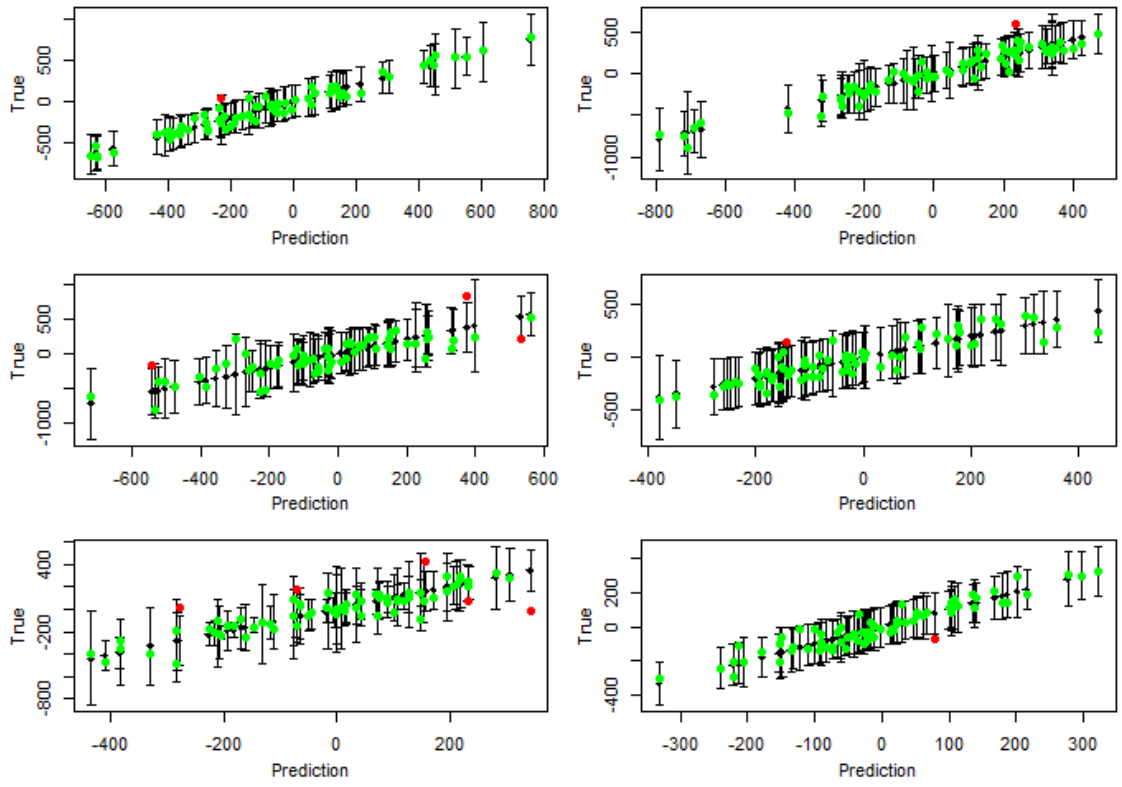

Figure B.12. Leave-one-out cross-validation plots for the emulators for the coefficients on the first six basis vectors of the RTMT rotated basis at wave 2.

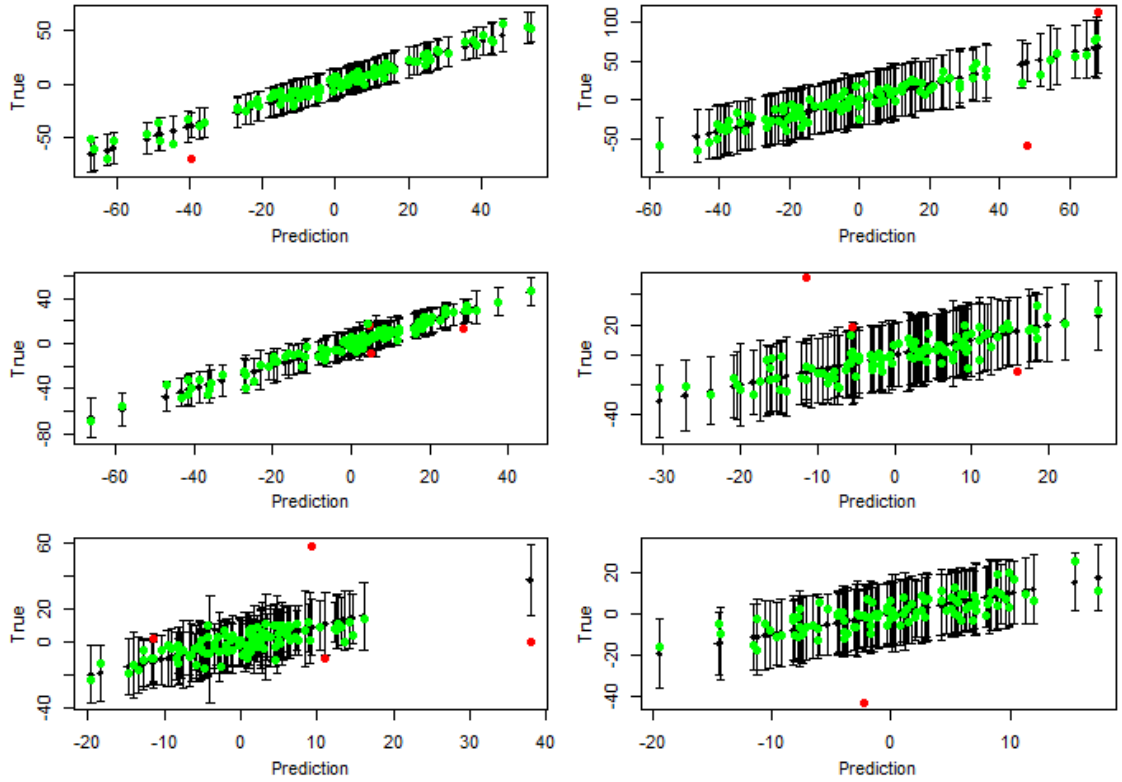

Figure B.13. Leave-one-out cross-validation plots for the emulators for the coefficients on the first six basis vectors of the TA rotated basis at wave 2.

C. Miscellaneous plots

C.1. Calibration traceplots

This section contains some of the traceplots produced when performing calibration for the spatial toy function, demonstrating convergence of the chains.

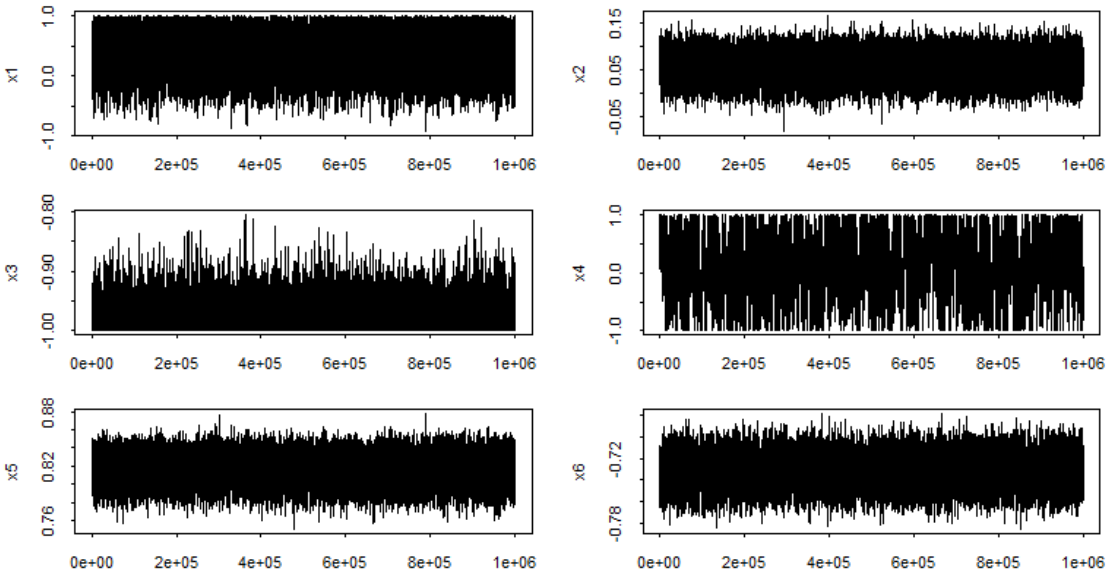

Figure C.1. The converged MCMC chains for the calibration of the toy function with the SVD basis $\mathbf{\Gamma}_4$. The initial parameter value for the MCMC was set to $\mathbf{x}^*$.

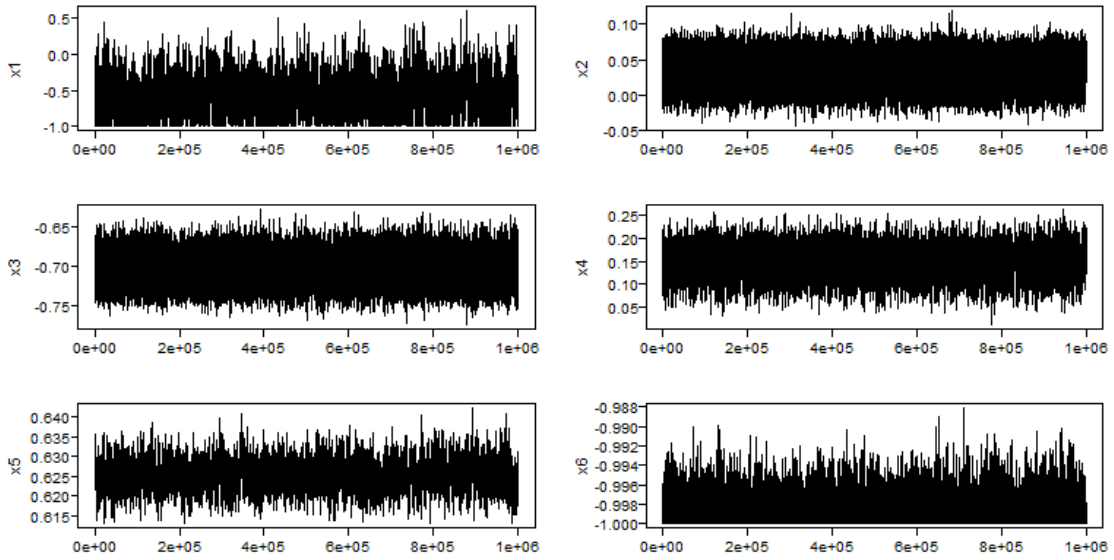

Figure C.2. The converged MCMC chains for the calibration of the toy function using the wave 1 rotated basis. The initial parameter value for the MCMC was set to $\mathbf{x}^*$.

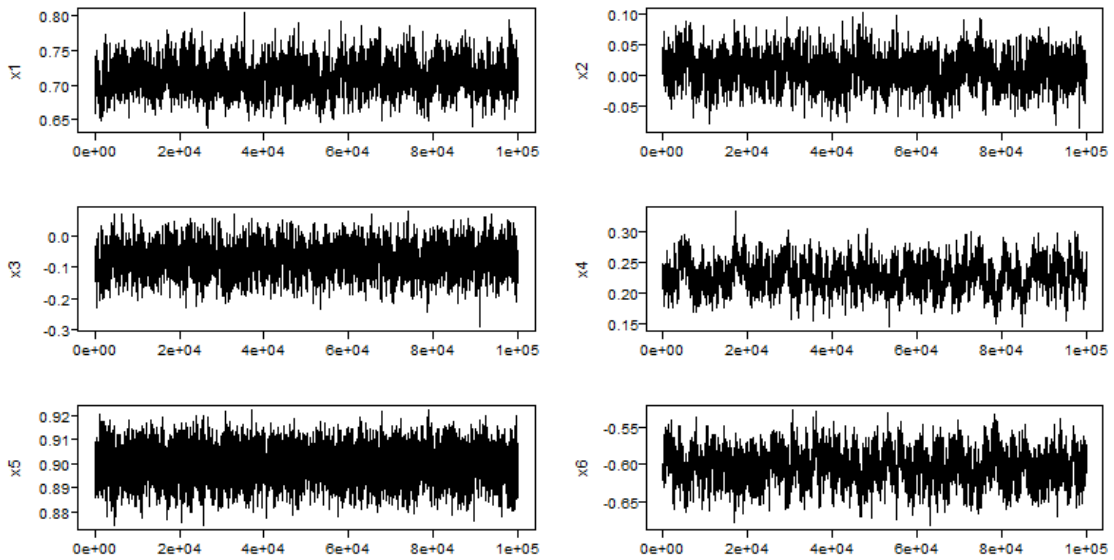

Figure C.3. The converged MCMC chains for the calibration of the toy function using the wave 2 rotated basis and emulators. The initial parameter value for the MCMC was set to $\mathbf{x}^*$.

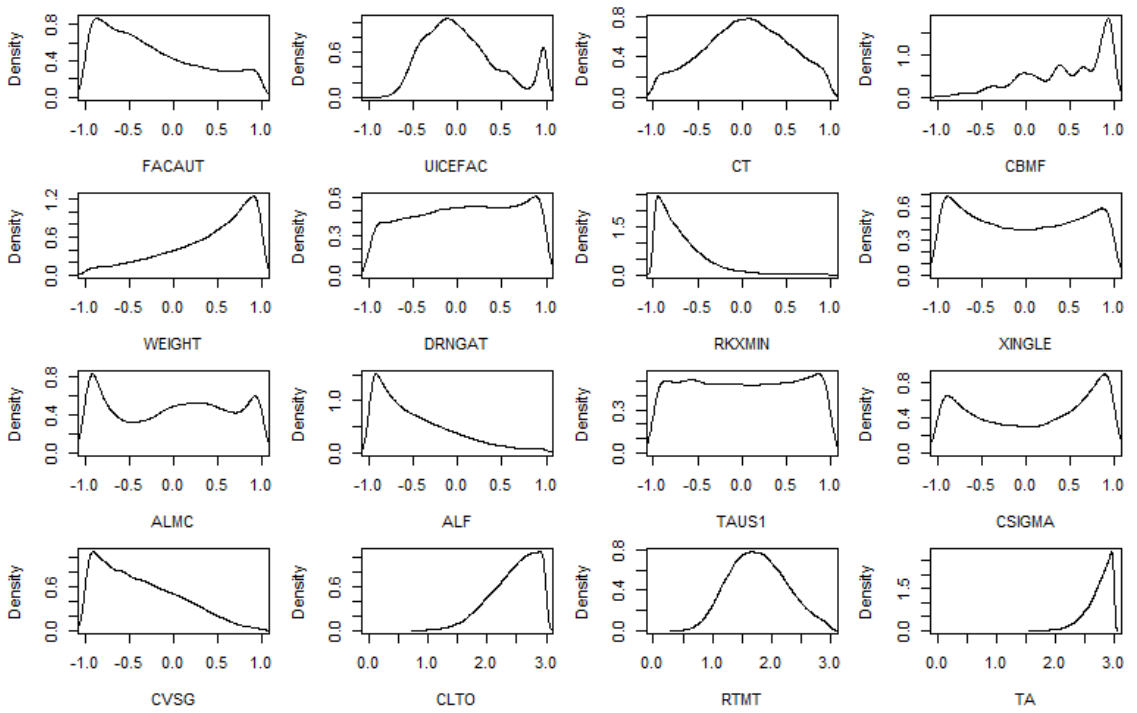

Figure C.4. Densities for the 13 parameters of CanAM4, for runs in the original wave 1 NROY space. The final three panels show the spread of coefficient implausibilities for CLTO, RTMT and TA within this NROY space, scaled so that the maximum implausibility is 3.

C.2. CanAM4 plots

In this section, we provide additional plots relevant to history matching of the CanAM4 model in Chapter 6.

Figure C.4 illustrates the composition of the wave 1 NROY space, as originally defined in Section 6.3, by showing the densities for each individual input parameter, for a sample from this NROY space. Figure C.5 shows a pairs plots for the wave 2 design that was run on CanAM4. Figure C.6 shows the sample of 20 input parameter settings that was used to fit the Bayesian hierarchical model to determine the coefficient bound for TA at wave 2, and the posterior density for this bound. Figure C.7 shows the input parameter densities for the wave 2 NROY space.

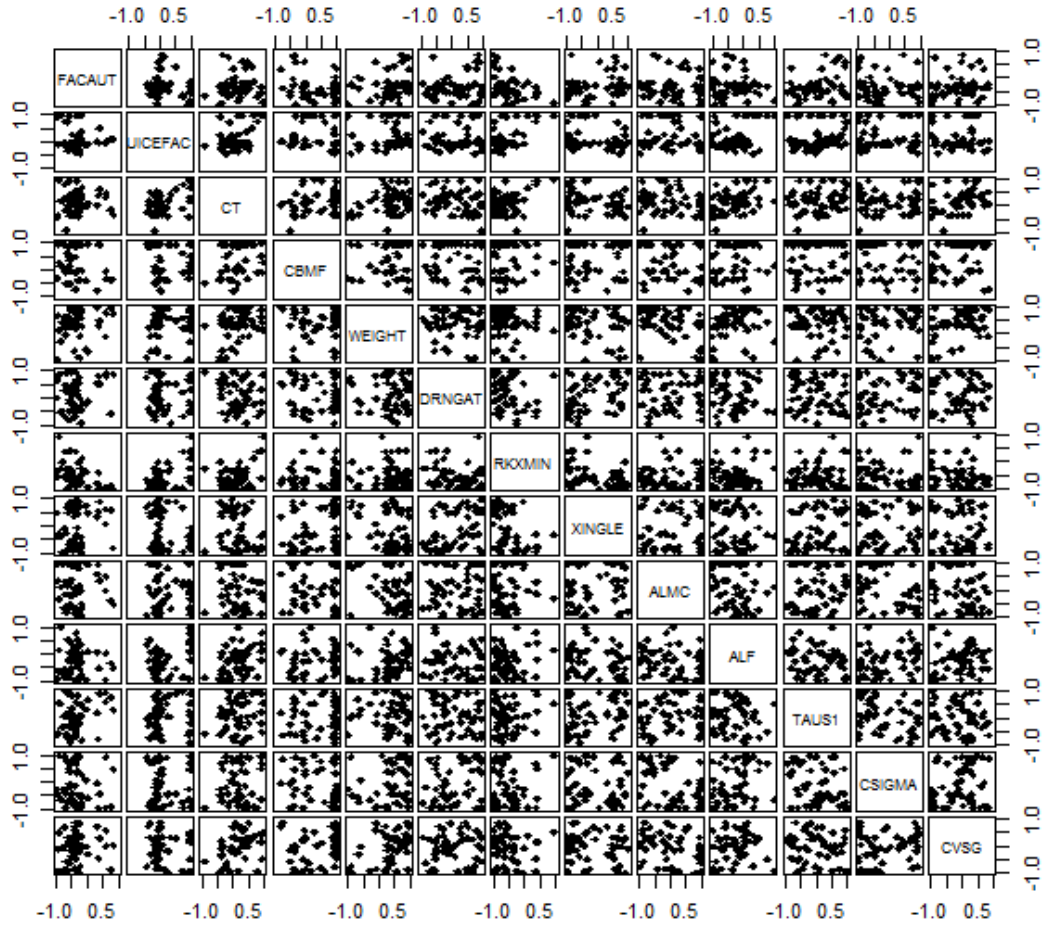

Figure C.5. Pairs plot showing the wave 2 design for CanAM4.

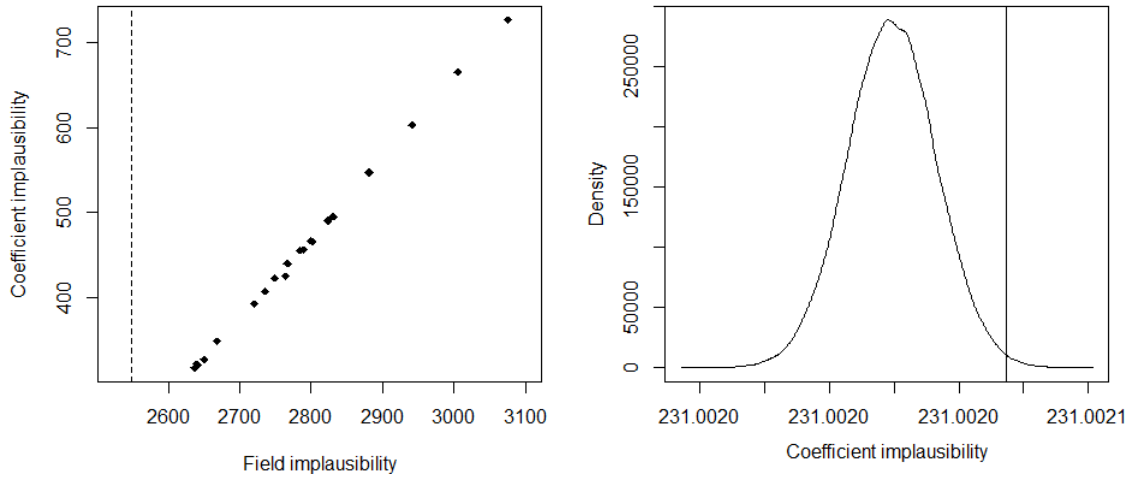

Figure C.6. Left: The coefficient and field implausibilities for 20 sampled parameter values, for the wave 2 TA rotated basis emulators. The dotted line shows the history matching bound for the field implausibility. Right: the posterior density for $\mathcal{I}_c|\mathcal{I}_f = T$, with the vertical line indicating the 99.5% value of this distribution.

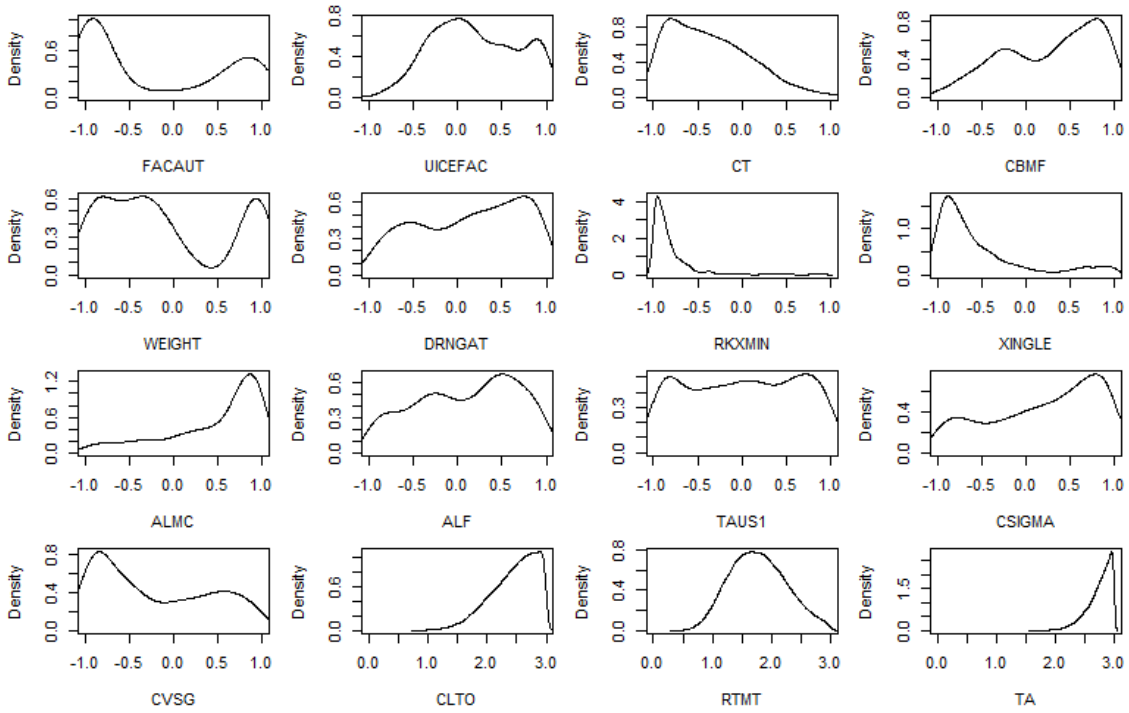

Figure C.7. Densities for the 13 parameters of CanAM4, for runs in the wave 2 NROY space. The final three panels show the spread of coefficient implausibilities for CLTO, RTMT and TA within this NROY space, scaled so that the maximum implausibility is 3.

D. Code for emulation, rotation and spatial calibration

In this appendix, we present some of the R code used in this thesis, with Section D.4 illustrating the application of some of our methods for calibrating and history matching, using the spatial toy function.

D.1. Emulation

In this section, we give the code used to fit emulators to scalar outputs, as in Chapter 3 for the toy functions and IC fault model, and in Chapters 4, 5 and 6 for basis coefficients, for the spatial toy function and CanAM4 output fields.

The function that fits the linear model and Gaussian process components of a scalar emulator, where 'EMULATE' is a function (not provided here) that fits a regression model:

```
FitEmulatorPars <- function(response, training, validation, cands, canfacs = NULL){
  train <- training[, c(cands, canfacs, response)]
  val <- validation[,c(cands, canfacs, response)]
  regmodel <- EMULATE(response, train, tcands = cands, tcanfacs = canfacs)
  whichnoise <- which(names(train)=="Noise")
  train <- train[,-whichnoise]
  val <- val[,-whichnoise]
  active <- regmodel$Names
  if (is.null(regmodel$Factors) == FALSE){active <- c(active, regmodel$Factors)}
  t <- dim(train)[2]
  colnames(train)[t] <- colnames(val)[t] <- "y"
  check <- 0
  count <- 0
  while (check == 0 & count < 2){
    bestpars <- matrix(numeric(10*(length(active)+1)), nrow = 10)
    maxl <- c(rep(0,10))
    for (i in 1:10){
```

```
      opt <- GenSA(NULL, MaxLikelihood, lower = c(rep(0.1,length(active)),0.001),
      upper = c(rep(10,length(active)),0.999), data = train[,c(active, "y")],
      regmodel = regmodel$linModel, control=list(max.call=100))
      bestpars[i,] <- opt$par
      maxl[i] <- opt$value
    }
    bestl <- which.min(maxl)
    opt2 <- GenSA(bestpars[bestl,], CrossValPars, lower =
     c(rep(0.1,length(active)),0.001), upper = c(rep(10,length(active)),0.999),
     data = train[,c(active, "y")], val = val[,c(active, "y")],
      regmodel = regmodel$linModel, control=list(max.call=10))
    if (opt2$value > 99998){
      check <- 0
      count <- count + 1
    }
    else {check <- 1}
  }
  if (count == 2){
    count2 <- 0
    while (opt2$value > 99998 & count2 < 20){
      opt2 <- GenSA(NULL, CrossValPars, lower = c(rep(0.1,length(active)),0.001),
      upper = c(rep(10,length(active)),0.999), data = train[,c(active, "y")],
      val = val[,c(active, "y")], regmodel = regmodel$linModel,
       control=list(max.call=10))
      if (opt2$value > 99998){
        count2 <- count2 + 1
      }
    }
  }
  if (opt2$value > 99998){print("Warning: emulator failed cross-validation or
  prediction test")}
  p <- t - 1
  d.op <- opt2$par[1:length(active)]
  nu.op <- opt2$par[length(opt2$par)]
  temp.em <- em(train[,active],train$y,d.op,nu.op,regmodel$linModel,active=NULL)
  cvem <- cv.em(temp.em,dim(train)[1])
  par(mfrow=c(1,2), mar=c(4,2,2,2))
  cv.plot(cvem)
  val.pred(temp.em, val[,active], val$y)
  fulldata <- rbind(train, val)
  regmodel.full <- lm(update(as.formula(regmodel$linModel), y~.),data=fulldata)
  emulator <- em(fulldata[,active],fulldata[,p+1],d.op,nu.op,regmodel.full,
  active=NULL)
  emulator$x <- fulldata[,1:p]
  emulator$y <- fulldata[,p+1]
  emulator$valid <- ifelse(opt2$value > 99998, 0, 1)
  return(emulator)
}
```

The function that carries out the cross-validation used in the fitting of the Gaussian process correlation lengths, as described in Section 3.3.3:

```
CrossValPars <- function(x, data, val, regmodel, type = "BR"){
  d <- x[-length(x)]
  nu <- x[length(x)]
  p <- length(d)
  n <- dim(data)[1]
  colnames(data)[p+1] <- colnames(val)[p+1] <- "y"
  new.em <- em(data[,1:p],data[,p+1],d,nu,regmodel,type)
  new.cv <- cv.em(new.em, K=n)
  new.outside95 <- sum(abs(data[,p+1]-new.cv$cvmean) >= qt(0.975,new.cv$df)*
  new.cv$cvse)
  check1 <- is.valid(new.outside95, n, 0.05)
  pred <- pred.em(new.em, val[,1:p])
  new.outside95v <- sum((abs(val[,(p+1)]-pred$post.m)>=qt(0.975,new.em$n -
  new.em$q)*sqrt(pred$post.cov)))
  check2 <- is.valid(new.outside95v, dim(val)[1], 0.05)
  output <- mean(new.cv$cvse) + ifelse(check1 == FALSE, 99999, 0) +
   ifelse(check2 == FALSE, 99999, 0)
  return(c(output, mean(new.cv$cvse), check1, check2))
}
```

The functions required for these cross-validation checks:

```
cv.em <- function(em,K){
  require(cvTools)
  data <- data.frame(em$x[,em$active],em$y)
  if (length(em$active) == 1){
    colnames(data)[1] <- em$active
  }
  n <- em$n
  d <- em$d
  nu <- em$nu
  type <- em$type
  if (n == K){
    basis <- attr(terms(em$regmodel),"term.labels")
    if (length(basis)==0) basis = "1"
    cv <- mclapply(1:n, function(i) leave.one.out(i,data,d,nu,basis,type))
    cv <- mclapply(cv,function(x) do.call(cbind,x))
    cv <- do.call(rbind,cv)
    cvmean <- cv[,1]
    cvse <- cv[,2]
    df <- n - length(basis) - 1
    return(list(x=em$x,y=em$y,cvmean=cvmean,cvse=cvse,df=df))
  }
  else {
    folds <- cvFolds(n, K, R = 1, type = "random")
    basis <- attr(terms(em$regmodel),"term.labels")
    if (length(basis)==0) basis = "1"
    cvmean <- c(rep(0,n))
    cvse <- c(rep(0,n))
    for(i in 1:K){
      train <- data[folds$subsets[folds$which != i], ]
      trainx <- train[,-length(train)]
      trainy <- train[,length(train)]
```

```
      val <- data[folds$subsets[folds$which == i], ]
      valx <- val[,-length(val)]
      valy <- val[,length(val)]
      mod <- paste("trainy~", paste(basis,collapse="+"))
      newlm <- lm(as.formula(mod), data = trainx)
      newem <- em(trainx,trainy,d,nu,newlm,type)
      pred <- pred.em(newem,valx,valy)
      cvmean[folds$subsets[folds$which == i]] <- pred$post.m
      cvse[folds$subsets[folds$which == i]] <- sqrt((pred$post.cov))
    }
    df <- summary(newlm)$df[2]
    return(list(x=em$x,y=em$y,cvmean=cvmean,cvse=cvse,folds=folds,df=df))
  }
}

leave.one.out <- function(i,data,d,nu,basis,type){
  p <- dim(data)[2]
  train <- data[-i, ]
  trainx <- train[,1:(p-1)]
  trainy <- train[,p]
  val <- data[i, ]
  valx <- val[,1:(p-1)]
  valy <- val[,p]
  if (p == 2){
    trainx <- as.data.frame(trainx)
    valx <- as.data.frame(valx)
    names(trainx) <- names(valx) <- names(data)[1]
  }
  mod <- paste("trainy~", paste(basis,collapse="+"))
  newlm <- lm(as.formula(mod), data = trainx)
  newem <- em(trainx,trainy,d,nu,newlm,type)
  pred <- pred.em(newem,valx,valy)
  cvmean <- pred$post.m
  cvse <- sqrt((pred$post.cov))
  return(list(cvmean=cvmean,cvse=cvse))
}

MaxLikelihood <- function(x, data, regmodel, type = "BR"){
  d <- x[-length(x)]
  nu <- x[length(x)]
  p <- length(d)
  n <- dim(data)[1]
  colnames(data)[p+1] <- "y"
  new.em <- em(data[,1:p],data[,p+1],d,nu,regmodel,type)
  A <- t(new.em$Q) %*% new.em$Q
  logL <- -0.5*determinant(A, logarithm = TRUE)$modulus -0.5*
  determinant(crossprod(new.em$h.c, new.em$h.c), logarithm=TRUE)$modulus -
  ((new.em$n - new.em$q)/2)*log(new.em$sigma2)
  output <- -logL
  return(output)
}

is.valid <- function(outside, n, tolerance = 0.05, level = 0.05){
```

```
  ind <- TRUE
  check1 <- 1 - pbinom(outside-1, n, level)
  if (check1 < tolerance){ind <- FALSE}
  check2 <- pbinom(outside, n, level)
  if (check2 < tolerance){ind <- FALSE}
  return(ind)
}
```

Plotting the validation checks:

```
cv.plot <- function(cvem,level=0.95){
  library(sfsmisc)
  mean <- cvem$cvmean
  se <- cvem$cvse
  y <- cvem$y
  df <- cvem$df
  z <- level + (1-level)/2
  upp <- max(c(mean+qt(z,df)*se,y))
  low <- min(c(mean-qt(z,df)*se,y))
  errbar(mean,mean,mean + qt(z,df)*se, mean - qt(z,df)*se,pch=18,
         main = "", xlab = "Predicted", ylab="True", ylim = c(low,upp))
  points(mean,y,pch=19,col = ifelse(abs(y-mean)<=qt(z,df)*se,"green","red"))
}

val.pred <- function(model, valdata, valobs, level=0.95){
  if (is.null(model$coefficients) == F){
    pred <- predict(model,valdata,interval="prediction",level=level)
    upp <- max(c(pred[,3],valobs))
    low <- min(c(pred[,2],valobs))
    errbar(pred[,1],pred[,1],pred[,3],pred[,2],cap = 0.015,pch=20,ylim=c(low,upp),
           main="",xlab = "Prediction",ylab="True")
    points(pred[,1],valobs,pch=19,col = ifelse(valobs>pred[,3] | valobs<pred[,2],
    "red","green"))}
  else {
    pred <- pred.em(model,valdata)
    mean <- pred$post.m
    se <- sqrt(pred$post.cov)
    z <- level + (1-level)/2
    df <- model$n - model$q
    upp <- max(c(mean+qt(z,df)*se,valobs))
    low <- min(c(mean-qt(z,df)*se,valobs))
    errbar(mean,mean,mean + qt(z,df)*se, mean - qt(z,df)*se,pch=18, xlab=
    "Prediction",ylab="True", ylim = c(low,upp))
    points(mean,valobs,pch=19,col = ifelse(abs(valobs-mean)<=qt(z,df)*se,
    "green","red"))
  }
}
```

Fitting emulators to multiple sets of coefficients:

```
BasisEmulators <- function(tData, NumberOfEms){
  lastCand <- which(names(tData)=="Noise")
```

```
  n <- dim(tData)[1]
  n1 <- 3*ceiling(n/4)
  samp <- sample(1:n, n)
  lapply(1:NumberOfEms, function(k) FitEmulatorPars(response=
  names(tData)[lastCand+k], training=tData[samp[1:n1],], validation=
  tData[samp[-(1:n1)],], cands=names(tData)[1:lastCand], canfacs=NULL))
}
```

The function that fits an emulator, given a set of correlation lengths and nugget parameter:

```
em <- function(x, y, d, nu, regmodel, type = "BR", active = NULL, ...){
  require(MASS)
  if (length(d) == 1){
    x <- as.data.frame(x)
    colnames(x)[1] <- names(d)
  }
  names(d) <- names(x)
  if (is.null(active) == TRUE){ ### active = NULL means that all variables
  in x are active
    x.ac <- x
    active <- names(x)
  }
  else {
    x.ac <- x[,active]
    d <- d[active]
  }
  if (length(active) == 1){
    x.ac <- as.data.frame(x.ac)
    colnames(x.ac) <- active
  }
  if (type == "BR"){
    if (nu == 1){
      pred = predict(regmodel, x.ac)
      error = y - pred
      H = model.matrix(regmodel, x.ac)
      n = dim(x.ac)[1]
      q = dim(H)[2]
      sigma2 = as.numeric(((n-q-2)^(-1))*(t(error)%*%error))
      return(list(x=x,y=y,regmodel=regmodel,nu=nu,error=error,H=H,n=n,q=q,
      sigma2=sigma2,type=type,active=active))
    }
    else {
      A = calcA(x.ac,d,nu,...)
      data <- as.data.frame(cbind(x.ac,y))
      if (is.vector(x) == TRUE){
        tempName <- names(summary(regmodel)$aliased)[2]
        names(data) <- c(tempName, "y")
      }
      glsmodel = lm.gls(update.formula(formula(regmodel),y~.),data,W=A,inverse=TRUE)
      pred = pred.gls(glsmodel, x.ac)
      error = y - pred
      Q = chol(A)
      H = model.matrix(regmodel, x.ac)
```

```
      n = dim(x.ac)[1]
      q = dim(H)[2]
      x.c = backsolve(Q, error, transpose = TRUE)
      h.c = backsolve(Q, H, transpose = TRUE)
      comp2 = chol2inv(chol(crossprod(h.c, h.c)))
      sigma2 = as.numeric(((n-q-2)^(-1))*crossprod(x.c,x.c))
      return(list(x=x,y=y,regmodel=glsmodel,d=d,nu=nu,Q=Q,error=error,H=H,n=n,
      q=q,x.c=x.c,h.c=h.c,comp2=comp2,sigma2=sigma2,
                  type=type,active=active,...))
    }
  }
  else {
    stop("Type must be 'LS' (Least Squares) or 'BR' (Bayes Regression)")
  }
}

pred.gls <- function(glsmodel,data){
  tt = terms(glsmodel)
  Terms = delete.response(tt)
  mm = model.frame(Terms, data, xlev = glsmodel$xlevels)
  x0 = model.matrix(Terms, mm, contrasts.arg = glsmodel$contrasts)
  prediction = x0 %*% glsmodel$coefficients
  return(prediction)
}
```

Evaluating an emulator at a design of input parameters, returning the expectation and variance for each point:

```
pred.em <- function(em, new, newy,...){
  require(fields)
  x = em$x
  n = em$n
  regmodel = em$regmodel
  nu = em$nu
  error = em$error
  H = em$H
  q = em$q
  sigma2 = em$sigma2
  type <- em$type
  active <- em$active
  if (length(active) == 1){
    x.ac <- as.data.frame(x[,active])
    names(x.ac) <- active
  }
  else {
    x.ac <- x[,active]
  }
  if (is.null(dim(new)) == T){
    new <- new
    new.ac <- as.data.frame(new)
    colnames(new.ac) <- active
  }
```

```
  else if ((dim(new)[1]*dim(new)[2]) == 1){
    new <- new
    new.ac <- as.data.frame(new)
    colnames(new.ac) <- active
  }
  else if ((dim(new)[1]*dim(new)[2]) == dim(new)[2] & length(active) == 1){
    new.ac <- as.data.frame(new[,active])
    names(new.ac) <- active
  }
  else {
    new = new[,names(x)]
    new.ac <- as.data.frame(new[,active])
    if (length(active) == 1){
      names(new.ac) <- active
    }
  }
  if (type == "BR"){
    if (nu == 1){
      postm = predict(regmodel,new.ac)
      tt = terms(regmodel)
      Terms = delete.response(tt)
      mm = model.frame(Terms, new.ac, xlev = regmodel$xlevels)
      hx = model.matrix(Terms, mm, contrasts.arg = regmodel$contrasts)
      postcov = sigma2 * (diag(dim(new.ac)[1]) + (hx)%*%chol2inv(
      chol(t(H)%*%H))%*%t(hx))
    }
    else {
      d = em$d
      tx = calctx(x.ac,new.ac,d,nu,...)
      Q = em$Q
      x.c = em$x.c
      h.c = em$h.c
      comp2 = em$comp2
      y.c = backsolve(Q, tx, transpose = TRUE)
      mean.adj = crossprod(y.c,x.c)
      cov.adj = crossprod(y.c,y.c)
      priormean = pred.gls(regmodel, new.ac)
      priorcov = calcA(new.ac, d, nu,...)
      postm = priormean + mean.adj
      postcov = priorcov - cov.adj
      tt = terms(regmodel)
      Terms = delete.response(tt)
      mm = model.frame(Terms, new.ac, xlev = regmodel$xlevels)
      hx = model.matrix(Terms, mm, contrasts.arg = regmodel$contrasts)
      comp1 = hx - crossprod(y.c, h.c)
      postcov = sigma2 * (postcov + comp1 %*% comp2 %*% t(comp1))
    }
  }
  return(list(post.m = postm, post.cov = diag(postcov)))
}
```

Predicting the output for multiple emulators (e.g. for basis coefficients):

```
MultiPred <- function(ems, design){
  Expectation <- matrix(0, nrow = dim(design)[1], ncol = length(ems))
  Variance <- matrix(0, nrow = dim(design)[1], ncol = length(ems))
  s <- dim(design)[1]/10
  for (i in 1:s){
    EmOutput <- lapply(1:length(ems), function(e) pred.em(ems[[e]],
     design[(10*(i-1) + 1):(10*i),]))
    for (j in 1:length(ems)){
      Expectation[(10*(i-1) + 1):(10*i),j] <- EmOutput[[j]]$post.m
      Variance[(10*(i-1) + 1):(10*i),j] <- EmOutput[[j]]$post.cov
    }
  }
  return(list(Expectation=Expectation, Variance=Variance))
}
```

Calculating the correlation matrices required in Gaussian process emulation:

```
calctx <- function(x,new,d,nu=0,cov = "gauss", d2 = c(rep(2,length(d)))){
  n = dim(x)[1] # number of training points
  m = dim(new)[1]
  if (is.null(m) == T){
    m <- 1
  }
  p = dim(x)[2] # number of parameters
  if (is.null(p) == T){
    p <- 1
  }
  stopifnot(p == length(d))
  if (p == 1 & m == 1){
    new <- new
  }
  else {
    new = new[,names(x)]
  }
  if (p == 1){
    x <- x
    new <- new
  }
  else {
    for (i in 1:p){
      x[,i] = as.numeric(as.character(x[,i]))
    }
    for (i in 1:p){
      new[,i] = as.numeric(as.character(new[,i]))
    }
  }
  if (cov == "gauss") {
    leng = as.matrix(rdist(scale(x,center=FALSE,scale=d),scale(new,
    center=FALSE,scale=d)))
    tx = (1 - nu)*exp(-(leng^2))
  }
  return(tx)
}
```

```
calcA <- function(x,d,nu,cov = "gauss",d2 = c(rep(2,length(d)))){
  n = dim(x)[1] # number of training points
  p = dim(x)[2] # number of parameters
  if (is.null(n) == TRUE){
    n <- length(x)
  }
  if (is.null(p) == TRUE){
    p <- 1
  }
  stopifnot(p == length(d))
  if (p == 1){
    x <- x
  }
  else {
    for (i in 1:p){
      x[,i] = as.numeric(as.character(x[,i]))
    }
  }
  if (cov == "gauss") {
    leng = as.matrix(dist(scale(x,center=FALSE,scale=d),method="euclidean",
    diag=TRUE,upper=TRUE))
    A = nu*diag(n) + (1 - nu)*exp(-(leng^2))
  }
  return(A)
}
```

D.2. Basis projection and rotation

In this section, we provide the code for applying the optimal rotation algorithm (Section 5.4).

Projecting spatial data onto a basis:

```
CalcScores <- function(data, basis){
  d <- dim(data)[2]
  if (is.null(d) == TRUE){
    d <- 1
  }
  p <- dim(basis)[2]
  n <- dim(basis)[1]
  if (is.null(p) == TRUE){
    p <- 1
  }
  if (d == 1){
    data <- as.vector(data)
  }
  V <- t(basis) %*% basis
  Q <- chol(V)
```

```
  y <- backsolve(Q, diag(p), transpose = TRUE)
  x <- backsolve(Q, t(basis) %*% data, transpose = TRUE)
  scores <- crossprod(y, x)
  return(t(scores))
}
```

Projecting a variance matrix onto an orthogonal basis:

```
VarProj <- function(mat, basis){
  proj <- t(basis) %*% mat %*% basis
  return(proj)
}
```

Reconstructing a field from a set of coefficients on a basis:

```
Recon <- function(scores, basis){
  if (is.null(dim(basis)[2]) == TRUE){
    recons <- basis*as.numeric(scores)
  }
  else {
    recons <- basis%*%as.numeric(scores)
  }
  return(recons)
}
```

Projecting the observations (or any general field) onto a basis, and reconstructing the field from this basis representation:

```
ReconObs <- function(obs, basis){
  nb <- is.null(dim(basis))
  if(!nb)
    basis1 <- basis[,1]
  else
    basis1 <- basis
  obs <- c(obs)
  mask <- which(is.na(obs-basis1))
  if(length(mask)>0){
    recons <- rep(NA, length(obs))
    obs <- obs[-mask]
    if(nb)
      basis <- basis[-mask]
    else
      basis <- basis[-mask,]
    proj <- CalcScores(obs, basis)
    recons.partial <- Recon(proj, basis)
    recons[-mask] <- recons.partial
  }
  else{
    proj <- CalcScores(obs, basis)
    recons <- Recon(proj, basis)
  }
```

```
  return(recons)
}
```

Calculating $V(\mathbf{B}, \mathbf{F})$, the proportion of ensemble variability that is explained by projection onto a basis $\mathbf{B}$:

```
VarExplained <- function(basis, data){
  n <- dim(data)[1]
  ens.size <- dim(data)[2]
  recon <- basis %*% t(CalcScores(data, basis))
  explained <- crossprod(c(recon))/crossprod(c(data))
  return(explained)
}
```

Calculating the reconstruction error $\mathcal{R}_{\mathbf{W}}(\mathbf{B}, \mathbf{z})$ given by projection and reconstruction with a basis:

```
WeightedMSE <- function(obs, recon, weight=NULL){
  A <- c(obs) - recon
  mask <- which(is.na(A))
  if(length(mask)>0){
    A <- A[-mask]
  }
  if (is.null(weight) == FALSE) {diagmat <- all(weight[lower.tri(weight)] == 0,
   weight[upper.tri(weight)] == 0)}
  if (is.null(weight) == TRUE){diagmat <- FALSE}
  if (diagmat == FALSE){
    if (is.null(weight) == TRUE){
      wmse <- crossprod(A)/(length(c(obs)) - length(mask))
    }
    else {
      if(length(mask)>0){
        warning("Implicit assumption that weight specified on the full field
        even though applying a mask to missing obs/ensemble grid boxes")
        weight <- weight[-mask,-mask]
      }
      Q <- chol(weight)
      y <- backsolve(Q, A, transpose = TRUE)
      wmse <- crossprod(y,y)/(length(c(obs))-length(mask))
    }
  }
  else {
    wmse <- crossprod(A/diag(weight), A)/(length(c(obs))-length(mask))
  }
  return(wmse)
}
```

Creating a VarMSE plot:

```
VarMSEplot <- function(DataBasis, obs, weight=NULL, ylim=NULL, min.line=TRUE,
bound=TRUE,...){
```

```
  p <- dim(DataBasis$tBasis)[2]
  PlotData <- matrix(numeric(p*2), nrow=p)
  PlotData[,1] <- errors(DataBasis$tBasis, obs, weight)*DataBasis$scaling^2
  for (i in 1:p){
    PlotData[i,2] <- VarExplained(DataBasis$tBasis[,1:i], DataBasis$CentredField)
  }
  plot(1:p, PlotData[,1], type="l", col="red",xlab = expression(k),
   ylab=expression(paste("R "[bold(W)], " (", bold(B)[k], ",", bold(z), ")",
    " / l")), ylim=ylim, ...)
  if(min.line)
    abline(h = PlotData[p,1], col="black", lty=1)
  if (bound == TRUE){
    abline(h = qchisq(0.995, length(obs))/length(obs), lty=2)
  }
  par(new = TRUE)
  plot(1:p, PlotData[,2], type="l", axes=FALSE, xlab=NA, ylab=NA, col="blue",
  ylim=c(0,1) ,...)
  axis(side = 4)
  mtext(side = 4, line = 2.5, expression(paste("V(", bold(B)[k], ",", bold(F), ")")),
   las=3)
  return(PlotData)
}

errors <- function(basis, obs, weight=NULL){
  p <- dim(basis)[2]
  err <- numeric(p)
  for (i in 1:p){
    recon <- ReconObs(obs, basis[,1:i])
    err[i] <- WeightedMSE(obs, recon, weight)
    }
  return(err)
}
```

Calculating the residual basis for a given set of basis vectors and an ensemble:

```
ResidBasis <- function(basis.p, data, orthonorm=TRUE){
  if (orthonorm == TRUE){
    basis.p <- orthonormalization(basis.p,basis=FALSE,norm=TRUE)
  }
  n <- dim(data)[1]
  p <- dim(data)[2]
  recons <- matrix(numeric(n*p), nrow=n)
  for (i in 1:p){
    recons[,i] <- ReconObs(data[,i],basis.p)
  }
  resids <- data - recons
  svd.resid <- svd(t(resids))
  new.basis <- cbind(basis.p, svd.resid$v)[,1:p]
  return(new.basis)
}
```

Finding a basis vector by defining linear combinations of the original basis:

```
ReconBasis <- function(weights, basis, q){
  n <- dim(basis)[1]
  p <- dim(basis)[2]
  if (q == 1){
    new.basis <- as.vector(tensor(basis, weights, 2, 1))
  }
  else {
    dim(weights) <- c(p, q)
    new.basis <- tensor(basis, weights, 2, 1)
  }
  return(new.basis)
}
```

Finding an optimal rotation for a basis:

```
RotateIterate <- function(k, DataBasis, obs, weight = NULL, var.threshold = 0.1,
vtot = 0.95, prior = NULL, orthonorm=TRUE,  ...){
  data <- DataBasis$CentredField
  basis <- DataBasis$tBasis
  t <- var.threshold
  n <- dim(basis)[1]
  ens.size <- dim(data)[2]
  obs <- c(obs)
  minMSE <- WeightedMSE(obs, ReconObs(obs, basis), weight)
  if (is.null(prior) == TRUE){
    prior <- c(1:dim(basis)[2])
  }
  mse <- var <- numeric(k)
  x <- NULL
  new.basis <- NULL
  for (i in 1:k){
    p <- dim(basis)[2]
    opt <- GenSA(c(1, rep(0, p-1)), WeightOptim,  lower = rep(-1, p*1),
    upper = rep(1, p*1), basis = basis, obs = obs, data = data, weight = weight,
    t = t, newvectors = new.basis, ...)
    best.patterns <- cbind(new.basis, ReconBasis(opt$par, basis, 1))
    basis <- ResidBasis(best.patterns, data, orthonorm = orthonorm)[,1:ens.size]
    # forces best.patterns to be orthonormal before doing SVD
    x <- c(x, opt$par)
    basisvars <- numeric(ens.size)
    for (j in 1:ens.size){
     basisvars <- VarExplained(basis[,1:j], data)
    }
    q <- which(basisvars >= vtot)[1]
    mse[i] <- WeightedMSE(obs, ReconObs(obs, basis[,1:q]), weight)
    var[i] <- VarExplained(basis[,i], data)
    new.basis <- cbind(new.basis, basis[,i])
    basis <- basis[,-(1:i)]
    if (round(mse[i],4) == round(minMSE,4)) break
  }
  new.basis <- cbind(new.basis, basis)[,1:ens.size]
  return(list(tBasis = new.basis, CentredField = DataBasis$CentredField,
  EnsembleMean = DataBasis$EnsembleMean, scaling = DataBasis$scaling,
```

```
   Months = DataBasis$Months, MSEq = mse, Varianceq = var, Opt = x))
}

WeightOptim <- function(x, basis, obs, data, weight, t = 0.95, newvectors = NULL){
  n <- dim(basis)[1]
  p <- dim(basis)[2]
  new.basis <- as.vector(tensor(basis, x, 2, 1))
  if (is.null(newvectors) == FALSE){
    new.basis <- cbind(newvectors, new.basis)
  }
  recon <- ReconObs(obs, new.basis)
  y <- WeightedMSE(obs, recon, weight)
  if (is.null(newvectors) == TRUE){
    v <- VarExplained(new.basis, data)
  }
  else {
    v <- VarExplained(new.basis[,dim(new.basis)[2]], data)
  }
  if (v < t){y <- 999999999}
  return(y)
}
```

D.3. History matching and Bayesian calibration

In this section, we present the code used for history matching and calibrating spatial output.

Firstly, a function that takes the basis, observations, and emulator expectations and variances for the basis coefficients, and history matches either using the field implausibility or coefficient implausibility:

```
HistoryMatch <- function(DataBasis, type = "scoresMV", Obs, Expectation,
Variance, Error = NULL, Disc = NULL){
  stopifnot(type == "scoresMV" | type == "field")
  q <- dim(Expectation)[2]
  stopifnot(q == dim(Variance)[2])
  basis <- DataBasis$tBasis[,1:q]
  l <- dim(basis)[1]
  size <- dim(Expectation)[1]
  V2 <- DataBasis$tBasis[,-(1:q)]
  DeletedScores <- CalcScores(DataBasis$CentredField, V2)
  EstVar <- apply(DeletedScores, 2, var)
  BasisVar <- V2 %*% diag(EstVar) %*% t(V2)
  if (type == "scoresMV"){
    impl <- numeric(size)
    BasisVar <- VarProj(BasisVar, basis)
    bound <- qchisq(0.995, q)
```

```
    Obs <- CalcScores(Obs, basis)
    if (is.null(Error) == FALSE){
      Error <- VarProj(Error, basis)
    }
    else {
      Error <- matrix(numeric(q^2), nrow = q)
    }
    Error <- Error + BasisVar
    if (is.null(Disc) == FALSE){
      Disc <- VarProj(Disc, basis)
    }
    else {
      Disc <- matrix(numeric(q^2), nrow = q)
    }
    impl <- as.numeric(mclapply(1:size, function(i) ImplMVScores(Expectation[i,],
    Variance[i,], Obs, Error, Disc)))
    nroy <- sum(impl < bound)/size*100
    inNROY <- impl < bound
  }
  if (type == "field"){
    impl <- numeric(size)
    if (is.null(Error) == TRUE){
      Error <- matrix(numeric(l^2), nrow = l)
    }
    if (is.null(Disc) == TRUE){
      Disc <- matrix(numeric(l^2), nrow = l)
    }
    Error <- Error + BasisVar
    impl <- as.numeric(mclapply(1:size, function(i) ImplField(basis,
     Expectation[i,], Variance[i,], Obs, Error, Disc)))
    bound <- qchisq(0.995, l)
    nroy <- sum(impl < bound)/size*100
    inNROY <- impl < bound
  }
  return(list(impl = impl, nroy = nroy, bound = bound, inNROY = inNROY))
}
```

Calculating the field and coefficient implausibilities:

```
ImplMVScores <- function(Expectation, Variance, Obs, Error, Disc){
  V <- Error + Disc + diag(Variance)
  Q <- chol(V)
  proj.output <- Expectation
  y <- backsolve(Q, as.vector(Obs - proj.output), transpose = TRUE)
  impl <- crossprod(y,y)
  return(impl)
}

ImplField <- function(Basis, Expectation, Variance, Obs, Error, Disc){
  proj.output <- Expectation
  recon.output <- Recon(proj.output, Basis)
  var.output <- diag(Variance)
  recon.var <- Basis %*% var.output %*% t(Basis)
```

```
  V <- Error + Disc + recon.var
  Q <- chol(V)
  y <- backsolve(Q, as.vector(Obs - recon.output), transpose = TRUE)
  impl <- crossprod(y,y)
  return(impl)
}
```

Fitting a Bayesian hierarchical model so that a more accurate bound for history matching on the coefficients can be found, with the prior distribution as used for the spatial toy function in Section 6.2.4:

```
CalImplModel <- function(x, ImplData, prior.x = logBoundPrior){
  n <- dim(ImplData)[1]
  Ic <- ImplData$Ic
  If <- ImplData$If
  x <- as.data.frame(t(x))
  logPrior <- as.numeric(logBoundPrior(x))
  if (logPrior == -Inf){
    logL <- -Inf
    return(logL)
  }
  else {
    w <- numeric(n)
    for (i in 1:n){w[i] <- as.numeric(exp(x[1] + x[2]*log(If[i])))}
    logC <- 0
    for (i in 1:n){
      sigmai <- x[5]/w[i]
      newlogC <- as.numeric(-0.5*log(2*sigmai*pi) - (1/(2*sigmai))*
      (Ic[i] - (x[3] + x[4]*If[i]))^2)
      logC <- logC + newlogC
    }
    logL <- logC + logPrior
    if (is.na(logL) == TRUE){logL <- -Inf}
    if (logL == +Inf){logL <- -Inf}
    return(logL)
  }
}

logBoundPrior <- function(x){
  if (x[4] <= 0 | x[2] >= 0 | x[5] <= 0 | x[1] < -10 | x[1] > 10 | x[3] > 0 |
  x[3] < -50){return(-Inf)}
  else {
    logb1 <- 9*log(x[4]) - x[4]
    loga1 <- 9*log(-x[2]) + x[2]
    logb0 <- -(x[3]^2)
    loga0 <- -(x[1]^2)
    sigma <- log(1/x[5])
    total <- logb1 + loga1 + logb0 + loga0 + sigma
    return (total)
  }
}
```

The functions used for Bayesian calibration of spatial output, where the uncertainty from the basis vectors that are not emulated is included:

```
Calibrate <- function(Ems, DataBasis, Obs, Error, Disc, Prior.x=logPriorDist, ...){
  q <- length(Ems)
  V2 <- DataBasis$tBasis[,-(1:q)]
  DeletedScores <- CalcScores(DataBasis$CentredField, V2)
  EstVar <- apply(DeletedScores, 2, var)
  BasisVar <- V2 %*% diag(EstVar) %*% t(V2)
  Error <- Error + BasisVar
  Posterior <- metrop(CalLikelihood, Ems=ems, Prior.x = Prior.x, DataBasis =
  DataBasis, Obs = obs, Error = Error, Disc = Disc, ...)
  return(Posterior)
}

CalLikelihood <- function(x, Ems, DataBasis, Obs, Error, Disc, VarNames,
Prior.x=logUnifDist){
  x <- as.data.frame(x)
  if (dim(x)[1] > dim(x)[2]) x <- as.data.frame(t(x))
  colnames(x) <- VarNames
  q <- length(Ems)
  Basis <- DataBasis$tBasis[,1:q]
  logPrior <- Prior.x(x)
  EmOutput <- lapply(1:q, function(e) pred.em(Ems[[e]], x))
  Expectation <- Variance <- numeric(q)
  for (i in 1:q){
    Expectation[i] <- EmOutput[[i]]$post.m
    Variance[i] <- EmOutput[[i]]$post.cov
  }
  Expectation <- Recon(Expectation, Basis)
  Variance <- Basis %*% diag(Variance) %*% t(Basis)
  V <- Error + Disc + Variance
  Q <- chol(V)
  y <- backsolve(Q, as.vector(Obs - Expectation))
  y <- crossprod(y,y)
  logL <- as.numeric(-0.5*determinant(V, logarithm=TRUE)$modulus - 0.5*y +
  logPrior)
  return(logL)
}

logUnifDist <- function(x){
  if (any(x < -1) || any(x > 1))
    return (-Inf)
  else {
    return (1)
  }
}
```

D.4. Example

In this section, we provide code for performing history matching and calibration of our spatial toy function.

First, we define the function (equation (A.5)):

```
basis1 <- basis2 <- basis3 <- basis4 <- basis5 <- basis6 <- basis7 <-
basis8 <- matrix(c(rep(0,100)),nrow=10)
for (i in 1:10){basis1[i,i] <- 1}
basis1[10,1] <- basis1[10,2] <- basis1[9,1] <- basis1[1,9] <- basis1[1,10] <-
basis1[2,10] <- -1
basis1[8,1] <- basis1[9,2] <- basis1[10,3] <- basis1[1,8] <- basis1[2,9] <-
basis1[3,10] <- -1
for (i in 1:9){basis2[i,i+1] <- 1}
for (i in 1:8){basis3[1,i] <- 1}
for (i in 1:5){basis3[2,i] <- 1}
for (i in 1:3){basis3[3,i] <- 1}
basis3[9,9] <- basis3[10,10] <- basis3[9,10] <- basis3[10,9] <- basis3[8,9] <-
basis3[8,10] <- basis3[7,2] <- basis3[7,3] <- basis3[7,4] <- basis3[8,2] <-
basis3[8,3] <- basis3[8,4] <- -1
basis3[4,1] <- basis3[5,2] <- basis3[5,3] <- basis3[5,4] <- basis3[6,2] <-
basis3[6,3] <- basis3[6,4] <- 1
for (i in 3:7){basis4[10,i] <- basis4[9,i] <- 1}
basis4[10,3] <- 0
basis4[6,1] <- basis4[6,2] <- basis4[7,1] <- basis4[6,3] <- basis4[1,1] <-
basis4[1,2] <- basis4[2,1] <- -1
basis4[5,2] <- basis4[5,3] <- basis4[2,3] <- basis4[2,4] <- basis4[3,3] <- 1
for (i in 8:10){basis5[4,i] <- basis5[5,i] <- basis5[6,i] <- basis5[7,i] <- 1}
for (i in 6:8){basis5[3,i] <- basis5[2,i] <- basis5[1,i] <- -1}
basis5[4,7] <- basis5[4,6] <- -1
basis5[1,8] <- 0
for (i in 8:9){basis5[i,10] <- -1}
for (i in 6:8){basis6[i,5] <- basis6[i,6] <- 1}
basis6[5,6] <- basis6[5,7] <- basis6[8,5] <- basis6[8,4] <- 1
for (i in 6:7){basis6[i,2] <- basis6[i,3] <- -1}
for (i in 6:8){basis6[10,i] <- -1}
basis6[9,9] <- basis6[9,10] <- basis6[1,4] <- basis6[1,5] <- basis6[8,2] <-
basis6[6,9] <- -1
basis7[2,10] <- basis7[3,10] <- basis7[3,9] <- basis7[3,8] <- basis7[10,8] <-
basis7[9,8] <- basis7[8,8] <- basis7[9,9] <- basis7[7,6] <- basis7[7,9] <-
basis7[5,7] <- basis7[3,2] <- basis7[4,2] <- basis7[1,8] <- basis7[2,8] <-
basis7[2,2] <- basis7[8,3] <- basis7[7,3] <- basis7[10,4] <- basis7[9,4] <-
basis7[8,7] <- 1
basis7[7,5] <- basis7[6,5] <- basis7[6,6] <- basis7[7,4] <- basis7[8,1] <-
basis7[8,2] <- basis7[1,3] <- basis7[1,4] <- basis7[10,6] <- basis7[10,5] <-
basis7[6,9] <- basis7[6,10] <- basis7[1,6] <- basis7[2,6] <- basis7[2,7] <-
basis7[10,9] <- -1
basis8[3,5] <- basis8[3,6] <- basis8[4,5] <- basis8[4,4] <- basis8[5,10] <-
basis8[6,10] <- basis8[1,9] <- basis8[9,7] <- 1
basis8[9,5] <- basis8[8,5] <- basis8[7,5] <- basis8[10,7] <- basis8[7,2] <-
```

```
basis8[5,4] <- basis8[7,1] <- basis8[10,1] <- basis8[7,10] <- basis8[7,7] <-
basis8[5,1] <- basis8[4,3] <- basis8[6,7] <- -1

fn <- function(x){
  fx <- 3*(10*x[2]^2*basis2 + 5*x[3]^2*basis2 + (x[3] + 1.5*x[1]*x[2])*basis3 +
  2*x[2]*basis4 + x[3]*x[1]*basis5 + (x[2]*x[1])*basis6 + (x[2]^3)*basis7 +
  ((x[2] + x[3])^2)*basis8 + 2) + 1.5*dnorm(x[4], 0.2, 0.1)*basis1*(x[5]/(1.3+
  x[6]))
  noise <- matrix(rnorm(100,0,0.05),nrow=10)
  fx <- fx + noise
  return(fx)
}
```

We sample an ensemble with n members from the full parameter space $\mathcal{X}$, converting the ensemble into the form required by our code, and calculating the SVD basis for the (centred) ensemble:

```
n <- 60
sample <- as.data.frame(2*maximinLHS(n,6) - 1)
colnames(sample) <- c("x1","x2","x3","x4","x5","x6")
data <- array(c(rep(0,100*n)), dim=c(10,10,n))
for (i in 1:n){
  data[,,i] <- fn(as.numeric(sample[i,]))
}
dim(data) <- c(100, n)

f1data <- NULL
f1data$EnsembleMean <- apply(data, 1, mean)
f1data$CentredField <- data - f1data$EnsembleMean
f1data$tBasis <- svd(t(f1data$CentredField))$v
f1data$scaling <- 1
```

So that we can history match, we define 100×100 observation error $\mathbf{\Sigma_e}$ and discrepancy variance $\mathbf{\Sigma_\eta}$ matrices (here we set these as the same matrix):

```
errvar <- discvar <- calcA(expand.grid(1:10,1:10), d = c(1,1), nu = 0)
```

We select a parameter value $\mathbf{x}^*$, and evaluate the function at this input, adding a sample from the observation error variance to give the observations $\mathbf{z}$, which we then centre using the ensemble mean:

```
random.e <- matrix(mvrnorm(1, c(rep(0,100)), errvar),nrow=10)
obs <- c(fn(c(0.7,0.01,0.01,0.25,0.8,-0.9)) + random.e)
obs <- obs - f1data$EnsembleMean
```

To assess the quality of the basis, we consider the VarMSE plot, and also plot the reconstruction of the observations, when the first four vectors of the SVD basis are used for

projection:

```
VarMSEplot(f1data, obs, errvar+discvar)
plot.field <- function(field, dim1, col = rainbow(100,start=0.1,end=0.8), ...){
  require(fields)
  x <- 1:dim1
  y <- 1:dim1
  image.plot(x, y, matrix(field, nrow = dim1), col = col, add = FALSE, ...)
}
plot.field(ReconObs(obs, f1data$tBasis[,1:4]), dim1 = 10)
```

As the truncated SVD basis is unsuitable for representing $\mathbf{z}$, we search for an optimal rotation of this basis:

```
f1rot <- RotateIterate(k = 3, f1data, obs, weight = errvar + discvar,
var.threshold = 0.2, vtot = 0.95, prior = NULL, control = list(max.time=60))
```

As for the SVD basis, we can assess the quality of the rotated basis by looking at the VarMSE plot, and the reconstruction of the observations when the first five basis vectors of the rotated basis are used for projection:

```
VarMSEplot(f1rot, obs, errvar+discvar)
plot.field(ReconObs(obs, f1rot$tBasis[,1:5]), dim1 = 10)
```

We project the ensemble onto the truncated rotated basis, and build emulators for these coefficients:

```
tData <- cbind(sample, runif(length(sample[,1]),-1,1), CalcScores(
f1rot$CentredField, f1rot$tBasis[,1:5]))
colnames(tData) <- c(colnames(sample), "Noise", "C1", "C2", "C3", "C4", "C5")
ems <- BasisEmulators(tData, 5)
```

Using these emulators (given that they passed the validation checks), we take a large Latin hypercube sample from parameter space, and evaluate the emulators at each input:

```
lhcsample <- as.data.frame(2*maximinLHS(100000,6) - 1)
colnames(lhcsample) <- c("x1","x2","x3","x4","x5","x6")
EmOutput <- MultiPred(ems, lhcsample)
```

Given these predictions, we can history match both using the field implausibility:

```
f1hmf <- HistoryMatch(DataBasis=f1rot, type="field", Obs=c(obs), Expectation=
EmOutput$Expectation, Variance=EmOutput$Variance, Error=errvar, Disc=discvar)
f1hmf$nroy
nroyspace <- lhcsample[which(f1hmf$impl < f1hmf$bound),]
```

and coefficient implausibility:

```
f1hmc <- HistoryMatch(DataBasis=f1rot, type="scoresMV", Obs=c(obs), Expectation=
EmOutput$Expectation, Variance=EmOutput$Variance, Error=errvar, Disc=discvar)
```

To estimate a more appropriate bound to rule out parameter settings using the coefficient implausibility, we fit the following Bayesian model, using a small sample of implausibilities:

```
lhcI <- as.data.frame(2*maximinLHS(20,6) - 1)
colnames(lhcI) <- colnames(lhcsample)
EmOutputI <- MultiPred(ems, lhcI)
If <- HistoryMatch(DataBasis=f1rot, type="field", Obs=c(obs), Expectation=
EmOutputI$Expectation, Variance=EmOutputI$Variance, Error=errvar, Disc=
discvar)$impl
Ic <- HistoryMatch(DataBasis=f1rot, type="scoresMV", Obs=c(obs), Expectation=
EmOutputI$Expectation, Variance=EmOutputI$Variance, Error=errvar, Disc=
discvar)$impl
tDataI <- as.data.frame(cbind(If, Ic))

MCMCimpl <- metrop(CalImplModel, initial=c(5,-2.5,-25,1,0.1),scale=c(0.1,2,2,
0.5,0.01), nbatch = 10000, ImplData = tDataI, prior.x = logBoundPrior)
```

The Metropolis-Hastings step requires careful tuning of the scale to ensure convergence. With this posterior distribution, we can either sample values of the model parameters, or fix these at a value from the posterior. Here, we take the final values from the chains, and draw samples for $\mathcal{I}_c|\mathcal{I}_f = T$, the chi-squared bound for 100-dimensional output:

```
icsamp <- numeric(100000)
xx <- MCMCimpl$final
for (i in 1:100000){
  mu <- as.numeric(xx[1] + xx[2]*log(qchisq(0.995,100)))
  wi <- exp(mu)
  mu2 <- as.numeric(xx[3] + xx[4]*(qchisq(0.995,100)))
  sigma <- as.numeric(xx[5]/wi)
  icsamp[i] <- rnorm(1, mu2, sd = sqrt(sigma))
}

plot(density(icsamp))

newbound <- quantile(icsamp, probs=c(0.995))
```

With this new bound, we can revisit history matching with the coefficient implausibility, and find the size of NROY space:

```
sum(f1hmc$impl < newbound) / length(f1hmc$impl)
nroyspaceC <- lhcsample[which(f1hmc$impl < f1hmc$bound),]
```

This should rule out less space than the coefficient implausibility did previously, and this NROY space is likely to be larger than that found with the field implausibility. However,

we should avoid ruling out runs consistent with the observations on the field.

We can also perform Bayesian calibration using the rotated basis and its emulators, reconstructing fields from predicted coefficients:

```
calw1 <- Calibrate(Ems=ems, DataBasis=f1rot, Obs=obs, Error=errvar, Disc=discvar,
VarNames=colnames(sample), Prior.x=logUnifDist, initial=c(rep(0,6)),
scale=c(rep(0.1,6)), nbatch=10000)
```

Bibliography

Abdi, H. (2003). Factor rotations in factor analyses. *Encyclopedia for Research Methods for the Social Sciences. Sage: Thousand Oaks, CA*, pages 792–795.

Aguilera, A. and Pérez-Aguila, R. (2004). General n-dimensional rotations.

Aksoy, A., Zhang, F., and Nielsen-Gammon, J. W. (2006). Ensemble-based simultaneous state and parameter estimation with MM5. *Geophysical Research Letters*, 33(12).

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

Andrianakis, I., McCreesh, N., Vernon, I., McKinley, T., Oakley, J., Nsubuga, R., Goldstein, M., and White, R. (2017). Efficient history matching of a high dimensional individual based hiv transmission model.

Andrianakis, I., Vernon, I. R., McCreesh, N., McKinley, T. J., Oakley, J. E., Nsubuga, R. N., Goldstein, M., and White, R. G. (2015). Bayesian History Matching of Complex Infectious Disease Models Using Emulation: A Tutorial and a Case Study on HIV in Uganda. *PLoS computational biology*, 11(1):e1003968.

Annan, J., Hargreaves, J., Edwards, N., and Marsh, R. (2005). Parameter estimation in an intermediate complexity earth system model using an ensemble kalman filter. *Ocean modelling*, 8(1):135–154.

Bayarri, M., Berger, J., Cafeo, J., Garcia-Donato, G., Liu, F., Palomo, J., Parthasarathy, R., Paulo, R., Sacks, J., and Walsh, D. (2007). Computer model validation with functional output. *The Annals of Statistics*, pages 1874–1906.

Bellprat, O., Kotlarski, S., Lüthi, D., and Schär, C. (2012). Objective calibration of regional climate models. *Journal of Geophysical Research: Atmospheres*, 117(D23).

Bhat, K., Haran, M., and Goes, M. (2010). Computer model calibration with multivariate spatial output: a case study. *Frontiers of Statistical Decision Making and Bayesian Analysis*, pages 168–184.

Bhat, K., Haran, M., Olson, R., and Keller, K. (2012). Inferring likelihoods and climate system characteristics from climate models and multiple tracers. *Environmetrics*, 23(4):345–362.

Bindoff, N. L., Stott, P. A., AchutaRao, K. M., Allen, M. R., Gillett, N., Gutzler, D., Hansingo, K., Hegerl, G., Hu, Y., Jain, S., et al. (2013). Detection and attribution of climate change: from global to regional.

Björck, Å. (1967). Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1):1–21.

Björck, Å. (1994). Numerics of Gram-Schmidt orthogonalization. *Linear Algebra and Its Applications*, 197:297–316.

Bounceur, N., Crucifix, M., and Wilkinson, R. (2015). Global sensitivity analysis of the climate-vegetation system to astronomical forcing: an emulator-based approach. *Earth System Dynamics*, 6(1):205.

Brynjarsdóttir, J. and O'Hagan, A. (2014). Learning about physical parameters: The importance of model discrepancy. *Inverse Problems*, 30(11):114007.

Chaboureau, J.-P. and Bechtold, P. (2005). Statistical representation of clouds in a regional model and the impact on the diurnal cycle of convection during Tropical Convection, Cirrus and Nitrogen Oxides (TROCCINOX). *Journal of Geophysical Research: Atmospheres*, 110(D17).

Chang, W., Applegate, P. J., Haran, M., and Keller, K. (2014a). Probabilistic calibration of a Greenland Ice Sheet model using spatially-resolved synthetic observations: toward projections of ice mass loss with uncertainties. *Geoscientific Model Development Discussions*, 7(2):1905–1931.

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

Chang, W., Haran, M., Applegate, P., Pollard, D., et al. (2017). Improving ice sheet model calibration using paleoclimate and modern data. *The Annals of Applied Statistics*, 10(4):2274–2302.

Chang, W., Haran, M., Olson, R., Keller, K., et al. (2014b). Fast dimension-reduced climate model calibration and the effect of data aggregation. *The Annals of Applied Statistics*, 8(2):649–673.

Chen, H., Loeppky, J. L., Sacks, J., Welch, W. J., et al. (2016). Analysis Methods for Computer Experiments: How to Assess and What Counts? *Statistical Science*, 31(1):40–60.

Conti, S., Gosling, J. P., Oakley, J., and O'Hagan, A. (2009). Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3):640–651.

Craig, P. S., Goldstein, M., Rougier, J. C., and Seheult, A. H. (2001). Bayesian Forecasting for Complex Systems using Computer Simulators. *Journal of the American Statistical Association*, 96(454):717–729.

Craig, P. S., Goldstein, M., Seheult, A., and Smith, J. (1996). Bayes linear strategies for matching hydrocarbon reservoir history. *Bayesian statistics*, 5:69–95.

Craig, P. S., Goldstein, M., Seheult, A. H., and Smith, J. A. (1997). 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.

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

Dee, D., Uppala, S., Simmons, A., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M., Balsamo, G., Bauer, P., et al. (2011). The ERA-Interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the royal meteorological society*, 137(656):553–597.

Dickinson, R. E., Errico, R. M., Giorgi, F., and Bates, G. T. (1989). A regional climate model for the western United States. *Climatic change*, 15(3):383–422.

Duffin, K. L. and Barrett, W. A. (1994). Spiders: a new user interface for rotation and visualization of n-dimensional point sets. In *Proceedings of the conference on Visualization'94*, pages 205–211. IEEE Computer Society Press.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Edwards, N. R., Cameron, D., and Rougier, J. (2011). Precalibrating an intermediate complexity climate model. *Climate dynamics*, 37(7-8):1469–1482.

Fahrmeir, L. and Tutz, G. (2001). Models for multicategorical responses: Multivariate extensions of generalized linear models. In *Multivariate Statistical Modelling Based on Generalized Linear Models*, pages 69–137. Springer.

Farah, M., Birrell, P., Conti, S., and Angelis, D. D. (2014). Bayesian Emulation and Calibration of a Dynamic Epidemic Model for A/H1N1 Influenza. *Journal of the American Statistical Association*, 109(508):1398–1411.

Foley, A., Holden, P., Edwards, N., Mercure, J., Salas, P., Pollitt, H., and Chewpreecha, U. (2016). Climate model emulation in an integrated assessment framework: a case study for mitigation policies in the electricity sector. *Earth System Dynamics*, 7(1):119.

Fricker, T. E., Oakley, J. E., and Urban, N. M. (2013). Multivariate Gaussian process emulators with nonseparable covariance structures. *Technometrics*, 55(1):47–56.

Furrer, R., Sain, S. R., Nychka, D., and Meehl, G. A. (2007). Multivariate Bayesian analysis of atmosphere–ocean general circulation models. *Environmental and ecological statistics*, 14(3):249–266.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2014). *Bayesian data analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA.

Gladstone, R. M., Lee, V., Rougier, J., Payne, A. J., Hellmer, H., Le Brocq, A., Shepherd, A., Edwards, T. L., Gregory, J., and Cornford, S. L. (2012). 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.

Golchi, S., Bingham, D., Chipman, H., and Campbell, D. (2015). Monotone emulation of computer experiments. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):370–392.

Goldstein, M. and Wooff, D. (2007). *Bayes Linear Statistics, Theory & Methods*, volume 716. John Wiley & Sons.

Golub, G. H. and Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420.

Gómez, F. A., Coleman-Smith, C. E., O'Shea, B. W., Tumlinson, J., and Wolpert, R. L. (2012). Characterizing the formation history of Milky Way like stellar halos with model emulators. *The Astrophysical Journal*, 760(2):112.

Gordon, C., Cooper, C., Senior, C. A., Banks, H., Gregory, J. M., Johns, T. C., Mitchell, J. F., and Wood, R. A. (2000). The simulation of SST, sea ice extents and ocean heat transports in a version of the Hadley Centre coupled model without flux adjustments. *Climate Dynamics*, 16(2-3):147–168.

Gramacy, R. B., Bingham, D., Holloway, J. P., Grosskopf, M. J., Kuranz, C. C., Rutter, E., Trantham, M., Drake, R. P., et al. (2015). Calibrating a large computer experiment simulating radiative shock hydrodynamics. *The Annals of Applied Statistics*, 9(3):1141–1168.

Gramacy, R. B. and Lee, H. K. (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3):713–722.

Gu, M., Berger, J. O., et al. (2016). Parallel partial Gaussian process emulation for

computer models with massive output. *The Annals of Applied Statistics*, 10(3):1317–1347.

Guillas, S., Rougier, J., Maute, A., Richmond, A., and Linkletter, C. (2009). Bayesian calibration of the Thermosphere-Ionosphere Electrodynamics General Circulation Model (TIE-GCM). *Geoscientific Model Development*, 2(2):137–144.

Hall, J. W., Manning, L. J., and Hankin, R. K. (2011). Bayesian calibration of a flood inundation model using spatial data. *Water Resources Research*, 47(5).

Harrild, D. M. and Henriquez, C. S. (2000). A computer model of normal conduction in the human atria. *Circulation research*, 87(7):e25–e36.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Hawkins, E. and Sutton, R. (2009). The potential to narrow uncertainty in regional climate predictions. *Bulletin of the American Meteorological Society*, 90(8):1095–1107.

Haylock, R. and O'Hagan, A. (1996). On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. *Bayesian statistics*, 5:629–637.

Heaton, M. J., Kleiber, W., Sain, S. R., and Wiltberger, M. (2015). Emulating and calibrating the multiple-fidelity Lyon–Fedder–Mobarry magnetosphere–ionosphere coupled computer model. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 64(1):93–113.

Henderson, D. A., Boys, R. J., Krishnan, K. J., Lawless, C., and Wilkinson, D. J. (2012). Bayesian emulation and calibration of a stochastic computer model of mitochondrial DNA deletions in substantia nigra neurons. *Journal of the American Statistical Association*.

Heo, Y., Augenbroe, G., and Choudhary, R. (2011). Risk analysis of energy-efficiency projects based on Bayesian calibration of building energy models. In *Building simulation*, pages 2579–2586.

Higdon, D. (1998). A process-convolution approach to modelling temperatures in the North Atlantic Ocean. *Environmental and Ecological Statistics*, 5(2):173–190.

Higdon, D. (2002). Space and space-time modeling using process convolutions. In *Quantitative methods for current environmental issues*, pages 37–56. Springer.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008a). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482).

Higdon, D., Kennedy, M., Cavendish, J. C., Cafeo, J. A., and Ryne, R. D. (2004). Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466.

Higdon, D., Nakhleh, C., Gattiker, J., and Williams, B. (2008b). A Bayesian calibration approach to the thermal problem. *Computer Methods in Applied Mechanics and Engineering*, 197(29):2431–2441.

Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*. SIAM.

Holden, P. and Edwards, N. (2010). Dimensionally reduced emulation of an AOGCM for application to integrated assessment modelling. *Geophysical Research Letters*, 37(21).

Holden, P., Edwards, N., Garthwaite, P., Fraedrich, K., Lunkeit, F., Kirk, E., Labriet, M., Kanudia, A., and Babonneau, F. (2013). PLASIM-ENTSem: a spatio-temporal emulator of future climate change for impacts assessment. *Geoscientific model development discussions*, 6(2):3349–3380.

Holden, P. B., Edwards, N. R., Garthwaite, P. H., and Wilkinson, R. D. (2015). Emulation and interpretation of high-dimensional climate model outputs. *Journal of Applied Statistics*, 42(9):2038–2055.

Hourdin, F., Mauritsen, T., Gettelman, A., Golaz, J. C., Balaji, V., Duan, Q., Folini, D., Ji, D., Klocke, D., Qian, Y., Rauser, F., Rio, C., Tomassini, L., Watanabe, M., and Williamson, D. (2016). The art and science of climate model tuning. *BAMS*. Revised Once.

Hurley, J. R. and Cattell, R. B. (1962). The Procrustes program: Producing direct rotation to test a hypothesized factor structure. *Systems Research and Behavioral Science*, 7(2):258–262.

Ingleby, B. and Huddleston, M. (2007). Quality control of ocean temperature and salinity profiles - Historical and real-time data. *Journal of Marine Systems*, 65(1):158–175.

Johns, T., Gregory, J., Ingram, W., Johnson, C., Jones, A., Lowe, J., Mitchell, J., Roberts, D., Sexton, D., Stevenson, D., et al. (2003). Anthropogenic climate change for 1860 to 2100 simulated with the HadCM3 model under updated emissions scenarios. *Climate Dynamics*, 20(6):583–612.

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

Jones, R. H. (1963). Stochastic processes on a sphere. *The Annals of mathematical statistics*, 34(1):213–218.

Juan Jose, R., Manuel, P., and Takemasa, M. (2013). Estimating model parameters with ensemble-based data assimilation: A review. . *2* , 91(2):79–99.

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200.

Kaiser, H. F. (1974). A note on the equamax criterion. *Multivariate behavioral research*, 9(4):501–503.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kaufman, C. G., Bingham, D., Habib, S., Heitmann, K., and Frieman, J. A. (2011). Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. *The Annals of Applied Statistics*, 5(4):2470–2492.

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13.

Kennedy, M. C. and O'Hagan, A. (2001a). Bayesian calibration of computer models.

Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(3):425–464.

Kennedy, M. C. and O'Hagan, A. (2001b). Supplementary details on Bayesian calibration of computer models. Technical report, Internal Report. URL http://www. shef. ac. uk/~ st1ao/ps/calsup. ps.

Keppenne, C. L. (2000). Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter. *Monthly Weather Review*, 128(6):1971–1981.

Kiehl, J., Hack, J., Bonan, G., Boville, B., Williamson, D., and Rasch, P. (1998). The national center for atmospheric research community climate model: CCM3*. *Journal of Climate*, 11(6):1131–1149.

Kuttler, K. (2012). *Linear algebra: theory and applications*. The Saylor Foundation.

Lee, L., Pringle, K., Reddington, C., Mann, G., Stier, P., Spracklen, D., Pierce, J., and Carslaw, K. (2013). The magnitude and causes of uncertainty in global model simulations of cloud condensation nuclei. *Atmospheric Chemistry and Physics*, 13(17):8879–8914.

Li, S., Zhang, S., Liu, Z., Yang, X., Rosati, A., Golaz, J.-C., and Zhao, M. (2016). The role of large-scale feedbacks in cumulus convection parameter estimation. *Journal of Climate*, 29(11):4099–4119.

Liu, F. and West, M. (2009). A dynamic modelling strategy for Bayesian computer model emulation. *Bayesian Analysis*, 4(2):393–411.

Madec, G. (2008). NEMO ocean engine. *Note du Pole de Modlisation*, (27).

McFarland, J., Mahadevan, S., Swiler, L., and Giunta, A. (2007). Bayesian calibration of the QASPR simulation. In *Forty-eighth AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference, Honolulu, HI.*

McNeall, D., Challenor, P., Gattiker, J., and Stone, E. (2013). The potential of an observational data set for calibration of a computationally expensive computer model. *Geoscientific Model Development*, 6(5):1715–1728.

Meehl, G., Covey, C., Delworth, T., Latif, M., McAvaney, B., Mitchell, J., Stouffer, R., and Taylor, K. (2007). The WCRP CMIP3 multi-model dataset: A new era in climate change research. *Bulletin of the American Meteorological Society*, 88:1383–1394.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

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

Morris, M. D., Mitchell, T. J., and Ylvisaker, D. (1993). Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255.

Murnaghan, F. D. (1962). *The unitary and rotation groups*, volume 3. Spartan books.

Murphy, J., Sexton, D., Jenkins, G., Boorman, P., Booth, B., Brown, C., Clark, R., Collins, M., Harris, G., Kendon, E., et al. (2009). UK climate projections science report: UKCP09. *Met Office Hadley Centre: Exeter, UK*.

Neuhaus, J. O. and Wrigley, C. (1954). The quartimax method. *British Journal of Statistical Psychology*, 7(2):81–91.

Nychka, D., Wikle, C., and Royle, J. A. (2002). Multiresolution models for nonstationary spatial covariance functions. *Statistical Modelling*, 2(4):315–331.

Oakley, J. (2002). Eliciting Gaussian process priors for complex computer codes. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 51(1):81–97.

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

Oughton, R. H. and Craig, P. S. (2016). Hierarchical Emulation: A Method for Modeling and Comparing Nested Simulators. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):495–519.

Oyebamiji, O. K., Edwards, N. R., Holden, P. B., Garthwaite, P. H., Schaphoff, S., and

Gerten, D. (2015). Emulating global climate change impacts on crop yields. *Statistical Modelling*, page 1471082X14568248.

Pope, V., Gallani, M., Rowntree, P., and Stratton, R. (2000). The impact of new physical parametrizations in the Hadley Centre climate model: HadAM3. *Climate Dynamics*, 16(2-3):123–146.

Pukelsheim, F. (1994). The three sigma rule. *The American Statistician*, 48(2):88–91.

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

Richman, M. B. (1986). Rotation of principal components. *Journal of climatology*, 6(3):293–335.

Rodgers, K. B., Blanke, B., Madec, G., Aumont, O., Ciais, P., and Dutay, J.-C. (2003). Extratropical sources of Equatorial Pacific upwelling in an OGCM. *Geophysical research letters*, 30(2).

Rodrigues, L. F. S., Vernon, I., and Bower, R. G. (2016). Constraints on galaxy formation models from the galaxy stellar mass function and its evolution. *Monthly Notices of the Royal Astronomical Society*, page stw3269.

Rougier, J. (2007). Probabilistic inference for future climate using an ensemble of climate model evaluations. *Climatic Change*, 81(3-4):247–264.

Rougier, J. (2008). Efficient Emulators for Multivariate Deterministic Functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843.

Rougier, J., Sexton, D. M., Murphy, J. M., and Stainforth, D. (2009). Analyzing the Climate Sensitivity of the HadSM3 Climate Model Using Ensembles from Different but Related Experiments. *Journal of Climate*, 22(13).

Roustant, O., Ginsbourger, D., and Deville, Y. (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software*, 51(1):1–55.

Rudin, W. (1976). Principles of Mathematical Analysis (International Series in Pure & Applied Mathematics).

Ruiz, J. J., Pulido, M., and Miyoshi, T. (2013). Estimating model parameters with ensemble-based data assimilation: Parameter covariance treatment. *J. Meteor. Soc. Japan*, 91:453–469.

Sacks, J., Schiller, S. B., and Welch, W. J. (1989a). Designs for Computer Experiments. *Technometrics*, 31(1):41–47.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989b). Design and analysis of computer experiments. *Statistical science*, pages 409–423.

Saltelli, A., Chan, K., Scott, E. M., et al. (2000). *Sensitivity analysis*, volume 1. Wiley New York.

Saltelli, A., Tarantola, S., and Chan, K.-S. (1999). A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56.

Salter, J. M. and Williamson, D. (2016). A comparison of statistical emulation methodologies for multi-wave calibration of environmental models. *Environmetrics*, 27(8):507–523.

Santner, T. J., Williams, B. J., and Notz, W. (2003). *The design and analysis of computer experiments*. Springer.

Schirber, S., Klocke, D., Pincus, R., Quaas, J., and Anderson, J. L. (2013). Parameter estimation using data assimilation in an atmospheric general circulation model: From a perfect toward the real world. *Journal of Advances in Modeling Earth Systems*, 5(1):58–70.

Sexton, D. M., Murphy, J. M., Collins, M., and Webb, M. J. (2011). Multivariate probabilistic projections using imperfect climate models part I: outline of methodology. *Climate dynamics*, 38(11-12):2513–2542.

Simpson, T. W., Poplinski, J., Koch, P. N., and Allen, J. K. (2001). Metamodels for computer-based engineering design: survey and recommendations. *Engineering with computers*, 17(2):129–150.

Tavassoli, Z., Carter, J. N., and King, P. R. (2005). An analysis of history matching errors. *Computational Geosciences*, 9(2-3):99–123.

Tavassoli, Z., Carter, J. N., King, P. R., et al. (2004). Errors in history matching. *SPE Journal*, 9(03):352–361.

Taylor, K. E., Stouffer, R. J., and Meehl, G. A. (2012). An overview of CMIP5 and the experiment design. *Bulletin of the American Meteorological Society*, 93(4):485.

Tripathy, R., Bilionis, I., and Gonzalez, M. (2016). Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *Journal of Computational Physics*, 321:191–223.

Turányi, T. (1990). Sensitivity analysis of complex kinetic systems. Tools and applications. *Journal of Mathematical Chemistry*, 5(3):203–248.

Vernon, I. and Goldstein, M. (2009). Bayes linear analysis of imprecision in computer models, with application to understanding galaxy formation. In *Proceedings of the Sixth International Symposium on Imprecise Probability: Theories and Applications*, pages 441–450. Society for Imprecise Probability: Theories and Applications.

Vernon, I., Goldstein, M., and Bower, R. G. (2010). Galaxy formation: a Bayesian uncertainty analysis. *Bayesian Analysis*, 5(4):619–669.

Vernon, I., Liu, J., Goldstein, M., Rowe, J., Topping, J., and Lindsey, K. (2016). Bayesian uncertainty analysis for complex systems biology models: emulation, global parameter searches and evaluation of gene functions. *arXiv preprint arXiv:1607.06358*.

Vidakovic, B. (2009). *Statistical modeling by wavelets*, volume 503. John Wiley & Sons.

von Salzen, K., Scinocca, J. F., McFarlane, N. A., Li, J., Cole, J. N., Plummer, D., Verseghy, D., Reader, M. C., Ma, X., Lazare, M., et al. (2013). The Canadian fourth generation atmospheric global climate model (CanAM4). Part I: representation of physical processes. *Atmosphere-Ocean*, 51(1):104–125.

Watson, G. N. (1995). *A treatise on the theory of Bessel functions*. Cambridge university press.

Wielicki, B. A., Barkstrom, B. R., Harrison, E. F., Lee III, R. B., Louis Smith, G., and Cooper, J. E. (1996). Clouds and the Earth's Radiant Energy System (CERES): An earth observing system experiment. *Bulletin of the American Meteorological Society*, 77(5):853–868.

Wilkinson, R. D. (2010). Bayesian calibration of expensive multivariate computer experiments. *Large-Scale Inverse Problems and Quantification of Uncertainty, Ser. Comput. Stat., edited by LT Biegler et al*, pages 195–216.

Williams, B., Higdon, D., Gattiker, J., Moore, L., McKay, M., Keller-McNulty, S., et al. (2006). Combining experimental data and computer simulations, with an application to flyer plate experiments. *Bayesian Analysis*, 1(4):765–792.

Williamson, D. (2015). Exploratory ensemble designs for environmental models using k-extended Latin Hypercubes. *Environmetrics*, 26(4):268–283.

Williamson, D. and Blaker, A. T. (2014). Evolving Bayesian emulators for structured chaotic time series, with application to large climate models. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):1–28.

Williamson, D., Blaker, A. T., Hampton, C., and Salter, J. (2015). Identifying and removing structural biases in climate models with history matching. *Climate Dynamics*, 45(5-6):1299–1324.

Williamson, D., Blaker, A. T., and Sinha, B. (2016). Tuning without over-tuning: parametric uncertainty quantification for the NEMO ocean model. *Geoscientific Model Development Discussions*.

Williamson, D., Goldstein, M., Allison, L., Blaker, A., Challenor, P., Jackson, L., and Yamazaki, K. (2013). 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.

Williamson, D., Goldstein, M., and Blaker, A. (2012). Fast linked analyses for scenario-based hierarchies. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(5):665–691.

Woodbury, M. A. (1950). Inverting modified matrices. *Memorandum report*, 42(106):336.

Worley, B. A. (1987). Deterministic uncertainty analysis. Technical report, Oak Ridge National Lab., TN (USA).

Yang Xiang, Gubian, S., Suomela, B., and Hoeng, J. (2013). Generalized Simulated Annealing for Efficient Global Optimization: the GenSA Package for R. *The R Journal Volume 5/1, June 2013*.

Zhang, T., Li, L., Lin, Y., Xue, W., Xie, F., Xu, H., and Huang, X. (2015). An automatic and effective parameter optimization method for model tuning. *Geoscientific Model Development*, 8(11):3579–3591.

Zou, L., Qian, Y., Zhou, T., and Yang, B. (2014). Parameter tuning and calibration of RegCM3 with MIT–Emanuel cumulus parameterization scheme over CORDEX East Asia domain. *Journal of Climate*, 27(20):7687–7701.