Department of Mathematical Sciences

Project IV (MATH4072)

Advances in Bayesian Emulation and the TENSE Framework

Author:
Jamie Reason

Supervisor:
Professor Ian Vernon

April 2024

Plagiarism Declaration

This piece of work is a result of my own work and I have complied with the Departments guidance on multiple submission and on the use of AI tools. Material from the work of others not involved in the project has been acknowledged, quotations and paraphrases suitably indicated, and all uses of AI tools have been declared.

Dedicated to my Dad,
Andrew Reason.

Contents

Chapter 1

Introduction

Complex computer models are ubiquitous across most scientific disciplines as they enable complex systems to be studied and better understood [1]. However, these computer models are often very computationally expensive to run, presenting a challenge to using them effectively. Emulators mimic a computer model from a limited number of runs which allows the computer model to be better understood over the input space.

In this project, we will consider *Bayesian Emulators*, introduced in the next chapter, which exploit prior beliefs about the computer models and give robust uncertainty quantification for the estimates of the computer model. This form of emulator excels at efficiently emulating computer models with a limited number of runs over high-dimensional input spaces. This type of emulator is highly effective for applications such as *History Matching* (section 4.3), where we want to tune computer models to match real-world observations. Bayesian emulators of this kind have been successfully used to emulate computer models in cosmology [1, 2, 3, 4], climate science [5, 6], vulcanology [7], systems biology [8, 9], engineering [10] and many other areas. We will not consider emulators suitable for data-rich regimes, or for computer models with highly complicated outputs such as fluid simulations. For these applications, the Bayesian emulator is inefficient and incorporating prior beliefs is less important [11].

In chapter 2, we will introduce the two main types of Bayesian emulators: Gaussian process (GP) emulators and Bayes Linear (BL) emulators. Gaussian processes are widely used in many areas of machine learning, not just for emulation [12]. Bayes Linear emulators are similar but use the Bayes Linear framework rather than the full Bayesian framework, which avoids making distributional assumptions [13]. We will introduce the Bayes Linear framework in chapter 2 and motivate why this approach may be preferred for emulating computer models.

An important step in constructing a Bayesian emulator is specifying the covariance structure; intuitively, this controls how related the outputs of the computer model are at two different sets of inputs. In chapter 3, we will explore a variety of covariance structures and how we can combine covariance functions to encode specific prior beliefs. In particular, we will look at non-stationary covariance functions and prove that a flexible construction is a valid covariance function. This will play a crucial role in the TENSE framework in chapter 5.

In chapter 4, we will adapt the simple emulator we will see in chapter 2 to incorporate a regression term. This advanced emulator addresses some of the main limitations of the simple emulator and enables more efficient emulation, especially in higher dimensions. In chapter 4 we also consider how we choose the points to run the computer model at, known as experimental design, and motivate the methods we introduce with a brief introduction to history matching [14].

Continuity is a core assumption that standard emulators exploit. With unstructured discontinuities at unknown locations, making meaningful estimates of the computer model at unseen locations is virtually impossible. However, some computer models possess structured partial discontinuities at known locations [15] but are otherwise continuous and well-behaved. In chapter 5 we will introduce

the *Torn Embedding Non-Stationary Emulation* (TENSE) framework [15], which enables computer models with partial known discontinuities to be effectively emulated with the discontinuity structure encoded. However, currently, the TENSE framework only applies to computer models with a 2D input space which limits its applicability. This project's most significant and novel contribution is extending the TENSE framework to arbitrary-dimension input spaces in section 5.3. We will conclude chapter 5 by applying the advanced emulator with the TENSE framework to an example 3D function containing a partial discontinuity.

All code used throughout this project was done in `R` and is available at `github.com/jreaso/mmath-thesis` in the supplementary GitHub repository [16].

Chapter 2

Introduction to Emulation

Many popular function approximations, such as deep neural networks (DNNs) and simple interpolation, rely on having large amounts of data and struggle to quantify uncertainty [17, 18]. However, computer models often model physical systems [7, 19, 20, 6, 21] which have properties we potentially have strong prior beliefs about. Additionally, data is often computationally expensive to acquire and very limited which makes it important to incorporate prior beliefs. The Bayesian approach is therefore natural to emulate computer models. *Bayesian emulators* are fast statistical constructs which approximate unknown functions by exploiting prior beliefs about the functions' smoothness and structure whilst quantifying uncertainty [1].

We will consider two main types of Bayesian emulators: the *Gaussian process emulator* and the *Bayes linear emulator*. Gaussian process (GP) emulators adopt the standard Bayes formalism and provide a full probabilistic specification over the function. This approach is widely used [22] and can be very powerful but requires a full probabilistic prior specification over the function. In contrast, Bayes linear (BL) emulators adopt the simpler Bayes linear approach which only requires the specification of second-order quantities. It can be argued that the Bayes linear approach is more principled for emulating computer models [13]. In this chapter, we first introduce Gaussian processes for emulation and then compare this to the Bayes linear approach.

2.1 Preliminaries

We are interested in modelling an expensive univariate computer model, which we treat as an unknown function $f : \mathcal{X} \to \mathbb{R}$. Unless otherwise stated, we will consider deterministic computer models and assume that we have run the computer model, f, at d points $X_D = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(d)}\}$. We write these in a design matrix,

$$\boldsymbol{X}_D = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(d)}\right)^{\mathsf{T}}. \tag{2.1}$$

We therefore know the values of f at these points and denote this as the vector

$$\mathbf{f}_D := f(\boldsymbol{X}_D) = \left(f(\boldsymbol{x}^{(1)}), \ldots, f(\boldsymbol{x}^{(d)})\right)^{\mathsf{T}}. \tag{2.2}$$

For this chapter, we will assume the input space is the unit N-cube, $\mathcal{X} = [0, 1]^N$. For many applications of emulation, this is a reasonable assumption, but in later chapters, we will consider more interesting spaces.

Remark 1. We can easily generalise to multi-output computer models, $\boldsymbol{f} = (f_1, \ldots, f_M)^{\mathsf{T}}$, by modelling each output separately.

2.2 Gaussian Process Emulation

Gaussian processes are flexible non-parametric Bayesian models [23] which are widely used for emulation [22, 24, 25] as well as machine learning [26, 27, 28], optimisation [29], time series analysis [30] and geostatistics (*kriging*) [31].

Treating computer models as unknown functions, we model $f(\boldsymbol{x})$ as a *stochastic process*. In comparison to Bayesian regression which performs inference over a vector space determined by finitely many basis functions, Gaussian processes perform inference directly in the function space [12]. Simply put, Gaussian processes define a distribution over f and the Gaussian nature makes it convenient to perform inference. In the next section, we will explore if enforcing the Gaussian nature is a sensible idea.

Whilst most applications of GPs consider noisy observations of f, we are concerned with deterministic computer models which give noise-free observations. We will now formally introduce Gaussian processes following [12].

Definition 2 (Gaussian process [12, 32])**.** *A stochastic process f, indexed by $\mathcal{X}$, is a* Gaussian process *if, for any finite set of inputs $\boldsymbol{X} \in \mathcal{X}^n$, the random vector $\mathbf{f} = f(\boldsymbol{X})$ is multivariate Gaussian.*

A Gaussian process is completely specified by its mean function $m(\boldsymbol{x})$ and its covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ [12] which are defined as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}(f(\boldsymbol{x})), \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}')), \end{aligned} \tag{2.3}$$

and we write the Gaussian process as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m, k). \tag{2.4}$$

Intuitively, covariance functions measure how similar we expect outputs to be at two different points. A simple covariance function is the isotropic version of the *squared exponential* covariance function. Different choices of covariance functions correspond to the GP representing different families of functions. In chapter 3, we will explore a variety of different choices for covariance functions.

Example 3. *The isotropic version of the squared exponential covariance function is given by,*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\theta^2}\right), \tag{2.5}$$

with correlation length θ *and variance parameter* σ^2*.*

Note that we write a finite set of inputs in a data matrix, $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^\mathsf{T} \in \mathcal{X}^n$, which is an $n \times N$ matrix where each row is an input $\boldsymbol{x} \in \mathcal{X}$. We can then define a covariance function acting on data matrices.

Definition 4 (Covariance matrix)**.** *Given a covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, we define an analogous function, $K : \mathcal{X}^n \times \mathcal{X}^m \to \mathcal{M}_{n,m}(\mathbb{R}_{\geq 0})$. Let $\boldsymbol{X}, \boldsymbol{X}'$ be data matrices, then*

$$K(\boldsymbol{X}, \boldsymbol{X}') = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}'_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}'_m) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}'_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}'_m) \end{bmatrix}. \tag{2.6}$$

We also write $K(\boldsymbol{X}) \equiv K(\boldsymbol{X}, \boldsymbol{X})$, although we often denote this as a variance matrix $\boldsymbol{\Sigma}$.

To illustrate the definition of a Gaussian process, explicitly consider the data matrix $\boldsymbol{X}_*$ of n inputs $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,N})^\mathsf{T} \in \mathcal{X}$. Since we are only considering finitely many inputs, the distribution of $\mathbf{f}_* := f(\boldsymbol{X}_*)$ is multivariate Gaussian,

$$\mathbf{f}_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

with mean $\boldsymbol{\mu}_* = m(\boldsymbol{X}_*) = (m(\boldsymbol{x}_1), \ldots, m(\boldsymbol{x}_n))^\mathsf{T}$ and covariance matrix $\boldsymbol{\Sigma}_* = K(\boldsymbol{X}_*)$. This gives a prior distribution over f at the n points. However, we would like to obtain a posterior distribution for $\mathbf{f}_*$ after observing f at a set of *design points* $\boldsymbol{X}_D$.

2.2.1 Conditioning on Observations

We assume we have observed f at design points $\boldsymbol{X}_D$ to be $\mathbf{f}_D = \boldsymbol{D}$ and are interested in finding the posterior distribution $f \mid \boldsymbol{D}$. Specifically, we will consider $\mathbf{f}_* = f(\boldsymbol{X}_*)$ for arbitrary points $\boldsymbol{X}_*$. From the definition of a GP, the joint prior for $\mathbf{f}_*$ and $\mathbf{f}_D$ is multivariate Gaussian,

$$\begin{bmatrix}\mathbf{f}_* \\ \mathbf{f}_D\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_* \\ \boldsymbol{\mu}_D\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_* & K(\boldsymbol{X}_*, \boldsymbol{X}_D) \\ K(\boldsymbol{X}_D, \boldsymbol{X}_*) & \boldsymbol{\Sigma}_D\end{bmatrix}\right), \tag{2.7}$$

where we explicitly write $\boldsymbol{\Sigma}_* \equiv K(\boldsymbol{X}_*, \boldsymbol{X}_*)$ and $\boldsymbol{\Sigma}_D \equiv K(\boldsymbol{X}_D, \boldsymbol{X}_D)$.

Exploiting the Gaussian structure, we can now analytically derive the posterior distribution.

Theorem 5 (GP predictive distribution [12])**.** *Given observations, $\mathbf{f}_D = \boldsymbol{D}$, of the computer model at points $\boldsymbol{X}_D$, the posterior distribution, $\mathbf{f}_*$, at arbitrary points $\boldsymbol{X}_*$ is also Gaussian distributed,*

$$\mathbf{f}_* \mid \mathbf{f}_D = \boldsymbol{D} \sim \mathcal{N}\left(\boldsymbol{\mu}_{*|D}, \boldsymbol{\Sigma}_{*|D}\right), \tag{2.8}$$

with posterior mean and variance given by,

$$\boldsymbol{\mu}_{*|D} = \boldsymbol{\mu}_* + K(\boldsymbol{X}_*, \boldsymbol{X}_D)\boldsymbol{\Sigma}_D^\dagger(\boldsymbol{D} - \boldsymbol{\mu}_D), \tag{2.9}$$

$$\boldsymbol{\Sigma}_{*|D} = \boldsymbol{\Sigma}_* - K(\boldsymbol{X}_*, \boldsymbol{X}_D)\boldsymbol{\Sigma}_D^\dagger K(\boldsymbol{X}_D, \boldsymbol{X}_*). \tag{2.10}$$

Proof. Adapted from [33, 34, 35], we explicitly condition on the prior. For notational convenience, we write $\boldsymbol{\Sigma}_{12} \equiv K(\boldsymbol{X}_D, \boldsymbol{X}_*)$ and $\boldsymbol{\Sigma}_{21} \equiv K(\boldsymbol{X}_*, \boldsymbol{X}_D) = \boldsymbol{\Sigma}_{12}^\mathsf{T}$. To make the conditioning step easier, we first rewrite the squared Mahalanobis distance. For this, we use the Schur complement [36], $\boldsymbol{\Sigma}_{*|D} \equiv \boldsymbol{\Sigma}_* - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger\boldsymbol{\Sigma}_{21}$ and the matrix blockwise inversion formula (A.6),

$$\boldsymbol{\Sigma}^\dagger = \begin{bmatrix}\boldsymbol{\Sigma}_* & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_D\end{bmatrix}^\dagger =: \begin{bmatrix}\boldsymbol{\Sigma}^* & \boldsymbol{\Sigma}^{12} \\ \boldsymbol{\Sigma}^{21} & \boldsymbol{\Sigma}^D\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Sigma}_*^\dagger & -\boldsymbol{\Sigma}_{*|D}^\dagger\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger \\ -\boldsymbol{\Sigma}_D^\dagger\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{*|D}^\dagger & \boldsymbol{\Sigma}_D^{-1} + \boldsymbol{\Sigma}_D^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{*|D}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^{-1}\end{bmatrix} \tag{2.11}$$

where $\boldsymbol{\Sigma}^{21} = \boldsymbol{\Sigma}^{12\mathsf{T}}$ since $\boldsymbol{\Sigma}^\dagger$ is symmetric. Using this, we write the squared Mahalanobis distance as

$$\begin{aligned}
(\mathbf{f} - \boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^\dagger(\mathbf{f} - \boldsymbol{\mu}) &= \begin{bmatrix}\mathbf{f}_* - \boldsymbol{\mu}_* \\ \mathbf{f}_D - \boldsymbol{\mu}_D\end{bmatrix}^\mathsf{T}\begin{bmatrix}\boldsymbol{\Sigma}^* & \boldsymbol{\Sigma}^{12} \\ \boldsymbol{\Sigma}^{21} & \boldsymbol{\Sigma}^D\end{bmatrix}\begin{bmatrix}\mathbf{f}_* - \boldsymbol{\mu}_* \\ \mathbf{f}_D - \boldsymbol{\mu}_D\end{bmatrix} \\
&= (\mathbf{f}_* - \boldsymbol{\mu}_*)^\mathsf{T}\boldsymbol{\Sigma}^*(\mathbf{f}_* - \boldsymbol{\mu}_*) + 2(\mathbf{f}_* - \boldsymbol{\mu}_*)^\mathsf{T}\boldsymbol{\Sigma}^{12}(\mathbf{f}_D - \boldsymbol{\mu}_D) + (\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}^D(\mathbf{f}_D - \boldsymbol{\mu}_D) \\
&= (\mathbf{f}_* - \boldsymbol{\mu}_*)^\mathsf{T}\boldsymbol{\Sigma}_{*|D}^\dagger(\mathbf{f}_* - \boldsymbol{\mu}_*) - 2(\mathbf{f}_* - \boldsymbol{\mu}_*)^\mathsf{T}\boldsymbol{\Sigma}_{*|D}^\dagger\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D) \\
&\quad + (\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D) + (\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}_D^\dagger\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{*|D}^\dagger\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D) \\
&= (\mathbf{f}_* - (\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D)))^\mathsf{T}\boldsymbol{\Sigma}_{*|D}^\dagger(\mathbf{f}_* - (\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D))) \\
&\quad + (\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D) \\
&= (\mathbf{f}_* - \boldsymbol{\mu}_{*|D})^\mathsf{T}\boldsymbol{\Sigma}_{*|D}^\dagger(\mathbf{f}_* - \boldsymbol{\mu}_{*|D}) + (\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_D).
\end{aligned} \tag{2.12}$$

Here we have substituted in $\boldsymbol{\mu}_{*|D} = \boldsymbol{\mu}_D + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_D^\dagger(\mathbf{f}_D - \boldsymbol{\mu}_*)$ since we write $\mathbf{f}_D = \boldsymbol{D}$ after observing $\mathbf{f}_D$. Now the joint prior is $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and using the law of conditional probability,

$$\begin{aligned}
p(\mathbf{f}_* \mid \mathbf{f}_D) &= \frac{p(\mathbf{f}_D, \mathbf{f}_*)}{p(\mathbf{f}_D)} = \frac{\mathcal{N}(\mathbf{f}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\mathcal{N}(\mathbf{f}_D; \boldsymbol{\mu}_D, \boldsymbol{\Sigma}_D)} \\
&= \frac{(2\pi)^{-\frac{d+n}{2}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{f} - \boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu})\right)}{(2\pi)^{-\frac{d}{2}}|\boldsymbol{\Sigma}_D|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}_D^{-1}(\mathbf{f}_D - \boldsymbol{\mu}_D)\right)} \\
&\propto \exp\left(-\frac{1}{2}(\mathbf{f} - \boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu}) + \frac{1}{2}(\mathbf{f}_D - \boldsymbol{\mu}_D)^\mathsf{T}\boldsymbol{\Sigma}_D^{-1}(\mathbf{f}_D - \boldsymbol{\mu}_D)\right).
\end{aligned} \tag{2.13}$$

Now finally, substituting (2.12) into (2.13) we get,

$$p(\mathbf{f}_* \mid \mathbf{f}_D) \propto \exp\left(-\frac{1}{2}(\mathbf{f}_* - \boldsymbol{\mu}_{*|D})^\mathsf{T} \boldsymbol{\Sigma}_{*|D}^\dagger (\mathbf{f}_* - \boldsymbol{\mu}_{*|D})\right) \tag{2.14}$$

which we identify as the kernel for a normal density, so $\mathbf{f}_* \mid D \sim \mathcal{N}\left(\boldsymbol{\mu}_{*|D}, \boldsymbol{\Sigma}_{*|D}\right)$. ■

Remark 6. $\boldsymbol{A}^\dagger$ denotes the MoorePenrose generalised inverse of $\boldsymbol{A}$ [37].

Since Theorem 5 holds for an arbitrary set of points $\boldsymbol{X}_*$, an immediate consequence is that the posterior of a Gaussian process is also a Gaussian process.

Corollary 7. *The posterior of a Gaussian process $f \sim \mathcal{GP}(m, k)$ is also a Gaussian process,*

$$f \mid D \sim \mathcal{GP}(m_D, k_D) \tag{2.15}$$

with posterior mean function $m_D(\boldsymbol{x})$ and posterior covariance function $k_D(\boldsymbol{x}, \boldsymbol{x}')$ defined as

$$\begin{aligned} m_D(\boldsymbol{x}) &= m(\boldsymbol{x}) + K(\boldsymbol{x}, \boldsymbol{X}_D)\boldsymbol{\Sigma}_D^\dagger(\boldsymbol{D} - m(\boldsymbol{D})), \\ k_D(\boldsymbol{x}, \boldsymbol{x}') &= k(\boldsymbol{x}, \boldsymbol{x}') - K(\boldsymbol{x}, \boldsymbol{X}_D)\boldsymbol{\Sigma}_D^\dagger K(\boldsymbol{X}_D, \boldsymbol{x}'), \end{aligned} \tag{2.16}$$

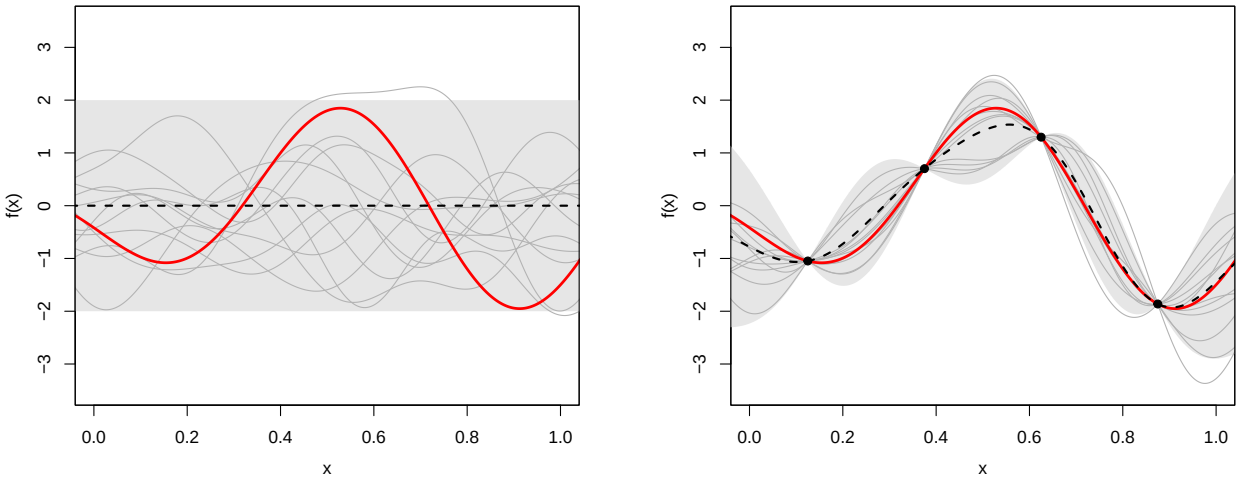
where $\boldsymbol{\Sigma}_D = K(\boldsymbol{X}_D)$.

We now consider a simple example of a 1D GP emulator.

Example 8 (1D GP emulator)**.** *Consider a simple trigonometric function on $\mathcal{X} = [0, 1]$,*

$$f(x) = \sin(2\pi x + 5) + \cos(3\pi x + 1).$$

Assuming we want to emulate this function, we can model f as a GP with zero-mean, $m(x) = 0$, and a squared exponential covariance function (3) with correlation length $\theta = 0.2$. Figure 2.1 shows how observing f at four points collapses the uncertainty at those points and even away from the observed points, the emulator makes a reasonable prediction.



(a) Prior (b) Posterior

Figure 2.1: GP prior (left) and posterior (right) from Example 8. In each panel, the red line is the true function $f(x)$, the dashed black line is the expectation of the GP and the light grey region is a 2σ interval corresponding to a 95% confidence region. The thin grey lines are 12 functions sampled at random from the GP.

Before we look at examples of emulators in higher dimensions, we will argue why the assumptions of the GP emulator may not always be reasonable. We will introduce an alternative emulator, the Bayes linear emulator which approaches emulation from the Bayes linear perspective.

2.3 Bayes Linear Methods

In the Bayesian formalism, the posterior is a full probability distribution, but this comes at the expense of needing to fully specify prior probability distributions. Taking Gaussian priors makes inference convenient, but it is rarely ever faithful to the Bayesian paradigm of an honest prior belief specification. Even in small problems, it may be impossible to fully specify our prior beliefs in the form of a full probability distribution. In [13], Goldstein and Wooff argue that the fully Bayesian approach falls victim to its ambition. In [38, 39], de Finetti proposes expectation as a primitive, rather than probability for subjectivist theory. The Bayes linear framework takes this approach and only requires second-order quantities to be specified. With a simpler belief specification, we can hope they are more honest representations of reality.

Before introducing Bayes linear emulators, we will briefly explore the Bayes linear framework more broadly. This section will explore what the Bayes linear approach is, how we specify our prior beliefs, and how we update our beliefs after observing data.

In this section, we follow the theory introduced in [13]. We will assume C is a collection of random quantities. We will denote a set of observed values as $D = \{D_1, \ldots, D_k\} \subset C$ and a set of unknown values as $B = \{B_1, \ldots, B_r\} \subset C$ with the vectors of random quantities in bold: $\boldsymbol{D}$ and $\boldsymbol{B}$. We want to predict the unknown values B given the observed values D with uncertainty. Core to the Bayes linear approach is that we specify our prior beliefs via second-order quantities:

- Expectations: $\mathbb{E}(\boldsymbol{B})$ and $\mathbb{E}(\boldsymbol{D})$.
- Variances and covariances: $\mathrm{Var}(\boldsymbol{D})$, $\mathrm{Var}(\boldsymbol{B})$ and $\mathrm{Cov}(\boldsymbol{D}, \boldsymbol{B})$.

In the next section, we will consider computer model outputs as our collection C (indexed by the input space $\mathcal{X}$), but in general, C can take many different forms [13]. To illustrate the ideas in this section, we will consider a simple example throughout.

Example 9. *Consider the height in feet of waves at four different beaches: W_1, W_2, W_3 and W_4. Our prior belief is that $\mathbb{E}(W_i) = 6$ for all i and the covariance matrix, $\boldsymbol{\Sigma}$ where $\boldsymbol{\Sigma}_{i,j} = \mathrm{Cov}(W_i, W_j)$, is,*

$$\boldsymbol{\Sigma} = \begin{bmatrix} 2.0 & -0.6 & 1.4 & -1.0 \\ -0.6 & 2.0 & -1.4 & 1.6 \\ 1.4 & -1.4 & 2.0 & -1.2 \\ -1.0 & 1.6 & -1.2 & 2.0 \end{bmatrix}.$$

We observe the values $\boldsymbol{D} = (W_1, W_2)^\mathsf{T} = (7.4, 4.1)^\mathsf{T}$ and aim to predict the values $\boldsymbol{B} = (W_3, W_4)^\mathsf{T}$.

The Bayes linear framework considers linear estimates of the unobserved variables B in terms of D. This approach enables tractable analytical solutions while still capturing the essential dependencies between variables. It can be observed that the GP posterior predictive mean (2.9) is a linear function of the observations D. Informally, this supports the notion of using linear estimates. To be precise, we use the *minimum variance unbiased* linear estimate of the unobserved variables, which we call the *adjusted expectation*.

Definition 10 (Adjusted expectation [13])**.** *The adjusted expectation of a random quantity X given D, is the linear combination,*

$$\mathbb{E}_D(X) = c_0 + \sum_{i=1}^{k} c_i D_i, \tag{2.17}$$

which minimises the expected squared error loss,

$$\mathbb{E}\left(\left[X - c_0 - \sum_{i=1}^{k} c_i D_i\right]^2\right). \tag{2.18}$$

This definition extends naturally to a vector of random quantities $\boldsymbol{B}$, and we can write $\mathbb{E}_D(\boldsymbol{B})$ in terms of second-order quantities.

Theorem 11 (Adjusted expectation [13])**.** *The adjusted expectation of random quantities $\boldsymbol{B}$ given D is,*

$$\mathbb{E}_D(\boldsymbol{B}) = \mathbb{E}(\boldsymbol{B}) + \operatorname{Cov}(\boldsymbol{B}, \boldsymbol{D}) \operatorname{Var}(\boldsymbol{D})^\dagger (\boldsymbol{D} - \mathbb{E}(\boldsymbol{D})) \tag{2.19}$$

Proof. It is sufficient to show this for a single random quantity X. Following [13], let $\boldsymbol{c}_* = (c_1, \ldots, c_k)^\mathsf{T}$ and let $T = \mathbb{E}([X - c_0 - \boldsymbol{c}_*^\mathsf{T}\boldsymbol{D}]^2)$ denote the squared error loss. The value of c_0 minimising T satisfies

$$\begin{aligned} \frac{\partial T}{\partial c_0} &= \mathbb{E}\left[2\left(X - c_0 - \boldsymbol{c}_*^\mathsf{T}\boldsymbol{D}\right)(-1)\right] \stackrel{!}{=} 0 \\ \implies c_0 &= \mathbb{E}(X) - \boldsymbol{c}_*^\mathsf{T}\,\mathbb{E}(\boldsymbol{D}). \end{aligned}$$

Substituting this into T gives $T = \mathbb{E}([(X - \mathbb{E}[X]) - \boldsymbol{c}_*^\mathsf{T}(\boldsymbol{D} - \mathbb{E}[\boldsymbol{D}])]^2)$. Rearranging and differentiating,

$$\begin{aligned} T &= \mathbb{E}([(X - \boldsymbol{c}_*^\mathsf{T}\boldsymbol{D}) - \mathbb{E}(X - \boldsymbol{c}_*^\mathsf{T}\boldsymbol{D})]^2) = \operatorname{Var}(X - \boldsymbol{c}_*^\mathsf{T}\boldsymbol{D}) \\ &= \operatorname{Var}(X) + \boldsymbol{c}_*^\mathsf{T} \operatorname{Var}(\boldsymbol{D})\boldsymbol{c}_* - 2\boldsymbol{c}_*^\mathsf{T} \operatorname{Cov}(\boldsymbol{D}, X) \\ \nabla_{\boldsymbol{c}_*} T &= (\operatorname{Var}(\boldsymbol{D}) + \operatorname{Var}(\boldsymbol{D})^\mathsf{T})\boldsymbol{c}_* - 2\operatorname{Cov}(\boldsymbol{D}, X) = 2(\operatorname{Var}(\boldsymbol{D})\boldsymbol{c}_* - \operatorname{Cov}(\boldsymbol{D}, X)) \stackrel{!}{=} 0 \\ \implies \boldsymbol{c}_* &= \operatorname{Var}(\boldsymbol{D})^\dagger \operatorname{Cov}(\boldsymbol{D}, X) \iff \boldsymbol{c}_*^\mathsf{T} = \operatorname{Cov}(X, \boldsymbol{D}) \operatorname{Var}(\boldsymbol{D})^\dagger. \end{aligned}$$

Finally, we can substitute the values of c_0 and $\boldsymbol{c}_*$ into (2.17) to obtain the desired result. ■

The adjusted expectation behaves as we would expect. For example, it is linear and conglomerable.

Lemma 12. *The adjusted expectation has the following properties [13]:*

1. *Linear:* $\mathbb{E}_D(\alpha_1\boldsymbol{B}_1 + \alpha_2\boldsymbol{B}_2) = \alpha_1\,\mathbb{E}_D(\boldsymbol{B}_1) + \alpha_2\,\mathbb{E}_D(\boldsymbol{B}_2)$.
2. *Conglomerable:* $\mathbb{E}(\mathbb{E}_D(\boldsymbol{B})) = \mathbb{E}(\boldsymbol{B})$.

Proof.

1. Linearity follows from Theorem 11 and linearity properties of expectation and covariance,

$$\begin{aligned} \mathbb{E}_D(\alpha_1\boldsymbol{B}_1 + \alpha_2\boldsymbol{B}_2) &= \mathbb{E}(\alpha_1\boldsymbol{B}_1 + \alpha_2\boldsymbol{B}_2) + \operatorname{Cov}(\alpha_1\boldsymbol{B}_1 + \alpha_2\boldsymbol{B}_2, \boldsymbol{D}) \operatorname{Var}(\boldsymbol{D})^\dagger (\boldsymbol{D} - \mathbb{E}(\boldsymbol{D})) \\ &= \alpha_1\,\mathbb{E}(\boldsymbol{B}_1) + \alpha_2 E(\boldsymbol{B}_2) \\ &\qquad + [\alpha_1 \operatorname{Cov}(\boldsymbol{B}_1, \boldsymbol{D}) + \alpha_2 \operatorname{Cov}(\boldsymbol{B}_2, \boldsymbol{D})] \operatorname{Var}(\boldsymbol{D})^\dagger (\boldsymbol{D} - \mathbb{E}(\boldsymbol{D})) \\ &= \alpha_1\,\mathbb{E}_D(\boldsymbol{B}_1) + \alpha_2 E_D(\boldsymbol{B}_2) \end{aligned}$$

2. Conglomerability also follows from Theorem 11,

$$\begin{aligned} \mathbb{E}(E_D(X)) &= \mathbb{E}\left[\mathbb{E}(X) + \operatorname{Cov}(X, \boldsymbol{D}) \operatorname{Var}(\boldsymbol{D})^\dagger (\boldsymbol{D} - \mathbb{E}(\boldsymbol{D}))\right] \\ &= \mathbb{E}(X) + \mathbb{E}\left[\operatorname{Cov}(X, \boldsymbol{D}) \operatorname{Var}(\boldsymbol{D})^\dagger (\boldsymbol{D} - \mathbb{E}(\boldsymbol{D}))\right] \\ &= \mathbb{E}(X) + \operatorname{Cov}(X, \boldsymbol{D}) \operatorname{Var}(\boldsymbol{D})^\dagger (\cancel{\mathbb{E}(\boldsymbol{D}) - \mathbb{E}(\boldsymbol{D})}) \\ &= \mathbb{E}(X). \end{aligned}$$

■

Example 9 (continuing from p. 7)**.** *Using 2.19, we can now explicitly calculate the adjusted expectation for $\boldsymbol{B} = (W_3, W_4)^\mathsf{T}$,*

$$\mathbb{E}_D(\boldsymbol{B}) = \begin{pmatrix} 6.0 \\ 6.0 \end{pmatrix} + \begin{pmatrix} 1.4 & -1.4 \\ -1.0 & 1.6 \end{pmatrix} \begin{pmatrix} 2.0 & -0.6 \\ -0.6 & 2.0 \end{pmatrix}^{-1} \left(\begin{pmatrix} 7.4 \\ 4.1 \end{pmatrix} - \begin{pmatrix} 6.0 \\ 6.0 \end{pmatrix} \right) \approx \begin{pmatrix} 6.40 \\ 4.24 \end{pmatrix}.$$

Definition 13 (Adjusted version [13]). *The adjusted version of $\boldsymbol{B}$ given D is the* residual *vector*

$$\mathbb{A}_D(\boldsymbol{B}) = \boldsymbol{B} - \mathbb{E}_D(\boldsymbol{B}) \tag{2.20}$$

.

The adjusted expectation (10) minimises the expectation of the adjusted version. We define the adjusted variance to be the variance of the adjusted version.

Definition 14 (Adjusted variance [13]). *The adjusted variance,* $\mathrm{Var}_D(\boldsymbol{B})$*, of $\boldsymbol{B}$ given D is defined as*

$$\mathrm{Var}_D(\boldsymbol{B}) = \mathbb{E}\left([\boldsymbol{B} - \mathbb{E}_D(\boldsymbol{B})]^2\right) = \mathrm{Var}(\mathbb{A}_D(\boldsymbol{B})). \tag{2.21}$$

Intuitively, this measures the variance in the adjusted expectation prediction $\mathbb{E}_D(\boldsymbol{B})$. Similarly to the adjusted expectation, the adjusted variance can be calculated with linear updates.

Theorem 15 (Adjusted variance [13]). *The adjusted variance of random quantities $\boldsymbol{B}$ given D is*

$$\mathrm{Var}_D(\boldsymbol{B}) = \mathrm{Var}(\boldsymbol{B}) - \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger\,\mathrm{Cov}(\boldsymbol{D}, \boldsymbol{B}) \tag{2.22}$$

Proof. We first partition $B = \mathbb{E}_D(\boldsymbol{B}) + \mathbb{A}_D(\boldsymbol{B})$ so that $\mathrm{Var}(B) = \mathrm{Var}(\mathbb{E}_D(\boldsymbol{B})) + \mathrm{Var}(\mathbb{A}_D(\boldsymbol{B})) + 2\,\mathrm{Cov}(\mathbb{E}_D(\boldsymbol{B}), \mathbb{A}_D(\boldsymbol{B}))$. First computing $\mathrm{Var}(\mathbb{E}_D(\boldsymbol{B}))$,

$$\begin{aligned}
\mathrm{Var}(\mathbb{E}_D(\boldsymbol{B})) &= \mathrm{Var}(\cancel{\mathbb{E}(\boldsymbol{B})}^{\text{const.}} + \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger(\boldsymbol{D} - \cancel{\mathbb{E}(\boldsymbol{D})}^{\text{const.}})) \\
&= \mathrm{Var}(\mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger \boldsymbol{D}) \\
&= \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger\,\mathrm{Var}(\boldsymbol{D})(\mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger)^\mathsf{T} \\
&= \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger\,\mathrm{Cov}(\boldsymbol{D}, \boldsymbol{B})
\end{aligned}$$

since $\mathrm{Var}(\boldsymbol{D})^\dagger$ is symmetric. Now we compute $\mathrm{Cov}(\mathbb{E}_D(\boldsymbol{B}), \mathbb{A}_D(\boldsymbol{B}))$,

$$\begin{aligned}
\mathrm{Cov}(\mathbb{E}_D(\boldsymbol{B}), \mathbb{A}_D(\boldsymbol{B})) &= \mathrm{Cov}(\mathbb{E}_D(\boldsymbol{B}), \boldsymbol{B} - \mathbb{E}_D(\boldsymbol{B})) \\
&= \mathrm{Cov}(\mathbb{E}_D(\boldsymbol{B}), \boldsymbol{B}) - \mathrm{Cov}(\mathbb{E}_D(\boldsymbol{B}), \mathbb{E}_D(\boldsymbol{B})) \\
&= \mathrm{Cov}(\cancel{\mathbb{E}(\boldsymbol{B})}^{\text{const.}} + \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger(\boldsymbol{D} - \cancel{\mathbb{E}(\boldsymbol{D})}^{\text{const.}}), \boldsymbol{B}) - \mathrm{Var}(\mathbb{E}_D(\boldsymbol{B})) \\
&= \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger\,\mathrm{Cov}(\boldsymbol{D}, \boldsymbol{B}) - \mathrm{Var}(\mathbb{E}_D(\boldsymbol{B})) \\
&= 0
\end{aligned}$$

using $\mathrm{Var}(\mathbb{E}_D(\boldsymbol{B}))$ calculated above. We can then find

$$\mathrm{Var}_D(\boldsymbol{B}) \equiv \mathrm{Var}(\mathbb{A}_D(\boldsymbol{B})) = \mathrm{Var}(\boldsymbol{B}) - \mathrm{Cov}(\boldsymbol{B}, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger\,\mathrm{Cov}(\boldsymbol{D}, \boldsymbol{B}).$$

■

Remark 16. By considering the block-wise formula for $\mathrm{Var}_D\left[(\boldsymbol{B}_1, \boldsymbol{B}_2)^\mathsf{T}\right]$, we can easily find the adjusted covariance between $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$,

$$\mathrm{Cov}_D(\boldsymbol{B}_1, \boldsymbol{B}_2) = \mathrm{Cov}(\boldsymbol{B}_1, \boldsymbol{B}_2) - \mathrm{Cov}(\boldsymbol{B}_1, \boldsymbol{D})\,\mathrm{Var}(\boldsymbol{D})^\dagger\,\mathrm{Cov}(\boldsymbol{D}, \boldsymbol{B}_2). \tag{2.23}$$

Continuing our working example for this section, we can now calculate the adjusted variance.

Example 9 (continuing from p. 7). *Using 2.22, we calculate the adjusted variance for $\boldsymbol{B} = (W_3, W_4)^\mathsf{T}$,*

$$\mathrm{Var}_D(\boldsymbol{B}) = \begin{pmatrix} 2.0 & -1.2 \\ -1.2 & 2.0 \end{pmatrix} - \begin{pmatrix} 1.4 & -1.4 \\ -1.0 & 1.6 \end{pmatrix} \begin{pmatrix} 2.0 & -0.6 \\ -0.6 & 2.0 \end{pmatrix}^{-1} \begin{pmatrix} 1.4 & -1.0 \\ -1.4 & 1.6 \end{pmatrix} \approx \begin{pmatrix} 0.49 & 0.20 \\ 0.20 & 0.57 \end{pmatrix}.$$

So an approximate 3σ prediction interval would be $[4.3, 8.5]$ for W_1 and $[2.0, 6.5]$ for W_2.

Remark 17. We can notice that the Bayes linear adjusted expectation (2.19) and adjusted variance (2.22) take an identical form to the posterior mean (2.9) and variance (2.10) in the GP predictive distribution. So a GP and BL emulator will take identical forms but with different interpretations.

Remark 18. The wave heights in Example 9 must be non-negative to make sense ($W_i \geq 0$). Enforcing a normal structure would be unfit for this situation. In general, we would like to avoid enforcing a structure which we are not sure is correct.

A desirable attribute of this framework is that updating beliefs only requires linear fitting rather than conditioning. Whilst this is true for GPs also, this is only due to the Gaussian structure which we would like to avoid.

Remark 19. There are many diagnostics in the Bayes linear framework which can be used to check if observations are consistent with prior beliefs. Definition 23 in the next section is a simple example.

2.4 Bayes Linear Emulation

Bayes linear emulation is similar to GP emulation and is still subjectivist, but is less fully specified, employing Bayes linear methods. For many applications of emulators to real computer models, not enforcing a strict probabilistic specification is desirable. In later chapters, we will focus on Bayes linear emulation, but in principle, much of the theory also applies to GP emulation.

We are interested in emulating a computer model f at n points simultaneously and write these points in a matrix,

$$\boldsymbol{X}_* = \left(\boldsymbol{x}_*^{(1)}, \ldots, \boldsymbol{x}_*^{(n)}\right)^\mathsf{T}. \tag{2.24}$$

Similarly to our notation for $\mathbf{f}_D$, we also write

$$\mathbf{f}_* := f(\boldsymbol{X}_*) = \left(f(\boldsymbol{x}_*^{(1)}), \ldots, f(\boldsymbol{x}_*^{(n)})\right)^\mathsf{T} \tag{2.25}$$

which is the unknown quantity we want to perform inference on. We will first consider a simple Bayes linear emulator $f(\boldsymbol{x}) = u(\boldsymbol{x})$, where $u(\boldsymbol{x})$ is a weakly stationary stochastic process.

Definition 20 (Weakly stationary stochastic process [40])**.** *A stochastic process $u(\boldsymbol{x})$ is weakly stationary if for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, the following conditions hold:*

1. $\mathbb{E}(u(\boldsymbol{x})) = m$ *(constant mean).*
2. $\mathrm{Cov}(u(\boldsymbol{x}), u(\boldsymbol{x}')) = r(\boldsymbol{x} - \boldsymbol{x}')$ *(translation invariant).*
3. $\mathbb{E}(u(\boldsymbol{x})^2) < \infty$ *(finite second moment).*

We therefore must specify a prior mean m and a stationary covariance function $k(\boldsymbol{x}, \boldsymbol{x}') = r(\boldsymbol{x} - \boldsymbol{x}')$ and we can then update our beliefs. With our notation, we can express this in the following corollary which follows directly from Theorem 11 and Theorem 15 with $\boldsymbol{B} = \mathbf{f}_* \equiv f(\boldsymbol{X}_*)$.

Corollary 21 (Simple Bayes linear emulator [41])**.** *For an unknown function f and a design matrix $\boldsymbol{X}_D$, the Bayes linear emulator adjusted expectation for f at points $\boldsymbol{X}_*$ is given by,*

$$\mathbb{E}_D(\mathbf{f}_*) = \mathbb{E}(\mathbf{f}_*) + \mathrm{Cov}(\mathbf{f}_*, \mathbf{f}_D)\,\mathrm{Var}(\mathbf{f}_D)^\dagger(\mathbf{f}_D - \mathbb{E}(\mathbf{f}_D)), \tag{2.26}$$

and the simple Bayes linear emulator adjusted variance is given by,

$$\mathrm{Var}_D(\mathbf{f}_*) = \mathrm{Var}(\mathbf{f}_*) - \mathrm{Cov}(\mathbf{f}_*, \mathbf{f}_D)\,\mathrm{Var}(\mathbf{f}_D)^\dagger\,\mathrm{Cov}(\mathbf{f}_D, \mathbf{f}_*). \tag{2.27}$$

Remark 22. Notice how the BL emulator expectation (2.26) and variance (2.27) equations take an identical form to the GP emulator posterior predictive mean (2.9) and variance (2.10). This means the expectation and variance of a simple BL emulator will be identical to those of a GP emulator.

If we ever observe the values of f at the points $\boldsymbol{X}_*$, we can use some simple diagnostics to see if our updated beliefs align with the observed values.

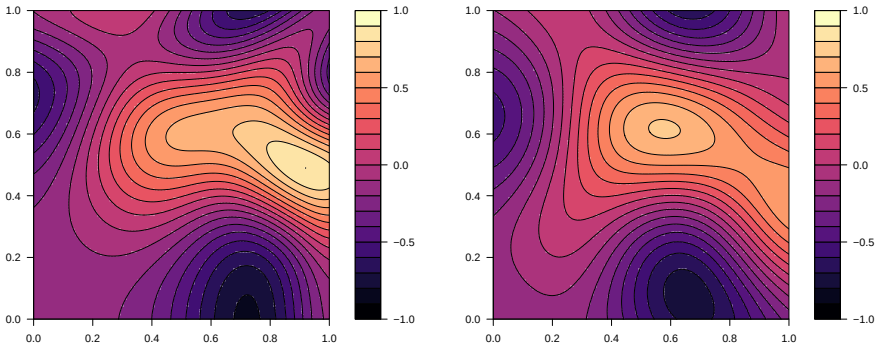
Definition 23 (Simple diagnostic [41])**.** *The univariate standardised residual of f at x is given as,*

$$S_D(f(x)) = \frac{f(x) - \mathbb{E}_D(f(x))}{\sqrt{\mathrm{Var}_D(f(x))}}. \tag{2.28}$$

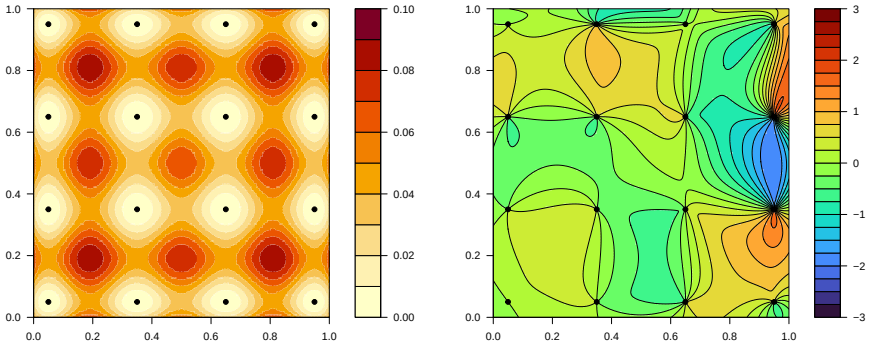
Pukelsheim's *three-sigma rule* [42] states that the probability of a random variable falling more than three standard deviations from its mean is less than 5%, assuming *any unimodal distribution*. Therefore, we can expect the standardised residual to not exceed 3 in absolute value more than 5% of the time. If this diagnostic takes extreme values regularly, it may suggest that there is a discrepancy between our prior beliefs and the observed values.

Remark 24. Pukelsheim's three-sigma rule only assumes that the distribution is unimodal [42]. Given that we are using a stochastic process to express uncertainty in a deterministic function, this is a reasonable assumption. With further distributional assumptions, we can achieve a tighter bound. Namely, with a normality assumption like in GP emulation, we can reduce the bound to two standard deviations. However, the Bayes linear framework helps us avoid making strong distributional assumptions like this.

Example 25 (2D Bayes linear emulation)**.** *Suppose we want to emulate the 2D function $f(\boldsymbol{x})$ shown in Figure 2.2a (given explicitly in the appendix B.1). Although, unlike most computer models, we can evaluate f quickly, we will try and emulate f with limited points as a demonstrative example. Specifically, we choose a simple grid of 16 points; we will see later why this choice may not be the best when we explore different designs. Figure 2.2 shows the emulator expectation (b) along with the emulator variance (c) and diagnostics (d).*



(a) True function, $f(\boldsymbol{x})$. (b) Emulator expectation, $\mathbb{E}_D(f(\boldsymbol{x}))$.



(c) Emulator variance, $\mathrm{Var}_D(f(\boldsymbol{x}))$. (d) Emulator diagnostics, $S_D(f(\boldsymbol{x}))$.

Figure 2.2: 2D Bayes linear emulator for $f(\boldsymbol{x})$ (B.1) as in Example 25.

Chapter 3

Covariance Functions

In the previous chapter, we saw that a crucial aspect of specifying prior beliefs for an emulator is specifying the covariance structure. Emulators exploit the notion that close points will have similar outputs, covariance functions define this notion of closeness [12]. So far, we have seen the isotropic form of the squared exponential covariance function (Example 3) which is a popular choice for functions which are believed to be infinitely differentiable [12]. In this section, we will introduce a variety of commonly used covariance structures for input spaces $\mathcal{X} \subset \mathbb{R}^N$ which reflect different prior beliefs about a function.

Recall a covariance function is defined as $k(\boldsymbol{x}, \boldsymbol{x}') := \text{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}'))$. We can immediately observe any covariance function must be symmetric, but not all symmetric functions $k(\boldsymbol{x}, \boldsymbol{x}')$ define a valid covariance structure [12].

Remark 26. We refer to an arbitrary symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as a *kernel*.

Definition 27 (Gram matrix [12])**.** *Given an arbitrary kernel $k(\cdot, \cdot)$ and a finite set of input points $\{\boldsymbol{x}_i\}_{i=1}^n$, we define the* Gram matrix *K whose entries are $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.*

Definition 28 (Positive semi-definite [43])**.** *A real matrix $\boldsymbol{M} \in \mathcal{M}_n(\mathbb{R})$ is positive semi-definite if for all $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a}^\mathsf{T} \boldsymbol{M} \boldsymbol{a} \geq 0$.*

A convenient way to check if a symmetric matrix is positive semi-definite is to check the sign of its eigenvalues.

Lemma 29. *A symmetric matrix $\boldsymbol{M} \in \mathcal{M}_n(\mathbb{R})$ is positive semi-definite if and only if all eigenvalues λ_i are non-negative [43].*

Proof. Following [43], we perform an eigendecomposition on $\boldsymbol{M} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$, where $\boldsymbol{D}$ is diagonal with eigenvalue entries and $\boldsymbol{P}$ is an orthogonal matrix. We can view $\boldsymbol{P}$ as an invertible linear transformation (mapping to eigenvector basis), and considering the change of variable $\boldsymbol{b} = \boldsymbol{P}^\mathsf{T} \boldsymbol{a}$ then,

$$\begin{aligned} & \boldsymbol{a}^\mathsf{T} \boldsymbol{M} \boldsymbol{a} = \boldsymbol{a}^\mathsf{T} \boldsymbol{P} \boldsymbol{D} \boldsymbol{P}^\mathsf{T} \boldsymbol{a} \geq 0 && \forall\ \boldsymbol{a} \in \mathbb{R}^n \\ \iff \quad & \boldsymbol{b}^\mathsf{T} \boldsymbol{D} \boldsymbol{b} \geq 0 && \forall\ \boldsymbol{b} \in \mathbb{R}^n \end{aligned}$$

and since $\boldsymbol{D}$ diagonal, the second inequality holds if and only if all diagonal entries (the eigenvalues) are non-negative. ∎

The following lemma we state without proof. See [12] for more discussion.

Lemma 30 (Valid covariance function [12])**.** *An arbitrary symmetric kernel $k(\cdot, \cdot)$ is a valid covariance function if and only if the Gram matrix of every finite set of arbitrary input points $\{\boldsymbol{x}_i\}_{i=1}^n$ is positive semi-definite.*

3.1 Stationary Covariance Functions

Definition 31 (Stationary covariance function)**.** *A covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ is* stationary *if it is a function of $\boldsymbol{\tau} := \boldsymbol{x} - \boldsymbol{x}'$ and therefore* invariant to translations *[12]. We will write general stationary covariance functions in the form*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 r(\boldsymbol{x} - \boldsymbol{x}'), \tag{3.1}$$

where $r(\mathbf{0}) = 1$.

Definition 32 (Isotropy)**.** *A stationary covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ is also* isotropic *if it is a function of $\|\boldsymbol{x} - \boldsymbol{x}'\|$ and therefore also* invariant to rotations *[12]. We will write a general isotropic covariance function in the form*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \rho(d), \tag{3.2}$$

where $d = \|\boldsymbol{x} - \boldsymbol{x}'\|$ and $\rho(0) = 1$.

Remark 33. We call an arbitrary stationary covariance function which is not isotropic *anisotropic.*

Remark 34. We may refer to $r(\boldsymbol{\tau})$ and $\rho(\tau)$ as both covariance functions and correlation functions. When referring to them as covariance functions, we are implicitly assuming they are a function of $\boldsymbol{x}$ and $\boldsymbol{x}'$ whereas we will refer to them as a correlation function when we want to highlight that $r(\mathbf{0}) = 1$ or $\rho(0) = 1$.

3.1.1 Examples of Stationary Covariance Functions

In emulation, a common choice of covariance function for infinitely differentiable functions [41] is the squared exponential covariance function, also called the exponentiated quadratic.

Definition 35 (Squared exponential covariance function [12])**.** *The standard stationary squared exponential covariance function is defined by*

$$r(\boldsymbol{x} - \boldsymbol{x}') = \exp\left(-(\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{x}')\right) \tag{3.3}$$

and the isotropic form seen in Example 3 can be recovered by taking $\boldsymbol{\Sigma} = \operatorname{diag}\{\theta^2, \ldots, \theta^2\}$,

$$r(\boldsymbol{x} - \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\theta^2}\right). \tag{3.4}$$

Here in the anisotropic form, the covariance matrix $\boldsymbol{\Sigma}$ governs the general Mahalanobis distances. In the isotropic form, θ is called the *correlation length*[1]. However, in many cases, assuming the function being emulated is infinitely differentiable may be too strong of an assumption, and often undesirable [44]. The Matérn family of correlation functions are also widely used and can be used to approximately encode a prior belief that the function is k-times differentiable.

Definition 36 (Matérn covariance function [45])**.** *The stationary Matérn covariance function is defined as*

$$r(\boldsymbol{x} - \boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(2\sqrt{\nu}\frac{d(\boldsymbol{x}, \boldsymbol{x}')}{\theta^2}\right)^\nu K_\nu \left(2\sqrt{\nu}\frac{d(\boldsymbol{x}, \boldsymbol{x}')}{\theta^2}\right) \tag{3.5}$$

with $d(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{x}')}$. Γ is the gamma function [46] and K_ν is the modified Bessel function of the second kind [47]. We have positive parameters θ (correlation length) and ν. Similarly to above, we can recover the isotropic version by taking $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|$.

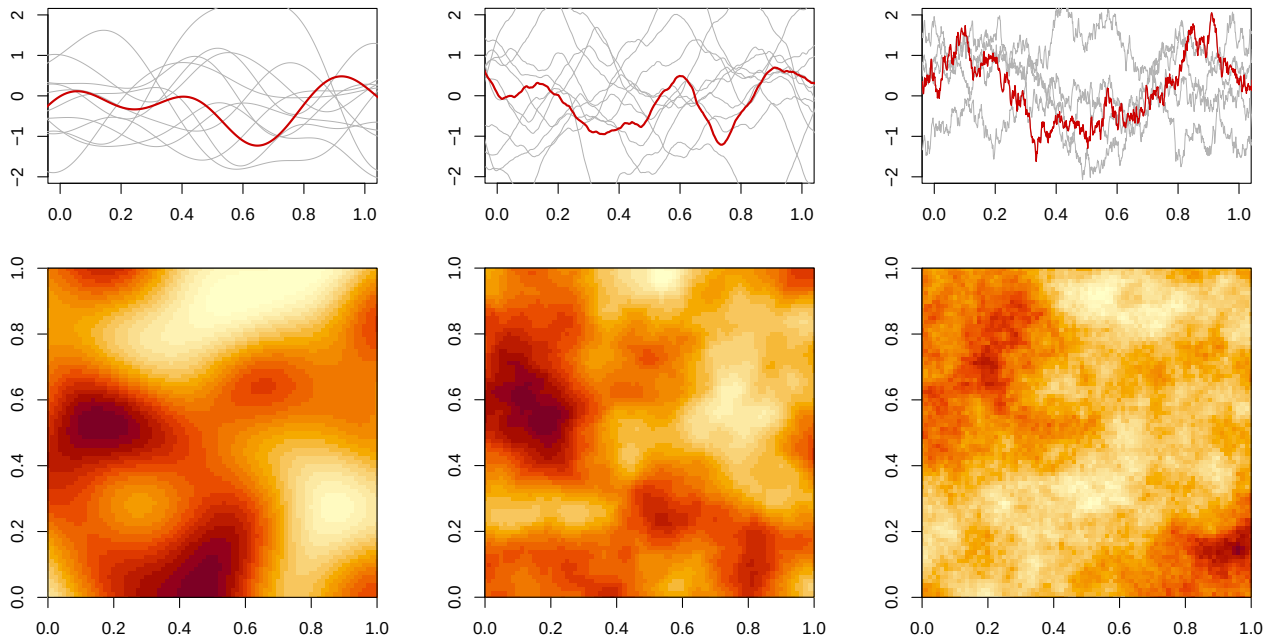
A process with a Matérn covariance structure is $\lceil \nu \rceil - 1$ times mean-square differentiable [12]. The Matérn covariance function simplifies for half-integer values of ν and a special case is the exponential covariance for $\nu = 1/2$.

[1] This may also be referred to as the length scale and is often denoted as ℓ or ρ instead.

Definition 37 (Exponential covariance function [41]). *The stationary exponential covariance function is defined as*

$$r(\boldsymbol{x} - \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{\theta}\right). \tag{3.6}$$

The exponential covariance function defines a non-differentiable process which is often too rough to use for computer models. This is shown in Figure 3.1c. In practice, the most common choices for ν when using a Matérn covariance function are $\nu = 3/2$ and $\nu = 5/2$ since it is usually difficult to distinguish between values of $\nu \geq 7/2$. In fact, the limit of the Matérn covariance function as $\nu \to \infty$ is the squared exponential covariance function [12]. Figure 3.1 shows the types of functions which each of these simple covariance structures can represent.



(a) Squared exponential. (b) Matérn, $\nu = \frac{3}{2}$. (c) Exponential.

Figure 3.1: 1D and 2D realisations of a GP prior with different stationary covariance structures.

Although so far we have only considered a restricted family of covariance functions, similar to the Matérn family, there are many more covariance functions widely used in different areas of machine learning. See [12] and [48] for a more detailed list of covariance functions.

3.2 Combining Covariance Functions

It is possible to combine covariance functions to adopt the features of both. The following Lemma is stated without proof.

Lemma 38 (Combining covariance functions [48]). *Given two covariance functions $k_A(\cdot,\cdot)$ and $k_B(\cdot,\cdot)$,*

1. *Multiplying k_A and k_B gives a valid covariance function,*

$$k_\times(\boldsymbol{x}, \boldsymbol{x}') = k_A(\boldsymbol{x}, \boldsymbol{x}') \times k_B(\boldsymbol{x}, \boldsymbol{x}'). \tag{3.7}$$

2. *Adding k_A and k_B gives a valid covariance function,*

$$k_+(\boldsymbol{x}, \boldsymbol{x}') = k_A(\boldsymbol{x}, \boldsymbol{x}') + k_B(\boldsymbol{x}, \boldsymbol{x}'). \tag{3.8}$$

Remark 39. Multiplying covariance functions can be thought of as similar to an AND operator since the resulting covariance function from multiplying two covariance functions has a high value when

both covariance functions are high. Conversely, adding covariance functions can be thought of as an OR operator in a similar way [48].

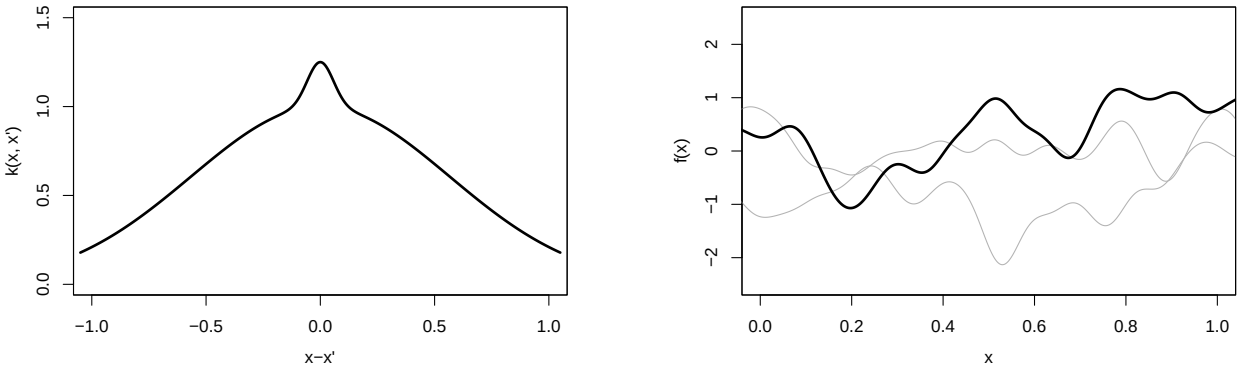
Remark 40. It is also possible to compose a covariance function with a standard function [49], but we will not investigate this here.

This will become particularly useful when we consider periodic covariance functions later, although this also justifies the scale mixture of covariance functions as in the following example.

Example 41. *Consider the scale mixture of squared exponential covariance functions:*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{4}\exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{0.08^2}\right) + \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{0.8^2}\right) \tag{3.9}$$

that is with $\theta_1 = 0.08$, $\theta_2 = 0.8$ *and* $\sigma_1^2 = 0.25$, $\sigma_2^2 = 1$. *This covariance function will have larger global and smaller local changes simultaneously. Figure 3.2 shows this covariance function along with some samples drawn from a GP prior with this covariance structure. It is possible to add an arbitrary number of scale mixtures together.*



(a) (b)

Figure 3.2: 1D realisations of a GP prior (right) with scale mixture covariance structure from Example 41 with covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ (left) from (3.9).

Definition 42 (White noise covariance function [49])**.** *The white noise covariance function is given by*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \mathbb{1}_{\boldsymbol{x}=\boldsymbol{x}'} \tag{3.10}$$

with variance parameter σ^2.

The white noise covariance function can be used to add independent, identically distributed noise. This is most commonly seen as a nugget term which can help account for numerical instability [41], although we will later see other uses for adding a nugget term.

Definition 43 (Nugget term [41])**.** *We can add a nugget term to an existing covariance function,*

$$\sigma^2(k(\boldsymbol{x}, \boldsymbol{x}') + \delta \mathbb{1}_{\boldsymbol{x}=\boldsymbol{x}'}) \tag{3.11}$$

where δ *is a small proportion of the total variance and we have assumed* $k(\boldsymbol{x}, \boldsymbol{x}) = 1$.

Remark 44. The parameters in the covariance functions such as θ and ν may be chosen to reflect prior information, or alternatively, they may be chosen suitably after observing the behaviour of the function. For example, a common approach is to choose parameters which maximise the likelihood [50], however, this does require enforcing distributional assumptions on our computer model and as discussed in the previous chapter, computer models are rarely realisations of a Gaussian process. A more robust method is to use cross-validation [12].

3.3 Periodic Covariance Functions

To motivate why we are interested in periodic covariance functions for emulating computer models, we will consider a simple example of a computer model for a four-link manipulator (robot arm).

Example 45 (Four-link manipulator)**.** *This example considers the toy computer model modelling the distance from the origin of a four-link manipulator with inputs as the lengths and angles of the four arms [51]. The explicit function is given by*

$$f(\boldsymbol{x}) = \sqrt{\left(\underbrace{\sum_{i=1}^{4} L_i \cos\left(\sum_{j=1}^{i} \theta_j\right)}_{u}\right)^2 + \left(\underbrace{\sum_{i=1}^{4} L_i \sin\left(\sum_{j=1}^{i} \theta_j\right)}_{v}\right)^2}. \tag{3.12}$$

This has the input parameters $\theta_i \in [0, 2\pi]$ (angles) and $L_i \in [0, 1]$ (arm segment length) for $i = 1, 2, 3, 4$ illustrated in Figure 3.3 below.

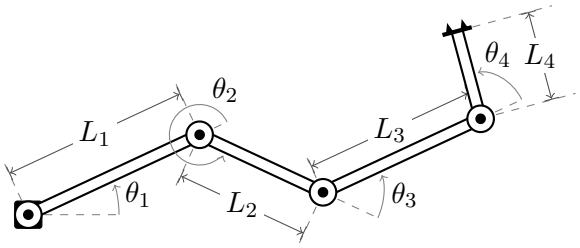
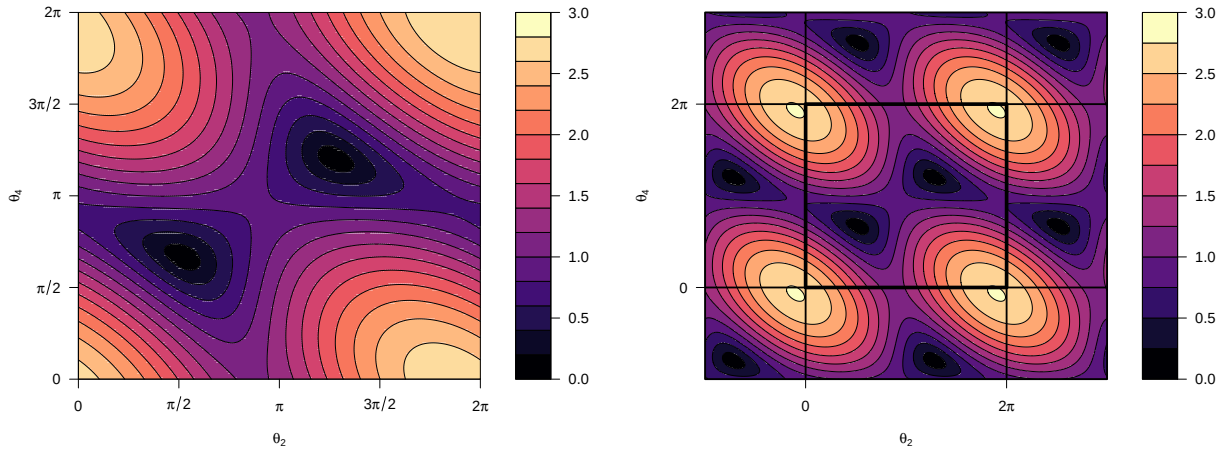


Figure 3.3: Diagram of the four-link manipulator in the uv-plane, adapted from [52].

We consider varying θ_2 and θ_4 whilst fixing all other parameters at $\theta_1 = 0$, $\theta_3 = \frac{\pi}{6}$, $L_1 = 0.8$, $L_2 = 0.4$, $L_3 = 0.6$, $L_4 = 1.0$ and we denote this function $g(\theta_2, \theta_4)$. This function is shown in Figure 3.4.



(a) True function.

(b) True function with an extended domain to show periodicity.

Figure 3.4: True function $g(\theta_2, \theta_4)$ from Example 45.

Before we emulate this function, we will consider periodic covariance functions. Periodic covariance functions exploit that by identifying boundaries, the distance between two points close to opposite sides of the boundary is actually small. This is illustrated in Figure 3.5.

The following definition is a simple example of a periodic covariance function which approximately behaves like the squared exponential covariance function for periodic domains.

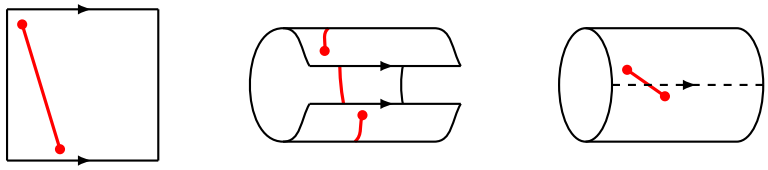


Figure 3.5: Relating edges in a non-deformative way. The red line represents the distance between two points which is shorter when the boundary identification is respected.

Definition 46 (1D periodic covariance function [48])**.** *The 1D periodic covariance function is defined as*

$$r(x - x') = \exp\left(-\frac{4}{\theta^2}\sin^2\left(\frac{\pi}{p}|x - x'|\right)\right) \tag{3.13}$$

where θ^2 is the correlation length and p is the period.

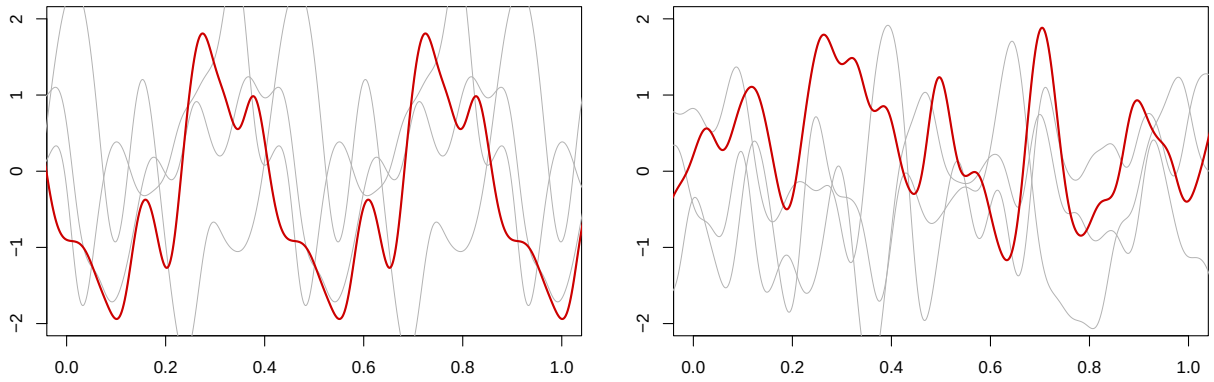
(3.13) is exactly periodic so this will respect periodic domains. However, there are some applications in which a domain may not be periodic, but it does possess a periodic nature. The following covariance function is obtained by multiplying the periodic covariance function above with the squared exponential covariance function resulting in a locally periodic nature.

Definition 47 (1D locally periodic covariance function [48])**.** *The 1D locally periodic covariance function is defined as*

$$r(x - x') = \exp\left(-\frac{4}{\theta^2}\sin^2\left(\frac{\pi}{p}\|x - x'\|\right)\right)\exp\left(-\frac{1}{\theta^2}\|x - x'\|^2\right) \tag{3.14}$$

with correlation length θ^2 and period p.

Figure 3.6 shows simple 1D functions which have the periodic and locally periodic covariance structure.



(a) (b)

Figure 3.6: 1D realisations of a GP prior with periodic (left) and locally periodic (right) covariance functions. In each case, we have used a correlation length of 0.35 and a period of 0.45.

Many stationary covariance functions defined on $\mathbb{R}$ can easily be used to construct stationary periodic covariance functions on a periodic domain $\mathbb{T} \subset \mathbb{R}$. However, we do require the covariance function to decay sufficiently quickly. The following theorem allows us to periodise some of the standard covariance functions we have considered (e.g. squared exponential, Matérn, exponential).

Theorem 48 (Periodisation in 1D [12, 53][2])**.** *Take the stationary covariance function $k(x, x') = \sigma^2 r(\tau)$, and assume*

$$|r(\tau)| \leq C\exp(-\alpha|\tau|), \tag{3.15}$$

[2]We have added the decay condition (3.15) not present in [12] as for certain covariance functions, the constant covariance function for example, (3.16) would diverge.

for constants C and $\alpha > 0$. Then the function

$$k_{\mathbb{T}}(x, x') = \sigma^2 \sum_{m \in \mathbb{Z}} r(\tau + mp) \tag{3.16}$$

is a valid covariance function with period p.

Proof. We first show that (3.16) converges,

$$\begin{aligned}
\left| \sum_{m \in \mathbb{Z}} r(\tau + mp) \right| &\leq \sum_{m \in \mathbb{Z}} |r(\tau + mp)| \\
&\leq \sum_{m \in \mathbb{Z}} C \exp\left(-\alpha |\tau + mp|\right) \\
&\leq \sum_{m \in \mathbb{Z}} C \exp\left(-\alpha (|mp| - |\tau|)\right)
\end{aligned}$$

by reverse triangle inequality since $|mp| \geq |\tau|$ for $m \neq 0$,

$$\begin{aligned}
&= \sum_{m \in \mathbb{Z}} C \exp(\alpha|\tau|) \exp(-\alpha|mp|) \\
&= 2C \exp \alpha|\tau| \left(1 + \sum_{m=1}^{\infty} \exp(-\alpha m p)\right) \\
&= 2C \exp \alpha|\tau| \left(1 + \frac{\exp(-\alpha p)}{1 - \exp(-\alpha p)}\right) < \infty.
\end{aligned}$$

Now given this converges, symmetry is trivial and for positive semi-definiteness, consider a finite collection $\{x_i\}_{i=1}^n \subset \mathbb{T}$ and constants $c_i \in \mathbb{R}$,

$$\begin{aligned}
\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k_{\mathbb{T}}(x_i, x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \sigma^2 \sum_{m \in \mathbb{Z}} r((x_i - x_j) + mp) \\
&= \sigma^2 \sum_{m \in \mathbb{Z}} \left(\underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j r((x_i - x_j) + mp)}_{\geq 0} \right) \geq 0.
\end{aligned}$$

since $r(\tau)$ is positive semi-definite. ■

Remark 49. Deriving an analytic form for the infinite sum requires additional work and may not be possible, however, we may approximate it with a truncated sum (since the covariance function decays to 0) but we would generally prefer an explicit form.

To create periodic covariance functions for an input space of dimension $N > 1$, we can multiply covariance functions for each dimension [48]. Many non-periodic multiple-dimension covariance functions can be seen to be constructed as the multiple of 1D covariance functions, the simplest example being the simple isotropic squared exponential covariance function,

$$\exp\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\theta^2}\right) = \exp\left(\frac{|x_1 - x_1'|}{\theta^2}\right) \cdots \exp\left(\frac{|x_N - x_N'|}{\theta^2}\right). \tag{3.17}$$

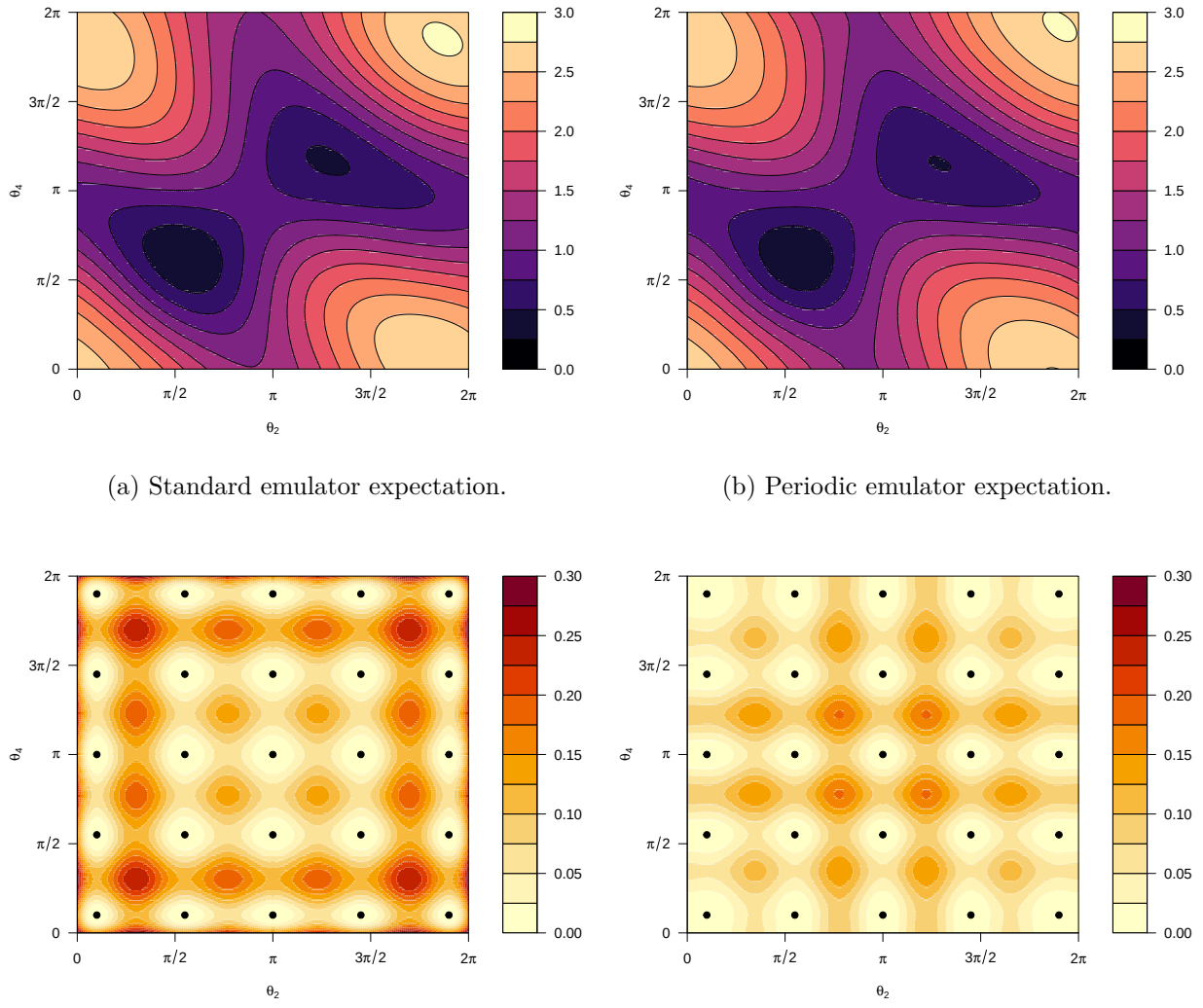
In this way, it is not difficult to extend Theorem 48 above to N dimensions with $M \leq N$ dimensions periodic and to use the squared exponential covariance function for the $N - M$ non-periodic dimensions.

Example 50 (2D periodic covariance function)**.** *Taking the same correlation length, θ^2 for each dimension and assuming the two dimensions have period p_1 and p_2 respectively, then the following is a periodic covariance function,*

$$r(\boldsymbol{x} - \boldsymbol{x}') = \exp\left\{-\frac{4}{\theta^2}\left[\sin^2\left(\frac{\pi}{p_1}\|x_1 - x_1'\|\right) + \sin^2\left(\frac{\pi}{p_2}\|x_2 - x_2'\|\right)\right]\right\}. \tag{3.18}$$

Returning to our motivating example, we can now use a 2D periodic covariance function to emulate the function.

Example 45 (continuing from p. 16)**.** *We will emulate $g(\theta_2, \theta_4)$ with a 5×5 grid design and we will compare a simple emulator with a standard squared exponential covariance structure to one with a periodic covariance structure as in Example 50.*



(a) Standard emulator expectation.

(b) Periodic emulator expectation.

(c) Standard emulator variance.

(d) Periodic emulator variance.

Figure 3.7

Figure 3.7 shows the expectation and the variance of the two emulators. We can see that whilst the emulators perform similarly in the centre, near the edges, the periodic emulator does a better job at not resorting to the prior expectation. This is explained by looking at the variances, the periodic emulator exploits the periodicity and so the edge points are as close to other points as any others. Conversely, for the standard emulator which is unaware of the periodic nature, the edges are far away from any other points and so the variance is much higher than elsewhere.

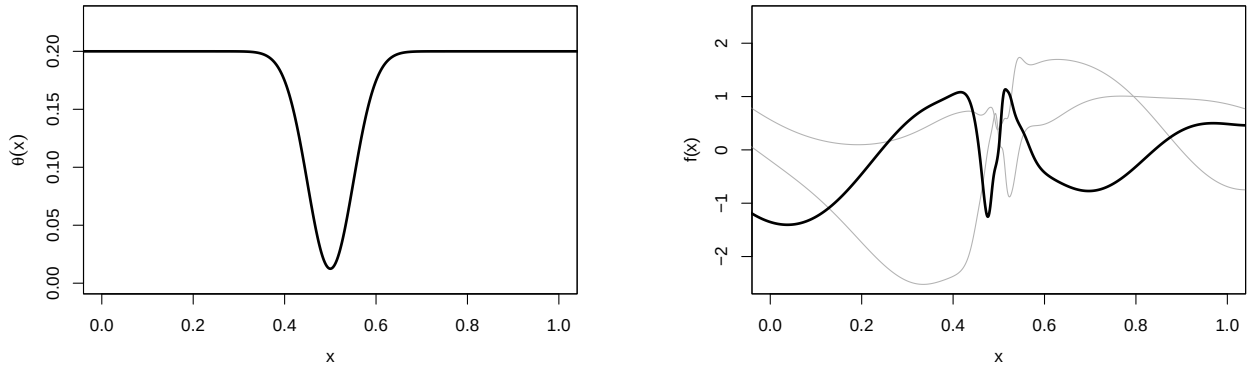
3.4 Non-Stationary Covariance Functions

Covariance structures can be delicate and modifying them without careful consideration can lead to non-valid covariance structures. For example, replacing θ in the 1D isotropic squared exponential covariance function with an arbitrary function $\theta(x, x')$ will not, in general, lead to a valid covariance structure. In this section, we will explore some examples of flexible non-stationary covariance functions.

Example 51 (Gibbs' covariance function [12])**.** *The simple anisotropic form of the squared exponential with different correlation lengths in different directions is given by Equation 3.3 with* $\boldsymbol{\Sigma} = \text{diag}(\theta_1^2, \ldots, \theta_N^2)$*. Gibbs' covariance function [12] takes a similar form which allows* θ_i *to be* $\boldsymbol{x}$*-dependant:*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \prod_{k=1}^{N} \left(\frac{2\theta_k(\boldsymbol{x})\theta_k(\boldsymbol{x}')}{\theta_k^2(\boldsymbol{x}) + \theta_k^2(\boldsymbol{x}')} \right)^{1/2} \exp\left(-\sum_{k=1}^{N} \frac{(x_k - x_k')^2}{\theta_k^2(\boldsymbol{x}) + \theta_k^2(\boldsymbol{x}')} \right). \tag{3.19}$$

Figure 3.8 shows a simple 1D example of this covariance function being used.



(a) (b)

Figure 3.8: 1D realisations of a GP prior (right) with Gibbs' simple non-stationary covariance structure from Example 51 with correlation length function $\theta(x)$ (left). Figure recreated from [12].

We can construct a more general non-stationary covariance function following [54] which introduces a flexible class of non-stationary covariance functions.

Theorem 52 (Generalised non-stationary covariance function [54, 45, 55])**.** *Let* $\rho(\tau) := \rho(\|\boldsymbol{x} - \boldsymbol{x}'\|)$ *be an isotropic covariance function and* $\boldsymbol{\Sigma}(\boldsymbol{x})$ *be an* $\boldsymbol{x}$*-dependant positive definite covariance matrix then,*

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \frac{2^{N/2} \det(\boldsymbol{\Sigma}(\boldsymbol{x}))^{1/4} \det(\boldsymbol{\Sigma}(\boldsymbol{x}'))^{1/4}}{\det(\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}'))^{1/2}} \rho\left(\sqrt{\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}')} \right) \tag{3.20}$$

with

$$\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \left(\frac{\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}')}{2} \right)^{-1} (\boldsymbol{x} - \boldsymbol{x}') \tag{3.21}$$

defines a valid non-stationary covariance function.

We can notice that (3.20) essentially replaces the squared distance $\tau^2 = \|\boldsymbol{x} - \boldsymbol{x}'\|^2 = (\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T}(\boldsymbol{x} - \boldsymbol{x}')$ with the quadratic form in (3.21). By doing this we introduce an $\boldsymbol{x}$-dependant covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{x})$ which governs the general Mahalanobis distance locally around $\boldsymbol{x}$. The quadratic form $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}')$ (3.21) simply takes the average of the two local covariance matrices. The prefactor in (3.20) ensures k is a valid covariance function. Before we can prove Theorem 52, we need to prove some auxiliary lemmas.

Definition 53. *The Gaussian kernel* $K_{\boldsymbol{x}}(\boldsymbol{u})$ *for an* $\boldsymbol{x}$*-dependent covariance matrix* $\boldsymbol{\Sigma}(\boldsymbol{x})$ *is,*

$$K_{\boldsymbol{x}}(\boldsymbol{u}) := \frac{1}{(2\pi)^{p/2} \det\left(\boldsymbol{\Sigma}(\boldsymbol{x})\right)^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{u})^\mathsf{T} \left(\boldsymbol{\Sigma}(\boldsymbol{x})\right)^{-1} (\boldsymbol{x} - \boldsymbol{u}) \right). \tag{3.22}$$

Remark 54. In the next few lemmas, we will denote covariance functions as $c(\boldsymbol{x}, \boldsymbol{x}')$ as not to confuse notation with $k(\boldsymbol{x}, \boldsymbol{x}')$ in Theorem 52.

Lemma 55. *Let* $K_{\boldsymbol{x}}(\boldsymbol{u})$ *be a Gaussian kernel for* $\mathcal{N}(\boldsymbol{x}, \boldsymbol{\Sigma}(\boldsymbol{x}))$ *where* $\boldsymbol{\Sigma}(\boldsymbol{x})$ *is an* $\boldsymbol{x}$*-dependent covariance matrix. Then the following covariance function*

$$c(\boldsymbol{x}, \boldsymbol{x}') = \int_{\mathbb{R}^N} K_{\boldsymbol{x}}(\boldsymbol{u}) K_{\boldsymbol{x}'}(\boldsymbol{u}) \, d\boldsymbol{u} \tag{3.23}$$

is positive semi-definite [54].

Proof. We follow the proof in [54], to show positive semi-definiteness, let $n \in \mathbb{N}$ and $\boldsymbol{x}_i \in \mathbb{R}^N$ then,

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j c(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \int_{\mathbb{R}^N} K_{\boldsymbol{x}_i}(\boldsymbol{u}) K_{\boldsymbol{x}_j}(\boldsymbol{u})\ d\boldsymbol{u} \\ &= \int_{\mathbb{R}^N} \sum_{i=1}^{n}\sum_{j=1}^{n} a_i K_{\boldsymbol{x}_i}(\boldsymbol{u}) a_j K_{\boldsymbol{x}_j}(\boldsymbol{u})\ d\boldsymbol{u} \\ &= \int_{\mathbb{R}^N} \sum_{i=1}^{n} a_i K_{\boldsymbol{x}_i}(\boldsymbol{u}) \sum_{j=1}^{n} a_j K_{\boldsymbol{x}_j}(\boldsymbol{u})\ d\boldsymbol{u} \\ &= \int_{\mathbb{R}^N} \left(\sum_{i=1}^{n} a_i K_{\boldsymbol{x}_i}(\boldsymbol{u})\right)^2 d\boldsymbol{u} \geq 0\end{aligned}$$

and so $c(\cdot,\cdot)$ is positive semi-definite and a valid covariance function. ■

Lemma 56. *The covariance function (3.23) given in Lemma 55 has closed form*

$$c(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{(2\pi)^{N/2}\det(\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}')^{1/2})} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^{\mathsf{T}}\left(\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}')\right)^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right). \tag{3.24}$$

Proof. Again following the proof in [54], recognise (3.23) as the convolution,

$$\int \phi_{\boldsymbol{A}}(\boldsymbol{u}-\boldsymbol{x})\phi_{\boldsymbol{B}}(\boldsymbol{u})\ d\boldsymbol{u} \tag{3.25}$$

where $\phi(\cdot)$ is the normal density function and $\boldsymbol{A} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}(\boldsymbol{x}))$, $\boldsymbol{B} \sim \mathcal{N}(\boldsymbol{x}', \boldsymbol{\Sigma}(\boldsymbol{x}'))$ and $\boldsymbol{A}$, $\boldsymbol{B}$ are independent. We now consider the transformation $\boldsymbol{W} = \boldsymbol{B} - \boldsymbol{A}$, $\boldsymbol{V} = \boldsymbol{B}$, which has Jacobian 1 and obtain the following equalities:

$$\begin{aligned}\int \phi_{\boldsymbol{A},\boldsymbol{B}}(\boldsymbol{u}-\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} &= \int \phi_{\boldsymbol{W},\boldsymbol{V}}(\boldsymbol{u} - (\boldsymbol{u}-\boldsymbol{x}), \boldsymbol{u}) d\boldsymbol{u} \\ &= \int \phi_{\boldsymbol{W},\boldsymbol{V}}(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} \\ &= \phi_{\boldsymbol{W}}(\boldsymbol{x})\end{aligned}$$

and since $\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{x}', \boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}'))$,

$$\phi_{\boldsymbol{W}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{N/2}\det(\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}'))^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^{\mathsf{T}}\left(\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}')\right)^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right).$$

■

We can recognise that by absorbing necessary constants into the covariance matrices in Lemma 56 (3.24), and dividing by the standard deviation function

$$\sigma(\boldsymbol{x}) := \sqrt{c(\boldsymbol{x}, \boldsymbol{x})} = \frac{1}{2^{N/2}\pi^{N/4}\det(\boldsymbol{\Sigma}(\boldsymbol{x}))^{1/4}}, \tag{3.26}$$

we can obtain the covariance function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \frac{2^{N/2}\det(\boldsymbol{\Sigma}(\boldsymbol{x}))^{1/4}\det(\boldsymbol{\Sigma}(\boldsymbol{x}'))^{1/4}}{\det(\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}'))^{1/2}} \exp\left(-(\boldsymbol{x}-\boldsymbol{x}')^{\mathsf{T}}\left(\frac{\boldsymbol{\Sigma}(\boldsymbol{x}) + \boldsymbol{\Sigma}(\boldsymbol{x}')}{2}\right)^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right), \tag{3.27}$$

which is a special case of Theorem 52, using the stationary isotropic squared exponential covariance function for $\rho(\tau)$. We are now able to prove Theorem 52 in generality.

Proof of Theorem 52. This proof follows the proof of Theorem 1 in [54] and is an application of Theorem 2 of [56], which states that the class of functions positive definite on Hilbert space[3] is identical with the class of functions of the form

$$\rho(\tau) = \int_0^\infty \exp(-\tau^2 s) d\mu(s) \tag{3.28}$$

where $\mu(\cdot)$ is non-decreasing and bounded and $s \geq 0$. Now the class of functions positive definite on Hilbert space is identical to the class of functions which are positive definite on $\mathbb{R}^p$ [56]; functions in this class can be seen to be scale mixtures of the squared exponential correlation where the measure $\mu(\cdot)$ determines the mixture of squared exponential correlations. The underlying stationary covariance function with argument $\sqrt{\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}')}$ can be expressed as,

$$\begin{aligned}\rho\left(\sqrt{\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}')}\right) &= \int_0^\infty \exp\left(-\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}')s\right) d\mu(s) \\ &= \int_0^\infty \exp\left(-(\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \left(\frac{\frac{\boldsymbol{\Sigma}(\boldsymbol{x})}{s} + \frac{\boldsymbol{\Sigma}(\boldsymbol{x}')}{s}}{2}\right)^{-1} (\boldsymbol{x} - \boldsymbol{x}')\right) d\mu(s)\end{aligned} \tag{3.29}$$

and by Lemma 56 absorbing s (non-negative) into the covariance matrices, we can write this exponential as a convolution of Gaussian kernels,

$$\rho\left(\sqrt{\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{x}')}\right) = \int_0^\infty \int_{\mathbb{R}^p} K_{\boldsymbol{x},s}(\boldsymbol{u}) K_{\boldsymbol{x}',s}(\boldsymbol{u}) \, d\boldsymbol{u} d\mu(s). \tag{3.30}$$

By Lemma 55, (3.30) can be seen to be positive semi-definite and so by Schoenberg's Theorem [56] and using a similar prefactor as in (3.27), we see that (3.20) defines a valid covariance structure. ■

Remark 57. Taking $\boldsymbol{\Sigma}(\boldsymbol{x}) = \boldsymbol{I}$ in Theorem 52 recovers the isotropic covariance function.

Remark 58. As a final remark for this chapter, we note that it is possible to construct covariance functions for a wide variety of non-euclidean domains [32, 57] and further advances in this area could enable expert knowledge of interactions between input variables to be incorporated.

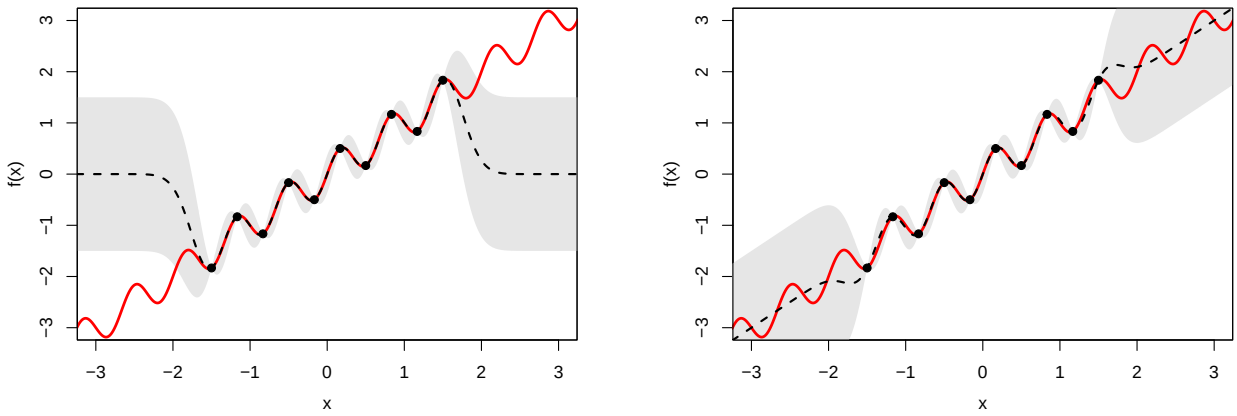
[3] The Hilbert space is the space of functions $\rho(\tau)$ belongs to.

Chapter 4

Advanced Emulation and Applications

4.1 Incorporating Explicit Basis Functions

In chapter 2, we examined a simple emulator which could effectively learn local structure in a computer model. However, if we examine this emulator away from design points we may observe that the expectation quickly resorts to the prior. Figure 4.1 shows this behaviour and we can see that this emulator fails to effectively learn global structure. This is particularly impactful on the emulator's predictive capabilities when the input space is high dimensional. This is because high dimensional input spaces are much larger and will likely have points far from any design points. To avoid this issue, we introduce a regression term to the emulator. The regression term can then learn the global structure of the function whilst a weakly stationary emulator can learn the local structure of the function.



(a) Constant prior expectation $\mathbb{E}[f(x)] = 0$.

(b) Prior expectation $\mathbb{E}(f(x)) = x$.

Figure 4.1: Simple weakly stationary emulator for $f(x) = x + \sin(3\pi x)/3$ (red) for different priors; (a) uses a simple constant prior and (b) uses a prior which represents the global structure. We can observe that the expectation (black dashed) quickly resorts to the prior whilst extrapolating.

Remark 59. Deep GPs have similar issues with extrapolation and will also quickly resort to the prior away from the data manifold.

Explicitly, our emulator will take the following form,

$$f(\boldsymbol{x}) = \sum_{j=1}^{p} \beta_j g_j(\boldsymbol{x}) + u(\boldsymbol{x}) \tag{4.1}$$

where $g_j(x)$ are appropriate basis functions with coefficients β_j (to be learned) and $u(\boldsymbol{x})$ is a weakly stationary process. We will denote $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\mathsf{T}$ and $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x}))^\mathsf{T}$.

Remark 60. The basis functions should be chosen appropriately to capture global structure which is likely to be present in the computer model. In practice, low-order polynomials are often used [41] but periodic functions may also be chosen if appropriate.

Staying in the Bayesian paradigm to preserve robust uncertainty, we will also specify priors over $\boldsymbol{\beta}$, that is, $\boldsymbol{\mu}_\beta := \mathbb{E}(\boldsymbol{\beta})$ and $\boldsymbol{\Sigma}_\beta := \mathrm{Var}(\boldsymbol{\beta})$. We typically assume $\boldsymbol{\beta}$ and $u(\boldsymbol{x})$ are uncorrelated a priori. This is reasonable given that the regression term is responsible for global structure whilst $u(\boldsymbol{x})$ is responsible for local structure.

We can now easily compute the prior expectation and covariance of (4.1) (see appendix A.7 and A.8) which may be used with Theorems 11 and 15. However, we will instead follow [41] and use the linearity of the adjusted expectation (Lemma 12) and consider the contribution to the adjusted expectation by each term in (4.1),

$$\mathbb{E}_D(f(\boldsymbol{x})) = \mathbb{E}_D(\boldsymbol{g}(\boldsymbol{x})^\mathsf{T}\boldsymbol{\beta} + u(\boldsymbol{x})) = \boldsymbol{g}(\boldsymbol{x})^\mathsf{T}\,\mathbb{E}_D(\boldsymbol{\beta}) + \mathbb{E}_D(u(\boldsymbol{x})). \tag{4.2}$$

But first, we introduce some notation; we write $\boldsymbol{g}(\cdot)$ for the row-wise action of $\boldsymbol{g}$ on a data-matrix and denote $\boldsymbol{g}(\cdot)$ acting on $\boldsymbol{X}_D$ and $\boldsymbol{X}_*$ as $\mathbf{G}_D$ and $\mathbf{G}_*$ respectively. That is,

$$\mathbf{G}_D \equiv \boldsymbol{g}(\boldsymbol{X}_D) := \begin{pmatrix} \boldsymbol{g}(\boldsymbol{x}^{(1)})^\mathsf{T} \\ \vdots \\ \boldsymbol{g}(\boldsymbol{x}^{(d)})^\mathsf{T} \end{pmatrix}, \qquad \mathbf{G}_* \equiv \boldsymbol{g}(\boldsymbol{X}_*) := \begin{pmatrix} \boldsymbol{g}(\boldsymbol{x}_*^{(1)})^\mathsf{T} \\ \vdots \\ \boldsymbol{g}(\boldsymbol{x}_*^{(n)})^\mathsf{T} \end{pmatrix}. \tag{4.3}$$

Similarly to our notation for $\mathbf{f}_D$ and $\mathbf{f}_*$, we will write $\mathbf{u}_D$ and $\mathbf{u}_*$ for $u(\cdot)$ acting on $\boldsymbol{X}_D$ and $\boldsymbol{X}_*$ respectively. For notational convenience, we will also write $\boldsymbol{\Omega}_D := \mathrm{Var}(\mathbf{u}_D)$ and $\boldsymbol{\Omega}_* := \mathrm{Var}(\mathbf{u}_*)$.

We first consider how we update the regression surface after observing $\mathbf{f}_D$.

Lemma 61. *After observing f at $\boldsymbol{X}_D$, applying the Bayes linear update to (4.1) yields an updated regression surface with*

$$\mathbb{E}_D(\boldsymbol{\beta}) = \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1}\right)^{-1}\left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{f}_D + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta\right), \tag{4.4}$$

$$\mathrm{Var}_D(\boldsymbol{\beta}) = \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1}\right)^{-1}. \tag{4.5}$$

Proof. See appendix subsection A.2.2. ■

Corollary 62 (Comparison to GLS estimate [41])**.** *We can rewrite (4.4) from Lemma 61 as*

$$\mathbb{E}_D(\boldsymbol{\beta}) = \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1}\right)^{-1}\left(\left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\right)\hat{\boldsymbol{\beta}}_{GLS} + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta\right) \tag{4.6}$$

where $\hat{\boldsymbol{\beta}}_{GLS} = \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\right)^{-1}\left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{f}_D\right)$ is the frequentist GLS estimator for $\boldsymbol{\beta}$. In this way, (4.6) is a weighted sum of the prior and the GLS estimate, which is weighted by the prior precision matrix [41]. Similarly, the variance in (4.5) is a combination of the GLS variance and prior precision,

$$\mathrm{Var}_D(\boldsymbol{\beta}) = \left(\mathrm{Var}(\hat{\boldsymbol{\beta}}_{GLS})^{-1} + \mathrm{Var}(\boldsymbol{\beta})^{-1}\right)^{-1}. \tag{4.7}$$

Remark 63 (Limiting behaviour [41])*.* In the vague prior limit for $\boldsymbol{\beta}$, that is $\boldsymbol{\Sigma}_\beta^{-1} \to \mathbf{0}$, we recover the frequentist GLS estimates,

$$\lim_{\boldsymbol{\Sigma}_\beta^{-1}\to\mathbf{0}} \mathbb{E}_D(\boldsymbol{\beta}) = \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \mathbf{0}\right)^{-1}\left(\left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\right)\hat{\boldsymbol{\beta}}_{GLS} + \mathbf{0}\boldsymbol{\mu}_\beta\right) = \hat{\boldsymbol{\beta}}_{GLS} \tag{4.8}$$

$$\lim_{\boldsymbol{\Sigma}_\beta^{-1}\to\mathbf{0}} \mathrm{Var}_D(\boldsymbol{\beta}) = \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \mathbf{0}\right)^{-1} = \mathrm{Var}(\hat{\boldsymbol{\beta}}_{GLS}). \tag{4.9}$$

In the additional limit of infinitely far apart runs, so they do not inform each other, we would get the OLS estimator [41].

Lemma 64. *Similarly to Lemma 61, we can update $u(\boldsymbol{x})$ for the points $\boldsymbol{X}_*$,*

$$\mathbb{E}_D(\mathbf{u}_*) = \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}(\mathbf{f}_D - \mathbf{G}_D\,\mathbb{E}_D(\boldsymbol{\beta})) \tag{4.10}$$

$$\operatorname{Var}_D(\mathbf{u}_*) = \boldsymbol{\Omega}_* - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\left(\boldsymbol{\Omega}_D^{-1} + \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\right)\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*). \tag{4.11}$$

Proof. See appendix subsection A.2.2. ■

Remark 65. The update in (4.10) takes the same form as the BL update for the simple emulator (2.26) where we are instead updating with the residual of the updated regression surface [41].

We can now combine Lemma 61 and 64 in the following theorem.

Theorem 66. *After observing f at $\boldsymbol{X}_D$, applying the Bayes linear update to $\mathbf{f}_*$ yields,*

$$\mathbb{E}_D(\mathbf{f}_*) = \mathbf{G}_*\,\mathbb{E}_D(\boldsymbol{\beta}) + \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}(\mathbf{f}_D - \mathbf{G}_D\,\mathbb{E}_D(\boldsymbol{\beta})) \tag{4.12}$$

$$\operatorname{Var}_D(\mathbf{f}_*) = \boldsymbol{\Omega}_* + \widetilde{\boldsymbol{G}}\operatorname{Var}_D(\boldsymbol{\beta})\widetilde{\boldsymbol{G}}^{\mathsf{T}} - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*) \tag{4.13}$$

where $\widetilde{\boldsymbol{G}} = \left(\mathbf{G}_* - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\right)$.

Proof. Following [41][1], we combine the results from Lemmas 61 and 64, and using linearity (12),

$$\begin{aligned}\mathbb{E}_D(\mathbf{f}_*) &= \mathbb{E}_D(\mathbf{G}_*\boldsymbol{\beta} + \mathbf{u}_*) = \mathbf{G}_*\,\mathbb{E}_D(\boldsymbol{\beta}) + \mathbb{E}_D(\mathbf{u}_*)\\ &= \mathbf{G}_*\,\mathbb{E}_D(\boldsymbol{\beta}) + \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}(\mathbf{f}_D - \mathbf{G}_D\,\mathbb{E}_D(\boldsymbol{\beta})).\end{aligned}$$

Computing the variance requires a little more work,

$$\begin{aligned}\operatorname{Var}_D(\mathbf{f}_*) &= \operatorname{Var}_D(\mathbf{G}_*\boldsymbol{\beta} + \mathbf{u}_*)\\ &= \mathbf{G}_*\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_*^{\mathsf{T}} + \operatorname{Var}_D(\mathbf{u}_*) + \operatorname{Cov}_D(\mathbf{G}_*\boldsymbol{\beta}, \mathbf{u}_*) + \operatorname{Cov}_D(\mathbf{u}_*, \mathbf{G}_*\boldsymbol{\beta}).\end{aligned} \tag{4.14}$$

Before we can evaluate this, we need to compute $\operatorname{Cov}_D(\mathbf{G}_*\boldsymbol{\beta}, \mathbf{u}_*)$, and using that $\boldsymbol{\beta}$ and $\mathbf{u}_*$ are uncorrelated a priori,

$$\begin{aligned}\operatorname{Cov}_D(\mathbf{G}_*\boldsymbol{\beta}, \mathbf{u}_*) &= \mathbf{G}_*(\cancel{\operatorname{Cov}(\boldsymbol{\beta}, \mathbf{u}_*)} - \operatorname{Cov}(\boldsymbol{\beta}, \mathbf{f}_D)\operatorname{Var}(\mathbf{f}_D)^{-1}\operatorname{Cov}(\mathbf{f}_D, \mathbf{u}_*)) && \text{by (2.23)}\\ &= -\mathbf{G}_*\operatorname{Cov}(\boldsymbol{\beta}, \mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)\operatorname{Var}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)^{-1}\operatorname{Cov}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D, \mathbf{u}_*)\\ &= -\mathbf{G}_*\operatorname{Var}(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}}(\mathbf{G}_D\operatorname{Var}(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}} + \operatorname{Var}(\mathbf{u}_D))^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\\ &= -\mathbf{G}_*\boldsymbol{\Sigma}_\beta\mathbf{G}_D^{\mathsf{T}}(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^{\mathsf{T}} + \boldsymbol{\Omega}_D)^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\\ &= -\mathbf{G}_*(\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*) && \text{by (A.2)}\\ &= -\mathbf{G}_*\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*). && (4.15)\end{aligned}$$

Now plugging (4.15) and (4.11) into (4.14), we get,

$$\begin{aligned}\operatorname{Var}_D(\mathbf{f}_*) = \mathbf{G}_*\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_*^{\mathsf{T}} + \boldsymbol{\Omega}_* - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\left(\boldsymbol{\Omega}_D^{-1} + \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\right)\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\\ - \left[\left(\mathbf{G}_*\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\right) + \left(\mathbf{G}_*\operatorname{Var}_D(\boldsymbol{\beta})\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\right)^{\mathsf{T}}\right]\\ = \boldsymbol{\Omega}_* + \left(\mathbf{G}_* - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\right)\operatorname{Var}_D(\boldsymbol{\beta})\left(\mathbf{G}_*^{\mathsf{T}} - \mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\right)\\ - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\end{aligned}$$

since $\left(\mathbf{G}_*\operatorname{Var}_D(\beta)\mathbf{G}_D^{\mathsf{T}}\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)\right)^{\mathsf{T}} = \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\operatorname{Var}_D(\beta)\mathbf{G}_*^{\mathsf{T}}$,

$$= \boldsymbol{\Omega}_* + \widetilde{\boldsymbol{G}}\operatorname{Var}_D(\boldsymbol{\beta})\widetilde{\boldsymbol{G}}^{\mathsf{T}} - \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\operatorname{Cov}(\mathbf{u}_D, \mathbf{u}_*)$$

as we have in (4.13). ■

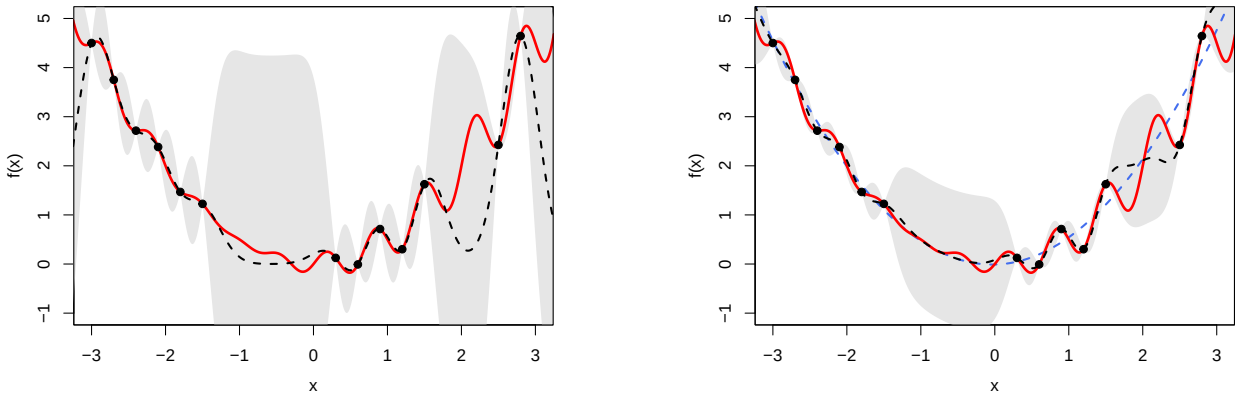
[1] [41] proves the equations for a single new point $\boldsymbol{x} \in \mathcal{X}$. We consider several points $\boldsymbol{X}_*$ which is trivial for expectations but taking variances, we also need to consider covariances between points which is non-trivial.

We now have a powerful emulator capable of capturing global and local structures in a computer model. We will now apply this to a simple example.

Example 67. *We consider the simple 1D function $f(x)$ (given explicitly in the appendix (B.2)) with a design constructed to highlight the issues with the simple emulator. We suppose we have weak prior beliefs and we build a simple emulator with prior expectation $E(f(x)) = 0$ and a squared exponential covariance structure with $\sigma^2 = 2$. The simple emulator is shown in Figure 4.2a and we can see that in regions with limited runs, the expectation quickly resorts to the prior and the emulator has high uncertainty. Conversely, we also build an advanced emulator with low-order polynomial basis functions $\{1, x, x^2\}$. We choose a covariance matrix with smaller $\sigma^2 = 0.16$ to reflect that we are more certain about the weakly stationary component. Additionally, we take the regression coefficients to have prior mean $\boldsymbol{\mu}_\beta = \mathbf{0}$ with vague prior variance matrix,*

$$\boldsymbol{\Sigma}_\beta = \begin{pmatrix} 8 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 8 \end{pmatrix},$$

The advanced emulator is shown in Figure 4.2b and the emulator expectation looks like a much better fit and has more reasonable uncertainty. In this example, there is a very prominent global trend, the more pronounced the global trend is, the more applicable the advanced emulator will be.



(a) Simple emulator.

(b) Advanced emulator.

Figure 4.2: Simple emulator (left) and advanced emulator (right) from Example 67. In each panel, the red line is the true function $f(x)$, the dashed black line is the emulator expectation and the light grey region is a 3σ interval for uncertainty. The advanced emulator additionally has a dashed blue line for the adjusted regression line.

4.1.1 Active and Inactive Inputs

Many real-world computer models have many input parameters and the high dimensional input spaces pose significant challenges for emulation. However, often only a few input parameters have a significant impact on the computer model output. This is especially true of computer models which have vector outputs, each output may only be impacted by a small number of the input parameters [8]. It is often helpful to only consider the function over the lower dimensional space of *active inputs* and add a nugget term $w(\boldsymbol{x})$ to absorb uncertainty due to inactive inputs [41]. Adding this to our advanced emulator which also incorporates regression terms, we have

$$f(\boldsymbol{x}) = \sum_{j=1}^{p} \beta_j g_j(\boldsymbol{x}_A) + u(\boldsymbol{x}_A) + w(\boldsymbol{x}) \tag{4.16}$$

where $\boldsymbol{x}_A$ are the active inputs. This form is widely used for emulation [1, 8].

4.2 Experimental Design

So far we have mostly ignored which points we have run the computer model at, which we call the design $\boldsymbol{X}_D$. In contrast to observational data, we usually have control over which points make up our design when emulating computer models; choosing sensible designs can drastically improve the performance of the emulators [24]. This area is broadly called *Experimental Design* and, in this section, we will briefly explore how we can choose sensible designs in the context of emulating computer models. There are two core questions we'd like to address:

1. How many design points do we need for our emulator to perform well?
2. Given a fixed budget of n runs, what is the best design we can use so that our emulator performs as well as it can?

The more expensive a computer model is to run, the more important it is to choose a good design. The naive choice for a design is to use a grid design like we have in Example 25. However, grid designs have serious problems as they scale very poorly to higher dimensions and can be very wasteful since run points can be very similar [41] (see Example 100 in the appendix). There is extensive literature on how good designs can be chosen [24] but we will focus mainly on designs which fulfil the following properties [41]:

- A design should be *space filling*, that is there should not be any large holes in the design.
- Runs should be as varied as possible so we gain as much information about the computer model as we can.

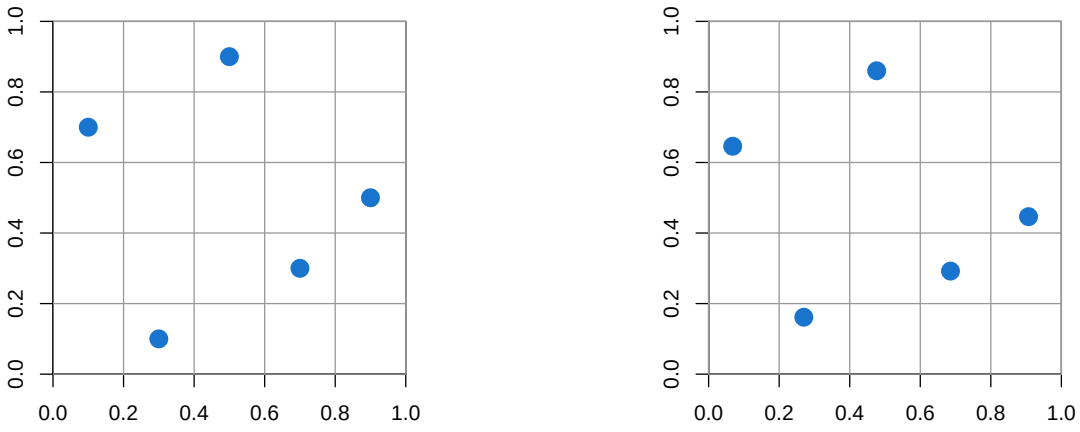
4.2.1 Latin Hypercube Designs

Latin Hypercube Sampling (LHS) developed in [58] and [59] allows efficient sampling of variables from multidimensional distributions. The *Latin Hypercube Design* (LHD) is a generic design widely used in the design of computer experiments [60].

Definition 68 (Latin Hypercube [41])**.** *A Latin Hypercube for an N dimensional space with n runs divides the range of each input variable x_i into n equally sized intervals. The Latin Hypercube is then formed by placing the n runs such that there is exactly one run in each interval for each input variable.*

Remark 69. We can additionally perturb points in the design for added variation [41].

Figure 4.3 shows an example of a random LHD along with that design randomly perturbed.



(a) Random LHD.

(b) Perturbed random LHD.

Figure 4.3: A random LHD of $n = 5$ points for $N = 2$ input dimensions.

This construction covers a wide range of values for each input variable and scales well to high dimensions. However, a random LHD is not guaranteed to be space-filling. There are two main methods we will look at to make space-filling LHDs, both rely on optimising an objective.

Remark 70. In subsequent chapters, we will still use grid designs, this is done for demonstration purposes and to highlight certain properties of the emulator and would not be used in practice.

Maximin LHD

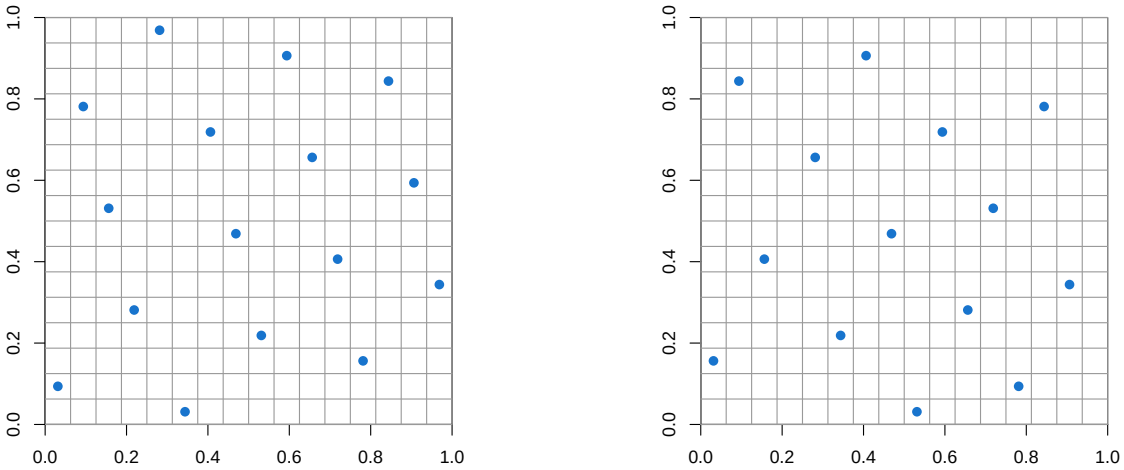
The maximin design problem is concerned with maximising the minimum distance between runs, which we can think of as spreading runs out as much as possible [60]. We will look specifically at the constrained maximin problem where we restrict to Latin Hypercube designs. Finding maximin LHDs can be time-consuming, especially in high dimensions [60], so we will consider heuristic maximin designs which are approximations, not guaranteed to be optimal. A simple algorithm to find a heuristic maximin LHD is outlined in [61] and involves an iterated local search (see Algorithm 1 in the appendix section A.3 for details). Figure 4.4a shows the resulting design after applying this algorithm for 16 points in 2D.

Audze-Eglaise LHD

Similarly, Audze-Eglaise designs are obtained by minimising the following objective [62]:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}. \tag{4.17}$$

This criterion is based on the analogy of charged particles and minimizing the forces between them [60]. [63] provide a genetic algorithm to find Audze-Eglaise Latin Hypercube designs and Figure 4.4b shows a simple Audze-Eglaise LHD in 2D (precomputed from [60]).



(a) Maximin LHD.

(b) Audze-Eglaise LHD.

Figure 4.4: Maximin (left) and Audze-Eglaise (right) Latin Hypercube designs of $n = 16$ points for $N = 2$ input dimensions.

Comparing the maximin and Audze-Eglaise designs in Figure 4.4, both seem to be reasonably space-filling. [64] compare Audze-Eglaise and Maximin LHDs and generally recommend the Audze-Eglais criterion over the maximin criterion. However, since experiment design is not the principal focus in this project, we will usually use a maximin design for convenience.

Remark 71. Due to computational constraints, it can often be easier to use precomputed designs. [60] provide near-optimal designs for a variety of input dimensions and design sizes, maximising different objectives.

4.2.2 Designs for Irregular Input Spaces

As we will see in the next section, we often want to construct a design for an unusually shaped region, $\mathcal{X}$, which we will assume lies in $[0,1]^N$ for simplicity. A simple approach to this is to generate a design for $[0,1]^N$ and reject all points not inside $\mathcal{X}$. This can be effective but is not ideal when the volume of $\mathcal{X}$ is very small and most points are being rejected. There are many approaches to constructively build a design for $\mathcal{X}$, and we will consider one of these methods.

Sequential Minimax

In our setup, we will assume that we have a finite set of points X (lying in $\mathcal{X}$) from which we can choose design points, and we want to find a design $X_D \subset X$ of d points. The goal with the sequential minimax algorithm is to iteratively choose points to add to our design to minimise the maximum distance from any point in X to a point in X_D. For a current design X'_D, the next addition to the design will be

$$\arg\min_{\boldsymbol{x}' \in X} \left[\max_{\boldsymbol{x} \in X} \left(\min_{\boldsymbol{x}_D \in X'_D \cup \{\boldsymbol{x}'\}} \|\boldsymbol{x} - \boldsymbol{x}_D\| \right) \right]. \tag{4.18}$$

Intuitively, this algorithm aims to build a design without holes. It is not feasible to consider the minimax distance of all possibilities of d points we could choose from X, and as so this is naturally a greedy algorithm. However, we can balance how myopic the algorithm is by considering p points to add at a time. Figure 4.5 shows a sequential minimax design for a simple example.

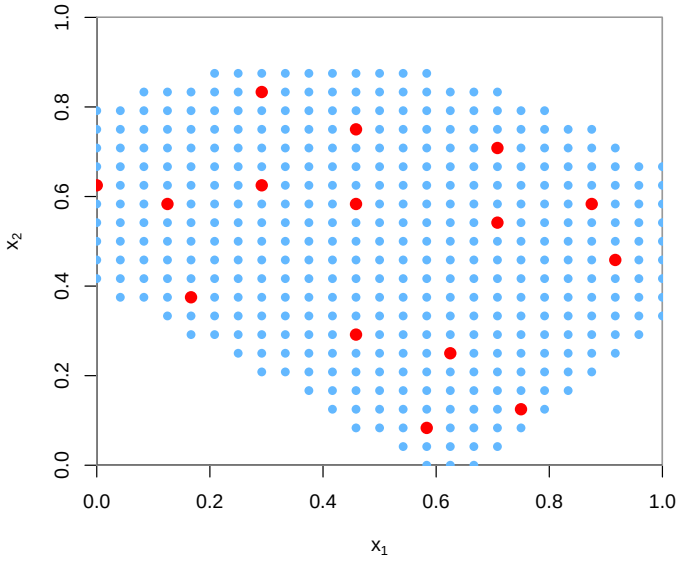


Figure 4.5: Sequential minimax design of $k = 15$ points added $p = 3$ at a time. The blue points mark the set of points X and the red points are the chosen points for the design X_D. See [16] for implementation details.

Remark 72. A concern with the minimax algorithm is that it can force points close together. Using a stochastic algorithm can help alleviate this issue as it will naturally perturb points slightly away from the optimum which may bunch points together (since the algorithm is myopic).

Remark 73. It is worth noting that the emulator variance for a particular design can be calculated without knowing the values the function takes at the design points. Therefore, the variance of an emulator resulting from a design can be evaluated before choosing to run the computer model for those design points. This can also be used as a criterion to iteratively choose points.

4.3 History Matching

A major motivation for emulating computer models is to learn about physical systems. One form this can take is model calibration, that is, finding all acceptable inputs to the computer model which give an output matching real-world observations. *History matching* is a widely used iterative global parameter search method [3] which exploits the rigorous uncertainty quantification of emulators to efficiently find *all* acceptable regions of the input space[2]. This section will briefly introduce history matching to motivate the benefits of building effective emulators. For a more thorough treatment of history matching, see [14]. Emulation is also widely used for optimisation, for example in engineering [10] and machine learning [65]. However, we will not explore Bayesian optimisation in this project.

History matching is used in many scientific disciplines including epidemiology [66, 67, 68, 69], systems biology [9, 8], cosmology [2, 3, 1], climate modelling [70, 71, 6, 5] and engineering [72]. The importance of history matching in these applications should not be understated. For example, in epidemiology, where disease models inform public health decisions, the efficacy of models directly relies on how well they can reproduce empirical data [69].

History matching assumes the existence of a physical process y which is measured through observations z [73], and modelled with a computer model $f(\boldsymbol{x})$ ($\boldsymbol{x} \in \mathcal{X}$). The task of history matching is to find the regions of the input space $\mathcal{X}^* \subset \mathcal{X}$ which give outputs which are "acceptable" matches to the observations. A responsible analysis should incorporate all the complexities and uncertainties present in the comparison of the model and the real world [8]. As such, a critical step in history matching is accounting for all sources of uncertainty.

Remark 74. In much of the literature, the region of the input space which provides an acceptable match to observations, which we denote $\mathcal{X}^*$, is instead simply denoted as $\mathcal{X}$. We choose not to adopt this notation to avoid a clash with notation for the input space.

Sources of Uncertainty

The major sources of uncertainty which we need to consider are [41]:

1. *Model Discrepancy*: fundamentally, the computer model is an imperfect approximation of the real physical system. As such, we express this discrepancy between the computer model and the real system via the statistical model:

$$y = f(\boldsymbol{x}^*) + \epsilon \tag{4.19}$$

   where $\boldsymbol{x}^*$ is the best input we could choose and ϵ is a (typically independent) random quantity representing the model discrepancy.

2. *Observational Error*: the observations we make of the physical system will inevitably have some small error which introduces uncertainty,

$$z = y + e \tag{4.20}$$

   where e is the observational error. This is often well understood.

Remark 75. Ignoring the model discrepancy is implicitly assuming that the computer model is perfect.

4.3.1 Implausibility Measure

At a high level, history matching iteratively rules out large regions of the input space as implausible and subsequently evaluates the computer model at new points in the remaining region. This refines the emulator's uncertainty in the remaining region and allows for more inputs to be ruled out. We repeat this process for several *waves* until we are reasonably sure all input points in the remaining region will give an output similar to observed data (allowing for uncertainty). The way we identify regions which are implausible is via an implausibility measure.

[2] Assuming prior assumptions are correct, history matching finds all acceptable inputs with high probability.

Definition 76 (Univariate implausibility [73])**.** *The implausibility measure, $I(\boldsymbol{x})$, for a function f with univariate output matching an observation z, with model discrepancy ϵ, and measurement error e is given as,*

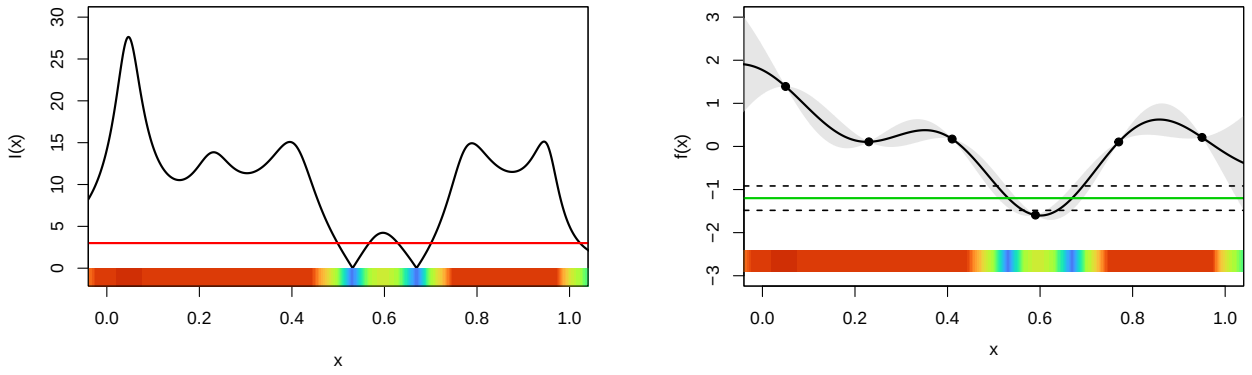
$$I^2(\boldsymbol{x}) = \frac{(\mathbb{E}_D[f(\boldsymbol{x})] - z)^2}{\text{Var}\left(\mathbb{E}_D[f(\boldsymbol{x})] - z\right)}, \tag{4.21}$$

$$I(\boldsymbol{x}) = \frac{|\mathbb{E}_D[f(\boldsymbol{x})] - z|}{\left(\text{Var}_D[f(\boldsymbol{x})] + \text{Var}[\epsilon] + \text{Var}[e]\right)^{1/2}}. \tag{4.22}$$

This is the error standardised by the standard deviation (uncertainty).

A low implausibility measure at an input means that given our current uncertainty (which could be high) we cannot rule this input out as a match. However, a high implausibility measure says that with high probability, the input does not lead to a match with the observation. As such, at each iteration (wave) of applying the algorithm, we can rule out any region with a high implausibility. A common choice for the *implausibility cutoff* is 3 [8], which guarantees there is *at most* a 5% chance of this being a plausible input by Pukelsheim's 3σ rule [42]. We refer to the regions which have an implausibility measure below the cutoff as *non-implausible*. This highlights that, although we cannot yet rule this region off as implausible, further runs may change this.

Example 77. *Consider a simple 1D function $f(x)$ (given explicitly in the appendix (B.4)) and assume we want to history match over the unit interval to the observed value $z = -1.2$. We also assume $\text{Var}(\epsilon) = 0.05^2$, $\text{Var}(e) = 0.08^2$ and a fixed design. Figure 4.6 shows the implausibility.*



(a) (b)

Figure 4.6: From Example 77, (a) shows the implausibility measure (black) with a red line marking the implausibility cutoff at 3. (b) shows the emulator expectation (black) with 3σ error bars (grey) and the observed value of z (green) with observational error bars (dashed). Both plots include a colour scale for the implausibility where blue is a low implausibility and red is a high implausibility.

Remark 78. There also exists specialised implausibility measures for functions with multivariate outputs [41].

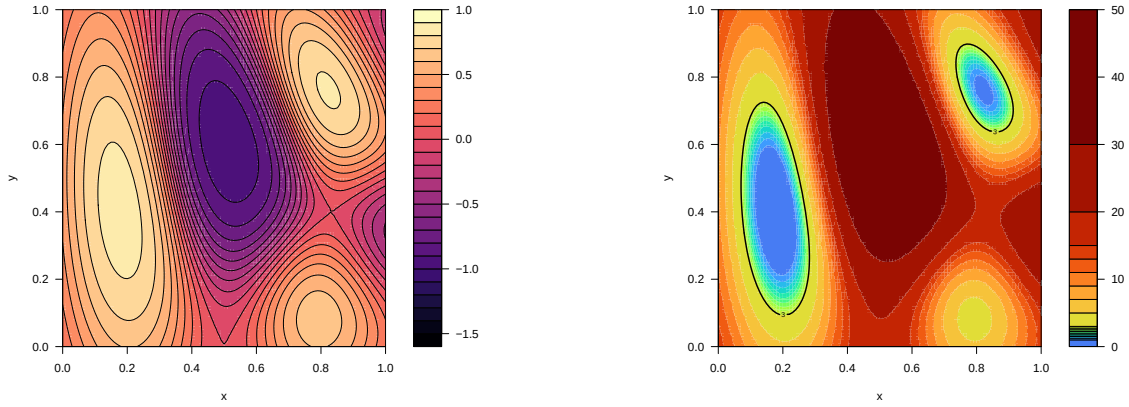
4.3.2 2D Iterative History Matching

To further motivate why effective emulators are useful, we will consider a simple 2D history-matching example. To avoid getting lost in the details, we omit the full history matching algorithm [41, 14] as the details are not particularly illuminating.

Example 79. *We consider the 2D function $f(x, y)$ (given explicitly in the appendix (B.5)) shown in Figure 4.7a. We consider the problem of history matching to the observed value $z = 0.85$ and assume a model discrepancy and observational error structure given by $\text{Var}(\epsilon) = 0.04^2$ and $\text{Var}(e) = 0.02^2$.*

The true implausibility, assuming a perfect emulator with no uncertainty, is shown in Figure 4.7b and shows two distinct regions.

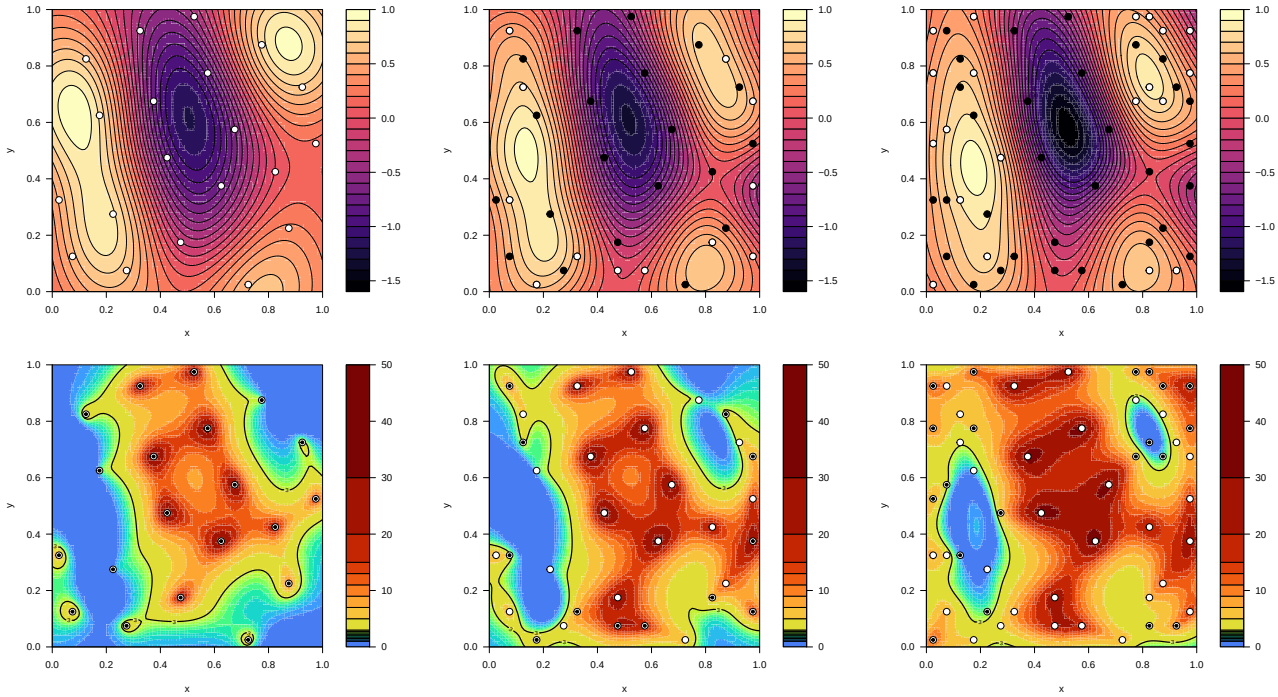
We perform three waves, in the first we use a maximin LHD of 20 points and rule out approximately half of the staring region as implausible. In waves two and three we use a minimax design chosen from the points lying on a 20×20 *grid of sizes 12 and 20 respectively. The emulator's expectation and implausibility is shown in Figure 4.8, and after three waves, the non-implausible region is similar to the optimal non-implausible region (shown in Figure 4.7b).*



(a) True function, $f(x, y)$.

(b) True implausibility.

Figure 4.7: True function and implausibility from Example 79.



(a) Wave 1.

(b) Wave 2.

(c) Wave 3.

Figure 4.8: Iterative history match for Example 79 over 3 waves showing the emulator expectation (top) and implausibility measure (bottom) over the initial region $\mathcal{X}$. The colouring of points distinguishes new design points from ones in existing waves; new points are either white (top) or white with a black central point (bottom).

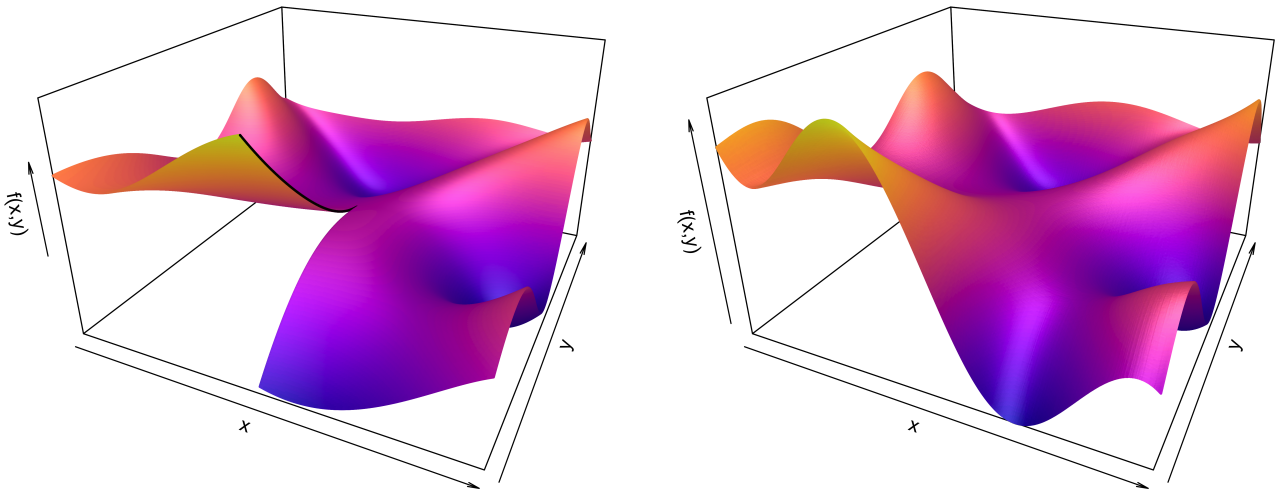
Remark 80. In Example 79, we continued to emulate over $\mathcal{X} = [0, 1]^2$ each wave. However, in a real application, once we have deemed a region implausible, we would not continue to emulate this region.

Chapter 5

Emulation with Partial Known Discontinuities

Several computer models are judged to be smooth over most of the input space whilst containing (potentially multiple) discontinuities at known locations. These discontinuities can arise for several reasons such as tipping points in the model [74] or physical barriers in the system being modelled. An example of a tipping point could be a cloud moving from open to closed cell behaviour [75]; an example of physical barriers could be a fault in an oil reservoir [15, 76, 77] or the boundary between ocean and sea ice. These discontinuities may be curved and may be *partial*, that is, they may not bisect the input space. In this chapter, we will explore how to emulate functions which have *multiple partial known discontinuities*.

Example 81. *As a simple example, consider the 2D function $f(x, y)$ shown in Figure 5.1a (given explicitly in the appendix B.6). This function has a simple partial discontinuity along $x = 0.5$, $y < 0.5$ and is otherwise smooth. Figure 5.1b shows the expectation for a simple emulator for $f(x, y)$ which has smoothed over the discontinuity.*



(a) True function $f(x, y)$ with discontinuity.

(b) Expectation of simple emulator.

Figure 5.1: 3D plot of $f(x, y)$ from Example 81 and the expectation of a simple emulator for $f(x, y)$. *A contour plot of $f(x, y)$ is provided in the appendix, in Figure B.2a.*

Remark 82. Unless explicitly stated otherwise, the term "smooth" is used in a colloquial sense to refer to functions which may be continuous, differentiable, or C^k-smooth.

Clearly, a standard emulator is not appropriate to emulate functions with discontinuities. Away from the discontinuity, we can expect a standard emulator to still perform well, but near the discontinuity,

the emulator's performance is likely going to be very poor. This should not be surprising, the emulator is unaware of the discontinuity and the covariance structure has been set up to capture a smooth function.

A simple and flexible approach to dealing with discontinuities is to partition the input space into disjoint subregions and emulate the function separately on each subregion. For example, Treed GPs [78] use simple rectangular subregions and [75] use Voronoi tessellations[1]. In practice, subregions can be defined to accommodate any arbitrary discontinuity which fully bisects the space. However, for partial discontinuities which do not naturally partition the space, this approach is not ideal. We ideally want to exploit the smoothness around the endpoints of a partial discontinuity, but this approach is unable to do so. Not exploiting this smoothness leads to less effective emulators which will require more known points to perform well. This issue becomes particularly important in higher dimensions [15].

A *Deep Gaussian Process* (DGP) [55] is a hierarchical model which consists of a composition of GPs where intermediary levels, called latent layers, enable the model to capture non-stationary behaviour [79]. DGPs are extremely flexible and are capable of modelling a rapid change around a discontinuity. However, this flexibility requires significantly more known points than is typically available for emulation [15], and has non-analytic uncertainty quantification [79]. Additionally, the DGP approach does not exploit our knowledge of the location of the discontinuities, but in principle it is possible to define a specialised non-stationary covariance structure which allows rapid change at the discontinuity [45]. However, in practice, this may be challenging, especially for complicated discontinuities. Another concern with these approaches is that a DGP made of smooth layers and a smooth non-stationary emulator both attempt to model the discontinuous function with a smooth function similar to Figure 5.2a. In many situations, the points very close to the discontinuity may be of particular interest, in which case these approaches will likely perform poorly [15].

It is clear that to fully represent the discontinuity, we need to explicitly incorporate the discontinuity in some way. It may be possible to manually adjust the covariance structure to decorrelate points either side of a discontinuity, but this approach is extremely fragile [15]; if not done with care this can result in a non-valid covariance structure. Namely, if we are using an isotropic covariance structure, it may be tempting to redefine our metric $\|\cdot\|$ in terms of the geodesic distance. However, this approach does not, in general, lead to a valid covariance structure, as shown in the example below. In the next section we will introduce a more principled approach to explicitly incorporate the discontinuity using torn embeddings.

Remark 83. In this context, the geodesic distance is simply defined as the distance of the shortest path between two points which does not intersect the discontinuity.

Example 84 (Non-validity of geodesic distance approach [15])**.** *Suppose we are using the isotropic squared exponential covariance structure (3.4) with $\sigma = 1$, $\theta = 1$ and geodesic distance $\|\cdot\|_{GD}$,*

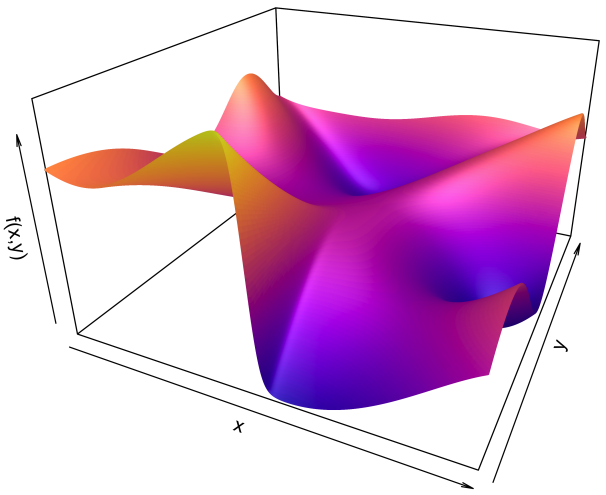
$$\operatorname{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}')) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|_{GD}^2\right).$$

Then consider the input space from Example 81 with points A, B, C, D, shown in Figure 5.2b and explicitly given as $A = (0.2, 0.2)^\mathsf{T}$, $B = (0.5, 0.5)^\mathsf{T}$, $C = (0.7, 0.1)^\mathsf{T}$ and $D = (0.5, 0.8)^\mathsf{T}$. The covariance matrix for $\mathbf{f} = (f(A), f(B), f(C), f(D))^\mathsf{T}$ is approximately

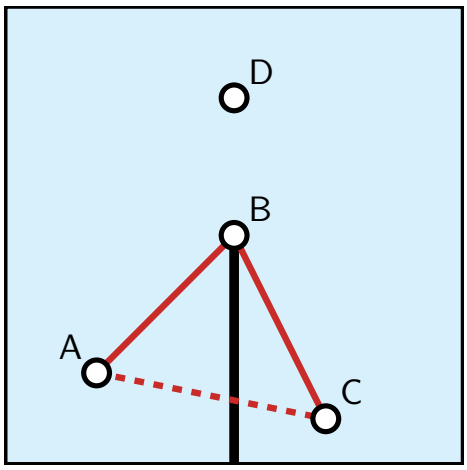
$$\operatorname{Cov}(\mathbf{f}, \mathbf{f}) \approx \begin{pmatrix} 1 & 0.835270 & 0.467913 & 0.637628 \\ 0.835270 & 1 & 0.818731 & 0.913931 \\ 0.467913 & 0.818731 & 1 & 0.588605 \\ 0.637628 & 0.913931 & 0.588605 & 1 \end{pmatrix}$$

and has a negative eigenvalue $\lambda \approx -0.0228794$. Thus the covariance matrix is not positive semi-definite, and so the covariance structure is not valid.

[1] [75] enables unknown discontinuities to be incorporated.



(a) A smooth function with a non-stationary rapid change where there was a discontinuity in Figure 5.1a.



(b) Points from Example 84 with the geodesic distance (full red) going around the discontinuity (black) and the direct distance (dashed red).

Figure 5.2: Attempts to deal with the discontinuity with non-stationary covariance functions.

5.1 Torn Embedding Emulation

A more principled approach to emulation with a partial known discontinuity is to embed the input space in a higher dimensional space via a torn embedding [15]. This will naturally decorrelate points either side of the discontinuity whilst preserving the smoothness away from the discontinuity. Crucially, this approach guarantees a valid covariance structure and induces a true discontinuity in the emulator. Additionally, although we are increasing the dimension of the space, this does not require additional runs of the computer model since all points of interest lie on a hypersurface.

This approach can be broken down simply as follows:

1. Define an embedding surface $v(\boldsymbol{x})$ which is torn along the discontinuity. For example, Figure 5.3a.
2. Embed the N-dimensional input space $\mathcal{X}$ in the $(N+1)$-dimensional space $\mathcal{V}$ via $\mathbf{v} : \mathcal{X} \to \mathcal{V}$ given by
$$\mathbf{v}(\boldsymbol{x}) = \begin{pmatrix} x_1 \\ \vdots \\ x_N \\ v(\boldsymbol{x}) \end{pmatrix}, \qquad \mathcal{V} := \mathbf{v}(\mathcal{X}). \tag{5.1}$$
3. Emulate as usual in $\mathcal{V}$ by first projecting points onto the embedding surface using $\mathbf{v}(\boldsymbol{x})$. We implicitly treat f as a function over $\mathcal{V}$ and this induces the covariance structure,
$$\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] \equiv \mathrm{Cov}[f(\mathbf{v}(\boldsymbol{x})), f(\mathbf{v}(\boldsymbol{x}'))]. \tag{5.2}$$

To elaborate on the covariance structure in $\mathcal{V}$-space, consider the following simple example:

Example 85. *Using a simple isotropic squared exponential covariance function as in (35) with $\theta = 1$ and $\sigma^2 = 1$, the covariance function in $\mathcal{V}$-space is*

$$\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \exp\left(-\|\mathbf{v}(\boldsymbol{x}) - \mathbf{v}(\boldsymbol{x}')\|^2\right) \tag{5.3}$$

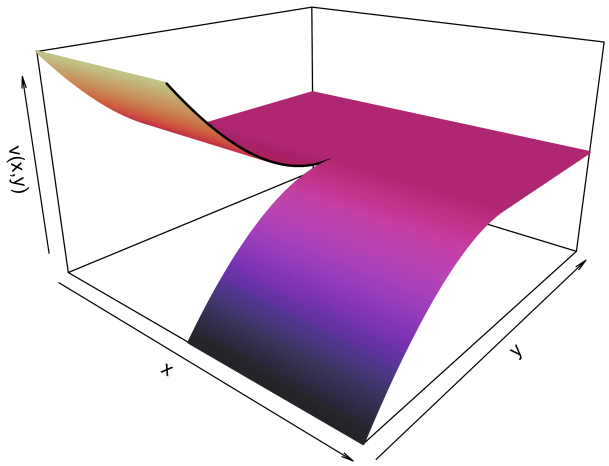
where $\|\cdot\|$ is the norm in $\mathbb{R}^{N+1}$.

We now apply this approach to our simple working example.

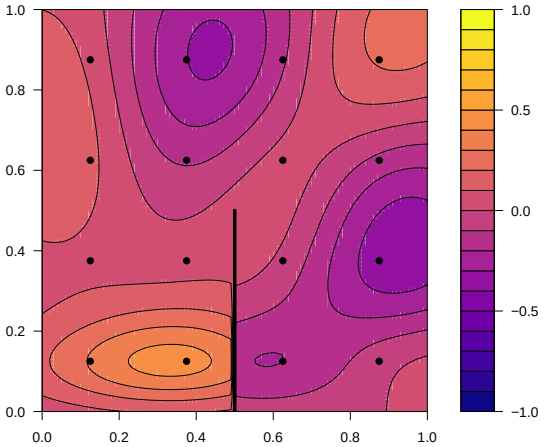
Example 81 (continuing from p. 33)**.** *To encode the known discontinuity in $f(x, y)$, we use the embedding surface*[2]

$$v(x, y) = 3\mathbb{1}_{\{y<0.5\}}(y - 0.5)^2 \operatorname{sgn}(0.5 - x) \tag{5.4}$$

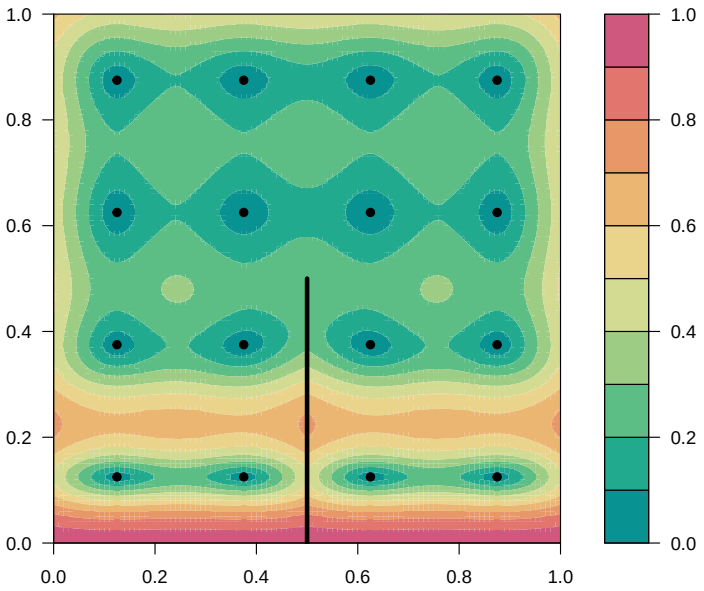
shown in Figure 5.3a. Applying a simple BL emulator, we get an emulator expectation with a discontinuity along the true discontinuity induced by $v(x, y)$, as seen in Figure 5.3. Note that we have used a grid design for the emulator for demonstration purposes only.



(a) Embedding surface $v(x, y)$ from Equation 5.4.



(b) Expectation of emulator.



(c) Standard deviation of emulator.

Figure 5.3: Simple torn embedding emulator for Example 81.

However, this approach does introduce a warping effect which needs to be dealt with. We can see this warping effect in the standard deviations of the simple torn embedding emulator in Figure 5.3c. The distance between points near the bottom of the plot has been stretched out where the embedding surface is steep which means the emulator is less certain and less effective than we would like.

[2]The function $\operatorname{sgn}(\cdot)$ is the sign, or signum function and returns the sign of a real number [80].

5.2 Introduction to the TENSE Framework

The torn embedding non-stationary emulation (TENSE) framework uses a non-stationary covariance structure which is carefully designed to reverse the warping effect from the torn embedding [15]. The TENSE framework is very flexible and enables the effective emulation of multiple arbitrary partial known discontinuities, whilst simultaneously respecting the discontinuities and fully exploiting any smoothness [15]. This approach was introduced in [15] for emulating 2D functions containing known discontinuities. In this chapter, we will generalise the TENSE framework to emulating functions over input spaces $\mathcal{X} \subset \mathbb{R}^N$ by embedding in $\mathcal{V} \subset \mathbb{R}^{N+1}$, building on the foundational work in [15].

We will assume that before embedding in $\mathcal{V}$ we desire to use a stationary covariance function similar to the examples we saw earlier. Explicitly, we will assume $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 r(\boldsymbol{\tau})$ ($\boldsymbol{\tau} = \boldsymbol{x} - \boldsymbol{x}'$) is stationary and in particular, a function of

$$(\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \boldsymbol{\Sigma}_\mathcal{X}^{-1} (\boldsymbol{x} - \boldsymbol{x}'). \tag{5.5}$$

For definiteness, we also assume the standard isotropic form,

$$\boldsymbol{\Sigma}_\mathcal{X} = \operatorname{diag}\{\theta^2, \ldots, \theta^2\}. \tag{5.6}$$

Although the approach that follows also applies to a general $\boldsymbol{\Sigma}_\mathcal{X}$ by applying a pre-transformation [15].

We will first focus on a reference point $\boldsymbol{x}_0$ and construct a covariance matrix $\boldsymbol{\Sigma}_\mathcal{V}$ which, when used with the covariance function in $\mathcal{V}$-space, reverses the warping effect locally around $\boldsymbol{x}_0$. We will then later be able to extend this to a non-stationary emulator which reverses the local warping effect over the whole input space using Theorem 52. For the local warping around $\boldsymbol{x}_0$ to be reversed, we require

$$(\mathbf{v}(\boldsymbol{x}) - \mathbf{v}(\boldsymbol{x}_0))^\mathsf{T} \boldsymbol{\Sigma}_\mathcal{V}^{-1} (\mathbf{v}(\boldsymbol{x}) - \mathbf{v}(\boldsymbol{x}_0)) \simeq (\boldsymbol{x} - \boldsymbol{x}_0)^\mathsf{T} \boldsymbol{\Sigma}_\mathcal{X}^{-1} (\boldsymbol{x} - \boldsymbol{x}_0) \tag{5.7}$$

which follows immediately from (5.2) and (5.5). Following [15], we now approximate the embedding surface $v(\boldsymbol{x})$ by the tangent plane at $\boldsymbol{x}_0$ which will allow us to reverse any linear warping effects in a neighbourhood around $\boldsymbol{x}_0$. Explicitly, we use a linear Taylor expansion,

$$\mathbf{v}(\boldsymbol{x}) - \mathbf{v}(\boldsymbol{x}_0) = T(\boldsymbol{x} - \boldsymbol{x}_0) + \mathcal{O}(\boldsymbol{x}^2), \qquad \text{where } T = \begin{bmatrix} \boldsymbol{I}_N \\ (\nabla v)^\mathsf{T} \end{bmatrix}. \tag{5.8}$$

Here, T is the linear embedding operator which raises points in $\mathcal{X}$ to the tangent plane in $\mathcal{V}$ and $\mathcal{O}(\boldsymbol{x}^2)$ represents second-order terms and above. For notational convenience, we denote partial derivatives of v at $\boldsymbol{x}_0$ by,

$$v_i := \frac{\partial v(\boldsymbol{x}_0)}{\partial x_i}, \quad \text{so } \nabla v = (v_1, \ldots, v_N)^\mathsf{T}. \tag{5.9}$$

Now if we substitute (5.8) into (5.7) and drop second-order terms and above, we obtain

$$(\boldsymbol{x} - \boldsymbol{x}_0)^\mathsf{T} T^\mathsf{T} \boldsymbol{\Sigma}_\mathcal{V}^{-1} T (\boldsymbol{x} - \boldsymbol{x}_0) \simeq (\boldsymbol{x} - \boldsymbol{x}_0)^\mathsf{T} \boldsymbol{\Sigma}_\mathcal{X}^{-1} (\boldsymbol{x} - \boldsymbol{x}_0) \tag{5.10}$$

$$\iff \qquad T^\mathsf{T} \boldsymbol{\Sigma}_\mathcal{V}^{-1} T \simeq \boldsymbol{\Sigma}_\mathcal{X}^{-1}. \tag{5.11}$$

From this equation, we will construct $\boldsymbol{\Sigma}_\mathcal{V}$. As we mentioned earlier, if we can find a form for $\boldsymbol{\Sigma}_\mathcal{V}$ which depends on $\boldsymbol{x}_0$, then we can easily build a non-stationary emulator using our $\boldsymbol{x}$-dependant covariance matrix $\boldsymbol{\Sigma}_\mathcal{V}(\boldsymbol{x})$, which will globally control the impact of the warping effect from the embedding surface. We will first derive the form of $\boldsymbol{\Sigma}_\mathcal{V}(\boldsymbol{x}_0)$ for $N = 2$ as in [15] and then generalise this argument to arbitrary N.

5.2.1 Constructing $\boldsymbol{\Sigma}_\mathcal{V}$ in 3D

For $N = 2$ ($\mathcal{V} \subset \mathbb{R}^3$), we construct $\boldsymbol{\Sigma}_\mathcal{V}(\boldsymbol{x}_0)$ to satisfy the projection constraint in (5.11) by aligning an orthonormal basis $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$ with the tangent plane at $\boldsymbol{x}_0$ and setting $\boldsymbol{\Sigma}_\mathcal{V}(\boldsymbol{x}_0)$ to be diagonal in this basis. We will show that doing this with appropriate eigenvalues will satisfy the projection constraint. This construction will allow for substantial controllable decorrelation across the discontinuities, which is not necessarily guaranteed for all covariance matrices satisfying Equation 5.11 [15].

The following construction comes directly from [15].

Setting up the Orthonormal Basis

We will construct the $\boldsymbol{w}$-basis (visualised in Figure 5.4) as follows [15]:

1. $\boldsymbol{w}_1$ will point in the maximally increasing direction of $v(x_1, x_2)$ on the tangent plane.
2. $\boldsymbol{w}_2$ will point along the level curve of $v(x_1, x_2)$ on the tangent plane.
3. $\boldsymbol{w}_3$ will be normal to the tangent plane.

By standard vector calculus results, for $\boldsymbol{w}_3$ to be normal to $v(x_1, x_2)$ at $\boldsymbol{x}_0$, we need

$$\boldsymbol{w}_3 \propto \boldsymbol{e}_3 - (v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2). \tag{5.12}$$

Similarly, for $\boldsymbol{w}_1$ to point in the maximally increasing direction, we need a 2-dimensional component parallel to $\nabla v(x_1, x_2)$ and after requiring orthogonality ($\langle \boldsymbol{w}_1, \boldsymbol{w}_3 \rangle = 0$) we obtain

$$\boldsymbol{w}_1 \propto v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + r^2\boldsymbol{e}_3, \tag{5.13}$$

where $r^2 \equiv v_1^2 + v_2^2$. We can then construct the remaining vector orthogonal to both $\boldsymbol{w}_1$ and $\boldsymbol{w}_3$ by taking the vector cross product:

$$\boldsymbol{w}_2 \propto \boldsymbol{w}_1 \times \boldsymbol{w}_2 \propto v_2(2 - r^2)\boldsymbol{e}_1 - v_1(2 - r^2)\boldsymbol{e}_2. \tag{5.14}$$

We can observe there is no $\boldsymbol{e}_3$ component which confirms $\boldsymbol{w}_2$ lies on the level curve. Finally, we choose appropriate normalising constants to obtain the orthonormal basis:

$$\boldsymbol{w}_1 = \frac{1}{c_1}\left(v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + r^2\boldsymbol{e}_3\right), \qquad \text{where } c_1^2 = r^2(r^2+1) \tag{5.15}$$

$$\boldsymbol{w}_2 = \frac{1}{c_2}\left(v_1\boldsymbol{e}_1 - v_2\boldsymbol{e}_2\right), \qquad \text{where } c_2^2 = r^2 \tag{5.16}$$

$$\boldsymbol{w}_3 = \frac{1}{c_3}\left(-v_1\boldsymbol{e}_1 - v_2\boldsymbol{e}_2 + \boldsymbol{e}_3\right), \qquad \text{where } c_3^2 = r^2 + 1 \tag{5.17}$$

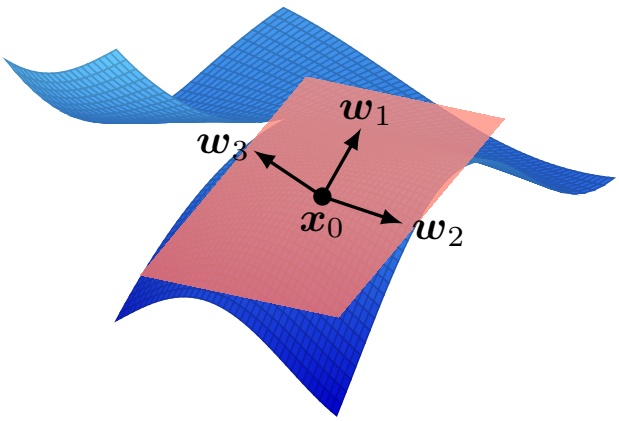


Figure 5.4: Visualisation of the $\boldsymbol{w}$-basis. An arbitrary embedding surface is plotted along with basis vectors relative to the point $\boldsymbol{x}_0$.

Remark 86. This construction of the $\boldsymbol{w}$-basis only spans $\mathcal{V}$ if $\nabla v(x_1, x_2) \neq \mathbf{0}$. If however, $\nabla v(x_1, x_2)$ is $\mathbf{0}$, this corresponds to a flat tangent plane and although we could take $\boldsymbol{w}_i = \boldsymbol{e}_i$, we will assume $\nabla v(x_1, x_2) \neq \mathbf{0} \iff r^2 \neq 0$. We will later see we derive a form for $\boldsymbol{\Sigma}_{\mathcal{V}}(\boldsymbol{x}_0)$ which behaves as expected for $r^2 = 0$.

Deriving $\boldsymbol{\Sigma}_{\mathcal{V}}$

As mentioned earlier, we construct $\boldsymbol{\Sigma}_{\mathcal{V}}$ to be diagonal in the $\boldsymbol{w}$-basis, and we suppose the corresponding eigenvalues are $\{\lambda_1^2, \lambda_2^2, \lambda_3^2\}$. Hence, we can write

$$\boldsymbol{\Sigma}_{\mathcal{V}} = \lambda_1^2 \boldsymbol{w}_1 \boldsymbol{w}_1^\mathsf{T} + \lambda_2^2 \boldsymbol{w}_2 \boldsymbol{w}_2^\mathsf{T} + \lambda_3^2 \boldsymbol{w}_3 \boldsymbol{w}_3^\mathsf{T}, \tag{5.18}$$

and similarly, since our basis is orthonormal, the inverse $\boldsymbol{\Sigma}_{\mathcal{V}}^{-1}$ is

$$\boldsymbol{\Sigma}_{\mathcal{V}}^{-1} = \frac{1}{\lambda_1^2} \boldsymbol{w}_1 \boldsymbol{w}_1^\mathsf{T} + \frac{1}{\lambda_2^2} \boldsymbol{w}_2 \boldsymbol{w}_2^\mathsf{T} + \frac{1}{\lambda_3^2} \boldsymbol{w}_3 \boldsymbol{w}_3^\mathsf{T}. \tag{5.19}$$

We want to choose the eigenvalues such that (5.11) is satisfied. Evaluating $T^\mathsf{T} \boldsymbol{\Sigma}_{\mathcal{V}}^{-1} T$,

$$T^\mathsf{T} \boldsymbol{\Sigma}_{\mathcal{V}}^{-1} T = T^\mathsf{T} \left(\frac{1}{\lambda_1^2} \boldsymbol{w}_1 \boldsymbol{w}_1^\mathsf{T} + \frac{1}{\lambda_2^2} \boldsymbol{w}_2 \boldsymbol{w}_2^\mathsf{T} + \frac{1}{\lambda_3^2} \boldsymbol{w}_3 \boldsymbol{w}_3^\mathsf{T} \right) T \tag{5.20}$$

$$= \frac{1}{\lambda_1^2} T^\mathsf{T} \boldsymbol{w}_1 \boldsymbol{w}_1^\mathsf{T} T + \frac{1}{\lambda_2^2} T^\mathsf{T} \boldsymbol{w}_2 \boldsymbol{w}_2^\mathsf{T} T \tag{5.21}$$

where $T^\mathsf{T} \boldsymbol{\Sigma}_{\mathcal{V}}^{-1} T$ does not depend on λ_3 since,

$$\boldsymbol{w}_3^\mathsf{T} T = \frac{1}{c_3} \begin{pmatrix} -v_1 & -v_2 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ v_1 & v_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}. \tag{5.22}$$

Remark 87. λ_3 is a free variable which controls the extent of decorrelation across the discontinuities [15].

Continuing to evaluate terms in (5.21),

$$\boldsymbol{w}_1^\mathsf{T} T = \frac{1}{c_1} \begin{pmatrix} v_1 & v_2 & r^2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ v_1 & v_2 \end{pmatrix} = \frac{1+r^2}{c_1} \begin{pmatrix} v_1 & v_2 \end{pmatrix} \tag{5.23}$$

$$\boldsymbol{w}_2^\mathsf{T} T = \frac{1}{c_2} \begin{pmatrix} -v_2 & v_1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ v_1 & v_2 \end{pmatrix} = \frac{1}{c_2} \begin{pmatrix} -v_2 & v_1 \end{pmatrix}. \tag{5.24}$$

Now using $c_1^2 = r^2(1+r^2)$ (5.15) and $c_2^2 = r^2$ (5.16),

$$\implies \quad \frac{1}{\lambda_1^2} T^\mathsf{T} \boldsymbol{w}_1 \boldsymbol{w}_1^\mathsf{T} T = \frac{(1+r^2)^2}{\lambda_1^2 c_1^2} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \begin{pmatrix} v_1 & v_2 \end{pmatrix} = \frac{1+r^2}{\lambda_1^2 r^2} \begin{pmatrix} v_1^2 & v_1 v_2 \\ v_1 v_2 & v_2^2 \end{pmatrix} \tag{5.25}$$

$$\frac{1}{\lambda_2^2} T^\mathsf{T} \boldsymbol{w}_2 \boldsymbol{w}_2^\mathsf{T} T = \frac{1}{\lambda_2^2 c_2^2} \begin{pmatrix} -v_2 \\ v_1 \end{pmatrix} \begin{pmatrix} -v_2 & v_1 \end{pmatrix} = \frac{1}{\lambda_2^2 r^2} \begin{pmatrix} v_2^2 & -v_1 v_2 \\ -v_1 v_2 & v_2^2 \end{pmatrix}. \tag{5.26}$$

Now we can combine the above equations (5.25, 5.26) and (5.21) with the projection constraint (5.11),

$$\begin{pmatrix} \frac{1}{\theta^2} & 0 \\ 0 & \frac{1}{\theta^2} \end{pmatrix} = \frac{1+r^2}{\lambda_1^2 r^2} \begin{pmatrix} v_1^2 & v_1 v_2 \\ v_1 v_2 & v_2^2 \end{pmatrix} + \frac{1}{\lambda_2^2 r^2} \begin{pmatrix} v_2^2 & -v_1 v_2 \\ -v_1 v_2 & v_2^2 \end{pmatrix} \tag{5.27}$$

$$= \frac{1}{r^2} \begin{pmatrix} \frac{v_2^2}{\lambda_2^2} + \frac{(1+r^2)v_1^2}{\lambda_1^2} & \left(\frac{1+r^2}{\lambda_1^2} - \frac{1}{\lambda_2^2} \right) v_1 v_2 \\ \left(\frac{1+r^2}{\lambda_1^2} - \frac{1}{\lambda_2^2} \right) v_1 v_2 & \frac{v_2^2}{\lambda_2^2} + \frac{(1+r^2)v_2^2}{\lambda_1^2} \end{pmatrix} \tag{5.28}$$

and equating the off-diagonal terms gives:

$$\left(\frac{1+r^2}{\lambda_1^2} - \frac{1}{\lambda_2^2} \right) v_1 v_2 = 0, \tag{5.29}$$

which holds if and only if one of the following cases holds:

1. $\lambda_1^2 = \lambda_2^2(1+r^2)$,

$$\implies \begin{pmatrix} \frac{1}{\theta^2} & 0 \\ 0 & \frac{1}{\theta^2} \end{pmatrix} = \frac{1}{r^2} \begin{pmatrix} \frac{v_2^2}{\lambda_2^2} + \frac{(1+r^2)v_1^2}{\lambda_2^2(1+r^2)} & 0 \\ 0 & \frac{v_2^2}{\lambda_2^2} + \frac{(1+r^2)v_2^2}{\lambda_2^2(1+r^2)} \end{pmatrix} = \begin{pmatrix} \frac{1}{\lambda_2^2} & 0 \\ 0 & \frac{1}{\lambda_2^2} \end{pmatrix} \iff \lambda_2^2 = \theta^2. \tag{5.30}$$

2. $v_1 = 0$ or $v_2 = 0$, by symmetry assume $v_1 = 0$,

$$\implies \begin{pmatrix} \frac{1}{\theta^2} & 0 \\ 0 & \frac{1}{\theta^2} \end{pmatrix} = \begin{pmatrix} \frac{1}{\lambda_2^2} & 0 \\ 0 & \frac{1+r^2}{\lambda_1^2} \end{pmatrix} \iff \lambda_2^2 = \theta^2 \text{ and } \lambda_1^2 = \lambda_2^2(1+r^2). \tag{5.31}$$

Therefore, the projection constraint (5.11) is satisfied by specifying the first two eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{V}}$ as

$$\lambda_1^2 = \theta^2(1+r^2) \qquad \text{and} \qquad \lambda_2^2 = \theta^2 \tag{5.32}$$

and the form of $\boldsymbol{\Sigma}_{\mathcal{V}}$ we chose in (5.18) is valid.

Remark 88 (Geometric intuition). Geometrically, in the maximally increasing direction (along $\boldsymbol{w}_1$), we are locally scaling the correlation length θ^2 by a factor of $(1+r^2)$; this is to account for the fact that in this direction, the distances are being stretched by the embedding surface. Conversely, along the level curve (in direction of $\boldsymbol{w}_2$) we are not scaling the correlation length as there is no local warping in this direction. For $\boldsymbol{w}_3$, which is, by construction, orthogonal to $v(x_1, x_2)$ at $\boldsymbol{x}_0$, it makes sense that this is a free parameter since we are only interested in points on the embedding surface. We will use this geometric intuition when generalising the TENSE framework to arbitrary dimension N.

We can now fully construct $\boldsymbol{\Sigma}_{\mathcal{V}}$ using equations (5.15), (5.16), (5.17), (5.18) and (5.32):

$$\lambda_1^2 \boldsymbol{w}_1 \boldsymbol{w}_1^{\mathsf{T}} = \frac{\lambda_1^2}{c_1^2} \begin{pmatrix} v_1 \\ v_2 \\ r^2 \end{pmatrix} \begin{pmatrix} v_1 & v_2 & r^2 \end{pmatrix} = \frac{\theta^2}{r^2} \begin{pmatrix} v_1^2 & v_1 v_2 & v_1 r^2 \\ v_1 v_2 & v_2^2 & v_2 r^2 \\ v_1 r^2 & v_2 r^2 & r^4 \end{pmatrix} \tag{5.33}$$

$$\lambda_2^2 \boldsymbol{w}_2 \boldsymbol{w}_2^{\mathsf{T}} = \frac{\lambda_2^2}{c_2^2} \begin{pmatrix} -v_2 \\ v_1 \\ 0 \end{pmatrix} \begin{pmatrix} -v_2 & v_1 & 0 \end{pmatrix} = \frac{\theta^2}{r^2} \begin{pmatrix} v_2^2 & -v_1 v_2 & 0 \\ -v_1 v_2 & v_1^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{5.34}$$

$$\lambda_3^2 \boldsymbol{w}_3 \boldsymbol{w}_3^{\mathsf{T}} = \frac{\lambda_3^2}{c_3^2} \begin{pmatrix} -v_1 \\ -v_2 \\ 1 \end{pmatrix} \begin{pmatrix} -v_1 & -v_2 & 1 \end{pmatrix} = \frac{\lambda_3^2}{r^2+1} \begin{pmatrix} v_1^2 & v_1 v_2 & -v_1 \\ v_1 v_2 & v_2^2 & -v_2 \\ -v_1 & -v_2 & 1 \end{pmatrix}. \tag{5.35}$$

Explicitly, $\boldsymbol{\Sigma}_{\mathcal{V}}(\boldsymbol{x}_0) = \lambda_1^2 \boldsymbol{w}_1 \boldsymbol{w}_1^{\mathsf{T}} + \lambda_2^2 \boldsymbol{w}_2 \boldsymbol{w}_2^{\mathsf{T}} + \lambda_3^2 \boldsymbol{w}_3 \boldsymbol{w}_3^{\mathsf{T}}$ and so

$$\boldsymbol{\Sigma}_{\mathcal{V}}(\boldsymbol{x}_0) = \begin{pmatrix} \theta^2 + \frac{\lambda_3^2 v_1^2}{r^2+1} & \frac{\lambda_3^2 v_1 v_2}{r^2+1} & v_1\left(\theta^2 - \frac{\lambda_3^2}{r^2+1}\right) \\ \frac{\lambda_3^2 v_1 v_2}{r^2+1} & \theta^2 + \frac{\lambda_3^2 v_2^2}{r^2+1} & v_2\left(\theta^2 - \frac{\lambda_3^2}{r^2+1}\right) \\ v_1\left(\theta^2 - \frac{\lambda_3^2}{r^2+1}\right) & v_2\left(\theta^2 - \frac{\lambda_3^2}{r^2+1}\right) & \theta^2 r^2 + \frac{\lambda_3^2}{r^2+1} \end{pmatrix}. \tag{5.36}$$

Using this expression in the covariance function whilst emulating in $\mathcal{V}$-space will reverse the warping effect from the embedding surface locally around $\boldsymbol{x}_0$. We will now look at how we can use the non-stationary covariance functions from section 3.4 to globally control the warping effect induced by the embedding surface.

Remark 89. Continuing from Remark 86, we can note that when $v(x_1, v_2)$ has no curvature at $\boldsymbol{x}_0$ ($\nabla v(x_1, x_1) = \boldsymbol{0}$), we obtain

$$\boldsymbol{\Sigma}_{\mathcal{V}}(\boldsymbol{x}_0) = \begin{pmatrix} \theta^2 & 0 & 0 \\ 0 & \theta^2 & 0 \\ 0 & 0 & \lambda_3^2 \end{pmatrix}. \tag{5.37}$$

Therefore, if the embedding surface is trivial, the covariance structure in $\mathcal{V}$-space would be identical to the covariance structure in $\mathcal{X}$-space.

Controlling the Warping Effect Globally

Whilst the covariance matrix in (5.36) will reverse the warping effect induced by the embedding surface locally around $\boldsymbol{x}_0$, we would like to reverse the warping effect globally. To achieve this, we will use a non-stationary covariance function. Specifically, we will consider the covariance functions from Theorem 52. This will allow us to use any isotropic covariance function for $\mathcal{X}$-space, and as discussed earlier, a pre-transformation can easily be used to extend this to arbitrary stationary covariance functions. By using this form of non-stationary covariance function, we are essentially averaging the covariance matrices $\boldsymbol{\Sigma}_{\mathcal{V}}(\boldsymbol{x})$ and $\boldsymbol{\Sigma}_{\mathcal{V}}(\boldsymbol{x}')$ (whilst maintaining a valid covariance structure) to resist the local warping effect of the embedding surface up to first order [15]. This approach does not always exactly reverse the warping effect and for distant points, higher-order effects may become noticeable. However, this is mostly mitigated by choosing a reasonable covariance function in $\mathcal{X}$-space (for example, squared exponential with modest correlation length) [15].

This approach, as introduced in [15], is referred to as the torn embedding non-stationary emulation (TENSE) framework, and is a powerful tool for emulating functions with known partial discontinuities. We will now apply the TENSE framework to some simple examples before generalising this approach to an arbitrary dimension N.

Example 81 (continuing from p. 33)**.** *We build a TENSE emulator with the same embedding surface (5.4) as before. Figure 5.5 shows the standard deviation for the emulator; comparing with the standard deviation plot of the standard torn embedding emulator in Figure 5.3c, we can see that the TENSE emulator has mostly counteracted the warping effect. For the expectation of the TENSE emulator, see Figure B.2b in the appendix.*

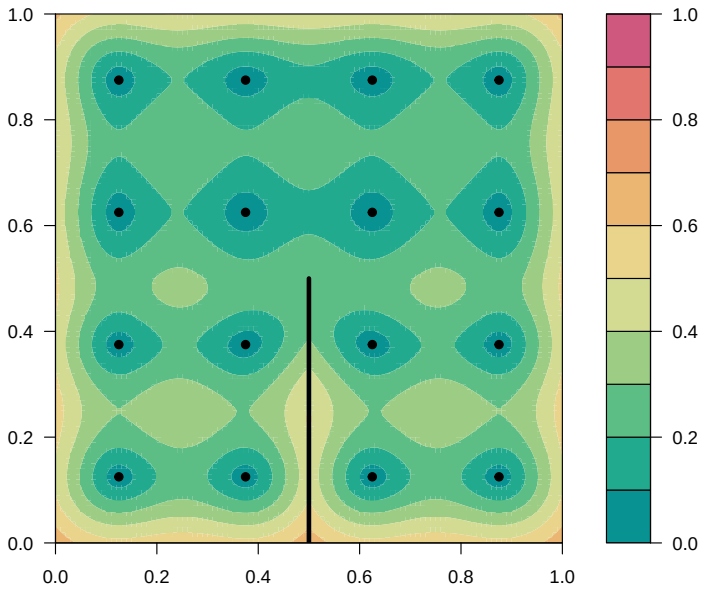
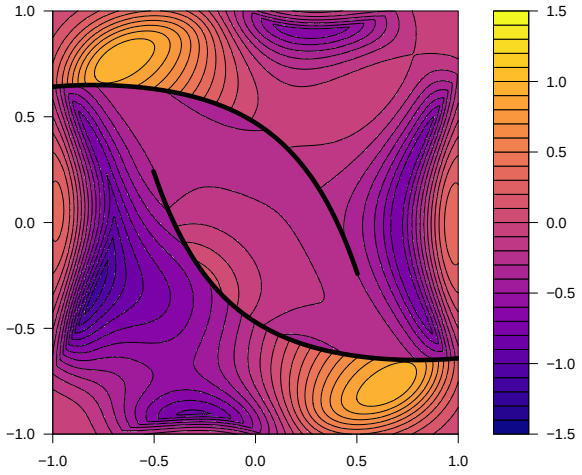


Figure 5.5: Standard deviation of the TENSE emulator for Example 81.

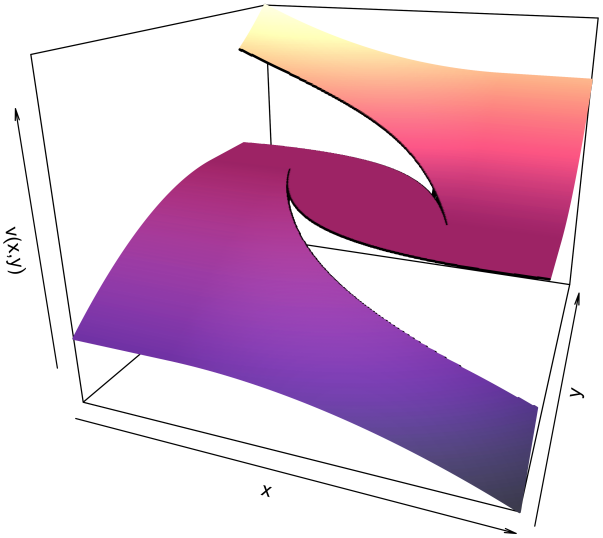
As mentioned previously, the TENSE framework is very flexible and can be applied to situations with multiple arbitrary discontinuities. The main challenge is usually finding a suitable embedding surface.

Example 90. *We now consider the 2D function $f(x, y)$ shown in Figure 5.6a with multiple curved partial discontinuities. The explicit definition is given in the appendix (B.8). We will build a TENSE emulator for this function with a Matérn ($\nu = \frac{5}{2}$) covariance structure; we have chosen this covariance function mainly to demonstrate the flexibility of TENSE with different covariance functions, however, it may not be the ideal choice. Similarly, we will use a grid design for demonstration but with a larger grid than in previous examples.*

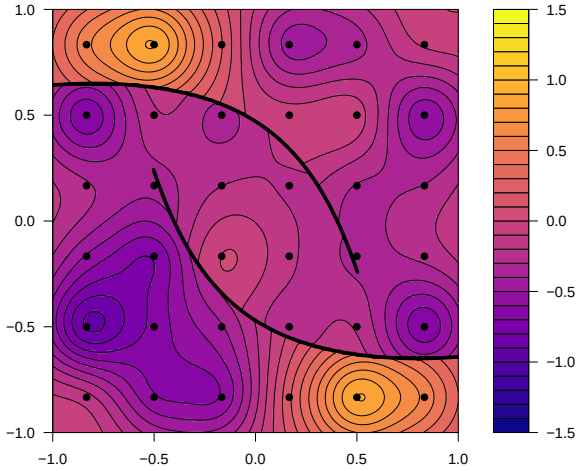
We encode the discontinuities with the embedding surface given in Figure 5.6b (given explicitly in the appendix (B.7)). Observing the emulator standard deviation in Figure 5.6d, we can see the TENSE framework has reversed the visible warping from the embedding surface. However, our choice of covariance structure is probably not ideal for this function.



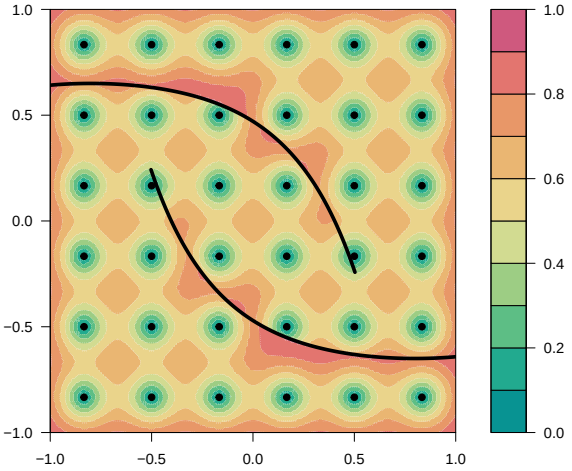
(a) True function $f(x, y)$ with discontinuities.



(b) Embedding surface $v(x, y)$.



(c) Expectation of emulator.



(d) Standard deviation of emulator.

Figure 5.6: Plot of $f(x, y)$ from Example 90, along with the embedding surface and TENSE emulator expectation and standard deviation. *A 3D plot of $f(x, y)$ is provided in the appendix, in Figure B.3.*

5.3 Generalising the TENSE Framework to Arbitrary Dimension

So far, we have considered the TENSE framework for 2D input spaces. We would like to extend this framework to be able to deal with higher-dimension input spaces. This would broaden the scope of computer models it can be applied to. For an arbitrary dimension, $\mathcal{X} \subset \mathbb{R}^N$ and $\mathcal{V} \subset \mathbb{R}^{N+1}$, we will construct a valid $\mathbf{\Sigma}_{\mathcal{V}}$ satisfying the projection constraint (5.11). This builds on the construction for $N = 2$ in [15] and similarly considers components of $\mathbf{\Sigma}_{\mathcal{V}}$ in the:

- maximally increasing direction where scaling is required,
- direction normal to the tangent space where scaling becomes a free variable,
- and directions inside the level space where no scaling should be done

but applies a geometric argument to generalise the construction.

Theorem 91. *For dimension $N \in \mathbb{N}$, the projection constraint (5.11) is satisfied by taking $\boldsymbol{\Sigma}_{\mathcal{V}}$ to be the positive semi-definite matrix,*

$$\boldsymbol{\Sigma}_{\mathcal{V}} = \lambda^2_{grad}\boldsymbol{M}_{grad} + \theta^2\boldsymbol{M}_{level} + \lambda^2\boldsymbol{M}_{norm} \tag{5.38}$$

where $\lambda^2_{grad} = \theta^2(1+r^2)$, $\lambda^2 \neq 0$ a free parameter and,

$$\boldsymbol{M}_{grad} = \frac{1}{r^2(1+r^2)}\boldsymbol{w}_{grad}\boldsymbol{w}_{grad}^{\mathsf{T}}, \qquad \text{where } \boldsymbol{w}_{grad} = \begin{pmatrix} v_1 \\ \vdots \\ v_N \\ r^2 \end{pmatrix} \tag{5.39}$$

$$\boldsymbol{M}_{level} = \begin{bmatrix} \boldsymbol{A} & 0 \\ 0 & 0 \end{bmatrix}, \qquad \text{where } \boldsymbol{A} = \boldsymbol{I}_N - \frac{1}{r^2}(\nabla v)(\nabla v)^{\mathsf{T}} \tag{5.40}$$

$$\boldsymbol{M}_{norm} = \frac{1}{1+r^2}\boldsymbol{w}_{norm}\boldsymbol{w}_{norm}^{\mathsf{T}}, \qquad \text{where } \boldsymbol{w}_{norm} = \begin{pmatrix} -v_1 \\ \vdots \\ -v_N \\ 1 \end{pmatrix} \tag{5.41}$$

where $r^2 \equiv v_1^2 + \ldots + v_N^2$ and $\boldsymbol{w}_{grad}$, $\boldsymbol{w}_{norm}$ are not necessarily unit vectors.

Before we can prove this Theorem, we must first show that the inverse of $\boldsymbol{\Sigma}_{\mathcal{V}}$ is analogous to the form in (5.19). Since we have constructed an ansatz (5.40) for $\boldsymbol{M}_{\text{level}}$, we cannot conclude this immediately from the orthonormal vector construction.

Lemma 92. *The inverse of $\boldsymbol{\Sigma}_{\mathcal{V}}$ from Theorem 91 is given by*

$$\boldsymbol{\Sigma}_{\mathcal{V}}^{-1} = \frac{1}{\lambda^2_{grad}}\boldsymbol{M}_{grad} + \frac{1}{\theta^2}\boldsymbol{M}_{level} + \frac{1}{\lambda^2}\boldsymbol{M}_{norm}. \tag{5.42}$$

Proof. By direct calculation,

$$\begin{aligned}
\boldsymbol{M}_{\text{grad}}\boldsymbol{M}_{\text{level}} &\propto \boldsymbol{w}_{\text{grad}}\begin{pmatrix} v_1 & \cdots & v_N & r^2 \end{pmatrix}\begin{pmatrix} \left(I_N - \frac{1}{r^2}(\nabla \boldsymbol{v})(\nabla \boldsymbol{v})^{\mathsf{T}}\right) & 0 \\ 0 & 0 \end{pmatrix} \\
&= \boldsymbol{w}_{\text{grad}}\begin{pmatrix} v_1\left(1 - \frac{1}{r^2}(v_1^2 + \ldots v_N^2)\right) & \cdots & v_N\left(1 - \frac{1}{r^2}(v_1^2 + \ldots v_N^2)\right) & 0 \end{pmatrix} \\
&= \boldsymbol{0} \\
\boldsymbol{M}_{\text{grad}}\boldsymbol{M}_{\text{norm}} &\propto \boldsymbol{w}_{\text{grad}}\begin{pmatrix} v_1 & \cdots & v_N & r^2 \end{pmatrix}\begin{pmatrix} -v_1 \\ \vdots \\ -v_N \\ 1 \end{pmatrix}\boldsymbol{w}_{\text{norm}}^{\mathsf{T}} \\
&= \boldsymbol{w}_{\text{grad}}\left(-v_1^2 - \ldots - v_N^2 + r^2\right)\boldsymbol{w}_{\text{norm}}^{\mathsf{T}} \\
&= \boldsymbol{0} \\
\boldsymbol{M}_{\text{level}}\boldsymbol{M}_{\text{norm}} &\propto \begin{pmatrix} \left(I_N - \frac{1}{r^2}(\nabla \boldsymbol{v})(\nabla \boldsymbol{v})^{\mathsf{T}}\right) & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} -v_1 \\ \vdots \\ -v_N \\ 1 \end{pmatrix}\boldsymbol{w}_{\text{norm}}^{\mathsf{T}} \\
&= \begin{pmatrix} v_1\left(-1 + \frac{r^2}{r^2}\right) \\ \vdots \\ v_N\left(-1 + \frac{r^2}{r^2}\right) \\ 0 \end{pmatrix}\boldsymbol{w}_{\text{norm}}^{\mathsf{T}} \\
&= \boldsymbol{0}
\end{aligned}$$

Similarly it can be shown that $\boldsymbol{M}_{\text{level}}\boldsymbol{M}_{\text{grad}} = \boldsymbol{0}$, $\boldsymbol{M}_{\text{norm}}\boldsymbol{M}_{\text{level}} = \boldsymbol{0}$ and $\boldsymbol{M}_{\text{norm}}\boldsymbol{M}_{\text{grad}} = \boldsymbol{0}$, whence,

$$\left(\frac{1}{\lambda_{\text{grad}}^2}\boldsymbol{M}_{\text{grad}} + \frac{1}{\theta^2}\boldsymbol{M}_{\text{level}} + \frac{1}{\lambda^2}\boldsymbol{M}_{\text{norm}}\right)\left(\lambda_{\text{grad}}^2\boldsymbol{M}_{\text{grad}} + \theta^2\boldsymbol{M}_{\text{level}} + \lambda^2\boldsymbol{M}_{\text{norm}}\right)$$

is given by,

$$\boldsymbol{M}_{\text{grad}}^2 + \boldsymbol{M}_{\text{level}}^2 + \boldsymbol{M}_{\text{norm}}^2 = \boldsymbol{M}_{\text{grad}} + \boldsymbol{M}_{\text{level}} + \boldsymbol{M}_{\text{norm}},$$

since $\boldsymbol{w}_{\text{grad}}^\mathsf{T}\boldsymbol{w}_{\text{grad}} = r^2(1+r^2)$, $\boldsymbol{w}_{\text{norm}}^\mathsf{T}\boldsymbol{w}_{\text{norm}} = 1 + r^2$ and

$$\boldsymbol{A}^2 = \boldsymbol{I}_N^2 - \frac{2}{r^2}(\nabla v)(\nabla v)^\mathsf{T} + \frac{1}{r^4}(\nabla v)\cancel{\left((\nabla v)^\mathsf{T}(\nabla v)\right)}^{r^2}(\nabla v)^\mathsf{T} = \boldsymbol{A}.$$

Now finally, for $i, j \leq N$,

$$[\boldsymbol{M}_{\text{grad}} + \boldsymbol{M}_{\text{level}} + \boldsymbol{M}_{\text{norm}}]_{i,j} = \left(\frac{v_i v_j}{r^2(1+r^2)}\right) + \left(\delta_{i,j} - \frac{v_i v_j}{r^2}\right) + \left(\frac{v_i v_j}{1+r^2}\right) = \delta_{i,j},$$

for $i \neq j$, $i = N+1$ (or analogously $j = N+1$),

$$[\boldsymbol{M}_{\text{grad}} + \boldsymbol{M}_{\text{level}} + \boldsymbol{M}_{\text{norm}}]_{i,j} = \frac{v_j}{1+r^2} + 0 - \frac{v_j}{1+r^2} = 0,$$

for $i = j = N+1$,

$$[\boldsymbol{M}_{\text{grad}} + \boldsymbol{M}_{\text{level}} + \boldsymbol{M}_{\text{norm}}]_{i,j} = \frac{r^2}{1+r^2} + 0 + \frac{1}{1+r^2} = 1,$$

and so we can conclude $\boldsymbol{M}_{\text{grad}} + \boldsymbol{M}_{\text{level}} + \boldsymbol{M}_{\text{norm}} = \boldsymbol{I}_{N+1}$ which in turn proves our assertion. ■

Now we can prove Theorem 91.

Proof of Theorem 91. Given $\boldsymbol{\Sigma}_{\mathcal{V}}^{-1}$ from Lemma 92, we show that $\boldsymbol{\Sigma}_{\mathcal{V}}$ satisfies the projection constraint (5.11),

$$\begin{aligned} T^\mathsf{T}\boldsymbol{\Sigma}_{\mathcal{V}}^{-1}T &= T^\mathsf{T}\left(\frac{1}{\lambda_{\text{grad}}^2}\boldsymbol{M}_{\text{grad}} + \frac{1}{\theta^2}\boldsymbol{M}_{\text{level}} + \frac{1}{\lambda^2}\boldsymbol{M}_{\text{norm}}\right)T \\ &= \frac{1}{\lambda_{\text{grad}}^2}T^\mathsf{T}\boldsymbol{M}_{\text{grad}}T + \frac{1}{\theta^2}T^\mathsf{T}\boldsymbol{M}_{\text{level}}T + \frac{1}{\lambda^2}T^\mathsf{T}\boldsymbol{M}_{\text{norm}}T. \end{aligned}$$

Computing this directly, we have,

$$\boldsymbol{w}_{\text{grad}}^\mathsf{T}T = \begin{pmatrix} v_1 & \cdots & v_N & r^2 \end{pmatrix}\begin{pmatrix} & \boldsymbol{I}_N & \\ v_1 & \cdots & v_N \end{pmatrix} = (1+r^2)\begin{pmatrix} v_1 & \cdots & v_N \end{pmatrix}$$

$$\boldsymbol{w}_{\text{norm}}^\mathsf{T}T = \begin{pmatrix} -v_1 & \cdots & -v_N & 1 \end{pmatrix}\begin{pmatrix} & \boldsymbol{I}_N & \\ v_1 & \cdots & v_N \end{pmatrix} = \boldsymbol{0}^\mathsf{T}$$

and so,

$$\begin{aligned} \frac{1}{\lambda_{\text{grad}}^2}T^\mathsf{T}\boldsymbol{M}_{\text{grad}}T &= \frac{1}{\theta^2(1+r^2)}\frac{(1+r^2)^2}{r^2(1+r^2)}(\nabla v)(\nabla v)^\mathsf{T} = \frac{1}{\theta^2 r^2}(\nabla v)(\nabla v)^\mathsf{T} \\ \frac{1}{\lambda^2}T^\mathsf{T}\boldsymbol{M}_{\text{norm}}T &= \boldsymbol{0} \\ \frac{1}{\theta^2}T^\mathsf{T}\boldsymbol{M}_{\text{level}}T &= \begin{pmatrix} & v_1 \\ \boldsymbol{I}_N & \vdots \\ & v_N \end{pmatrix}\frac{1}{\theta^2}\begin{pmatrix} \left(I_N - \frac{1}{r^2}(\nabla v)(\nabla v)^\mathsf{T}\right) & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{pmatrix}\begin{pmatrix} & \boldsymbol{I}_N & \\ v_1 & \cdots & v_N \end{pmatrix} \\ &= \frac{1}{\theta^2}\left(\boldsymbol{I}_N - \frac{1}{r^2}(\nabla v)(\nabla v)^\mathsf{T}\right). \end{aligned}$$

Hence,

$$\begin{aligned} T^\intercal \boldsymbol{\Sigma}_{\mathcal{V}}^{-1} T &= \frac{1}{\cancel{\theta^2 r^2}} \cancel{(\nabla v)(\nabla v)^\intercal} + \frac{1}{\theta^2} \boldsymbol{I}_N - \frac{1}{\cancel{\theta^2 r^2}} \cancel{(\nabla v)(\nabla v)^\intercal} \\ &= \frac{1}{\theta^2} \boldsymbol{I}_N = \boldsymbol{\Sigma}_{\mathcal{X}}^{-1}. \end{aligned}$$

So $\boldsymbol{\Sigma}_{\mathcal{V}}$ satisfies the projection constraint (5.11). It only remains to show that $\boldsymbol{\Sigma}_{\mathcal{V}}$ is indeed positive semi-definite. Analogously to what we did in the $N = 2$ case, we can construct an orthonormal basis for $\mathcal{V}$,

$$\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N, \boldsymbol{w}_{N+1},$$

where $\boldsymbol{w}_1 = \frac{\boldsymbol{w}_{\text{grad}}}{\|\boldsymbol{w}_{\text{grad}}\|}$, $\boldsymbol{w}_{N+1} = \frac{\boldsymbol{w}_{\text{norm}}}{\|\boldsymbol{w}_{\text{norm}}\|}$ and the level space is spanned by $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N$ which can be found by Gram-Schmidt. The construction for $\boldsymbol{\Sigma}_{\mathcal{V}}$ is then analogous to the case for $N = 2$ but now we have replaced

$$\theta^2 \left(\boldsymbol{w}_2 \boldsymbol{w}_2^\intercal + \ldots + \boldsymbol{w}_N \boldsymbol{w}_N^\intercal \right)$$

with $\theta^2 \boldsymbol{M}_{\text{level}}$ as an ansatz to avoid computing this basis directly. However, we can still conclude that via this construction, the eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{V}}$ are $\theta^2(1 + r^2)$, θ^2 (multiplicity $N - 1$) and λ^2 which are all positive, and hence by Lemma 29, we can conclude $\boldsymbol{\Sigma}_{\mathcal{V}}$ is positive semi-definite. ■

We will now apply this theory to an example in the following section.

5.4 Applying TENSE in 3D

In this section, we will consider a 3D function $f(x, y, z)$ (given explicitly in the appendix (B.14)) with a discontinuity. Explicitly, the input space is $\mathcal{X} = [0, 1]^3$ and the region of discontinuity is a semi-ellipse lying on the plane $y = 0.5$; the discontinuous region is all points such that $(x - 0.5)^2 + (0.5z)^2 \leq \frac{1}{3}$. This is shown in Figure 5.7 below.

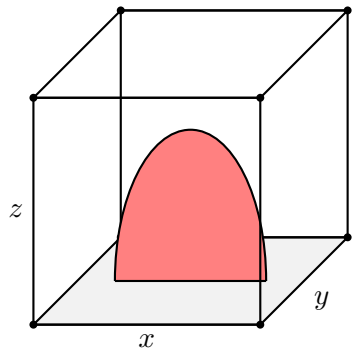


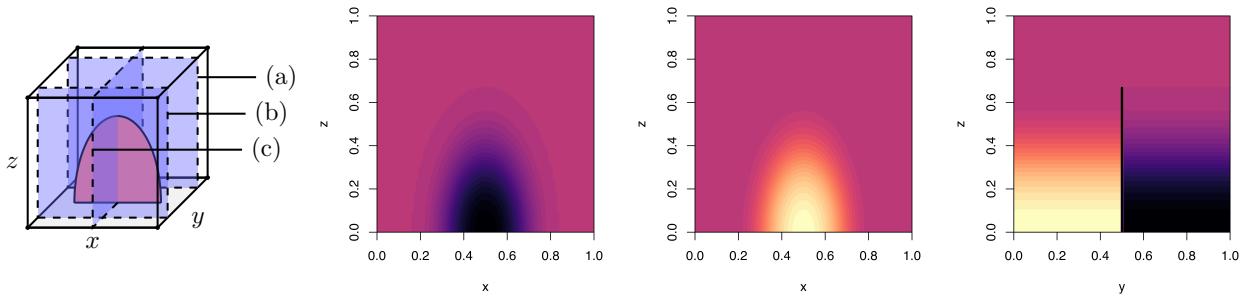
Figure 5.7: Diagram of the input space for f with the discontinuity in red.

Since the domain of this function is 3D, we will visualise slices of the input domain. See Figure 5.9 for slices of the true function.

In higher dimensions, constructing an embedding surface can be challenging. The details of how we constructed $v(\boldsymbol{x})$ can be found in subsection B.3.3. Figure 5.8 shows the embedding surface we have used here at critical slices, it only depends on y via a step function at $y = 0.5$ to sign either side of the discontinuity.

5.4.1 Advanced TENSE Emulator

We emulate the true function with an advanced non-stationary TENSE emulator, which includes a regression term and implements the TENSE framework. The accompanying code can be found in [16].



(a) $y = 0.8$. (b) $y = 0.2$. (c) $x = 0.5$.

Figure 5.8: Embedding surface $v(x, y, z)$ (B.13) at three slices (shown left).

We assume we have the prior knowledge that the following basis functions could be useful: 1, x, y, z, x^2, y^2, z^2, xz. Since this is not a realistic computer model and we do not have any strong prior beliefs, we make somewhat arbitrary best-guess prior estimates for the regression coefficients, but with knowledge of the function definition in mind to mimic a realistic scenario. However, we place a high prior variance to reflect a low confidence in our prior estimates.

Since we want to accurately emulate the function over the entire input domain (unlike history matching), we will use a large design of $d = 80$ points. Specifically, we use a precomputed maximin design from [60]. We choose to use a squared exponential covariance function with correlation length $\theta = 0.22$ as the base stationary covariance function, and we choose the free variable which controls the decorrelation across the discontinuity to be $\lambda^2 = 0.06$. In this case, these values were optimally selected by minimising the mean squared error (MSE) of the emulator expectation; however, in a real application this approach would not be available, and these values should be chosen based on prior beliefs.

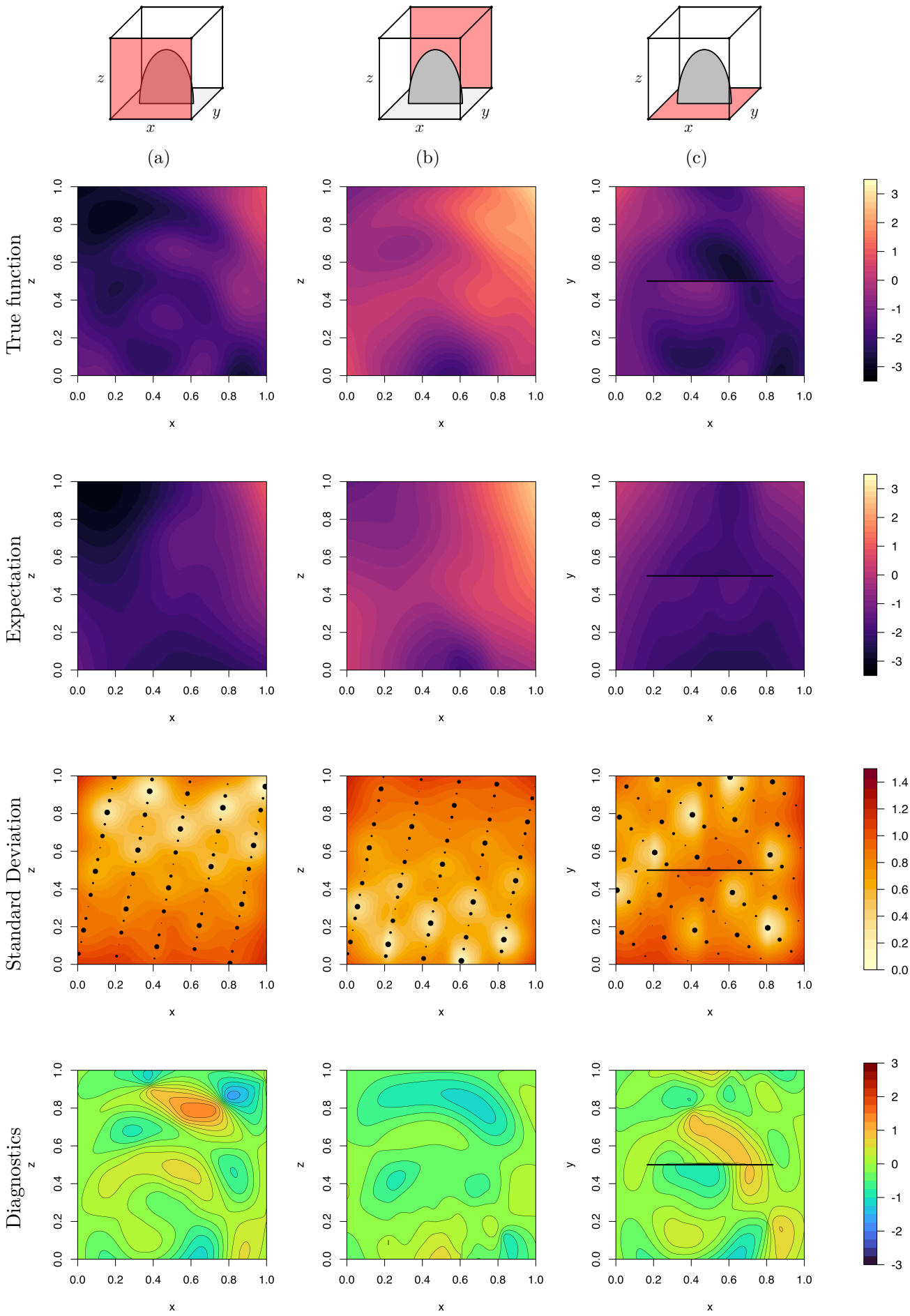
Figure 5.9 (across subsequent pages) shows the true function, emulator expectation, emulator standard deviation and diagnostics for nine different slices (the faces/boundaries and three internal slices). Each column represents a different slice (visualised with a diagram at the top). Table 5.1 shows the coefficients for the regression surface of the emulator; they are not negligible which suggests the addition of the regression surface was sensible for this function.

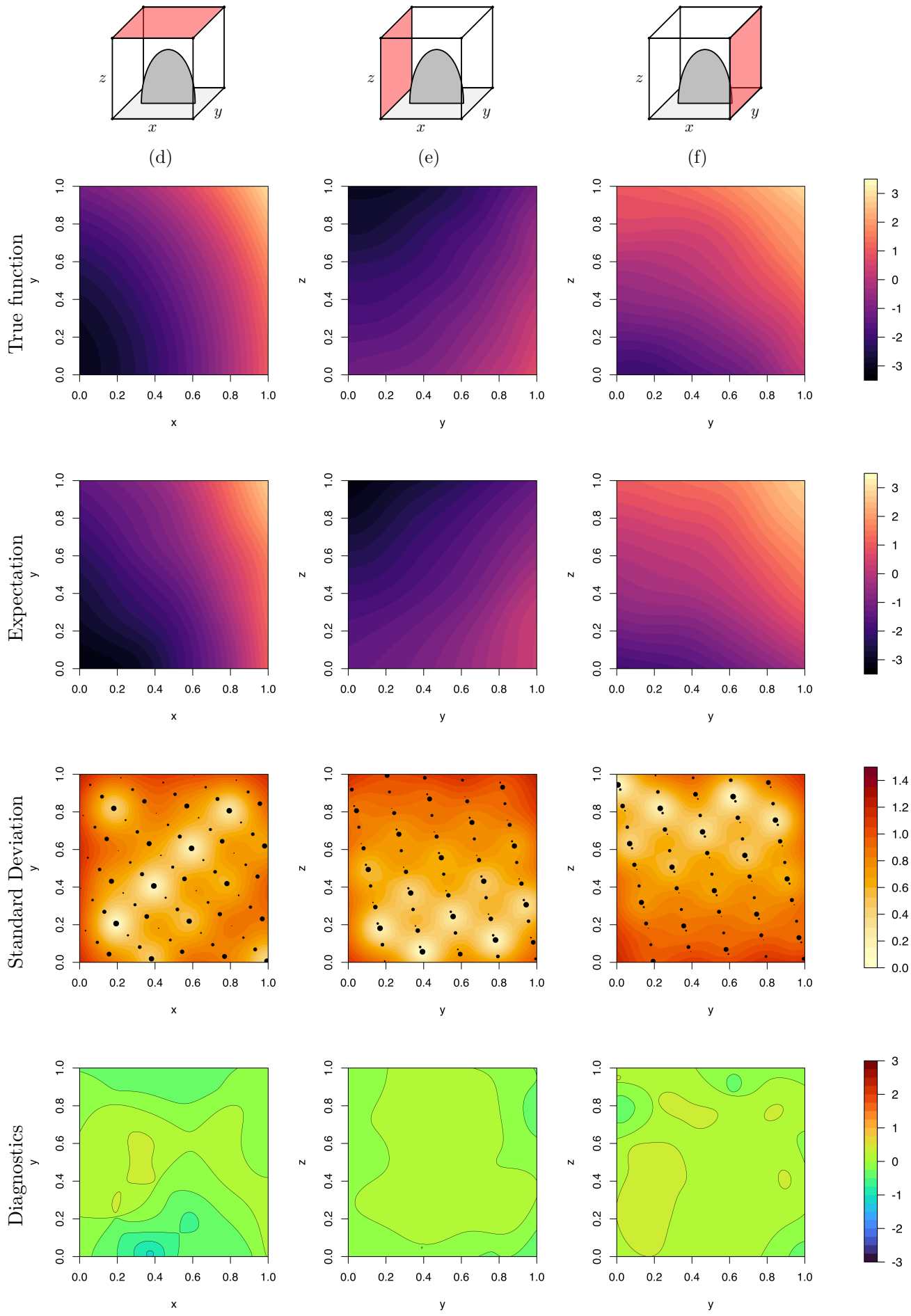
Basis Function	1	x	y	z	x^2	y^2	z^2	xz
Coefficient	-1.386	-3.907	0.534	-0.964	3.272	1.191	-0.823	4.807

Table 5.1: Regression coefficients for basis functions in the emulator.

Comparing the true function to the emulator's expectation, the emulator performs very well. Looking at the diagnostics, the absolute error of the emulator stays within one standard deviation, suggesting the standard deviations are on the cautious side. The emulator fails to capture some of the smaller details, but this is to be expected and can be improved by taking a larger design.

Remark 93. For code implementing the TENSE in arbitrary dimension, see Appendix C. For full details, advanced variants and examples, see the GitHub repository (`github.com/jreaso/mmath-thesis`) [16].


z
y
x
(a)
(b)
(c)
True function
Expectation
Standard Deviation
Diagnostics
x
y
z
0.0
0.2
0.4
0.6
0.8
1.0
3
2
1
0
-1
-2
-3
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0


(d)
(e)
(f)
True function
Expectation
Standard Deviation
Diagnostics
x
y
z

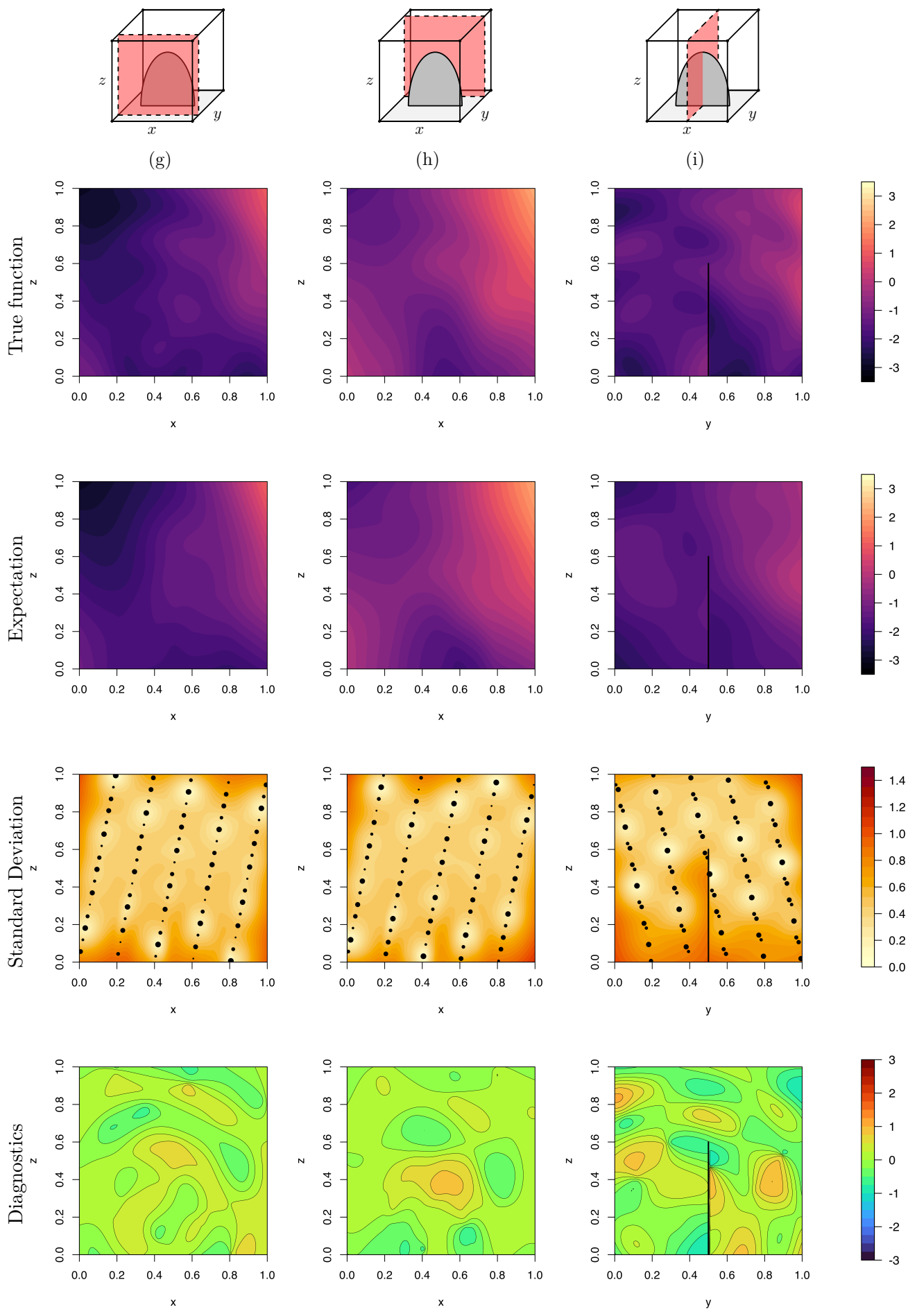


Figure 5.9: Emulator plots for $f(x, y, z)$ at different slices. In the standard deviation plots, the design points are also plotted and their size is proportional to the distance from the plane.

Chapter 6

Conclusion

In this project, we looked at the Bayes Linear framework and built a powerful Bayesian emulator capable of efficiently emulating a wide class of computer models, exploiting prior beliefs. This project has focused on summarizing the key theoretical concepts that are foundational for constructing efficient emulators. We also looked at the TENSE framework, which extends the class of functions (computer models) we can emulate to include functions with structured partial discontinuities. The work in section 5.3 builds on the foundational work introducing the TENSE framework [15] and enables the TENSE framework to be applied to functions with arbitrary dimension input spaces. However, there are still many interesting directions to explore in this area. A logical next step would be to apply this theory to real-world computer models.

As discussed in [15], complicated networks of discontinuities may involve embedding in higher dimensions than the TENSE framework currently allows. However, we may also gain additional flexibility in embedding by allowing embedding functions $\mathbf{v}$ which involve transformations to the initial inputs, such as an embedding surface wrapped back over itself. Although we briefly considered periodic domains, this project has mostly focused on emulating computer models with input domains in $\mathbb{R}^N$. However, it is possible to construct emulators for computer models defined on manifolds [32]; it would be interesting to explore the TENSE framework in this context. An example of an application this could be applied to is global ocean models where land masses enforce discontinuities on the spherical input domain.

Another current limitation of the TENSE framework is the type of base covariance function we can have in $\mathcal{X}$-space. The current construction allows for a wide class of stationary covariance functions, however, the TENSE framework is not currently able to reverse the warping induced by an embedding surface whilst preserving a non-stationary covariance structure. This may be possible with an intermediary (not necessarily torn) embedding surface which we do not reverse the warping of, but we leave this exploration to future work.

To conclude, this project provides a comprehensive overview of Bayesian emulation through the lens of the Bayes linear framework. It investigates in detail the TENSE framework, for emulating a special class of computer models with partial known discontinuities. The proposed extensions and future research directions showcase the immense potential of these approaches in advancing our understanding and analysis of complex systems with Bayesian emulation.

References

[1] Ian Vernon, Michael Goldstein, and Richard Bower. Galaxy formation: Bayesian history matching for the observable universe. *Statistical Science*, 29(1):81–90, 2014.

[2] Richard G. Bower, Ian Vernon, Michael Goldstein, Andrew J. Benson, Cedric G. Lacey, Carlton M. Baugh, Shaun Cole, and Carlos S. Frenk. The parameter space of galaxy formation. *Monthly Notices of the Royal Astronomical Society*, 407(4):20172045, August 2010.

[3] Ian Vernon, Michael Goldstein, and Richard G. Bower. Galaxy formation: a bayesian uncertainty analysis. *Bayesian Analysis*, 5(4):619–670, 2010.

[4] Katrin Heitmann, David Higdon, Martin White, Salman Habib, Brian J. Williams, Earl Lawrence, and Christian Wagner. The coyote universe ii: Cosmological models and precision emulation of the nonlinear matter power spectrum. *The Astrophysical Journal*, 705(1):156174, October 2009.

[5] Frédéric Hourdin, Brady Ferster, Julie Deshayes, Juliette Mignot, Ionela Musat, and Daniel Williamson. Toward machine-assisted tuning avoiding the underestimation of uncertainty in climate change projections. *Science Advances*, 9(29):2758, 2023.

[6] Philip B. Holden, Neil R. Edwards, James Hensman, and Richard D. Wilkinson. Abc for climate: dealing with expensive simulators, 2015.

[7] M. Bayarri, James Berger, Eliza Calder, Keith Dalbey, Simon Lunagomez, Abani Patra, E Bruce Pitman, Elaine Spiller, and Robert Wolpert. Using statistical and computer models to quantify volcanic hazards. *Technometrics*, 51:402–413, 11 2009.

[8] Ian Vernon, Junli Liu, Michael Goldstein, Jim Rowe, Jennifer Topping, and Keith Lindsey. Bayesian uncertainty analysis for complex systems biology models: emulation, global parameter searches and evaluation of gene functions. *BMC Systems Biology*, 12(1):1607–06358, 2018.

[9] Samuel E. Jackson, Ian Vernon, Junli Liu, and Keith Lindsey. Understanding hormonal crosstalk in arabidopsis root development via emulation and history matching. *Statistical Applications in Genetics and Molecular Biology*, 19(2):20180053, 2020.

[10] Alexander I. J. Forrester, András Sóbester, and Andy J. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide.* John Wiley & Sons, Ltd, 2008. First published: 18 July 2008.

[11] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020.

[12] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[13] David Wooff and Michael Goldstein. *Bayes linear statistics: theory & methods.* John Wiley & Sons, 2004.

[14] Andrew Iskauskas, Ian Vernon, Michael Goldstein, Danny Scarponi, Trevelyan J. McKinley, Richard G. White, and Nicky McCreesh. Emulation and history matching using the hmer package, 2023.

[15] Ian Vernon, Jonathan Owen, and Jonathan Carter. Bayesian emulation for computer models with multiple partial discontinuities, 2022.

[16] Jamie Reason. Advances in emulation and the tense framework supplementary code. `https://github.com/jreaso/mmath-thesis`, 2024.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[18] Wenchong He and Zhe Jiang. A survey on uncertainty quantification methods for deep neural networks: An uncertainty source perspective, 2023.

[19] Peter S. Craig, Michael Goldstein, Allan H. Seheult, and James A. Smith. Pressure matching for hydrocarbon reservoirs: A case study in the use of bayes linear strategies for large computer experiments. 04 2000.

[20] Michael Goldstein, Allan Seheult, and Ian Vernon. *Assessing Model Adequacy*, chapter 26, pages 435–449. John Wiley & Sons, Ltd, 2013.

[21] The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant arabidopsis thaliana. *Nature*, 408(6814):796–815, Dec 2000.

[22] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

[23] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[24] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–423, 1989.

[25] Kim J. Krishnan Conor Lawless Daniel A. Henderson, Richard J. Boys and Darren J. Wilkinson. Bayesian emulation and calibration of a stochastic computer model of mitochondrial dna deletions in substantia nigra neurons. *Journal of the American Statistical Association*, 104(485):76–87, 2009.

[26] Radford M. Neal. Regression and classification using Gaussian process priors (with discussion). In J. M. Bernardo et al., editors, *Bayesian statistics 6*, pages 475–501. Oxford University Press, 1998.

[27] Christopher K.I. Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[28] Christoffer Riis, Francisco Antunes, Frederik Boe Hüttel, Carlos Lima Azevedo, and Francisco Câmara Pereira. Bayesian active learning with fully bayesian gaussian processes, 2023.

[29] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Parallelised bayesian optimisation via thompson sampling. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 133–142. PMLR, 09–11 Apr 2018.

[30] Giorgio Corani, Alessio Benavoli, and Marco Zaffalon. *Time Series Forecasting with Gaussian Processes Needs Priors*, page 103117. Springer International Publishing, 2021.

[31] Sudipto Banerjee, Bradley P. Carlin, and Alan E. Gelfand. *Hierarchical Modeling and Analysis for Spatial Data*. CRC Press, Boca Raton, FL, 2 edition, 2015.

[32] Iskander Azangulov, Andrei Smolensky, Alexander Terenin, and Viacheslav Borovitskiy. Stationary kernels and gaussian processes on lie groups and their homogeneous spaces i: the compact case, 2023.

[33] Ben. Deriving the conditional distributions of a multivariate normal distribution. Stats Stack Exchange, 2019.

[34] Book of Statistical Proofs. Conditional distributions of the multivariate normal distribution.

[35] Theodore W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley, 1958.

[36] Wikipedia contributors. Schur complement. `https://en.wikipedia.org/w/index.php?title=Schur_complement`, 2023.

[37] Adi Ben-Israel and Thomas N.E. Greville. *Generalized Inverses: Theory and Applications*. Springer, New York, NY, 2nd edition, 2003.

[38] Bruno de Finetti. *Theory of Probability*, volume I. John Wiley, New York, 1974.

[39] Bruno de Finetti. *Theory of Probability*, volume II. John Wiley, New York, 1975.

[40] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1991.

[41] Ian Vernon. Uncertainty quantification iv. Lecture notes for MATH4337, 2023.

[42] Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

[43] Wikipedia contributors. Definite matrix. `https://en.wikipedia.org/w/index.php?title=Definite_matrix`, 2024.

[44] Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer New York, NY, 1 edition, 1999.

[45] Christopher J. Paciorek and Mark J. Schervish. Spatial modelling using a new class of nonstationary covariance functions. *Environmetrics*, 17(5):483–506, 2006.

[46] Wikipedia contributors. Gamma function. `https://en.wikipedia.org/w/index.php?title=Gamma_function`, 2024.

[47] Wikipedia contributors. Bessel function. `https://en.wikipedia.org/w/index.php?title=Bessel_function`, 2024.

[48] David Kristjanson Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

[49] Nicolas Durrande. Kernel design. GP Summer School, September 2015.

[50] Richard J. Rossi. *Mathematical Statistics: An Introduction to Likelihood Based Inference*. John Wiley Sons, Hoboken, NJ, 2018.

[51] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved April 5, 2024, from `http://www.sfu.ca/~ssurjano`.

[52] Kjell Magne Fauske. Annotated manipulator diagram, 2006.

[53] Bernhard Sch"olkopf and Alexander J. Smola. *Learning with Kernels*. MIT press, 2002.

[54] Christopher Joseph Paciorek. *Nonstationary Gaussian Processes for Regression and Spatial Modelling*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 2003.

[55] Mathew M. Dunlop, Mark A. Girolami, Andrew M. Stewart, and Aretha L. Teckentrup. How deep are deep gaussian processes? *Journal of Machine Learning Research*, 19:1–46, 2018.

[56] Isaac J. Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, 39(4):811–841, 1938.

[57] Iskander Azangulov, Andrei Smolensky, Alexander Terenin, and Viacheslav Borovitskiy. Stationary kernels and gaussian processes on lie groups and their homogeneous spaces ii: non-compact symmetric spaces, 2023.

[58] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[59] Art B. Owen. A central limit theorem for latin hypercube sampling. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54:541–551, 1992.

[60] Edwin van Dam, Dick den Hertog, Bart Husslage, and Gijs Rennen. Space-filling designs. `https://www.spacefillingdesigns.nl`, 2015.

[61] Andrea Grosso, A.R.M.J.U. Jamali, and M. Locatelli. Finding maximin latin hypercube designs by iterated local search heuristics. *European Journal of Operational Research*, 197(2):541–547, 2009.

[62] Pteris Audze and Valdemrs Eglais. New approach for planning out of experiments. *Problems of Dynamics and Strengths*, 35:104–107, 1977.

[63] Stuart J. Bates, Johann Sienz, and Vassili V. Toropov. Formulation of the optimal latin hypercube design of experiments using a permutation genetic algorithm. In *AIAA 2004-2011*, pages 1–7, 2004.

[64] Mattias Liefvendahl and Rafa Stocki. A study on algorithms for optimization of latin hypercubes. *Journal of Statistical Planning and Inference*, 136(9):3231–3247, 2006.

[65] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[66] Michael Dunne, Hossein Mohammadi, Peter Challenor, Rita Borgo, Thibaud Porphyre, Ian Vernon, Elif E. Firat, Cagatay Turkay, Thomas Torsney-Weir, Michael Goldstein, Richard Reeve, Hui Fang, and Ben Swallow. Complex model calibration through emulation, a worked example for a stochastic epidemic model. *Epidemics*, 39:100574, 2022.

[67] Ian Vernon, Jonathan Owen, Joseph Aylett-Bullock, Carolina Cuesta-Lazaro, Jonathan Frawley, Arnau Quera-Bofarull, Aidan Sedgewick, Difu Shi, Henry Truong, Mark Turner, Joseph Walker, Tristan Caulfield, Kevin Fong, and Frank Krauss. Bayesian emulation and history matching of june. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 380(2233):20220039, 2022.

[68] Danny Scarponi, Andrew Iskauskas, Rebecca A. Clark, Ian Vernon, Trevelyan J. McKinley, Michael Goldstein, Christinah Mukandavire, Arminder Deol, Chathika Weerasuriya, Roel Bakker, Richard G. White, and Nicky McCreesh. Demonstrating multi-country calibration of a tuberculosis model using new history matching and emulation package - hmer. *Epidemics*, 43:100678, 2023.

[69] Ioannis Andrianakis, Nick McCreesh, Ian Vernon, Trevelyan McKinley, Jeremy Oakley, Rebecca Nsubuga, and Richard White. Efficient history matching of a high dimensional individual-based hiv transmission model. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):694–719, 2017.

[70] James M. Salter, Daniel B. Williamson, and Viatcheslav V. Kharin. Uncertainty quantification for computer models with spatial output using calibration-optimal bases. *Journal of the American Statistical Association*, 114(528):1800–1814, 2019.

[71] Daniel Williamson, Michael Goldstein, Lesley Allison, Adam Blaker, Peter Challenor, Laura Jackson, and Kuniko Yamazaki. History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. *Climate Dynamics*, 41(7-8):1703–1729, 2013.

[72] Hailiang Du, Wei Sun, Michael Goldstein, and Gareth P. Harrison. Optimization via statistical emulation and uncertainty quantification: Hosting capacity analysis of distribution networks. *IEEE Access*, 9:118472–118483, 2021.

[73] Ioannis Andrianakis, Ian Vernon, Nick McCreesh, Trevelyan McKinley, Jeremy Oakley, Rebecca Nsubuga, and Richard White. Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on hiv in uganda. *PLoS Computational Biology*, 11(1):e1003968, 2015.

[74] John Bissell, Camila Caiado, Sarah Curtis, Michael Goldstein, and Brian Straughan. *Tipping Points: Modelling Social Problems and Health*. Wiley, Sep 2015.

[75] Christopher A. Pope, John Paul Gosling, Stuart Barber, Jill S. Johnson, Takanobu Yamaguchi, Graham Feingold, and Paul G. Blackwell. Gaussian process modeling of heterogeneity and discontinuities using voronoi tessellations. *Technometrics*, 63(1):53–63, 2021.

[76] TNO. Olympus oil reservoir model input decks, 2017.

[77] Eage/tno workshop on olympus field development optimization, 2018.

[78] Robert B. Gramacy and Herbert K. H. Lee. Bayesian treed gaussian process models with an application to computer modeling, 2009.

[79] Amanda Sauer, Robert B. Gramacy, and David Higdon. Active learning for deep gaussian process surrogates. *Technometrics*, pages 1–15, 2022.

[80] Wikipedia contributors. Sign function. `https://en.wikipedia.org/w/index.php?title=Sign_function`, 2024.

[81] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2 edition, 2002.

[82] Wikipedia contributors. Woodbury matrix identity. `https://en.wikipedia.org/wiki/Woodbury_matrix_identity`, 2023.

[83] Tzon-Tzer Lu and Sheng-Hua Shiou. Inverses of 2 Œ 2 block matrices. *Computers Mathematics with Applications*, 43(1):119–129, 2002.

[84] Wikipedia contributors. Invertible matrix — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Invertible_matrix#Blockwise_inversion`, 2024.

[85] Hongzhi Wang, Qian Xiao, and Abhyuday Mandal. *LHD: Latin Hypercube Designs*, 2021. R package version 1.3.3.

[86] Rob Carnell. *lhs: Latin Hypercube Samples*, 2022. R package version 1.1.6.

[87] Wong Jeffrey. *pdist: Partitioned Distance Function*, 2022. R package version 1.2.1.

[88] Karline Soetaert. *plot3D: Plotting Multi-Dimensional Data*, 2024. R package version 1.4.1.

[89] Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec, and Pedro Despouy. *plotly: Create Interactive Web Graphics via plotly.js*, 2024. R package version 4.10.4.

[90] Phil Chalmers. *SimDesign: Structure for Organizing Monte Carlo Simulation Designs*, 2021. R package version 2.15.

[91] Simon Garnier. *viridisLite: Colorblind-Friendly Color Maps (Lite Version)*, 2023. R package version 0.4.2.

Appendix A

Supplementary Theoretical Material

A.1 Matrix Identities

Lemma 94. *For matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, assuming inverses exist,*

$$\boldsymbol{A}(\boldsymbol{BA}+\boldsymbol{I})^{-1} = (\boldsymbol{AB}+\boldsymbol{I})^{-1}\boldsymbol{A}. \tag{A.1}$$

Proof. Assuming necessary inverses exist, following proof in [41],

$$\begin{aligned} \boldsymbol{ABA}+\boldsymbol{A} &= (\boldsymbol{AB}+\boldsymbol{I})\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{BA}+\boldsymbol{I}) \\ &\implies \boldsymbol{A}(\boldsymbol{BA}+\boldsymbol{I})^{-1} = (\boldsymbol{AB}+\boldsymbol{I})^{-1}\boldsymbol{A}. \end{aligned}$$

■

Lemma 95. *For invertible matrices $\boldsymbol{A}$ and $\boldsymbol{C}$[1] and matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ of appropriate dimensions,*

$$\boldsymbol{AB}(\boldsymbol{DAB}+\boldsymbol{C})^{-1} = \left(\boldsymbol{BC}^{-1}\boldsymbol{D}+\boldsymbol{A}^{-1}\right)^{-1}\boldsymbol{BC}^{-1}. \tag{A.2}$$

Proof. Following proof in [41],

$$\begin{aligned} \boldsymbol{AB}(\boldsymbol{DAB}+\boldsymbol{C})^{-1} &= \boldsymbol{AB}\left[\boldsymbol{C}\left(\boldsymbol{C}^{-1}\boldsymbol{DAB}+I\right)\right]^{-1} \\ &= \boldsymbol{AB}\left(\boldsymbol{C}^{-1}\boldsymbol{DAB}+I\right)^{-1}\boldsymbol{C}^{-1} \\ &= \left(\boldsymbol{ABC}^{-1}\boldsymbol{D}+I\right)^{-1}\boldsymbol{ABC}^{-1} && \text{by (A.1)} \\ &= \left[\boldsymbol{A}\left(\boldsymbol{BC}^{-1}\boldsymbol{D}+\boldsymbol{A}^{-1}\right)\right]^{-1}\boldsymbol{ABC}^{-1} \\ &= \left(\boldsymbol{BC}^{-1}\boldsymbol{D}+\boldsymbol{A}^{-1}\right)^{-1}\boldsymbol{A}^{-1}\boldsymbol{ABC}^{-1} \\ &= \left(\boldsymbol{BC}^{-1}\boldsymbol{D}+\boldsymbol{A}^{-1}\right)^{-1}\boldsymbol{BC}^{-1}. \end{aligned}$$

■

Lemma 96. *For a square matrix $\boldsymbol{P}$,*

$$(\boldsymbol{I}+\boldsymbol{P})^{-1} = \boldsymbol{I}-(\boldsymbol{I}+\boldsymbol{P})^{-1}\boldsymbol{P}. \tag{A.3}$$

Proof. Following proof in [41],

$$\begin{aligned} (\boldsymbol{I}+\boldsymbol{P})^{-1} &= (\boldsymbol{I}+\boldsymbol{P})^{-1}(\boldsymbol{I}+\boldsymbol{P}-\boldsymbol{P}) \\ &= (\boldsymbol{I}+\boldsymbol{P})^{-1}(\boldsymbol{I}+\boldsymbol{P})-(\boldsymbol{I}+\boldsymbol{P})^{-1}\boldsymbol{P} \\ &= \boldsymbol{I}-(\boldsymbol{I}+\boldsymbol{P})^{-1}\boldsymbol{P}. \end{aligned}$$

■

[1] $\boldsymbol{A}$ and $\boldsymbol{C}$ do not need to be the same dimension.

Lemma 97 (Sherman-Morrison-Woodbury Identity [81, 82])**.** *For invertible matrices* $\boldsymbol{A}$ *and* $\boldsymbol{C}$ *and matrices* $\boldsymbol{B}$ *and* $\boldsymbol{D}$ *of appropriate dimensions,*

$$(\boldsymbol{A}+\boldsymbol{BCD})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{C}^{-1}+\boldsymbol{DA}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{DA}^{-1}. \tag{A.4}$$

Proof. Following proof in [41],

$$\begin{aligned}
(\boldsymbol{A}+\boldsymbol{BCD})^{-1} &= \left(\boldsymbol{A}\left[I+\boldsymbol{A}^{-1}\boldsymbol{BCD}\right]\right)^{-1} \\
&= \left(I+\boldsymbol{A}^{-1}\boldsymbol{BCD}\right)^{-1}\boldsymbol{A}^{-1} \\
&= \left[I-\left(I+\boldsymbol{A}^{-1}\boldsymbol{BCD}\right)^{-1}\boldsymbol{A}^{-1}\boldsymbol{BCD}\right]\boldsymbol{A}^{-1} && \text{by (A.3)} \\
&= \boldsymbol{A}^{-1} - \left(I+\boldsymbol{A}^{-1}\boldsymbol{BCD}\right)^{-1}\boldsymbol{A}^{-1}\boldsymbol{BCDA}^{-1} \\
&= \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(I+\boldsymbol{CDA}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{CDA}^{-1} && \text{by (A.1)} \\
&= \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left[\boldsymbol{C}^{-1}\left(I+\boldsymbol{CDA}^{-1}\boldsymbol{B}\right)\right]^{-1}\boldsymbol{DA}^{-1} \\
&= \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{C}^{-1}+\boldsymbol{DA}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{DA}^{-1}.
\end{aligned}$$

■

Corollary 98. *A special case of Lemma 97 is when* $\boldsymbol{B}=\boldsymbol{D}=\boldsymbol{I}$ *and we obtain*

$$(\boldsymbol{A}+\boldsymbol{C})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\left(\boldsymbol{C}^{-1}+\boldsymbol{A}^{-1}\right)^{-1}\boldsymbol{A}^{-1}. \tag{A.5}$$

Lemma 99 (Block matrix inverse formula [83, 84])**.**

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \left(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\right)^{-1} & -\left(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{BD}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\left(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\right)^{-1} & \boldsymbol{D}^{-1}+\boldsymbol{D}^{-1}\boldsymbol{C}\left(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{BD}^{-1} \end{bmatrix} \tag{A.6}$$

.

Proof. Ommited, see [83]. ■

A.2 Advanced Emulator

A.2.1 Advanced Emulator Prior Expectation and Covariance

The prior expectation for the emulator with a regression term [41] can be computed by,

$$\begin{aligned}
\mathbb{E}(f(\boldsymbol{x})) &= \mathbb{E}\left(\boldsymbol{g}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\beta}+u(\boldsymbol{x})\right) \\
&= \boldsymbol{g}(\boldsymbol{x})^{\mathsf{T}}\,\mathbb{E}(\boldsymbol{\beta})+0 \\
&= \boldsymbol{g}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\mu}_{\beta},
\end{aligned} \tag{A.7}$$

and similarly, the covariance can be calculated as follows,

$$\begin{aligned}
\mathrm{Cov}(f(\boldsymbol{x}),f(\boldsymbol{x}')) &= \mathrm{Cov}\left(\boldsymbol{g}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\beta}+u(\boldsymbol{x}),\ \boldsymbol{g}(\boldsymbol{x}')^{\mathsf{T}}\boldsymbol{\beta}+u(\boldsymbol{x}')\right) \\
&= \boldsymbol{g}(\boldsymbol{x})^{\mathsf{T}}\,\mathrm{Cov}(\boldsymbol{\beta},\boldsymbol{\beta})\boldsymbol{g}(\boldsymbol{x}')+\mathrm{Cov}(u(\boldsymbol{x}),u(\boldsymbol{x}')) \\
&= \boldsymbol{g}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\Sigma}_{\beta}\boldsymbol{g}(\boldsymbol{x}')+\sigma^2 r(\boldsymbol{x}-\boldsymbol{x}').
\end{aligned} \tag{A.8}$$

A.2.2 Advanced Emulator with Regression Surface

Proof of Lemma 61. This proof follows [41] using Theorem 11 and Theorem 15. We can explicitly calculate $\mathbb{E}_D(\boldsymbol{\beta})$,

$$\begin{aligned}
\mathbb{E}_D(\boldsymbol{\beta}) &= \mathbb{E}(\boldsymbol{\beta}) + \operatorname{Cov}(\boldsymbol{\beta}, \mathbf{f}_D) \operatorname{Var}(\mathbf{f}_D)^{-1}(\mathbf{f}_D - \mathbb{E}(\mathbf{f}_D)) \\
&= \boldsymbol{\mu}_\beta + \operatorname{Cov}(\boldsymbol{\beta}, \mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D) \operatorname{Var}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)^{-1}(\mathbf{f}_D - \mathbf{G}_D\,\mathbb{E}(\boldsymbol{\beta})) \\
&= \boldsymbol{\mu}_\beta + \operatorname{Var}(\boldsymbol{\beta})\mathbf{G}_D^\mathsf{T}(\mathbf{G}_D \operatorname{Var}(\boldsymbol{\beta})\mathbf{G}_D^\mathsf{T} + \operatorname{Var}(\mathbf{u}_D))^{-1}(\mathbf{f}_D - \mathbf{G}_D\boldsymbol{\mu}_\beta)
\end{aligned}$$

using that $\operatorname{Cov}(\boldsymbol{\beta}, \mathbf{u}_D) = \mathbf{0}$ (assumed a priori),

$$\begin{aligned}
&= \boldsymbol{\mu}_\beta + \boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T}(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}(\mathbf{f}_D - \mathbf{G}_D\boldsymbol{\mu}_\beta) \\
&= \boldsymbol{\mu}_\beta + (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\mathbf{G}_D\boldsymbol{\Omega}_D^{-1}(\mathbf{f}_D - \mathbf{G}_D\boldsymbol{\mu}_\beta) && \text{by (A.2)} \\
&= (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\left[(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})\boldsymbol{\mu}_\beta + \mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}(\mathbf{f}_D - \mathbf{G}_D\boldsymbol{\mu}_\beta)\right] \\
&= (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\left[\cancel{\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\boldsymbol{\mu}_\beta} + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta + \mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{f}_D - \cancel{\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\boldsymbol{\mu}_\beta}\right] \\
&= (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}(\boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta + \mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{f}_D)
\end{aligned}$$

as we have in (4.4). Similarly, we can explicitly calculate $\operatorname{Var}_D(\boldsymbol{\beta})$,

$$\begin{aligned}
\operatorname{Var}_D(\boldsymbol{\beta}) &= \operatorname{Var}(\boldsymbol{\beta}) - \operatorname{Cov}(\boldsymbol{\beta}, \mathbf{f}_D) \operatorname{Var}(\mathbf{f}_D)^{-1} \operatorname{Cov}(\mathbf{f}_D, \boldsymbol{\beta}) \\
&= \boldsymbol{\Sigma}_\beta - \operatorname{Cov}(\boldsymbol{\beta}, \mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D) \operatorname{Var}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)^{-1} \operatorname{Cov}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D, \boldsymbol{\beta}) \\
&= \boldsymbol{\Sigma}_\beta - \operatorname{Var}(\boldsymbol{\beta})\mathbf{G}_D^\mathsf{T}(\mathbf{G}_D \operatorname{Var}(\boldsymbol{\beta})\mathbf{G}_D^\mathsf{T} + \operatorname{Var}(\mathbf{u}_D))^{-1}\mathbf{G}_D \operatorname{Var}(\boldsymbol{\beta})
\end{aligned}$$

again using that $\operatorname{Cov}(\boldsymbol{\beta}, \mathbf{u}_D) = \mathbf{0}$,

$$\begin{aligned}
&= \boldsymbol{\Sigma}_\beta - \boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T}(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta \\
&= \boldsymbol{\Sigma}_\beta - \left(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1}\right)^{-1}\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta && \text{by (A.2)} \\
&= (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\left[(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})\boldsymbol{\Sigma}_\beta - \mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta\right] \\
&= (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\left[\cancel{\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta} + \boldsymbol{I} - \cancel{\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta}\right] \\
&= (\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}
\end{aligned}$$

as we have in (4.5). ■

Proof of Lemma 64. Following [41],

$$\begin{aligned}
\mathbb{E}_D(\mathbf{u}_*) &= \cancel{\mathbb{E}(\mathbf{u}_*)} + \operatorname{Cov}(\mathbf{u}_*, \mathbf{f}_D) \operatorname{Var}(\mathbf{f}_D)^{-1}(\mathbf{f}_D - \mathbb{E}(\mathbf{f}_D)) \\
&= \operatorname{Cov}(\mathbf{u}_*, \mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D) \operatorname{Var}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)^{-1}(\mathbf{f}_D - \mathbf{G}_D\boldsymbol{\mu}_\beta) \\
&= \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}(\mathbf{f}_D - \mathbf{G}_D\boldsymbol{\mu}_\beta) \\
&= \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\left[(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}\mathbf{f}_D - (\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta\boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta\right]
\end{aligned}$$

using (A.2), $(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}\mathbf{G}_D\boldsymbol{\Sigma}_\beta = \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}$

$$= \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\left[(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}\mathbf{f}_D - \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta\right]$$

using (A.4), $(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1} = \boldsymbol{\Omega}_D^{-1} - \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}$

$$\begin{aligned}
&= \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\Big[(\boldsymbol{\Omega}_D^{-1} - \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1})\mathbf{f}_D \\
&\qquad\qquad - \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta\Big] \\
&= \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}\left[\mathbf{f}_D - \mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{f}_D + \boldsymbol{\Sigma}_\beta^{-1}\boldsymbol{\mu}_\beta)\right] \\
&= \operatorname{Cov}(\mathbf{u}_*, \mathbf{u}_D)\boldsymbol{\Omega}_D^{-1}(\mathbf{f}_D - \mathbf{G}_D\,\mathbb{E}_D(\boldsymbol{\beta}))
\end{aligned}$$

as in (4.10), and as for the variance,

$$\begin{aligned}
\mathrm{Var}_D(\mathbf{u}_*) &= \mathrm{Var}(\mathbf{u}_*) - \mathrm{Cov}(\mathbf{u}_*, \mathbf{f}_D)\,\mathrm{Var}(\mathbf{f}_D)^{-1}\,\mathrm{Cov}(\mathbf{f}_D, \mathbf{u}_*) \\
&= \boldsymbol{\Omega}_* - \mathrm{Cov}(\mathbf{u}_*, \mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)\,\mathrm{Var}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)^{-1}\,\mathrm{Cov}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D, \mathbf{u}_*) \\
&= \boldsymbol{\Omega}_* - \mathrm{Cov}(\mathbf{u}_*, \mathbf{u}_D)\,\mathrm{Var}(\mathbf{G}_D\boldsymbol{\beta} + \mathbf{u}_D)^{-1}\,\mathrm{Cov}(\mathbf{u}_D, \mathbf{u}_*) \\
&= \boldsymbol{\Omega}_* - \mathrm{Cov}(\mathbf{u}_*, \mathbf{u}_D)(\mathbf{G}_D\boldsymbol{\Sigma}_\beta\mathbf{G}_D^\mathsf{T} + \boldsymbol{\Omega}_D)^{-1}\,\mathrm{Cov}(\mathbf{u}_D, \mathbf{u}_*) \\
&= \boldsymbol{\Omega}_* - \mathrm{Cov}(\mathbf{u}_*, \mathbf{u}_D)\left(\boldsymbol{\Omega}_D^{-1} + \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D(\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\mathbf{G}_D + \boldsymbol{\Sigma}_\beta^{-1})^{-1}\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\right)\mathrm{Cov}(\mathbf{u}_D, \mathbf{u}_*) \quad \text{by (A.4)} \\
&= \boldsymbol{\Omega}_* - \mathrm{Cov}(\mathbf{u}_*, \mathbf{u}_D)\left(\boldsymbol{\Omega}_D^{-1} + \boldsymbol{\Omega}_D^{-1}\mathbf{G}_D\,\mathrm{Var}_D(\boldsymbol{\beta})\mathbf{G}_D^\mathsf{T}\boldsymbol{\Omega}_D^{-1}\right)\mathrm{Cov}(\mathbf{u}_D, \mathbf{u}_*)
\end{aligned}$$

as we have in (4.11). ■

A.3 Experimental Design

Algorithm 1 Heuristic Maximin LHD

$\boldsymbol{X}_{\text{best}} \leftarrow \boldsymbol{X}_0$ (Initial random LHD)
for $k = 1$ to 1000 **do**
  Find pairs of closest runs in $\boldsymbol{X}_{\text{best}}$ and randomly pick one run $\boldsymbol{x}_{\text{close}}$
  Randomly pick another point $\boldsymbol{x}$ from $\boldsymbol{X}_{\text{best}}$
  Switch indices of $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{close}}$ the two points to obtain a new design $\boldsymbol{X}_{\text{new}}$
  Compute maximin distance d_{new} of $\boldsymbol{X}_{\text{new}}$
  if $d_{\text{new}} > d_{\text{best}}$ **then**
    $\boldsymbol{X}_{\text{best}} \leftarrow \boldsymbol{X}_{\text{new}}$
    $d_{\text{best}} \leftarrow d_{\text{new}}$
  end if
end for
return $\boldsymbol{X}_{\text{best}}$

Appendix B

Supplementary Material for Examples

B.1 Introduction to Emulation

B.1.1 Simple BL Emulator

Example 25 (continuing from p. 11)**.** *The explicit definition of* $f(x_1, x_2)$ *is:*

$$f(x_1, x_2) = A_1 + \frac{5}{3}A_2 + \frac{1}{3}B - \frac{1}{2}, \tag{B.1}$$

with

$$\begin{aligned} A_1 &= \exp\left(-10\left((x_1 - 0.4)^2 + (x_2 - 0.7)^2\right)\right) \\ A_2 &= \exp\left(-20\left((x_1 - 0.9)^2 + (x_2 - 0.5)^2\right)\right) \\ B &= \cos\left(6\left(x_1^2 - x_2^2\right)\right). \end{aligned}$$

B.2 Advanced Emulation and Applications

B.2.1 Advanced 1D Emulator

Example 67 (continuing from p. 26)**.** *The explicit definition of* $f(x)$ *is:*

$$f(x) = \frac{1}{2}x^2 + \frac{1+x}{5}\sin(3\pi x). \tag{B.2}$$

B.2.2 Grid Design Issues

Example 100 (Adapted from [41])**.** *Consider the function in Figure B.1a which has clear periodic behaviours along the x-axis. The explicit definition is*

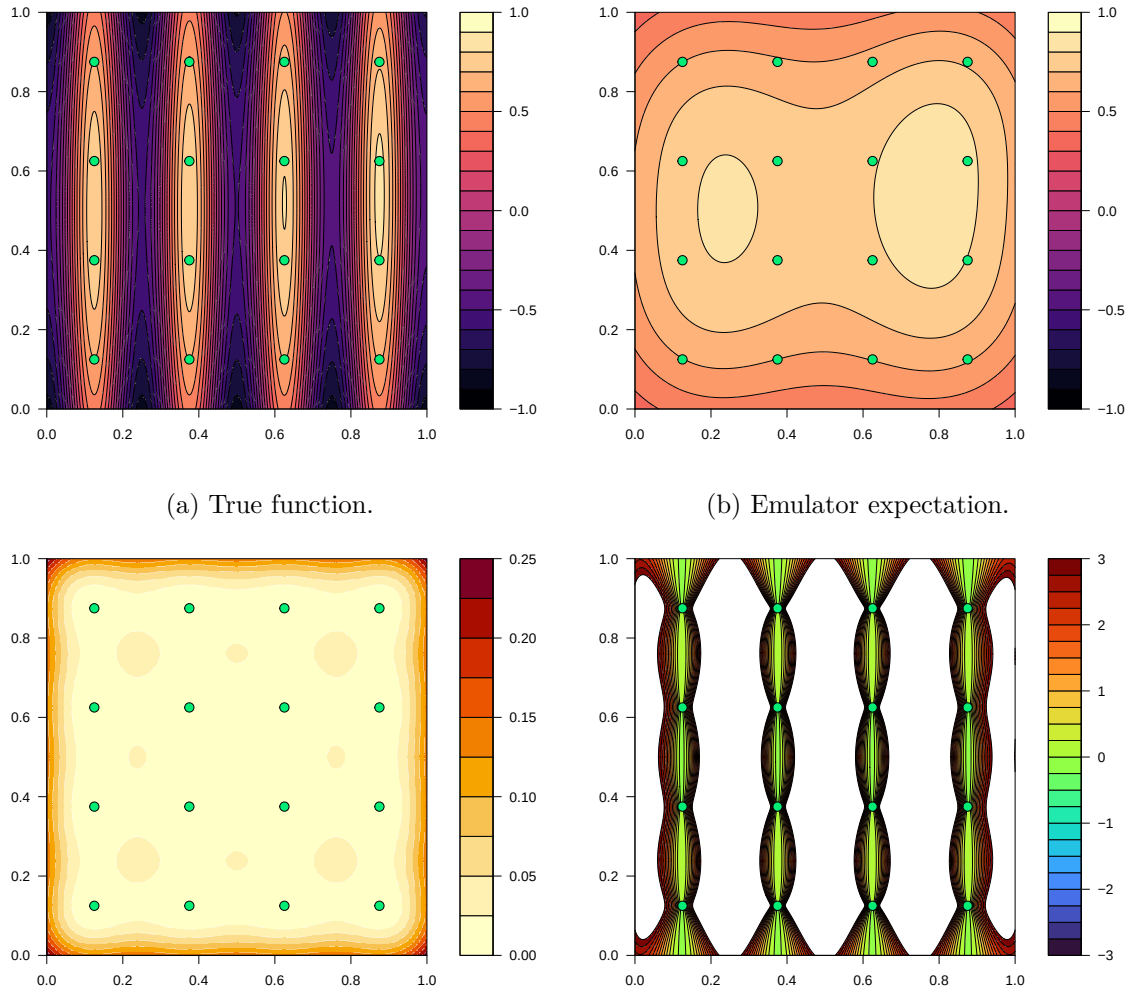
$$f(x, y) = 1.3\left(\sin(4\pi x)^2 - 0.4 + \frac{x^2 y}{10} - (y - 0.5)^2\right). \tag{B.3}$$

We also choose a 4×4 *grid design which coincides with the periodicity. Building a simple emulator for this function, we see the expectation completely misses all of the interesting behaviour in the x-axis and seriously fails diagnostics.*

B.2.3 1D Implausibility

Example 77 (continuing from p. 31)**.** *The explicit definition of* $f(x)$ *is:*

$$f(x) = \sin(2\pi x + 1) + 0.6\cos(5\pi x). \tag{B.4}$$



(a) True function.

(b) Emulator expectation.

(c) Emulator variance.

(d) Emulator diagnostics.

Figure B.1: The true function in Example 100 along with the emulator expectation, variance and diagnostics. Design points are in green and the white regions of the diagnostics plot are where the emulator has failed diagnostics.

B.2.4 2D History Match

Example 79 (continuing from p. 31)**.** *The explicit definition of* $f(x, y)$ *is:*

$$f(x,y) = \tanh\left[(\sin(3\pi x) + \cos(3\pi xy))\exp\left\{-3\left(\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2\right)\right\}\right]. \tag{B.5}$$

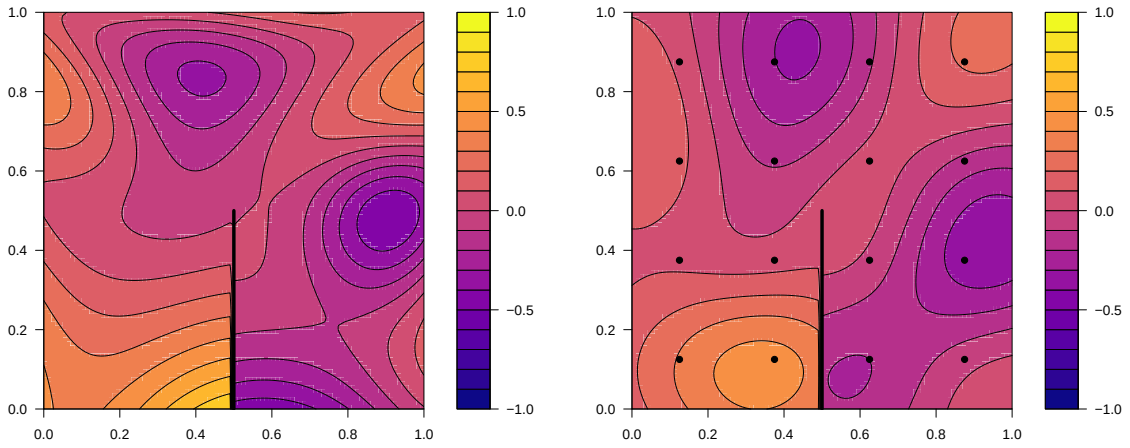
B.3 Emulation with Partial Known Discontinuities

B.3.1 Simple Discontinuity

Example 81 (continuing from p. 33)**.** *The explicit definition of* $f(x, y)$ *is:*

$$f(x,y) = \frac{7}{20} - \frac{5}{3}\mathbb{1}_{(y<0.5)}(y-0.5)^2(1-2\mathbb{1}_{(x<0.5)})\left(\sin(3x)+x\right) - \frac{1}{6}\cos\left(8\left(\left((x-0.5)^2-y^2\right)\right)\right.$$

$$\left. - \frac{2}{3}\left(\exp\left\{-10\left((x-0.4)^2+(y-0.7)^2\right)\right\} + \exp\left\{-10\left((x-0.9)^2+(y-0.5)^2\right)\right\}\right). \tag{B.6}$$



(a) Contour plot of $f(x, y)$.

(b) Expectation contour plot.

Figure B.2: True function (left) and TENSE emulator expectation (right) from Example 81.

B.3.2 Multiple Curved Discontinuities

Example 90 (continuing from p. 41)**.** *Explicitly, the embedding surface is given by,*

$$v(x,y) = -\mathbb{1}_{\{y<0.25\}} \cdot \mathbb{1}_{\{y<A(x)\}} \left(\frac{1}{2}\mathbb{1}_{\{x>x_0\}}(x-x_0)^2 + (y-0.25)^2 \right) + \mathbb{1}_{\{y>-0.25\}} \cdot \mathbb{1}_{\{y>-A(-x)\}} \left(\frac{1}{2}\mathbb{1}_{\{x<-x_0\}}(x+x_0)^2 + (y+0.25)^2 \right) \quad \text{(B.7)}$$

where $A(x) = \frac{1}{5}\exp((x-0.8)^2) - 0.85$ *and* $x_0 \approx 0.50566$ *is given by* $A^{-1}(x)$*. The true function is given by,*

$$f(x,y) = \left(B_1(x,y) - \frac{4}{5} + B_2(x,y) \right) B_3(x,y), \quad \text{(B.8)}$$

with

$$B_1(x,y) = 2\tanh\left(\left| v(x,y)\left(y\sin^2(xy-4)\right) + 5\left(\sqrt{\cos(x^2-y^2)} - 1.1 \right) + 0.8 \right| \right) \quad \text{(B.9)}$$

$$B_2(x,y) = \exp\left(-5\left((x+0.5)^2 + (y+0.3)^2\right)\right)\left(2\mathbb{1}_{\{y<A(x)\}} - 1\right)\mathbb{1}_{\{y<0.25\}}(y-0.25) \quad \text{(B.10)}$$

$$B_3(x,y) = \tanh\left(10\exp\left(-3(x^4+y^4)\right)\right). \quad \text{(B.11)}$$

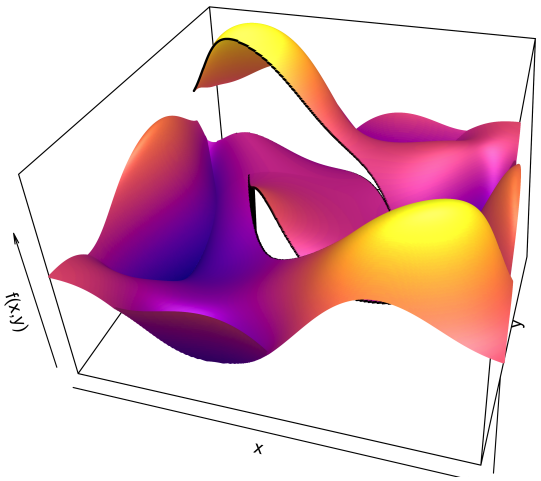


Figure B.3: 3D plot of $f(x, y)$ from Example 90.

B.3.3 Applying TENSE in 3D

The embedding surface for section 5.4 is constructed out of a sigmoidal function whose derivative has a compact domain. This ensures the embedding surface has the properties we want. The sigmoidal function used to construct the embedding surface is given as,

$$s(d) = 2d^2 \mathbb{1}_{d\in[0,0.5)} + (1 - 2(1-d)^2)\mathbb{1}_{d\in[0.5,1]} + \mathbb{1}_{d\geq 1}, \tag{B.12}$$

and is shown below in Figure B.4.

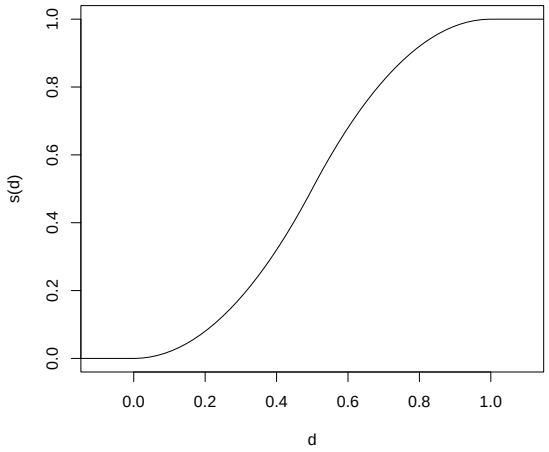


Figure B.4: Sigmoidal function from (B.12).

The explicit form of the embedding surface is then defined as,

$$v(x,y,z) = \left(1 - s\left(3\sqrt{(x-0.5)^2 + (0.5z)^2}\right)\right)(1 - 2\mathbb{1}_{y>0.5}). \tag{B.13}$$

This can be interpreted as the smoothed and warped (to account for the elliptical shape of discontinuity) distance in the xz-plane from the point $x = 0.5$, $z = 0$. The sigmoidal function ensures the embedding surface is 0 away from the discontinuity (in the xz-plane). Finally, the sign of the embedding surface is defined by whether $y > 0.5$ or not, this induces the discontinuity at the desired locations.

The explicit definition of the function in section 5.4 is given in terms of the embedding surface,

$$f(x,y,z) = R(x,y,z) + \frac{1}{2}\left(A(x,y,z) + B(x,y,z) + C(x,y,z) + v(x,y,z)\right), \tag{B.14}$$

where $v(x,y,z)$ is the embedding surface we have already defined in (B.13) and,

$$R(x,y,z) = \left(3(x-0.4)^2 + 2y^2 + 5(x-0.3)(z-0.3) + 0.65\right) \tag{B.15}$$
$$A(x,y,z) = (0.6G_1(x,z) + G_2(x,z)) \cdot (v(x,y,z) + 1) \tag{B.16}$$
$$G_1(x,z) = \exp\left(-50((x-0.5)^2 + (z-0.6)^2)\right) \tag{B.17}$$
$$G_2(x,z) = \exp\left(-30((x-0.8)^2 + (z-0.4)^2)\right) \tag{B.18}$$
$$B(x,y,z) = \cos(12\sqrt{2(x-0.4)^2 + (y-0.3)^2 + (z-0.2)^2}) \cdot(\sin(3x) + 0.2) \cdot (\cos(6y) + 0.2) \cdot (\cos(8z) + 0.2) \tag{B.19}$$
$$C(x,y,z) = \tanh(1 + v(x,y,z)) \cdot (1 - z) \tag{B.20}$$

Appendix C

Code for Arbitrary Dimension TENSE

Below are `R` code snippets implementing the TENSE framework for arbitrary dimension. For full details, advanced variants and examples, see the GitHub repository (`github.com/jreaso/mmath-thesis`) [16].

The code provided in the supplementary repository uses the R packages `LHD` [85], `lhs` [86], `pdist` [87], `plot3D` [88], `plotly` [89], `SimDesign` [90], and `viridisLite` [91].

`TENSE.R` (Vectorised TENSE covariance functions)

```r
# Stationary CF (Squared Exponential)
k_S <- function(d) exp(-d^2)

# Sigma_V matrix as a function of embedding surface derivatives
Sigma_V <- function(vdX, theta=1, lambda2=1) {
  N <- ncol(vdX)
  theta2 <- theta^2

  Sigma_V_func <- function(vd) {
    r2 <- sum(vd^2)
    if (r2 == 0){ # zero derivative so no scaling
      Sigma_V <- diag(theta2, N+1)
      Sigma_V[N+1, N+1] <- lambda2
    } else {
      w_grad <- c(vd, r2)
      w_norm <- c(-vd, 1)
      Sigma_V <- ((theta2/r2) * outer(w_grad, w_grad))
                + (theta2 * rbind(cbind(diag(N) - outer(vd, vd)/r2, 0), 0))
                + ((lambda2/(1+r2)) * outer(w_norm, w_norm))
    }
    return(Sigma_V)
  }
  return(lapply(1:nrow(vdX), function(i) Sigma_V_func(vdX[i, ])))
}

# Quadratic form for NS CF
Q <- function(x, y, Mx, My) {
  (x-y) %*% solve((Mx + My)/2) %*% (x-y)
}
```

```
30
31 ### Non-Stationary TENSE Covariance Function
32 # Use to compute variance (not covariance) matrices
33 k_NS <- function(VX, vdX, k_S=k_S, theta=1, lambda2=1, sigma=1) {
34   n <- nrow(VX)
35   N <- ncol(VX) - 1
36   Sigma_V_list <- Sigma_V(vdX, theta=theta, lambda2=lambda2)
37   k_NS_func <- function(i, j) {
38     if (i==j){
39       return(1)
40     } else if (j < i) {
41       return(NA) # do not compute lower triangular entries
42     } else {
43       xi <- VX[i,]; xj <- VX[j,]
44       Sigma_Vi <- Sigma_V_list[[i]]; Sigma_Vj <- Sigma_V_list[[j]]
45       Qij <- Q(x=xi, y=xj, Mx=Sigma_Vi, My=Sigma_Vj)
46       return(2^((N+1)/2) * det(Sigma_Vi)^(1/4) * det(Sigma_Vj)^(1/4)
47              / det(Sigma_Vi + Sigma_Vj)^(1/2) * k_S(d=sqrt(Qij)))
48     }
49   }
50   Cmat <- outer(1:n, 1:n, Vectorize(k_NS_func))
51   Cmat[lower.tri(Cmat)] <- t(Cmat)[lower.tri(Cmat)] # copy across diagonal
52   return(sigma^2 * Cmat)
53 }
54
55 # use for computing covariance matrices
56 k_NS2 <- function(VX1, VX2, vdX1, vdX2, k_S=k_S, theta=1, lambda2=1, sigma=1) {
57   n1 <- nrow(VX1); n2 <- nrow(VX2)
58   N <- ncol(VX1) - 1
59   Sigma_V_list1 <- Sigma_V(vdX1, theta=theta, lambda2=lambda2)
60   Sigma_V_list2 <- Sigma_V(vdX2, theta=theta, lambda2=lambda2)
61
62   k_NS_func <- function(i, j) {
63     x1 <- VX1[i,]; x2 <- VX2[j,]
64     Sigma_V1 <- Sigma_V_list1[[i]]; Sigma_V2 <- Sigma_V_list2[[j]]
65     Qij <- Q(x1, x2, Sigma_V1, Sigma_V2)
66     return(2^((N+1)/2) * det(Sigma_V1)^(1/4) * det(Sigma_V2)^(1/4)
67            / det(Sigma_V1 + Sigma_V2)^(1/2) * k_S(d=sqrt(Qij)))
68   }
69   Cmat <- outer(1:n1, 1:n2, Vectorize(k_NS_func))
70   return(sigma^2 * Cmat)
71 }
```

simple_NS_emulator2.R (Simple emulator using TENSE covariance function)

```
simple_NS_emulator2 <- function(VxD, D, VxP, vdxD, vdxP, k_S=k_S, mu=0, theta=1,
                                lambda2=1, sigma=1, just_var=T){
  n <- length(D); nP <- nrow(VxP)

  # Compute covariance matrices, Var[xD]
  Var_D <- k_NS(VX=VxD, vdX=vdxD, k_S=k_S, theta=theta, lambda2=lambda2, sigma=sigma)
  # Cov[xD, xP]
  Cov_DB <- k_NS2(VX1=VxD, VX2=VxP, vdX1=vdxD, vdX2=vdxP, k_S=k_S, theta=theta,
                  lambda2=lambda2, sigma=sigma)

  Var_D_inv <- solve(Var_D)
  Cov_BD_Var_D_inv <- t(Cov_DB) %*% Var_D_inv
  E_D <- rep(mu, n)
  E_B <- rep(mu, nP)

  # Perform BL expectation update
  ED_B <- E_B + Cov_BD_Var_D_inv %*% (D - E_D)

  if (just_var) { # compute diagonal only
    VarD_B <- rep(sigma^2, nP) - diag(Cov_BD_Var_D_inv %*% Cov_DB)
  } else {
    # Var[xP]
    Var_B <- k_NS(VX=VxP, vdX=vdxP, k_S=k_S, theta=theta, lambda2=lambda2, sigma=sigma)
    VarD_B <- Var_B - Cov_BD_Var_D_inv %*% t(Cov_BD)
  }
  return(list("ED_fP"=ED_B, "VarD_fP"=VarD_B))
}
```