A Numerical Study of the Effects of the Parameterization of the Gaussian Process

by

Jodie Lynn Foster

B.Sc., The University of British Columbia, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The College of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

July 2017

The undersigned certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis entitled: A Numerical Study of the Effects of the Parameterization of the Gaussian Process submitted by Jodie Lynn Foster in partial fulfilment of the requirements of the degree of Master of Science

Jason Loeppky, Irving K. Barber School of Arts and Sciences

Supervisor, Professor

John Braun, Irving K. Barber School of Arts and Sciences

Supervisory Committee Member, Professor

Paramjit Gill, Irving K. Barber School of Arts and Sciences

Supervisory Committee Member, Professor

Julian Cheng, School of Engineering

University Examiner, Professor

July 28, 2017

(Date Submitted to Grad Studies)

Abstract

Mathematical models implemented as computer code are gaining widespread use across the sciences and engineering. In some cases these model are even replacing physical experiments. The computational complexity of the models typically limits the number of runs that can be performed. In such cases, the Gaussian process is used to emulate the true computer model so that experiments are performed on the emulator. The applicability of the emulator is closely related to the quality of the fitted Gaussian process model. A key step in fitting the Gaussian process is estimating the unknown correlation parameters using a numerical optimizer. It is well known that maximum likelihood estimation is invariant to the parameterization of the model. When the mapping from one parametrization to another is injective this leads to an equivalence of the likelihood and in a Bayesian context an equivalence of the posterior for carefully chosen prior distributions. In the context of a Gaussian process model, we show that the parameterization of the model is in fact critical to achieving meaningful results and ensuring convergence of the optimizer. This thesis is aimed at providing practical advice on the best parameterization for the Gaussian process model. The approach presented here implements a simulation study on a wide range of test problems and a large number of parameterizations. We illustrate that the parameterization can have a huge effect on the fitted model and show that many of the commonly used parameterizations are in fact sub-optimal for fitting the Gaussian process.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First and foremost I would like to thank to my supervisor, Dr. Jason Loeppky. Your inspiration, guidance, encouragement, and support have been invaluable for my success during this project.

I would also like to thank my supervisory committee: Dr. Paramjit Gill and Dr. John Braun. I sincerely appreciate your support, and thank you for dedicating your time to my work.

My gratitude goes to Dr. Patricia Lasserre for her constant support and suggestions, as well as to Dr. Shirin Golchi for her invaluable insight and help in the work presented here.

I wish to say thank you to the many students and friends I have had the pleasure of knowing or working with during my studies. My lab mates especially, provided me with encouragement, kind words, and coffee breaks to ensure that I would be able to power through debugging. Words cannot express how thankful I am to have had you by my side during this experience.

Finally, a special thanks to Spencer Hunt, Trent Jensen, and Rachel Stone who were the first to encourage me to peruse graduate studies; without you, this thesis would never have been started. For this, I would like to say “Thank you.”

Dedication

For my friends and especially my family; thank you for your unconditional love, encouragement, and support though my, at times tumultuous, journey.

Chapter 1

Introduction

Mathematical models are used across the statistical, mathematical, and engineering sciences. Mathematical models are implemented as computer code when physical experiments are costly or impossible to implement, replacing the physical experiments. The science being modeled by the mathematical code is often complex, requires a large number of variables and they can be slow to run. Statistical models, such as a Gaussian process can be used to approximate the code [5, 20–22] when the code is expensive to run.

Successfully implementing a Gaussian process surrogate model involves three major steps: choosing a design, specifying the form of the Gaussian process and then estimating the parameters in the Gaussian process. To choose a design that is appropriate for fitting a Gaussian process involves selecting an appropriate sample size as well as the design itself.

The exact choice of the form of the Gaussian process depends on the choice of the regression function as well as the choice of the correlation function. This thesis focuses on different selections for the correlation functions and the resultant effect on the fittings of the Gaussian process. Figure 1.1 shows error values from five separate runs of fitting the Gaussian process model (denoted as five separate lines) for eleven different Gaussian process correlation functions. In the figure, the correlation functions used for fitting the Gaussian process produced different final results (shown by the different error values on the y-axis. These different results suggest a problem of invariance; the different correlation functions should produce the same results, however, as shown in Figure 1.1, this is not the case. [3] consider efficiently numerically optimizing the Gaussian process. They implement several algorithms for choosing the starting values and optimize the GP using one parametrization and illustrate this on a few examples. In this thesis we greatly extend this work to consider several parameterizations and a much larger class of test functions.

This thesis focuses on determining which parameterization of the correlation function performs the best over a set of test functions and can be denoted as the 'go-to' parameterization to fit a Gaussian process. The remainder of the thesis is organized as follows, in Chapter 2 we formally

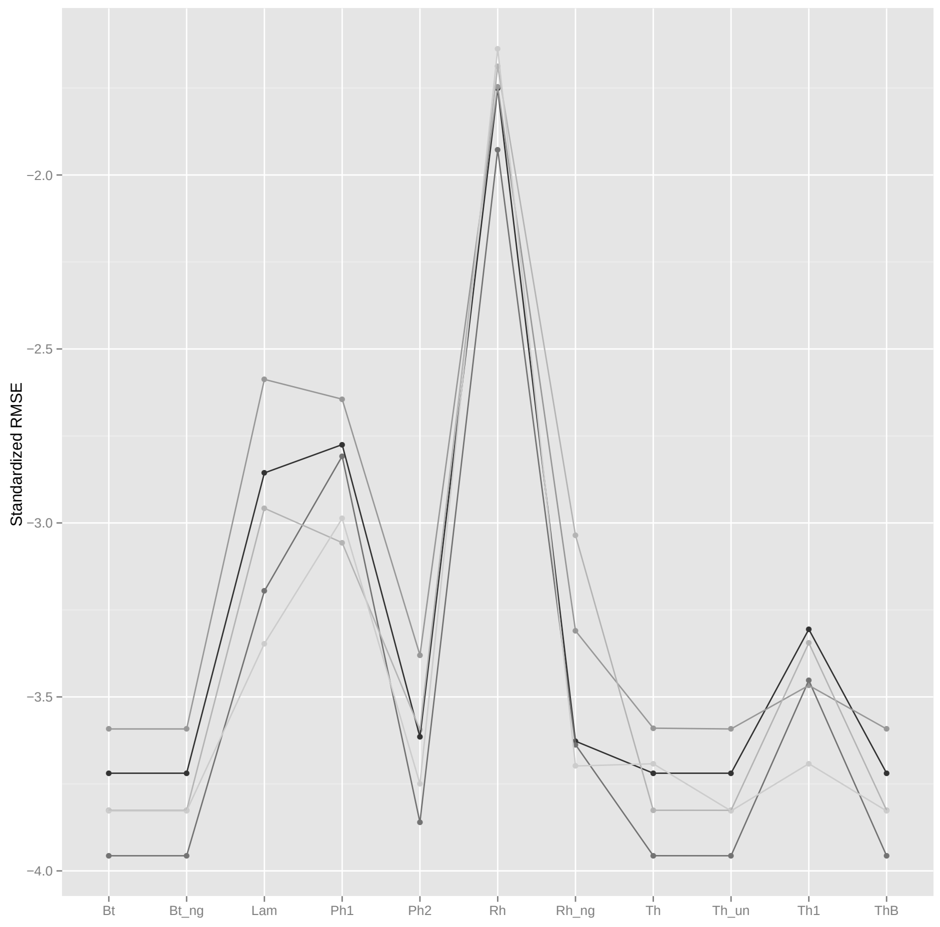

Figure 1.1: Root mean squared errors for 5 separate Gaussian process fittings for 11 different methods.

introduce the Gaussian process and outline the method of fitting a Gaussian process model. In Chapter 3 we delve into the three steps outlined above, choosing a design, discussing various parameterization options, and estimating the parameter values, while hinting at the problem of invariance. Chapter 4 introduces an example test function (the Borehole function) and the testing strategies used for the full simulation study. Chapter 5 performs the full test simulations and discusses the analysis and results. Finally, we conclude in Chapter 6 with recommendations.

Chapter 2

The Gaussian Process

2.1 Computer Models

Computer models are mathematical representations of physical experimentations that are implemented using computer code. These techniques are used commonly throughout the physical sciences when physical experimentation is expensive or time-consuming. Observations are made by running a computer model at a set of input points. This is typically referred to as a computer experiment. The computer code is deterministic - producing identical output for different runs of a given input - though the output is unknown before running the model. Consider a black-box function $\eta(\boldsymbol{x})$ with input $\boldsymbol{x} = (x_1, \ldots, x_d)$, where (for simplicity) $\boldsymbol{x} \in [0,1]^d$. The black-box function, $\eta(\boldsymbol{x})$, is known in the way that we know the code takes an input $\boldsymbol{x}$ and produces an output $y(\boldsymbol{x})$, however, the complexity of the code essentially makes it an unknown function that requires modelling. Thus, we treat the function $\eta(\boldsymbol{x})$ as unknown. From a Bayesian standpoint, the uncertainty of the output can be expressed using a stochastic process which is known as a statistical emulator. We can use a Gaussian process (GP) to represent the unknown function [15]. After building a GP emulator using runs of the computer code, analyses can be made without making more computer code runs. One such analysis is to predict output for a new set of locations not run.

2.2 The Gaussian Process

As stated above, we can use a Gaussian Process to represent the unknown function $\eta(\boldsymbol{x})$. The Gaussian process can be formally defined as a collection of random variables, any finite number of which have a joint Gaussian Distribution [17]. Given two vectors of data $\boldsymbol{x}$ and $\boldsymbol{x}^*$, the mean function $m(\boldsymbol{x})$ and the covariance function $V(\boldsymbol{x}, \boldsymbol{x}^*)$ are defined as

$$m(\boldsymbol{x}) = \mathbf{E}[f(\boldsymbol{x})], \tag{2.1}$$

and

$$V(\boldsymbol{x}, \boldsymbol{x}^*) = \mathbf{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}^*) - m(\boldsymbol{x}^*))] \tag{2.2}$$

respectively.

2.3 Gaussian Process Emulation

In this section, we will discuss the idea of the Gaussian process emulator. Given the unknown function $\eta(\boldsymbol{x})$ with input $\boldsymbol{x} = (x_1/ldots, x_d)$ where $\boldsymbol{x} \in [0,1]^d$ and corresponding output $\boldsymbol{y} = (y(x_1), \ldots, y(x_d))^T$. The uncertainty of the function $\eta(\boldsymbol{x})$ is described as a Gaussian process with mean function $m(\boldsymbol{x})$ and covariance function $V(\boldsymbol{x}, \boldsymbol{x}^*)$. Formally, if $\eta(\boldsymbol{x})$ has a Gaussian process distribution, then for every $n = 1, 2, \ldots$, the joint distribution of $\eta(\boldsymbol{x}^{(1)})$, $\ldots$, $\eta(\boldsymbol{x}^{(n)})$ is multivariate normal for all $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$. The choice of $m(\boldsymbol{x})$ can be any function of $\boldsymbol{x}$, but the choice of $V(\boldsymbol{x}, \boldsymbol{x}^*)$ must produce a covariance matrix with elements $\{\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})\}$ that satisfies the property of being nonnegative definite. Prior information of $\eta(\boldsymbol{x})$ is represented by a Gaussian process with mean and covariance functions, $m(\boldsymbol{x})$ and $V(y(\boldsymbol{x}), y(\boldsymbol{x}^*))$ respectively. The distribution of $\eta(\boldsymbol{x})$ is

$$\eta(\boldsymbol{x})|\beta, \sigma^2, \theta \sim GP(m(\boldsymbol{x}), V(y(\boldsymbol{x}), y(\boldsymbol{x}^*))), \tag{2.3}$$

where the mean function $m(\boldsymbol{x})$ is given by

$$m(\boldsymbol{x}) = \boldsymbol{F}^T(\boldsymbol{x})\boldsymbol{\beta}, \tag{2.4}$$

where $\boldsymbol{F}^T(\boldsymbol{x}) = (\boldsymbol{F}_1(\boldsymbol{x}), \ldots, \boldsymbol{F}_p(\boldsymbol{x}))^T$ is a set of known basis functions representing the mean and $\boldsymbol{\beta}$ is a set of unknown regression coefficients. Considering that $\eta(\boldsymbol{x}^{(i)})$ is unknown, it is feasible to assume that $\boldsymbol{F}(\boldsymbol{x}) = \mathbf{1}$ with $\mathbf{1}$ being a vector of ones as it allows the covariance function to model all of the signal [4]. The covariance function $V(y(\boldsymbol{x}), y(\boldsymbol{x}^*))$ is given by

$$V(y(\boldsymbol{x}), y(\boldsymbol{x}^*)) = \sigma^2 R\left(\boldsymbol{x}, \boldsymbol{x}^* | \boldsymbol{\theta}\right), \tag{2.5}$$

where σ^2 is an unknown scale parameter and R is a known correlation function with a vector of unknown correlation parameters $\boldsymbol{\theta}$. The choice of $R(\boldsymbol{x}, \boldsymbol{x}^* | \boldsymbol{\theta})$ should ensure that the covariance matrix is nonnegative definite for any set of input points. The choice for the exact form of R will be discussed in Section 2.5.1 and Section 3.2. At this point, we will consider the general function R. Suppose that we have a set of n inputs $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$, and the corresponding outputs $\boldsymbol{y} = (\eta(\boldsymbol{x}^{(1)}), \ldots, \eta(\boldsymbol{x}^{(n)}))^T$. These are denoted as the training data. According to Equation 2.3, the distribution of the outputs $\boldsymbol{y}$ is multivariate normal,

$$\boldsymbol{y}|\beta, \sigma^2, \theta \sim N(\boldsymbol{F}^T\boldsymbol{\beta}, \sigma^2\boldsymbol{R}), \tag{2.6}$$

where $\boldsymbol{R}$ is the correlation matrix comprised of elements

$$R_{ij} = R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}, |\boldsymbol{\theta}). \tag{2.7}$$

Using multivariate normal distribution conditioning techniques [7], it can be shown that

$$\eta(\boldsymbol{x})|\beta, \sigma^2, \theta, \boldsymbol{y} \sim \mathrm{GP}(m^*(\boldsymbol{x}), V^*(y(\boldsymbol{x}), y(\boldsymbol{x}^*))), \tag{2.8}$$

where

$$m^*(\boldsymbol{x}) = \boldsymbol{F}^T\boldsymbol{\beta} + \left(\boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})\right) \tag{2.9}$$

and

$$V^*(\boldsymbol{x}, \boldsymbol{x}^*) = \sigma^2\left(R(\boldsymbol{x}, \boldsymbol{x}^*|\boldsymbol{\theta}) + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}^*)\right) \tag{2.10}$$

where $\boldsymbol{r}(\boldsymbol{x}) = \left(R(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \ldots, R(\boldsymbol{x}, \boldsymbol{x}^{(n)})\right)^T$. Using a weak prior for (β, σ^2), $p(\beta, \sigma^2) \propto \sigma^{-2}$, and combining with Equation 2.6 using Bayes' Theorem, the posterior for (β, σ^2) is a normal inverse-gamma distribution,

$$\beta|\boldsymbol{y}, \sigma^2, \boldsymbol{\theta} \sim N(\hat{\beta}, \sigma^2(\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}), \tag{2.11}$$

where

$$\hat{\beta} = (\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{y}, \tag{2.12}$$

and

$$\sigma^2|\boldsymbol{y}, \boldsymbol{\theta} \propto \mathrm{InvGam}\left(\frac{n-q}{2}, \frac{(n-q-2)\hat{\sigma}^2}{2}\right), \tag{2.13}$$

where

$$\hat{\sigma}^2 = \frac{\boldsymbol{y}^T(\boldsymbol{R}^{-1} - \boldsymbol{R}^{-1}\boldsymbol{F}(\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1})\boldsymbol{y}}{n-q-2}. \tag{2.14}$$

Again, following [7], we can then integrate β our of the product of Equation 2.8 and Equation 2.11 to achieve

$$\eta(\boldsymbol{x})|\boldsymbol{y}, \sigma^2 \propto GP(m'(\boldsymbol{x}), V'(\boldsymbol{x}, \boldsymbol{x}^*)), \tag{2.15}$$

where

$$m'(\boldsymbol{x}) = \boldsymbol{F}^T(\boldsymbol{x})\hat{\beta} + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\hat{\beta}), \tag{2.16}$$

and

$$\begin{aligned} V'(\boldsymbol{x},\boldsymbol{x}) =& \sigma^2 \left(R(\boldsymbol{x},\boldsymbol{x}^*|\boldsymbol{\theta}) - \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}^*) \right. \\ & + (\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}\boldsymbol{F})(\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}(\boldsymbol{F}^{*T} - \boldsymbol{r}^T(\boldsymbol{x}^*)\boldsymbol{R}^{-1}\boldsymbol{F})^T. \end{aligned} \tag{2.17}$$

We can now integrate σ^2 out of the product of Equation 2.13 and Equation 2.15 to obtain

$$\eta(\boldsymbol{x})|\boldsymbol{y},\boldsymbol{\theta} \propto \text{StudentProcess}(n-q, m'(\boldsymbol{x}), V''(\boldsymbol{x},\boldsymbol{x}^*)), \tag{2.18}$$

where

$$V''(\boldsymbol{x},\boldsymbol{x}^*) = \frac{\hat{\sigma}^2}{\sigma^2} V'(\boldsymbol{x},\boldsymbol{x}^*). \tag{2.19}$$

2.4 Estimating Theta

Following [5], we use an empirical Bayesian approach to find values for θ by maximizing the likelihood. The likelihood for the GP is

$$\begin{aligned} L(\boldsymbol{\beta},\sigma^2,\boldsymbol{\theta}|\boldsymbol{y}) =& (2\pi)^{-n/2}(\sigma^2)^{-n/2}|\boldsymbol{R}|^{-1/2} \\ & \cdot \exp\left\{ -\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{F}\boldsymbol{\beta})^T\boldsymbol{R}^{-1/2}(\boldsymbol{y}-\boldsymbol{F}\boldsymbol{\beta}) \right\}. \end{aligned}$$

Taking the natural logarithm yields the log-Likelihood function [28], which is,

$$\begin{aligned} l(\boldsymbol{\beta},\sigma^2,\boldsymbol{\theta}|\boldsymbol{y}) =& -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2}\log(|\boldsymbol{R}|) \\ & - \frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{F}\boldsymbol{\beta})^T\boldsymbol{R}^{-1/2}(\boldsymbol{y}-\boldsymbol{F}\boldsymbol{\beta}). \end{aligned}$$

Maximizing with respect to each of the parameters $\boldsymbol{\beta}$ and σ^2 results in the estimates of

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{y} \tag{2.20}$$

and

$$\hat{\sigma}^2 = \frac{1}{n}\left(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}}\right)^T\boldsymbol{R}^{-1}\left(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}}\right). \tag{2.21}$$

We can use the above two substitutions for $\boldsymbol{\beta}$ and σ^2 to provide the profile likelihood function [28] that relies only on $\boldsymbol{\theta}$, which is contained within $\boldsymbol{R}$

$$l(\boldsymbol{\theta}|\boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\hat{\sigma}^2) - \frac{1}{2}\log(|\boldsymbol{R}|) - \frac{n}{2}. \tag{2.22}$$

Finding the maximum of the likelihood function with respect to the parameter, $\boldsymbol{\theta}$, requires numerically optimizing Equation 2.22.

2.5 Modelling

2.5.1 Correlation Function

The successfulness of the Gaussian process is dependent upon the choice of the correlation function R. There are some general properties that we would like to see characterized by the function. They are as follows:

- Replicate observations should be perfectly correlated, so $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(j)}$ then $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = 1$.
- Points that are close to one another should have function values that are highly correlated. That is, if $||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}||^2 = \sum_{k=1}^{d}(x_k^{(i)} - x_k^{(j)})^2$ is small, then $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ should be close to one.
- Points that are far away from one another should have function values that are basically unrelated (not correlated). That is, if $||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}||^2$ is large then $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ should be close to zero.

There are many different correlation functions available for use. The most common are the Matérn, power exponential, and the Gaussian correlation functions. The Gaussian correlation function is widely used as the correlation function of choice [18, 25, 27] and we choose to use it due to its characteristic of being indefinitely differentiable, and therefore is a very smooth function. The Gaussian correlation function for a d-dimensional function has the general form of

$$R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} | \boldsymbol{\theta}\right) = \prod_{k=1}^{d} \exp\left\{-\theta_k \left(x_k^{(i)} - x_k^{(j)}\right)^2\right\}, \tag{2.23}$$

where $\theta_k \in [0, \infty)$ for all $k = 1, \ldots, d$. The values of the parameter $\boldsymbol{\theta}$ control the activity of the function being modelled. A value of $\theta_k = 0$ removes the kth dimension from the correlation completely. Specifically, a value of $\theta_k = 0$ in the kth dimension corresponds to an overall value of 1 in the product for that dimension. This value of 1 results in no change, or effect, on the final value of R due to the kth dimension. This means that the kth dimension does not explain any of the variability of the function output, $\boldsymbol{y}$. Oppositely, large values of θ_k correspond to a more active or fluctuating function in the kth dimension. Some implementations of a GP use varying forms of the parameter θ [19] and [21]. These various parameterizations will be explored further in Section 3.2.

2.5.2 Prediction

Consider the case of predicting the output, $m'(\boldsymbol{x})$ for a single point $\boldsymbol{x}^*$, where $\boldsymbol{x}^* = \boldsymbol{x}^{(i)}$ for some $i = 1, \ldots, n$. In this case, $\boldsymbol{F}^* = f^T(\boldsymbol{x}^{(i)})$, and $\boldsymbol{r}^T = \big(R(\boldsymbol{x}^i, \boldsymbol{x}^{(1)}, \ldots, R(\boldsymbol{x}^i, \boldsymbol{x}^{(n)})\big)$, which is the ith row of the original variance-covariance matrix $\boldsymbol{R}$. That is,

$$\boldsymbol{r} = \boldsymbol{R}\,(0, \ldots, 0, 1, 0, \ldots, 0)\,,$$

where the 1 is in the ith position. We will denote this vector as $\boldsymbol{u}_i$ as it is the ith unit vector, or ith row of the identity matrix $\boldsymbol{I}$. $\boldsymbol{r}$ can then be denotd as:

$$\boldsymbol{r} = \boldsymbol{R}\boldsymbol{u}_i.$$

Multiplying both sides of the above equation by the inverse of $\boldsymbol{R}$ we obtain,

$$\boldsymbol{R}^{-1}\boldsymbol{r} = \boldsymbol{u}_i.$$

Take the transpose of each side to obtain

$$\boldsymbol{r}^T\boldsymbol{R}^{-1} = \boldsymbol{u}_i^T.$$

We can then substitute the above result (as well as the simplification of $F^* = f^T(\boldsymbol{x}^{(i)})$ into Equation 2.16:

$$\begin{aligned}
\boldsymbol{m}^* &= \boldsymbol{F}^*\boldsymbol{\beta} + (\boldsymbol{r}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})) \\
&= f^T(\boldsymbol{x}^{(i)})\boldsymbol{\beta} + (\boldsymbol{u}_i^T(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})) \\
&= f^T(\boldsymbol{x}^{(i)})\boldsymbol{\beta} + y^{(i)} - f^T(\boldsymbol{x}^{(i)})\boldsymbol{\beta} \\
&= y^{(i)}
\end{aligned}$$

obtaining the original output value $y^{(i)}$ as the prediction of the point $\boldsymbol{x}^{(i)}$.

Now let us consider the case of predicting $\boldsymbol{m}'$ for the full set of n training points; $\boldsymbol{x}^* = \boldsymbol{x}$. In this case, our original dataset $\boldsymbol{x} = \boldsymbol{x}^*$ which means that $\boldsymbol{r} = \boldsymbol{R}$. Recalling that $\boldsymbol{R}$ is the correlation matrix and is therefore symmetric, we have $\boldsymbol{R}^T = \boldsymbol{R}$. Finally, seeing as there are the same number of points in $\boldsymbol{x}$ and $\boldsymbol{x}^*$ then $\boldsymbol{F}^* = \boldsymbol{F}$. We can then calculate $\boldsymbol{m}'$:

$$\begin{aligned}
\boldsymbol{m}^* &= \boldsymbol{F}^*\boldsymbol{\beta} + (\boldsymbol{r}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})) \\
&= \boldsymbol{F}\boldsymbol{\beta} + (\boldsymbol{R}^T\boldsymbol{R}^{-1}(\boldsymbol{y} - F\boldsymbol{\beta})) \\
&= \boldsymbol{F}\boldsymbol{\beta} + (\boldsymbol{R}\boldsymbol{R}^{-1}(\boldsymbol{y} - F\boldsymbol{\beta})) \\
&= \boldsymbol{F}\boldsymbol{\beta} + (I(\boldsymbol{y} - F\boldsymbol{\beta})) \\
&= \boldsymbol{F}\boldsymbol{\beta} + \boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta} \\
&= \boldsymbol{y}
\end{aligned}$$

2.6 Optimization

Finding the maximum likelihood estimates requires maximizing Equation 2.22 with respect to the unknown vector $\boldsymbol{\theta}$. There are several optimization methods that could be used to find the maximum likelihood estimated. These include genetic algorithms [14], mesh adaptive seeded algorithms [2], simplex algorithms (such as the Nelder-Mead simplex algorithm [6]), and gradient based methods [24]. Genetic algorithms will converge to the global maximum if enough iterations are performed, which means they can be very slow. Both the mesh adaptive seeded algorithms and simplex algorithm do not require the gradient in order to perform the optimization. Typically, gradient-based algorithms are faster than non-gradient based algorithms, providing the gradient is easy to compute. In the case of the GP, the computation of the gradient is straightforward and the computational cost is less than computing the likelihood meaning that gradient based methods should be ideal in this situation. In order to deal with the possibly bounded search space on $\boldsymbol{\theta}$ we suggest using the bounded Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS-B) method or its unbounded counterpart, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. The purpose of the L-BFGS-B algorithm is to minimize a d dimensional nonlinear function, $f(x)$ where

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

where $\boldsymbol{l}$ and $\boldsymbol{u}$ are vectors containing the lower and upper bounds on the variable $\boldsymbol{x}$ respectively. Some, or all of the variables may not have bounds resulting in unconstrained optimization. At each iteration of the algorithm an approximation to the Hessian matrix is updated. This matrix is used to define a quadratic model of the function. Using a two-stage approach, a search direction is then calculated; first the gradient projection method is used to identify active variables and then the quadratic model is approximately minimized with respect to the remaining free variables. This search direction is defined as the vector leading from the currently position to the approximate minimizing location. Finally, a line search is performed along the search direction [30]. The Broyden-Fletcher-Goldfarb-Shanno method is implemented using the gradient, but will require a method to choose starting values for the parameter with which we are optimizing in respect to. It is possible to implement BFGS without a gradient as the optim function will compute a finite-difference approximation. [16] shows that fitting the Gaussian process without using the gradient works well, but we could make use the knowledge of the gradient function which we compute below in Equation

2.26 for our optimization.

2.6.1 Gradient

The exact form of the analytical gradient will depend on the correlation function used, but the general form of the gradient can be calculated without it. To calculate the gradient we use the profile log-likelihood function from Equation 2.22

$$l(\boldsymbol{\theta}|\hat{\boldsymbol{\beta}}, \hat{\sigma}^2, \boldsymbol{y}) \quad = \quad -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\hat{\sigma}^2) - \frac{1}{2}\log(|\boldsymbol{R}|) - \frac{n}{2} \tag{2.25}$$

The gradient is then the derivative of this function with respect to the parameter of interest. In this derivation we will use the parameter θ_ℓ for illustrative purposes. The ℓ^{th} entry of the gradient vector, $\boldsymbol{G}$, taken with respect to θ_ℓ is calculated as

$$\begin{aligned}
G_{\ell,\theta_\ell} &= \frac{\partial}{\partial\theta_\ell}\left[l(\boldsymbol{\theta}|\hat{\boldsymbol{\beta}}, \hat{\sigma}^2, \boldsymbol{y})\right] \\
&= \frac{\partial}{\partial\theta_\ell}\left[-\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\hat{\sigma}^2) - \frac{1}{2}\log(|\boldsymbol{R}|) - \frac{n}{2}\right] \\
&= -\frac{n}{2}\frac{1}{\hat{\sigma}^2}\frac{\partial}{\partial\theta_\ell}[\hat{\sigma}^2] - \frac{1}{2}\frac{1}{|\boldsymbol{R}|}\frac{\partial}{\partial\theta_\ell}[|\boldsymbol{R}|] \\
&= -\frac{n}{2}\frac{1}{\hat{\sigma}^2}\frac{\partial}{\partial\theta_\ell}\left[\frac{1}{n}(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}})^T\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}})\right] - \frac{1}{2}\frac{1}{|\boldsymbol{R}|}\frac{\partial}{\partial\theta_\ell}[|\boldsymbol{R}|] \\
G_{\ell,\theta_\ell} &= \frac{1}{2\hat{\sigma}^2}(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}})^T\boldsymbol{R}^{-1}\frac{\partial}{\partial\theta_\ell}[\boldsymbol{R}]\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{F}\hat{\boldsymbol{\beta}}) \\
&\quad - \frac{1}{2}\frac{1}{|\boldsymbol{R}|}|\boldsymbol{R}|\texttt{tr}\left(\boldsymbol{R}^{-1}\frac{\partial}{\partial\theta_\ell}[\boldsymbol{R}]\right).
\end{aligned} \tag{2.26}$$

The gradient is now solely in terms of $\frac{\partial}{\partial\theta_\ell}[\boldsymbol{R}]$. For the case of the Gaussian correlation function introduced in Section 2.5.1, we can calculate $\frac{\partial}{\partial\theta_\ell}R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}|\boldsymbol{\theta}\right)$ by taking the derivative of Equation 2.23 with respect to θ_ℓ.

$$\frac{\partial}{\partial\theta_\ell}R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}|\boldsymbol{\theta}\right) = \exp\left\{\sum_{k=1}^{d} -\theta_k\left(x_k^{(i)} - x_k^{(j)}\right)^2\right\}\left(-\left(x_\ell^{(i)} - x_\ell^{(j)}\right)^2\right). \tag{2.27}$$

Substituting the above into Equation 2.26 and repeating for each of the dimensions, $\ell = 1, \ldots, d$, we can achieve the full analytical gradient for use in optimization. In the next chapter we discuss the full implementation of the GP model which involves selecting an initial design, specifying the exact parameterization of the correlation matrix, maximizing the likelihood and predicting a set of new observations.

Chapter 3

Implementation

In practice, fitting a Gaussian process requires selecting the initial design, optimizing the log-likelihood function in Equation 2.22 with respect to the parameter of choice by estimating the parameter values, and then predicting values for a new set of input points.

3.1 Design

The problem of selecting the initial input values, $\boldsymbol{x}$, is a common topic in computer modelling. The approach we will take is to treat the input values themselves as random variables. This method models the uncertainty of the input values themselves.

We will refer to the initial input data as the design of the experiment. Due to our predicting functions defined in Equations 2.16 and 2.17, it is important to create a design that is space-filling.

The prediction equations are interpolators, as discussed in Section 2.5.2, which results in the prediction error at a given input site being a function of its location (relative to the design points $\boldsymbol{x}$). If we were to select a non-space-filling design, it may yield predictors that perform poorly in the areas that are sparsely observed.

There are many different approaches that we could use in the design process; we could select points using a line, a simple random sample, or a latin hypercube sample, just to name a few. The random latin hypercube sampling method has been used by many as the approach in designing experiments, [13, 20, 23, 28] and it is the method that we will use to create the initial design for our study. First introduced by [12], the latin hypercube sampling method allows all portions of the sample space for a single dimension, S, to be represented. The method starts by dividing the sample space into n equal-width intervals, $S_i \in S$, where n is the sample size, and $i = 1 \dots n$. For simplicity, we will continue with a sample space of $S = [0, 1]$. To form the latin hypercube sample, we will select points from $\boldsymbol{v}$, where $\boldsymbol{v} = \left\{ \frac{R_i + \varepsilon_i}{n-1} | R_i = i - 1, i = 1 \dots n \right\}$. Using the points in $\boldsymbol{v}$, we will give them a random ordering forming a vector. We continue this method for each of the d dimensions we can then match the vectors as columns in a matrix to form the final design. This results in an n-by-d matrix where each column is some permutation of the elements contained in $\boldsymbol{v}$.

There are different methods to set the values of ε_i. We could set a specific value such as $\varepsilon_i = 0$, or $\varepsilon_i = 1/2$, or we could use a distribution such as $\varepsilon_i \sim UNIF(0, 1)$. Selecting a value of $\varepsilon_i = 0$ forces the points to the boundary of the space, whereas $\varepsilon = 1/2$ places them at the midpoint of each space.

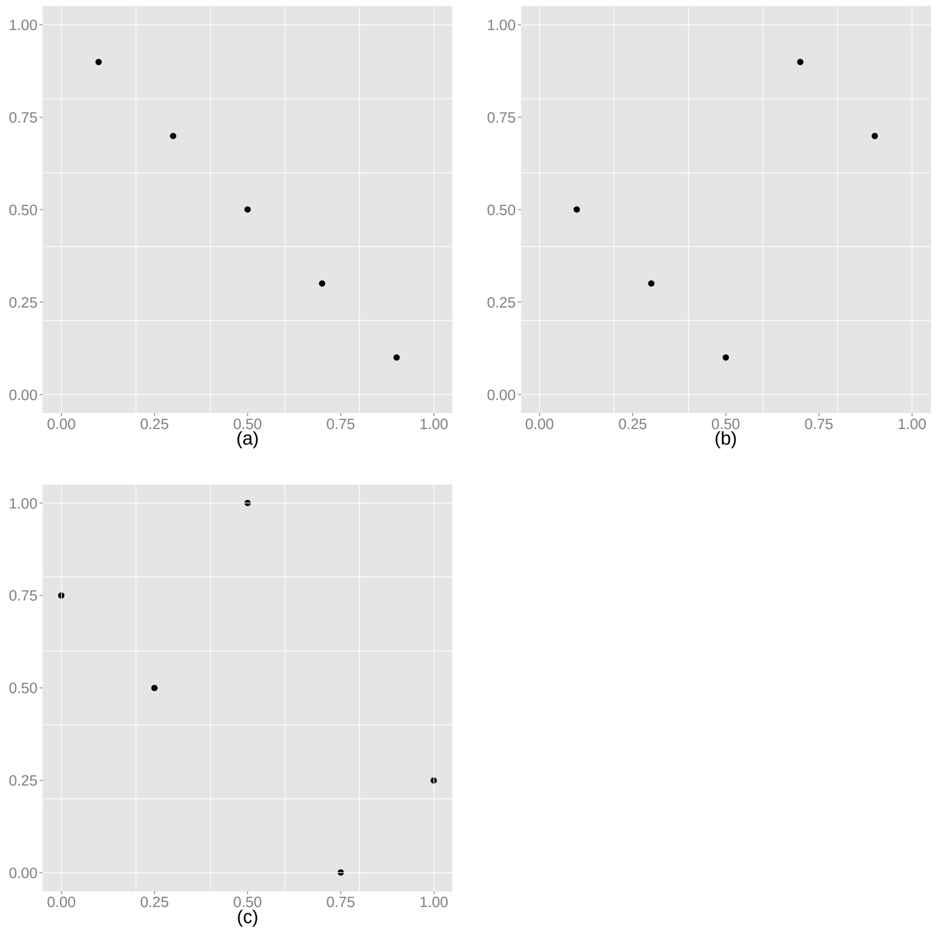

Figure 3.1: Three different samples for the initial input points for $n = 5$ input points with $d = 2$.

Three different realizations of a random latin hypercube sample, with $d = 2$ and $n = 5$, are contained in Figure 3.1. All of Figure 3.1 (a), (b), and (c) are valid latin hypercube samples with each subsection S_i (each visual ‘row’ or ‘column’ division of the graph) containing a point. Figure 3.1 (a) and (b) are two different realizations of the latin hypercube sampling method using $\varepsilon_i = 1/2$ and (c) uses $\varepsilon_i = 0$.

In terms of being a space filling design, the first graph, (a), would be considered a poor design as it leaves two large areas of the sample space not sampled and the same argument can be made for (b) and the area along the diagonal. As well, the use of $\varepsilon_i = 1/2$ in (a) and (b) does not allow for the boundary points to be sampled which can allow for boundary effects to not be caught when fitting the GP. Graph (c) uses $\varepsilon_i = 0$ and forces points to be along the outer boundaries (0 and 1) which will allow boundary effects to be captured in the model created by the Gaussian process. For the this thesis we place points on the boundary by choosing $\varepsilon = 0$.

The maximin latin hypercube sample [9], maximizes the distance between the points in the design, can be used to ensure that the resulting latin hypercube design is space filling. However, in order to make the problem of solving the Gaussian process for an experiment more difficult, we limit our design to a random latin hypercube sample. This will ensure that results generalize to designs that are space filling.

3.2 Correlation Functions

The major focus of this thesis is efficiently optimizing the likelihood, which we will show is highly dependent on the choice of the parameterization of the likelihood. Revisiting the correlation function R introduced in Section 2.5.1 we will explore some other forms (parameterizations) found in the literature. We will denote these different methods listed below as different parameterizations of the correlation function. We wish to investigate these different parameterizations to evaluate if a best all around parameterization can be found to fit a Gaussian process.

Recall the correlation function (parameterization) in Equation 2.23 presented in Section 2.5.1. We will denote this parameterization as theta, or θ.

The next parameterization will be called theta2, or θ^2, and is a variation of the theta parameterization. The correlation function is

$$R\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}|\boldsymbol{\theta}\right)=\prod_{k=1}^{d}\exp\left\{-\theta_k^2\left(x_k^{(i)}-x_k^{(j)}\right)^2\right\}. \tag{3.1}$$

The domain of θ^2 is the same as for the θ case, $\theta_k\in[0,\infty),\forall k$.The θ^2 case spreads out the small values close to zero. The main difference between the two comes to light in the derivative $\frac{\partial}{\partial\theta_\ell}[\boldsymbol{R}]$:

$$\frac{\partial}{\partial\theta_\ell}R\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}|\boldsymbol{\theta}\right)=\exp\left\{\sum_{k=1}^{d}-\theta_k^2\left(x_k^{(i)}-x_k^{(j)}\right)^2\right\}\left(-2\left(x_\ell^{(i)}-x_\ell^{(j)}\right)^2\theta_\ell\right). \tag{3.2}$$

The next parameterization is a form of the Gaussian correlation function that is used in the DiceKriging package in the program R [19]. We will denote it as phi2, or ϕ^2. It varies from the previous parameterizations by introducing the parameter in the denominator of the exponent. The correlation function for phi2 is

$$R\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}|\boldsymbol{\phi}\right)=\prod_{k=1}^{d}\exp\left\{-\left(\frac{x_k^{(i)}-x_k^{(j)}}{\sqrt{2}\phi_k}\right)^2\right\}. \tag{3.3}$$

Due to the parameter being in the denominator, the domain is now $\phi_k\in(0,\infty),\forall k$. The partial derivative for the gradient is calculated to be

$$\frac{\partial}{\partial\phi_\ell}R\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}|\boldsymbol{\phi}\right)=\exp\left\{\sum_{k=1}^{d}\frac{-1}{2\phi_k^2}\left(x_k^{(i)}-x_k^{(j)}\right)^2\right\}\left(\phi_\ell^{-3}\left(x_\ell^{(i)}-x_\ell^{(j)}\right)^2\right). \tag{3.4}$$

The next parameter beta (β) was introduced by [16] and is the parameterization used in the study done by [3].It takes a different approach to the Gaussian correlation form raising the parameter as an exponent:

$$R\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}|\boldsymbol{\beta}\right)=\prod_{k=1}^{d}\exp\left\{-10^{\beta_k}\left(x_k^{(i)}-x_k^{(j)}\right)^2\right\}. \tag{3.5}$$

The domain for β is $\beta_k\in(-\infty,\infty),\forall k$, and the derivative can be calculated as

$$\frac{\partial}{\partial\beta_\ell}R\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}|\boldsymbol{\beta}\right)=\exp\left\{\sum_{k=1}^{d}-10^{\beta_k}\right\}\left(-\ln(10)\left(x_\ell^{(i)}-x_\ell^{(j)}\right)^2 10^{\beta_\ell}\right). \tag{3.6}$$

The next parameterization was suggested by [8] and [10] and it places the parameter in the base of the equation rather than the exponent. The domain of this parameter, labeled rho (ρ), is restricted to $\rho_k \in (0,1], \forall k$ and the correlation function is

$$R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} | \boldsymbol{\rho}\right) = \prod_{k=1}^{d} \rho_k^{4\left(x_k^{(i)} - x_k^{(j)}\right)^2}. \tag{3.7}$$

The derivative can be shown to be

$$\begin{aligned}\frac{\partial}{\partial \rho_\ell} R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} | \boldsymbol{\rho}\right) = &\left(\prod_{k \neq \ell} \rho_k^{4\left(x_k^{(i)} - x_k^{(j)}\right)^2}\right) \\ &\cdot \left(4\left(x_\ell^{(i)} - x_\ell^{(j)}\right)^2 \rho_\ell^{4\left(x_\ell^{(i)} - x_\ell^{(j)}\right)^2 - 1}\right),\end{aligned} \tag{3.8}$$

which can be simplified to

$$\begin{aligned}\frac{\partial}{\partial \rho_\ell} R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} | \boldsymbol{\rho}\right) = &\left(\prod_{k=1}^{d} \rho_k^{4\left(x_k^{(i)} - x_k^{(j)}\right)^2}\right) \\ &\cdot \left(\frac{4\left(x_\ell^{(i)} - x_\ell^{(j)}\right)^2}{\rho_\ell}\right).\end{aligned} \tag{3.9}$$

The final parameterization will be labeled as lambda (λ). It is a rearrangement of the ρ parameterization with relation

$$\lambda_k = \log(\rho_k) - \log(1 - \rho_k),$$

or

$$\rho_k = \frac{\exp\{\lambda_k\}}{1 + \exp\{\lambda_k\}}.$$

Using the above relation, the correlation function is

$$R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} | \boldsymbol{\lambda}\right) = \prod_{k=1}^{d} \left(\frac{\exp\{\lambda_k\}}{1 + \exp\{\lambda_k\}}\right)^{4\left(x_k^{(i)} - x_k^{(j)}\right)^2}, \tag{3.10}$$

with corresponding derivative

$$\begin{aligned}\frac{d}{d\lambda_\ell} R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} | \boldsymbol{\lambda}\right) = &\left(\prod_{k=1}^{d} \left(\frac{\exp\{\lambda_k\}}{1 + \exp\{\lambda_k\}}\right)^{4\left(x_k^{(i)} - x_k^{(j)}\right)^2}\right) \\ &\cdot 4\left(x_\ell^{(i)} - x_\ell^{(j)}\right)^2 \frac{1 - 2\exp\{\lambda_\ell\}}{1 + \exp\{\lambda_\ell\}}.\end{aligned} \tag{3.11}$$

The five parameterizations presented above as well as the one first introduced in Section 2.5.1 will be used in fitting the model for the Gaussian process. As specified in Section 2.6, two methods of optimization (L-BFGS-B and BFGS) along with two versions of the gradient (analytical and numerical) will be used. Table 3.1 is a summary table specifying the parameterization for the correlation function, label used to differentiate between fits, the optimization method used, the bounds used (in the case of the bounded optimizer), and the gradient used in fitting the model through the Gaussian process.

Table 3.1: Summary table specifying: parameterization, optimization method, and type of gradient used in the Gaussian process.

Parameter	Label	Optimizer	Bounds	Gradient
β	Bt	BFGS		analytical
β	Bt_ng	BFGS		numerical
λ	Lam	BFGS		analytical
ϕ^2	Ph1	L-BFGS-B	$(0, 5]$	analytical
ϕ^2	Ph2	L-BFGS-B	$(0, 100]$	analytical
ρ	Rh	L-BFGS-B	$(0, 1]$	analytical
ρ	Rh_ng	L-BFGS-B	$(0, 1]$	numerical
θ	Th1	L-BFGS-B	$[0, \infty)$	analytical
θ^2	Th	L-BFGS-B	$[0, \infty)$	analytical
θ^2	Th_un	BFGS		analytical

3.2.1 Invariance

Likelihood theory shows that maximum likelihood estimates are invariant for one-to-one transformations of the parameters [29]. That is to say, if $\hat{\theta}$ is the maximum likelihood estimator (MLE) of θ, for any function $\tau(\theta)$, the MLE of $\tau(\theta)$ is $\tau(\hat{\theta})$. To show this, consider $\eta = \tau(\theta)$ and the defined induced likelihood function, L^*, as

$$L^*(\eta|\boldsymbol{x}) = \sup_{\{\theta:\tau(\theta)=\eta\}} L(\theta|\boldsymbol{x}).$$

Let $\hat{\eta}$ denote the value that maximizes $L^*(\eta|\boldsymbol{x})$. We want to show that $L^*(\hat{\eta}|\boldsymbol{x}) = L^*[\tau(\hat{\theta})|\boldsymbol{x}]$. As previously stated, the maxima of L and L^* coincide, and we have

$$\begin{aligned}
L^*(\hat{\eta}|\boldsymbol{x}) &= \sup_{\eta} \sup_{\{\theta:\tau(\theta)=\eta\}} L(\theta|\boldsymbol{x}) \\
&= \sup_{\theta} L(\theta|\boldsymbol{x}) \\
&= L(\hat{\theta}|\boldsymbol{x})
\end{aligned}$$

where the second equality follows because the iterated maximization is equal to the unconditional maximization over θ which is attained at $\hat{\theta}$, by definition. Further we can show

$$\begin{aligned}
L(\hat{\theta}|\boldsymbol{x}) &= \sup\{\theta : \tau(\theta) = \tau(\hat{\theta})\} L(\theta|\boldsymbol{x}) \\
&= L^*[\tau(\hat{\theta})|\boldsymbol{x}].
\end{aligned}$$

Using the two above equalities we can show that $L^*(\hat{\eta}|\boldsymbol{x}) = L^*(\tau(\hat{\theta})|\boldsymbol{x})$, and that $\tau(\hat{\theta})$ is the maximum likelihood estimator of $\tau(\theta)$.

A problem arises, with respect to invariance, within the $\boldsymbol{\phi}^2$ parameterization. Consider the case in the $\boldsymbol{\theta}$ parameterization when $\theta_k = 0$; a value of $\theta_k = 0$ corresponds to a value of $\phi_k = \infty$ which is numerically unattainable and results in a lack of invariance. Numerically optimizing $\boldsymbol{\phi}$ requires a limit to be placed for the upper bound which renders the situation non-invariant. Forcing ϕ_k away from ∞ brings the parameter from non-active to active status and can result in overfitting. Different limits on the upper bound can result not only in different maximum likelihood estimates for $\hat{\boldsymbol{\phi}}$, but also differing values for the corresponding log-likelihood (i.e. differing results). The remainder of this thesis is focused on effectively numerically optimizing the likelihood function $L(\boldsymbol{\theta})$ and in turn investigating the effects due to the boundary.

3.3 Optimizing in R

Using the parameterization, optimization, and gradient pairings provided in Table 3.1, we use the statistical programming language R to perform the optimization. The initial input points for $\boldsymbol{x}$, or the design of the experiment, are created using the random latin hypercube method as described in Section 3.1. As mentioned earlier, the latin hypercube sample allows for a space filling sample to be chosen. We require the design to be space filling so that any effects or trends due to a point being a boundary point will be captured in the model fit by the Gaussian process.

To numerically optimize the likelihood function itself, the R function **optim**() is used. The **optim**() function will minimize a provided function rather than maximize it, so the log-likelihood and gradient functions are modified to reflect this. For the remainder of this thesis, we will talk of minimizing the log-likelihood function where a minimal value translates to the optimal value.

The optimization function requires a set of starting values for the parameters. We use a *warm start* method in order to provide these values. A total of $15 \cdot d$ points are generated for each of θ^2 and ϕ^2 using the random latin hypercube method used to generate the input points $\boldsymbol{x}$. For θ^2, the points were generated to be within $[0, \min(0.4 \cdot d, 1.6)]$, and for ϕ^2 the points were within $[10 \times 10^{-10}, \min(0.4 \cdot d, 1.9)]$. We then calculate the log-likelihood value for each of the $30 \cdot d$ points. The twenty points that yield the smallest log-likelihood values, regardless of initial parameterization (θ^2 or ϕ^2), are then used as the twenty starting values for the **optim**() function for all parameterization fittings.

Running the **optim**() function twenty times will result in the output of the optimal log-likelihood values and their corresponding optimal parameters. The best (minimum) log-likelihood value out of these twenty is denoted as the winner of this run and it, along with its corresponding parameter values, are returned as the final result of this run.

3.4 Assessing Fit

Since this thesis is devoted to assessing the 'best' parametrization of the likelihood it is important to determine how we define the term best. The one obvious choice of best is simply based on the finding the largest value of the likelihood. However, one could also considered measures based on the predictive performance of the fitted model. Since all of the example are based on toy problems which mimic real problems we can obtain a large set of points for validation. Given a set of m new locations not used for fitting we can run the code and obtain outputs $y(\boldsymbol{x}^{(j)})$. The two obvious methods for validation are the Root mean squared error and the maximum error. In order to compare different function we consider the standardized values of the measures.

The standardized root mean squared error is given as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(y(\boldsymbol{x}^{(j)}) - \hat{y}(\boldsymbol{x}^{(j)})^2}{\sum_{i=1}^{m}(y(\boldsymbol{x}^{(j)}) - \bar{y}_t)^2}},$$

where $\bar{y}_t$ is the mean of the training data. The standardized max error is defined as

$$AME = \frac{\max_{i=1,\ldots,m} \left| y(\boldsymbol{x}^{(j)}) - \hat{y}(\boldsymbol{x}^{(j)}) \right|}{\max_{i=1,\ldots,m} \left| y(\boldsymbol{x}^{(j)}) - \bar{y}_t \right|}.$$

If each of the methods is equivalent then each of these measures should all be the same. If the methods are not different then smaller values of the likelihood are worse and larger values of the AME and RMSE are worse. We will use each of these three measures together to assess the 'best' parameterization.

Chapter 4

Borehole Example

Consider the Borehole function which calculates the flow rate through a borehole using eight input variables ($d = 8$). The function is

$$f(\boldsymbol{x}) = \frac{2\pi x_3(x_4 - x_6)}{\ln(x_2/x_1)\left(1 + \frac{2x_7x_3}{\ln(x_2/x_1)x_1^2x_8}\right) + \frac{x_3}{x_5}}.$$

The variables and their appropriate ranges are given below:

– radius of the borehole (m) – $x_1 \in [0.05, 0.15]$

– radius of influence (m) – $x_2 \in [100, 50\ 000]$

– transmissivity of upper aquifer (m^2/yr) – $x_3 \in [63\ 070, 115\ 600]$

– potentiometric head of upper aquifer (m) – $x_4 \in [990, 1\ 110]$

– transmissivity of lower aquifer (m^2/yr) – $x_5 \in [63.1, 116]$

– potentiometric head of lower aquifer (m) – $x_6 \in [700, 820]$

– length of borehole (m) – $x_7 \in [1\ 120, 1\ 680]$

– hydraulic conductivity of borehole (m/yr) – $x_8 \in [9\ 855, 12\ 045]$.

This function is considered easy to fit using a Gaussian process. Using the procedure outlined in Chapter 3, we will fit the model for the borehole function using the Gaussian process. We start by selecting the design of the initial points by using the random latin-hypercube sampling design of size $7 \cdot d = 56$ points. Now that we have the initial input points, we can fit the model using the Gaussian process. We will use the Gaussian correlation function parameterizations as outlined in Sections 2.5.1 and 3.2. A total of eleven different models are to be created with the parameter-optimizer-gradient pairs outlined in Table 3.1. These different methods all used the same starting design, of the $7 \cdot d$ initial points.

The starting values of the parameters are chosen using a warm start method. Initially, $15 \cdot d$ starting values for the θ^2 parameterization are generated using a random latin hypercube sample in $[0, \min(0.4 \cdot d, 1.6)]$. Another $15 \cdot d$ starting points are generated for the ϕ^2 parameterization on the interval $[10^{-10}, \min(0.4 \cdot d, 1.9)]$. The log-likelihood values are calculated for both sets of $15 \cdot d$ points. These $30 \cdot d$ log-likelihood values are sorted and the twenty smallest values are chosen to be the starting values of the warm start. The parameter values corresponding to the winning fits are then translated to be in terms of both θ^2 and ϕ^2 using the relations:

$$\theta = \frac{1}{\sqrt{2\phi^2}},$$

and

$$\phi = \frac{1}{\sqrt{2\theta^2}}.$$

Now we can use the following relations to translate the initial starting points to be in terms of all other parameterizations

$$\theta = \sqrt{\theta^2},$$

$$\beta = \log_{10}(\theta^2),$$

$$\rho = \exp\left\{\frac{-\theta^2}{4}\right\},$$

$$\lambda = \frac{-\theta^2}{4} - \log\left(1 - \exp\left\{\frac{-\theta^2}{4}\right\}\right).$$

Each parameterization will then start on even ground by starting with equivalent parameter values. The warm start method allows us to use the computationally cheaper method (compared to optimization) of calculating the log-likelihood values to find parameter locations of lower log-likelihood values from which to start our optimization. The warm start saves in terms of the number of function calls during the optimization as well as computational time. The twenty sets of parameter points are used as the starting parameter values for a run though the optimizer. The output will consist of twenty sets of optimal parameter points and the corresponding optimal log-likelihood values. The final result of the run, deemed the optimal solution for this fitting, is the minimum log-likelihood value of the twenty values and its corresponding parameter values.

Table 4.1 contains the optimal log-likelihood values for a single run of fitting the model. Multiple methods (β, β(no gradient), θ^2, θ^2(unbounded), and θ^2(β starting points)), were able to achieve the same result (to three decimal places). Others (specifically λ, ϕ^2(upper bound = 5), ρ, and θ)

Table 4.1: Log-likelihood values for the 10 optimizations for one fitting.

Fit	Method	Log-likelihood	RMSE	Max error
1	Bt	125.016	0.393	6.514
2	Bt_ng	125.016	0.393	6.514
3	Lam	164.347	2.866	15.858
4	Ph1	161.518	3.458	20.967
5	Ph2	130.226	0.500	7.787
6	Rh	214.874	36.496	67.146
7	Rh_ng	125.984	0.486	6.669
8	Th	125.016	0.393	6.514
9	Th_un	125.016	0.393	6.514
10	Th1	136.746	1.019	10.144
11	ThB	125.016	0.393	6.514

performed poorly, achieving log-likelihood values greater much greater than optimal value achieved, 125.016. In this small example we will repeated the above process four more times to get a total of five different results.

We repeated the process by starting with generating a new set of input points, $\boldsymbol{x}$, for each of the replicates as well as a new set of starting points for the parameters before running them through the optimization process and getting the outputs. Figure 4.1 shows the log-likelihood value results for each of the five simulations. The eleven parameterizations are along the x-axis with labels corresponding to Method column in Table 4.1. Each of the simulations is represented by a line. The first simulation ran, the one where the values are shown in Table 4.1, is shown as the red line along with the four replicates. Figure 4.2 shows the standardized root mean squared errors from fitting the five simulations (the individual root mean squared errors divided by the root mean squared error from the trivial predictor, y. Looking at both Figure 4.1 and Figure 4.2 we can start to see a pattern across the parameterizations. The parameterizations that performed well on the first simulation (smaller log-likelihood and root mean squared error values) as seen by the red line, also performed well on the other four replicates. Similarly, the parameterizations that performed poorly with high log-likelihood and root mean square error values on the first simulation also performed poorly on the following replicates.

The drastic differences in the final log-likelihood values achieved by the different parameterizations hint at the problem of invariance outlined in Section 3.2.1. As stated, likelihood theory shows that maximum likelihood

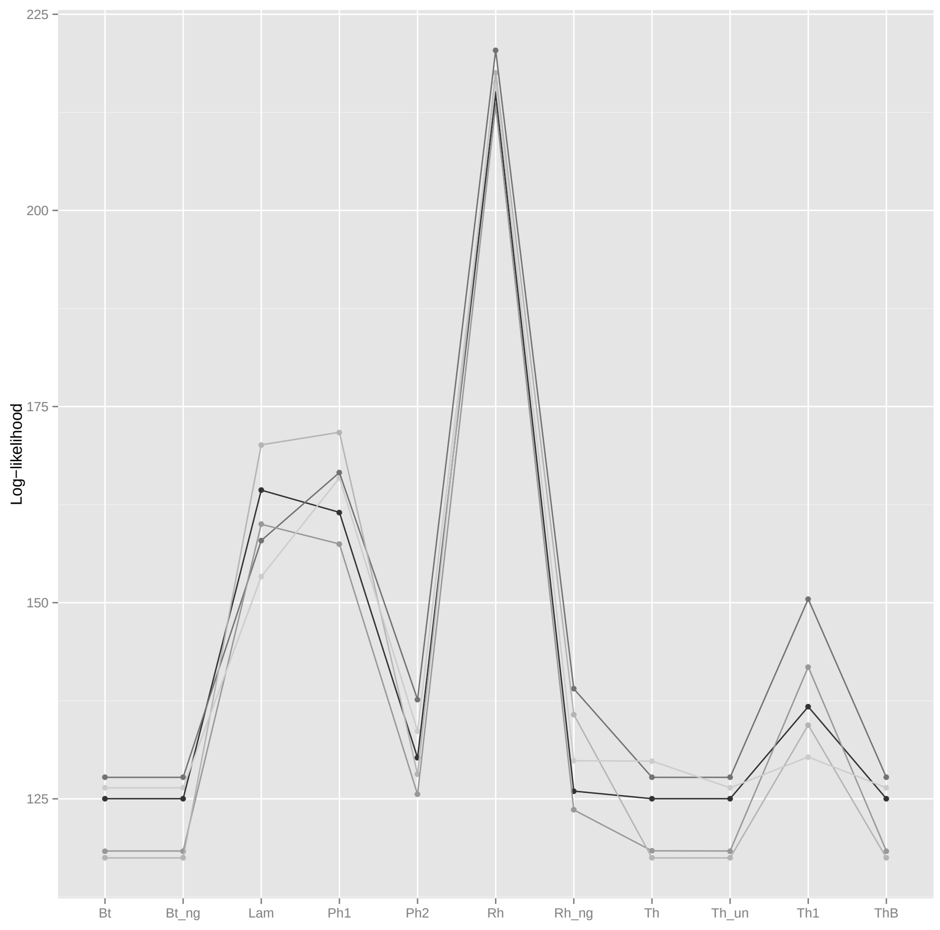

Figure 4.1: Log-likelihood values for the 5 different fittings of the Borehole test function with 11 parameter-optimizer combinations.

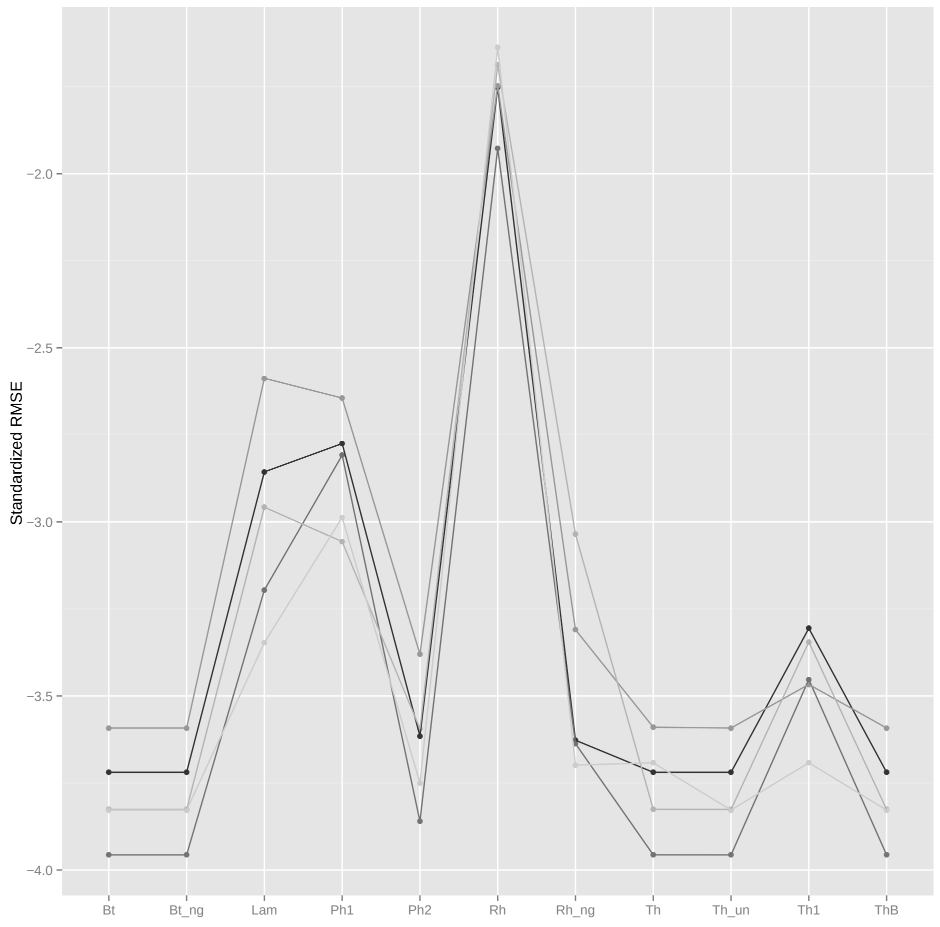

Figure 4.2: Standardized root mean squared errors for the 5 different fittings of the Borehole test function with 11 parameter-optimizer combinations.

estimates are invariant for one-to-one transformations. This would mean that for the transformations of the theta parameter, we expect to see the same optimal log-likelihood values produced from optimizing the different parameterizations. As can be seen in Figure 4.1, this is not the case. For the two fits using the phi parameterization there is a difference due to the bounds imposed. In the original parameterization, theta, the bounds were $\theta \in [0, \infty)$. Looking back at the translation between the ϕ and θ^2 parameterizations we have

$$\theta = \frac{1}{\sqrt{2\phi^2}}.$$

The lower bound value of $\theta = 0$ corresponds to a value of ∞ for ϕ, and the upper bound value of ∞ for θ to a value of 0 for ϕ. A value of $\phi = 0$ is not possible due to the parameter being in the denominator of the exponent and therefore a restriction is imposed upon the domain of ϕ resulting in a domain of $(0, \infty)$. This non-equivalence in the domains results in the violation of the assumptions of invariance.

Taking a closer look at the two ϕ parameterizations, the only difference in their implementation is the value of the upper bounds used. Figure 4.3 shows the log-likelihood values for the five different fits for just the two ϕ parameterizations. The fit using upper bound equal to 5 is denoted as a value of 1 on the x-axis and upper bound equal to 100 is denoted as 2. We can see that there is clearly a difference between the two log-likelihood results. Looking closer at the values of the upper bounds, consider the case of $\phi = 5$. In the θ^2 parameterization, this is equivalent to $\theta = \frac{1}{\sqrt{2 \cdot 25}}$ as a lower bound. The case of $\theta_k \neq 0, \forall k = 1, \ldots, d$ forces activity in each of the d dimensions. Even using the larger value of $\phi = 100$ results in a lower bound to nearly 0. This allows for inactive dimensions to become less active than the upper bound value of 5, but there would still be forced activity. Comparing these two cases to the default upper bound used by the R package DiceKriging, $\phi = 2$, and we can use the same relation to show that the DiceKriging scenario leads to a higher lower bound in the θ^2 case and therefore results in more activity than the two bounds that we use here.

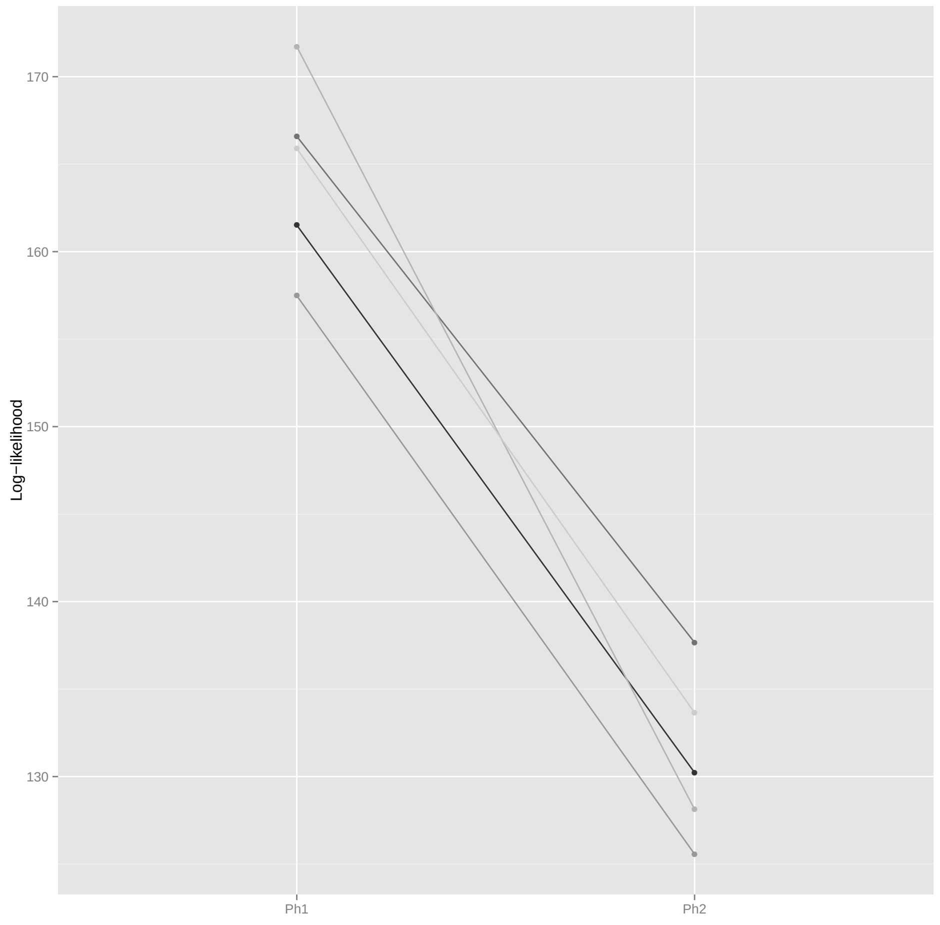

Figure 4.3: A closer examination of the log-likelihood values of the two ϕ^2 parameterizations using upper bounds of 5 (left) and 100 (right).

Chapter 5

Numerical Testing

5.1 Full Simulation Study

The goal of this thesis is to determine which parameterization works the best across a set of various test problems. It is our wish to provide one parameterization to be deemed the 'go to' parameterization for anyone wishing to fit a Gaussian process. In order to determine which method performs the best we will need to test the different parameterizations on a wide range of test problems. We use fourteen different functions of varying dimensions ($d = 2$ to $d = 10$) for the full simulation. The full list of test functions can be found in Appendix A.

A total of eight hundred and forty tests are performed. A single test consists of fitting the model for each of the parameterizations using the method outlined in Chapter 3. Three different sample sizes are used for the creation of the initial design, $\boldsymbol{x}$: $7{\cdot}d$, $10{\cdot}d$, and $15{\cdot}d$. [11] provide evidence supporting the initial design choice of $10 \cdot d$ as being an effective design. Overall, there are a total of forty-two different sample size and function combinations. We repeat each of these forty-two tests using the replication procedure used to get the four additional replicates done in Chapter 4. We repeat to get a total of twenty replicates of each sample size-function combination. For a total of eight hundred and forty tests. We treat each of the simulations and replicates as separate problems with each needing analysis to determine which parameterization yields the best results for that problem.

5.2 Numerical Results

As we to answer the question of which optimizer is the best using our large set of test problems, the hope is that the best solver on our set of test problems will also be the best solver on a set of currently untested problems. Performing the full simulation outlined in Section 5.1 results in solutions for each of the eight hundred and forty tests. The solutions consist of the optimal log-likelihood values for each of the parameterizations as well as their corresponding parameter values. The obvious choice is to benchmark using the log-likelihood value itself to determine which solution is the optimal solution. The minimum log-likelihood value out of the eleven different parameterization options is the optimal solution for that run. However, using the log-likelihood values for all of the eight hundred and forty tests to determine which parameterization performs the best is not feasible as there is no way to make direct comparisons between the parameterizations. Figure 5.1 shows the distributions of the log-likelihood values for each of the parameterizations. The solutions come from the runs of fitting a Gaussian process for different functions. These different functions' optimal log-likelihood values may be drastically different from one another rendering a between-parameterization comparison uninformative. The graph does not allow us to link the log-likelihood values to the functions and test cases that they belong to, so it is not possible to determine if the log-likelihood value achieved was good or not.

5.3 Error Analysis

There are many different tools that are available for analysis of statistical models. Briefly mentioned in Chapter 4 two of the techniques that we will use are the root mean squared error, and the absolute maximum error. Recall the RMSE for a set of m prediction points

$$RMSE = \sum_{i=1}^{m} \sqrt{(y_i - \hat{y}_i)^2},$$

where y_i is the true value of the test function at input $\boldsymbol{x}_i$ and $\hat{y}_i$ is the predicted value calculated using the model produced from the Gaussian process calculated using $\boldsymbol{x}_i$. Similarly the absolute maximum error AME is calculated for the prediction points as

$$AME = \max_{i=1,\ldots,m} |y_i - \hat{y}_i|,$$

with the same definitions of y_i and $\hat{y}_i$ as above. We standardize the errors by dividing by the errors obtained by the trivial predictor of $\hat{y}_i = \bar{\boldsymbol{y}}$, where $\bar{\boldsymbol{y}}$ is the mean of the responses originally used in fitting the Gaussian process. Using these standardized measure of errors, any value greater than 1 is considered to be extremely ill fitting as they perform worse than the trivial predictor. As well, any predictors with a value greater than 0.5 are deemed poor fitting. Finally, we take the logarithm (base 10) to spread the data

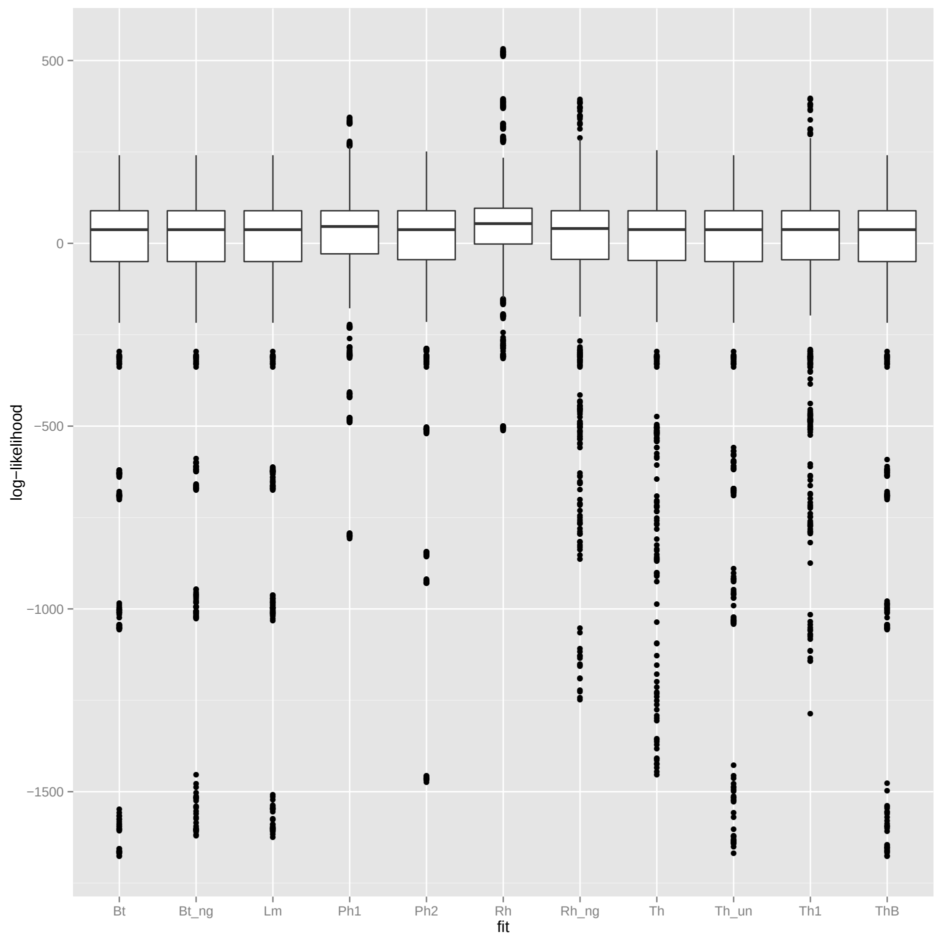

Figure 5.1: A boxplot of the log-likelihood values across all parameterizations for the full test.

outwards because the original values are small and close to zero. Now, an extremely ill fitting model is one that produces an error value above 0, and an poor fit is any fit with an error above $\log_{10}(0.5) = -0.301$.

Both the RMSE and AME create one single measure of error using all of the predicted values from the fitting as well as the true function values. For both measures, a smaller value represents a better solution; the smaller the value the smaller the difference between the predicted value and true function value and therefore the better the fit. Figure 5.2 shows two side-by-side box plots for the RMSE and AME values for the full set of simulation test cases. Looking at the RMSE box plot, the majority of the values are below zero, and there is no distinct difference between the various fits. The AME box plot is similar with a smaller spread, and some values greater than zero, but no real difference between the distributions of the parameterizations. Using the box plots, it is difficult to determine a difference between the distributions of the different parameterizations and therefore determine which method yields the best results.

5.4 Performance Analysis

The log-likelihood is the most important tool to be used in evaluating the performance of a model, but the error terms are also an important approach in looking for discrepancies. One of the main purposes of fitting a Gaussian process is to gain a model from which we can predict for new points. The discrepancies occur when we have a fit-produced model that has the best log-likelihood value, but it might produce predictions that yield large errors and therefore be a bad fit prediction-wise.[1] argue for the use of a simple graphical method for analyzing statistical methods. The graph they suggest, the Empirical-Cumulative-Distribution-Function (ECDF), yields some advantages in assessing the performance of the different methods presented. The main advantage of the ECDF (when comparing to other commonly used methods such as the box plot) is that it displays every data point while also allowing for easy interpretation. As well, the ECDF allows for multiple methods to be displayed simultaneously and for direct comparisons between the methods to be made easily. Figure 5.3 shows the ECDF's of both the RMSE's and the AME's for each of the parameterizations.

Compared to the boxplot representation in Figure 5.2, in the ECDF we can see distinct differences in the distributions of the parameterizations. There is a set of parameterizations that are clearly set away from the others concentrated on the left side of the graph. The ECDF highlight the func-

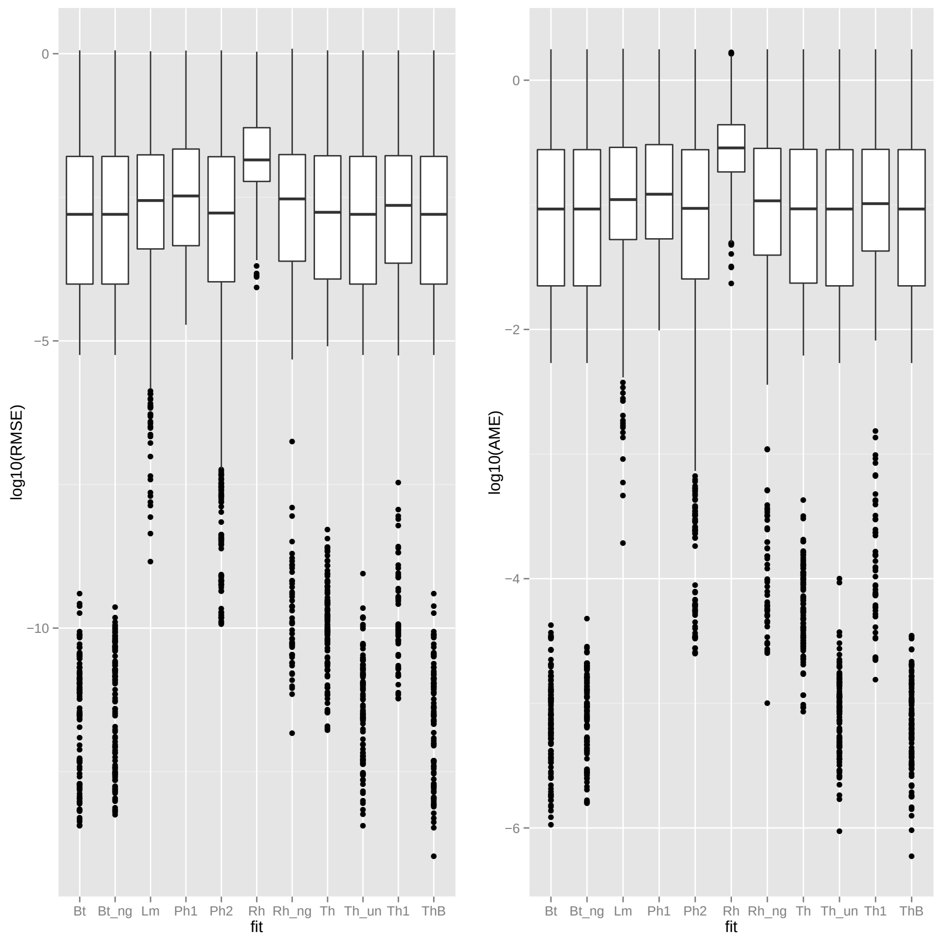

Figure 5.2: Box plots of the logarithm (base ten) of both of the RMSE and AME values .

tions that appear on the left side of the graphs, and which have the largest concentration of small error values and therefore the largest concentration of models (or fits) that perform well. The same evaluation can be performed on the AME graph.

As well as using the different error values, we can create a metric from the log-likelihood values, L_p, to assess the performance of the fits of the Gaussian process. We define the performance using the following definition of L_p:

$$L_p = \frac{(\ell_p - \ell^*)}{|\ell^*|},$$

where ℓ_p is the log-likelihood solution found by the solver, and ℓ^* is the known, optimal, log-likelihood value. The problem with the statistic L_p is that the true solution value, ℓ_*, is not known and instead we are required to use the best (minimum) log-likelihood value found by all of the possible solvers for that run of the problem. We will denote this new measure as the performance of the parameterization. This performance metric allows us to use information from the log-likelihood values without the problem of different optimal values for the different test functions. Small values of L_p indicate that the parameterization achieved the optimal log-likelihood value for that run or a value close to the optimal value found.

Allowing visualization of each of the eleven parameterizations on one graph, the ECDF allows for a quick and reliable decision to be made regarding which method(s) may be outperforming others on the large set of test problems. Figure 5.4 shows the ECDF for the performance of the eleven parameterizations across all test problems. As with the error ECDF's, the methods performing well are those that are concentrated on the left side of the graph. Figure 5.4 shows that there are three methods that have an ECDF curve outperforming the others. These methods are β, θ^2(unbounded), and θ^2_β. Figure 5.5 ignores the remaining parameterizations and focuses on just the top three just listed. Looking at the line for θ^2(unbounded) in Figure 5.5, we can see that it starts at a value of approximately 0.38 on the y-axis. This means that in (roughly) 38% of the test cases, the θ^2(unbounded) parameterization achieved the optimal (the minimum) log-likelihood value. Acknowledging that multiple parameterizations can achieve the optimal log-likelihood value, we can also note that θ^2_β achieved the optimal value approximately 35% of the test cases, and β, 25%.

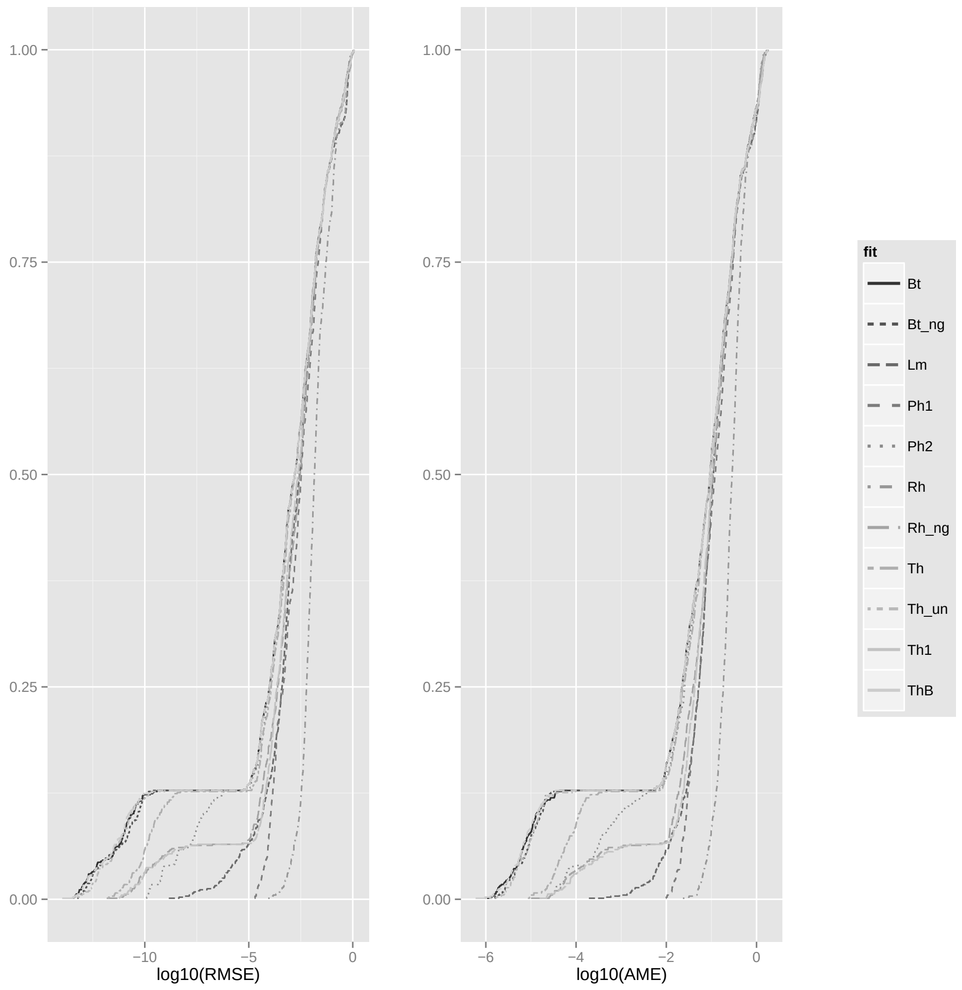

Figure 5.3: Empirical cumulative distribution functions for the logarithm (base ten) of the RMSE and AME values.

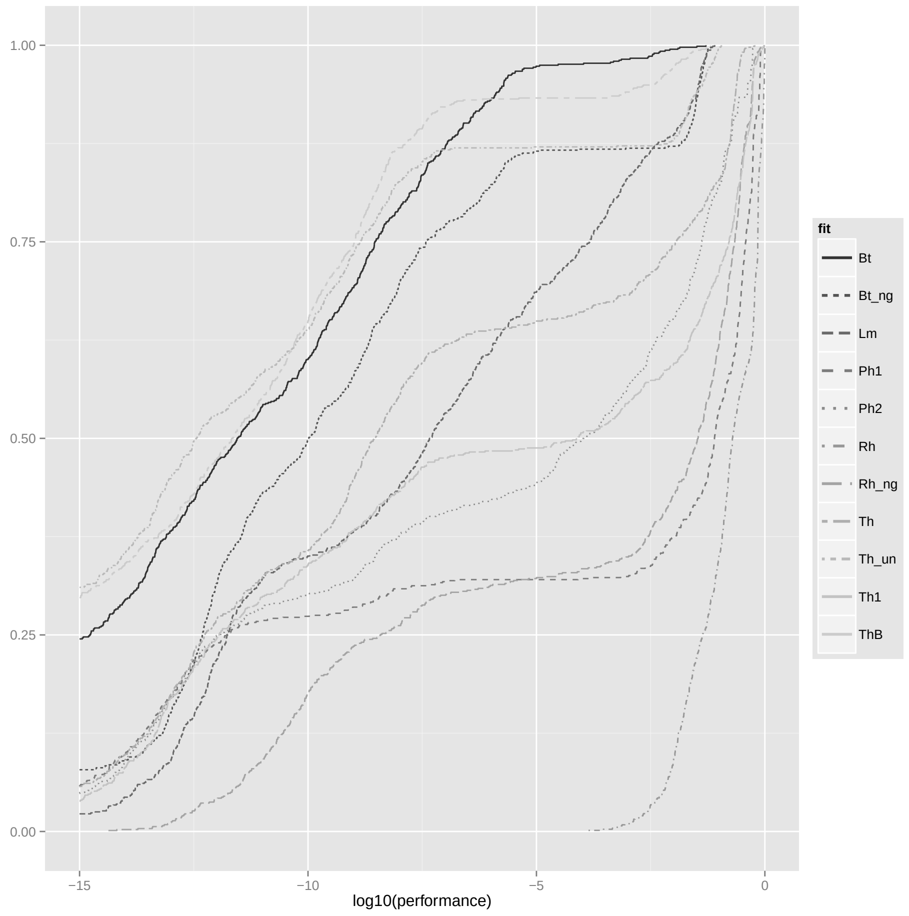

Figure 5.4: Empirical cumulative distribution functions for the performance of each parameterization.

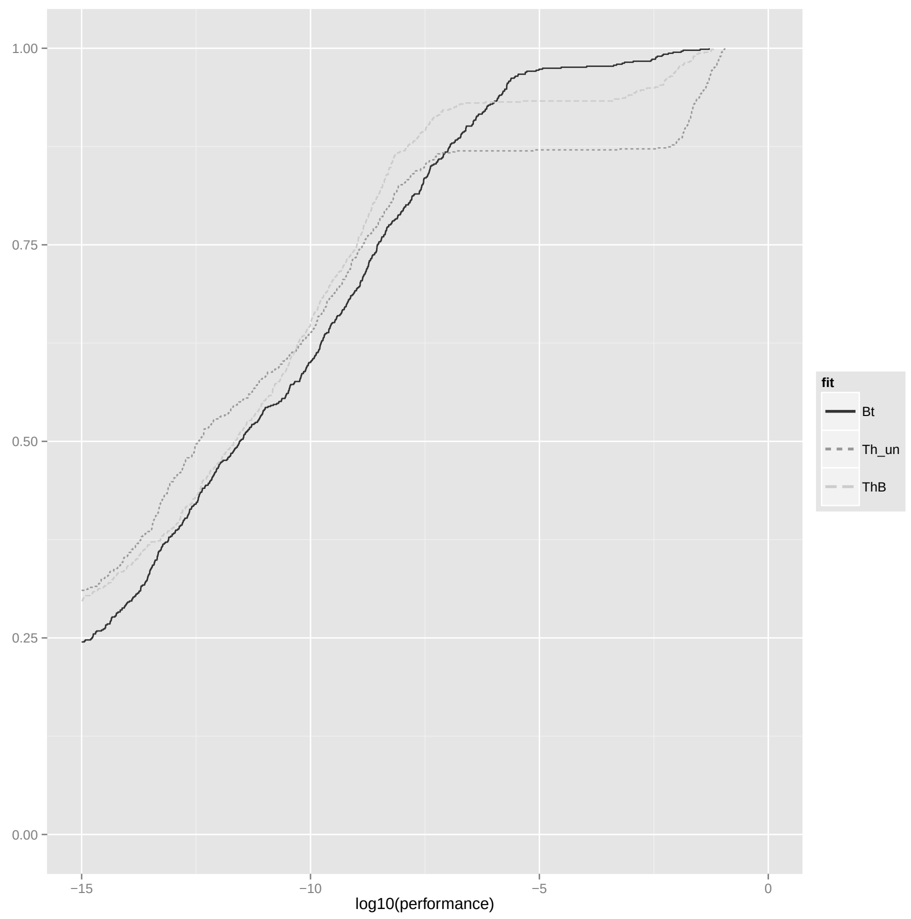

Figure 5.5: Empirical cumulative distribution functions for the top three parameterizations with respect to performance.

5.5 Results and Discussion

We have explored three different ways to analyze the large set of data produced by the full test simulation: the root mean squared error, absolute maximum error, and performance metric (calculated using the log-likelihood). Looking back at the empirical cumulative distribution function graphs of the RMSE and AME values in Figure 5.3, there are four parameterizations that fall to the left of the other seven and therefore have the highest proportion of small error values. These four are β, β(no gradient), θ^2(unbounded), and θ^2_β. The parameterizations that perform the worst, by falling on the right side of the graph, are ρ, ϕ(upper bound 5), and λ. These poor performing parameterizations are the parameterizations with the highest concentrations of large RMSE and AME values when compared with the rest.

In Markov chain Monte Carlo settings, [8] and [10] have had much success with the rho parameterization. However this is not true when numerically optimizing the likelihood. Rho performs quite poorly with the smallest (and best) error values being greater than a large portion of the error values in the four best parameterizations. It is interesting to revisit the ϕ parameterizations and note that the only difference between the two ϕ parameterizations is the upper bound values of 5 and 100, but the curves in the error ECDF's, and thus the solutions achieved, are not the same.

Figure 5.6 shows the ECDF's of just the top four parameterizations: β, β(no gradient), θ^2(unbounded), and θ^2_β. With the exception of the first initial increase from 0 to 0.125, the graphs are identical to one another. There is no apparent difference between the distributions of the errors for the four parameterizations.

With no obvious difference between the top four parameterizations in terms of errors, we need to delve deeper into the performance results based on the log-likelihood values in order to pick the top parameterization. Looking back at the best parameterizations (β, β(no gradient), θ^2(unbounded), and θ^2_β), we can see that out of the top four, three are the same as the top parameterizations found by analyzing Figure 5.5. The θ^2(unbounded) and θ^2_β parameterizations have very similar curves and perform better than the β parameterization until roughly the 0.87 and 0.93 marks on the y-axis respectively. The problematic area is the points where the β parameterization performs better than the $theta$(unbounded) and θ^2_β parameterizations (performance > -8). With the β curve being to the left of both of the other two, this means that the θ^2(unbounded) and θ^2_β parameterizations have a larger concentration of points at the higher end of performance values. Recall that

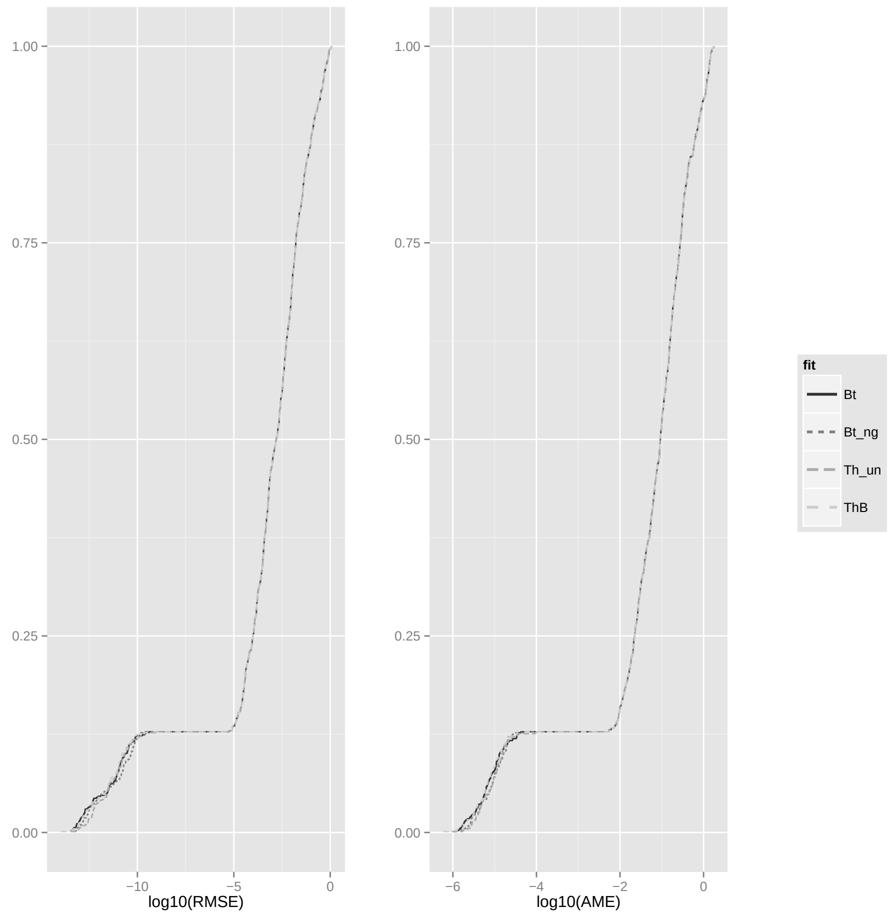

Figure 5.6: Empirical cumulative distribution functions of RMSE and AME for the top four parameterizations with respect to errors.

performance is defined as

$$L_p = \frac{(\ell_p - \ell^*)}{|\ell^*|},$$

and the larger the value, the further away the solution is from the best solution found. That is to say that the θ^2(unbounded) and θ^2_β parameterizations have a higher concentration of problems where they found solutions that were further away from the best solution found. In order to explore these values more we will look at only the observations from the three best parameterizations (β, θ^2(unbounded), and θ^2_β) where performance is greater than -8. Figure 5.7 shows the empirical cumulative distribution functions for the RMSE and AME values of the three parameterizations for the subset of data that fit the criterion listed above. Looking at the ECDF's for the errors, we can see that the β curve falls to the right (with higher values) of the other two. The θ^2(unbounded) and θ^2_β parameterizations have similar curves with the exception of the area between 0.5 and 0.75 on the y-axis. In that section θ^2(unbounded) has the upper hand. Overall, the two parameterizations have a similar performance, error wise.

The β parameterization is out of contention for the title of best parameterization because in this section of problematic performance values, it may have a better performance value than the other two parameterizations, but the errors produced are much larger than the other two parameterizations. This shows that the log-likelihood value might have been smaller, but the model itself performs poorly and produces estimates that are far from the true function values. Overall, we can say that the θ^2(unbounded) and θ^2_β parameterizations have the best performance, error wise, for the problematic section of high performance values.

We are aiming to pick the parameterization that performs the best over all of the problems presented in the full simulation. Currently, we are able to narrow down the list of contenders to two parameterizations whose performances are relatively similar in terms of both the log-likelihood values (performance) as well as the errors (root-mean-squared-error and absolute maximum error). These parameterizations are the θ^2(unbounded) and the θ^2_β parameterizations. The main differences (and downside) of the θ^2_β parameterization is the way in which the starting points of the parameter values are generated. The optimization was started at points equivalent to the parameter outputs of the finished optimization of the β run. Time wise the θ^2_β run requires that the β parameterization run first and then we can perform the θ^2_β parameterization, where as, the θ^2(unbounded) parameterization only requires one run in order to perform the optimization and fit

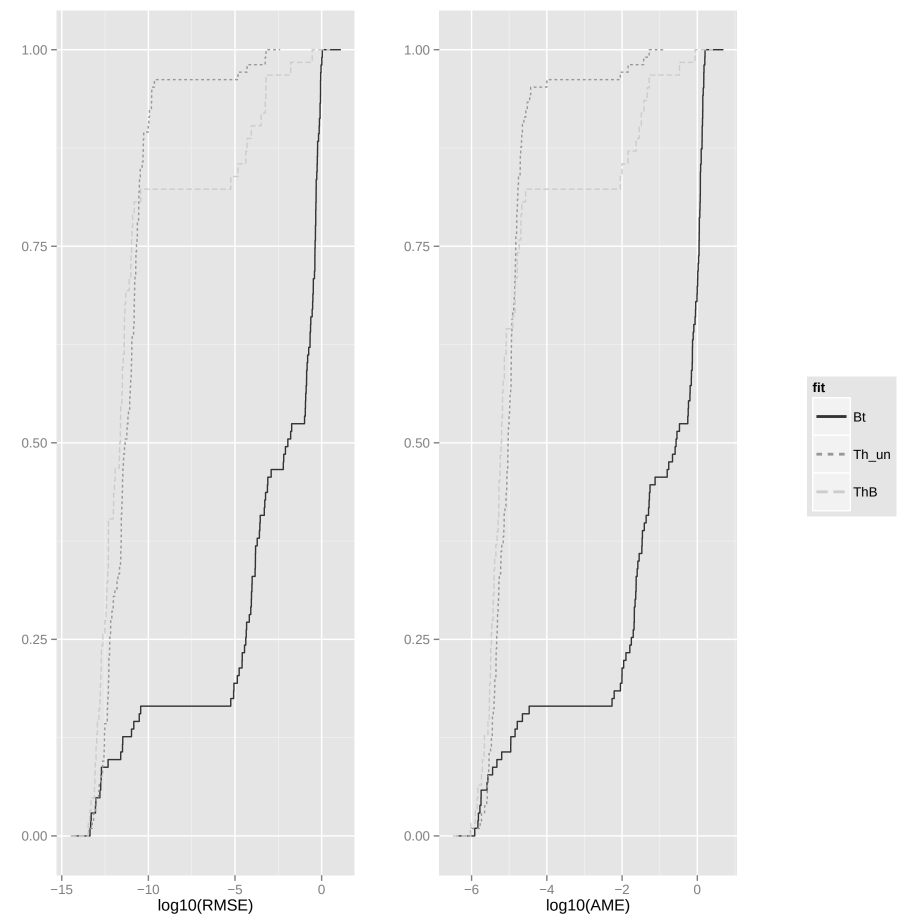

Figure 5.7: Empirical cumulative distribution functions of RMSE and AME for the top thee parameterizations (with respect to performance) where performance is greater than -8.

the full Gaussian process and achieve similar results. Combining what we know with the performance analysis and error analysis, with the amount of time required to run the parameterizations, the parameterization of choice for the best parameterization is the θ^2(unbounded) parameterization.

Chapter 6

Conclusions

In this study, we used the numerical procedure of fitting a Gaussian process for a large set of test problems. In fitting the Gaussian process, we used eleven different parameterization-optimization pairings in order to determine which parameterization performs the best so that it may be deemed the go-to parameterization.

Despite the likelihood function having the property of being invariant, numerically we have seen that this is not the case. Optimization produces different results even though the optimizer started with equivalent values.

We started by picking an appropriate design using the random latin hypercube design using $n = 7 \cdot d$, $10 \cdot d$, and $15 \cdot d$. We provided six different parameterizations that have all been discussed in literature. We then used the statistical programming language R to optimize the log-likelihood function in order to produce estimates for the parameter values for fitting the model.

The results of each of the eight hundred and forty tests performed provided us with optimal log-likelihood values, and the parameter values used to achieve the log-likelihood. We looked at three different tools in order to evaluate which of the parameterizations performed the best: root mean squared error, absolute maximum error, and performance as defined in Section 5.4.

We first visualized the distributions of the root mean squared errors and absolute maximum errors. Four parameterizations performed better than the rest in terms of errors produced in predicting: beta, beta (no gradient), theta2, and theta (beta). We further investigated the performance metric also using an empirical cumulative distribution function. From the ECDF's we outlined three parameterizations that outperformed the other seven: beta (no gradient), theta2, and theta (beta). Between the errors and performance we can see that the top three performing parametrizations are beta (no gradient), theta2 and theta (beta) as they achieve desirable log-likelihood values and produce estimates that have low errors. We analyzed these three further in order to determine which is the best.

The parameterizations had virtually identical error ECDF's and therefore the performance of the three was further analyzed. The three parameterizations have similar performance curves up until -8. The area of difference is in the right hand side of the graph where the solutions are far away from the best solution found. We further analyzed this problematic area by looking at the error values associated with the observations with larger performance values (greater than -8). The beta parameterization may perform better than the other two performance wise, but the models achieved in the Gaussian process produce poor estimates. The theta2 and theta (beta) parameterizations have similar problematic area error curves. The tie-breaker between the theta2 and theta (beta) curves is time, specifically the time required to fit Gaussian process and obtain a model. The theta2 parameterization requires only one run through the optimization process to fit and form the model whereas the theta (beta) parameterization requires one run through the process first for the beta parameterization and then again through the theta2 parameterization.

This thesis set out to find which parameterization of the correlation function in the Gaussian process performs the best. It was our wish to determine which parameterization should be the recommended parameterization for those wanting to fit a Gaussian process. Based on the evidence provided in this thesis the parameterization denoted as being the best is the parameterization in terms of θ^2 using the analytical gradient and the unbounded optimizer.

The performance of the parameterization rho is surprising. The rho parameterization has had much success in Markov chain Monte Carlo settings and further investigation into the parameter and why it performs poorly should be considered in the future.

Bibliography

[1] Arnold, A. and Loeppky, J. L. (2015), "The Problem with Assessing Statistical Methods," *arXiv preprint arXiv:1510.01417*, Submitted to The American Statistician. → pages 32

[2] Audet, C. and Dennis Jr, J. E. (2006), "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on optimization*, 17, 188–217. → pages 10

[3] Butler, A., Haynes, R., Humphries, T., and Ranjan, P. (2014), "Efficient optimization of the likelihood function in Gaussian process modelling," *Computational Statistics & Data Analysis*, 73, 40–52. → pages 1, 16

[4] Chen, H., Loeppky, J. L., Sacks, J., and Welch, W. J. (2016), "Analysis Methods for Computer Experiments: How to Assess and What Counts?" *Statistical Science*, 31, 40–60. → pages 5

[5] Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991), "Bayesian Prediction of Deterministic Functions, With Applications to the Design and Analysis of Computer Experiments," *Journal of the American Statistical Association*, 86, 953–963. → pages 1, 7

[6] Dennis, J. and Woods, D. J. (1987), "Optimization on microcomputers: The Nelder-Mead simplex Algorithm," *New computing environments: microcomputers in large-Scale computing*, 116–122. → pages 10

[7] Handcock, M. S. and Stein, M. L. (1993), "A Bayesian Analysis of Kriging," *Technometrics*, 35, 403–410. → pages 6

[8] Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008), "Computer Model Calibration Using High-Dimensional Output," *Journal of the American Statistical Association*, 103, 570–583. → pages 17, 38

[9] Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990), "Minimax and Maximin Distance Designs," *Journal of Statistical Planning and Inference*, 26, 131–148. → pages 15

[10] Linkletter, C., Bingham, D., Hengartner, N., Higdon, D., and Ye, K. Q. (2006), "Variable Selection for Gaussian Process Models in Computer Experiments," *Technometrics*, 48, 478–490. → pages 17, 38

[11] Loeppky, J. L., Sacks, J., and Welch, W. J. (2009), "Choosing the Sample Size of a Computer Experiment: A Practical Guide," *Technometrics*, 51, 366–376. → pages 29

[12] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239–245. → pages 13

[13] Morris, M. D. and Mitchell, T. J. (1995), "Exploratory Designs for Computational Experiments," *Journal of Statistical Planning and Inference*, 43, 381–402. → pages 13

[14] Mühlenbein, H., Schomisch, M., and Born, J. (1991), "The parallel genetic algorithm as function optimizer," *Parallel computing*, 17, 619–632. → pages 10

[15] O'Hagan, A. (1978), "Curve Fitting and Optimal Design for Prediction," *JRSSB*, 40, 1–42. → pages 4

[16] Ranjan, P., Haynes, R., and Karsten, R. (2011), "A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data," *Technometrics*, 53, 366–378. → pages 10, 16

[17] Rasmussen, C. E. and Williams, C. K. (2006), *Gaussian processes for machine learning*, vol. 1, MIT press Cambridge. → pages 4

[18] Rasmussen, C. E. and Williams, C. K. I. (2006), *Gaussian Processes for Machine Learning*, Cambridge, MA: The MIT Press. → pages 8

[19] Roustant, O., Ginsbourger, D., and Deville, Y. (2012), "DiceKriging, DiceOptim: Two R packages for the analysis of computer experimentRouGinDev2012s by kriging-based metamodeling and optimization," . → pages 8, 16

[20] Sacks, J., Schiller, S. B., and Welch, W. J. (1989), "Designs for Computer Experiments," *Technometrics*, 31, 41–47. → pages 1, 13

[21] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments (with Discussion)," *Statistical Science*, 4, 409–435. → pages 8

[22] Santner, T. J. ., Williams, B. J. ., and Notz, W. I. (2003), *The Design and Analysis of Computer Experiments*, New York: Springer. → pages 1

[23] Santner, T. J., Williams, B. J., and Notz, W. I. (2013), *The design and analysis of computer experiments*, Springer Science & Business Media. → pages 13

[24] Shanno, D. F. and Kettler, P. C. (1970), "Optimal Conditioning of Quasi-Newton Methods," *Mathematics of Computation*, 24, 657–664. → pages 10

[25] Stein, M. L. (1999), *Interpolation of Spatial Data, Some Theory for Kriging*, New York: Springer. → pages 8

[26] Surjanovic, S. and Bingham, D. (2015), "Virtual Library of Simulation Experiments:," . → pages 48

[27] Wang, J. M., Fleet, D. J., and Hertzmann, A. (2008), "Gaussian process dynamical models for human motion," *IEEE transactions on pattern analysis and machine intelligence*, 30, 283–298. → pages 8

[28] Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992), "Screening, Predicting, and Computer Experiments," *Technometrics*, 34, 15–25. → pages 7, 13

[29] Zehna, P. W. et al. (1966), "Invariance of maximum likelihood estimators," *Annals of Mathematical Statistics*, 37, 744. → pages 18

[30] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997), "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, 23, 550–560. → pages 10

Appendix

Appendix A

Test Functions

The fourteen functions used for the simulation study are from [26]. The functions are as follows:

Borehole function ($d = 8$)

$$f(\boldsymbol{x}) = \frac{2\pi x_3(x_4 - x_6)}{\ln(x_2/x_1)\left(1 + \frac{2x_7x_3}{\ln(x_2/x_1)x_1^2x_8}\right) + \frac{x_3}{x_5}}$$

$$\begin{aligned}
x_1 &\in [0.05, 0.15]\\
x_2 &\in [100, 50\ 000]\\
x_3 &\in [63\ 070, 115\ 600]\\
x_4 &\in [990, 1\ 110]\\
x_5 &\in [63.1, 116]\\
x_6 &\in [700, 820]\\
x_7 &\in [1\ 120, 1\ 680]\\
x_8 &\in [9\ 855, 12\ 045]
\end{aligned}$$

Branin function ($d = 2$)

$$f(\boldsymbol{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

$$\begin{aligned}
x_1 &\in [-5, 10]\\
x_2 &\in [0, 15]
\end{aligned}$$

Currin et al. (1988) Exponential function ($d = 2$)

$$f(\boldsymbol{x}) = \left(1 - \exp\left\{-\frac{1}{2x_2}\right\}\right) \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}$$

$$x_i \in [0,1]\ \forall i = 1,2$$

Dette and Pepelyshev (2010) 8-Dimensional function ($d = 8$)

$$f(\boldsymbol{x}) = 4(x_1 - 2 + 8x_2 - 8x_2^2)^2 + (3 - 4x_2)^2 + 16\sqrt{x_3 + 1}(2x_3 - 1)^2 + \sum_{i=4}^{8} i \ln\left(1 + \sum_{j=3}^{i} j\right)$$

$$x_i \in [0,1]\ \forall i = 1,\ldots,8$$

Dette and Pepelyshev (2010) Curved function ($d = 3$)

$$f(\boldsymbol{x}) = 4(x_1 - 2 + 8x_2 - 8x_2^2)^2 + (3 - 4x_2)^2 + 16\sqrt{x_3 + 1}(2x_3 - 1)^2$$

$$x_i \in [0,1]\ \forall i = 1,2,3$$

Dette and Pepelyshev (2010) Exponential function ($d = 3$)

$$f(\boldsymbol{x}) = 100\left(\exp\left\{\frac{-1}{x_1^{1.75}}\right\} + \exp\left\{\frac{-2}{x_2^{1.5}}\right\} + \exp\left\{\frac{-2}{x_3^{1.25}}\right\}\right)$$

$$x_i \in [0,1]\ \forall i = 1,2,3$$

Friedman function ($d = 5$)

$$f(\boldsymbol{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$$

$$x_i \in [0,1]\ \forall i = 1,\ldots,5$$

Gramacy and Lee (2009) ($d = 6$)

$$f(\boldsymbol{x}) = \exp\left\{\sin\left((0.9(x_1 + 0.48))^{10}\right)\right\} + x_2 x_3 + x_4$$

x_5 and x_6 are inactive.A small error term $\varepsilon(0, 0.05^2)$ is added.

$$x_i \in [0,1]\ \forall i = 1,\ldots,6$$

Lim et al. (2002) Nonpolynomial function ($d = 2$)

$$f(\boldsymbol{x}) = \frac{1}{6}\left[(30 + 5x_1 \sin(5x_1))(4 + \exp\{-5x_2\}) - 100\right]$$

$$x_i \in [0,1] \; \forall i = 1,2$$

Linkletter et al. (2006) Decreasing Coefficients function ($d = 10$)

$$f(\boldsymbol{x}) = 0.2x_1 + \frac{0.2}{2}x_2 + \frac{0.2}{4}x_3 + \frac{0.2}{8}x_4 + \frac{0.2}{16}x_5 + \frac{0.2}{32}x_6 + \frac{0.2}{64}x_7 + \frac{0.2}{128}x_8$$

x_9 and x_{10} are inactive. A small error term $\varepsilon(0, 0.05^2)$ is added.

$$x_i \in [0,1] \; \forall i = 1, \ldots, 10$$

Linkletter et al. (2006) Sinusoidal function ($d = 10$)

$$f(\boldsymbol{x}) = \sin(x_1) + \sin(5x_2)$$

$x_3, \ldots, x_{10}$ are inactive. A small error term $\varepsilon(0, 0.05^2)$ is added.

$$x_i \in [0,1] \; \forall i = 1, \ldots, 10$$

OTL Circuit function ($d = 6$)

$$f(\boldsymbol{x}) = \frac{\left(\frac{12x_2}{x_1+x_2} + 0.74\right)x_6(x_5+9)}{x_6(x_5+9)+x_3} + \frac{11.35x_3}{x_6(x_5+9)+x_3} + \frac{0.74x_3x_6(x_5+9)}{(x_6(x_5+9)+x_3)x_4}$$

$$\begin{aligned} x_1 &\in [50, 150] \\ x_2 &\in [25, 70] \\ x_3 &\in [0.5, 3] \\ x_4 &\in [1.2, 2.5] \\ x_5 &\in [0.25, 1.2] \\ x_6 &\in [50, 300] \end{aligned}$$

Piston Simulation function ($d = 7$)

$$f(\boldsymbol{x}) = 2\pi\sqrt{\frac{x_1}{x_4 + x_2^2 \frac{x_5 x_3}{x_7} \frac{x_6}{V^2}}},$$

where

$$V = \frac{x_2}{2x_4}\left(\sqrt{A^2 + 4x_4\frac{x_5x_3}{x_7}x_6} - A\right)$$

and

$$A = x_5x_2 + 19.62x_1 - \frac{x_4x_3}{x_2}$$

$$\begin{aligned} x_1 &\in [30, 60] \\ x_2 &\in [0.005, 0.020] \\ x_3 &\in [0.002, 0.010] \\ x_4 &\in [1\ 000, 5\ 000] \\ x_5 &\in [90\ 000, 110\ 000] \\ x_6 &\in [290, 296] \\ x_7 &\in [340, 360] \end{aligned}$$

Wing Weight function ($d = 10$)

$$f(\boldsymbol{x}) = 0.036x_1^{0.758}x_2^{0.0035}\left(\frac{x_3}{\cos^2(x_4)}\right)^{0.6}x_5^{0.006}x_6^{0.04}\left(\frac{100x_7}{\cos(x_4)}\right)^{-0.3}(x_8x_9)^{0.49}+x_1x_{10}$$

$$\begin{aligned} x_1 &\in [150, 200] \\ x_2 &\in [220, 300] \\ x_3 &\in [6, 10] \\ x_4 &\in [-10, 10] \\ x_5 &\in [16, 45] \\ x_6 &\in [0.5, 1] \\ x_7 &\in [0.08, 0.18] \\ x_8 &\in [2.5, 6] \\ x_9 &\in [1\ 700, 2\ 500] \\ x_10 &\in [0.025, 0.08] \end{aligned}$$