NON-LINEAR DYNAMICS IDENTIFICATION USING GAUSSIAN PROCESS PRIOR MODELS WITHIN A BAYESIAN CONTEXT

By

Keith Neo Kian Seng

PhD Thesis

Submitted to
National University of Ireland, Maynooth (NUIM)
Department of Electronic Engineering

Hamilton Institute
National University of Ireland, Maynooth
Maynooth, Co. Kildare
Ireland

2008

Research Supervisor: Professor W. E. Leithead

Preface

This dissertation is prepared at the Hamilton Institute, National University of Ireland, Maynooth (NUIM) of Ireland, in partial fulfilment and conformity with the requirements for the degree of Doctor of Philosophy (Ph.D.) in Electronic Engineering.

The dissertation describes the work carried out between October 2003 and February 2007, under the supervision of Professor Bill Leithead. With the exception of Section 3.2.1 and Section 3.5.1, which has been done in collaboration with Yunong Zhang and where explicit reference is made to the work of others, this dissertation is the result of my own work. It has not, nor has any similar dissertation, been submitted for a degree or any qualification at this or any other university. This length of this dissertation does not exceed sixty thousand in words.

December 2007

Keith Neo Kian Seng

Abstract

Gaussian process prior models are known to be a powerful non-parametric tool for stochastic data modelling. It employs the methodology of Bayesian inference in using evidence or data to modify or refer some prior belief. Within the Bayesian context, inference can be used for several purposes, such as data analysis, filtering, data mining, signal processing, pattern recognition and statistics. In spite of the growing popularity of stochastic data modelling in several areas, such as machine learning and mathematical physics, it remains generally unexplored within the realm of nonlinear dynamic systems, where parametric methods are much more mature and more widely accepted.

This thesis seeks to explore diverse aspects of mathematical modelling of nonlinear dynamic systems using Gaussian process prior models, a simple yet powerful stochastic approach to modelling. The focus of the research is on the application of non-parametric stochastic models to identify nonlinear dynamic systems for engineering applications, especially where data is inevitably corrupted with measurement noise. The development of appropriate Gaussian process prior models, including various choices of classes of covariance functions, is also described in detail.

Despite its good predictive nature, Gaussian regression is often limited by several $O(N^3)$ operations and $O(N^2)$ memory requirements during optimisation and prediction. Several fast and memory efficient methods, including modification of the log-likelihood function and hyperparameter initialisation procedure to speed up computations, are explored. In addition, fast algorithms based on the generalised Schur algorithm are developed to allow Gaussian process to handle large-scale time-series datasets.

Models based on multiple independent Gaussian processes are explored in the thesis. These can be split into two main sections, with common explanatory variable and with different explanatory variables. The two approaches are based on different philosophies and theoretical developments. The benefit of having these models is to allow independent components with unique characteristics to be identified and extracted from the data.

The above work is applied to a real physical wind turbine data, consisting of 24,000 points of the wind speed, rotor speed and the blade pitch angle measurement data. A case study is presented to demonstrate the utility of Gaussian regression and encourage further application to the identification of nonlinear dynamic systems.

Finally, a novel method using a compound covariance matrix to exploit both the time-series and state-space aspects of the data is developed. This is referred to as the state-space time-series Gaussian process. The purpose of this approach is to enable Gaussian regression to be applied on nonlinear dynamic state-space datasets with large number of data points, within an engineering context.

Acknowledgements

Many people have contributed in one way or another to the thesis. The one that makes a difference has to be Professor Bill Leithead, my supervisor. His constant support, patience and encouragement are always there. In addition, I sincerely appreciate Professor Douglas Leith, for accepting me to do my research at Hamilton Institute. There are several people whom I wish to mention. Stuart put me up at his house for the first couple of days since my arrival in Glasgow. While in Scotland, it gives me the opportunity to meet Mary, Sergio and Ilias, whom I have become very good friends with.

I met Tianji and Dr. Emanuele Ragnoli when I came over to Ireland. In addition, there is also Parisa, who happens to be my only office "room-mate", and has to put up with me at times. I have to give special thanks to Rosemary and Kate for the wonderful support over the past few years.

In terms of work, I am deeply grateful to Dr. Yunong Zhang for his help, rendering constructive suggestions, comments and feedback. There is also Dr. Roderick Murray-Smith who will not hesitate to share his innovative ideas anytime.

I am deeply grateful to my parents for their support and encouragement during my time abroad. Furthermore, I wish to express my deepest, sincere appreciation to my wife, Daphne, for having walked through with me.

I feel that I have matured a lot more in terms of personality, mentality, calmness and awareness during the past few years. I have changed my understandings and views towards certain things. At the end of the day, I feel that I have benefited a lot and am glad that I have chosen to walk this path.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Anyone who has never made a mistake has never tried anything new."
- Albert Einstein (1879 – 1955)

1.1 Background

Data modelling is encountered in several fields of research. When presented with an unknown set of data obtained from a system, it is of interest to analyse the measurements (or observations) and identify the system, which can be nonlinear in many cases. For example, it might be the future shares fluctuation in the stock market that investors are trying to predict, the dynamics of a wind turbine machine that engineers are interested in simulating, spam filtering using the Bayesian learning methodology, or the action-reaction relationship between genes which biologists and scientists are attempting to identify. With several unknown underlying factors that remain unclear to investigators, deterministic modelling is usually not preferred. Alternatively, stochastic modelling which provides a probabilistic description of the model given the data is a better alternative for modelling stochastic data. The model can be improved by the incorporation of prior knowledge about the data (system). It (the model) is represented by the prior beliefs about the system and the information about the system as provided by the data. This model can then be used to make

inferences using the rules of probability theory, within a Bayesian context. Gaussian process prior model is, therefore, one of the keys to successful implementation of Bayesian methods in nonlinear dynamic systems. The purpose of using Gaussian processes is not limited to just a mathematical breakthrough, but is also applicable to solving real world problems.

It is perhaps of interest to go back in time a little further. The starting point was Bernoulli's work (Bernoulli, 1713) on Binomial distributions and the relationship of uncertainty to probability. Half a century later, Bayes followed up on Bernoulli's work and developed a mathematical structure (Bayes, 1763) that was previously lacking. With that structure, inferences can be made using models belonging to a distribution. However, this was only applicable to Binomial distribution at that time. It was not until another 50 years later that Laplace extended Bayes' theorem to include all possible distributions (Laplace, 1812). Unfortunately, mathematicians at that time were not keen on probabilistic modelling, which was often opposed by objective frequentist views.

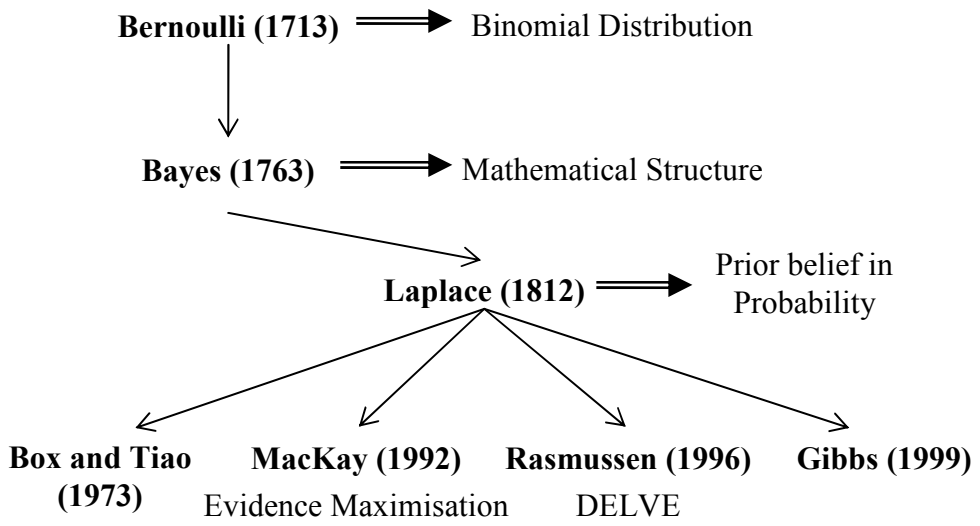


Figure 1 History of Bayesian Inference

Since then, Bayesian inference was largely ignored. It was only in recent decades that interest began to grow, particularly in the area of stochastic modelling (Lu and Adachi, 1989; Berman, 1990; Archambeau et al., 2007). Based on the work by Laplace, several works were developed in the late 1990s. Figure 1 presents a visual

timeline of the development of Bayesian inference. Beginning in the late 1990s, the interest in Gaussian processes has grown rapidly, with the aim of using them as models for regression (Williams and Rasmussen, 1996) and classification (Williams and Barber, 1998). Despite its growing popularity, dynamic modelling using Gaussian processes is still considered to be at its infancy (Murray-Smith et al., 1999; Kocijan et al., 2003). One of the many reasons is because there are several underlying constraints restricting the use of probabilistic modelling in dynamic systems. Data from nonlinear dynamic systems have particular features unlike many others. Hence, deterministic parametric modelling is preferred more within the engineering communities. This thesis aims to explore the area of identification of nonlinear dynamic systems, and handling large-scale measurement dataset using Gaussian process prior models.

1.2 Outline

The underlying motivation for the work reported in this thesis is the development of practical Gaussian regression based approaches to the identification of nonlinear dynamic systems from noisy measurements. Essentially, the question addressed is "What Gaussian process prior model should be used when applying Gaussian regression to identifying nonlinear dynamic systems?". In addition, related improvements to the Gaussian regression algorithms are developed.

In Chapter 2, the Gaussian process methodology is discussed, starting with the introduction of Gaussian random functions and joint probability distributions. The prior model is defined in detail, including the various classes of covariance functions that are used to parameterise the prior model.

Although Gaussian regression is an effective tool, fast algorithms are required to overcome the expensive computational demands and requirements. The use of Hessian information in the optimisation routine to improve convergence, modification of the log-likelihood function to reduce the number of hyperparameters to be trained and development of fast algorithms, which are capable of handling large-scale

datasets of size up to one million data points and beyond, are all relevant. These improvements to the algorithms are investigated in Chapter 3.

When the nonlinear relationship underlying measured data consists of two or more independent components, it may be required to extract one or both of the components. A Gaussian regression approach to so doing based on multiple Gaussian process models is proposed. Two cases are considered. In the first case, the independent components have different explanatory variables. In the second case, the independent components have same explanatory variable. These are discussed in Chapter 4.

Chapters 3 and 4 present improvements to the Gaussian regression algorithms. A case study illustrating the concepts and methods developed is presented in Chapter 5. The data are noisy site measurements for a commercial wind turbine machine, consisting of rotor speed, blade pitch angle and the nacelle anemometer measurement of wind speed.

In Chapter 6, the Gaussian process prior model, to be used when applying Gaussian regression to identify nonlinear dynamic systems, is investigated. In this context, the data set is typically obtained as a time-series but the underlying nonlinear relationship is dependent on some explanatory variable other than time. A prior model based on a pair of independent Gaussian processes that caters for this dual nature of the data is proposed. The dual nature model of Chapter 6 enables the time-series aspect of the data to be exploited by pre-filtering, particularly, when combined with the multiple Gaussian process prior models of Chapter 4. In combination with the fast and efficient algorithms of Chapter 3, it enables greatly increased data sets to be used.

In Chapter 7, the application of the methods and algorithms developed in the preceding chapters to nonlinear dynamic system identification is discussed and conclusions are drawn.

1.3 Achievements

This thesis describes the work carried out between October 2003 and February 2007, under the supervision of Professor Bill Leithead. With the exception of Section 3.2.1 and Section 3.5.1 that were done in collaboration with Yunong Zhang and where explicit references are made to include the work of others, this thesis is the result of my own work. Some published works include "Gaussian regression based on models with two stochastic processes", presented to IFAC 2005, "Wind turbine rotor acceleration: identification using Gaussian regression", published in ICINCO 2005 and "Multi-frequency scale Gaussian regression for noisy time-series data", which was submitted to UKACC 2006.

The major achievements of this thesis are the following: fast and efficient algorithms are developed for a class of Gaussian process prior models; a novel multiple Gaussian process prior model is developed for extracting and identifying components with different characteristics; a dual nature Gaussian process prior model for use when applying Gaussian regression to nonlinear dynamic system identification is developed.

Chapter 2

Gaussian Process Models

2.1 Brief Introduction

In this chapter, Gaussian process modelling within a Bayesian context is introduced. Gaussian process models have some similarities to certain classes of Artificial Neural Networks (ANNs). The Radial Basis Function (RBF), a specific class of ANNs, becomes a Gaussian process when the number of nodes in the feature-vector or weighting tends to infinity. This places Gaussian process modelling very firmly within the scope of machine learning, though it is more commonly encountered in the field of statistical research. Early work concerned with Gaussian process models includes O'Hagan (1978 and 1994), but it did not spark general interest. However, from the late 1990s, following publications by Mackay (1998) and Williams (1999), interest quickly grew in the application of Gaussian process models to data analysis (Gibbs and Mackay, 2000; Sambu et al., 2000; Yoshioka and Ishii, 2001).

2.2 Random Variables – Joint, Marginal and Conditional Probability

Gaussian process models and their applications have started to change many perspectives and challenge many traditional concepts. Before any further discussion on Gaussian process models, it is appropriate to review marginal probabilities, conditional probabilities and joint probability distributions.

Consider $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$, a finite set of continuous random stochastic variables, $\mathbf{x}_i \in \mathrm{P}^p, 1 \le i \le N$, and assume that they are described by a joint probability distribution $p(\mathbf{X})$. Let $\mathbf{X}_\mathrm{A}$ and $\mathbf{X}_\mathrm{B}$ be two subsets of $\mathbf{X}$ such that $\mathbf{X}_\mathrm{A} \cap \mathbf{X}_\mathrm{B} = \phi$ and $\mathbf{X}_\mathrm{A} \cup \mathbf{X}_\mathrm{B} = \mathbf{X}$. It follows that the *marginal probability* of $\mathbf{X}_\mathrm{A}$ is

$$p(\mathbf{X}_\mathrm{A}) = \int p(\mathbf{X}_\mathrm{A}, \mathbf{X}_\mathrm{B}) d\mathbf{X}_\mathrm{B}$$

The discrete case of the marginal probability of $\mathbf{X}_\mathrm{A}$ is obtained by replacing the integral with a sum. If the set $\mathbf{X}_\mathrm{A}$ contains more than one variable, then the marginal probability itself is a joint probability. The joint distribution for $\mathbf{X}$ is equal to the product of the marginals provided that $\mathbf{X}_\mathrm{A}$ and $\mathbf{X}_\mathrm{B}$ are independent. However, it is always assumed in this thesis that the variables are not necessarily independent.

The conditional probability distribution of $\mathbf{X}_\mathrm{A}$ given $\mathbf{X}_\mathrm{B}$ is defined as

$$p(\mathbf{X}_\mathrm{A} \mid \mathbf{X}_\mathrm{B}) = \frac{p(\mathbf{X}_\mathrm{A}, \mathbf{X}_\mathrm{B})}{p(\mathbf{X}_\mathrm{B})}$$

for $p(\mathbf{X}_\mathrm{B}) > 0$, where $p(\mathbf{X}_\mathrm{B})$ is known as the normalising probability. If $\mathbf{X}_\mathrm{A}$ and $\mathbf{X}_\mathrm{B}$ are independent, then the marginal probability $p(\mathbf{X}_\mathrm{A})$ and the conditional probability $p(\mathbf{X}_\mathrm{A} \mid \mathbf{X}_\mathrm{B})$ are equal.

Given the conditional probabilities $p(\mathbf{X}_\mathrm{A} \mid \mathbf{X}_\mathrm{B})$ and $p(\mathbf{X}_\mathrm{B} \mid \mathbf{X}_\mathrm{A})$, it follows from Bayes' theorem that

$$p(\mathbf{X}_\mathrm{A} \mid \mathbf{X}_\mathrm{B}) = \frac{p(\mathbf{X}_\mathrm{A}) p(\mathbf{X}_\mathrm{B} \mid \mathbf{X}_\mathrm{A})}{p(\mathbf{X}_\mathrm{B})}$$

It is easy to use the above theorem and formulations to perform further conditioning on other variables.

2.3 Gaussian Random Functions

The Gaussian function is one of the simplest of all possible random functions. Its random nature is fully characterised by its mean function and covariance function (Pugachev, 1967; Papoulis, 1991). Depending on the explanatory variable of the Gaussian random functions, it is usually known as a Gaussian stochastic process if the argument is time domain; or a Gaussian random field if it represents a state belonging to some space.

Consider the following stochastic field, $f(\mathbf{z})$, $\forall \mathbf{z} \subseteq \mathrm{P}^{D}$, with mean function, $m(\mathbf{z}) = \mathrm{E}[\mathrm{f}(\mathbf{z})]$ as $\mathbf{z}$ varies, and covariance function, $C(\mathbf{z}, \mathbf{z}') = \mathrm{cov}[\mathrm{f}(\mathbf{z}), \mathrm{f}(\mathbf{z}')] = \mathrm{E}[\mathrm{f}(\mathbf{z})\mathrm{f}(\mathbf{z}')] - m(\mathbf{z})m(\mathbf{z}')$ as $\mathbf{z}$ and $\mathbf{z}'$ vary.

The stochastic Gaussian process $f(\mathbf{z})$ can be denoted by

$$f(\mathbf{z}) \sim \Gamma\Pi(m(\mathbf{z}), C(\mathbf{z}, \mathbf{z}'))$$

A Gaussian process can also be thought of as a generalisation of multivariate Gaussian random variables to infinite sets. When a stochastic process is Gaussian, all the joint probability distributions are multivariate normal. Therefore, for any given set of explanatory variables, $\mathbf{Z}_N \equiv \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, the corresponding random variables $\mathbf{F}_N \equiv \{\mathrm{f}(\mathbf{z}_1), \ldots, \mathrm{f}(\mathbf{z}_n)\}$ have a n-dimensional normal distribution

$$p(\mathrm{f}(\mathbf{z}_1), \ldots, \mathrm{f}(\mathbf{z}_n) \mid \mathbf{z}_1, \ldots, \mathbf{z}_n) \sim \mathrm{N}(\mathbf{m}, \Sigma)$$

where $\mathbf{m}$ is a n x 1 vector of the mean values and Σ is a n x n covariance matrix between all points of the input explanatory variable,

$$\Sigma = \begin{bmatrix} C(\mathbf{z}_1, \mathbf{z}_1) & \cdots & C(\mathbf{z}_1, \mathbf{z}_n) \\ \vdots & \ddots & \vdots \\ C(\mathbf{z}_n, \mathbf{z}_1) & \cdots & C(\mathbf{z}_n, \mathbf{z}_n) \end{bmatrix}$$

i.e. $\Sigma_{ij} = C(\mathbf{z}_i, \mathbf{z}_j)$.

2.4 Regression Using Gaussian Processes

Regression is one of the most common data modelling problems and several methods exist to handle it. The Bayesian approach to regression is discussed with the emphasis on the application to Gaussian process modelling. The Gaussian process prior model and posterior model are introduced.

2.4.1 Gaussian Process Model

What is a Gaussian process prior model? A Gaussian function is a stochastic process, whereas a Gaussian process model is a mathematical model for a nonlinear relationship, f(**z**), which depends on some explanatory variable, **z**. In a random function model, any particular nonlinear relationship is one realisation of the random function, or more precisely, the model for all possible nonlinear relationships is simply the class of realisations for the random function. This model places a probability distribution over the set of all possible relationships; to be precise, for any finite set of values for the explanatory variable, [$\mathbf{z}_1$,…,$\mathbf{z}_N$], the joint probability distribution for [f($\mathbf{z}_1$),…,f($\mathbf{z}_N$)] is specified. The random function is chosen to be Gaussian, thus defining the Gaussian process model for the nonlinear relationship. It is completely specified by its mean function, $m(\mathbf{z})$, and its covariance function, $C(\mathbf{z}, \mathbf{z}')$, see §2.3.

It is important to note that a Gaussian process model is a non-parametric model. In standard models, the function is explicitly parameterised. An example of a parametric model is the class of linear functions, $y = \mathtt{a}z + \mathtt{b}$ with the values of `a` and `b` belonging to some set. In Gaussian process prior models, a probability distribution is placed on the space of all possible functions. In the former, only a restricted class of functions dependent on the parameterisation is possible, whereas all functions are possible in the latter, but not with equal probabilities.

In Gaussian regression, a Gaussian process model with a particular choice of the Gaussian field or process is selected; this is the *Gaussian process prior model*. (The selection process may involve a specification of a class of mean functions and covariance functions, parameterised by some set of hyperparameters with the values of the hyperparameters being subsequently set. This selection is informed by any prior knowledge relevant to the nature of the nonlinear relationship, i.e. general features, such as periodicity, and specific features, such as appropriate lengthscales, etc.). The Gaussian process prior model is then conditioned on data to obtain the posterior model. The posterior model remains a Gaussian field or a Gaussian process. The model of all possible nonlinear relationships conditioned on the data is the class of realisations for the *Gaussian process posterior model*. It places a modified probability distribution over the set of all possible relationships, to be precise, the joint probability distributions for $[\mathrm{f}(\mathbf{z}'_1),...,\mathrm{f}(\mathbf{z}'_{N'})]$, for any finite set of values for the explanatory variable $[\mathbf{z}'_1,...,\mathbf{z}'_{N'}]$, for any N'. The mean of the posterior model as a function is interpreted to be the best fit to the data. The confidence interval as a function of $\mathbf{z}$, defined to be twice the standard deviation, is used to express the uncertainty of the fit to the data.

Gaussian process models are widely used today in regression and classification problems, ranging from data analysis to applications in system identification. A simple example is presented below to illustrate how Gaussian process prior models can be used in a regression problem with a one-dimensional explanatory variable.

Example 2.1 *(Simple Regression in One-Dimension)*. Consider a simple regression problem for a one-dimensional explanatory variable, denoted by z. In Figure 2, four sample functions are drawn at random from the *prior* distribution over functions specified by a particular Gaussian process. It is observed that the functions are rather smooth in nature reflecting the choice of prior to represent prior belief about the nature of the underlying relationship. Note that, it is assumed that the prior mean, at any particular value of z is zero. The means at any fixed z over the functions depicted in Figure 2 are not particularly close to zero. However, the values of the mean of f(z) for any fixed z tend to zero as more sample functions are included. The two times

standard deviations at any value of z of the sample functions are also illustrated in the figure by dashed lines. As the prior variance of the Gaussian process does not depend on z, the standard deviation does not change as z varies.

Suppose that a data set, $\Delta = \{(z_1, \mathrm{f}(z_1)), (z_2, \mathrm{f}(z_2))\}$ consisting of two observations $(z_1, \mathrm{f}(z_1)) = (-1.0, 1.5)$ and $(z_2, \mathrm{f}(z_2)) = (2.5, -1.0)$, is given. It is required to consider only functions that pass through these two data points exactly. The grey lines shown in Figure 3 are sample functions drawn from the posterior distribution over the functions. Based on all possible realisations of the Gaussian process prior model, the prediction of the mean values of the posterior distribution is portrayed in bold. The confidence intervals indicating the uncertainty of this mean prediction is also shown. Notice that the uncertainty reduces when it is close to the two data points, and enlarges as it moves away from the observations. Likewise, if more data points are included in the dataset, the uncertainty close to these data points is also reduced.

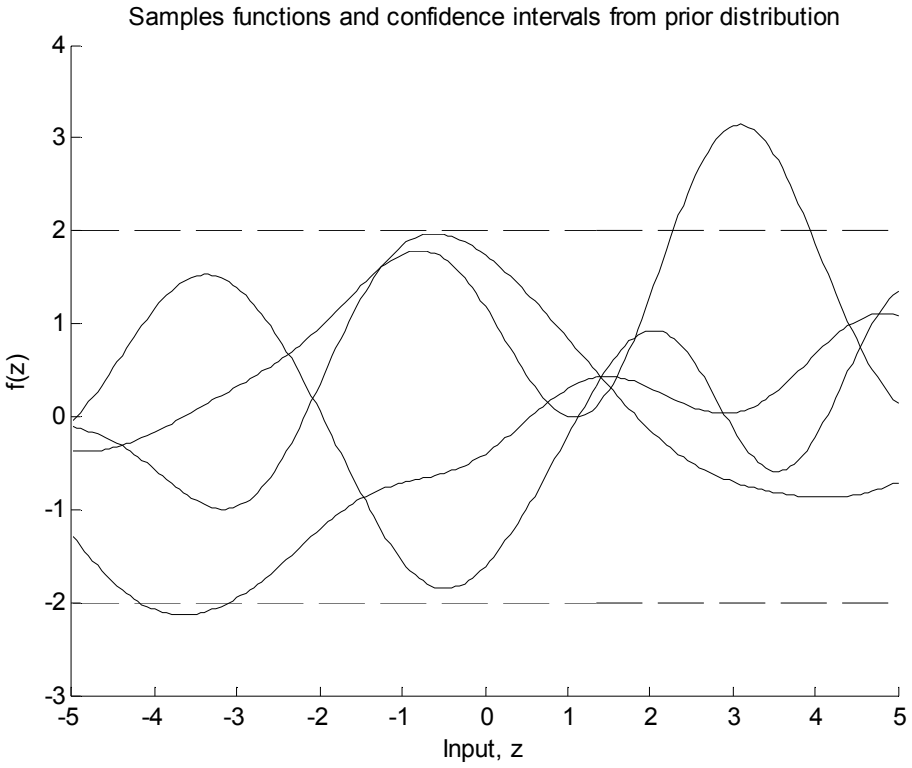


Figure 2 Four samples are drawn from the prior distribution. The dashed lines indicate the two times standard deviations.

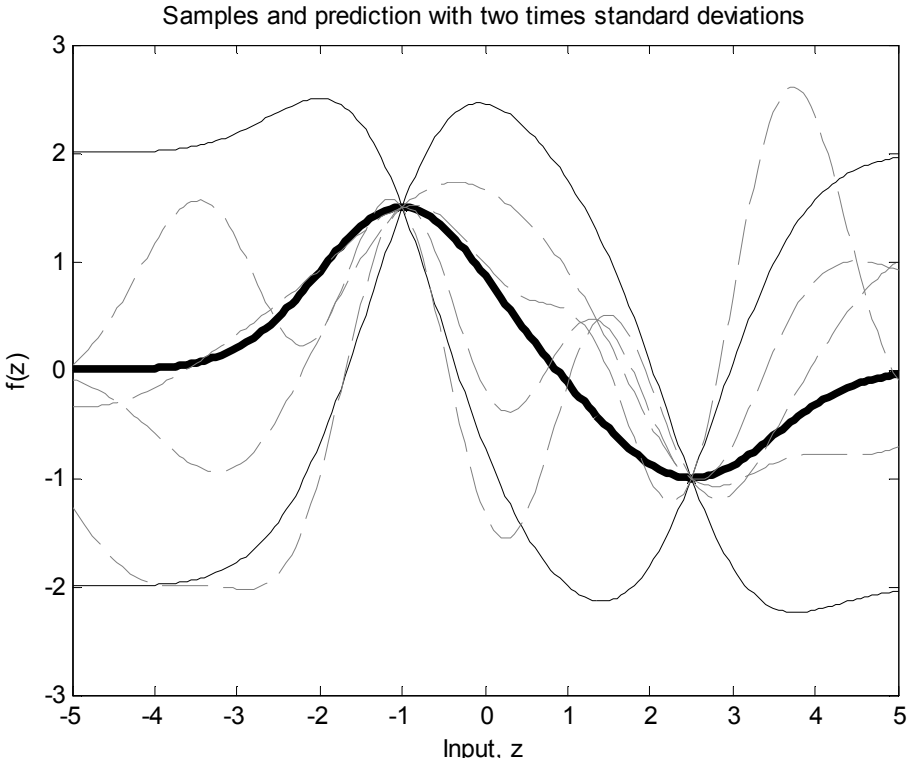


Figure 3 Mean prediction (in bold) with two times standard deviations (black). Four samples (grey) from the posterior are shown as dashed lines.

Let the set of values of the explanatory variable be denoted by $\mathbf{Z} \equiv \{\mathbf{z}_i\}_{i=1}^{N}$ and the set of function values corresponding to $\mathbf{Z}$ be denoted by $\mathbf{F} \equiv \{\mathrm{f}(\mathbf{z}_i)\}_{i=1}^{N}$. For both the Gaussian process prior model and posterior model, the joint probability distribution for $\mathbf{F}$ is

$$p(\mathbf{F} \mid \mathbf{Z}) = \frac{1}{Z} \exp\left[-\frac{1}{2}(\mathbf{F}-\mu)^{\mathrm{T}} Q^{-1}(\mathbf{F}-\mu)\right] \sim \mathrm{N}(\mu, Q) \tag{1}$$

where Q is the covariance matrix and μ is the mean vector. In this thesis, the Gaussian process prior model is generally zero-mean, assuming no prior information is available to contradict this hypothesis. It follows that the probability distribution becomes

$$p(\mathbf{F} \mid \mathbf{Z}) \sim \mathrm{N}(0, \Sigma)$$

Non-zero mean prior processes have also been covered in the literature (Pugachev, 1967; O'Hagan, 1978; Cressie, 1993).

Why use Gaussian regression? In the literature, several regression techniques have been developed, e.g. artificial neural network (ANN) and automatic relevance determination (ARD). With a rich pool of techniques available, it almost seems that this area of interest has matured. Gaussian regression was used in the early 1960s, but was confined mainly to the statistics community. Their general-purpose capability was neglected with only a narrow range of applications. It is only during the 1990s that Gaussian regression started to arouse wider interest.

2.4.2 Bayesian Regression

In Bayesian regression, a nonlinear relationship, f(**z**), is assumed to underlie some measured dataset, $\Delta = \{(\mathbf{z}_n, y_n)\}_{n=1}^{N}$, where $\mathbf{z}_n$ denotes the values of the explanatory variable of dimension D (covariates) and y_n denotes the scalar measured value (target). Given this set of N observations, it is of interest to infer the nonlinear relationship, f(**z**). Subsequently, predictions of the function value for new values of the explanatory variable can be made. The measured values are assumed to be corrupted by measurement noise. A variety of noise models has been investigated (MacKay, 1997; Gibbs, 1997; Goldberg et al., 1998; Murray-Smith and Girard, 2001) over the last decade or so. However, attention is focused here on additive Gaussian white noise, which is statistically independent and identically distributed across the observation data. The corresponding relationship of measurement data to noise is denoted by

$$y_i = \mathrm{f}(\mathbf{z}_i) + \varepsilon_i \qquad \forall i = 1,\ldots,N \tag{2}$$

where $\mathrm{f}(\mathbf{z}_i)$ is the noise-free value at $\mathbf{z}_i$ and the noise, $\varepsilon_i \sim \mathrm{N}(0,\sigma)$. The set of values of the explanatory variable is denoted by $\mathbf{Z} \equiv \{\mathbf{z}_i\}_{i=1}^{N}$ and the set of corresponding target values by the vector $\mathbf{Y} \equiv \{y_i\}_{i=1}^{N}$. The set of function values corresponding to the explanatory variable $\mathbf{Z}$ is denoted by vector $\mathbf{F} \equiv \{\mathrm{f}(\mathbf{z}_i)\}_{i=1}^{N}$. Finally, the set of values for the noise vector is described by $\mathbf{e} \equiv \{\varepsilon_i\}_{i=1}^{N}$. For notational simplicity, let

$$w_i = \mathrm{f}(\mathbf{z}_i)$$
$$\therefore y_i = w_i + \varepsilon_i$$

Given a prior joint probability distribution, $p(\mathbf{F})$, over the space of function values, **F**, and a prior joint probability distribution, $p(\mathbf{e})$, over the noise, the probability of the data is given by

$$p(\mathbf{Y} \mid \mathbf{Z}) = \int p(\mathbf{Y} \mid \mathbf{Z}, \mathbf{F}, \mathbf{e}) p(\mathbf{F}) p(\mathbf{e}) d\mathbf{F} d\mathbf{e} \tag{3}$$

Consider the vector $\mathbf{Y}' \equiv [y_1, ..., y_N, w_{N+1}]$ constructed from **Y** and w_{N+1}, the conditional probability distribution of w_{N+1} can be written as

$$p(w_{N+1} \mid \Delta, \mathbf{z}_{N+1}) = \frac{p(\mathbf{Y}' \mid \mathbf{Z}, \mathbf{z}_{N+1})}{p(\mathbf{Y} \mid \mathbf{Z})}$$

This conditional distribution can be used to make predictions about w_{N+1}. Generally, given the data-noise relationship in (2), $p(\mathbf{Y} \mid \mathbf{Z}, \mathbf{F}, \mathbf{e})$ is simply a product of delta functions $\prod_i \delta(\varepsilon_i = y_i - \mathrm{f}(\mathbf{z}_i))$. By assuming Gaussian noise and integrating over the noise data, it follows that equation (3) becomes

$$p(\mathbf{Y} \mid \mathbf{Z}, \beta) = \int p(\mathbf{F}) \frac{1}{Z} \exp\left\{ -\frac{1}{2\beta} \sum_{i=1}^{N} [y_i - \mathrm{f}(\mathbf{z}_i)]^2 \right\} d\mathbf{F}$$

where β is the variance of the noise and Z is a normalising constant. In the specific case where **F** is Gaussian, $p(\mathbf{Y}|\mathbf{Z})$ is itself Gaussian with mean defined as the sum of the means of **F** and **e** and the variance defined as the sum of the variances of **F** and **e**.

2.4.3 Posterior Joint Probability Distribution

Given the prior distribution, the posterior distribution is derived by conditioning it on data. Assume the general case of predicting w_{N+1}, the value of the nonlinear relationship at $\mathbf{z}_{N+1}$ given the dataset $\Delta = \{\mathbf{Z}, \mathbf{Y}\}$ where $\mathbf{Z} = [\mathbf{z}_1 ..., \mathbf{z}_N]^{\mathrm{T}}$ and $\mathbf{Y} = [y_1 ..., y_N]^{\mathrm{T}}$, the conditional probability for w_{N+1} is calculated using Bayes' theorem

$$p(w_{N+1} \mid \Delta, \mathbf{z}_{N+1}) = \frac{p(\mathbf{Y}_{N+1} \mid \mathbf{Z}, \mathbf{z}_{N+1})}{p(\mathbf{Y} \mid \mathbf{Z})}$$

where $\mathbf{Y}_{N+1} = \left[\mathbf{Y}^{\mathrm{T}}, w_{N+1}\right]^{\mathrm{T}}$.

It follows from the zero-mean Gaussian prior assumption that

$$p\left(w_{N+1}, \mathbf{Y} \mid \mathbf{z}_{N+1}\right) \propto \exp\left[-\frac{1}{2}\begin{bmatrix} w_{N+1} & \mathbf{Y}^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} \kappa & k_{N+1}^{\mathrm{T}} \\ k_{N+1} & Q_N \end{bmatrix}^{-1}\begin{bmatrix} w_{N+1} \\ \mathbf{Y} \end{bmatrix}\right]$$

where κ is E[w_{N+1}, w_{N+1}], the ij^{th} element of the covariance matrix Q_N is E[y_i, y_j], and the i^{th} element of vector k_{N+1} is E[y_i, w_{N+1}]. Both κ and k_{N+1} depends on the posterior explanatory variable. It might be easier to explain the relationship of the covariance matrix as shown in (4). A larger covariance matrix, Q_{N+1}, is constructed from a smaller matrix, Q_N, vector k_{N+1} and scalar κ.

$$\overbrace{\begin{matrix} \\ N+1 \end{matrix}}^{N+1}\left\{\begin{matrix} \\ N+1 \end{matrix}\right.\begin{bmatrix} \\ Q_{N+1} \\ \\ \end{bmatrix} = \left[\begin{array}{c|c} \kappa & k_{N+1}^{\mathrm{T}} \\ \hline k_{N+1} & Q_N \end{array}\right] \tag{4}$$

Generally, the data that is used for training should be different from the data that is to be used for prediction, i.e. $\mathbf{z}_{N+1} \notin \mathbf{Z}_N \subseteq \Delta$. However, this distinction is frequently ignored because of insufficient data.

Applying the partitioned matrix lemma, it follows that

$$p\left(w_{N+1} \mid \Delta, \mathbf{z}_{N+1}\right) \propto \exp\left[-\frac{1}{2}\left(w_{N+1} - \hat{w}\right)\lambda_{\mathrm{z}}^{-1}\left(w_{N+1} - \hat{w}\right)\right]$$

with the mean,

$$\hat{w} = k_{N+1}^{\mathrm{T}} Q_N^{-1} \mathbf{Y}$$

interpreted to be a fit for the data and

$$\lambda_{\mathrm{z}} = \kappa - k_{N+1}^{\mathrm{T}} Q_N^{-1} k_{N+1}$$

interpreted to be the variance for the posterior. It is interesting to note that both the mean and the variance contain Q_N^{-1} and not Q_{N+1}^{-1}. $\hat{w}$ is interpreted to be the best fit to

the data at $\mathbf{z}_{N+1}$. The standard deviation for the mean is simply the square-root of λ_z, i.e. $\sqrt{\lambda_z}$. Generally, a 95% confidence interval is used on the best fit of the data; that is, two times the value of the standard deviation.

N' predictions, $\mathbf{W}' = [w'_1,..., w'_{N'}]^{\mathrm{T}}$, can be made simultaneously at different values of the explanatory variable, $\{\mathbf{z}'_i\}_{i=1}^{N'}$. It follows that, for the posterior joint probability distribution, the mean vector is

$$\mathbf{W}' = \Lambda_{N'}^{\mathrm{T}} Q_N^{-1} \mathbf{Y}$$

and the covariance matrix is

$$\Lambda'_z = \Lambda_\kappa - \Lambda_{N'}^{\mathrm{T}} Q_N^{-1} \Lambda_{N'}$$

where Λ_κ is $\mathrm{E}[\mathbf{W}'\mathbf{W}'^{\mathrm{T}}]$ and $\Lambda_{N'}$ is $\mathrm{E}[\mathbf{Y}\mathbf{W}'^{\mathrm{T}}]$.

The posterior process model requires the definition of the mean function and the covariance function. The mean function is defined by $\hat{w}$ as $\mathbf{z}$ varies. From the prediction for w and w' at $\mathbf{z}$ and $\mathbf{z}'$, the covariance function for the posterior process model is defined using the joint probability distribution for w and w', namely by $\mathrm{E}[ww'] - \mathrm{E}[w]\mathrm{E}[w']$, as $\mathbf{z}$ and $\mathbf{z}'$ vary.

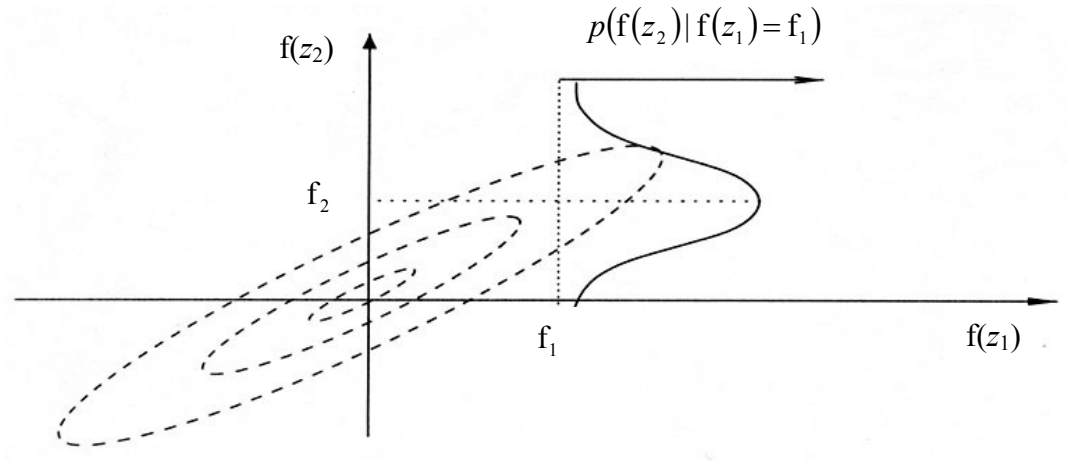


Figure 4 Conditional probability distribution.

An illustration of the conditional probability is provided by the simple example depicted in Figure 4. The figure shows the Gaussian joint probability density of two function values, f(z_1) and f(z_2). Given a particular value of $\mathrm{f}(z_1) = \mathrm{f}_1$, the Gaussian conditional distribution of $p(\mathrm{f}(z_2) | \mathrm{f}(z_1) = \mathrm{f}_1)$ can be determined and calculated to find the most probable prediction of f(z_2) given $\mathrm{f}(z_1) = \mathrm{f}_1$ as shown in Figure 4.

2.5 Model Selection

How to choose the covariance function? Various classes of covariance functions exist (Abramowitz and Stegun, 1965; Stein, 1999). Some are general-purpose, whereas others are more case specific. Basically, covariance functions can be classified into two types; that is, stationary (Yaglom, 1987) and non-stationary (Neal, 1996) covariance functions. Different classes of covariance functions are discussed in §2.6. Covariance functions, characterised by a set of free parameters, are the fundamental building blocks of Gaussian processes. Generally, these free parameters are known as hyperparameters[1]. Based on the prior knowledge about the nonlinear relationship, a particular class of covariance functions that is best suited for the application is first chosen, i.e. choosing a specific form of the mean function and covariance function parameterised by some set of hyperparameters, Θ. These hyperparameters characterise the class of Gaussian process from which the Gaussian process prior model is to be chosen (Rasmussen and Williams, 2006). Hence, a proper choice of the class of covariance functions is essential to the development of Gaussian regression.

The values of the hyperparameters, Θ, of the Gaussian process prior model need to be specified. Two methods are commonly used. In the first approach, specific values are chosen for Θ by maximising the likelihood function. In the second method, a Bayesian approach is used to place a prior over the values of hyperparameters. The former method is known as Likelihood maximisation (MacKay, 1992) and the latter is the Monte Carlo approach (Williams and Rasmussen, 1996; Neal, 1997), and both are equally legitimate techniques to define the model.

[1] Hyperparameters are normally referred to parameters of the covariance function to emphasise their characteristics belonging to a non-parametric model.

2.5.1 Likelihood Maximisation

Once a specific class of covariance functions is chosen, the values for the hyperparameters must be selected. Likelihood maximisation allows hyperparameters Θ_p to be chosen for the model,

$$p\left(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}, \Theta_P\right)$$

In Likelihood maximisation, the most probable hyperparameter values are computed. This can be done through the use of a standard gradient-based optimisation algorithm, such as the conjugate gradient method and the trust-region method. Gradient-based optimisation requires user-supplied gradient information, or more specifically, first order derivative information of the likelihood function.

To obtain a model given the data, the hyperparameters are adapted to maximise the log-likelihood function, or equivalently, minimise the negative log-likelihood function,

$$\Lambda(\Theta) = \frac{1}{2}\log|Q| + \frac{1}{2}\mathbf{Y}^{\mathrm{T}} Q^{-1}\mathbf{Y} \tag{5}$$

where |.| refers to the determinant operator of a matrix. It follows that its derivative with respect to hyperparameter θ_i is

$$\frac{\partial\Lambda(\Theta)}{\partial\theta_i} = \frac{1}{2}tr\left\{Q^{-1}\frac{\partial Q}{\partial\theta_i}\right\} - \frac{1}{2}\mathbf{Y}^{\mathrm{T}} Q^{-1}\frac{\partial Q}{\partial\theta_i}Q^{-1}\mathbf{Y} \tag{6}$$

where *tr*(.) is the trace operator of a matrix.

During the optimisation procedure, two potential problems arise as a result of training the hyperparameters. Firstly, because of the multi-modal log-likelihood function, finding the most probable (and sensible) values of the hyperparameters is highly dependent on the chosen initial values. The log-likelihood function is generally multi-modal with respect to Θ and these modes correspond to different values of the hyperparameters resulting in different priors (see Example 2.2). This is partly due to

the optimisation algorithm being used with different initial values for the hyperparameters. Proper choice of initial values for the hyperparameters ensures that optimisation converges faster to a maximum point. This is discussed in more detail in Chapter 3.4. Secondly, it is important to note that every evaluation of the negative log-likelihood function and its gradient information requires the evaluation of Q^{-1}. Exact inversion of a matrix has an expensive computational cost of $O(N^3)$ operations. This results in an extremely time-consuming process when training large-scale datasets. Moreover, exact matrix inversion computation has an $O(N^2)$ memory requirement and standard MATLAB optimisation routines normally fail, due to lack of memory space, at around $N = 3{,}000$.

Example 2.2 *(Optimisation with Log-Likelihood Function)*. Suppose a dataset $\Delta = \{z_i, y_i\}_{i=1}^{5}$ has five measurement points, i.e. {-1,4}, {-2.5,1.5}, {0.5,0.5}, {3,2} and {4,2.1}, as shown in Figure 5 and Figure 6. Given this prior information, a specific class of covariance functions is selected for the Gaussian process prior models, such that

$$C_{\mathrm{f}}(z_i, z_j) = a \exp\left[-\frac{d}{2}(z_i - z_j)^2 \right]$$

The correlation between measurement points, y_i and y_j, is

$$C(y_i, y_j) = C_{\mathrm{f}}(z_i, z_j) + b\delta_{ij}$$

where a, b and d are hyperparameters. Hyperparameters in the covariance function are adapted to minimise the function (5). Two different sets of initial values for the optimisation are selected. With one set of initial values, the optimisation converges to a minimum point with hyperparameter values of $a = 1.6$, $d = 0.3$ and $b = 1.4\text{x}10^{-14}$. The resulting prediction with confidence intervals is shown in Figure 5. With the second set of initial values, the optimisation also converges to a minimum point with adapted hyperparameter values $a = 1.9$, $d = 0.002$ and $b = 0.53$. Its prediction is illustrated in Figure 6. The latter indicates that the fit is a long lengthscale, whereas the former is a short lengthscale. Clearly, both are maxima of the likelihood function. Thus, it is obvious that the log-likelihood function is a multi-modal nonlinear function.

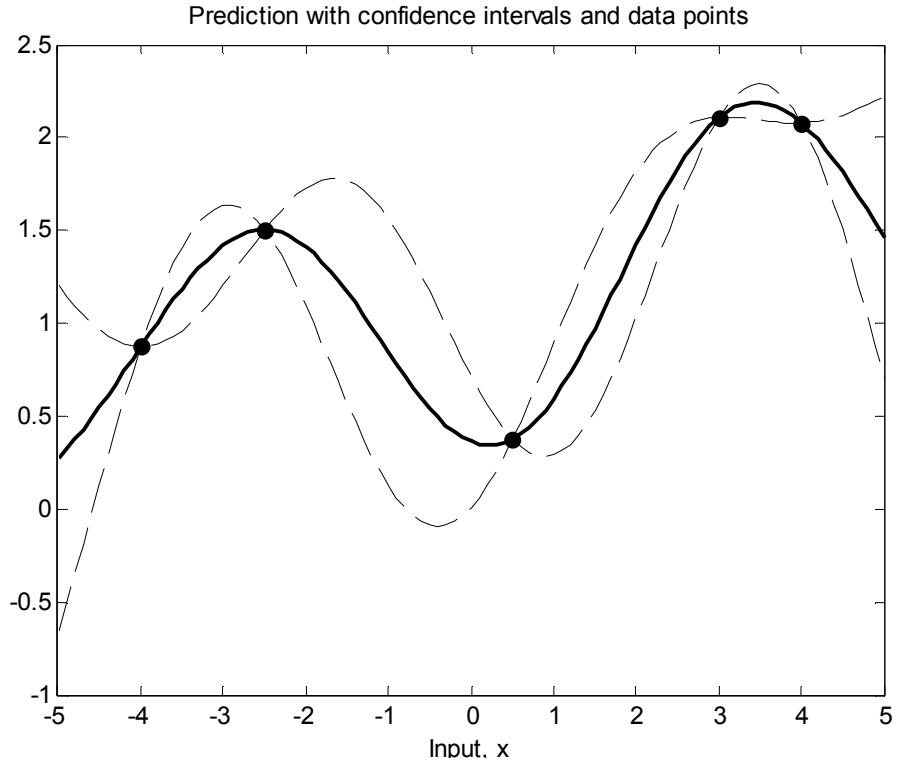


Figure 5 Data points and prediction with two times standard deviations of the fit from using a set of hyperparameter values.

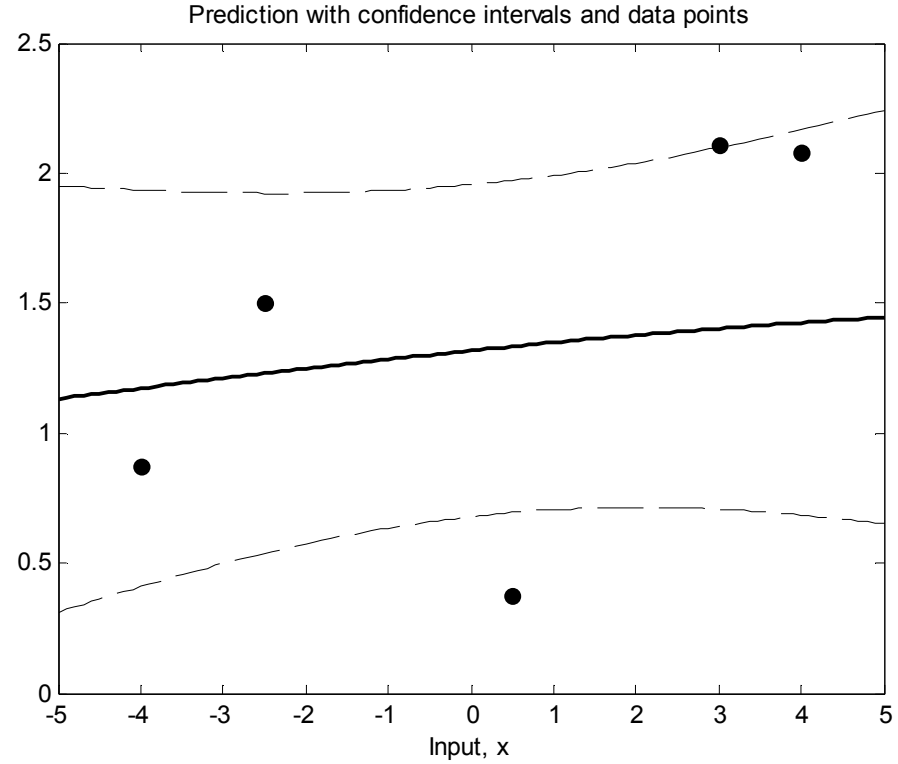


Figure 6 Data points and prediction with two times standard deviations of the fit from using a different set of hyperparameter values.

2.5.2 Monte Carlo Method

Hyperparameters are not necessarily assigned specific values but have prior distributions imposed on them. To select the model with hyperparameters having prior distributions, an approach known as the Monte Carlo Markov chain is used. In Gaussian process, the Monte Carlo approach uses the idea of sampling to approximate the posterior joint probability distribution. This approach has a completely different philosophy from Likelihood maximisation.

Ideally, it is plausible to integrate over all the undetermined hyperparameters; that is,

$$p(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}) = \int p(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}, \Theta) p(\Theta \mid \Delta, C_{\mathrm{f}}) d\Theta \tag{7}$$

where C_{f} is defined as the covariance function for the model. Unfortunately, computing this integration is analytically difficult, particularly for an arbitrary covariance function C_{f}, an alternative method is required if Gaussian processes are to be computed in a less complicated way.

In the Bayesian context, the posterior probability distribution over the weights is

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

It follows that the posterior joint probability distribution of Θ can be written as

$$p(\Theta \mid \Delta, C_{\mathrm{f}}) = \frac{p(\mathbf{Y} \mid \mathbf{Z}, C_{\mathrm{f}}, \Theta) p(\Theta)}{p(\mathbf{Y} \mid \mathbf{Z}, C_{\mathrm{f}})} \tag{8}$$

The probability, $p(\mathbf{Y} \mid \mathbf{Z}, C_{\mathrm{f}}, \Theta)$, is the probability of the target values given the set of values of the explanatory variable, $\mathbf{Z}$, and the covariance function, C_{f}. The term $p(\Theta)$ denotes the prior distribution of the hyperparameters that define the mean function and covariance function. The normalising constant $p(\mathbf{Y} \mid \mathbf{Z}, C_{\mathrm{f}})$ is also known as the marginal likelihood and is independent of Θ. Thus, it can be safely ignored here as the intent here is to compute the most probable hyperparameters. The normalising constant is given by

$$p(\mathbf{Y} \mid \mathbf{Z}, C_{\mathrm{f}}) = \int p(\mathbf{Y} \mid \mathbf{Z}, C_{\mathrm{f}}\Theta)p(\Theta)d\Theta \tag{9}$$

The posterior in (8) integrates the information about the likelihood and the prior together and encapsulates the knowledge of the hyperparameters.

By approximating the integral in (7), the equation can be re-written as

$$p(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}) \cong \frac{1}{T}\sum_{t=1}^{T} p(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}, \Theta_t)$$

where Θ_t are samples drawn from the posterior distribution over Θ, $p(\Theta \mid \Delta, C_{\mathrm{f}})$. Since each term in the summation of the above equation is a Gaussian, the Monte Carlo approximation is a mixture of Gaussians. It is worthwhile noting that accuracy increases as more samples are drawn from the posterior over Θ. However, if the sample is not taken from a particular region of hyperparameter space that has a high associated probability to the posterior, then the accuracy of Monte Carlo approximation will be poor. Much research on Monte Carlo methods for Gaussian process regression, such as Hybrid Monte Carlo algorithm (Duane et al., 1987), has been undertaken.

If it is assumed that the posterior joint probability distribution over Θ has a sharp peak around the region near Θ_P relative to $p(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}, \Theta_P)$, then the approximation of $p(w_{N+1} \mid \mathbf{z}_{N+1}, \Delta, C_{\mathrm{f}}, \Theta_P)$ gives similar result to the Likelihood Maximisation approach.

A comparison of these two methods has been undertaken by Gibbs (1997) and Rasmussen (1996). In general, there is no conclusive answer as to which method is superior. The Likelihood maximisation method of obtaining the values for the hyperparameters is generally good as predictions made using these values are often found to be very close to those using the true posterior joint probability distribution (MacKay, 1993). The Monte Carlo approach has been found to perform better for smaller datasets when matrix storage is not an issue. Computation using this approach results in better solutions for a fixed amount of computational time. On the other hand, Likelihood maximisation is preferred for computations involving large datasets.

It was found (Rasmussen, 1996) that the optimisation method produces more accurate results and faster computations in large datasets.

2.6 Covariance Functions

In studying Gaussian processes, it is necessary to understand the role of the covariance function, a crucial and elementary component. A covariance function specifies the covariance between pairs of random variables. In a stochastic process, the covariance function determines the correlation between the function values and provides a distinct indication of how they are connected to each other. For instance, in supervised learning, a basic assumption is that data points that are close to each other are likely to have similar target values, and therefore have higher correlation compared to data points that are far apart. Similarly, in Gaussian process, the covariance function defines the closeness or similarity of function values in the Euclidean space of inputs. A variety of covariance functions has been investigated by several researchers (Gibbs, 1997; Mackay, 1998; Rasmussen, 1996).

2.6.1 Stationary Covariance Functions

This section explores some covariance functions that are commonly used in the machine learning community. Attention is focused on a few classes of covariance functions relevant to the application of interest here. Covariance functions can be classified into stationary and non-stationary ones. Primarily, a *stationary covariance function* is one that is function of $\mathbf{z} - \mathbf{z}'$, that is invariant to translations in the input space. The covariance function of a stationary process can also be represented by the Fourier transform of a positive measure.

Theorem 2.1 *(Bochner's theorem)*. A complex-valued function k on P^{Δ} is the covariance function of a weakly stationary mean square continuous complex-valued random process on P^{Δ} if and only if it can be represented as

$$k(\tau) = \int_{\mathrm{P}^D} e^{2\pi i s.\tau} d\mu(s)$$

where μ is a positive finite measure (Rasmussen and Williams (2006)).

One such example is the *squared exponential*[2] covariance function

$$\mathrm{cov}\left(\mathrm{f}(\mathbf{z}_i), \mathrm{f}(\mathbf{z}_j)\right) = C_{\mathrm{f}}(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{1}{2}\left|\mathbf{z}_i - \mathbf{z}_j\right|^2\right) \tag{10}$$

For this particular covariance function, the covariance approaches unity when the inputs are very close to each other, and decreases as distance in the input space increases. Note that the covariance of the outputs is written as a function of the input. This squared exponential covariance function is also positive definite, and a covariance function is said to be positive definite if

$$\int C_{\mathrm{f}}(\mathbf{z}_i, \mathbf{z}_j)\mathrm{f}(\mathbf{z}_i)\mathrm{f}(\mathbf{z}_j) d\mu(\mathbf{z}_i) d\mu(\mathbf{z}_j) > 0$$

It is also interesting to establish if a covariance function is mean square continuous and differentiable. There are applications which require infinite differentiability in the stochastic process, which is discussed in later chapters. Firstly, to describe mean continuous and differentiability of a stochastic process, let $\mathbf{z}_1$, $\mathbf{z}_2$,… be a sequence of data points and $\mathbf{z}^*$ be a fixed point in P^{Δ}, such that $\left|\mathbf{z}_k - \mathbf{z}^*\right| \to 0$ as $k \to \infty$. Consequently, a stochastic process f($\mathbf{z}$) is mean continuous in $\mathbf{z}^*$ if $\mathrm{E}\left[\left|\mathrm{f}(\mathbf{z}_k) - \mathrm{f}(\mathbf{z}^*)\right|^2\right] \to 0$. If this is true for all $\mathbf{z}^* \in A \subseteq \mathrm{P}^{\Delta}$, then f($\mathbf{z}$) is known to be continuous in mean square over A (Adler, 1981). The mean square derivative of the stochastic field, f($\mathbf{z}$) in the i^{th} direction is defined as

$$\frac{\partial \mathrm{f}(\mathbf{z})}{\partial z_i} = \lim_{\lambda \to 0} \frac{\mathrm{f}(\mathbf{z} + \lambda \mathbf{e}_i) - \mathrm{f}(\mathbf{z})}{\lambda} \tag{11}$$

where the limit exists in mean square and $\mathbf{e}_i$ denotes the unit basis vector in the i^{th} direction, is well-defined in the limit as $\lambda \to 0$, such that the complete description exists for all the necessary probability distributions. The expectation of the mean square derivative of the stochastic field, f($\mathbf{z}$), in the i^{th} direction as $\mathbf{z}$ varies, is

[2] The squared exponential covariance function is sometimes known as the Radial Basis Function or Gaussian.

interpreted to be the derivative of the fit to f(**z**). Provided that the mean function and covariance function are sufficiently differentiable, it is well known (O'Hagan, 1978) that the derivative stochastic process is itself Gaussian and that

$$\mathrm{E}\left[\frac{\partial \mathrm{f}(\mathbf{z})}{\partial z_i}\right] = \frac{\partial h_{\mathrm{f}}(\mathbf{z})}{\partial z_i}; \quad h_{\mathrm{f}}(\mathbf{z}) = \mathrm{E}[\mathrm{f}(\mathbf{z})] \tag{12}$$

where z_i denotes the i^{th} element of **z**; that is, the expected value of the derivative stochastic process is just the derivative of the expected value of the stochastic process. Furthermore,

$$\mathrm{E}\left[\frac{\partial \mathrm{f}(\mathbf{z}_u)}{\partial z_i}, \frac{\partial \mathrm{f}(\mathbf{z}_v)}{\partial z_j}\right] = \nabla_i^1 \nabla_j^2 C_{\mathrm{f}}(\mathbf{z}_u, \mathbf{z}_v); \qquad C_{\mathrm{f}}(\mathbf{z}_u, \mathbf{z}_v) = \mathrm{E}[\mathrm{f}_u(\mathbf{z}), \mathrm{f}_v(\mathbf{z})] \tag{13}$$

where $\nabla_i^1 C_{\mathrm{f}}(\mathbf{z}_u, \mathbf{z}_v)$ denotes the partial derivative of $C_{\mathrm{f}}(\mathbf{z}_u,\mathbf{z}_v)$ with respect to the i^{th} element of its first argument, etc. The above procedure can be repeated to construct second derivative processes. It follows that the covariance of the mean square derivative (11) is given by $\partial^2 C_{\mathrm{f}}(\mathbf{z}_u, \mathbf{z}_v) / \partial z_i \partial z_j$. The procedure can be extended to higher order derivatives. Note that, for a squared exponential covariance function, its second order partial derivative exists for all $\mathbf{z} \subseteq \mathrm{P}^{\Delta}$. In addition, the squared exponential covariance function has infinite order of partial derivatives.

The squared exponential covariance function (10) has a basic form dependent only on the explanatory variable. Parameters can be introduced to span different class of squared exponential covariance functions, such as in (14) and (15).

$$\mathrm{cov}(\mathrm{f}(\mathbf{z}_i), \mathrm{f}(\mathbf{z}_j)) = \exp\left(-\frac{d}{2}|\mathbf{z}_i - \mathbf{z}_j|^2\right) \tag{14}$$

$$\mathrm{cov}(\mathrm{f}(\mathbf{z}_i), \mathrm{f}(\mathbf{z}_j)) = a \exp\left(-\frac{d}{2}|\mathbf{z}_i - \mathbf{z}_j|^2\right) \tag{15}$$

In (14), the parameter, d, defines the characteristic lengthscale of the correlation between pairs of input data points. This covariance function has mean square derivative of all orders and thus is infinitely differentiable. It is also a clear indication that the derivatives are smooth. Although it has been argued that such strong smoothness in the characteristics of the covariance function (Stein, 1999) is

unrealistic in real life scenario, it is still probably the most widely used kernel in the machine learning community. The parameter, *a*, in (15) relates to the amplitude of the measured variable.

Another type of stationary covariance function is the Matérn class of covariance functions, given by

$$\mathrm{cov}\left(\mathrm{f}\left(\mathbf{z}_i\right),\mathrm{f}\left(\mathbf{z}_j\right)\right)=\frac{2^{1-\nu}}{\Gamma\left(\nu\right)}\left(\frac{\sqrt{2\nu}\left|\mathbf{z}_i-\mathbf{z}_j\right|}{l}\right)^{\nu}Q_{\nu}\left(\frac{\sqrt{2\nu}\left|\mathbf{z}_i-\mathbf{z}_j\right|}{l}\right) \tag{16}$$

where v and l are positive parameters, and Q_v is a modified Bessel function (Abramowitz and Stegun, 1965). Named after the work of Matérn, the process for the Matérn class covariance function is *k*-times mean square differentiable, if and only if $v > k$.

Another interesting class of covariance function is the γ-exponential covariance function given by

$$\mathrm{cov}\left(\mathrm{f}\left(\mathbf{z}_i\right),\mathrm{f}\left(\mathbf{z}_j\right)\right)=\exp\left\{-\left(\frac{\left|\mathbf{z}_i-\mathbf{z}_j\right|}{l}\right)^{\gamma}\right\} \qquad \forall\ 0<\gamma\le 2$$

This covariance function is similar to the Matérn class covariance function, except it is not mean square differentiable if γ is not equal to 2. Several other covariance functions also exist in the literature, such as piece-wise polynomial covariance function and the exponential covariance function of the Ornstein-Uhlenbeck process (Rasmussen and Williams, 2006).

2.6.1.1 Squared Exponential Covariance Function

Throughout this thesis, unless otherwise specified, the correlation between measured values of the following form is used

$$C\left(y_i,y_j\right)=C_{\mathrm{f}}\left(\mathbf{z}_i,\mathbf{z}_j\right)+b\delta_{ij}=a\exp\left\{-\frac{1}{2}\left(\mathbf{z}_i-\mathbf{z}_j\right)^{\mathrm{T}}D\left(\mathbf{z}_i-\mathbf{z}_j\right)\right\}+b\delta_{ij} \tag{17}$$

where a, b and $D = diag\{d_1,...,d_k\}$ are a set of values of hyperparameters. The hyperparameter, a, gives the overall vertical scale relative to the mean of the Gaussian process in the output space. The hyperparameter, b, represents the noise model, indicating the noise variance in the data. The term, d_i, refers to the lengthscale in the i^{th} dimension of the explanatory variable, $\mathbf{z}$. It characterises the distance over which the amount of averaging of data is done. For example, a short lengthscale means there is more contribution of nearby values of the input explanatory variable than values that are far apart. On the other hand, a long lengthscale would expect averaging to be done over a larger distance; values far apart still contribute to a reasonable amount to the smoothing of the input values of the explanatory variable. For dataset with one-dimensional explanatory variable, the correlation between measured values can be simplified to

$$C(y_i, y_j) = a \exp\left\{-\frac{d}{2}(\mathrm{z}_i - \mathrm{z}_j)^2\right\} + b\delta_{ij}$$

where $\mathrm{z} \in \mathrm{P}$.

Discussion 2.1 *(Squared Exponential Covariance Functions of GP Derivative)*. In the context of the squared exponential covariance functions; these functions are infinitely differentiable. The covariance between a derivative observation and function observation, and covariance between two derivative observations are shown in (18) and (19) respectively, where z^k refers to the k^{th} dimension of the explanatory variable $\mathbf{z}$. Let the underlying function be $w = \mathrm{f}(\mathbf{z})$ such that w_i refers to the i^{th} entry of w and w_i^m refers to the i^{th} entry of the derivative of w.

$$\mathrm{cov}(w_i^m, w_j) = \frac{\partial}{\partial \mathrm{z}^m}\mathrm{cov}(w_i, w_j) \tag{18}$$

$$\mathrm{cov}(w_i^m, w_j^n) = \frac{\partial^2}{\partial \mathrm{z}^m \partial \mathrm{z}^n}\mathrm{cov}(w_i, w_j) \tag{19}$$

where $\mathrm{cov}(w_i^m, w_j)$ is the covariance between a derivative point and a function point, and $\mathrm{cov}(w_i^m, w_j^n)$ is the covariance between corresponding input derivative points.

The following identities are necessary for the construction of the derivative Gaussian process prior model.

$$\operatorname{cov}\left(w_i, w_j\right) = a \exp\left\{-\frac{1}{2}\left(\mathbf{z}_i - \mathbf{z}_j\right)^{\mathrm{T}} D\left(\mathbf{z}_i - \mathbf{z}_j\right)\right\} \tag{20}$$

$$C\left(w_i^m, w_j\right) = -a d_m \left(\mathrm{z}_i^m - \mathrm{z}_j^m\right) \exp\left\{-\frac{1}{2}\left(\mathbf{z}_i - \mathbf{z}_j\right)^{\mathrm{T}} D\left(\mathbf{z}_i - \mathbf{z}_j\right)\right\} \tag{21}$$

and

$$C\left(w_i^m, w_j^n\right) = a d_m \left\{\delta_{m,n} - d_n \left(\mathrm{z}_i^m - \mathrm{z}_j^m\right)\left(\mathrm{z}_i^n - \mathrm{z}_j^n\right)\right\} \exp\left\{-\frac{1}{2}\left(\mathbf{z}_i - \mathbf{z}_j\right)^{\mathrm{T}} D\left(\mathbf{z}_i - \mathbf{z}_j\right)\right\} \tag{22}$$

2.6.2 Non-stationary Covariance Functions

Most of the priors are stationary, although all posteriors are automatically non-stationary. For completeness, a brief introduction of non-stationary covariance functions is introduced in this section. Non-stationary covariance functions are commonly used in neural network, e.g. mappings that describes a single hidden layer neural network

$$f\left(\mathbf{z}\right) = b + \sum_{i=1}^{N} v_i h\left(\mathbf{z}; \mathbf{u}_i\right)$$

where v_i refers to the hidden-to-output weight and $h(\mathbf{z};\mathbf{u})$ is the hidden unit transfer function, which depends on the input-to-hidden weights $\mathbf{u}$. A common feature vector function is $h(\mathbf{z};\mathbf{u}) = \tanh(\mathbf{z}.\mathbf{u})$. This sigmoid kernel is viewed as a non-stationary covariance function. Other variants of non-stationary covariance functions also exist, e.g. the Wiener process. Further details can be traced to related research work in Grimmett and Stirzaker (1992).

Other non-stationary covariance functions which are used in a later chapter are the linear form covariance function and quadratic form covariance function, and are briefly discussed here. The linear covariance function is defined in equation (23) and the quadratic covariance function in equation (24).

$$\operatorname{cov}\left(f\left(\mathbf{z}_i\right), f\left(\mathbf{z}_j\right)\right) = \sum_{k=1}^{K} w_k \mathrm{z}_i^k \mathrm{z}_j^k \tag{23}$$

$$\operatorname{cov}\left(f\left(\mathbf{z}_i\right), f\left(\mathbf{z}_j\right)\right) = \sum_{k=1}^{K} w_k \left(\mathrm{z}_i^k\right)^2 \left(\mathrm{z}_j^k\right)^2 \tag{24}$$

$\{w_k\}_{k=1}^{K}$ are defined to be the values of the hyperparameters and $\mathbf{z}_i^k$ defined to be the k^{th} element of the i^{th} value of the explanatory variable, $\mathbf{z}$. Unlike stationary covariance functions, nearby data points do not necessarily have strong correlation as shown in equations (23) and (24). These two covariance functions are useful if there is a belief that some linear or quadratic trend exists in the data. An application of using non-stationary covariance functions such as these is discussed in Chapter 5.3.

Chapter 3

Fast Algorithm Implementation for Gaussian Regression

3.1 Computation Issues

Gaussian process regression involves several matrix computations of $O(N^3)$ operations, such as matrix inversion and the calculation of the log-determinant of a $N \times N$ covariance matrix (Leithead et al, 2005c). In addition, these covariance matrices have an $O(N^2)$ storage requirement, for any explicit computations. These limitations effectively restrict the number of training cases, N, to at most a few thousand cases. Though it may be possible to use super computers to handle these computations, this approach is undoubtedly effective but inefficient. To overcome the computational limitation issues and cater for large-scale dataset application, numerous authors have recently suggested a wealth of sparse approximations (Schwaighofer and Tresp, 2003; Seeger et al., 2003; Smola and Bartlett, 2001). Quiñonero-Candela and Rasmussen (2005) have further provided a unified view of sparse Gaussian process approximation, which includes a comparison of work published by various authors. Common to all these approximation methods is that only a subset of the latent variables is treated exactly, with the remaining variables given some approximate, but

computationally cheaper approach (Quiñonero-Candela and Rasmussen, 2005). However, it is of interest of this thesis to investigate and research upon exact implementation of Gaussian process using all the latent variables, instead of a subset of them; hence the approach of using sparse approximation methodologies is avoided. In this chapter, fast and memory efficient algorithms are developed for the class of full, exact implementation of Gaussian process models to handle large number of training cases, e.g. one million data points. This work is in contrast to other authors, whose works are focused either upon deterministic approach or method that does not correspond exactly to a Gaussian process (Quiñonero-Candela and Rasmussen, 2005).

3.2 Effective Hessian Matrix Exploitation

Discussions on maximum likelihood estimation (MLE) optimisation problems are rarely available in the Gaussian regression literature, and are mostly centred on steepest-descent and conjugate-gradient approaches. Although these gradient-based optimisation algorithms are sufficient to guarantee convergence to stationary points, they are not fast enough.

Hessian, or more precisely the second order derivative information, provides additional information to check the nature of the converged solution, such as maxima or minima stationary point, and provides the possibility of rejecting saddle points. It is known that Hessian information provides a more efficient and effective optimisation (Moller, 1993), especially for ravine-type problems. Most large-scale gradient-based optimisation, such as the trust-region algorithm (MathWorks, 2003), is able to employ Hessian information to speed up the training procedure.

This section reviews the work of Zhang and Leithead (2005), where the optimisation performance, from explicit use of the Hessian matrix on a particular class of covariance functions, can be improved hence allowing faster convergence. Its exact implementation is then compared with approximation to the second order information using finite-differencing. It follows from the negative log-likelihood function, (5), and its derivative, (6), that its second order derivative is

$$\frac{\partial^2 \Lambda(\theta)}{\partial\theta_i \partial\theta_j} = \frac{1}{2} tr\left\{ Q^{-1} \frac{\partial^2 Q}{\partial\theta_i \partial\theta_j} - Q^{-1} \frac{\partial Q}{\partial\theta_i} Q^{-1} \frac{\partial Q}{\partial\theta_j} \right\} - \frac{1}{2} \mathbf{Y}^{\mathrm{T}} Q^{-1} \left\{ \frac{\partial^2 Q}{\partial\theta_i \partial\theta_j} - 2 \frac{\partial Q}{\partial\theta_i} Q^{-1} \frac{\partial Q}{\partial\theta_j} \right\} Q^{-1} \mathbf{Y} \tag{25}$$

The Hessian derivation is shown in Appendix B.

3.2.1 Simplification of Hessian Matrices

Zhang and Leithead (2005) have shown that the explicit computation of the Hessian matrix, as given by (25) is rather efficient, but further improvement is possible by exploiting the chosen form of covariance function (17). Since the squared-exponential covariance function is often encountered and used in this thesis, this simplification method shall be discussed here as a formality.

As the hyperparameters, $\theta = \{a, b, D\}$, are constrained to be positive scalars, they can be re-written to take exponential form, i.e., $a = e^{\alpha}$, $b = e^{(\alpha+\beta)}$ and $D = e^{\Gamma}$, where $\Gamma = diag\{\gamma_1,\ldots,\gamma_k\}$. These revised exponential hyperparameters $\bar{\theta} = \{\alpha, \beta, \gamma_1, ..., \gamma_k\}$ are adapted to minimise the negative log-likelihood function in an unconstrained optimisation, between minus infinity and plus infinity. The covariance matrix is modified to

$$Q = \Lambda_Q + \exp(\alpha + \beta) I$$

where the covariance matrix Λ_Q is defined as

$$\Lambda_Q(\mathbf{z}_i, \mathbf{z}_j) = \exp(\alpha) \exp\left\{ -\frac{1}{2} (\mathbf{z}_i - \mathbf{z}_j)^{\mathrm{T}} D (\mathbf{z}_i - \mathbf{z}_j) \right\}$$

with $D = e^{\Gamma} = diag\left\{e^{\gamma_1}, ..., e^{\gamma_k}\right\}$. The first and second order partial derivatives of Q with respect to hyperparameter α are

$$\frac{\partial Q}{\partial\alpha} = \frac{\partial^2 Q}{\partial\alpha^2} = Q;\ \frac{\partial^2 Q}{\partial\alpha\partial\beta} = \frac{\partial Q}{\partial\beta} = e^{(\alpha+\beta)} I;\ \frac{\partial^2 Q}{\partial\alpha\partial\gamma_k} = \frac{\partial Q}{\partial\gamma_k}, \qquad \forall k = 1,...,K$$

The partial derivatives of Q with respect to β are

$$\frac{\partial Q}{\partial \beta} = \frac{\partial^2 Q}{\partial \beta^2} = e^{(\alpha+\beta)} I \; ; \; \frac{\partial^2 Q}{\partial \beta \partial \gamma_k} = 0 \; , \qquad \forall k = 1,...,K$$

and the partial derivatives of Q with respect to γ are

$$\frac{\partial Q}{\partial \gamma_k} = -\frac{1}{2} \exp(\gamma_k) \left(\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)} \right)^2 \Lambda_Q$$

$$\frac{\partial^2 Q}{\partial \gamma_k \partial \gamma_p} = \begin{cases} \frac{\partial Q}{\partial \gamma_k} \left\{ 1 - \frac{1}{2} \exp(\gamma_p) \left(\mathbf{z}_i^{(p)} - \mathbf{z}_j^{(p)} \right)^2 \right\} & , \text{if } p = k \\ \frac{\partial Q}{\partial \gamma_k} \left\{ -\frac{1}{2} \exp(\gamma_p) \left(\mathbf{z}_i^{(p)} - \mathbf{z}_j^{(p)} \right)^2 \right\} & , \text{if } p \neq k \end{cases}$$

Due to symmetry of the Hessian matrix, $\partial^2 Q / (\partial \gamma_k \partial \gamma_p)$ need only be computed for $\forall k = 1,...,K$ and $k < p \leq K$. It follows from (25) that the α-related Hessian terms simplify to

$$\mathrm{H}_{11} = \frac{\partial^2 \Lambda}{\partial \alpha^2} = \frac{1}{2} \mathbf{Y}^{\mathrm{T}} Q^{-1} \mathbf{Y}$$

$$\mathrm{H}_{1(k+1)} = \frac{\partial^2 \Lambda}{\partial \alpha \partial \gamma_k} = \frac{1}{2} \mathbf{Y}^{\mathrm{T}} Q^{-1} \frac{\partial Q}{\partial \gamma_k} Q^{-1} \mathbf{Y} \; , \forall k = 1,...,K$$

$$\mathrm{H}_{1(K+2)} = \frac{\partial^2 \Lambda}{\partial \alpha \partial \beta} = \frac{\exp(\alpha + \beta)}{2} \left(\mathbf{Y}^{\mathrm{T}} Q^{-1} Q^{-1} \mathbf{Y} \right)$$

In addition, the β-related Hessian terms simplify to

$$\mathrm{H}_{(K+2)(k+1)} = \frac{\partial^2 \Lambda}{\partial \beta \partial \gamma_k}$$

$$= \frac{\exp(\alpha + \beta)}{2} \left\{ 2 \mathbf{Y}^{\mathrm{T}} Q^{-1} \frac{\partial Q}{\partial \gamma_k} Q^{-1} Q^{-1} \mathbf{Y} - tr\left(Q^{-1} \frac{\partial Q}{\partial \gamma_k} Q^{-1} \right) \right\}, \forall k = 1,...,K$$

$$\mathrm{H}_{(K+2)(K+2)} = \frac{\partial^2 \Lambda}{\partial \beta^2} = \frac{\partial \Lambda}{\partial \beta} - \frac{\exp(2(\alpha + \beta))}{2} \left\{ tr\left(Q^{-1} Q^{-1} \right) - 2 \mathbf{Y}^{\mathrm{T}} Q^{-1} Q^{-1} Q^{-1} \mathbf{Y} \right\}$$

Finally, the γ-related Hessian term is

$$\mathrm{H}_{(k+1)(p+1)} = \frac{\partial^2 \Lambda}{\partial \gamma_k \partial \gamma_p} = \frac{1}{2} tr\left(Q^{-1} \frac{\partial^2 Q}{\partial \gamma_k \partial \gamma_p} - Q^{-1} \frac{\partial Q}{\partial \gamma_k} Q^{-1} \frac{\partial Q}{\partial \gamma_p} \right)$$

$$-\frac{1}{2} \mathbf{Y}^{\mathrm{T}} Q^{-1} \left(\frac{\partial^2 Q}{\partial \gamma_k \partial \gamma_p} - 2 \frac{\partial Q}{\partial \gamma_k} Q^{-1} \frac{\partial Q}{\partial \gamma_p} \right) Q^{-1} \mathbf{Y} \quad , \forall k = 1,...,K$$

Since $Q^{-1}\frac{\partial Q}{\partial \gamma_k}$, for $k = 1,\ldots, K$, is frequently required at each iteration in the Gaussian regression implementation, these matrices are computed once and stored for subsequent usage.

3.2.2 Experimental Result

A straightforward example is chosen for this experiment, since the focus of this section is about understanding the theoretical foundations of the simplification approach rather than investigating the necessary heuristics needed to turn the scheme into actual practical algorithms.

TABLE I Performance comparison between optimisations user-supplied Hessian and Hessian approximation

Result	Iterations		Timing (minutes)	
Dataset size, N	Approx.	User	Approx.	User
200	64.2	64.3	0.13	0.05
400	58.4	58.4	0.53	0.19
600	85.9	85.5	2.11	0.86
800	123.8	123.9	5.60	2.07
1,000	88.0	86.6	7.47	2.75
1,200	139.9	145.7	18.45	7.41
1,400	97.3	98.8	18.34	7.10
1,600	107.6	107.7	29.73	12.02
1,800	150.6	150.2	53.53	20.64
2,000	149.7	147.2	78.92	31.99

Based on time-series datasets, unconstrained optimisation is performed to compare explicit user-supplied Hessian to approximated Hessian results, where user only supplies log-likelihood and gradient information. Experiment is carried on an Intel® Pentium® IV 2.8GHz machine with 512MB RAM.

The chosen test function is $f(z) = \sin(z) + 0.6\cos(5z)$, where $z \in [0,10]$ is a one-dimensional explanatory variable. Gaussian white noise, n_i, of variance 0.01 is added to the function, i.e. $y(z_i) = f(z_i) + n_i$, for $i = 1,\ldots, N$. For each set of data size, N, 10 sample tests, each with different noise and starting values for the optimisation, are conducted. The performance, in terms of timing and number of iterations, of both user-supplied Hessian information and Hessian approximation by finite-differencing

is being investigated and compared, with the results tabulated in TABLE I. The average timing and number of iterations are calculated for every N. Clearly, performance is better with user-supplied Hessian information, with an efficiency of about 1.5 to 2.5 times faster being evident.

3.3 Efficient Optimisation by Hyperparameter Reduction

Another efficient yet simple optimisation technique, through modifying the log-likelihood function, is introduced in this section. This is a novel work that has not been discussed previously. Covariance functions are generally dependent on some parameter set, for instance in the case of explanatory variable that is one-dimensional, the squared exponential covariance function is given by

$$a\exp\left\{-\frac{d}{2}\left(\mathrm{z}_i-\mathrm{z}_j\right)^2\right\}$$

with the parameter set being $\{a, d\}$. It follows that the covariance function defining the noisy data is

$$C\left(\mathrm{z}_i,\mathrm{z}_j\right)=a\exp\left\{-\frac{d}{2}\left(\mathrm{z}_i-\mathrm{z}_j\right)^2\right\}+b\delta_{ij} \tag{26}$$

where b is the noise variance hyperparameter and δ_{ij} is the Kronecker delta function. As mentioned earlier, the training procedure consists of $O(N^3)$ operations. This section demonstrates how a simple modification to the log-likelihood function and adapting the covariance function to it can speed up the optimisation routine.

Let $b = an$, the covariance function (26) is re-written to the form

$$C\left(\mathrm{z}_i,\mathrm{z}_j\right)=a\left[\exp\left\{-\frac{d}{2}\left(\mathrm{z}_i-\mathrm{z}_j\right)^2\right\}+n\delta_{ij}\right]$$

Next, denote $C = aC_n$, where C_n is a normalised covariance function. It follows that the negative log-likelihood function, Λ, can be written in terms of P, the covariance matrix for the covariance function C_n.

$$\Lambda(\theta) = \frac{1}{2}\log|P(d,n)| + \frac{1}{2a}\mathbf{Y}^{\mathrm{T}}P(d,n)^{-1}\mathbf{Y} + \frac{N}{2}\log a \tag{27}$$

where |.| is the determinant operator and N is the dataset size. The gradient with respect to a hyperparameter, θ, is zero at the turning point. Thus, the partial derivative of the log-likelihood function with respect to a is obtained and equated to zero, i.e., $\frac{\partial\Lambda(\theta)}{\partial a} = 0$. Hence,

$$a = \frac{\mathbf{Y}^{\mathrm{T}}P(d,n)^{-1}\mathbf{Y}}{N} \tag{28}$$

Substituting the solution back into (27), and removing non-hyperparameter terms and redundant factors, the revised log-likelihood function is formulated as

$$\overline{\Lambda}(\theta) = \log|P(d,n)| + N\log\left|\mathbf{Y}^{\mathrm{T}}P(d,n)^{-1}\mathbf{Y}\right| \tag{29}$$

Subsequently, the derivative with respect to the hyperparameter, θ_i, is

$$\frac{\partial\overline{\Lambda}(\theta)}{\partial\theta_i} = tr\left[P^{-1}\left(\frac{\partial P}{\partial\theta_i}\right)\right] - \frac{N}{\mathbf{Y}^{\mathrm{T}}P^{-1}\mathbf{Y}}\left[\mathbf{Y}^{\mathrm{T}}P^{-1}\left(\frac{\partial P}{\partial\theta_i}\right)P^{-1}\mathbf{Y}\right]$$

where tr(.) is the trace operator. The Hessian information is then derived as follow,

$$\frac{\partial^2\overline{\Lambda}(\theta)}{\partial\theta_i\partial\theta_j} = tr\left[P^{-1}\left(\frac{\partial^2 P}{\partial\theta_i\partial\theta_j}\right) - P^{-1}\left(\frac{\partial P}{\partial\theta_i}\right)P^{-1}\left(\frac{\partial P}{\partial\theta_j}\right)\right]$$
$$-\frac{N}{\left[\mathbf{Y}^{\mathrm{T}}P^{-1}\mathbf{Y}\right]^2}\left[\mathbf{Y}^{\mathrm{T}}P^{-1}\left(\frac{\partial P}{\partial\theta_i}\right)P^{-1}\mathbf{Y}\right]\left[\mathbf{Y}^{\mathrm{T}}P^{-1}\left(\frac{\partial P}{\partial\theta_j}\right)P^{-1}\mathbf{Y}\right]$$
$$-\frac{N}{\mathbf{Y}^{\mathrm{T}}P^{-1}\mathbf{Y}}\left[\mathbf{Y}^{\mathrm{T}}P^{-1}\left\{\frac{\partial^2 P}{\partial\theta_i\partial\theta_j} - 2\left(\frac{\partial P}{\partial\theta_j}\right)P^{-1}\left(\frac{\partial P}{\partial\theta_j}\right)\right\}P^{-1}\mathbf{Y}\right]$$

Discussion 3.1 *(Solution for a Maximum Point)*: Note that (28) is a solution for a maximum point. The proof is illustrated by substituting (27) and (28) into the equation above.

$$\begin{aligned}\frac{\partial^2 \Lambda(\theta)}{\partial a^2} &= \frac{\partial}{\partial a}\left(\frac{\partial \Lambda(\theta)}{\partial a}\right) \\ &= \frac{\partial}{\partial a}\left\{\frac{\mathbf{Y}^{\mathrm{T}} P^{-1}\mathbf{Y}}{2a^2} - \frac{N}{2a}\right\} \\ &= \frac{N}{2a^2} - \frac{\mathbf{Y}^{\mathrm{T}} P^{-1}\mathbf{Y}}{a^3} \\ &= \frac{1}{2a^2}\left[N - \frac{2\left(\mathbf{Y}^{\mathrm{T}} P^{-1}\mathbf{Y}\right)}{a}\right] \\ &= \frac{1}{2a^2}\left[N - 2N\right] < 0, \quad \forall a \in \mathrm{P}^+, \forall N \in \mathrm{Z}^+\end{aligned}$$

Hence, the solution is a maximum point. The optimisation of $\overline{\Lambda}$ is purely dependent on *d* and *n*.

Unlike §3.2.2, Hessian implementation may not necessarily be as efficient when applied on the revised log-likelihood function (29). This is due to additional computations of matrix-matrix tensor products required for the revised log-likelihood function. The performance of using hyperparameter reduction is investigated in the next sub-section.

Discussion 3.2 *(Dataset with scalar explanatory variable)*: Assume a dataset with scalar explanatory variable, i.e. $z \in \mathrm{P}$, is available and that the lengthscale hyperparameter, *d*, is fixed (scalar). From (29), the partial derivative with respect to *n* is reduced to

$$\frac{\partial \overline{\Lambda}(n)}{\partial n} = n\left\{tr\left(P^{-1}\right) - \frac{N}{\mathbf{Y}^{\mathrm{T}} P^{-1}\mathbf{Y}}\left[\mathbf{Y}^{\mathrm{T}} P^{-1} P^{-1}\mathbf{Y}\right]\right\} \tag{30}$$

3.3.1 Experimental Result

The hyperparameter reduction method is an appealing approach for low-dimensional datasets, e.g. datasets with explanatory variable that is one-dimensional. The advantage is clearly evident since the optimisation routine trains only two hyperparameters; instead of three. As the dimension increases, the number of hyperparameters required to be adapted also increases. As a result, the effect of the

hyperparameter reduction becomes insignificant as the dimension of the explanatory variable becomes large, i.e. two-dimensional and above.

To compare the performance of using the modified log-likelihood function from the standard log-likelihood function, a simple experiment is carried out on ten samples of the time-series data, with each sample having different data size, N, such that $N = \{200, 400, \ldots, 2000\}$. Gaussian regression is applied on these ten samples, whereby different noise data is introduced in each sample. In addition, the initial values for the optimisation routine are chosen to be different for every sample. The timing and number of iterations for convergence are tabulated in TABLE II. Note that, Hessian information is supplied in both cases.

TABLE II illustrated that by using hyperparameter reduction approach, the number of iterations required for convergence is greatly reduced; that is, at least ten times fewer as many recurrences to train the hyperparameters as the standard approach. Consequently, the convergence is much faster. It is apparent that by eliminating the requirement to train an additional hyperparameter, it reduces the optimisation complexity, therefore speeding up the process.

TABLE II Performance comparison between optimisations using standard and revised log-likelihood function

Result	Iterations		Timing (minutes)	
Dataset size, N	3-hyp	2-hyp	3-hyp	2-hyp
200	82.8	8.3	0.08	0.01
400	153.3	6.9	0.87	0.05
600	199.9	8.5	3.00	0.15
800	191.5	7.5	5.36	0.26
1,000	164.2	9.7	8.21	0.64
1,200	161.2	8.1	12.89	0.84
1,400	121.7	6.9	14.39	1.05
1,600	170.4	6.7	30.51	1.55
1,800	159.1	6.5	36.46	1.95
2,000	116.5	7.1	36.29	2.88

3-hyp refers to the standard optimisation techniques using the negative log-likelihood function (5) and adapting three hyperparameters of the squared exponential covariance function, whereas the 2-hyp refers to the use of revised log-likelihood function (29), in which two hyperparameters are adapted to maximise the function. Both cases are performed with user-supplied Hessian information. Experiment is carried on an Intel® Pentium® IV 3.0GHz machine with 512MB RAM.

The hyperparameter reduction technique is particularly useful for one-dimensional dataset. Dataset with explanatory variable that is more than one dimension may not benefit much from this approach, such that its benefit reduces as the number of training hyperparameters increases. The evidence is shown in TABLE III, where a similar experiment is conducted on datasets with explanatory variable that is two-dimensional.

TABLE III Performance comparison between optimisations using standard and revised log-likelihood function on dataset with explanatory variable that is two-dimensional.

Result	Iterations		Timing (hour)	
Dataset size, *N*	4-hyp	3-hyp	4-hyp	3-hyp
484	15.7	18.3	0.005	0.010
787	22.2	15.0	0.022	0.036
1,156	28.0	23.0	0.075	0.122
1,600	42.8	23.5	0.293	0.300
2,116	40.1	21.5	1.077	0.689
2,704	38.9	25.5	3.550	3.740

3-hyp refers to the standard optimisation techniques using the negative log-likelihood function (5) and adapting three hyperparameters of the squared exponential covariance function, whereas the 2-hyp refers to the use of revised log-likelihood function (29), in which two hyperparameters are adapted to maximise the function. Both cases are performed with user-supplied Hessian information. Experiment is carried on an Intel® Pentium® IV 3.0GHz machine with 512MB RAM.

Apparently, from the table, the hyperparameter reduction approach is perhaps more suitable for datasets with explanatory variable that is one-dimensional.

3.4 Hyperparameter Initialisation

Hyperparameter initialisation is another novel idea that can be applied on Gaussian regression to speed up optimisation routine. However, the condition is much more restrictive, as it can only be applied to time-series datasets. As discussed in Chapter 2.5 on model selection, the minimisation of the negative log-likelihood function (5) is not a simple convex problem; multiple local minima exist within the log-likelihood space mapping. These local minima can be associated with different aspects of the

data. For example, a time-series data consisting of a long lengthscale component and a short lengthscale component, one minimum may correspond to the former and the other corresponds to the latter (see Chapter 4.2 for more detail). Depending on the choice of initial values for the hyperparameters, the outcome of the optimisation could be a model of either. To acquire faster optimisation of the training procedure and proper values of the hyperparameters, it is essential that appropriate initial values are chosen for the optimisation routine. A procedure for doing so is presented in this section.

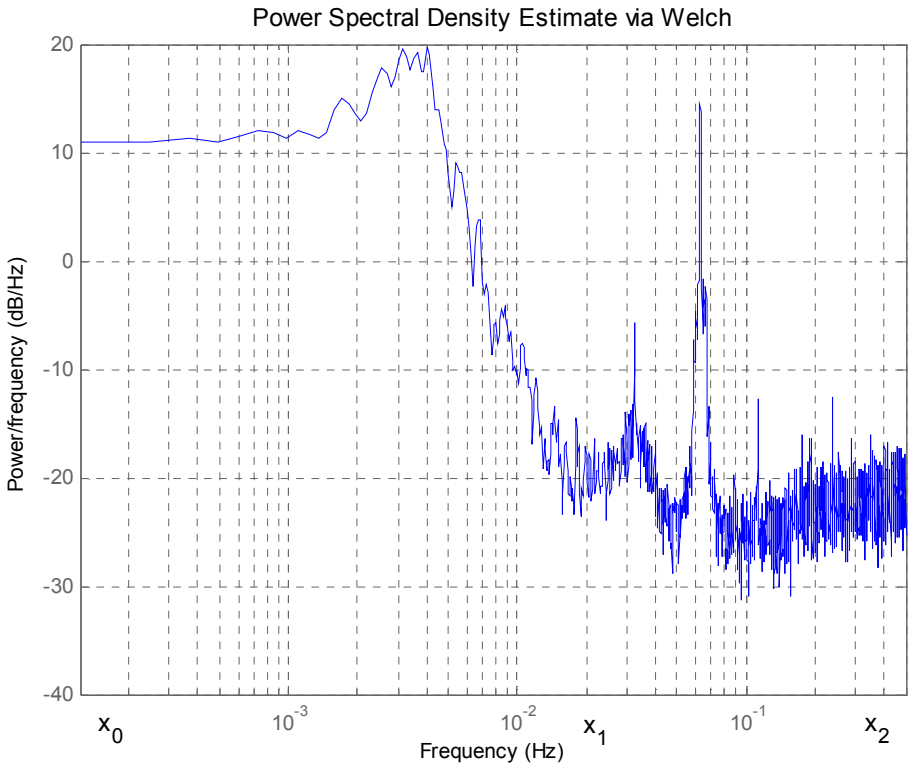


Figure 7 Power spectrum of a simple data.

Suppose that the mean of the time-series data is zero (if not it can always be made to be zero). The initial values of the hyperparameters $\theta = \{a, d, b\}$ for the covariance function (26) are determined by the following procedure, using data with the power spectral density, as shown in Figure 7, as an illustration.

Procedure 3.1 *(Hyperparameter initialisation procedure)*:

1. Provided the time-series data is of sufficient length, its variance is roughly equal to (a + b), since the noise hyperparameter is $b = \mathrm{E}\left[\varepsilon_i, \varepsilon_j\right]$ and the amplitude hyperparameter is $a = \mathrm{E}\left[\mathrm{f}_{t_i}, \mathrm{f}_{t_j}\right] = \dfrac{\mathbf{Y}^{\mathrm{T}}\mathbf{Y}}{N}$, where $\{\varepsilon_i\}_{i=1}^{N}$ denotes the noise data. Let ς_y and ς_n, respectively, be the variances of the measured data and the measurement noise. It follows that, since $b = an$,

$$v = \frac{\varsigma_n}{\varsigma_y} \approx \frac{b}{a+b} = \frac{n}{1+n}$$

2. The value for ς_y is easily estimated. Since different values of the hyperparameters, especially the lengthscale hyperparameter, correspond to models with different lengthscale, the value of ς_n depends on the choice of time-series components that is interpreted to be noise. For example, the spectral density in Figure 7 clearly indicates that the corresponding time-series data consists of several components with different lengthscales. Only the long lengthscale component with frequency less than χ_1 might be of interest whereas all remaining components with higher frequency are interpreted as noise. In this case, ς_n would be estimated as the cumulative sum of the spectrum between χ_1 and χ_2, the Nyquist rate. Hence, a^* and n^*, the respective initial values for a and n, are obtained from

$$a^* = (1-v)\varsigma_y$$
$$n^* = v/(1-v)$$

3. Let $Q(\theta) = aP(d,n)$ in (5). The negative log-likelihood function becomes (27). The hyperparameter a can be eliminated from $\Lambda(\theta)$ by minimising the value of a as a function of d and n, as shown in (28) and (29). The revised log-likelihood function is thus reformulated to be dependent on only two hyperparameters, i.e. d and n.
4. The initial value, d^*, for the lengthscale hyperparameter, d, is obtained by solving the nonlinear equation (28).

The hyperparameter values a^*, d^* and n^*, obtained by Procedure 3.1 are appropriate initial values for minimising the negative log-likelihood function, for either (5) or (29). The latter has the advantage of being dependent on two hyperparameters and so

converges faster (details are covered in §3.3). Two cases arise as a result from this procedure. In the first, all hyperparameters are adjusted during the optimisation to converge on a nearby local minimum corresponding to the prior model with the required lengthscale characteristic. In the second, the optimisation may fail to locate a suitable local minimum, when all the hyperparameters are adjusted. In the latter situation, d^* is required to be held constant during the optimisation. It may then be necessary to adjust manually the value of d^* and repeat the optimisation to obtain the prior model with the required lengthscale characteristic.

3.5 Efficient, Fast Algorithms for Time-series Gaussian Processes

In the previous few sections, Gaussian process computations are sped up by altering the training procedures, viz. Hessian implementation, hyperparameter reduction and hyperparameter initialisation. Subsequent discussions are focused on developing fast algorithms for time-series data. Note that, the fast algorithms introduced in this section are not approximation methods; unlike fast sparse approximation approaches reviewed by Quiñonero-Candela and Rasmussen (2005).

Given that a data is time-series, using the squared-exponential covariance function (26), it can be shown (Sayed and Kailath, 1994; Sayed et al., 1994) that the corresponding covariance matrix is structured, i.e. a Toeplitz (or quasi-Toeplitz) matrix. Structured matrices are known to have low displacement rank and therefore can be exploited to speed up computations.

The interest of this chapter is to acquire fast algorithms that are capable of dealing with large-scale time-series data; or specifically covariance matrices that are Toeplitz or block-Toeplitz type. As such, matrix manipulation is essential and the following algorithms are briefly evaluated. Firstly, some super fast algorithms (Wang and Krishna, 1989; Stewart, 2003) to solve block-Toeplitz systems in near $O(N)$ operation exist. However, these approaches remain numerically unstable and therefore are not worthwhile using. The second method, introduced by Sayed and Kailath (1994), is the Kalman filtering algorithm, which is based on the discrete-time Riccati recursion to

handle time-invariant data. However, it requires a very special structure in the Toeplitz matrix (Sayed et al., 1994), which does not correspond to the required covariance function in this case, and hence is rather restrictive in nature. This approach is therefore also not viable. Finally, the third method is to utilise the Kalman filtering algorithm and adapting it to suit the required covariance function; that is, the modified generalised Schur algorithm.

Two fast algorithms are discussed in this section; the modified Durbin-Levinson's algorithm and the modified generalised Schur algorithm. The former, based on Durbin's, Levinson's and Trench's algorithms, is developed by Leithead et al. (2005c) and provides the motivation to develop the latter algorithm. The modified Durbin-Levinson's algorithm has the potential to perform Gaussian regression on a very large-scale dataset, e.g. up to one million data points and beyond. However, this algorithm is applicable only to strictly Toeplitz matrices, or to be exact, covariance matrices constructed from time-series data with fixed sampling interval. This limits the use of the modified Durbin-Levinson's algorithm to special cases of structured matrices.

In many cases, time-series datasets may contain missing measurement data. As a result, the covariance matrix is no longer pure Toeplitz; instead it is a Toeplitz-like matrix, i.e. a Toeplitz-Block-Toeplitz matrix. Despite this, the matrix itself is still structured. It is known that the generalised Schur algorithm is capable of factoring general structured matrices (Sayed et al., 1994), hence this fact is readily used to extend the modified generalised Schur algorithm to handle Toeplitz-like matrices, or rather, matrices with low displacement rank. Typically, covariance matrix of time-series data with missing gaps has low displacement rank, K. The performance of the modified generalised Schur algorithm is not as fast as the Durbin-Levinson's algorithm, but it is an $O(KN^2)$ operation, where K is much smaller than N, e.g. the value of K is approximately less than 10, as compared to N.

3.5.1 Modified Durbin-Levinson's Algorithm

Developed to reduce $O(N^3)$ computational complexity and $O(N^2)$ memory storage requirement, the modified Durbin-Levinson's algorithm (Leithead et al., 2005c) specifically exploits the Toeplitz structure of the covariance matrix (26) for a time-series dataset. It can be shown that the derivative for the covariance matrix is also a symmetrical Toeplitz matrix, of the form

$$Q = \begin{pmatrix} q_0 & q_1 & q_2 & \cdots & q_{N-1} \\ q_1 & q_0 & q_1 & \ddots & \vdots \\ q_2 & q_1 & q_0 & \ddots & q_2 \\ \vdots & \ddots & \ddots & \ddots & q_1 \\ q_{N-1} & \cdots & q_2 & q_1 & q_0 \end{pmatrix}$$

which can be represented by its first column vector, $q = \begin{bmatrix} q_0 & q_1 & q_2 & \cdots & q_{N-1} \end{bmatrix}^{\mathrm{T}}$. Note that the inverse of Q is not necessarily a Toeplitz matrix. The matrix operations needed in Gaussian regression can be classified into three different $O(N^3)$ operations; specifically, the log-determinant of Q, computation of $Q^{-1}\mathbf{Y}$ and trace of $(Q^{-1}P)$, where P is also a Toeplitz matrix. According to Golub and Van Loan (1996), Trench's algorithm inverts Q with $13N^2/4$ operations whereas Levinson's algorithm can solve $Q^{-1}\mathbf{Y}$ with $4N^2$ operations. However, explicitly solving Q^{-1} does not work due to extremely large memory requirement to store the matrix. A Durbin-Levinson framework was presented (Leithead et al., 2005c) to adapt the algorithms towards an efficient and economical computational scheme. The key to solving memory demand issue is by using the concept of vector-level storage. This framework is known to handle very large time-series datasets for Gaussian processes, e.g. one million data points. Experiments are carried out in §3.7 to illustrate that the modified Durbin-Levinson's algorithm is capable of handling very large datasets through $O(N^2)$ computational complexity and $O(N)$ storage requirement.

Due to its limitation in handling Toeplitz-like matrices, the modified Schur algorithm is developed.

3.5.2 Modified Generalised Schur Algorithm

The modified Durbin-Levinson's algorithm, developed by Zhang et al. (2005c), has its limitation. Primarily, it is incapable of handling more general structured matrices, such as Toeplitz-like matrices with low displacement ranks.

As outlined in the previous section, the two main procedures involving $O(N^3)$ operations are the matrix inversion and the log-determinant of the covariance matrix, Q. Most of the matrix operations, including the calculation of the log-determinant of Q, can be solved using the generalised Schur algorithm, through the computation of the Schur's complement of any N x N Hermitian (Toeplitz-like) matrix, P. This was shown by Kailath (1999), Chandrasekaran and Sayed (1996, 1999a, 1999b), such that the generalised Schur algorithm can be extended for the application of Gaussian processes.

The covariance matrix (26) for time-series data has a special displacement structure, that is, a matrix with a low displacement rank. In this section, the focus is mainly on the exploitation of Toeplitz, or rather semi-block-Toeplitz (Toeplitz-like) matrices using Schur algorithm to reduce computational burden.

Consider a positive-definite Hermitian matrix $\mathrm{P} \in \mathrm{X}^{N \times N}$, such that the triangular decomposition is denoted by

$$\mathrm{P} = \Lambda \Delta^{-1} \Lambda^{*} \tag{31}$$

where $\Delta = diag\{d_0, ..., d_{N-1}\}$ is a diagonal matrix and Λ^{*} refers to the complex conjugate of Λ. The lower triangular matrix Λ is normalised in a way such that $\{d_i\}$ appears on its main diagonal. This decomposition can be obtained through the Schur reduction algorithm, also known as the Gaussian elimination procedure, in a recursive manner to yield the so-called LDL-decomposition. Schur reduction is also known as the matrix factorisation procedure. Throughout the thesis, P is assumed to be real. The complex case can be treated in a similar way.

The Schur reduction algorithm to factor P is generally $O(N^3)$. However, when P possesses a special displacement structure, the computation burden can inherently be significantly reduced by exploiting the low displacement rank of P. The key procedure lies in the triangularisation of a matrix by a sequence of *J*-unitary operations of a prearray formed from the data at certain iteration; that is, the information needed to form the prearray for the next iteration can be read out from the entries of the triangularised prearray at the current iteration. Hence, no explicit equation, except a few simple ones, is required.

The Schur algorithm focuses exclusively on strongly regular Hermitian Toeplitz-like matrices that satisfy

$$\mathrm{P} - \Phi \mathrm{P} \Phi^{*} = \Gamma \vartheta^{-1} \Gamma^{*} \qquad \vartheta = \vartheta^{*}, \vartheta^{2} = I \tag{32}$$

for some full rank generator matrix, Γ, and lower triangular matrix, Φ. The diagonal elements of Φ satisfy

$$1 - f_i f_j^{*} \neq 0 \tag{33}$$

where f_j^{*} is the complex conjugate of f_j so that P can be defined uniquely by ϑ, Γ and Φ. ϑ is known as the signature matrix, defined to be *J*-unitary, $\left(I_p \oplus -I_q\right)$, where $K = p + q$ is the total number of real eigen-values for any full rank Γ. Scalars p and q refers to the number of positive and negative eigen-values, respectively. Φ is chosen to be strictly lower-triangular shift matrix in this thesis. A condensed form of the generalised Schur algorithm (Kailath, 1999) is shown below.

Algorithm 3.1 (Generalised Schur algorithm): Given a matrix, $\mathrm{P} \in \mathrm{X}^{N \times N}$, that satisfies (32) and (33) for some full rank $\Gamma \in \mathrm{X}^{N \times K}$. Start with $\Gamma_0 = \Gamma$ and perform the following steps for $i = 0,1,\ldots,n\text{-}1$:

1. Let g_i be the top row of Γ_i, and partition Φ as

$$\Phi = \begin{bmatrix} \overbrace{\overline{\Phi}_i}^{i} & \overbrace{0}^{n-i} \\ \widetilde{\Phi}_i & \Phi_i \end{bmatrix} \begin{matrix} \} & i \\ \} & n-i \end{matrix}$$

   The object of interest here is Φ_i, which is obtained by ignoring the first i rows and columns of Φ and define f_i to be the top left element of Φ_i.

Next, compute l_i by solving the linear system of equations[3] where g_i^* is the complex conjugate transpose of g_i:

$$\left(I_{n-i} - f_i^* \Phi_i\right) l_i = \Gamma_i \vartheta g_i^*$$

Defining Φ to be strictly lower-triangular shift matrix, it follows that

$$l_i = \Gamma_i \vartheta g_i^*$$

Then, the first element of l_i is

$$d_i = g_i \vartheta g_i^* / \left(1 - f_i f_i^*\right) = g_i \vartheta g_i^*$$

2. Obtain an explicit form of Γ_{i+1} as shown

$$\begin{bmatrix} 0 \\ \Gamma_{i+1} \end{bmatrix} = \left\{ \Gamma_i + \left(\Phi_i - I_{n-1}\right) \Gamma_i \frac{\vartheta g_i g_i^*}{g_i \vartheta g_i^*} \right\} \Theta_i$$

where Θ_i is any J-unitary matrix. Notice that Γ_{i+1} has one row less that Γ_i.

3. Finally, $\{l_i\}$ defines the successive columns of Λ, and d_i, the successive diagonal elements of Δ, such that $P = \Lambda \Delta^{-1} \Lambda^*$.

The generator matrix, Γ, being highly non-unique, can be obtained in various forms in which the Schur algorithm applies. Furthermore, it is possible to obtain different arbitrary J-unitary matrix, Θ_i, as long as the following condition applies

$$\Theta \vartheta \Theta^* = \vartheta = \Theta^* \vartheta \Theta \tag{34}$$

Given the flexibility in choosing the generator matrix, $\Gamma\Theta$ can be used as any J-unitary matrix to obtain the proper form of the algorithm. Γ is said to be in proper form if its first nonzero row has only a single nonzero entry, either in the first or last column of the top row of Γ. These issues have been addressed by Kailath (1999), Chandrasekaran and Sayed (1999a, 1999b).

Generally, representing structured matrices by Γ and Φ in finite precision induces round-off errors. It is important that Γ should be in proper form so that reasonably good error bound is achieved. To do so, the following four enhancements (Chandrasekaran and Sayed, 1999a) are incorporated in the Schur algorithm.

[3] Φ is not only a triangularly matrix, it is generally sparse and often diagonal or bidiagonal, thus it makes the equation fairly easy to solve, particularly for our case $\Phi = Z$ or $\Phi = Z \oplus Z$.

1. Careful implementation of the hyperbolic rotation[4].
2. Careful implementation of the Blaschke-vector product[5].
3. Enforce positive-definiteness of successive Schur complements.
4. Control of potential growth of successive generator matrices.

Chandrasekaran and Sayed (1999a) concluded that for most positive-definite structured matrices, the modified Schur algorithm is backward stable, when enhanced with the hyperbolic rotation and householder (or any related) transformation.

The determination of the displacement-generating matrices $\{\Gamma, \Phi, \vartheta\}$, also known as the rank-revealing decomposition, is discussed here. Note that, the procedure to obtain these matrices takes into account the context of Gaussian regression. Nevertheless, it is also applicable to some non Gaussian regression contexts. The displacement-generating matrices require the use of augmentation matrices, P, for the generalised Schur algorithm. The matrix P in (32) can be assumed to be real, symmetric and positive-definite. The displacement matrix, ΔP, is defined as $\Delta \mathrm{P} = \mathrm{P} - \Phi \mathrm{P} \Phi^{\mathrm{T}}$ and Φ^{T} is the matrix transpose of Φ. Since P is positive-definite, it is crucial that the successive Schur complements are also theoretically positive-definite. If the displacement rank of ΔP is K, then ϑ is simply defined to be a J-unitary matrix,

$$\vartheta = \left(I_{K/2} \oplus -I_{K/2} \right)$$

such that the numbers of positive and negative eigenvalues are the same. The matrix, $\Phi = \left(\mathrm{Z}_{N_1} \oplus \mathrm{Z}_{N_2} \oplus \mathrm{Z}_{N_3} \oplus \ldots \right)$, is designated to be a strictly lower-triangular shift matrix, depending on the number of inner Toeplitz-blocks inside P. For example, if P consists of two by two block-Toeplitz matrices as shown below, such that T is a Toeplitz matrix, then $\Phi = \left(\mathrm{Z}_{N_1} \oplus \mathrm{Z}_{N_2} \right)$.

$$\mathrm{P} = \left[\begin{array}{c|c} \mathrm{T}_{aa} & \mathrm{T}_{ab}^{\mathrm{T}} \\ \hline \mathrm{T}_{ab} & \mathrm{T}_{bb} \end{array} \right] \qquad , \mathrm{T}_{aa} \in \mathrm{P}^{N_1 \times N_1}, \mathrm{T}_{bb} \in \mathrm{P}^{N_2 \times N_2}, \mathrm{T}_{ab} \in \mathrm{P}^{N_2 \times N_1}$$

Matrix Z_N is defined here to be a square lower-triangular shift matrix with ones on the first subdiagonal and zeros elsewhere (i.e. a lower-triangular Jordan block with

[4] Hyperbolic rotation is implemented to rotate the top row of Γ_{i+1} to proper form.
[5] Blaschke-vector product has been explained in the past literature (Chandrasekaran and Sayed, 1999a).

eigenvalue equal to zero). The generator matrix, Γ, can be obtained by the following procedure.

Procedure 3.2 *(Obtaining the generator matrix)*:

1. Let $\mathrm{P} \in \mathrm{P}^{N\times N}$ be a symmetrical and Hermitian matrix, with low displacement rank, $K << N$, such that the reduced-row echelon form (RREF) is

$$\Delta\mathrm{P} = \left[\begin{array}{c|c} \widetilde{\mathrm{A}} & \widetilde{\mathrm{B}}^{\mathrm{T}} \\ \hline \widetilde{\mathrm{B}} & 0 \end{array}\right] \equiv \left[\mathrm{B} \;\vdots\; 0\right] + \left[\begin{array}{c} \mathrm{B}^{\mathrm{T}} \\ \hline 0 \end{array}\right] - \left[\begin{array}{c|c} \mathrm{A} & 0 \\ \hline 0 & 0 \end{array}\right]$$

   where $\mathrm{B}^{\mathrm{T}} = \left[\widetilde{\mathrm{A}} \;\vdots\; \widetilde{\mathrm{B}}^{\mathrm{T}}\right]$ and $\mathrm{A} = \widetilde{\mathrm{A}} \in \mathrm{P}^{K/2\times K/2}$ is symmetric. Any symmetric Hermitian block-Toeplitz matrix can be transformed into a matrix with the above RREF by permuting its rows and columns. Let $\Phi E = ED$ be the eigen-value decomposition of

$$\Phi = \left[\begin{array}{c|c} \mathrm{A} & I \\ \hline \mathrm{B}^{\mathrm{T}}\mathrm{B} - \mathrm{AA} & 0 \end{array}\right]$$

   where D is a diagonal matrix. The non-zero eigenvalues of ΔP are real and positive, and are identical to the eigenvalues of Φ. In addition, the eigenvectors for the non-zero eigenvalues of ΔP are real and equal to the columns of $\mathrm{B}X + \left[I \;\vdots\; 0\right]^{\mathrm{T}} Y = \Psi \in \mathrm{P}^{N\times K}$ where $\left[X^{\mathrm{T}} \;\vdots\; Y^{\mathrm{T}}\right]^{\mathrm{T}} = E \in \mathrm{P}^{K\times K}$. The proof is explained in Proof 3.1.

2. Some eigenvalues of Γ can be very similar. For numerical reasons, the computed eigenvalues and eigenvectors may consist of complex conjugate pairs, i.e. D and E are complex matrices. Although the imaginary part of these computed eigenvalues are extremely small, the imaginary parts of the eigen vectors can be large. Hence, to ensure D and E are real, the following corrections are made

$$D \to \mathrm{real}(D)$$
$$E \to \mathrm{real}(E) + \mathrm{imag}(E)$$

3. Each column of Ψ is an eigenvector of ΔP with eigenvalue belonging to the diagonal elements of D. However, since all the eigenvalues are not distinct, the columns of Ψ are not automatically orthonormal as required.

To enforce orthogonality, the columns of Ψ are updated recursively for $k = 2,\ldots,K$ such that,

$$\Psi_k \to \Psi_k - \sum_{i=1}^{k-1} \frac{\Psi_i \left(\Psi_i^{\mathrm{T}} \Psi_k\right)}{\Psi_i^{\mathrm{T}} \Psi_i}$$

where Ψ_i is the i^{th} column of Ψ. To obtain orthonormality, the columns are then rescaled such that, $\forall k$,

$$\Psi_k \to \frac{\Psi_k}{\sqrt{\Psi_k^{\mathrm{T}} \Psi_k}}$$

4. With $\Psi \in \mathrm{P}^{N \times K}$ obtained from above, every column of Ψ is an eigenvector of $\Delta\mathrm{P}$ and is orthonormal with every other columns. The respective diagonal elements of D are therefore the corresponding eigenvalues. Thus, the following is obtained, $\Delta\mathrm{P} \cong \Psi D \Psi^{\mathrm{T}}$ (Orthogonality is shown in Lemma 3.1). D has the decomposition

$$D = \Xi \left(H \Omega H^{\mathrm{T}}\right) \Xi$$

where H is the unitary permutation matrix separating the positive and negative eigenvalues, λ_i, from each other. Ξ is a diagonal matrix with its diagonal elements comprised of the square-root of the absolute values of the eigenvalues, λ_i; that is,

$$\Xi = diag\left\{\sqrt{|\lambda_1|},\ldots,\sqrt{|\lambda_K|}\right\}$$

The required decomposition of $\Delta\mathrm{P}$ is obtained with

$$\vartheta = \Omega$$
$$\Gamma = \Psi \Xi H$$

that is, $\Delta\mathrm{P} \cong \Gamma \vartheta \Gamma^{\mathrm{T}}$, where $\Gamma \underline{\underline{\Delta}} \Omega \Xi H$ is the generator matrix for $\Delta\mathrm{P}$.

Proof 3.1 (Eigenvalues and eigenvectors of Φ ≡ eigenvalues and related-eigenvectors of ΔP): Let $\mathrm{P} \in \mathrm{P}^{N \times N}$ be a Toeplitz-like symmetrical and Hermitian matrix with low displacement rank such that

$$\Delta\mathrm{P} = \mathrm{P} - \Phi \mathrm{P} \Phi^{\mathrm{T}}$$

where $\Phi = \left(Z_{N_1} \oplus Z_{N_2} \oplus Z_{N_3} \oplus \ldots\right)$ is a strictly lower-triangular shift matrix and Z_N is a square lower-triangular shift matrix with ones on the first subdigonal and zeros elsewhere. To prove that the eigenvalues of $\Delta\mathrm{P}$ are the eigenvalues of Φ, and the eigenvectors of $\Delta\mathrm{P}$ are related to the eigenvectors of Φ, let

$$\Delta \mathrm{P} = \left[\begin{array}{c|c} \widetilde{\mathrm{A}} & \widetilde{\mathrm{B}}^{\mathrm{T}} \\ \hline \widetilde{\mathrm{B}} & 0 \end{array}\right] \equiv \left[\mathrm{B} \mid 0\right] + \left[\begin{array}{c} \mathrm{B}^{\mathrm{T}} \\ \hline 0 \end{array}\right] - \left[\begin{array}{c|c} \mathrm{A} & 0 \\ \hline 0 & 0 \end{array}\right]$$

where $\mathrm{B}^{\mathrm{T}} = \left[\widetilde{\mathrm{A}} \mid \widetilde{\mathrm{B}}^{\mathrm{T}}\right]$ and $\mathrm{A} = \widetilde{\mathrm{A}} \in \mathrm{P}^{K/2 \times K/2}$ is symmetric. Also, let the eigenvector be $E = \mathrm{B}x + \left[\begin{array}{c} I \\ \hline 0 \end{array}\right] y$ and eigenvalue be λ. Thus,

$$\left[\begin{array}{c|c} \widetilde{\mathrm{A}} & \widetilde{\mathrm{B}}^{\mathrm{T}} \\ \hline \widetilde{\mathrm{B}} & 0 \end{array}\right] E = \left\{\left[\mathrm{B} \mid 0\right] + \left[\begin{array}{c} \mathrm{B}^{\mathrm{T}} \\ \hline 0 \end{array}\right] - \left[\begin{array}{c|c} \mathrm{A} & 0 \\ \hline 0 & 0 \end{array}\right]\right\} \left\{\mathrm{B}x + \left[\begin{array}{c} I \\ \hline 0 \end{array}\right] y\right\}$$

By expanding the right-hand side,

$$\mathrm{BA}x + \mathrm{B}y + \left[\begin{array}{c} \mathrm{B}^{\mathrm{T}}\mathrm{B} \\ \hline 0 \end{array}\right] x + \left[\begin{array}{c} \mathrm{A} \\ \hline 0 \end{array}\right] y - \left[\begin{array}{c} \mathrm{AA} \\ \hline 0 \end{array}\right] x - \left[\begin{array}{c} \mathrm{A} \\ \hline 0 \end{array}\right] y \equiv \lambda \left\{\mathrm{B}x + \left[\begin{array}{c} I \\ \hline 0 \end{array}\right] y\right\}$$

$$\Rightarrow \mathrm{BA}x + \mathrm{B}y + \left[\begin{array}{c} \mathrm{B}^{\mathrm{T}}\mathrm{B} \\ \hline 0 \end{array}\right] x - - \left[\begin{array}{c} \mathrm{AA} \\ \hline 0 \end{array}\right] x \equiv \lambda \left\{\mathrm{B}x + \left[\begin{array}{c} I \\ \hline 0 \end{array}\right] y\right\}$$

By comparing both sides,

$$\left.\begin{array}{l} \mathrm{BA}x + \mathrm{B}y = \lambda \mathrm{B}x \\ \mathrm{B}^{\mathrm{T}}\mathrm{B} - \mathrm{AA} = \lambda y \end{array}\right\} \Rightarrow \Phi \left[\begin{array}{c} x \\ \hline y \end{array}\right] = \left[\begin{array}{c|c} \mathrm{A} & I \\ \hline \mathrm{B}^{\mathrm{T}}\mathrm{B} - \mathrm{AA} & 0 \end{array}\right] \left[\begin{array}{c} x \\ \hline y \end{array}\right] = \lambda \left[\begin{array}{c} x \\ \hline y \end{array}\right]$$

As $\mathrm{B}^{\mathrm{T}}\mathrm{B} - \mathrm{AA}$ is full rank and non-singular, this further implies that the inverse of $\mathrm{B}^{\mathrm{T}}\mathrm{B} - \mathrm{AA}$ exists. Thus, by taking the eigenvalue decomposition, the eigenvalues and eigenvectors of Φ is simply the corresponding eigenvectors and eigenvalues of $\Delta \mathrm{P}$.

Lemma 3.1 *(Orthogonality and the SVD).* A set of vectors $\{x_1, ..., x_p\}$ in P^m is orthogonal if $x_i^{\mathrm{T}} x_j = 0$ whenever $i \neq j$ and orthonormal if $x_i^{\mathrm{T}} x_j = \delta_{ij}$. Orthogonal vectors are maximally independent as they point in different directions. The vectors $v_1, ..., v_k$ form an orthonormal basis for a subspace $S \subseteq \mathrm{P}^m$ if they are orthonormal and span *S*. It is possible to form an orthogonal matrix $Q \in \mathrm{P}^{m \times m}$ by extending such a basis to a full orthonormal basis $\{v_1, ..., v_m\}$ for P^m (Golub and Van Loan, 1996).

The development of orthogonal matrices can be useful in singular value decomposition. If A is a real m-by-m matrix, then there exist orthogonal matrices

$$U = [u_1,...,u_m] \in \mathrm{P}^{m\times m} \text{ and } V = [v_1,...,v_m] \in \mathrm{P}^{m\times m}$$

such that

$$U^{\mathrm{T}} AV = diag\{\sigma_1,...,\sigma_m\} \in \mathrm{P}^{m\times m}$$

where $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \geq 0$.

The σ_i are the singular values of A and the vectors u_i and v_i are the i^{th} left singular vector and i^{th} right singular vector respectively. It is easy to verify that

$$\left.\begin{aligned} Av_i &= \sigma_i u_i \\ A^{\mathrm{T}} u_i &= \sigma_i v_i \end{aligned}\right\} i = 1:m$$

Thus, the SVD expansion for matrix A (Golub and Van Loan, 1996) is obtained by

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^{\mathrm{T}}$$

If A is symmetrical, $u = v$, thus, the eigenvectors are the singular vectors of u and v, while the eigenvalues are the singular values of A.

3.6 Application of Schur Algorithm in Gaussian Processes

Following the previous subsection, it is of interest to exploit the generalised Schur algorithm to speed up the computations in Gaussian processes. Several $O(N^3)$ computations, such as log-determinant and inversion of covariance matrices, are bottlenecks in the training procedure. The modified Schur algorithm, as described in §3.5.2, is used to obtain fast factorisation for the variables in both log-likelihood function and its derivatives, with the chosen covariance function, (26). For notation simplicity, the first and second order derivatives of $Q(\theta)$ with respect to d, $\partial Q/\partial d$ and $\partial^2 Q/\partial d^2$, are denoted by Φ_1 and Φ_2, respectively. The hyperparameters in (26) are defined using exponentials, as discussed in §3.2.1, during the training process.

Note that from this section onwards, N refers to the size of Q, the covariance matrix (26), and $\overline{N}$ denotes the size of P (and Γ).

3.6.1 Fast Factorisation using Vector-level Storage Procedure

The utility of the generalised Schur algorithm provides a direct factorisation of the log-determinant of Q and $Q^{-1}\mathbf{Y}$ during the evaluation of the negative log-likelihood function. The terms, $\text{tr}(Q^{-1})$, $tr(Q^{-1}\Phi_1)$ and $\Phi_1 Q^{-1}\mathbf{Y}$ are necessary to complete the derivative function for the optimisation procedure. Vector $\mathbf{Y}$ is defined here to be the scalar outcomes of every explanatory variable $\mathbf{z}$. Hessian information, whilst optional, can also be included in the optimisation procedure. The second order derivative of the negative log-likelihood function requires further direct factorisation for $Q^{-1}Q^{-1}$, $Q^{-1}\Phi_1 Q^{-1}\Phi_1$, $tr\left(Q^{-1}\Phi_2\right)$ and $Q^{-1}\Phi_1 Q^{-1}$.

Although the generalised Schur algorithm provides a direct factorisation of a positive-definite matrix P in the form of $\Lambda\Delta^{-1}\Lambda^{\mathrm{T}}$, in actual fact no matrices, other than the generator matrix Γ, is explicitly stored in the process. Leithead et al. (2005c) introduced vector-level storage algorithm while handling large-scale data (Leithead et al., 2005d) using Durbin-Levinson's algorithm in Gaussian process, to avoid any $O(N^2)$ storage requirement. In this proposed Schur algorithm, the largest matrix to be stored is the generator matrix, $\Gamma \in \mathrm{P}^{\overline{N}\times K}$, but since $K << \overline{N}$, it is sufficiently small to be classified as vector-level storage.

3.6.2 Matrix Structure and Schur complements

Factorisation of the Schur complement is instrumental towards the reconstruction of matrices with low displacement rank. It is perhaps easier to first introduce some notation. Recalling from §3.5.2 for the real case of direct factorisation (31), $\mathrm{P} = \Lambda\Delta^{-1}\Lambda^{\mathrm{T}}$, it follows that there is a representation for its inverse such that

$$\mathrm{P}^{-1} = \left(\Lambda^{-\mathrm{T}}\Delta\Lambda^{-1}\right) = \Omega\Delta^{-1}\Omega^{\mathrm{T}}$$

where Ω is an upper triangular matrix, defined as $\Omega \triangleq \Lambda^{-T}\Delta$. To extend towards a general case of Schur factorisation, it is perhaps important to investigate the Schur factorisation and its complements in detail.

Suppose that P is an extended augmentation matrix[6], partitioned as shown below

$$\mathrm{P} = \begin{bmatrix} \overbrace{T_{11}}^{N} \overbrace{T_{12}}^{\bar{N}-N} \\ T_{21} T_{22} \end{bmatrix} \begin{matrix} \} & N \\ \} & \bar{N} - N \end{matrix}$$

where T_{ij}, $\forall \{i, j\} \in \{1,2\}$ are Hermitian Toeplitz-like matrices, it follows that the Schur complements of the leading N x N block of P is $T_{22} - T_{21} T_{11}^{-1} T_{12}$. This can be shown using the Schur reduction algorithm (Kailath, 1999). The size of T_{11} need not necessarily be the same as that of T_{22}. By applying the generalised Schur algorithm, not only can (the generators of) T_{11} be determined, but both T_{11} and $T_{22} - T_{21} T_{11}^{-1} T_{12}$, the Schur complement of P, can be simultaneously factorised. Note that, only the symmetrical case for real P is considered, though it can be extended to cases for non-Hermitian complex matrices.

To further clarify the procedure, N recursive iterations of the generalised Schur algorithm is first applied to a generator of the matrix P, to provide the first N columns and first N diagonal entries of the triangular factorisation of P, to be denoted by and Λ and Δ, respectively. The matrices, Λ and Δ, are partitioned as shown

$$\Lambda = \begin{bmatrix} \overbrace{\bar{L}}^{N} \vdots \overbrace{0}^{\bar{N}-N} \\ \cdots \quad \cdots \\ \tilde{L} \vdots \hat{L} \end{bmatrix} \begin{matrix} \} & N \\ \\ \} & \bar{N} - N \end{matrix}; \Delta = \begin{bmatrix} \overbrace{\bar{D}}^{N} \vdots \overbrace{0}^{\bar{N}-N} \\ \cdots \quad \cdots \\ \tilde{D} \vdots \hat{D} \end{bmatrix} \begin{matrix} \} & N \\ \\ \} & \bar{N} - N \end{matrix}$$

where $\bar{L}$ and $\hat{L}$ are lower triangular matrices. If $T_{21} = T_{12} = I$, $\tilde{L}$ would be an upper triangular matrix. In situations with extended matrices, the augmentation matrix may contain more semi-Toeplitz blocks within each partitioned block. Hence, $\tilde{L}$ may have to be split into

$$\tilde{L} = \begin{bmatrix} \mathrm{Y}^{\mathrm{T}} \vdots \varsigma^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

[6] Augmentation matrix, or simply known as extended matrix, may contain several semi-block-Toeplitz matrices, with low displacement rank.

such that Y and ς are full matrices. It follows from Kailath's (1999) explanation of the Schur reduction algorithm that P may be interpreted as follows

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} = \begin{bmatrix} \overline{L} \\ \widetilde{L} \end{bmatrix} \overline{D}^{-1} \begin{bmatrix} \overline{L}^{\mathrm{T}} & \widetilde{L}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \widehat{L}\widetilde{D}^{-1}\widehat{L} \end{bmatrix}$$

such that, following the Schur reduction algorithm, the Schur complement of P admits $\nabla_{\mathrm{P}} = \widehat{L}\widetilde{D}^{-1}\widehat{L}^{T} = T_{22} - T_{21}T_{11}^{-1}T_{12}$. Equating terms on both sides of the equation, it concludes that

$$T_{11} = \overline{L}\overline{D}^{-1}\overline{L}^{\mathrm{T}} \text{ and } T_{21}T_{11}^{-1}T_{12} = \widetilde{L}\overline{D}^{-1}\widetilde{L}^{\mathrm{T}}$$

Hence, the first N recursive steps of the algorithm not only provide the triangular factorisation of T_{11}, but also the triangular factorisation of $T_{21}T_{11}^{-1}T_{12}$. Unfortunately, this mathematical concept does not apply to a general matrix, Φ, e.g. where $\Phi = diag\{f_0,...,f_{N-1}\}$ is a diagonal matrix.

For simplicity, matrices Y and ς are used in the following section, but never explicitly stored; instead only every subsequent column vectors of Y and ς, viz. τ and ω (illustrated in the next few sections), are computed and temporarily stored in each iterative step. In this way, vector-level storage is implemented to resolve any preliminary $O(N^2)$ memory storage issue.

3.6.3 Useful Augmentation (Extended) Matrices in Gaussian regression

This section investigates the choice of augmentation matrices in more specific details, particularly in relation to Gaussian regression. Augmentation matrix, P, is used to obtain a fast factorisation for the Schur algorithm. The corresponding generator matrix, Γ, is then computed and stored. The choice of the lower-triangular shift matrix, Φ, is dependent on the choice of augmentation matrix, P, such that its choice satisfy (32) to within a reasonable degree of precision.

During the optimisation procedure in Gaussian regression, only the log-likelihood function and its derivative information are required to ensure successful convergence. The modified generalised Schur algorithm can be exploited to compute the following terms, which are typically $O(N^3)$-operations, if calculated explicitly.

- $Q^{-1}\mathbf{Y}$
- $\mathrm{tr}(Q^{-1})$
- $\log |Q|$
- $\Phi_1 Q^{-1}\mathbf{Y}$
- $\mathrm{tr}(Q^{-1}\Phi_1)$

Factoring (Q^{-1}) and $(\Phi_1 Q^{-1})$

The following extended matrix P can be used to obtain both $(\Phi_1 Q^{-1}\mathbf{Y})$ and $(Q^{-1}\mathbf{Y})$, by computing the direct factorisation of $Q^{-1}\Phi_1$ and Q^{-1},

$$\mathrm{P} = \left[\begin{array}{c|cc} -Q & \Phi_1 & I \\ \hline \Phi_1 & 0 & 0 \\ I & 0 & 0 \end{array}\right] \Rightarrow \nabla_{\mathrm{P}}^{1} = \begin{bmatrix} \Phi_1 Q^{-1}\Phi_1 & \Phi_1 Q^{-1} \\ Q^{-1}\Phi_1 & Q^{-1} \end{bmatrix} \tag{35}$$

where Q is a matrix with low displacement rank, as defined by the function in (26). The displacement rank of P can be easily calculated by working out the RREF of the displacement matrix, $\Delta\mathrm{P}$. For example, if both Q and Φ_1 are strictly Toeplitz matrices, such that P is quasi-Toeplitz, then the displacement rank of P is 4. ∇_{P}^{1} is denoted to be the Schur complement of the first (1,1) block of P, with corresponding LDL-decomposition given by

$$\nabla_{\mathrm{P}}^{1} = \begin{bmatrix} \Phi_1 Q^{-1}\Phi_1 & \Phi_1 Q^{-1} \\ Q^{-1}\Phi_1 & Q^{-1} \end{bmatrix} \equiv \begin{bmatrix} \mathrm{Y} \\ \varsigma \end{bmatrix} \overline{D}^{-1} \begin{bmatrix} \mathrm{Y} & \varsigma^{\mathrm{T}} \end{bmatrix}$$

such that ς is an upper-triangular matrix, with ones along its diagonal, and $\overline{D}$ a diagonal matrix. A direct factorisation for Q^{-1} and $Q^{-1}\Phi_1$ is simultaneously obtained after N iterations of the generalised Schur algorithm. It follows that the required trace operations is computable in $O(N^2)$ procedure,

$$tr\left(Q^{-1}\Phi_1\right)=\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{u_{ij}v_{ij}}{d_i}$$

$$tr\left(Q^{-1}\right)=\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{v_{ij}^2}{d_i}$$

where u_{ij} and v_{ij} represents the i^{th} columns and j^{th} rows of Y and ς, respectively, and d_i, the i^{th} element on the diagonal of $\overline{D}$. The trace operation is further explained in the algorithm below.

Algorithm 3.2 (Trace operations using Schur algorithms): The trace operations involving the generalised Schur algorithm are described in detail here. Given the generalised Schur algorithm as described in *Algorithm 3.1*, the matrices Y and ς are obtained column by column. Start with $\rho=0$ and $\psi=0$. The following steps are performed for i = 0,1,…, N-1:

1. Obtain l_i and d_i from the generalised Schur algorithm (*Algorithm 4.1*). Then, define $\overline{u}_i$ and $\overline{v}_i$ as shown:

$$l_i=\begin{bmatrix}\overline{u}_i\\ \hline \overline{v}_i\end{bmatrix}$$

2. Compute the following:

$$\rho=\rho+\frac{\overline{u}_i^{\mathrm{T}}\overline{v}_i}{d_i}\ \text{and}\ \psi=\psi+\frac{\overline{v}_i^{\mathrm{T}}\overline{v}_i}{d_i}$$

The final values of ρ and ψ are the corresponding values for the trace operations of $tr\left(Q^{-1}\Phi_1\right)$ and $tr\left(Q^{-1}\right)$, respectively.

Factoring log-determinant of Q

Similarly, the log-determinant of Q can be computed by

$$\log|Q|=\sum_{i=1}^{N}\log\|d_i\|$$

where ||.|| is an absolute operator to ensure d_i is positive. Vectors $Q^{-1}\mathbf{Y}$ and $\Phi_1 Q^{-1}\mathbf{Y}$ are obtainable in $O(N^2)$ fashion by

$$\tau_k = \sum_{i=k}^{N} \sum_{j=1}^{i} \frac{v_{ik} v_{ij} y_j}{d_i}$$

$$\omega_k = \sum_{i=1}^{N} \sum_{j=1}^{i} \frac{u_{ik} v_{ij} y_j}{d_i}$$

where τ_k and ω_k are the k^{th} entries of the respective vectors, $Q^{-1}\mathbf{Y}$ and $\Phi_1 Q^{-1}\mathbf{Y}$, and y_j is the j^{th} entry of the vector $\mathbf{Y}$. The operations to obtain these vectors are explained in the algorithm below. Upon obtaining the required two traces, two vectors and the log-determinant of Q, the log-likelihood function and its derivative information can be computed in $O(N^2)$ fashion, with a few more arithmetic and vector-vector products that are $O(N)$-operations.

Algorithm 3.3 (Obtaining vectors using Schur algorithms): The procedure to obtain the vectors τ_k and ω_k, involving the generalised Schur algorithm is described here. From the generalised Schur algorithm as described in *Algorithm 3.1*, the matrices Y and ς are again obtained column by column. Let $\mathbf{Y}$ be the vector consisting of the target values. Starting with vectors $\varphi = 0$ and $\varsigma = 0$, where $\varphi, \varsigma \in \mathrm{P}^{N\times 1}$, the following steps are carried out for $i = 0,1,\ldots, N\text{-}1$:

1. Obtain l_i and d_i from the generalised Schur algorithm (*Algorithm 3.1*). Then, define $\overline{u}_i$ and $\overline{v}_i$ as shown:

$$l_i = \begin{bmatrix} \overline{u}_i \\ \overline{v}_i \end{bmatrix}$$

2. Compute the following:

$$\varphi = \varphi + \frac{\overline{v}_i \bullet \left(\overline{v}_i \bullet \mathbf{Y}\right)}{d_i} \text{ and } \varsigma = \varsigma + \frac{\overline{u}_i \bullet \left(\overline{v}_i \bullet \mathbf{Y}\right)}{d_i}$$

where • denotes the Hadamard product or entrywise product.

The final values of φ and ς are the corresponding values for the operations of $(Q^{-1}\mathbf{Y})$ and $(\Phi_1 Q^{-1}\mathbf{Y})$, respectively.

3.6.4 Hessian Information in Optimisation Routine

Although (35) is sufficient for the optimisation routine, the inclusion of Hessian could provide a richer and more detailed analysis of additional information for training. Without user-supplied Hessian, the optimisation approximates Hessian by finite-differencing. Zhang and Leithead (2005) introduced the use of Hessian information in optimisation routine for Gaussian processes to provide more accurate and allow faster convergence towards a local minimum.

Subsequent to the results obtained in §3.6.3, Hessian manipulation requires a few additional terms with slightly different augmentation matrices, to have a complete description of the second order derivative information; mainly,

- $\Phi_2 Q^{-1}\mathbf{Y}$
- $tr(Q^{-1}\Phi_2)$
- $tr(Q^{-1}Q^{-1})$
- $Q^{-1}Q^{-1}\mathbf{Y}$
- $\Phi_1 Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$
- $\mathrm{tr}(\Phi_1 Q^{-1}\Phi_1 Q^{-1})$
- $\mathrm{tr}(Q^{-1}\Phi_1 Q^{-1})$

Factoring ($Q^{-1}\Phi_2$)

The terms $\Phi_2 Q^{-1}\mathbf{Y}$ and $\mathrm{tr}(Q^{-1}\Phi_2)$ are obtained using the previous augmentation matrix (35), by replacing the term Φ_1 with Φ_2. The procedure is, hence, trivial.

Remark 3.1: Before defining more augmentation matrices for Hessian factorisation, denote Λ_{2N} and Δ_{2N} by removing the first N rows and first N columns of Λ and Δ, respectively. The matrices Λ_{2N} and Δ_{2N} are partitioned as follow

$$\Lambda_{2N} = \begin{bmatrix} \overbrace{\bar{L}_{2N}}^{N} \vdots \overbrace{0}^{\bar{N}-N} \\ \cdots\cdots\cdots \\ \tilde{L}_{2N} \vdots \hat{L}_{2N} \end{bmatrix} \begin{matrix} \} & N \\ \} & \bar{N}-N \end{matrix}; \quad \Delta_{2N} = \begin{bmatrix} \overbrace{\bar{D}_{2N}}^{N} \vdots \overbrace{0}^{\bar{N}-N} \\ \cdots\cdots\cdots \\ \tilde{D}_{2N} \vdots \hat{D}_{2N} \end{bmatrix} \begin{matrix} \} & N \\ \} & \bar{N}-N \end{matrix}$$

where $\overline{L}_{2N}$ and $\widehat{L}_{2N}$ are lower triangular matrices.

Factoring ($Q^{-1}Q^{-1}$)

The following augmentation matrix is used to obtain a direct factorisation for $Q^{-1}Q^{-1}$,

$$\widehat{\mathrm{P}} = \left[\begin{array}{c|cc} I & Q & 0 \\ \hline Q & 0 & I \\ 0 & I & 0 \end{array}\right] \Rightarrow \nabla^1_{\widehat{\mathrm{P}}} = \begin{bmatrix} QQ & I \\ I & 0 \end{bmatrix} \Rightarrow \nabla^2_{\widehat{\mathrm{P}}} = Q^{-1}Q^{-1}$$

where $\nabla^1_{\widehat{\mathrm{P}}}$ and $\nabla^2_{\widehat{\mathrm{P}}}$ are successive Schur complements of the (1,1) and (2,2) blocks of P, after N and $2N$ iterations, respectively. Once again, the displacement rank of $\widehat{\mathrm{P}}$ is 4 when Q is Toeplitz. Although this augmentation matrix provides a very fast direct factorisation for $Q^{-1}Q^{-1}$, due to numerical issues, inverting QQ by Schur algorithm after first N iterations frequently causes highly inaccurate results. This is because the condition number of QQ is very large. It is vital to keep that condition number close to the condition number of Q, instead of QQ, if the generalised Schur algorithm is to be used. Hence, an alternative solution is to modify $\widehat{\mathrm{P}}$ to

$$\widehat{\mathrm{P}} = \left[\begin{array}{c|cc} Q+\mu I & Q & 0 \\ \hline Q & 0 & I \\ 0 & I & 0 \end{array}\right] \Rightarrow \nabla^1_{\widehat{\mathrm{P}}} = \begin{bmatrix} Q(Q+\mu I)^{-1}Q & I \\ I & 0 \end{bmatrix} \Rightarrow \nabla^2_{\widehat{\mathrm{P}}} = Q^{-1} + \mu Q^{-1}Q^{-1} \qquad (36)$$

where μ is a constant factor to be defined in §3.6.6. Despite the modification to the augmentation matrix, the displacement rank remains at 4. It follows that the corresponding (2,2) block of the Schur complement is the sum of Q^{-1} and $\mu Q^{-1}Q^{-1}$. Since the $tr(Q^{-1})$ and $Q^{-1}\mathbf{Y}$ are computable with the augmentation matrix (35) in §3.6.3, only the factorisation of $tr(Q^{-1}Q^{-1})$ and $Q^{-1}Q^{-1}\mathbf{Y}$ is of interest here. The latter are acquired directly via simple arithmetic $O(N)$ computations. The Schur complement can also be written in the form

$$\nabla^2_{\widehat{\mathrm{P}}} \cong \widetilde{L}_{2N}\overline{D}^{-1}_{2N}\widetilde{L}^{\mathrm{T}}_{2N}$$

where v_{ij} is the i^{th} column and j^{th} row of $\widetilde{L}_{2N} \in \mathrm{P}^{N\times N}$, an upper triangular matrix, and $\overline{D}_{2N}$ a diagonal matrix, constructed from N+1 to $2N$ iterations of the Schur algorithm. It follows that

$$tr\left(Q^{-1}Q^{-1}\right)=\frac{1}{\mu}\left[\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{v_{ij}^2}{d_i}-tr\left(Q^{-1}\right)\right]$$

$$\hat{\tau}_k=\frac{1}{\mu}\left[\sum_{i=k}^{N}\sum_{j=1}^{i}\frac{v_{ik}v_{ij}y_j}{d_i}-\tau_k\right]$$

where $\hat{\tau}_k$ is the k^{th} entry of the vector, $Q^{-1}Q^{-1}\mathbf{Y}$. Also, note that $\hat{L}_{2N}$ was never explicitly stored. The algorithm for the operation of $tr\left(Q^{-1}Q^{-1}\right)$ is achieved with the same algorithm as before; that is, *Algorithm 3.2*, and the algorithm for obtaining $Q^{-1}Q^{-1}\mathbf{Y}$ is achieved with the same algorithm with *Algorithm 3.3*.

Factoring ($Q^{-1}\Phi_1Q^{-1}$) and ($Q^{-1}\Phi_1Q^{-1}\Phi_1$)

Similar methodology can be used to derive the augmentation matrix $\breve{\mathrm{P}}$ to obtain fast factorisation for ($Q^{-1}\Phi_1Q^{-1}\Phi_1$) and ($Q^{-1}\Phi_1Q^{-1}$), with

$$\breve{\mathrm{P}}=\left[\begin{array}{c|ccc} Q+\eta\Phi_1 & Q & -\Phi_1 & 0 \\ \hline Q & 0 & -\Phi_1 & I \\ -\Phi_1 & -\Phi_1 & 0 & 0 \\ 0 & I & 0 & 0 \end{array}\right]$$

$$\Rightarrow \nabla^1_{\breve{\mathrm{P}}}=\left[\begin{array}{c|cc} -Q\left(Q+\eta\Phi_1\right)^{-1}Q & Q\left(Q+\eta\Phi_1\right)^{-1}\Phi_1-\Phi_1 & I \\ \hline \Phi_1\left(Q+\eta\Phi_1\right)^{-1}Q-\Phi_1 & -\Phi_1\left(Q+\eta\Phi_1\right)^{-1}\Phi_1 & 0 \\ I & 0 & 0 \end{array}\right] \tag{37}$$

$$\Rightarrow \nabla^2_{\breve{\mathrm{P}}}=\left[\begin{array}{cc} -\Phi_1Q^{-1}\left(Q-\eta\Phi_1\right)Q^{-1}\Phi_1 & -\eta\Phi_1Q^{-1}\Phi_1Q^{-1} \\ -\eta Q^{-1}\Phi_1Q^{-1}\Phi_1 & Q^{-1}\left(Q+\eta\Phi_1\right)Q^{-1} \end{array}\right]$$

where η is some constant factor to be determined in §3.6.6, and $\nabla^2_{\breve{\mathrm{P}}}$ is the Schur complement after $2N$ iterations, or to be precise, of the (2,2) block of $\breve{\mathrm{P}}$. If Q and Φ_1 are Toeplitz matrices, the displacement rank of $\breve{\mathrm{P}}$ is 6. Subsequently, its LDL-decomposition is given by

$$\nabla^2_{\breve{\mathrm{P}}}\cong\widetilde{L}_{2N}\overline{D}^{-1}_{2N}\widetilde{L}^{\mathrm{T}}_{2N}=\begin{bmatrix}\mathrm{Y}_{2N}\\ \varsigma_{2N}\end{bmatrix}\overline{D}^{-1}_{2N}\begin{bmatrix}\mathrm{Y}^{\mathrm{T}}_{2N} & \varsigma^{\mathrm{T}}_{2N}\end{bmatrix}$$

where $\mathrm{Y}_{2N} \in \mathrm{P}^{N\times N}$ is a full matrix, $\varsigma_{2N} \in \mathrm{P}^{N\times N}$, an upper triangular matrix and $D_{2N} \in \mathrm{P}^{N\times N}$, a diagonal matrix. These are obtained from the N+1 to 2N iterations of the Schur algorithm. The required traces are computed as follow,

$$tr\left(Q^{-1}\Phi_1 Q^{-1}\right) = \frac{1}{\eta}\left[\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{v_{ij}^2}{d_i} - tr\left(Q^{-1}\right)\right]$$

$$tr\left(Q^{-1}\Phi_1 Q^{-1}\Phi_1\right) = -\frac{1}{\eta}\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{u_{ij}v_{ij}}{d_i}$$

where u_{ij} and v_{ij} are i^{th} columns and j^{th} rows of Y_{2N} and ς_{2N}, respectively. Again, the algorithm for these trace operations is the same as given in *Algorithm 3.2*. Clearly,

$$\breve{\tau}_k = -\frac{1}{\eta}\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{u_{ik}v_{ij}y_j}{d_i}$$

computes the vector $\Phi_1 Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$, where $\breve{\tau}_k$ is the k^{th} entry of $(\Phi_1 Q^{-1}\Phi_1 Q^{-1}\mathbf{Y})$. Again, the algorithm for this operation is the same as given in *Algorithm 3.3*.

3.6.5 Predictions and Standard Deviations

Given the Gaussian process prior models, the interest is to obtain the predictive mean and variance for any finite set of values of the explanatory variable.

Factoring terms belonging to the posterior joint probability distribution

From the posterior joint probability distribution given in Chapter 2.4.3, it follows that the augmentation matrix $\hat{\mathrm{P}}$, used to compute the direct factorisations for the predictions and standard deviations, is

$$\hat{\mathrm{P}} = \left[\begin{array}{c|cc} -Q & \Lambda_{21} & \mathbf{Y} \\ \hline \Lambda_{21}^{\mathrm{T}} & 0 & 0 \\ \mathbf{Y} & 0 & 0 \end{array}\right] \Rightarrow \nabla^1_{\hat{\mathrm{P}}} = \begin{bmatrix} \Lambda_{21}^{\mathrm{T}}Q^{-1}\Lambda_{21} & \Lambda_{21}^{\mathrm{T}}Q^{-1}\mathbf{Y} \\ \mathbf{Y}^{\mathrm{T}}Q^{-1}\Lambda_{21} & \mathbf{Y}Q^{-1}\mathbf{Y} \end{bmatrix} \cong \begin{bmatrix} \mathrm{Y} \\ \varsigma \end{bmatrix} \overline{D}^{-1}\begin{bmatrix} \mathrm{Y}^{\mathrm{T}} & \varsigma^{\mathrm{T}} \end{bmatrix} \qquad (38)$$

where $\nabla^1_{\hat{\mathrm{P}}}$ is the Schur complement of the (1,1) block of $\hat{\mathrm{P}}$, after N iterations. If Q and Λ_{21} are Toeplitz matrices, the displacement rank of $\hat{\mathrm{P}}$ is 6, and not 4. This is

because of the presence of vector **Y** in the augmentation matrix. Unlike (35), (36) and (37), $\varsigma \in \mathrm{P}^{1\times N}$ is a row vector and $\mathrm{Y} \in \mathrm{P}^{h\times N}$ is a full matrix, such that h is the number of new outcomes to be predicted. It follows that the prediction for $\hat{\mathbf{y}}$ and the corresponding standard deviation $\hat{\mathbf{s}}$ are obtained as shown below

$$\hat{y}_k = \sum_{i=1}^{N} \frac{u_{ik} v_i}{d_i}, \; \hat{s}_k = \sqrt{a - \sum_{i=1}^{N} \frac{u_{ik}^2}{d_i}}$$

where $\hat{y}_k$ and $\hat{s}_k$ are k^{th} entries of the respective vectors, $\hat{\mathbf{y}}$ and $\hat{\mathbf{s}}$, and a is the adapted hyperparameter of the covariance function (26). The algorithm to compute the prediction and standard deviation is illustrated below.

Algorithm 3.4 (Obtaining prediction and standard deviation using Schur algorithms): The operations to obtain the vectors $\hat{\mathbf{y}}$ and $\hat{\mathbf{s}}$, involving the generalised Schur algorithm, are explained here. From the generalised Schur algorithm described in *Algorithm 3.1*, the matrices Y and ς are obtained column by column. Starting with vectors $\hat{\mathbf{y}} = 0$ and $\mathbf{t} = 0$, where $\hat{\mathbf{y}}, \mathbf{t} \in \mathrm{P}^{\overline{N}\times 1}$, the following steps are performed for i = 0,1,…, N-1:

1. Obtain l_i and d_i from the generalised Schur algorithm (*Algorithm 3.1*). Then, define $\overline{u}_i$ and $\overline{v}_i$ as shown:

$$l_i = \begin{bmatrix} \overline{u}_i \\ \overline{v}_i \end{bmatrix}$$

2. Compute the following:

$$\hat{\mathbf{y}} = \hat{\mathbf{y}} + \frac{\overline{u}_i \bullet \overline{v}_i}{d_i} \text{ and } \mathbf{t} = \mathbf{t} + \frac{\overline{u}_i \bullet \overline{u}_i}{d_i}$$

where • denotes the Hadamard product or entrywise product.

The final values of $\hat{\mathbf{y}}$ and $\hat{\mathbf{s}} = \sqrt{a - \mathbf{t}}$ are the corresponding values for the prediction and standard deviation of the posterior joint probability distribution.

The derivative observation of the Gaussian regression can also be computed using the same augmentation matrix (38). The prior covariance function is assumed to have

been chosen to be (26). Let the covariance matrix for the expectation between the derivative observation and the measurements be $\dot{\Lambda}_{21}$. By substituting Λ_{21} in (38) with $\dot{\Lambda}_{21}$, the derivative predictions (and its confidence intervals) can be acquired by fast factorisation using the generalised Schur algorithm. It follows that the derivative predictions for $\hat{\mathbf{y}}'$ and standard deviation, $\hat{\mathbf{s}}'$, are

$$\hat{y}_k' = \sum_{i=1}^{N} \frac{u_{ik} v_i}{d_i}, \quad \hat{s}_k' = \sqrt{a\tilde{d} - \sum_{i=1}^{N} \frac{u_{ik}^2}{d_i}}$$

where $\hat{y}_k'$ and $\hat{s}_k'$ are the k^{th} entries of the respective vectors, $\hat{\mathbf{y}}'$ and $\hat{\mathbf{s}}'$, and $\tilde{d}$ is the lengthscale hyperparameter, *d*, value as defined in (26).

This augmentation matrix works on the condition that the values of the explanatory variable of the posterior are sampled at fixed interval; that is, the resulting covariance matrix is also Toeplitz-like.

From §3.6.3 to §3.6.5, much of the concepts are based on *Q* being a Toeplitz matrix. It is apparent that the augmentation matrices can be extended for *Q* to be a block-Toeplitz-block matrix; for example, in the case where the time-series data contain missing gaps. Consequently, every increment in the displacement rank of *Q* results in an increment in the displacement rank of P.

3.6.6 Convergence Factors in Augmentation Matrices

This section describes how the values of μ and η in (36) and (37) respectively are chosen. A good choice for these values is needed to ensure that computational errors are reasonably small. On the other hand, a poor choice of the values is likely to have significant impact on the accuracy of the generalised Schur algorithm.

Obtaining μ

There is some relationship between the constant, μ, and the condition number of the Schur complement of (36). μ is used to ensure that the condition number does not

become too large at every iterative step of the generalised Schur algorithm. Given the first N iterations of the algorithm, the Schur complement of the (1,1) block is $Q(Q+\mu I)^{-1}Q$. However,

$$\left\|Q(Q+\mu I)^{-1}Q\right\| \le \|Q\| \qquad \forall \mu \in \mathrm{P}^{+}$$

where ||.|| refers to the condition number. It is important to keep $\left\|Q(Q+\mu I)^{-1}Q\right\|$ as small as possible, but the value of μ cannot be too small such that its contribution to $(Q+\mu I)^{-1}$ becomes negligible. Focusing on $\left\|(Q+\mu I)^{-1}Q\right\|$,

$$\begin{aligned}\left\|(Q+\mu I)^{-1}Q\right\| &= \left\|(Q+\mu I)Q^{-1}\right\| \\ &= \left\|I+\mu Q^{-1}\right\| \\ &= \frac{1+\mu\times\max\left[\ell\left(Q^{-1}\right)\right]}{1+\mu\times\min\left[\ell\left(Q^{-1}\right)\right]} \\ &= \frac{1+\mu/\min\left[\ell(Q)\right]}{1+\mu/\max\left[\ell(Q)\right]} \approx 1+\frac{\mu}{\min\left[\ell(Q)\right]}\end{aligned}$$

where $\ell(Q)$ denotes the eigenvalues of Q. It follows that a reasonable choice for μ is $\mu \approx \min\left[\ell(Q)\right]$. Since $Q = \Lambda + bI$, where Λ is a covariance matrix. The smallest eigenvalue of Λ can be extremely small and is insignificant compared to b, except when the lengthscales are large compared to the range of explanatory variable presented. Therefore, it is apparent that the best value of μ should be b.

In the case where Λ is a sparse or a diagonal matrix, b will no longer be a good choice for μ. However, this is trivial, because the smallest eigenvalue of Λ (assuming Λ is now a diagonal matrix) is simply the smallest element of the diagonal of Λ. More detail cases are discussed in Appendix C.

Obtaining η

In the case of η, it is no longer a condition number problem. Instead, it is an issue of ensuring the (1,1) block of the Schur complement of the augmentation matrix (37) to remain positive-definite. It follows that the "optimal" value for η is 1. The following shows the theoretical verification.

Proof 3.2 (Choice of η = 1): Given the covariance function with explanatory variable that is one-dimensional is of the form in (26), let $C(z_i, z_j) = aC_n(z_i, z_j) + b\delta_{ij}$ and a = 1, such that C and C_n are normalised. The covariance function C_n is

$$C_n(z_i, z_j) = \exp\left\{-\frac{d}{2}(z_i - z_j)^2\right\}$$

Let $d = e^{\gamma}$, it follows that

$$\begin{aligned}\frac{\partial^2 C_n}{\partial z_i \partial z_j} &= \left[d - d^2(z_i - z_j)^2\right]\exp\left\{-\frac{d}{2}(z_i - z_j)^2\right\} \\ &= dC_n + 2d\frac{\partial C_n}{\partial \gamma}\end{aligned}$$

Rearranging,

$$\frac{\partial C_n}{\partial \gamma} = \frac{1}{2d}\frac{\partial^2 C_n}{\partial z_i \partial z_j} - \frac{1}{2}C_n$$

Using this property, let Q be the matrix for the covariance function C and $\overline{P}$ be the matrix for the covariance function C_n, the following equation can be reformulated to

$$\begin{aligned}\left[Q + \eta\frac{\partial \overline{P}}{\partial \gamma}\right] &\equiv \overline{P} + bI + \eta\frac{\partial \overline{P}}{\partial \gamma} \\ &= \frac{\eta}{2d}\frac{\partial^2 \overline{P}}{\partial z_i \partial z_j} + \left(1 - \frac{\eta}{2}\right)\overline{P} + bI\end{aligned} \tag{39}$$

To ensure that the left-hand side of (39) is positive-definite, the right-hand side of the equation also has to be positive-definite. Thus, the constant η is bounded between 0 and 2 to guarantee positive-definiteness for the matrix on the left-hand side of the equation.

Examples to support the above statement are illustrated in Figure 8 to Figure 11. The experiments clearly substantiate that the suggested choice for η is 1. It is evident from Figure 8 to Figure 10 that the error estimates are much lower for η between 0 and 2, than values outside this range. The poor accuracy is a result of the Schur complements not being positive-definite. Figure 11 confirms that the errors are accumulated during the computations of the generalised Schur algorithm.

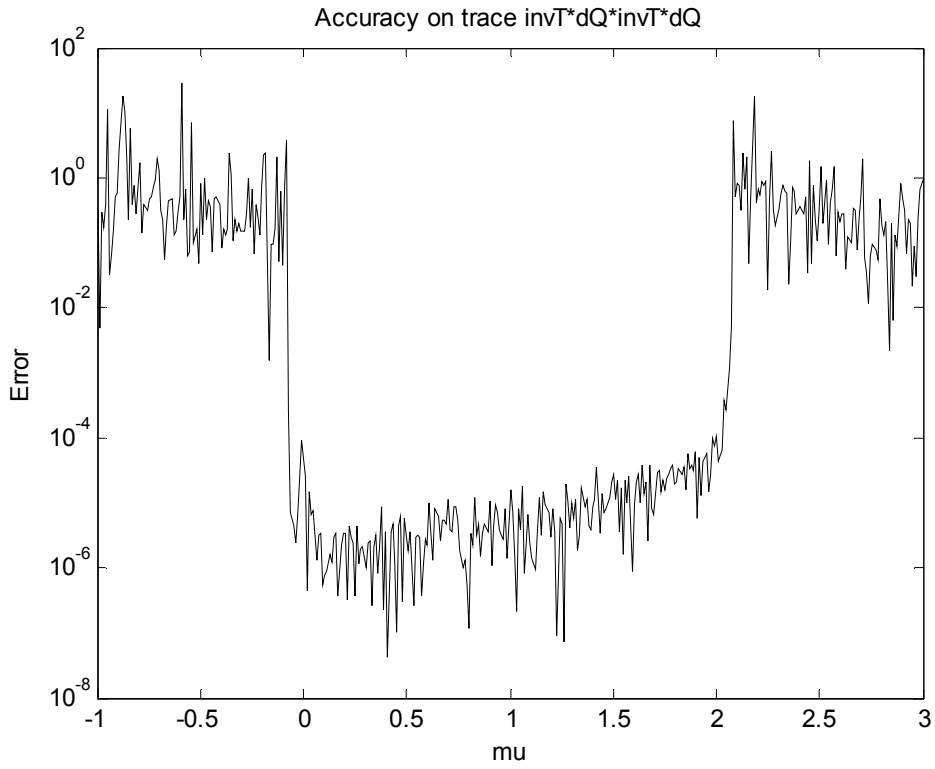


Figure 8 Error from the trace operation of $(Q^{-1}\Phi_1 Q^{-1}\Phi_1)$.

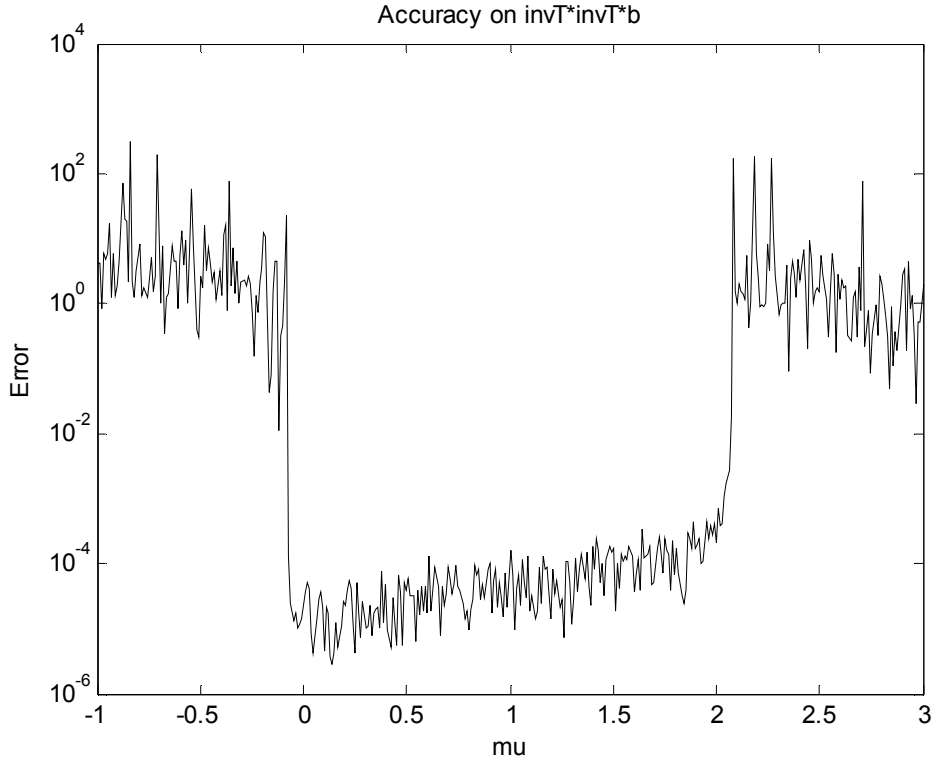


Figure 9 Error from the computation of vector $(\Phi_1 Q^{-1}\Phi_1 Q^{-1}\mathbf{Y})$.

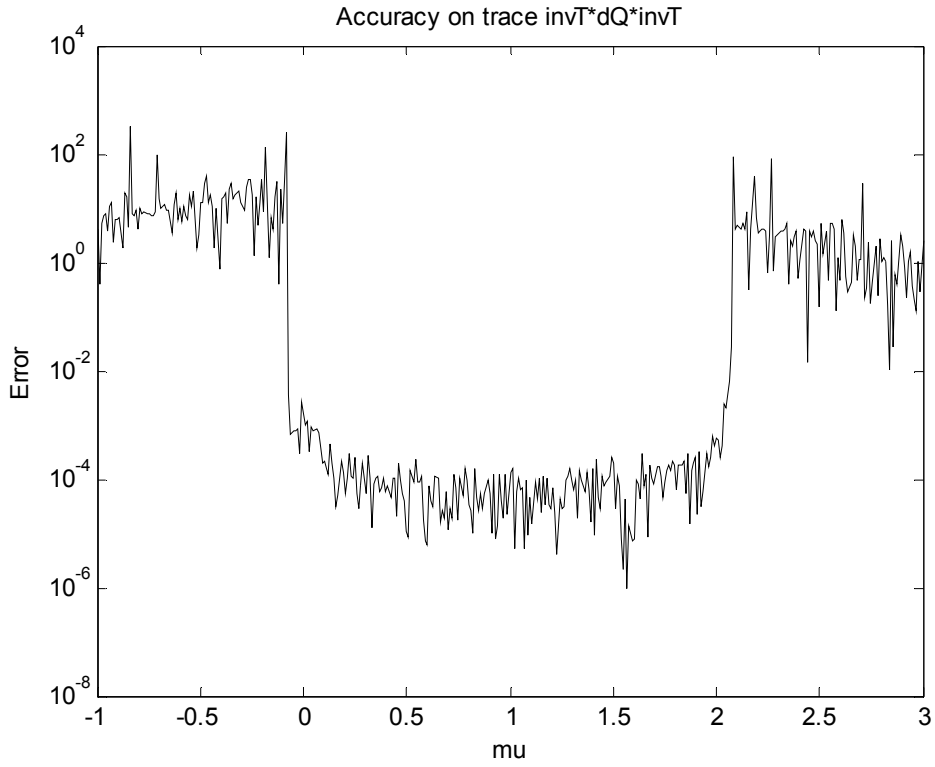


Figure 10 Error from the trace operation of $(Q^{-1}\Phi_1 Q^{-1})$.

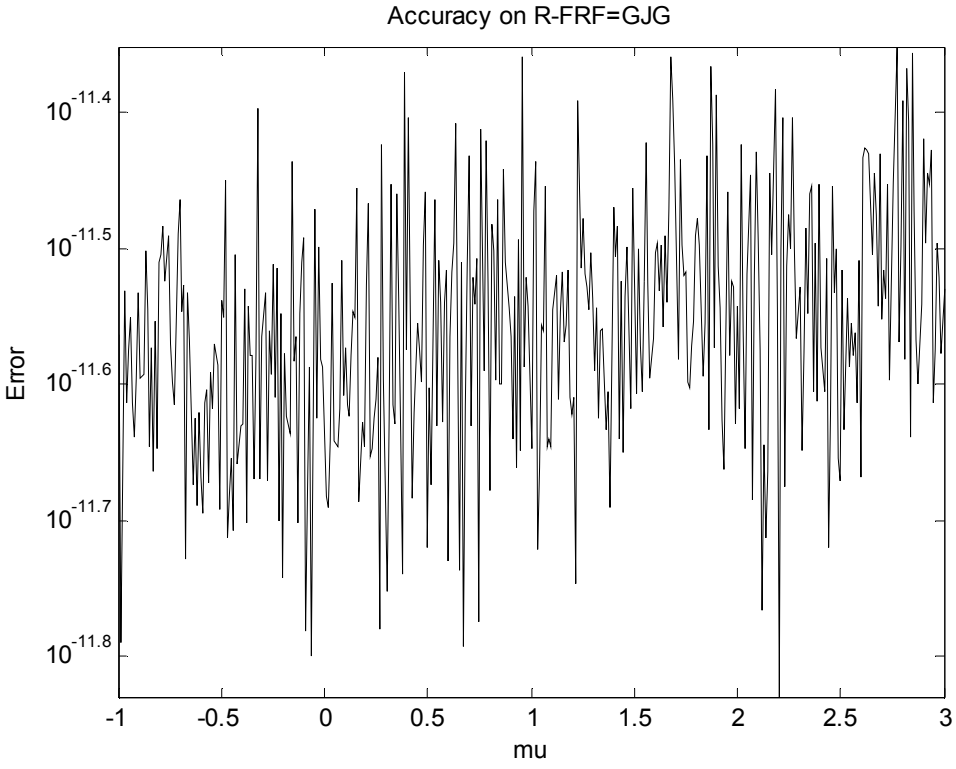


Figure 11 Error estimation of the Schur decomposition of $P - \Phi P \Phi^T \equiv \Gamma \vartheta \Gamma^T$.

From the theoretical explanations and numerous experiments, it follows that a good choice of μ can be chosen to be equivalent to the noise variance, b, to ensure relatively optimal and accurate results. Similarly, for η, it has been shown that any value between 0 and 2 is suitable, as long as the matrix $Q+\eta(\partial Q/\partial\gamma)$ is confined to be positive-definite. An appropriate choice for η is naturally 1, since it conveniently lies in the middle of the boundary, and also for the reason of simplifying calculations.

3.7 Numerical Experiments

The effectiveness of the generalised Schur algorithms is analysed in the following five sub-sections. To exploit its capabilities and provide a standard benchmark, the generalised Schur algorithm is programmed in both MATLAB and C languages, i.e. scripts are written in MATLAB and C codes[7]. Standard MATLAB operations, without the use of any fast algorithm, are also made available to provide additional comparison. The experiments are conducted on time-series datasets.

Firstly, individual Schur functions, written in both coding languages, are compared with explicit standard MATLAB operations, in terms of accuracy and performance. The effects of hyperbolic rotation[8] on the Schur algorithms are also evaluated on both coding languages. Secondly, a similar test is conducted with the same criteria, except on time-series data with one missing gap. Next, the focus is on the Gaussian process optimisation routine, to compare the timing performance and the accuracy of the convergence to the correct local minima between the C codes, MATLAB codes and standard MATLAB functions using time-series datasets. Subsequently, the same test is conducted on time-series datasets with one missing gap. The final optimisation test compares the performance of the generalised Schur algorithm and modified Durbin-Levinson's algorithm, developed by Leithead and Zhang's (2005c) as discussed in §3.5.1. This final test considers only strictly time-series (pure Toeplitz) data.

[7] The C codes are essential because MATLAB are not efficient in performing loops operations, e.g. for loops and while loops. C codes are known to be both memory and computationally efficient, especially for fast algorithms.
[8] Generalised Schur algorithm is performed with and without hyperbolic rotation.

To avoid numerical inaccuracies in the algorithm, the value of the hyperparameter n for the correlation function between y_i and y_j

$$\mathrm{E}\left[y_i, y_j\right] = a\left\{\exp\left[-\frac{d}{2}\left(\mathrm{z}_i - \mathrm{z}_j\right)^2\right] + n\delta_{ij}\right\}$$

is constrained for $n > 1 \times 10^{-5}$ throughout the experiments. Due to machines having finite precision, the constraint is imposed primarily to ensure that the errors from the eigenvalue decomposition of Procedure 3.2, whilst obtaining the generating matrix, are minimal. This is to further avoid results of the generalised Schur algorithm computation from being affected by these errors.

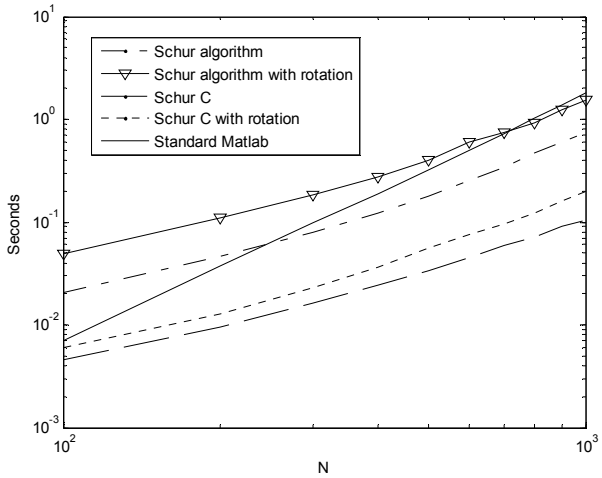
The experiments are carried out on an Intel® Pentium® IV 2.8GHz machine with 512MB memory running Linux Operating System. The MEX-C codes are compiled using GCC 3.5.1, optimised to the architecture of the machine. The installed MATLAB version is R14.

3.7.1 Test One (Function Test, Data without Gap)

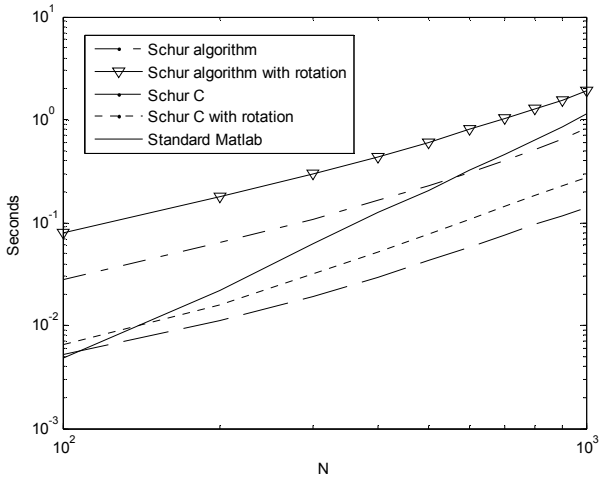
A run of 100 Gaussian process-generated samples per data size, N, $\forall N = \{100,200,...,1000\}$ is carried out to evaluate the performance of the modified generalised Schur algorithm using augmentation matrix (35) with the standard MATLAB functions. The functions involved in the calculations are $tr(Q^{-1})$, $\mathrm{tr}(Q^{-1}\Phi_1)$, $\log|Q|$, $Q^{-1}\mathbf{Y}$, $\mathbf{Y}^{\mathrm{T}}Q^{-1}\mathbf{Y}$, $\mathbf{Y}^{\mathrm{T}}Q^{-1}Q^{-1}\mathbf{Y}$, $\Phi_1 Q^{-1}\mathbf{Y}$ and $\mathbf{Y}^{\mathrm{T}}Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$. Note that, instead of applying a direct determinant operation, MATLAB's computation of the log-determinant of Q is calculated using the Cholesky decomposition as shown below.

$$\log|Q| = 2\sum \log\{diag(\Lambda)\}$$

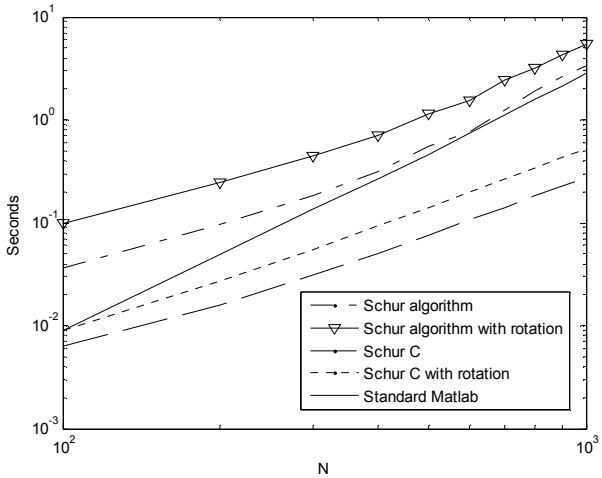
such that $Q = \Lambda^{\mathrm{T}}\Lambda$. Hyperparameter values and noise data are randomly chosen for every sample to ensure reasonably unbiased test results. The range of the values of the explanatory variable is between 0 and 100. To avoid numerical breakdown of the algorithm, randomly-generated lengthscale hyperparameters are also constrained, i.e. $d > 1 \times 10^{-5}$.



a) Timing for tr(Q^{-1}), $tr(Q^{-1}\Phi_1)$, log|Q|, $Q^{-1}\mathbf{Y}$, $\mathbf{Y}^{\mathrm{T}}Q^{-1}\mathbf{Y}$, $\mathbf{Y}^{\mathrm{T}}Q^{-1}Q^{-1}\mathbf{Y}$, $\Phi_1Q^{-1}\mathbf{Y}$ and $\mathbf{Y}^{\mathrm{T}}Q^{-1}\Phi_1Q^{-1}\mathbf{Y}$.



b) Timing for $tr(Q^{-1}Q^{-1})$ and $Q^{-1}Q^{-1}\mathbf{Y}$.



c) Timing for $tr(Q^{-1}\Phi_1Q^{-1}\Phi_1)$, $\Phi_1Q^{-1}\Phi_1Q^{-1}\mathbf{Y}$ and $tr(Q^{-1}\Phi_1Q^{-1})$.

Figure 12 Computation time on function test for data without gap.

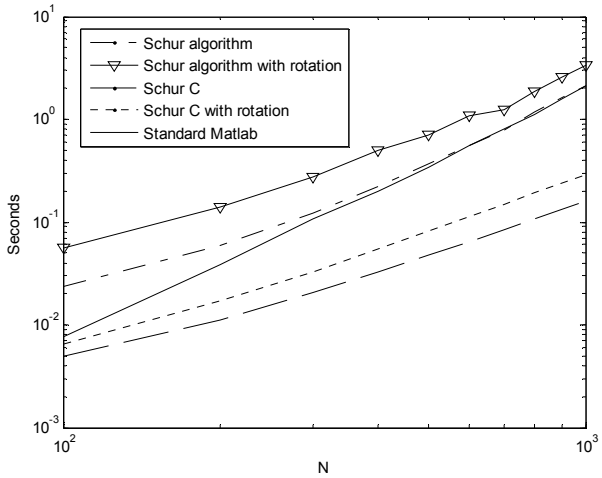
The $O(N^3)$ flops from Figure 12 are clearly visible when using standard MATLAB operations to calculate the required values for the Gaussian process. Generally, the Schur algorithm that has been programmed in MATLAB is slow for small dataset, but its advantage supersedes from N = 500 onwards. Both C codes (with and without hyperbolic rotation) produce remarkable speedup over the standard MATLAB operation from the start. Schur algorithm with hyperbolic rotation programmed in MATLAB language shows slower improvement, but apparently remains $O(N^2)$.

The mean relative accuracies of the generalised Schur algorithm, tabulated in TABLE C – I in Appendix C, are compared to that using standard MATLAB functions. Except for the one-off case of $tr(Q^{-1}\Phi_1 Q^{-1}\Phi_1)$ in which the hyperbolic rotation in MEX-C code have resulted in lower accuracy; otherwise the accuracy improvements from the introduction of hyperbolic rotation, compared to non-hyperbolic rotation algorithm in handling time-series data, are insignificant. In conclusion, introducing hyperbolic rotation in time-series data does not yield more accurate results than those without hyperbolic rotation, as can be seen from TABLE C – I.

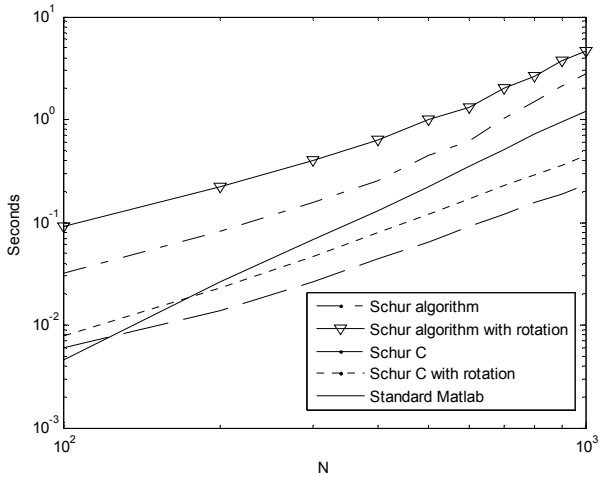
3.7.2 Test Two (Function Test, Data with one Gap)

Test One is repeated in Test Two, keeping all conditions unchanged, except performed on data with one missing gap. The performance of the various Schur algorithm codes is illustrated in Figure 13. Both C codes have shown to perform impressively fast, despite the doubling of the displacement rank of the covariance matrix in Test One. Whilst the generalised Schur algorithm written in MATLAB shows little improvement, its poor performance can be attributed to its weakness in handling multiple loop operations that exist in the Schur algorithm.

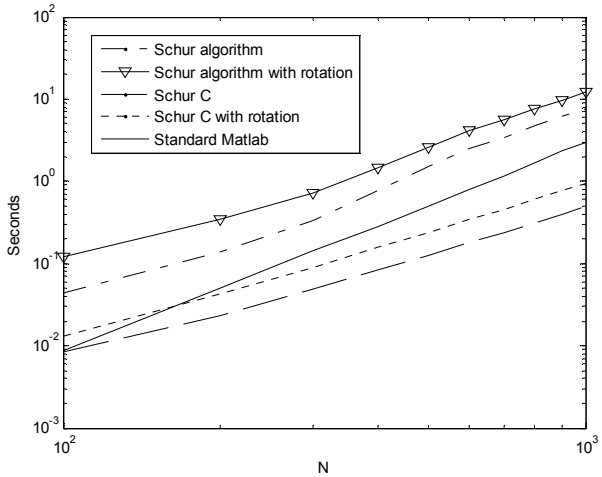
The relative accuracies of the generalised Schur algorithm programmed in both MATLAB and C codes (both including with and without hyperbolic rotations) are compared to those using standard MATLAB functions. The results are tabulated in TABLE C – II in the Appendix.



a) Timing for tr(Q^{-1}), $tr(Q^{-1}\Phi_1)$, log|Q|, $Q^{-1}\mathbf{Y}$, $\mathbf{Y}^{\mathrm{T}}Q^{-1}\mathbf{Y}$, $\mathbf{Y}^{\mathrm{T}}Q^{-1}Q^{-1}\mathbf{Y}$, $\Phi_1Q^{-1}\mathbf{Y}$ and $\mathbf{Y}^{\mathrm{T}}Q^{-1}\Phi_1Q^{-1}\mathbf{Y}$.



b) Timing for $tr(Q^{-1}Q^{-1})$ and $Q^{-1}Q^{-1}\mathbf{Y}$.



c) Timing for $tr(Q^{-1}\Phi_1Q^{-1}\Phi_1)$, $\Phi_1Q^{-1}\Phi_1Q^{-1}\mathbf{Y}$ and $tr(Q^{-1}\Phi_1Q^{-1})$.

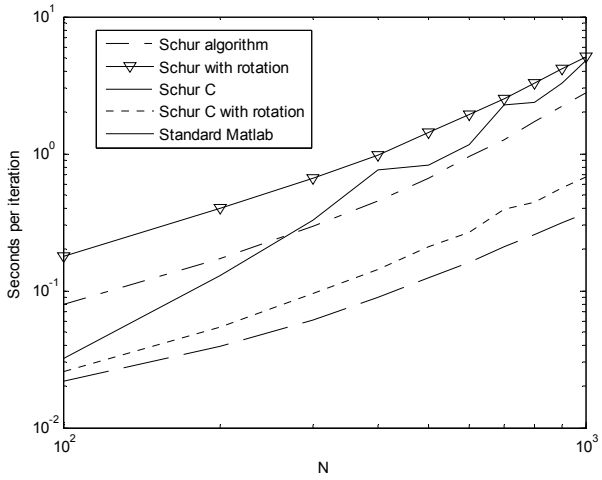
Figure 13 Computation time on function test for data with gap.

The benefit of implementing hyperbolic rotation in the generalised Schur algorithm is more apparent for the case of time-series data with a single gap. The relative errors of those implemented with hyperbolic rotations are lower than those without, with the only exception of $tr(Q^{-1}\Phi_1 Q^{-1}\Phi_1)$, which was observed to have much better accuracy from the use of hyperbolic rotation as compared to other functions.

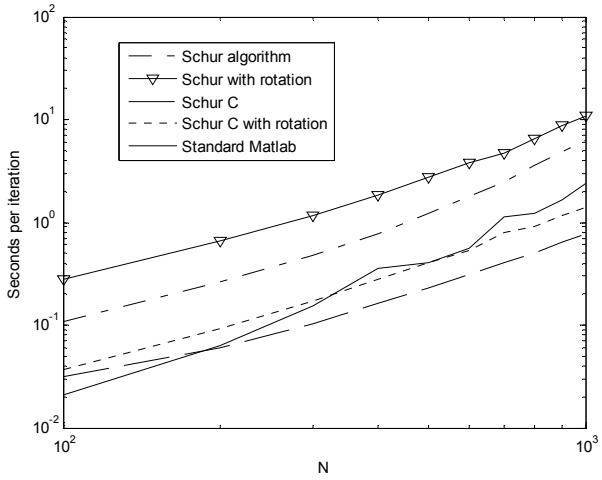
3.7.3 Test Three (GP Test, Data without Gap)

The modified Schur algorithm is then applied in practice to the training procedure of the Gaussian process prior models. Tests are conducted for data sizes $N = \{100,200,...,1000\}$ with each data size having 20 samples. Figure 14 illustrates the timing per iteration for each of the five approaches. The average number of iterations that the optimisation takes to converge is collated in TABLE C – III of Appendix C.

All the ten optimisations, for every sample test, converge to the same local minima; hence, the hyperparameter values are similar. The unevenness of the graph in Figure 14 is perfectly normal, since the average number of iterations for every data size, N, required for convergence differs from one another. Despite the lack of smoothness, the benefits of the Schur algorithm are apparent. It is noted that the standard MATLAB operation takes the longest time to converge than any of the other four methods.



a) Training (per iteration) of GP using derivative information only.



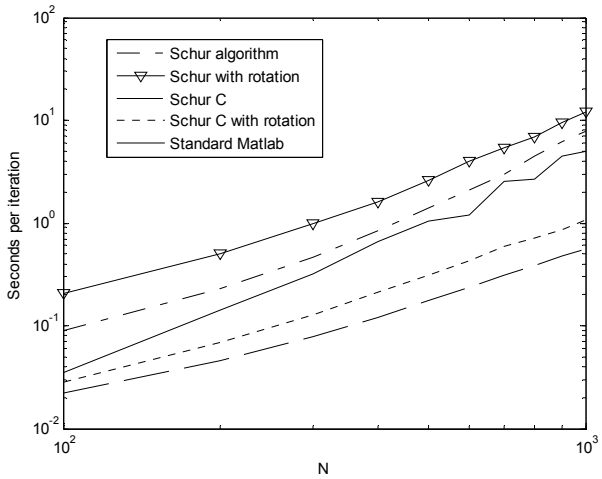
b) Training (per iteration) of GP using both derivative and Hessian information.

Figure 14 Timing (per iteration) of GP training on data without gap.

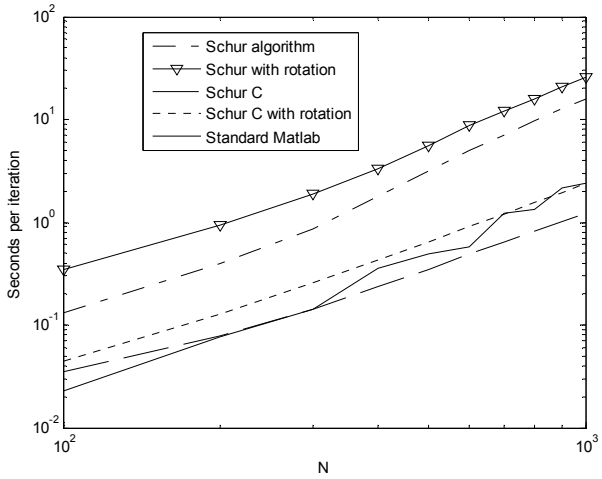
3.7.4 Test Four (GP Test, Data with one Gap)

Subsequently, Test Three is repeated on time-series data having one missing gap. The performance of the modified Schur algorithm programmed in MATLAB language demonstrates its poor efficiency of handling multiple "loop"-operations, e.g. "for-loops", as shown in Figure 15. Regardless of hyperbolic rotation implementation, the Schur algorithm codes written in MATLAB fails to prove its worth. Alternatively,

MEX-C codes are observed to have performed up to expectations. This is due to C codes being more efficient in handling algorithms with multiple "loop"-operations, despite the increment of the displacement rank. The number of iterations required for convergence is listed in TABLE C – IV of Appendix C.



a) Training (per iteration) of GP using derivative information only.



b) Training (per iteration) of GP using both derivative and Hessian information.

Figure 15 Timing (per iteration) of GP training on data with a missing gap.

3.7.5 Test Five (Durbin-Levinson versus Schur Algorithm)

The final test compares the performance of the two fast algorithms, viz. the generalised Schur algorithm and the modified Durbin-Levinson's algorithm (Leithead et al., 2005c). The latter algorithm is only capable of handling covariance matrices

that are strictly Toeplitz, i.e. time-series data with fixed sampling interval. A sine function, dependent on explanatory variable, $z \in [0,10]$, is chosen to be the test data. The outcome is $y_i = \sin(z_i) + \zeta_i$, where ζ is additive Gaussian white noise of variance 0.1. Gaussian regression using standard gradient-based optimisation routine is performed on ten different data sizes, $N = \{10{,}000, 20{,}000, ..., 100{,}000\}$. Note that, due to extremely large dataset size, optimisation using standard MATLAB functions is not possible. These two fast algorithms are programmed in MEX-C language. The rates of convergence using these fast algorithms are tabulated in TABLE IV**.**

TABLE IV Performance (timing) between modified Durbin-Levinson's algorithm and modified generalised Schur algorithm

Data size	Timing per iteration (hour)		
N	S	DL	Speedup (S/DL)
10,000	0.0182	0.0021	8.6
20,000	0.0807	0.0097	8.4
30,000	0.1829	0.0243	7.5
40,000	0.3208	0.0526	6.1
50,000	0.5021	0.0949	5.3
60,000	0.7427	0.1519	4.9
70,000	0.9720	0.2164	4.5
80,000	1.2793	0.2898	4.4
90,000	1.7092	0.4014	4.3
100,000	1.8577	0.4597	4.0

S refers to generalised Schur algorithm with no hyperbolic rotation. DL refers to Durbin-Levinson's algorithm. Both algorithms are compiled using Mex-C codes. The table illustrates the timings for time-series Gaussian regression using two different fast algorithms. The timings (per iteration) shown are calculated based on successful convergence of the Gaussian process optimisation routine.

Both algorithms results in the optimisation converging to similar local minima during the training process. It was noticed that the modified Durbin-Levinson's algorithm produces a 25-fold improvement over the Schur algorithm, when the data size increases from N = 4,000 to N = 10,000. The main reason is that no generator matrix is required for the former; only vector-level storage algorithm and the reflection coefficient is used, thus having better computational and storage efficiency. When the experiment continues for $N \geq$ 10,000, the speedup advantage of Durbin-Levinson's algorithm gradually drops to approximately 4-fold speedup, as shown in TABLE IV.

The reason why the Durbin-Levinson's algorithm outperforms the generalised Schur algorithm by a large margin at the beginning is mainly because of its vector storage feature; only the reflection coefficient and vector are stored. Although the latter also uses the idea of vector storage, it is more demanding in the storage requirements, i.e. the generator matrix Γ and intermediate vectors of the Schur algorithms. There are two possible reasons with regards to the decline in the ratio of the speedup factors as data size increases from 4,000 to 10,000. Firstly, the modified Durbin-Levinson's algorithm has reached the system's available memory resources, resulting in more read-write operations on the hard disk. Access to data stored on the hard disk is slower compared to data stored on the system's memory, hence impedes the speed performances of the fast algorithms. Secondly, the function calls in the modified Durbin-Levinson's algorithm increase to a large extent such that it becomes a bottleneck within the algorithm. In the case of the generalised Schur algorithm, the number of function calls does not increase as much.

Despite the impressive speed benefit of the modified Durbin-Levinson's algorithm, it also has its drawbacks. The algorithm can only handle cases with matrices that are strictly Toeplitz. For instance, it cannot be applied to time-series data with missing gaps. This limitation hinders the identification of real dynamic systems, from which data obtained at fixed sampling intervals could potentially have missing information. The generalised Schur algorithm, on the other hand, has the capability to handle such situation, i.e. Toeplitz-like matrices. Secondly, the Durbin-Levinson's algorithm limits training procedure to using only user-supplied Gradient information, whereas the Schur algorithm allows both user-supplied Hessian and Gradient optimisation. Finally, the computation of the standard deviations of data points (except training data itself) is impossible with the modified Durbin-Levinson's algorithm. Explicit computation of the posterior remains $O(N^3)$ with $O(N^2)$ memory requirement. Nonetheless, the generalised Schur algorithm allows computation of the predictive means and variances with $O(\mathrm{k}N^2)$ operations.

3.7.6 Experimentation Summary

The relative errors of the outputs from the Schur algorithm in Test One and Test Two, as compared to that from using standard MATLAB function, are approximately between 10^{-8} and 10^{-13}. By keeping (32) to a high level of accuracy, it follows that the errors resulting from the Schur algorithm are minimal. In cases with very low displacement rank, e.g. displacement rank of 2, there is little differences between the errors obtained from the algorithms with hyperbolic rotation and those without, as long as subsequent Schur complement of each iteration is positive-definite. Clearly, it is much faster not to include the hyperbolic rotation. As displacement rank increases, the benefit of hyperbolic rotation becomes apparent.

All the optimisations in Test Three and Test Four have converged to the correct optimum points, thus resulting in similar hyperparameter values for every sample data. Similarly, the generalised Schur algorithm works sufficiently well, even without the implementation of the hyperbolic rotation. Since only data with one gap has been tested, the use of hyperbolic rotation is justifiable only if higher displacement rank of the augmentation matrix is encountered.

The modified Durbin-Levinson's algorithm has proven in the final test to be superior, in terms of speed performance, than the generalised Schur algorithm. However, the latter is capable of handling time-series data with missing gaps, using Hessian information in the optimisation routine and computing the standard deviation (of any data point).

Theoretically, both algorithms are capable of handling large-scale data size of up to one million data points and beyond, only to be constrained by the memory capacity of the machine. As memory chips are relatively inexpensive in today's market; therefore the algorithms are more restricted by processor time. Generally, the modified Durbin-Levinson's algorithm is highly recommended for strictly time-series data, whereas the Schur algorithm is more suitable for time-series data with missing gaps. An application using the generalised Schur algorithm is presented in §3.8.

3.8 Application of Schur Algorithm on Contest Data

The effectiveness of the modified generalised Schur algorithm is demonstrated by application of Gaussian regression on a dataset of 5,000 points sampled at 1Hz (CATS Benchmark, 2004). The data contains four gaps, specifically at the following locations, (981s – 1000s), (1981s – 2000s), (2981s – 3000s), (3981s – 4000s) and (4981s – 5000s). Before applying Gaussian regression, the number of separable components in the data is investigated to determine whether a single Gaussian process model with compound covariance function or a multiple Gaussian process model (Leithead et al., 2005b) is required. However, it is not of interest to identify the individual components in the data, which is discussed in §4.4. The intent here is to integrate the generalised Schur algorithm in Gaussian regression to perform analysis on the data. The time-series data is shown in Figure 16 with the spectral density function shown in Figure 17. The data has a component with lengthscale longer than the gap at frequencies less than 0.045Hz. Any other component is treated as noise. Gaussian regression, using the correlation function (26), is employed to identify the low frequency component.

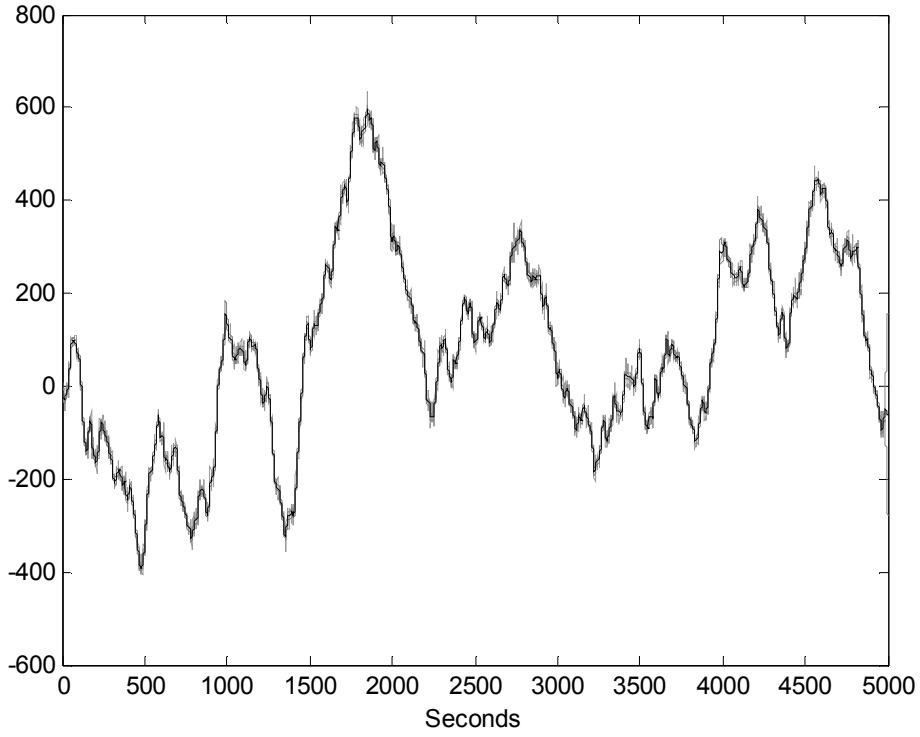


Figure 16 Time-series data with several missing gaps.

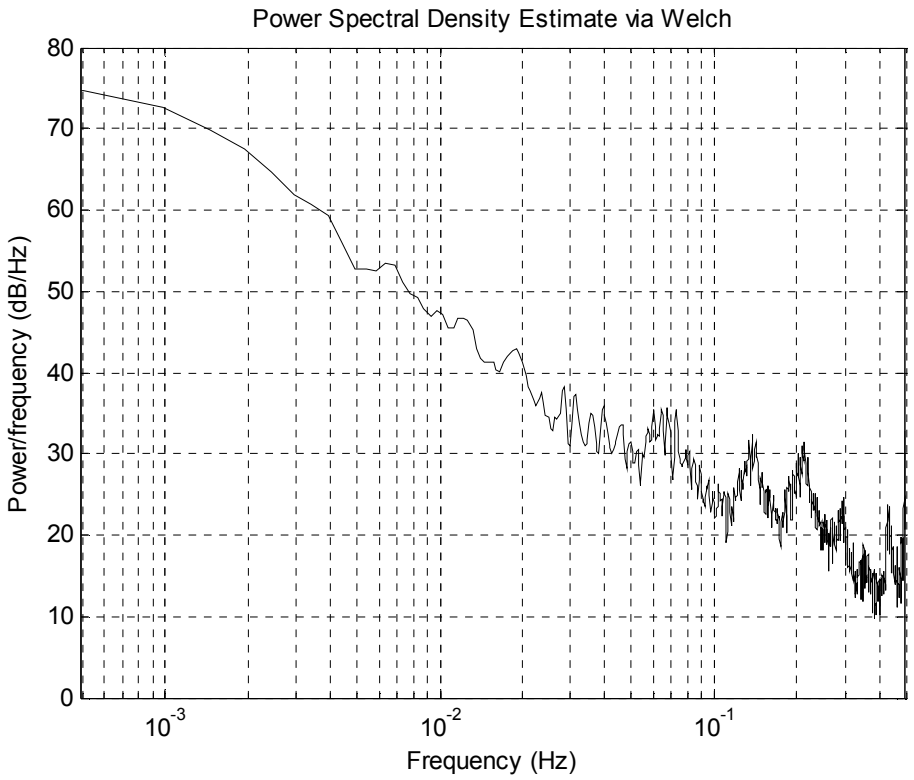


Figure 17 Power spectra of the data showing the presence of multiple components.

The values of the hyperparameters for a, d and b are 2.422 x 10^4, 0.00379 and 154.52, respectively. The optimisation procedure took approximately 27 minutes and 7 iterations to converge. A typical section from 950s and 1250s, including one of the missing gaps, is shown in Figure 18. The prediction with two times standard deviations (black lines) demonstrates the filtering of noise from the original data. The fit of the data is predicted over the gap as shown in the figure, with the confidence intervals being much wider over the missing data region.

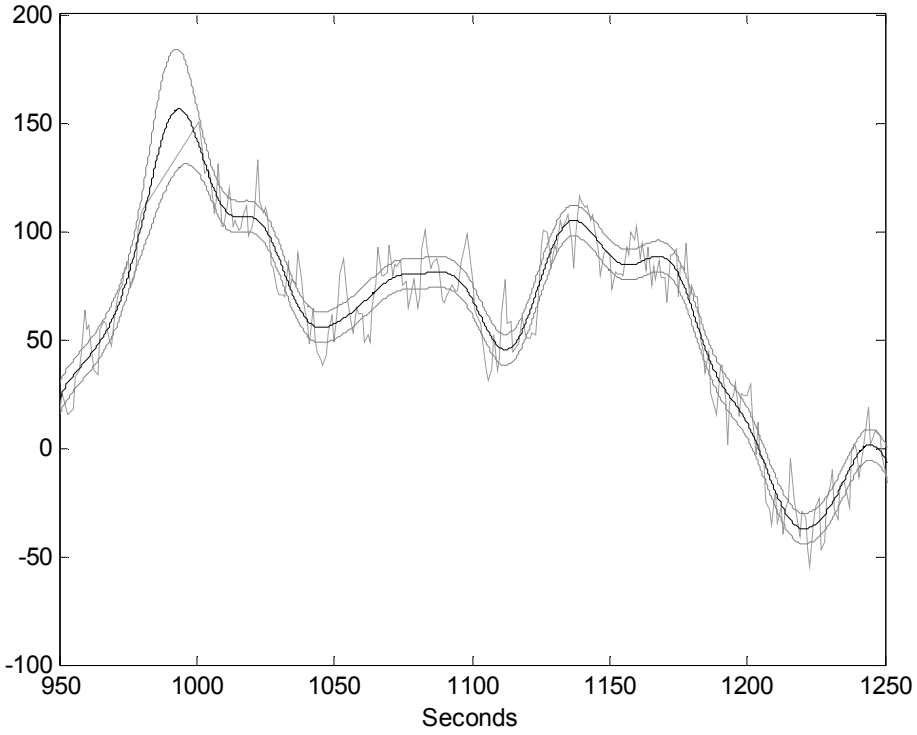


Figure 18 Noisy data (grey), prediction (black) and confidence intervals.

3.9 Conclusions

The implementation of the generalised Schur algorithm developed in this chapter is vital to the incorporation of Gaussian process model with fast time-series applications. It is worthwhile noting that by implementing the codes in C language ensures faster computations than coding in MATLAB. Hyperbolic rotation is not necessary if the augmentation matrix has low displacement rank. The generalised Schur algorithm is

particularly useful in, but not limited to, handling large-scale time-series data, with missing gaps.

Besides modified as a fast algorithm, the generalised Schur algorithm can also be used to estimate the increments from a large dataset; that is, to identify a nonlinear dynamic system in incremental form. The identification process introduces correlation of the noise present at different values of the time parameter, but this is lessened by the measured time parameter following the selection of only a subset. This methodology is described in Chapter 6, where a novel idea to integrate state-space and time-series domains together within a single stochastic model is developed.

Chapter 4

Multiple Gaussian Processes

4.1 Introduction

In this chapter, the nonlinear relationship underlying the measured data set is interpreted to consist of two or more additive components. The components may have the same explanatory variable or different explanatory variables. A Gaussian regression methodology to extract the components, with the class of functions for each represented by a distinct Gaussian process, is presented. Gaussian regression based on these compound multiple Gaussian process models is distinct from Gaussian regression based on a single Gaussian process with a compound covariance function (Mardia and Marshall, 1984) and has not previously been investigated. Their utility is illustrated with respect to a data set with the explanatory variable having variable density and with respect to a 5,000-point time-series data set with missing gaps (CATS Benchmark, 2004).

4.2 Gaussian Regression with Two Stochastic Processes

Consider the situation when the nonlinear relationship underlying measured data has the form, $h(\mathbf{z},\mathbf{w}) = f(\mathbf{z}) + g(\mathbf{w})$. The two components have different characteristics and can have the same explanatory variable, i.e. $\mathbf{z} = \mathbf{w}$, or different explanatory variables, i.e. $\mathbf{z} \neq \mathbf{w}$. The task is to extract either or both of f(**z**) and g(**w**).

Let the set of measurements be $\{y_i, (\mathbf{z}_i, \mathbf{w}_i)\}_{i=1}^{N}$, where $y_i = h(\mathbf{z}_i, \mathbf{w}_i) + n_i = f(\mathbf{z}_i) + g(\mathbf{w}_i) + n_i$ and n_i is additive measurement noise. Possible models for the classes of functions, f(**z**) and g(**w**), are the Gaussian processes, $f_{\mathbf{z}}$ and $g_{\mathbf{w}}$, respectively. It follows that $h_{\mathbf{z},\mathbf{w}} = (f_{\mathbf{z}} + g_{\mathbf{w}})$ is itself a Gaussian process and a possible model for the class of functions, $h(\mathbf{z},\mathbf{w})$. With the mean functions of $f_{\mathbf{z}}$ and $g_{\mathbf{w}}$ zero, the mean function of $h_{\mathbf{z},\mathbf{w}}$ is zero and its covariance function is the sum of the covariance functions of $f_{\mathbf{z}}$ and $g_{\mathbf{w}}$. The joint prior probability distribution for $\mathbf{F} = [f_{\mathbf{z}_1} \quad \cdots \quad f_{\mathbf{z}_N}]^{\mathrm{T}}$, $\mathbf{G} = [g_{\mathbf{w}_1} \quad \cdots \quad g_{\mathbf{w}_N}]^{\mathrm{T}}$ and $\mathbf{H} = \mathbf{F} + \mathbf{G} = [I \quad I][\mathbf{F}^{\mathrm{T}} \quad \mathbf{G}^{\mathrm{T}}]^{\mathrm{T}}$ has mean zero and covariance matrix

$$\mathrm{E}\left[\begin{bmatrix}\mathbf{F}\\ \mathbf{G}\\ \mathbf{H}\end{bmatrix}\begin{bmatrix}\mathbf{F}^{\mathrm{T}} & \mathbf{G}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}}\end{bmatrix}\right] = \begin{bmatrix}\Lambda_{\mathrm{FF}} & \Lambda_{\mathrm{FG}} & \Lambda_{\mathrm{FH}}\\ \Lambda_{\mathrm{GF}} & \Lambda_{\mathrm{GG}} & \Lambda_{\mathrm{GH}}\\ \Lambda_{\mathrm{HF}} & \Lambda_{\mathrm{HG}} & \Lambda_{\mathrm{HH}}\end{bmatrix} \tag{40}$$

where $\Lambda_{\mathrm{FF}} = \mathrm{E}[\mathbf{F}\mathbf{F}^{\mathrm{T}}]$, $\Lambda_{\mathrm{GG}} = \mathrm{E}[\mathbf{G}\mathbf{G}^{\mathrm{T}}]$, $\Lambda_{\mathrm{HH}} = \mathrm{E}[\mathbf{H}\mathbf{H}^{\mathrm{T}}]$, $\Lambda_{\mathrm{FG}} = \Lambda_{\mathrm{GF}}^{\mathrm{T}} = \mathrm{E}[\mathbf{F}\mathbf{G}^{\mathrm{T}}]$, $\Lambda_{\mathrm{FH}} = \Lambda_{\mathrm{HF}}^{\mathrm{T}} = \mathrm{E}[\mathbf{F}\mathbf{H}^{\mathrm{T}}]$ and $\Lambda_{\mathrm{GH}} = \Lambda_{\mathrm{HG}}^{\mathrm{T}} = \mathrm{E}[\mathbf{G}\mathbf{H}^{\mathrm{T}}]$. Note that

$$\begin{bmatrix}\Lambda_{\mathrm{FH}}\\ \Lambda_{\mathrm{GH}}\end{bmatrix} = \begin{bmatrix}\Lambda_{\mathrm{FF}} + \Lambda_{\mathrm{FG}}\\ \Lambda_{\mathrm{GF}} + \Lambda_{\mathrm{GG}}\end{bmatrix} = \begin{bmatrix}\Lambda_{\mathrm{FF}} & \Lambda_{\mathrm{FG}}\\ \Lambda_{\mathrm{GF}} & \Lambda_{\mathrm{GG}}\end{bmatrix}\begin{bmatrix}I\\ I\end{bmatrix} \tag{41}$$

and

$$\Lambda_{\mathrm{HH}} = \Lambda_{\mathrm{FF}} + \Lambda_{\mathrm{FG}} + \Lambda_{\mathrm{GF}} + \Lambda_{\mathrm{GG}} = \begin{bmatrix}I & I\end{bmatrix}\begin{bmatrix}\Lambda_{\mathrm{FF}} & \Lambda_{\mathrm{FG}}\\ \Lambda_{\mathrm{GF}} & \Lambda_{\mathrm{GG}}\end{bmatrix}\begin{bmatrix}I\\ I\end{bmatrix} \tag{42}$$

Conditioning on the data in the usual manner, the joint posterior probability distribution has mean vector and covariance matrix

$$\begin{bmatrix}\hat{\mathbf{F}}\\ \hat{\mathbf{G}}\\ \hat{\mathbf{H}}\end{bmatrix}=\begin{bmatrix}\Lambda_{FH}\\ \Lambda_{GH}\\ \Lambda_{HH}\end{bmatrix}Q^{-1}\mathbf{Y} \text{ and } \begin{bmatrix}\Lambda_{FF} & \Lambda_{FG} & \Lambda_{FH}\\ \Lambda_{GF} & \Lambda_{GG} & \Lambda_{GH}\\ \Lambda_{HF} & \Lambda_{HG} & \Lambda_{HH}\end{bmatrix}-\begin{bmatrix}\Lambda_{FF}\\ \Lambda_{GG}\\ \Lambda_{HH}\end{bmatrix}Q^{-1}\begin{bmatrix}\Lambda_{FF} & \Lambda_{GG} & \Lambda_{HH}\end{bmatrix}$$

respectively, where $\mathbf{Y}^{\mathrm{T}}=\begin{bmatrix}y_1 & \cdots & y_N\end{bmatrix}^{\mathrm{T}}$ and $Q=\Lambda_{HH}+\mathbf{B}$ with $\mathbf{B}=\mathrm{E}\left[\begin{bmatrix}n_1 & \cdots & n_N\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}n_1 & \cdots & n_N\end{bmatrix}\right]$, the measurement noise covariance matrix.

When the correct values, **G**, are known, then Gaussian regression could then be applied to the data, $\mathbf{Y}-\mathbf{G}$, with the underlying nonlinear relationship modelled by $f_{\mathbf{z}}$ alone. The prediction for **F**, given **Y** and **G**, is $\hat{\mathbf{F}}=\Lambda_{FF}Q_F^{-1}\left[\mathbf{Y}-\mathbf{G}\right]$, where $Q_F=\Lambda_{FF}+\mathbf{B}$. Similarly, when the correct values, **F**, are known, the prediction for **G**, given **Y** and **F**, is $\hat{\mathbf{G}}=\Lambda_{GG}Q_G^{-1}\left[\mathbf{Y}-\mathbf{F}\right]$, where $Q_G=\Lambda_{GG}+\mathbf{B}$. Hence, for consistency, assuming that the predictions for **F** and **G** are reasonably accurate,

$$\begin{bmatrix}\hat{\mathbf{F}}\\ \hat{\mathbf{G}}\end{bmatrix}=\begin{bmatrix}\Lambda_{FF}Q_F^{-1}[\mathbf{Y}-\mathbf{G}]\\ \Lambda_{GG}Q_G^{-1}[\mathbf{Y}-\mathbf{F}]\end{bmatrix}\approx\begin{bmatrix}\Lambda_{FF}Q_F^{-1}[\mathbf{Y}-\hat{\mathbf{G}}]\\ \Lambda_{GG}Q_G^{-1}[\mathbf{Y}-\hat{\mathbf{F}}]\end{bmatrix} \tag{43}$$

When the two Gaussian processes, $f_{\mathbf{z}}$ and $g_{\mathbf{w}}$, are independent, then the cross-correlation between them is zero; that is, $\Lambda_{FG}=\Lambda_{GF}=0$. The predictions for **F** and **G** then become

$$\begin{bmatrix}\hat{\mathbf{F}}\\ \hat{\mathbf{G}}\end{bmatrix}=\begin{bmatrix}\Lambda_{FF}\\ \Lambda_{GG}\end{bmatrix}Q^{-1}\mathbf{Y} \tag{44}$$

with $Q=\Lambda_{FF}+\Lambda_{GG}+\mathbf{B}$. In addition, since $\begin{bmatrix}\mathbf{Y}-\hat{\mathbf{G}}\\ \mathbf{Y}-\hat{\mathbf{F}}\end{bmatrix}=\begin{bmatrix}\Lambda_{FF}+\mathbf{B}\\ \Lambda_{GG}+\mathbf{B}\end{bmatrix}Q^{-1}\mathbf{Y}$,

$$\begin{bmatrix}\hat{\mathbf{F}}\\ \hat{\mathbf{G}}\end{bmatrix}=\begin{bmatrix}\Lambda_{FF}Q_F^{-1}[\mathbf{Y}-\hat{\mathbf{G}}]\\ \Lambda_{GG}Q_G^{-1}[\mathbf{Y}-\hat{\mathbf{F}}]\end{bmatrix} \tag{45}$$

Hence, the necessary self-consistency condition, (43), for a prediction to be reasonably accurate is met when $f_{\mathbf{z}}$ and $g_{\mathbf{w}}$ are independent.

The above discussion suggests the following Gaussian regression approach to the extraction of the two components. The prior model for the class of possible function pairs, $(f(\mathbf{z}),g(\mathbf{w}))$, is modelled by the pair $(f_{\mathbf{z}},g_{\mathbf{w}})$, where $f_{\mathbf{z}}$ and $g_{\mathbf{w}}$ are independent

zero mean Gaussian processes with covariance functions $C_f(\mathbf{z},\mathbf{z}')$ and $C_g(\mathbf{w},\mathbf{w}')$, respectively. The pair, $(f_{\mathbf{z}}, g_{\mathbf{w}})$, thus has zero joint mean function and joint covariance function

$$\mathrm{E}\left[\begin{bmatrix} f_{\mathbf{z}} \\ g_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} f_{\mathbf{z}'} & g_{\mathbf{w}'} \end{bmatrix}\right] = \begin{bmatrix} C_f(\mathbf{z},\mathbf{z}') & 0 \\ 0 & C_g(\mathbf{w},\mathbf{w}') \end{bmatrix} \tag{46}$$

When conditioned in the usual manner on the data **Y**, the posterior model is the Gaussian process pair, $(\tilde{f}_{\mathbf{z}}, \tilde{g}_{\mathbf{w}})$, with respectively, joint mean function and joint covariance function

$$\begin{bmatrix} \tilde{m}_f(\mathbf{z}) \\ \tilde{m}_g(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} \hat{f}_{\mathbf{z}} \\ \hat{g}_{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathbf{z}F} \\ \Lambda_{\mathbf{w}G} \end{bmatrix} Q^{-1}\mathbf{Y}$$

and (47)

$$\begin{bmatrix} \tilde{C}_{ff}(\mathbf{z},\mathbf{z}') & \tilde{C}_{fg}(\mathbf{z},\mathbf{w}') \\ \tilde{C}_{gf}(\mathbf{w},\mathbf{z}') & \tilde{C}_{gg}(\mathbf{w},\mathbf{w}') \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathbf{z}\mathbf{z}'} - \Lambda_{\mathbf{z}F}Q^{-1}\Lambda_{F\mathbf{z}'} & -\Lambda_{\mathbf{z}F}Q^{-1}\Lambda_{F\mathbf{w}'} \\ -\Lambda_{\mathbf{w}F}Q^{-1}\Lambda_{F\mathbf{z}'} & \Lambda_{\mathbf{w}\mathbf{w}'} - \Lambda_{\mathbf{w}F}Q^{-1}\Lambda_{F\mathbf{w}'} \end{bmatrix}$$

where $\Lambda_{\mathbf{z}\mathbf{z}'} = \mathrm{E}[f_{\mathbf{z}} f_{\mathbf{z}'}]$, $\Lambda_{\mathbf{w}\mathbf{w}'} = \mathrm{E}[g_{\mathbf{w}} g_{\mathbf{w}'}]$, $\Lambda_{\mathbf{z}F} = \Lambda_{F\mathbf{z}}^{\mathrm{T}} = \mathrm{E}[f_{\mathbf{z}} F]$ and $\Lambda_{\mathbf{w}G} = \Lambda_{G\mathbf{w}}^{\mathrm{T}} = \mathrm{E}[g_{\mathbf{w}} G]$. The associated prior model for the combined nonlinear relationship, $h(\mathbf{z},\mathbf{w})$, is the Gaussian process, $h_{\mathbf{w},\mathbf{z}} = f_{\mathbf{z}} + g_{\mathbf{w}}$. Its mean is zero and its compound covariance function is

$$C_h[(\mathbf{w},\mathbf{z})(\mathbf{w}',\mathbf{z}')] = C_f(\mathbf{z},\mathbf{z}') + C_g(\mathbf{w},\mathbf{w}')$$

When conditioned on the data, **Y**, the posterior model for h(**z**,**w**) has mean function and covariance function

$$\tilde{m}_h(\mathbf{w},\mathbf{z}) = \hat{h}_{\mathbf{w},\mathbf{z}} = \Lambda_{(\mathbf{w},\mathbf{z})H}Q^{-1}\mathbf{Y} \text{ and}$$

$$\tilde{C}_h((\mathbf{w},\mathbf{z}),(\mathbf{w}',\mathbf{z}')) = \Lambda_{(\mathbf{w},\mathbf{z})(\mathbf{w}',\mathbf{z}')} - \Lambda_{(\mathbf{w},\mathbf{z})H}Q^{-1}\Lambda_{H(\mathbf{w}',\mathbf{z}')}$$

where $\Lambda_{(\mathbf{w},\mathbf{z})(\mathbf{w}',\mathbf{z}')} = \mathrm{E}[h_{\mathbf{w},\mathbf{z}} h_{\mathbf{w}',\mathbf{z}'}]$ and $\Lambda_{(\mathbf{w},\mathbf{z})H} = \mathrm{E}[h_{\mathbf{w},\mathbf{z}}\mathbf{H}]$. Note that

$$\Lambda_{HH} = \Lambda_{FF} + \Lambda_{GG},\ Q = \Lambda_{HH} + \mathbf{B} = \Lambda_{FF} + \Lambda_{GG} + \mathbf{B}$$

and

$$\Lambda_{(\mathbf{w},\mathbf{z})(\mathbf{w}',\mathbf{z}')} = \Lambda_{\mathbf{z}\mathbf{z}'} + \Lambda_{\mathbf{w}\mathbf{w}'},\ \Lambda_{(\mathbf{w},\mathbf{z})H} = \Lambda_{\mathbf{z}F} + \Lambda_{\mathbf{w}G}$$

The above approach to extracting components is applied below to an example in which the nonlinear relationship has two components with different explanatory variables.

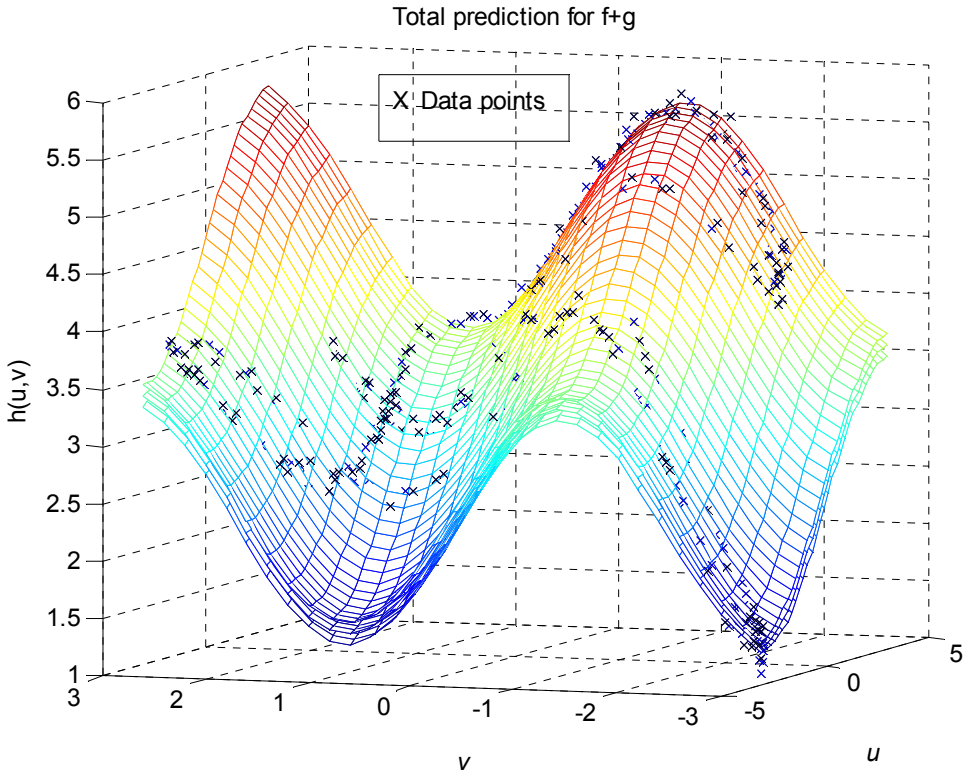


Figure 19 Data, prediction and confidence intervals.

Example 4.1: The two components are $\Phi(u) = 2 + \tanh(0.5u) + 0.1u$ and $\Gamma(v) = 1.5 - \sin(1.5v)$ with scalar explanatory variables, u and v, respectively. The domains of the explanatory variables are $[-3 \le u \le 3]$ and $[-3 \le v \le 3]$. The data set consists of 600 measurements, $\{y_i, (u_i, v_i)\}_{i=1}^{N=600}$, with additive Gaussian white noise of variance 0.01; that is, $y_i = \Phi(u_i) + \Gamma(v_i) + n_i$ with n_i the additive noise. The explanatory variable pairs, $\{(u_i, v_i)\}_{i=1}^{N=600}$, lie on a reasonably smooth trajectory in the (u_i, v_i) plane. The noisy data, denoted by crosses, is shown in Figure 19.

The prior model for the components are the zero mean independent Gaussian processes, Φ_u and Γ_v, with the exponential squared covariance functions, (48),

$$C_\Phi(u,u') = a_\Phi \exp\left(-d_\Phi (u-u')^2\right) \text{ and } C_\Gamma(v,v') = a_\Gamma \exp\left(-d_\Gamma (v-v')^2\right) \qquad (48)$$

The equivalent single Gaussian process prior model, $\mathrm{H}_{u,v} = \Phi_u + \Gamma_v$, is mean zero with compound covariance function $\left(C_\Phi(u,u') + C_\Gamma(v,v')\right)$. The correlation between two noise values, n_i and n_j, is $b\delta_{ij}$, where δ_{ij} is the kronecker-delta and between two measurements, y_i and y_j, is

$$\mathrm{E}\left[y_i, y_j\right] = C_\Phi\left(u_i, u_j\right) + C_\Gamma\left(v_i, v_j\right) + b\delta_{ij} \tag{49}$$

Applying Gaussian regression based on the single Gaussian process, $\mathrm{H}_{u,v}$, the values for the hyperparameters are determined by maximising the likelihood of the data, $\mathbf{Y} = \left[y_1, \cdots, y_N\right]^{\mathrm{T}}$, given the correlation, (49), between the data points. The hyperparameters values so obtained, are a_Φ=1.512, d_Φ=0.0782, a_Γ=8.058, d_Γ=0.311 and b=0.0093. The prediction for $\mathrm{H}(u,v) = \Phi(u) + \Gamma(v)$ over the domain, $\left[-3 \le u \le 3\right]$ and $\left[-3 \le v \le 3\right]$, and the associated confidence intervals are shown in Figure 19. In addition, for i=1,…,N, the value, y_i, together with the prediction for $\mathrm{h}(u_i, v_i)$ and the confidence interval is plotted against the index, i, in Figure 20. Because the (u_i, v_i) lie on a smooth trajectory, the plotted curve in Figure 20 is also smooth.

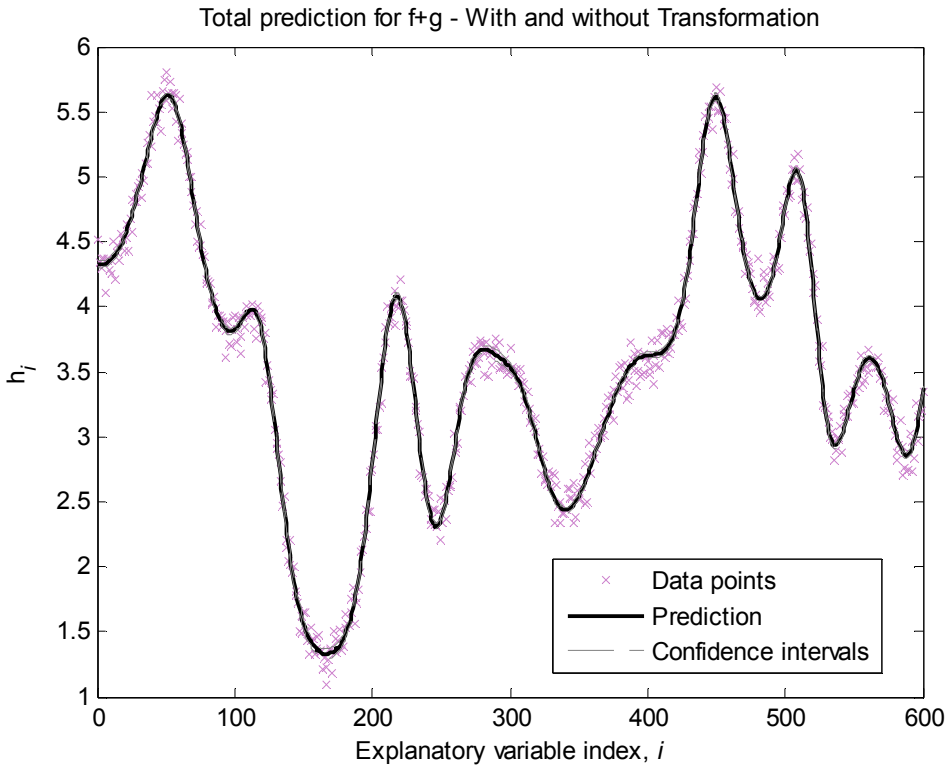


Figure 20 Data, total prediction and confidence intervals on time-series scale.

The two components $\Phi(u)$ and $\Gamma(v)$ are extracted using (47) as discussed above. The prediction and confidence interval for $\Phi(u)$ and $\Gamma(v)$, are depicted in Figure 21 and Figure 22, respectively.

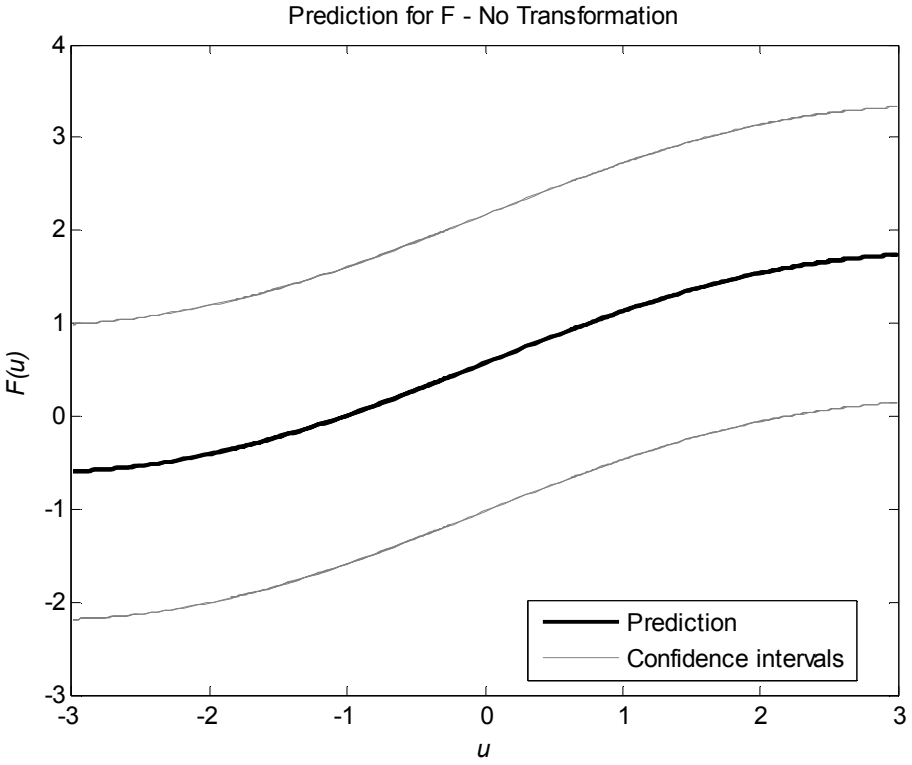


Figure 21 Prediction and confidence intervals for Φ.

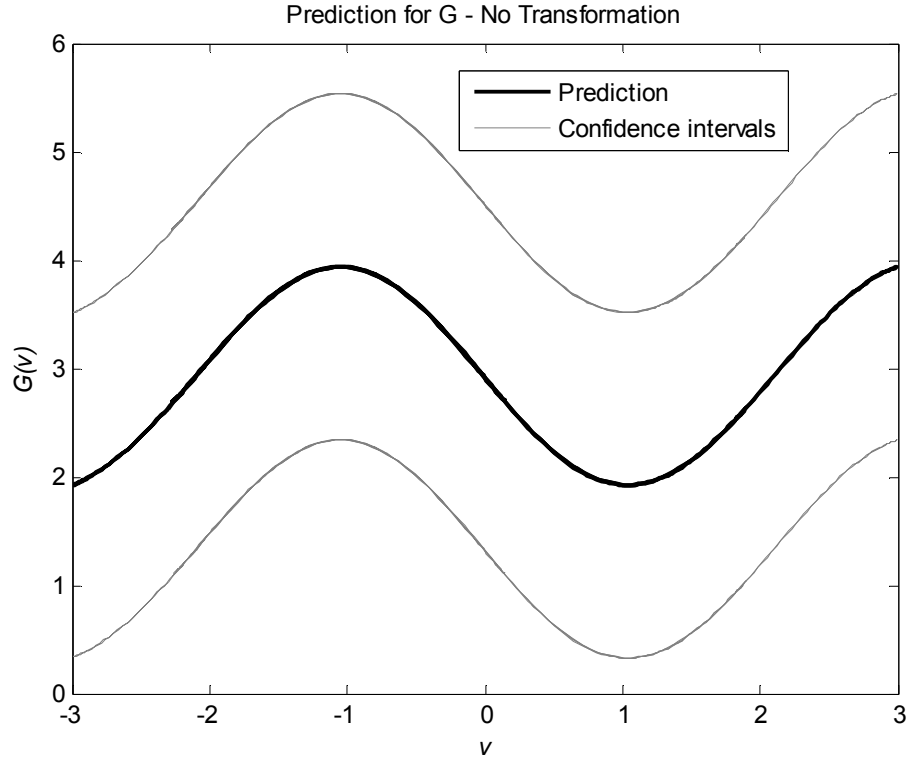


Figure 22 Prediction and confidence intervals for Γ.

Comparing the confidence intervals in Figure 21 and Figure 22 to the confidence interval in Figure 20, the former are clearly very much broader than the latter. Indeed, the confidence intervals for the two components are excessive and the predictions for the two components are too uncertain to be of any real value. This difficulty is not specific to Example 4.1 or to the different explanatory variable case but is generic, see §4.3 and §4.4. The method, based on a multiple Gaussian process prior model for extracting the components, investigated in this sub-section, is not effective. An improved approach is necessary.

4.3 Multiple Gaussian Processes Models with Different Explanatory Variables

In this sub-section, the case with the underlying nonlinear relationship having two components with different explanatory variables, i.e. $h(\mathbf{w},\mathbf{z})|_{\mathbf{w}=\mathbf{u},\mathbf{z}=\mathbf{v}} = f(\mathbf{u}) + g(\mathbf{v})$ with $\mathbf{u} \neq \mathbf{v}$, is considered. The requirement remains to extract either or both of $f(\mathbf{u})$ and $g(\mathbf{v})$.

4.3.1 Cause of Excessively Wide Confidence Intervals

The method for extracting the two components, suggested in §4.2, fails when applied in Example 4.1. The problem is the very broad confidence intervals and so excessive uncertainty in the separate predictions for the two components. These very broad confidence intervals arise because an arbitrary constant can be added to $f(\mathbf{u})$ and subtracted from $g(\mathbf{v})$ without changing $h(\mathbf{u},\mathbf{v})$. The measured values are unchanged as is the likelihood of the data. The confidence intervals for $f(\mathbf{u})$ and $g(\mathbf{v})$ include a large factor to account for the uncertainty introduced by this arbitrary constant. This issue is not peculiar to Example 4.1 but is generic pertaining whenever there is freedom to add a constant to $f(\mathbf{u})$, whilst subtracting it from $g(\mathbf{v})$. To improve the method, suggested in §4.2, requires the freedom to add/subtract a constant to be removed.

One approach to removing the uncertainty due to the arbitrary constant would be by at least one of the Gaussian processes, $f_{\mathbf{u}}$ and g_{v}, being a non-stationary Gaussian process that would exclude the addition or subtraction of an arbitrary constant. The effectiveness in this case of the method, suggested in §4.2, is illustrated by Example 4.2.

Example 4.2: The two components are $\Phi(\mathbf{u})$ and $\Gamma(v)$ with explanatory variables, $\mathbf{u} = (u_1, u_2)$ and scalar v. The dataset consists of 1,599 measurements, $\{y_i, (\mathbf{u}_i, v_i)\}_{i=1}^{N=1599}$, with additive Gaussian white noise; that is, $y_i = \Phi(\mathbf{u}_i) + \Gamma(v_i) + n_i$ with n_i the additive noise.

The prior model for the class of possible functions for $\Phi(\mathbf{u})$ is the zero mean Gaussian process, $\Phi_{\mathbf{u}}$, with the squared exponential covariance function, (50),

$$C_{\Phi}(\mathbf{u}, \mathbf{u}') = a_{\Phi} \exp\left\{-\frac{1}{2}\sum_{k=1}^{2} d_{\Phi_k}\left[(\mathbf{u})_k - (\mathbf{u}')_k\right]^2\right\}, \ \mathbf{u} = \{u_1, u_2\} \tag{50}$$

where $(\mathbf{u})_k$ is the k^{th} element of $\mathbf{u}$. Two prior models for the class of possible functions for $\Gamma(v)$ are considered, namely, the zero mean Gaussian process, Γ_{v}, corresponding to linear functions and to quadratic functions passing through the origin, see Appendix D. Both of these Gaussian processes are non-stationary and cannot accommodate the addition or subtraction of an arbitrary constant. The Gaussian processes, $\Phi_{\mathbf{u}}$ and Γ_{v}, are independent.

Using Linear Covariance Function

The covariance function for the Gaussian process, Γ_v, is

$$C_{\Gamma}(v_i, v_j) = w_{\Gamma} v_i v_j \tag{51}$$

It models the linear functions such that $\Gamma(0) = 0$. The equivalent Gaussian process prior model, $H_{\mathbf{u},v} = \Phi_{\mathbf{u}} + \Gamma_{v}$, is zero mean with compound covariance function $\left(C_{\Phi}(\mathbf{u}, \mathbf{u}') + C_{\Gamma}(v, v')\right)$. The correlation between two measurements, y_i and y_j, is

$$\mathrm{E}\left[\mathrm{y}_i, \mathrm{y}_j\right] = a_\Phi \exp\left\{-\frac{1}{2}\sum_{k=1}^{2} d_{\Phi_k}\left[(\mathbf{u}_i)_k - (\mathbf{u}_j)_k\right]^2\right\} + w_\Gamma v_i v_j + b\delta_{ij} \tag{52}$$

Applying Gaussian regression based on the single Gaussian process, $\mathrm{H}_{u,v}$, the values for the hyperparameters are determined by maximising the likelihood of the data, $\mathbf{Y} = \left[\mathrm{y}_1, \cdots, \mathrm{y}_N\right]^{\mathrm{T}}$, given the correlation, (52), between the data points. The hyperparameter values, so obtained, are $a_\Phi = 14.7357$, $d_{\Phi_1} = 2.308 \times 10^{-4}$, $d_{\Phi_2} = 2.084 \times 10^{-5}$, $w_\Gamma = 0.018$ and $b = 0.098$. For i=1,…,N, the prediction for $\mathrm{h}(\mathbf{u}_i, v_i)$ and the confidence interval is plotted in Figure 23.

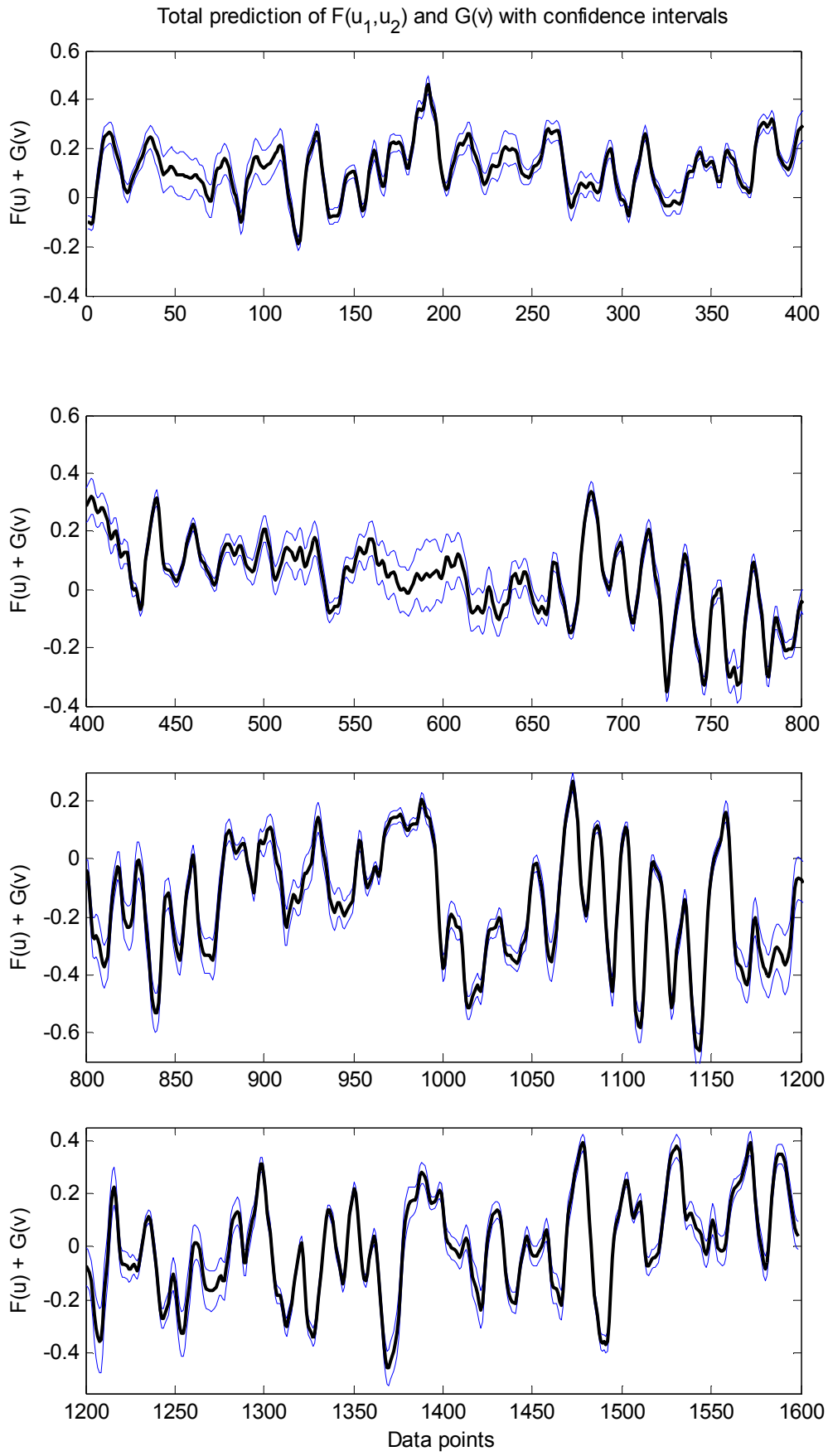


Figure 23 Total prediction for $\mathrm{h}(\mathbf{u}_i, v_i)$ and its confidence intervals.

The two components, $\Phi(\mathbf{u})$ and $\Gamma(v)$, are extracted in the usual manner using (47). The prediction and confidence interval for $\Phi(\mathbf{u})$ and $\Gamma(v)$ are depicted in Figure 24 and Figure 25, respectively.

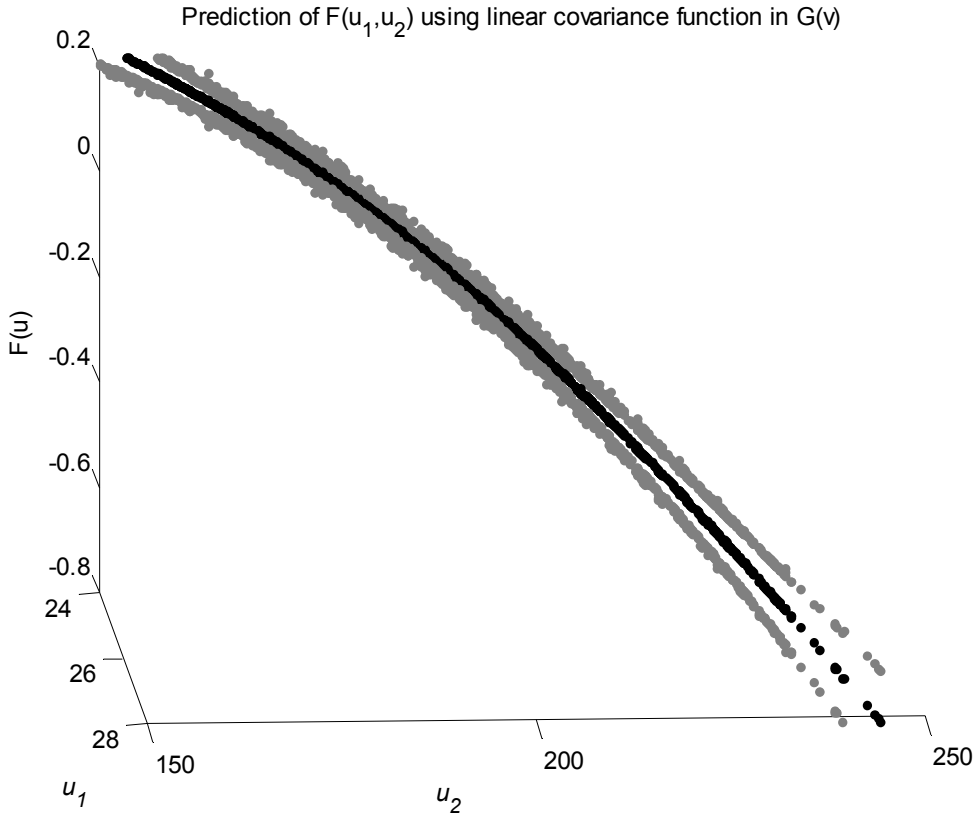


Figure 24 Prediction and confidence intervals of extracted Φ component with Γ_v having linear covariance function.

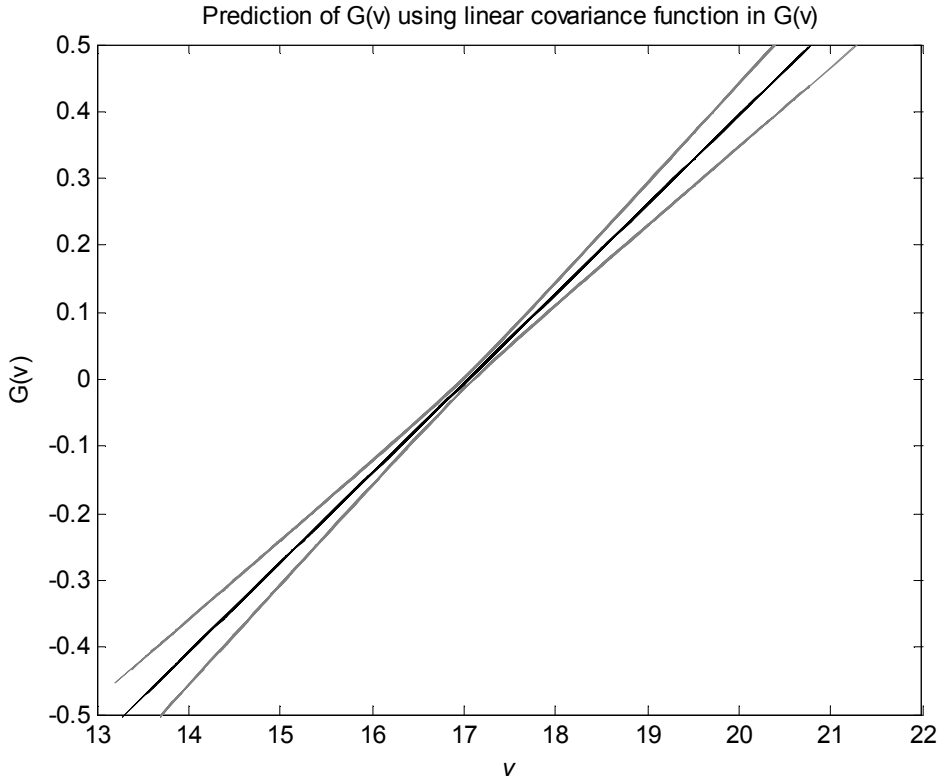


Figure 25 Prediction and confidence intervals of extracted Γ component with Γ_v having linear covariance function.

Comparing the confidence intervals in Figure 24 and Figure 25 to the confidence intervals in Figure 23, the former are no wider than the latter, in marked contrast to Example 4.1.

Using Quadratic Covariance Function

The covariance function for the Gaussian process, Γ_v, is given by (53).

$$C_\Gamma(v_i, v_j) = w_\Gamma (v_i)^2 (v_j)^2 \tag{53}$$

It models quadratic functions without a linear term such that Γ(0) = 0. The correlation between two measurements, y_i and y_j, is

$$\mathrm{E}[y_i, y_j] = a_\Phi \exp\left\{-\frac{1}{2}\sum_{k=1}^{2} d_{\Phi_k}\left[(\mathbf{u}_i)_k - (\mathbf{u}_j)_k\right]^2\right\} + w_\Gamma (v_i)^2 (v_j)^2 + b\delta_{ij} \tag{54}$$

Applying Gaussian regression based on the single Gaussian process, $\mathrm{H}_{u,v} = \Phi_{\boldsymbol{u}} + \Gamma_v$, the values for the hyperparameters are determined by maximising the likelihood of the data, $\mathbf{Y} = \left[\mathrm{y}_1, \cdots, \mathrm{y}_N\right]^{\mathrm{T}}$, given the correlation, (54), between the data points. The hyperparameter values, so obtained, are $a_\Phi = 2.51$, $d_{\Phi_1} = 1.3 \times 10^{-3}$, $d_{\Phi_2} = 5.05 \times 10^{-5}$, $w_\Gamma = 1.515 \times 10^{-5}$ and $b = 0.0979$. For i=1,…,N, the prediction for $\mathrm{h}\left(\mathbf{u}_i, v_i\right)$ and the confidence interval is plotted in Figure 26.

The two components, $\Phi(\mathbf{u})$ and $\Gamma(v)$, are extracted in the usual manner using (47). The prediction and confidence intervals for $\Phi(\mathbf{u})$ and $\Gamma(v)$ are depicted in Figure 27 and Figure 28, respectively.

Comparing the confidence intervals in Figure 27 and Figure 28 to the confidence intervals in Figure 26, the former are narrower than the latter, in contrast to Example 4.1.

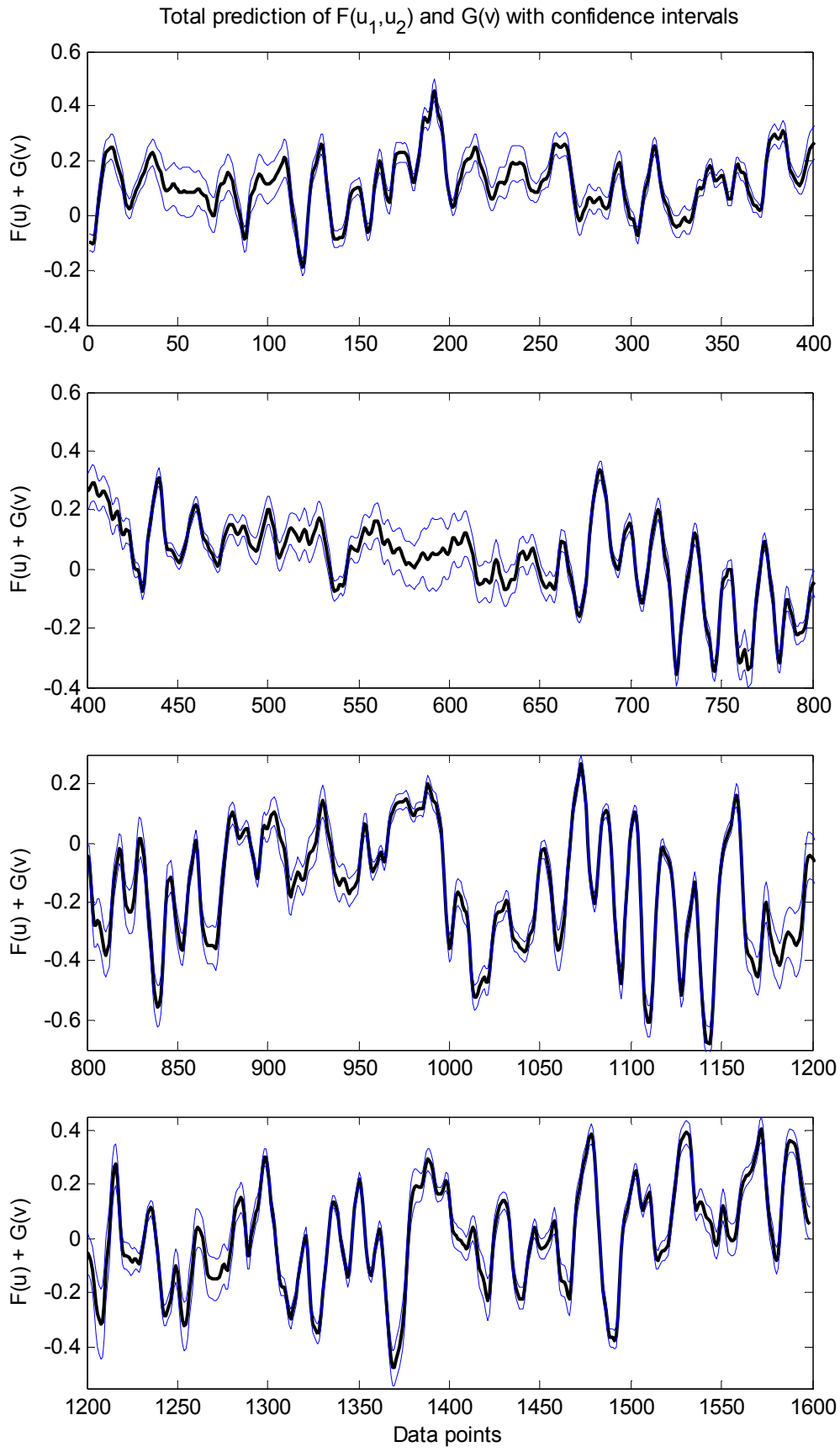


Figure 26 Prediction for $\mathrm{h}(\mathbf{u}, v)$ and its confidence intervals.

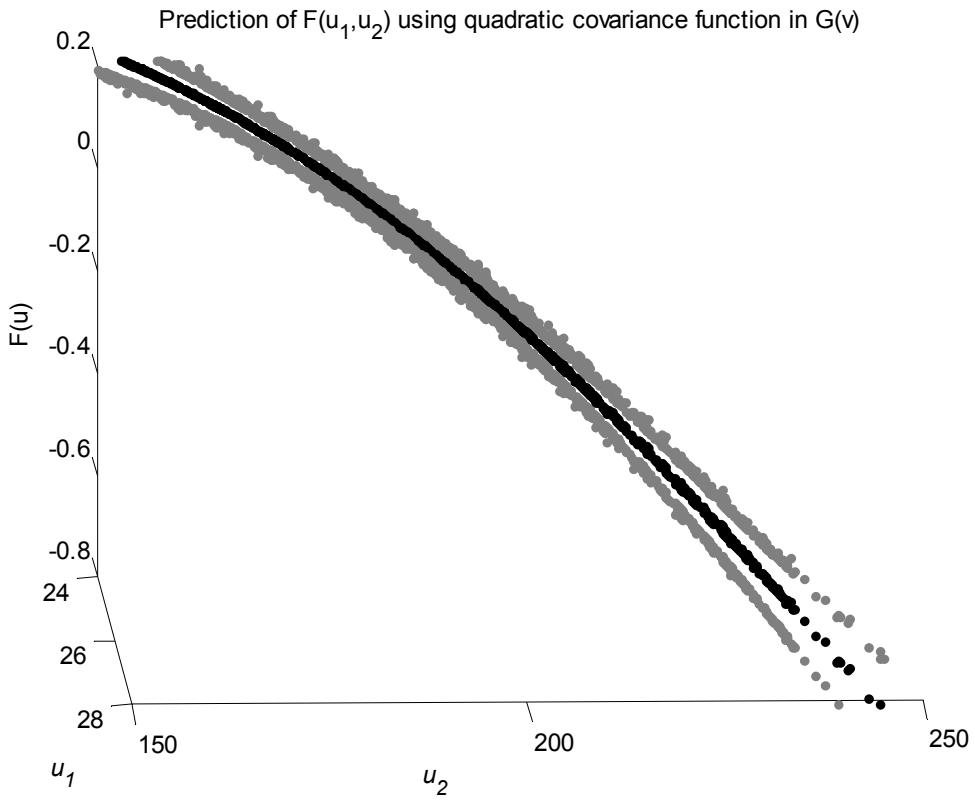


Figure 27 Prediction and confidence intervals of extracted Φ component with Γ_v using quadratic covariance function.

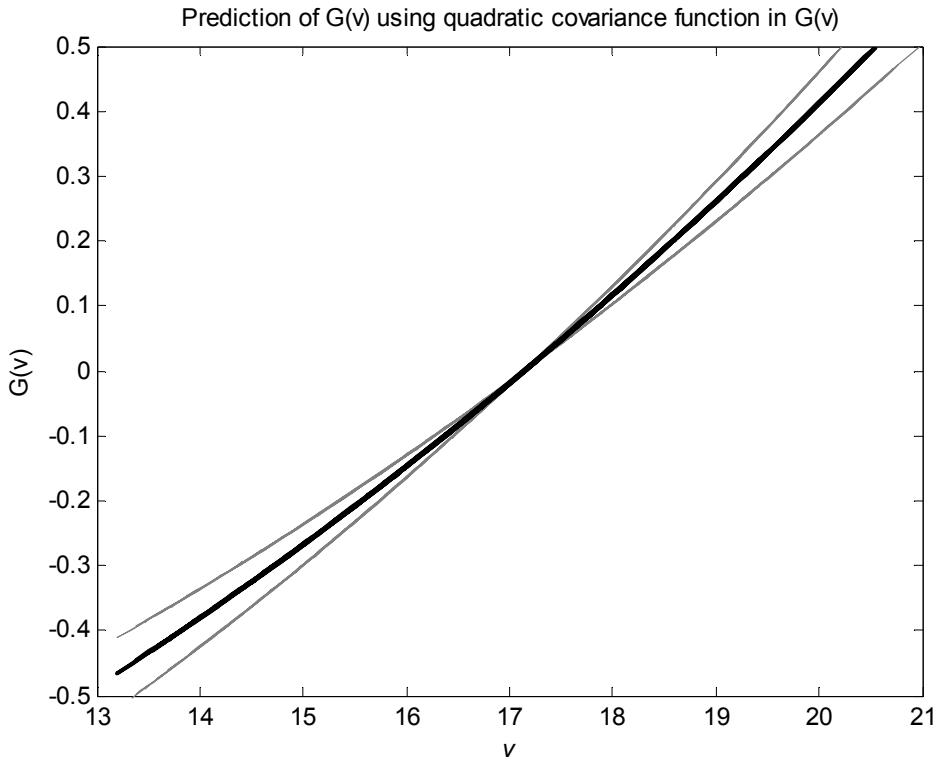


Figure 28 Prediction and confidence intervals of extracted Γ component with Γ_v using quadratic covariance function.

With both non-stationary Gaussian processes, Γ_v, considered in Example 4.2, the method based on two Gaussian process prior model for extracting the components, proposed in §4.2, is effective, providing useful predictions for the two components. Having identified the cause of the excessively wide confidence intervals observed in Example 4.1 and the remedy, specifically, to remove the freedom to add or subtract an arbitrary constant to the two components, the method for extracting components, discussed in §4.2, must be modified to remove that freedom in the general case.

4.3.2 Freedom of Choice in Two Gaussian Process Model

In §4.2, a Gaussian regression approach to extracting the two components, f(**z**) and g(**w**), from a nonlinear relationship, h(**z**,**w**)=f(**z**)+g(**w**), underlying measured data, $\{y_i,(\mathbf{z}_i,\mathbf{w}_i)\}_{i=1}^N$, is suggested. The approach is based on modelling the class of function pairs, (f(**z**),g(**w**)), by the independent Gaussian process pair, ($f_{\mathbf{z}}$,$g_{\mathbf{w}}$). However, the model is not uniquely determined by the data, only the single Gaussian

process model, $h_{\mathbf{z},\mathbf{w}}=f_{\mathbf{z}}+g_{\mathbf{w}}$, for the class of functions, $h(\mathbf{z},\mathbf{w})$, and there remains a substantial freedom of choice, see §4.3.1. A particular manifestation of this freedom of choice is explored in this Sub-section, specifically, the freedom to transform the Gaussian process pair, $(f_{\mathbf{z}},g_{\mathbf{w}})$.

Given the specific set of values of the explanatory variables, $\{\mathbf{z}_i,\mathbf{w}_i\}_{i=1}^{N}$, corresponding to the measurement data set, the transformed prior model is defined by the Gaussian process pair, $(f'_{\mathbf{z}},g'_{\mathbf{w}})$, such that

$$\begin{bmatrix} f'_{\mathbf{z}} \\ g'_{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} f_{\mathbf{z}} \\ g_{\mathbf{w}} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} = \left[\begin{array}{cc|c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \begin{bmatrix} f_{\mathbf{z}} \\ g_{\mathbf{w}} \\ \hline \mathbf{F} \\ \mathbf{G} \end{bmatrix} \tag{55}$$

and

$$\begin{bmatrix} \mathbf{F}' \\ \mathbf{G}' \end{bmatrix} = \mathrm{T} \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} \tag{56}$$

$$\text{with } \mathrm{T} = \left[I + \begin{bmatrix} T \\ \hline -T \end{bmatrix} \right] \text{ and } T^{\mathrm{T}} = \begin{bmatrix} T_{\mathbf{z}_1}^{\mathrm{T}} & \cdots & T_{\mathbf{z}_N}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} T_{\mathbf{w}_1}^{\mathrm{T}} & \cdots & T_{\mathbf{w}_N}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

where $\mathbf{F} = [f_{\mathbf{z}_1} \ \cdots \ f_{\mathbf{z}_N}]^{\mathrm{T}}$, $\mathbf{F}' = [f'_{\mathbf{z}_1} \ \cdots \ f'_{\mathbf{z}_N}]^{\mathrm{T}}$, $\mathbf{G} = [g_{\mathbf{w}_1} \ \cdots \ g_{\mathbf{w}_N}]^{\mathrm{T}}$ and $\mathbf{G}' = [g'_{\mathbf{w}_1} \ \cdots \ g'_{\mathbf{w}_N}]^{\mathrm{T}}$. Note that (56) does not necessitate the equality of $T_{\mathbf{z}}$ and $T_{\mathbf{w}}$ at other values of the explanatory variables. The transformed pair, $(f'_{\mathbf{z}},g'_{\mathbf{w}})$, has zero joint mean function and joint covariance function

$$\begin{bmatrix} C'_{\mathrm{ff}}(\mathbf{z},\mathbf{z}') & C'_{\mathrm{fg}}(\mathbf{z},\mathbf{w}') \\ C'_{\mathrm{gg}}(\mathbf{z},\mathbf{w}') & C'_{\mathrm{gg}}(\mathbf{w},\mathbf{w}') \end{bmatrix} = \mathrm{E}\left[\begin{bmatrix} f'_{\mathbf{z}} \\ g'_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} f'_{\mathbf{z}'} & g'_{\mathbf{w}'} \end{bmatrix} \right] = \begin{bmatrix} \Lambda'_{\mathbf{z}\mathbf{z}'} & \Lambda'_{\mathbf{z}\mathbf{w}'} \\ \Lambda'_{\mathbf{w}\mathbf{z}'} & \Lambda'_{\mathbf{w}\mathbf{w}'} \end{bmatrix}$$

$$= \left[\begin{array}{cc|c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \mathrm{E}\left[\left[\begin{array}{c} f_{\mathbf{z}} \\ g_{\mathbf{w}} \\ \hline \mathbf{F} \\ \mathbf{G} \end{array}\right] \left[\begin{array}{cc|cc} f_{\mathbf{z}'} & g_{\mathbf{w}'} & \mathbf{F}^T & \mathbf{G}^T \end{array}\right] \right] \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hline T_{\mathbf{z}}^{\mathrm{T}} & -T_{\mathbf{w}}^{\mathrm{T}} \end{array}\right] \tag{57}$$

$$
\begin{aligned}
&= \left[\begin{array}{cc:cc} 1 & 0 & T_{\mathbf{z}} & \\ 0 & 1 & -T_{\mathbf{w}} & \end{array}\right]
\left[\begin{array}{cc:cc} \Lambda_{\mathbf{zz}'} & \Lambda_{\mathbf{zw}'} & \Lambda_{\mathbf{z}\mathrm{F}} & \Lambda_{\mathbf{z}\mathrm{G}} \\ \Lambda_{\mathbf{zw}'} & \Lambda_{\mathbf{ww}'} & \Lambda_{\mathbf{w}\mathrm{F}} & \Lambda_{\mathbf{w}\mathrm{G}} \\ \hdashline \Lambda_{\mathrm{F}\mathbf{z}'} & \Lambda_{\mathrm{F}\mathbf{w}'} & \Lambda_{\mathrm{FF}} & \Lambda_{\mathrm{FG}} \\ \Lambda_{\mathrm{G}\mathbf{z}'} & \Lambda_{\mathrm{G}\mathbf{w}'} & \Lambda_{\mathrm{GF}} & \Lambda_{\mathrm{GG}} \end{array}\right]
\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hdashline T_{\mathbf{z}'}^{\mathrm{T}} & -T_{\mathbf{w}'}^{\mathrm{T}} \end{array}\right] \\
&= \begin{bmatrix} \Lambda_{\mathbf{zz}'} & 0 \\ 0 & \Lambda_{\mathbf{ww}'} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \Lambda_{\mathrm{F}\mathbf{z}'} & 0 \\ 0 & \Lambda_{\mathrm{G}\mathbf{w}'} \end{bmatrix} \\
&\quad + \begin{bmatrix} \Lambda_{\mathbf{z}\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathbf{w}\mathrm{G}} \end{bmatrix} \begin{bmatrix} T_{\mathbf{z}'}^{\mathrm{T}} & T_{\mathbf{w}'}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \Lambda_{\mathrm{FF}} & 0 \\ 0 & \Lambda_{\mathrm{GG}} \end{bmatrix} \begin{bmatrix} T_{\mathbf{z}'}^{\mathrm{T}} & T_{\mathbf{w}'}^{\mathrm{T}} \end{bmatrix}
\end{aligned}
$$

where $\Lambda'_{\mathbf{zz}'} = \mathrm{E}\left[\mathrm{f}'_{\mathbf{z}}\mathrm{f}'_{\mathbf{z}'}\right]$, $\Lambda'_{\mathbf{w}'\mathbf{z}} = \Lambda'_{\mathbf{zw}'} = \mathrm{E}\left[\mathrm{f}'_{\mathbf{z}}\mathrm{g}'_{\mathbf{w}'}\right]$ and $\Lambda'_{\mathbf{ww}'} = \mathrm{E}\left[\mathrm{g}'_{\mathbf{w}}\mathrm{g}'_{\mathbf{w}'}\right]$.

It follows that the prior probability distribution for $\left[\mathbf{F}'^{\mathrm{T}} \quad \mathbf{G}'^{\mathrm{T}}\right]^{\mathrm{T}}$ has zero mean and covariance matrix

$$
\mathrm{T}\begin{bmatrix} \Lambda_{\mathrm{FF}} & 0 \\ 0 & \Lambda_{\mathrm{GG}} \end{bmatrix}\mathrm{T}^{\mathrm{T}} \tag{58}
$$

The prior model for $\mathrm{h}_{\mathbf{w},\mathbf{z}}$, the prior joint probability distribution for **H** and **Y** and the likelihood of the data all remain unchanged since

$$
\mathrm{h}'_{\mathbf{z}_i,\mathbf{w}_i} = (\mathrm{f}_{\mathbf{z}_i} + \mathrm{g}_{\mathbf{w}_i}) + (T_{\mathbf{z}_i} - T_{\mathbf{w}_i})\begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} = \mathrm{h}_{\mathbf{z}_i,\mathbf{w}_i} \quad , \quad i = 1,\cdots,N
$$

$$
[I \quad I]\mathrm{T} = [I \quad I]
$$

and

$$
\mathbf{H}' = \begin{bmatrix} I & I \end{bmatrix}\begin{bmatrix} \mathbf{F}' \\ \mathbf{G}' \end{bmatrix} = \begin{bmatrix} I & I \end{bmatrix}\mathrm{T}\begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} = \begin{bmatrix} I & I \end{bmatrix}\begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} = \mathbf{H}
$$

Hence, the measured data set is equally well explained by all the transformed models, defined by (55) and (56); that is, with all possible choices of T.

Conditioning on the data in the usual manner, the posterior model becomes the Gaussian process pair, $(\widetilde{\mathrm{f}}'_{\mathbf{z}}, \widetilde{\mathrm{g}}'_{\mathbf{w}})$, with, respectively, joint mean function and joint covariance function

$$\begin{bmatrix} \widetilde{\mathrm{m}}'_{\mathrm{f}}(\mathbf{z}) \\ \widetilde{\mathrm{m}}'_{\mathrm{g}}(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{f}}'_{\mathbf{z}} \\ \hat{\mathrm{g}}'_{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} \Lambda'_{\mathbf{z}\mathrm{H}} \\ \Lambda'_{\mathbf{w}\mathrm{H}} \end{bmatrix} [\Lambda'_{\mathrm{HH}} + \mathbf{B}]^{-1} \mathbf{Y} = \mathrm{E}\left[\begin{bmatrix} \mathrm{f}'_{\mathbf{z}} \\ \mathrm{g}'_{\mathbf{w}} \end{bmatrix} \mathbf{H}' \right] [\Lambda'_{\mathrm{HH}} + \mathbf{B}]^{-1} \mathbf{Y}$$

$$= \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \mathrm{E}\left[\left[\begin{array}{c} \mathrm{f}_{\mathbf{z}} \\ \mathrm{g}_{\mathbf{w}} \\ \hdashline \mathbf{F} \\ \mathbf{G} \end{array}\right] \mathbf{H} \right] Q^{-1} \mathbf{Y} = \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathbf{z}\mathrm{H}} \\ \Lambda_{\mathbf{w}\mathrm{H}} \\ \hdashline \Lambda_{\mathrm{FH}} \\ \Lambda_{\mathrm{GH}} \end{array}\right] Q^{-1} \mathbf{Y} = \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \left[\begin{array}{c} \hat{\mathrm{f}}_{\mathbf{z}} \\ \hat{\mathrm{g}}_{\mathbf{w}} \\ \hdashline \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{array}\right]$$

$$= \begin{bmatrix} \hat{\mathrm{f}}_{\mathbf{z}} \\ \hat{\mathrm{g}}_{\mathbf{w}} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{bmatrix}$$

and

$$\begin{bmatrix} \widetilde{C}'_{\mathrm{ff}}(\mathbf{z},\mathbf{z}') & \widetilde{C}'_{\mathrm{fg}}(\mathbf{z},\mathbf{w}') \\ \widetilde{C}'_{\mathrm{gf}}(\mathbf{z},\mathbf{w}') & \widetilde{C}'_{\mathrm{gg}}(\mathbf{w},\mathbf{w}') \end{bmatrix} = \begin{bmatrix} \Lambda'_{\mathbf{z}\mathbf{z}'} & \Lambda'_{\mathbf{z}\mathbf{w}'} \\ \Lambda'_{\mathbf{w}\mathbf{z}'} & \Lambda'_{\mathbf{w}\mathbf{w}'} \end{bmatrix} - \begin{bmatrix} \Lambda'_{\mathbf{z}\mathrm{H}} \\ \Lambda'_{\mathbf{w}\mathrm{H}} \end{bmatrix} [\Lambda'_{\mathrm{HH}} + \mathbf{B}]^{-1} \begin{bmatrix} \Lambda'_{\mathrm{H}\mathbf{z}'} & \Lambda'_{\mathrm{H}\mathbf{w}'} \end{bmatrix}$$

$$= \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \left[\begin{array}{cc:cc} \Lambda_{\mathbf{z}\mathbf{z}'} & 0 & \Lambda_{\mathbf{z}\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathbf{w}\mathbf{w}'} & 0 & \Lambda_{\mathbf{w}\mathrm{G}} \\ \hdashline \Lambda_{\mathrm{F}\mathbf{z}'} & 0 & \Lambda_{\mathrm{FF}} & 0 \\ 0 & \Lambda_{\mathrm{G}\mathbf{w}'} & 0 & \Lambda_{\mathrm{GG}} \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hdashline T_{\mathbf{z}'}^{\mathrm{T}} & -T_{\mathbf{w}'}^{\mathrm{T}} \end{array}\right]$$

$$- \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \left[\begin{array}{c} \Lambda_{\mathbf{z}\mathrm{H}} \\ \Lambda_{\mathbf{w}\mathrm{H}} \\ \hdashline \Lambda_{\mathrm{FH}} \\ \Lambda_{\mathrm{GH}} \end{array}\right] Q^{-1} \left[\begin{array}{cc:cc} \Lambda_{\mathrm{H}\mathbf{z}'} & \Lambda_{\mathrm{H}\mathbf{w}'} & \Lambda_{\mathrm{HF}} & \Lambda_{\mathrm{HG}} \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hdashline T_{\mathbf{z}'}^{\mathrm{T}} & -T_{\mathbf{w}'}^{\mathrm{T}} \end{array}\right]$$

$$= \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \left[\begin{array}{cc:cc} \widetilde{\Lambda}_{\mathbf{z}\mathbf{z}'} & \widetilde{\Lambda}_{\mathbf{z}\mathbf{w}'} & \widetilde{\Lambda}_{\mathbf{z}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{z}\mathrm{G}} \\ \widetilde{\Lambda}_{\mathbf{z}\mathbf{w}'} & \widetilde{\Lambda}_{\mathbf{w}\mathbf{w}'} & \widetilde{\Lambda}_{\mathbf{w}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{w}\mathrm{G}} \\ \hdashline \widetilde{\Lambda}_{\mathrm{F}\mathbf{z}'} & \widetilde{\Lambda}_{\mathrm{F}\mathbf{w}'} & \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{G}\mathbf{z}'} & \widetilde{\Lambda}_{\mathrm{G}\mathbf{w}'} & \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hdashline T_{\mathbf{z}'}^{\mathrm{T}} & -T_{\mathbf{w}'}^{\mathrm{T}} \end{array}\right]$$

$$= \begin{bmatrix} \widetilde{\Lambda}_{\mathbf{z}\mathbf{z}'} & \widetilde{\Lambda}_{\mathbf{z}\mathbf{w}'} \\ \widetilde{\Lambda}_{\mathbf{w}\mathbf{z}'} & \widetilde{\Lambda}_{\mathbf{w}\mathbf{w}'} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{F}\mathbf{z}'} & \widetilde{\Lambda}_{\mathrm{F}\mathbf{w}'} \\ \widetilde{\Lambda}_{\mathrm{G}\mathbf{z}'} & \widetilde{\Lambda}_{\mathrm{G}\mathbf{w}'} \end{bmatrix}$$

$$+ \begin{bmatrix} \widetilde{\Lambda}_{\mathbf{z}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{z}\mathrm{G}} \\ \widetilde{\Lambda}_{\mathbf{w}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{w}\mathrm{G}} \end{bmatrix} \begin{bmatrix} T_{\mathbf{z}'}^{\mathrm{T}} & T_{\mathbf{w}'}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} \begin{bmatrix} T_{\mathbf{z}'}^{\mathrm{T}} & T_{\mathbf{w}'}^{\mathrm{T}} \end{bmatrix}$$

where

$$\begin{bmatrix} \widetilde{\Lambda}_{\mathbf{z}\mathbf{z}'} & \widetilde{\Lambda}_{\mathbf{z}\mathbf{w}'} \\ \widetilde{\Lambda}_{\mathbf{w}\mathbf{z}'} & \widetilde{\Lambda}_{\mathbf{w}\mathbf{w}'} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathbf{z}\mathbf{z}'} & 0 \\ 0 & \Lambda_{\mathbf{w}\mathbf{w}'} \end{bmatrix} - \begin{bmatrix} \Lambda_{\mathbf{z}\mathrm{H}} \\ \Lambda_{\mathbf{w}\mathrm{H}} \end{bmatrix} Q^{-1} \begin{bmatrix} \Lambda_{\mathrm{H}\mathbf{z}'} & \Lambda_{\mathrm{H}\mathbf{w}'} \end{bmatrix}$$

$$\begin{bmatrix} \widetilde{\Lambda}_{\mathrm{F}\mathbf{z}} & \widetilde{\Lambda}_{\mathrm{F}\mathbf{w}} \\ \widetilde{\Lambda}_{\mathrm{G}\mathbf{z}} & \widetilde{\Lambda}_{\mathrm{G}\mathbf{w}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \widetilde{\Lambda}_{\mathbf{z}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{z}\mathrm{G}} \\ \widetilde{\Lambda}_{\mathbf{w}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{w}\mathrm{G}} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathbf{z}\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathbf{w}\mathrm{G}} \end{bmatrix} - \begin{bmatrix} \Lambda_{\mathbf{z}\mathrm{H}} \\ \Lambda_{\mathbf{w}\mathrm{H}} \end{bmatrix} Q^{-1} \begin{bmatrix} \Lambda_{\mathrm{HF}} & \Lambda_{\mathrm{HG}} \end{bmatrix}$$

$$\begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{FG}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathrm{FF}} & 0 \\ 0 & \Lambda_{\mathrm{GG}} \end{bmatrix} - \begin{bmatrix} \Lambda_{\mathrm{FH}} \\ \Lambda_{\mathrm{GH}} \end{bmatrix} Q^{-1} \begin{bmatrix} \Lambda_{\mathrm{HF}} & \Lambda_{\mathrm{HG}} \end{bmatrix}$$

Hence,

$$\begin{bmatrix} \widetilde{\mathrm{f}}'_{\mathbf{z}} \\ \widetilde{\mathrm{g}}'_{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathrm{f}}_{\mathbf{z}} \\ \widetilde{\mathrm{g}}_{\mathbf{w}} \end{bmatrix} + \begin{bmatrix} T_{\mathbf{z}} \\ -T_{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix} = \left[\begin{array}{cc:c} 1 & 0 & T_{\mathbf{z}} \\ 0 & 1 & -T_{\mathbf{w}} \end{array}\right] \begin{bmatrix} \widetilde{\mathrm{f}}_{\mathbf{z}} \\ \widetilde{\mathrm{g}}_{\mathbf{w}} \\ \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix} \tag{59}$$

where $\widetilde{\mathbf{F}} = [\widetilde{\mathrm{f}}_{\mathbf{z}_1} \quad \cdots \quad \widetilde{\mathrm{f}}_{\mathbf{z}_N}]^{\mathrm{T}}$ and $\widetilde{\mathbf{G}} = [\widetilde{\mathrm{g}}_{\mathbf{w}_1} \quad \cdots \quad \widetilde{\mathrm{g}}_{\mathbf{w}_N}]^{\mathrm{T}}$.

It follows that the posterior probability distribution for $\begin{bmatrix} \mathbf{F}'^{\mathrm{T}} & \mathbf{G}'^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ has, respectively, mean vector and covariance matrix

$$\begin{bmatrix} \hat{\mathbf{F}}' \\ \hat{\mathbf{G}}' \end{bmatrix} = \begin{bmatrix} \Lambda'_{\mathrm{FH}} \\ \Lambda'_{\mathrm{GH}} \end{bmatrix} [\Lambda'_{\mathrm{HH}} + \mathbf{B}]^{-1} \mathbf{Y} = \mathrm{T} \begin{bmatrix} \Lambda_{\mathrm{FF}} \\ \Lambda_{\mathrm{GG}} \end{bmatrix} Q^{-1} \mathbf{Y} = \mathrm{T} \begin{bmatrix} \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{bmatrix} \tag{60}$$

and

$$\mathrm{T} \left(\begin{bmatrix} \Lambda_{\mathrm{FF}} & 0 \\ 0 & \Lambda_{\mathrm{GG}} \end{bmatrix} - \begin{bmatrix} \Lambda_{\mathrm{FF}} \\ \Lambda_{\mathrm{GG}} \end{bmatrix} Q^{-1} \begin{bmatrix} \Lambda_{\mathrm{FF}} & \Lambda_{\mathrm{GG}} \end{bmatrix} \right) \mathrm{T}^{\mathrm{T}} = \mathrm{T} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} \mathrm{T}^{\mathrm{T}} \tag{61}$$

Similarly to the prior, the posterior model for $\mathrm{h}_{\mathbf{z},\mathbf{w}}$ and the posterior joint probability distribution for **H** and **Y** remain unchanged from those for the untransformed Gaussian process pair model.

To distinguish and so choose between the possible models, information, additional to the measurement data, **Y**, is required. The nature of that additional information and the selected choice is driven by the application context.

Remark 4.1: Comparing (55) and (59), the Gaussian process pairs, $(\mathrm{f}'_{\mathbf{z}}, \mathrm{g}'_{\mathbf{w}})$ and $(\mathrm{f}_{\mathbf{z}}, \mathrm{g}_{\mathbf{w}})$, and the Gaussian process pairs, $(\widetilde{\mathrm{f}}'_{\mathbf{z}}, \widetilde{\mathrm{g}}'_{\mathbf{w}})$ and $(\widetilde{\mathrm{f}}_{\mathbf{z}}, \widetilde{\mathrm{g}}_{\mathbf{w}})$, are related by the same transformation. The prior and posterior probability distributions for $\begin{bmatrix} \mathbf{F}'^{\mathrm{T}} & \mathbf{G}'^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ are related to the prior and posterior probability distributions for $\begin{bmatrix} \mathbf{F}^{\mathrm{T}} & \mathbf{G}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ by the same transformation, T. Therefore, the choice of transformation, that is, the choice of

Gaussian process pair model, can equally well be made with respect to either prior or posterior models.

4.3.3 Improved Two GP Model with Different Explanatory Variables

Reverting to the case when two components with different explanatory variables, f(**u**) and g(**v**) with $\mathbf{u} \neq \mathbf{v}$, underlie the measured data $\{y_i, (\mathbf{u}_i, \mathbf{v}_i)\}_{i=1}^N$, the transformation of the Gaussian process pair ($f_{\mathbf{u}}$,$g_{\mathbf{v}}$), discussed in the sub-section 4.3.2, is exploited to remove the freedom to add/subtract an arbitrary constant to the components.

Given the specific set of values of the explanatory variables, $\{\mathbf{u}_i, \mathbf{v}_i\}_{i=1}^N$, corresponding to the measurement data set, the transformation (55) is chosen such that

$$T_{\mathbf{w}} = T_{\mathbf{z}} = [-\mathrm{X}^{\mathrm{T}} \,\vdots\, \mathrm{X}^{\mathrm{T}}]/(2N) \quad \text{with} \quad \mathrm{X} = [1 \;\; 1 \;\; \cdots \;\; 1]^{\mathrm{T}} \in \mathrm{P}^{N} \tag{62}$$

Hence, following §4.3.2 but with **z**=**u** and **w**=**v**, the prior model becomes the transformed Gaussian process pair

$$\begin{bmatrix} \mathrm{f}'_{\mathbf{u}} \\ \mathrm{g}'_{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathrm{f}_{\mathbf{u}} \\ \mathrm{g}_{\mathbf{v}} \end{bmatrix} + \tfrac{1}{2N} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} -\mathrm{X}^{\mathrm{T}} & \mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} = \left[\begin{array}{cc:cc} 1 & 0 & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \\ 0 & 1 & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \end{array}\right] \begin{bmatrix} \mathrm{f}_{\mathbf{u}} \\ \mathrm{g}_{\mathbf{v}} \\ \hdashline \mathbf{F} \\ \mathbf{G} \end{bmatrix}$$

and, on conditioning on the data, the posterior model becomes the transformed Gaussian process pair

$$\begin{bmatrix} \widetilde{\mathrm{f}}'_{\mathbf{u}} \\ \widetilde{\mathrm{g}}'_{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathrm{f}}_{\mathbf{u}} \\ \widetilde{\mathrm{g}}_{\mathbf{v}} \end{bmatrix} + \tfrac{1}{2N} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} -\mathrm{X}^{\mathrm{T}} & \mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix} = \left[\begin{array}{cc:cc} 1 & 0 & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \\ 0 & 1 & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \end{array}\right] \begin{bmatrix} \widetilde{\mathrm{f}}_{\mathbf{u}} \\ \widetilde{\mathrm{g}}_{\mathbf{v}} \\ \hdashline \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix}$$

Furthermore,

$$\begin{bmatrix} \mathbf{F}' \\ \mathbf{G}' \end{bmatrix} = \mathrm{T} \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} \text{ and } \begin{bmatrix} \widetilde{\mathbf{F}}' \\ \widetilde{\mathbf{G}}' \end{bmatrix} = \mathrm{T} \begin{bmatrix} \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix} \text{ with } \mathrm{T} = \begin{bmatrix} I - \mathrm{X}\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & \mathrm{X}\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \\ \mathrm{X}\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & I - \mathrm{X}\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \end{bmatrix}$$

Hence,

$$\begin{bmatrix} \mathrm{X}^{\mathrm{T}} & -\mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{F}' \\ \mathbf{G}' \end{bmatrix} = \begin{bmatrix} \mathrm{X}^{\mathrm{T}} & -\mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{F}}' \\ \widetilde{\mathbf{G}}' \end{bmatrix} = 0$$

or equivalently

$$\mathrm{X}^{\mathrm{T}}\mathbf{F}' = \mathrm{X}^{\mathrm{T}}\mathbf{G}' \text{ and } \mathrm{X}^{\mathrm{T}}\widetilde{\mathbf{F}}' = \mathrm{X}^{\mathrm{T}}\widetilde{\mathbf{G}}'$$

Remark 4.2: The prior model no longer belongs to the class of all Gaussian process pairs but to Σ_M, the class of all Gaussian process pairs $(\mathrm{f}_{\mathbf{u}}, \mathrm{g}_{\mathbf{v}})$, subject to the condition, $\mathrm{X}^{\mathrm{T}}\mathbf{F} = \mathrm{X}^{\mathrm{T}}\mathbf{G}$, where $\mathbf{F} = [\mathrm{f}_{\mathbf{u}_1} \ \cdots \ \mathrm{f}_{\mathbf{u}_N}]^{\mathrm{T}}$, $\mathbf{G} = [\mathrm{g}_{\mathbf{v}_1} \ \cdots \ \mathrm{g}_{\mathbf{v}_N}]^{\mathrm{T}}$ and $\mathrm{X} = [1 \ \cdots \ 1]^{\mathrm{T}} \in \mathrm{P}^N$. The posterior model belongs to the same set of Gaussian process pairs. The condition, $\mathrm{X}^{\mathrm{T}}\mathbf{F} = \mathrm{X}^{\mathrm{T}}\mathbf{G}$, precludes the addition or subtraction of an arbitrary constant to the individual components.

It follows from above that a suitable modified procedure for extracting the two components when they have different explanatory variables is the following.

1. Choose the hyperparameters on the basis of the single Gaussian process model, $\mathrm{h}_{\mathbf{z},\mathbf{w}}$, by maximising the likelihood of the data, $\mathbf{Y}$.
2. Determine the predictions and confidence intervals by the standard approach of Section 4.1 with a pair of independent Gaussian processes.
3. Modify the predictions and confidence intervals using the transformation, $T_{\mathbf{u}} = T_{\mathbf{v}} = [-\mathrm{X}^{\mathrm{T}} \ \vdots \ \mathrm{X}^{\mathrm{T}}]/(2N)$ with $\mathrm{X} = [1 \ \cdots \ 1]^{\mathrm{T}} \in \mathrm{P}^N$.

The prediction for $\mathrm{f}'_{\mathbf{u}}$ is

$$\hat{\mathrm{f}}'_{\mathbf{u}} = \hat{\mathrm{f}}_{\mathbf{u}} - \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}}(\hat{\mathbf{F}} - \hat{\mathbf{G}})$$

with variance

$$\begin{aligned} &\widetilde{\Lambda}_{\mathbf{u}\mathbf{u}'} - \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}}(\widetilde{\Lambda}_{\mathrm{F}\mathbf{u}'} - \widetilde{\Lambda}_{\mathrm{G}\mathbf{u}'}) - \tfrac{1}{2N}(\widetilde{\Lambda}_{\mathbf{u}\mathrm{F}} - \widetilde{\Lambda}_{\mathbf{u}\mathrm{G}})\mathrm{X} \\ &\quad + \left(\tfrac{1}{2N}\right)^2 \begin{bmatrix} \mathrm{X}^{\mathrm{T}} & -\mathrm{X}^{\mathrm{T}} \end{bmatrix} \mathrm{X}^{\mathrm{T}}(\widetilde{\Lambda}_{\mathrm{FF}} - \widetilde{\Lambda}_{\mathrm{FG}} - \widetilde{\Lambda}_{\mathrm{GF}} + \widetilde{\Lambda}_{\mathrm{GG}})\mathrm{X} \end{aligned}$$

and the prediction for $\mathrm{g}'_{\mathbf{v}}$ is

$$\hat{\mathrm{g}}'_{\mathbf{v}} = \hat{\mathrm{g}}_{\mathbf{v}} + \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}}(\hat{\mathbf{F}} - \hat{\mathbf{G}})$$

with variance

$$\widetilde{\Lambda}_{\mathbf{u}\mathbf{u}'} - \tfrac{1}{2N} C^{\mathrm{T}} (\widetilde{\Lambda}_{\mathrm{F}\mathbf{u}'} - \widetilde{\Lambda}_{\mathrm{G}\mathbf{u}'}) - \tfrac{1}{2N} (\widetilde{\Lambda}_{\mathbf{u}\mathrm{F}} - \widetilde{\Lambda}_{\mathbf{u}\mathrm{G}})\mathrm{X} + \left(\tfrac{1}{2N}\right)^2 \begin{bmatrix} \mathrm{X}^{\mathrm{T}} & -\mathrm{X}^{\mathrm{T}} \end{bmatrix} \mathrm{X}^{\mathrm{T}} (\widetilde{\Lambda}_{\mathrm{FF}} - \widetilde{\Lambda}_{\mathrm{FG}} - \widetilde{\Lambda}_{\mathrm{GF}} + \widetilde{\Lambda}_{\mathrm{GG}})\mathrm{X}$$

since the joint mean is

$$\begin{bmatrix} \hat{\mathrm{f}}'_{\mathbf{u}} \\ \hat{\mathrm{g}}'_{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{f}}_{\mathbf{u}} \\ \hat{\mathrm{g}}_{\mathbf{v}} \end{bmatrix} + \tfrac{1}{2N} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} -\mathrm{X}^{\mathrm{T}} & \mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{bmatrix} = \left[\begin{array}{cc:cc} 1 & 0 & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \\ 0 & 1 & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \end{array}\right] \begin{bmatrix} \hat{\mathrm{f}}_{\mathbf{u}} \\ \hat{\mathrm{g}}_{\mathbf{v}} \\ \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{bmatrix}$$

and the joint covariance is

$$\begin{bmatrix} \widetilde{\Lambda}_{\mathbf{u}\mathbf{u}'} & \widetilde{\Lambda}_{\mathbf{u}\mathbf{v}'} \\ \widetilde{\Lambda}_{\mathbf{v}\mathbf{u}'} & \widetilde{\Lambda}_{\mathbf{v}\mathbf{v}'} \end{bmatrix} + \begin{bmatrix} -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \\ \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{F}\mathbf{u}'} & \widetilde{\Lambda}_{\mathrm{F}\mathbf{v}'} \\ \widetilde{\Lambda}_{\mathrm{G}\mathbf{u}'} & \widetilde{\Lambda}_{\mathrm{G}\mathbf{v}'} \end{bmatrix} + \begin{bmatrix} \widetilde{\Lambda}_{\mathbf{u}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{u}\mathrm{G}} \\ \widetilde{\Lambda}_{\mathbf{v}\mathrm{F}} & \widetilde{\Lambda}_{\mathbf{v}\mathrm{G}} \end{bmatrix} \begin{bmatrix} -\tfrac{1}{2N}\mathrm{X} & \tfrac{1}{2N}\mathrm{X} \\ \tfrac{1}{2N}\mathrm{X} & -\tfrac{1}{2N}\mathrm{X} \end{bmatrix}$$
$$+ \begin{bmatrix} -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \\ \tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} & -\tfrac{1}{2N}\mathrm{X}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} \begin{bmatrix} -\tfrac{1}{2N}\mathrm{X} & \tfrac{1}{2N}\mathrm{X} \\ \tfrac{1}{2N}\mathrm{X} & -\tfrac{1}{2N}\mathrm{X} \end{bmatrix}$$
$$= \begin{bmatrix} \widetilde{\Lambda}_{\mathbf{u}\mathbf{u}'} & \widetilde{\Lambda}_{\mathbf{u}\mathbf{v}'} \\ \widetilde{\Lambda}_{\mathbf{v}\mathbf{u}'} & \widetilde{\Lambda}_{\mathbf{v}\mathbf{v}'} \end{bmatrix} + \tfrac{1}{2N} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} \mathrm{X}^{\mathrm{T}} (\widetilde{\Lambda}_{\mathrm{F}\mathbf{u}'} - \widetilde{\Lambda}_{\mathrm{G}\mathbf{u}'}) & \mathrm{X}^{\mathrm{T}} (\widetilde{\Lambda}_{\mathrm{F}\mathbf{v}'} - \widetilde{\Lambda}_{\mathrm{G}\mathbf{v}'}) \end{bmatrix}$$
$$+ \tfrac{1}{2N} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} (\widetilde{\Lambda}_{\mathbf{u}\mathrm{F}} - \widetilde{\Lambda}_{\mathbf{u}\mathrm{G}})C \\ (\widetilde{\Lambda}_{\mathbf{v}\mathrm{F}} - \widetilde{\Lambda}_{\mathbf{v}\mathrm{G}})C \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix}$$
$$+ \left(\tfrac{1}{2N}\right)^2 \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mathrm{X}^{\mathrm{T}} (\widetilde{\Lambda}_{\mathrm{FF}} - \widetilde{\Lambda}_{\mathrm{FG}} - \widetilde{\Lambda}_{\mathrm{GF}} + \widetilde{\Lambda}_{\mathrm{GG}})\mathrm{X} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

Example 4.3: Step 3 of the above procedure is applied to Example 4.1. The resulting prediction and confidence intervals for Φ(u) are depicted in Figure 29 and for Γ(v) in Figure 30. Due to the removal of the arbitrary additive constant, the confidence intervals in Figure 29 and Figure 30 are much narrower than those of Figure 21 and Figure 22. In fact, they are narrower than the confidence intervals in Figure 20, since in the latter, the intervals encompass the combined uncertainty in Φ(u) and Γ(v). The predictions remain unchanged other than a uniform vertical adjustment corresponding to fixing the arbitrary constant due to meeting the condition $\mathrm{X}^{\mathrm{T}}\mathbf{F} = \mathrm{X}^{\mathrm{T}}\mathbf{G}$.

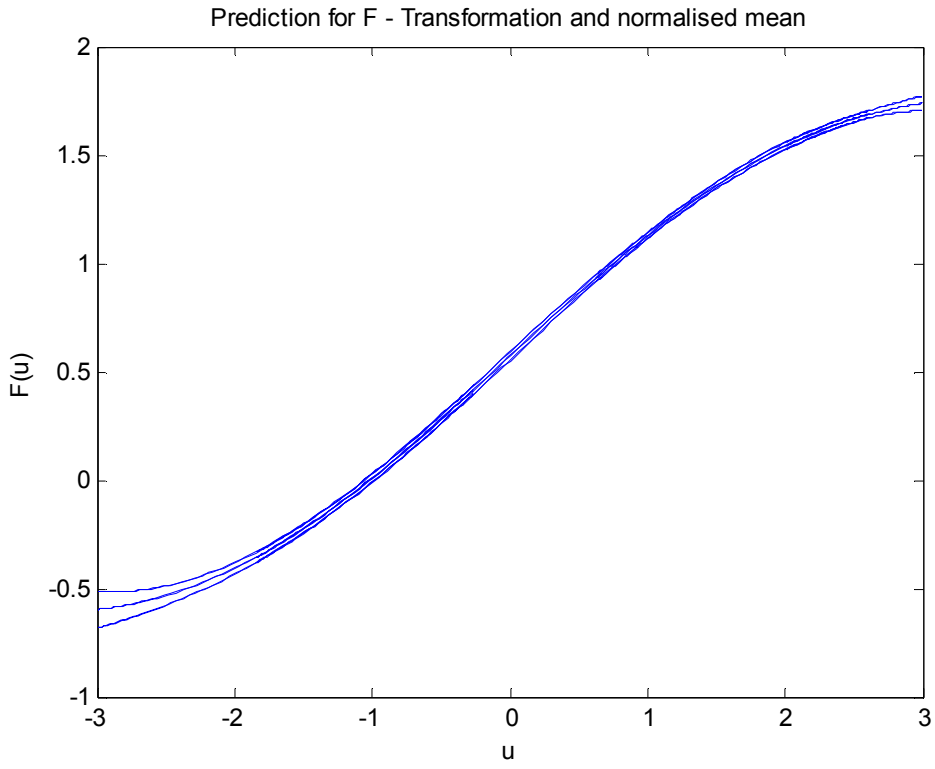


Figure 29 Prediction and confidence intervals of Φ after normalisation.

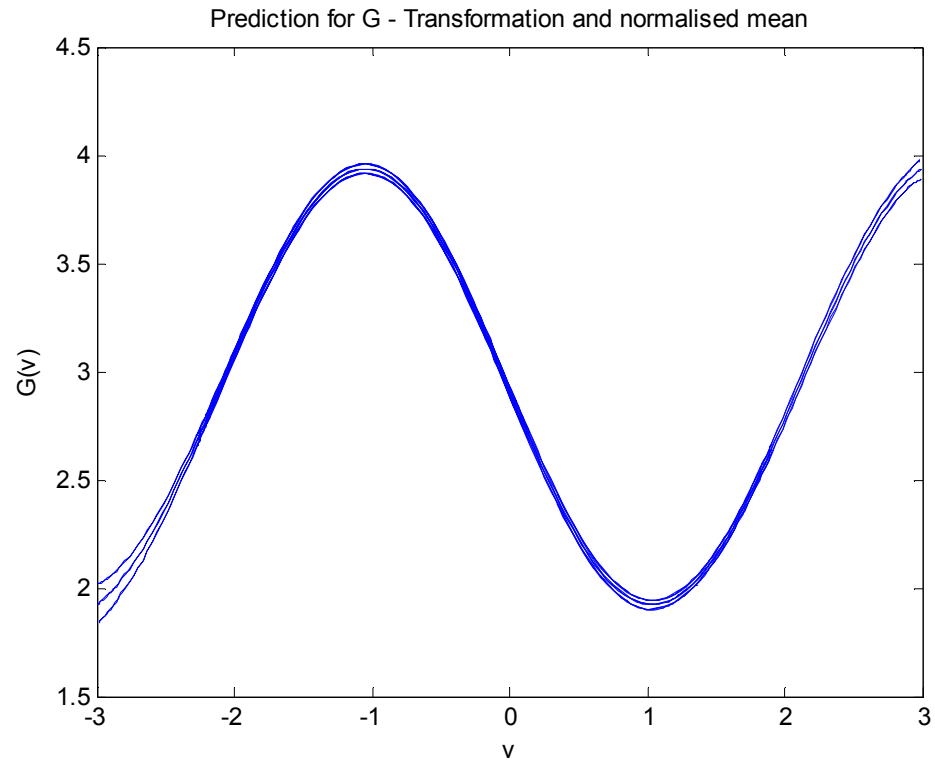


Figure 30 Prediction and confidence intervals of Γ after normalisation.

Example 4.4: Returning to Example 4.2, the prior model for the component, $\Gamma(\mathrm{v})$, is chosen to be the stationary Gaussian process, Γ_{v}, with exponential squared covariance function

$$C_{\Gamma}(\mathrm{v},\mathrm{v}') = a_{\Gamma}\exp\left\{-\frac{1}{2}d_{\Gamma}(\mathrm{v}-\mathrm{v}')^2\right\} \tag{63}$$

Applying steps 1, 2 and 3 of the above procedure, the hyperparameter values are $a_{\Phi} = 2.66$, $a_{\Gamma} = 1.02$, $d_{\Phi_1} = 1.1\times10^{-3}$, $d_{\Phi_2} = 4.5\times10^{-5}$, $d_{\Gamma} = 9.9\times10^{-3}$ and $b = 0.0978$. The predictions and confidence intervals for $\Phi(\mathbf{u})$ and $\Gamma(\mathrm{v})$ are depicted in Figure 31 and Figure 32, respectively.

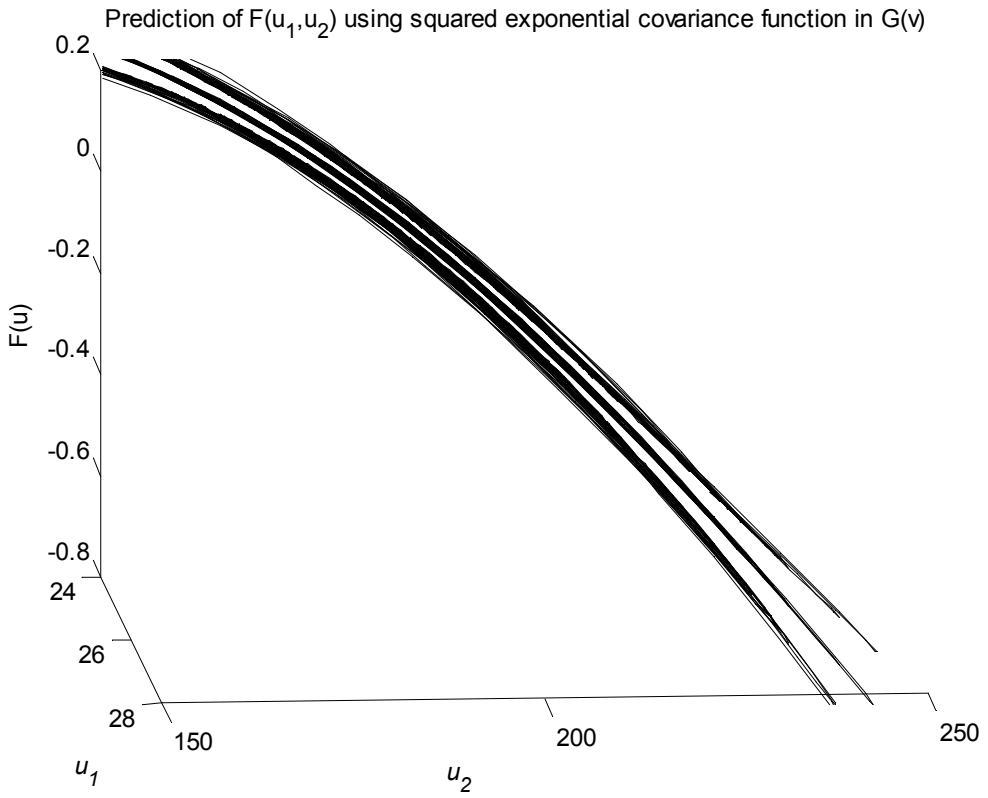


Figure 31 Prediction and confidence intervals of extracted Φ component with Γ_v using squared exponential covariance function.

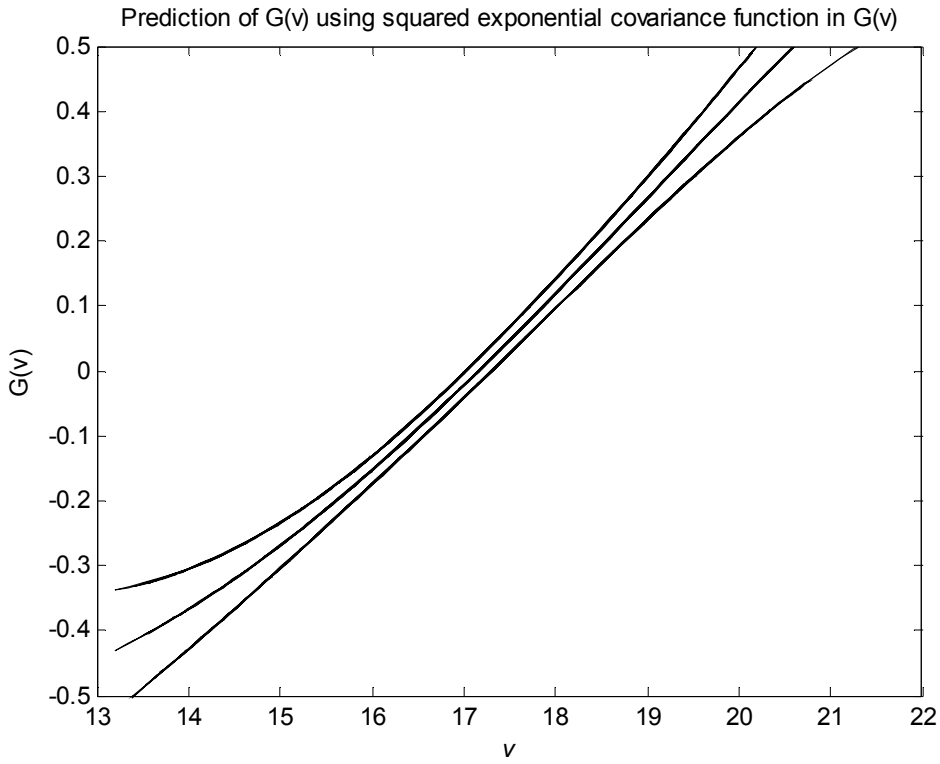


Figure 32 Prediction and confidence intervals of extracted Γ component with Γ_v using squared exponential covariance function.

Comparing Figure 31 to Figure 24 and Figure 27, and Figure 32 to and Figure 25 and Figure 28, the confidence intervals are commensurate with those obtained with the non-stationary Gaussian process models considered in Example 4.2. The freedom to add or subtract an arbitrary constant to the components is removed by Step 3 and, hence, the confidence intervals are not overly wide.

When the nonlinear relationship underlying measured data has two additive components with different explanatory variables, Steps 1, 2 and 3 provide an improved procedure for extracting the two components. The confidence intervals are no longer excessive and the predictions for the two components are no longer too uncertain to be of any real value. Any arbitrary additive constant is removed by condition, $X^T\mathbf{F} = X^T\mathbf{G}$, see Remark 4.2.

4.4 Multiple Gaussian Processes Model with Same Explanatory Variable

In this sub-section, the case with the underlying nonlinear relationship having two components with the same variable, i.e. $\mathrm{h}(\mathbf{z},\mathbf{w})|_{\mathbf{w}=\mathbf{z}} = \mathrm{h}(\mathbf{z}) = \mathrm{f}(\mathbf{z}) + \mathrm{g}(\mathbf{z})$, is considered. The requirement remains to extract either or both of f(**z**) and g(**z**). The difficulty with excessively wide confidence intervals persists.

Example 4.5: The underlying nonlinear relationship has two components each dependent on the scalar explanatory variable, z, i.e. h(z) = f(z) + g(z). The Gaussian process models, f_z and g_z, for the classes of all possible components both have an exponential squared covariance function of the form $a\exp\left\{-\frac{d}{2}(z_i - z_j)^2\right\}$. The hyperparameters for f_z are a_f = 1.8 and d_f = 2.5, and the hyperparameters for g_z are a_g = 0.95 and d_g = 120. The measurement noise is assumed to be Gaussian white noise with variance, b = 0.04. A data set, $\{\mathrm{y}_i, z_i\}_{i=1}^{800}$, is obtained for the above two component nonlinear relationship where $\mathrm{y}_i = \mathrm{f}(z_i) + \mathrm{g}(z_i) + \mathrm{n}_i$, $\{\mathrm{f}(z_i)\}_{i=1}^{800}$ and $\{\mathrm{g}(z_i)\}_{i=1}^{800}$ are realisations for the stochastic processes, f_z and g_z, respectively, sampled at 100Hz and $\{\mathrm{n}_i\}_{i=1}^{800}$ are the additive noise values. The correlation between two measurements is

$$\mathrm{E}[\mathrm{y}_i, \mathrm{y}_j] = a_\mathrm{f}\exp\left(-\frac{d_\mathrm{f}}{2}(z_i - z_i)^2\right) + a_\mathrm{g}\exp\left(-\frac{d_\mathrm{g}}{2}(z_i - z_i)^2\right) + b\delta_{ij}$$

where δ_{ij} is the Kronecker delta.

Using the known hyperparameter values, the procedure in §4.2 based on two independent Gaussian processes is applied to the data set. The data values and prediction for h(z) together with the prediction error and confidence intervals are shown in Figure 33.

The predictions for the two components, f(z) and g(z), together with their confidence intervals are shown in Figure 34. Comparing the confidence intervals in Figure 34 to the confidence intervals in Figure 33, the former are clearly very much broader than

the latter. As in Example 4.1, the confidence intervals for the two components are excessive and the predictions too uncertain to be of any real value. This difficulty is not specific to Example 4.5 but is generic to the case with the components having the same explanatory variable.

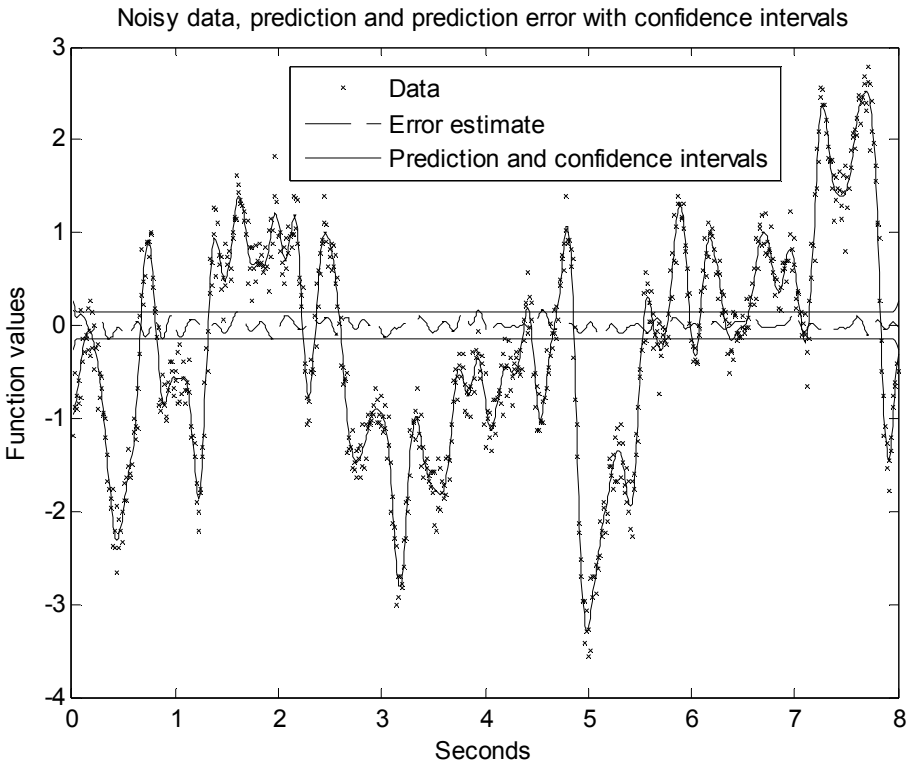


Figure 33 Two lengthscale data (xx), prediction (--), error and confidence interval (==).

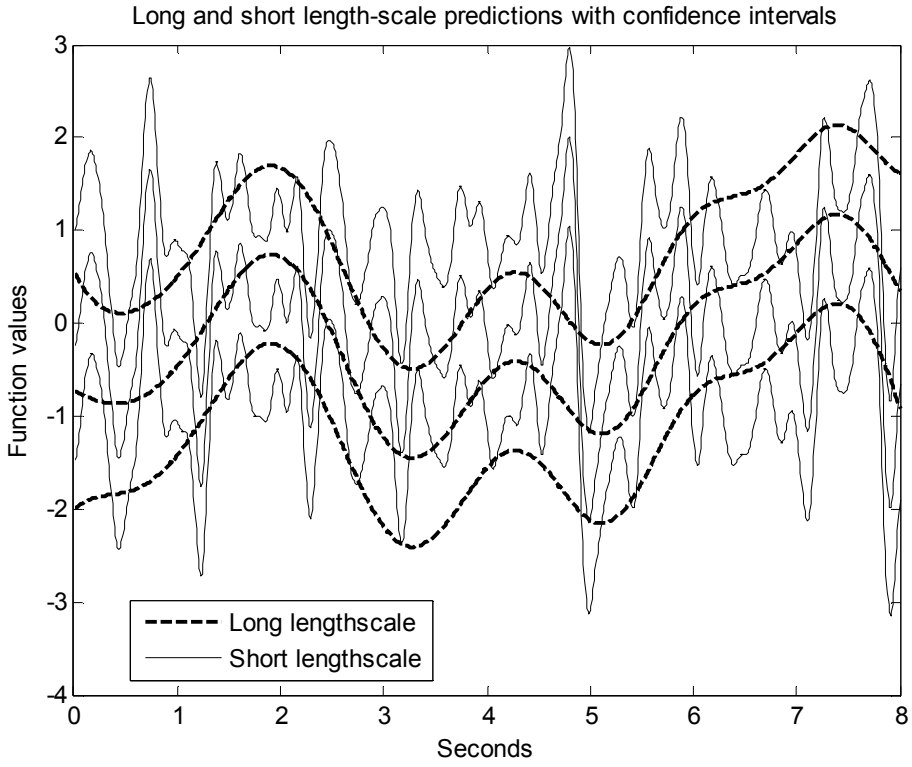


Figure 34 Prediction and confidence intervals of the posterior joint probability distribution.

4.4.1 Improved Two GP Model with Same Explanatory Variables

When the underlying nonlinear relationship has two components with the same explanatory variable, i.e. $h(\mathbf{z}) = f(\mathbf{z}) + g(\mathbf{z})$, the method for extracting the two components, suggested in §4.2, fails as it did in the case with the components having different explanatory variables, see §4.2 and §4.3. The confidence intervals for the individual components, f(**z**) and g(**z**), are much wider than the confidence interval for the total nonlinear relationship, h(**z**). The predictions for the individual components are, thus, very weak and of little utility. The reason for the wide confidence intervals remains the same. There is uncertainty of attribution between the two components; that is, an arbitrary term can be added to f(**z**) and subtracted from g(**z**) without changing h(**z**). Unlike the case with the two components having different explanatory variables, the arbitrary term is no longer a constant but depends on the explanatory variable, **z**. When that ambiguity is removed in the case with different explanatory variables, the confidence intervals of the individual components become much narrower. A similar approach is required to remove the ambiguity and so tighten the

confidence intervals for the individual components, when the explanatory variables are the same. An appropriate method is described below.

As discussed in §4.3.2 but with **w**=**z**, the Gaussian process pair model for the class of function pairs (f(**z**),g(**z**)), is not uniquely determined by the data only the single Gaussian process model for the class of functions, h(**z**). There remains a substantial freedom of choice. A particular manifestation of this freedom of choice is explored in §4.3.2, specifically, the freedom to transform the independent Gaussian process pair, $(\mathrm{f}_{\mathbf{z}},\mathrm{g}_{\mathbf{z}})$, defining the prior model or equivalently to transform the Gaussian process pair, $(\tilde{\mathrm{f}}_{\mathbf{z}},\tilde{\mathrm{g}}_{\mathbf{z}})$, defining the posterior model. Similarly to §4.3, the transformation is exploited to remove the uncertainty of attribution between the two components.

The context within which the two Gaussian processes model is being applied, must inform the choice of the transformation. From the applications, from which motivation for the development of the two Gaussian processes model arises, the context has the following attributes.

a. The component, f(**z**), is the major component; that is, the part of the data explained by it must be as large as possible.
b. The components, f(**z**) and g(**z**), represent separate unrelated aspects of the underlying nonlinear relationship.

Given the specific set of values of the explanatory variables, $\{\mathbf{z}_i\}_{i=1}^{N}$, corresponding to the measurement data set, the transformation (64) is chosen such that

$$T=[0 \;\vdots\; \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}Q_{\mathrm{F}}^{-1}] \tag{64}$$

where $Q_{\mathrm{F}}=\Lambda_{\mathrm{FF}}+\mathbf{B}$, $\Lambda_{\mathrm{Ff}_{\mathbf{z}}}^{\mathrm{T}}=\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}=\mathrm{E}[\mathrm{f}_{\mathbf{z}}\mathbf{F}^{\mathrm{T}}]$ and $\Lambda_{\mathrm{FF}}=\mathrm{E}[\mathbf{F}\mathbf{F}^{\mathrm{T}}]$. Hence, following §4.3.2 but with **w**=**z**, the prior model becomes the transformed Gaussian process pair

$$\begin{bmatrix}\mathrm{f}'_{\mathbf{z}}\\ \mathrm{g}'_{\mathbf{z}}\end{bmatrix}=\begin{bmatrix}\mathrm{f}_{\mathbf{z}}\\ \mathrm{g}_{\mathbf{z}}\end{bmatrix}+\begin{bmatrix}1\\ -1\end{bmatrix}[0 \;\vdots\; \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}Q_{\mathrm{F}}^{-1}]\begin{bmatrix}\mathbf{F}\\ \mathbf{G}\end{bmatrix}=\left[\begin{array}{cc:c}1 & 0 & \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}Q_{\mathrm{F}}^{-1}\\ 0 & 1 & -\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}Q_{\mathrm{F}}^{-1}\end{array}\right]\left[\begin{array}{c}\mathrm{f}_{\mathbf{z}}\\ \mathrm{g}_{\mathbf{z}}\\ \hdashline \mathbf{G}\end{array}\right]=\begin{bmatrix}\mathrm{f}_{\mathbf{z}}+\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}Q_{\mathrm{F}}^{-1}\mathbf{G}\\ \mathrm{g}_{\mathbf{z}}-\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}}Q_{\mathrm{F}}^{-1}\mathbf{G}\end{bmatrix}$$

and, on conditioning on the data, the posterior model becomes the transformed Gaussian process pair

$$\begin{bmatrix} \widetilde{\mathrm{f}}'_{\mathbf{z}} \\ \widetilde{\mathrm{g}}'_{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathrm{f}}_{\mathbf{z}} \\ \widetilde{\mathrm{g}}_{\mathbf{z}} \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} [0 \quad \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1}] \begin{bmatrix} \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathrm{f}}_{\mathbf{z}} + \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \widetilde{\mathbf{G}} \\ \widetilde{\mathrm{g}}_{\mathbf{z}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \widetilde{\mathbf{G}} \end{bmatrix}$$

Furthermore,

$$\begin{bmatrix} \mathbf{F}' \\ \mathbf{G}' \end{bmatrix} = \mathrm{T} \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} = \begin{bmatrix} \mathbf{F} + \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \mathbf{G} \\ (I - \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1}) \mathbf{G} \end{bmatrix} \text{ and } \begin{bmatrix} \widetilde{\mathbf{F}}' \\ \widetilde{\mathbf{G}}' \end{bmatrix} = \mathrm{T} \begin{bmatrix} \widetilde{\mathbf{F}} \\ \widetilde{\mathbf{G}} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{F}} + \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \widetilde{\mathbf{G}} \\ (I - \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1}) \widetilde{\mathbf{G}} \end{bmatrix} \tag{65}$$

$$\text{with } \mathrm{T} = \begin{bmatrix} I & \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \\ 0 & I - \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \end{bmatrix}.$$

The joint mean and covariance function for the posterior Gaussian process pair model, $(\widetilde{\mathrm{f}}'_{\mathbf{z}}, \widetilde{\mathrm{g}}'_{\mathbf{z}})$, is provided by **Theorem 4.1**.

Theorem 4.1: **(a)** With the transformation chosen to be $T = [0 \mid \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1}]$, the joint mean and joint covariance function for the posterior Gaussian process pair model, $(\widetilde{\mathrm{f}}'_{\mathbf{z}}, \widetilde{\mathrm{g}}'_{\mathbf{z}})$, have the following equivalent forms.

(i)

$$\begin{bmatrix} \hat{\mathrm{f}}'_{\mathbf{z}} \\ \hat{\mathrm{g}}'_{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{f}}_{\mathbf{z}} \\ \hat{\mathrm{g}}_{\mathbf{z}} \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} [0 \quad \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1}] \begin{bmatrix} \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{bmatrix}$$

and

$$\begin{aligned} \begin{bmatrix} \widetilde{\Lambda}'_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} & \widetilde{\Lambda}'_{\mathrm{f}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} \\ \widetilde{\Lambda}'_{\mathrm{g}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} & \widetilde{\Lambda}'_{\mathrm{g}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} \end{bmatrix} &= \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} & \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} \\ \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} & \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} \end{bmatrix} \\ &+ \begin{bmatrix} 0 & \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \\ 0 & -\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F}\mathrm{f}_{\mathbf{z}'}} & -Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F}\mathrm{f}_{\mathbf{z}'}} \end{bmatrix} \\ &+ \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} & \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{G}} \\ \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{F}} & \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{G}} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F}\mathrm{f}_{\mathbf{z}'}} & -Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F}\mathrm{f}_{\mathbf{z}'}} \end{bmatrix} + \begin{bmatrix} 0 & \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \\ 0 & -\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{F}\mathrm{f}_{\mathbf{z}'}} & \widetilde{\Lambda}_{\mathrm{F}\mathrm{g}_{\mathbf{z}'}} \\ \widetilde{\Lambda}_{\mathrm{G}\mathrm{f}_{\mathbf{z}'}} & \widetilde{\Lambda}_{\mathrm{G}\mathrm{g}_{\mathbf{z}'}} \end{bmatrix} \end{aligned} \tag{66}$$

where

$$\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} = \Lambda^T_{\mathrm{F}\mathrm{f}_{\mathbf{z}}} = \Lambda_{\mathbf{z}\mathrm{F}},\ \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{G}} = \Lambda^T_{\mathrm{G}\mathrm{f}_{\mathbf{z}}} = \Lambda_{\mathbf{z}\mathrm{G}},\ \Lambda_{\mathrm{g}_{\mathbf{z}}\mathrm{F}} = \Lambda^T_{\mathrm{F}\mathrm{g}_{\mathbf{z}}} = \Lambda_{\mathbf{w}\mathrm{F}}|_{\mathbf{w}=\mathbf{z}},\ \Lambda_{\mathrm{g}_{\mathbf{z}}\mathrm{G}} = \Lambda^T_{\mathrm{G}\mathrm{g}_{\mathbf{z}}} = \Lambda_{\mathbf{w}\mathrm{G}}|_{\mathbf{w}=\mathbf{z}}$$

$$\widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} = \widetilde{\Lambda}_{\mathbf{z}\mathbf{z}'},\ \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} = \widetilde{\Lambda}^T_{\mathrm{g}_{\mathbf{z}'}\mathrm{f}_{\mathbf{z}}} = \widetilde{\Lambda}_{\mathbf{z}\mathbf{w}'}|_{\mathbf{w}'=\mathbf{z}'},\ \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} = \widetilde{\Lambda}^T_{\mathrm{f}_{\mathbf{z}'}\mathrm{g}_{\mathbf{z}}} = \widetilde{\Lambda}_{\mathbf{w}\mathbf{z}'}|_{\mathbf{w}=\mathbf{z}},\ \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} = \widetilde{\Lambda}_{\mathbf{w}\mathbf{w}'}|_{\mathbf{w}=\mathbf{z},\mathbf{w}'=\mathbf{z}'}$$

$$\widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} = \widetilde{\Lambda}^T_{\mathrm{F}\mathrm{f}_{\mathbf{z}}} = \widetilde{\Lambda}_{\mathbf{z}\mathrm{F}},\ \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{G}} = \widetilde{\Lambda}^T_{\mathrm{G}\mathrm{f}_{\mathbf{z}}} = \widetilde{\Lambda}_{\mathbf{z}\mathrm{G}},\ \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{F}} = \widetilde{\Lambda}^T_{\mathrm{F}\mathrm{g}_{\mathbf{z}}} = \widetilde{\Lambda}_{\mathbf{w}\mathrm{F}}|_{\mathbf{w}=\mathbf{z}},\ \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{G}} = \widetilde{\Lambda}^T_{\mathrm{G}\mathrm{g}_{\mathbf{z}}} = \widetilde{\Lambda}_{\mathbf{w}\mathrm{G}}|_{\mathbf{w}=\mathbf{z}}$$

$$\widetilde{\Lambda}'_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} = \widetilde{\Lambda}'_{\mathbf{z}\mathbf{z}'},\ \widetilde{\Lambda}'_{\mathrm{f}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} = \widetilde{\Lambda}'^T_{\mathrm{g}_{\mathbf{z}'}\mathrm{f}_{\mathbf{z}}} = \widetilde{\Lambda}'_{\mathbf{z}\mathbf{w}'}|_{\mathbf{w}'=\mathbf{z}},\ \widetilde{\Lambda}'_{\mathrm{g}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} = \widetilde{\Lambda}'^T_{\mathrm{f}_{\mathbf{z}'}\mathrm{g}_{\mathbf{z}}} = \widetilde{\Lambda}'_{\mathbf{w}\mathbf{z}'}|_{\mathbf{w}=\mathbf{z}},\ \widetilde{\Lambda}'_{\mathrm{g}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} = \widetilde{\Lambda}'_{\mathbf{w}\mathbf{w}'}|_{\mathbf{w}=\mathbf{z},\mathbf{w}'=\mathbf{z}'}$$

(ii)

$$\begin{bmatrix} \hat{\mathrm{f}}_{\mathbf{z}} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \hat{\mathbf{G}} \\ \hat{\mathrm{g}}_{\mathbf{z}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \hat{\mathbf{G}} \end{bmatrix}$$

and

$$\begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} & 0 \\ 0 & \widetilde{\Lambda}_{\mathrm{g_z g_{z'}}} + \widetilde{\Lambda}_{\mathrm{f_z g_{z'}}} + \widetilde{\Lambda}_{\mathrm{g_z f_{z'}}} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \end{bmatrix} \tag{67}$$

(iii)

$$\begin{bmatrix} \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \\ (\Lambda_{\mathrm{f_z F}} + \Lambda_{\mathrm{g_z G}}) Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \end{bmatrix}$$

and

$$\begin{bmatrix} (\Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}}) & 0 \\ 0 & \begin{matrix} [(\Lambda_{\mathrm{f_z f_{z'}}} + \Lambda_{\mathrm{g_z g_{z'}}}) - (\Lambda_{\mathrm{f_z F}} + \Lambda_{\mathrm{g_z G}}) Q^{-1} (\Lambda_{\mathrm{F f_{z'}}} + \Lambda_{\mathrm{G g_{z'}}})] \\ - (\Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}}) \end{matrix} \end{bmatrix} \tag{68}$$

(b) The posterior model pair of Gaussian process is

$$\begin{bmatrix} \widetilde{\mathrm{f}}'_{\mathbf{z}} \\ \widetilde{\mathrm{g}}'_{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \widehat{\mathrm{f}}'_{\mathbf{z}} \\ \widetilde{\mathrm{h}}_{\mathbf{z}} - \widehat{\mathrm{f}}'_{\mathbf{z}} \end{bmatrix}$$

where $\widehat{\mathrm{f}}'_{\mathbf{z}} = \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y}$, $\widetilde{\mathrm{h}}_{\mathbf{z}} = \Lambda_{\mathrm{h_z H}} Q^{-1} \mathbf{Y}$.

Proof 4.1: **(a)** Several identities are required, namely,

$$\widetilde{\Lambda}_{\mathrm{G g_{z'}}} = \Lambda_{\mathrm{G g_{z'}}} - \Lambda_{\mathrm{GG}} Q^{-1} \Lambda_{\mathrm{G g_{z'}}} = (Q - \Lambda_{\mathrm{GG}}) Q^{-1} \Lambda_{\mathrm{G g_{z'}}} = Q_{\mathrm{F}} Q^{-1} \Lambda_{\mathrm{G g_{z'}}}$$

$$\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{G g_{z'}}} = \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{G g_{z'}}} = -\widetilde{\Lambda}_{\mathrm{f_z g_{z'}}}$$

$$\widetilde{\Lambda}_{\mathrm{g_z G}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} = \Lambda_{\mathrm{g_z G}} Q^{-1} \Lambda_{\mathrm{F f_{z'}}} = -\widetilde{\Lambda}_{\mathrm{g_z f_{z'}}}$$

$$\widetilde{\Lambda}_{\mathrm{GG}} = \Lambda_{\mathrm{GG}} - \Lambda_{\mathrm{GG}} Q^{-1} \Lambda_{\mathrm{GG}} = (Q - \Lambda_{\mathrm{GG}}) Q^{-1} \Lambda_{\mathrm{GG}} = Q_{\mathrm{F}} Q^{-1} \Lambda_{\mathrm{GG}}$$

$$\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} = \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} = -\widetilde{\Lambda}_{\mathrm{f_z G}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} = -\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{G f_{z'}}}$$

(i) It follows immediately from §4.3.2 with the transformation defined by $T = [0 \mid \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1}]$.

(ii) The joint mean follows immediately from **(a)**. The joint covariance is

$$\begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} & \widetilde{\Lambda}_{\mathrm{f_z g_{z'}}} \\ \widetilde{\Lambda}_{\mathrm{g_z f_{z'}}} & \widetilde{\Lambda}_{\mathrm{g_z g_{z'}}} \end{bmatrix} + \begin{bmatrix} 0 & \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \\ 0 & -\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{F f_{z'}}} & \widetilde{\Lambda}_{\mathrm{F g_{z'}}} \\ \widetilde{\Lambda}_{\mathrm{G f_{z'}}} & \widetilde{\Lambda}_{\mathrm{G g_{z'}}} \end{bmatrix} + \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f_z F}} & \widetilde{\Lambda}_{\mathrm{f_z G}} \\ \widetilde{\Lambda}_{\mathrm{g_z F}} & \widetilde{\Lambda}_{\mathrm{g_z G}} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} & -Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \\ 0 & -\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} & -Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \end{bmatrix}$$
$$= \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} & \widetilde{\Lambda}_{\mathrm{f_z g_{z'}}} \\ \widetilde{\Lambda}_{\mathrm{g_z f_{z'}}} & \widetilde{\Lambda}_{\mathrm{g_z g_{z'}}} \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} -\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} & -\widetilde{\Lambda}_{\mathrm{f_z g_{z'}}} \end{bmatrix}$$
$$+ \begin{bmatrix} -\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \\ -\widetilde{\Lambda}_{\mathrm{g_z f_{z'}}} \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \begin{bmatrix} 1 & -1 \end{bmatrix}$$
$$= \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} & 0 \\ 0 & \widetilde{\Lambda}_{\mathrm{g_z g_{z'}}} + \widetilde{\Lambda}_{\mathrm{f_z g_{z'}}} + \widetilde{\Lambda}_{\mathrm{g_z f_{z'}}} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \end{bmatrix}$$

(iii) The joint mean is

$$\begin{bmatrix} \hat{\mathrm{f}}_{\mathbf{z}} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \hat{\mathbf{G}} \\ \hat{\mathrm{g}}_{\mathbf{z}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \hat{\mathbf{G}} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathrm{f_z F}} Q^{-1} \mathbf{Y} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{GG}} Q^{-1} \mathbf{Y} \\ \Lambda_{\mathrm{g_z G}} Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{GG}} Q^{-1} \mathbf{Y} \end{bmatrix}$$
$$= \begin{bmatrix} \Lambda_{\mathrm{f_z F}} Q^{-1} \mathbf{Y} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} (Q - Q_{\mathrm{F}}) Q^{-1} \mathbf{Y} \\ \Lambda_{\mathrm{g_z G}} Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} (Q - Q_{\mathrm{F}}) Q^{-1} \mathbf{Y} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathrm{f_z F}} Q^{-1} \mathbf{Y} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y} - \Lambda_{\mathrm{f_z F}} Q^{-1} \mathbf{Y} \\ \Lambda_{\mathrm{g_z G}} Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y} + \Lambda_{\mathrm{f_z F}} Q^{-1} \mathbf{Y} \end{bmatrix}$$
$$= \begin{bmatrix} \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \\ (\Lambda_{\mathrm{f_z F}} + \Lambda_{\mathrm{g_z G}}) Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \end{bmatrix}$$

and the joint covariance is

$$\begin{bmatrix} \Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} & 0 \\ 0 & [(\Lambda_{\mathrm{f_z f_{z'}}} + \Lambda_{\mathrm{g_z g_{z'}}}) - (\Lambda_{\mathrm{f_z F}} + \Lambda_{\mathrm{g_z G}}) Q^{-1} (\Lambda_{\mathrm{F f_{z'}}} + \Lambda_{\mathrm{G g_{z'}}})] \\ & - (\Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}}) \end{bmatrix}$$

since

$$\begin{aligned} \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} &= \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \\ &= \widetilde{\Lambda}_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q^{-1} (Q - Q_{\mathrm{F}}) Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \\ &= (\Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{F f_{z'}}}) - (\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{F f_{z'}}}) \\ &= \Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \end{aligned}$$

and

$$\begin{aligned} &\widetilde{\Lambda}_{\mathrm{g_z g_{z'}}} + \widetilde{\Lambda}_{\mathrm{f_z g_{z'}}} + \widetilde{\Lambda}_{\mathrm{g_z f_{z'}}} + \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} \\ &= (\Lambda_{\mathrm{g_z g_{z'}}} - \Lambda_{\mathrm{g_z G}} Q^{-1} \Lambda_{\mathrm{G g_{z'}}}) - \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{G g_{z'}}} - \Lambda_{\mathrm{g_z G}} Q^{-1} \Lambda_{\mathrm{F f_{z'}}} + (\Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q^{-1} \Lambda_{\mathrm{F f_{z'}}}) \\ &= [(\Lambda_{\mathrm{f_z f_{z'}}} + \Lambda_{\mathrm{g_z g_{z'}}}) - (\Lambda_{\mathrm{f_z F}} + \Lambda_{\mathrm{g_z G}}) Q^{-1} (\Lambda_{\mathrm{F f_{z'}}} + \Lambda_{\mathrm{G g_{z'}}})] - (\Lambda_{\mathrm{f_z f_{z'}}} - \Lambda_{\mathrm{f_z F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{F f_{z'}}}) \end{aligned}$$

(b) The cross-covariance function between $\widetilde{\mathrm{f}}'_{\mathbf{z}}$ and $\widetilde{\mathrm{h}}_{\mathbf{z}'}$ equals the covariance function for $\widetilde{\mathrm{f}}'_{\mathbf{z}}$ since

$$\begin{aligned}
&\mathrm{E}\left[[\widetilde{\mathrm{f}}'_{\mathbf{z}} - \mathrm{f}'_{\mathbf{z}}][\widetilde{\mathrm{h}}_{\mathbf{z}'} - \mathrm{h}_{\mathbf{z}'}]\right] = \mathrm{E}\left[[\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1}(\mathbf{Y}-\mathbf{G}) - \mathrm{f}_{\mathbf{z}}][\mathbf{Y}^{\mathrm{T}} Q^{-1} \Lambda_{\mathrm{Hh}_{\mathbf{z}'}} - \mathrm{h}_{\mathbf{z}'}]\right] \\
&= \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \mathrm{E}[(\mathbf{Y}-\mathbf{G})\mathbf{Y}^{\mathrm{T}}] Q^{-1} \Lambda_{\mathrm{Hh}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \mathrm{E}[(\mathbf{Y}-\mathbf{G})\mathrm{h}_{\mathbf{z}'}] - \mathrm{E}[\mathrm{f}_{\mathbf{z}}\mathbf{Y}^{\mathrm{T}}] Q^{-1} \Lambda_{\mathrm{Hh}_{\mathbf{z}'}} + \mathrm{E}[\mathrm{f}_{\mathbf{z}}\mathrm{h}_{\mathbf{z}'}] \\
&= \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} Q_{\mathrm{F}} Q^{-1} \Lambda_{\mathrm{Hh}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{Ff}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q^{-1} \Lambda_{\mathrm{Hh}_{\mathbf{z}'}} + \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} \\
&= \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{Ff}_{\mathbf{z}'}} = \mathrm{E}\left[[\widetilde{\mathrm{f}}'_{\mathbf{z}} - \mathrm{f}'_{\mathbf{z}}][\widetilde{\mathrm{f}}'_{\mathbf{z}} - \mathrm{f}']\right]
\end{aligned}$$

It follows immediately from (a) (iii) that $\widetilde{\mathrm{g}}'_{\mathbf{z}} = \widetilde{\mathrm{h}}_{\mathbf{z}} - \widehat{\mathrm{f}}'_{\mathbf{z}}$ as required.

From Theorem 4.1, the prediction for $\mathrm{f}'_{\mathbf{z}}$ is

$$\hat{\mathrm{f}}'_{\mathbf{z}} = \hat{\mathrm{f}}_{\mathbf{z}} + \Lambda_{\mathbf{z}\mathrm{F}} Q_{\mathrm{F}}^{-1} \hat{\mathbf{G}} = \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \mathbf{Y}$$

with variance

$$\widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{Ff}_{\mathbf{z}'}} = \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{Ff}_{\mathbf{z}'}}$$

and the prediction for $\mathrm{g}'_{\mathbf{z}}$ is

$$\hat{\mathrm{g}}'_{\mathbf{z}} = \hat{\mathrm{g}}_{\mathbf{z}} - \Lambda_{\mathbf{z}\mathrm{F}} Q_{\mathrm{F}}^{-1} \hat{\mathbf{G}} = (\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} + \Lambda_{\mathrm{g}_{\mathbf{z}}\mathrm{G}}) Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \mathbf{Y}$$

with variance

$$\begin{aligned}
&\widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} + \widetilde{\Lambda}_{\mathrm{f}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}} + \widetilde{\Lambda}_{\mathrm{g}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} + \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \widetilde{\Lambda}_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{Ff}_{\mathbf{z}'}} \\
&= [(\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} + \Lambda_{\mathrm{g}_{\mathbf{z}}\mathrm{g}_{\mathbf{z}'}}) - (\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} + \Lambda_{\mathrm{g}_{\mathbf{z}}\mathrm{G}}) Q^{-1} (\Lambda_{\mathrm{Ff}_{\mathbf{z}'}} + \Lambda_{\mathrm{Gg}_{\mathbf{z}'}})] - (\Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{f}_{\mathbf{z}'}} - \Lambda_{\mathrm{f}_{\mathbf{z}}\mathrm{F}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{Ff}_{\mathbf{z}'}})
\end{aligned}$$

Furthermore, the prediction and covariance matrix for $[\mathbf{F}'^{\mathrm{T}}, \mathbf{G}'^{\mathrm{T}}]^{\mathrm{T}}$ are

$$\begin{bmatrix} \hat{\mathbf{F}}' \\ \hat{\mathbf{G}}' \end{bmatrix} = \mathrm{T} \begin{bmatrix} \hat{\mathbf{F}} \\ \hat{\mathbf{G}} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \\ (\Lambda_{\mathrm{FF}} + \Lambda_{\mathrm{GG}}) Q^{-1} \mathbf{Y} - \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \end{bmatrix}$$

and

$$\begin{aligned}
&\mathrm{T} \begin{bmatrix} \widetilde{\Lambda}_{\mathrm{FF}} & \widetilde{\Lambda}_{\mathrm{FG}} \\ \widetilde{\Lambda}_{\mathrm{GF}} & \widetilde{\Lambda}_{\mathrm{GG}} \end{bmatrix} \mathrm{T}^{T} \\
&= \begin{bmatrix} \Lambda_{\mathrm{FF}} - \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \Lambda_{\mathrm{FF}} & 0 \\ 0 & [(\Lambda_{\mathrm{FF}} + \Lambda_{\mathrm{GG}}) - (\Lambda_{\mathrm{FF}} + \Lambda_{\mathrm{GG}}) Q^{-1} (\Lambda_{\mathrm{FF}} + \Lambda_{\mathrm{GG}})] \\ & -(\Lambda_{\mathrm{FF}} - \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1}) \Lambda_{\mathrm{FF}} \end{bmatrix}
\end{aligned} \tag{69}$$

respectively. Alternative expressions for this prediction and covariance matrix are

$$\begin{bmatrix} \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \\ \mathbf{B} Q^{-1} \Lambda_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \mathbf{Y} \end{bmatrix} \text{ and } \begin{bmatrix} \Lambda_{\mathrm{FF}} Q_{\mathrm{F}}^{-1} \mathbf{B} & 0 \\ 0 & \mathbf{B} Q^{-1} \Lambda_{\mathrm{GG}} Q_{\mathrm{F}}^{-1} \mathbf{B} \end{bmatrix} \tag{70}$$

since

$$(\Lambda_{FF}+\Lambda_{GG})Q^{-1}\mathbf{Y}-\Lambda_{FF}Q_F^{-1}\mathbf{Y}=\mathbf{B}Q_F^{-1}\mathbf{Y}-\mathbf{B}Q^{-1}\mathbf{Y}=\mathbf{B}Q^{-1}\Lambda_{GG}Q_F^{-1}\mathbf{Y}$$

$$\Lambda_{FF}-\Lambda_{FF}Q_F^{-1}\Lambda_{FF}=\Lambda_{FF}Q_F^{-1}\mathbf{B}$$

$$\begin{aligned}&[(\Lambda_{FF}+\Lambda_{GG})-(\Lambda_{FF}+\Lambda_{GG})Q^{-1}(\Lambda_{FF}+\Lambda_{GG})]-(\Lambda_{FF}-\Lambda_{FF}Q_F^{-1})\Lambda_{FF}\\&=(\Lambda_{FF}+\Lambda_{GG})Q^{-1}\mathbf{B}-\Lambda_{FF}Q_F^{-1}\mathbf{B}=\mathbf{B}Q_F^{-1}\mathbf{B}-\mathbf{B}Q^{-1}\mathbf{B}=\mathbf{B}Q^{-1}\Lambda_{GG}Q_F^{-1}\mathbf{B}\end{aligned}$$

Remark 4.3: By Theorem 4.1 (a) (ii), the Gaussian processes, $\tilde{f}_z'$ and $\tilde{g}_z'$, are independent. In this manner, context attribute (b) is met.

Since the Gaussian processes, $\tilde{f}_z'$ and $\tilde{g}_z'$, are independent, the sum of the variances for $\tilde{f}_z'$ and $\tilde{g}_z'$ equals the variance of $\tilde{h}_z$ and so the confidence intervals on the predictions are minimal. Adding an arbitrary term to f(**z**) whilst subtracting it from g(**z**) is equivalent to changing Gaussian processes, f_z and g_z, and the Gaussian processes, $\tilde{f}_z'$ and $\tilde{g}_z'$, in such a way that they are no longer independent. Hence, choosing the transformation to make $\tilde{f}_z'$ and $\tilde{g}_z'$ independent removes the ambiguity of attribution between the two components as required.

From Theorem 4.1 (a) (iii), the following interpretation of the transformation is possible. The term, $\Lambda_{FF}Q_F^{-1}\hat{\mathbf{G}}$, can be interpreted as an estimate of the contribution to $\hat{\mathbf{G}}$ explainable by the Gaussian process, f_z. A similar interpretation of $\Lambda_{FF}Q_F^{-1}\tilde{\mathbf{G}}$ can be made. Hence in (65) and (69), the transformation can be interpreted as removing from $\tilde{\mathbf{G}}$ and $\hat{\mathbf{G}}$ the part that can be explained in terms of the Gaussian process, f_z, and adding it to $\tilde{\mathbf{F}}$ and $\hat{\mathbf{F}}$, respectively. Furthermore from Theorem 4.1 (a) (ii), the posterior Gaussian process model for the class of possible f(**z**), namely $\tilde{f}_z' = \hat{f}_z'$, is the prior model conditioned on Y as if the data set is simply a realisation of f_z plus the noise only. The posterior Gaussian process model for the class of possible g(**z**) is $\tilde{g}_z' = \tilde{h}_z - \tilde{f}_z'$, the difference between the Gaussian process model for the class of possible h(**z**) and the Gaussian process model for the class of possible f(**z**). In this manner, the part of the data explained by f_z is made as large as possible and context attribute (a) is met.

The transformation, $T = [0 \mid \Lambda_{\mathbf{f}_z \mathrm{F}} Q_{\mathrm{F}}^{-1}]$, as required removes the ambiguity of attribution between the two components in a manner consistent with the context attributes.

The fact that the transformation applies equally to the posterior as to the prior, see Remark 4.1, is exploited here to enable the posterior model to meet the context attributes. It follows that a suitable modified procedure for extracting the two components when they have the same explanatory variables is the following.

1. Determine the predictions and confidence intervals by the standard approach of §4.1 with a pair of independent Gaussian processes.
2. Modify the predictions and confidence intervals using the transformation, $T_{\mathbf{z}} = [\Lambda_{\mathbf{z}\mathrm{F}} Q_{\mathrm{F}}^{-1} \mid 0]$.

The selection of the hyperparameter values is discussed in §4.4.4.

Example 4.6: Step 2 of the above procedure is applied to Example 4.5. The resulting predictions and confidence intervals for f(**z**) and g(**z**) are depicted in Figure 35. Due to the removal of the arbitrary term, the confidence intervals in Figure 35 are much narrower than those of Figure 34. Because of the independence of the posterior Gaussian process pair, they are narrower than the confidence intervals in Figure 33.

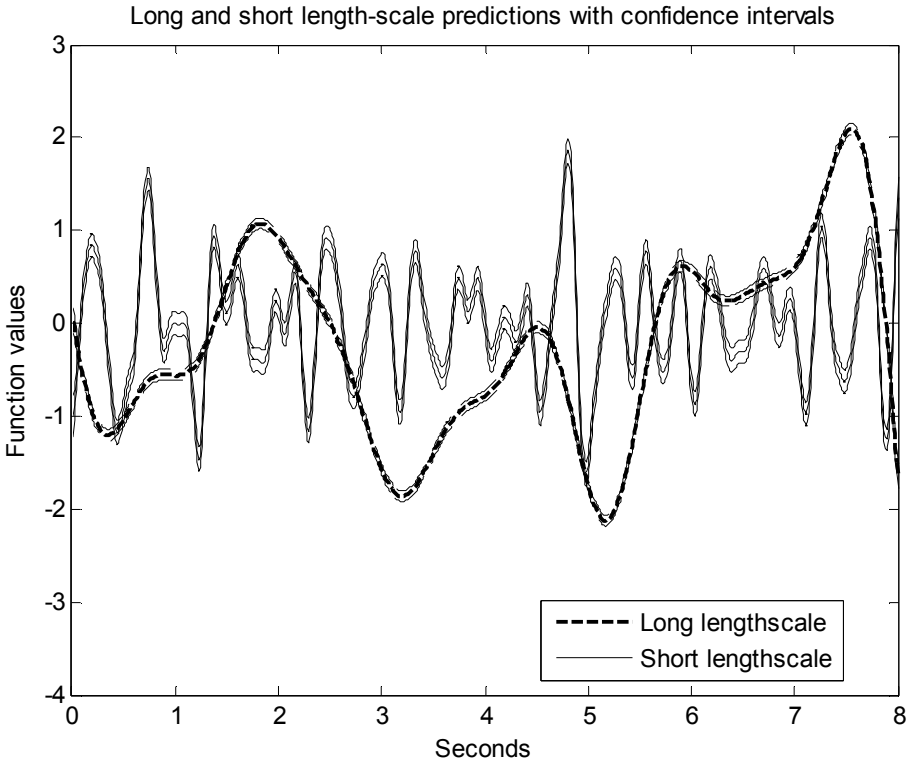


Figure 35 Predictions and confidence intervals of F(**z**) and G(**z**) after transformation.

Confirmation that the transformation acts to remove any contribution to the prediction, $\hat{\mathbf{G}}$, explainable in terms of $f_{\mathbf{z}}$ and adds it to the prediction, $\hat{\mathbf{F}}$, is provided by Example 4.6. Since the explanatory variable is scalar, f(**z**) can be considered the long lengthscale component and g(**z**) the short lengthscale component. Consider the decomposition of **H** and $\hat{\mathbf{H}}$ with respect to the eigenvectors of the covariance matrix for **H**. Since $\hat{\mathbf{H}} = \Lambda_{\mathrm{HH}} Q^{-1}\mathbf{Y}$ with covariance matrix, $\Lambda_{\mathrm{HH}} Q^{-1}\mathbf{B}$, and **B** is diagonal, $v_i^T.\hat{\mathbf{H}} = b^{-1}\lambda_i v_i^T.\mathbf{Y}$ where v_i is the i$^{\text{th}}$ eigenvector of the covariance matrix with eigenvalue, λ_i, and $\mathbf{B} = bI$; that is, the factors with respect to a basis consisting of the eigenvectors of the covariance matrix are scaled by the ratio of the eigenvalue to the noise. The eigenvectors that extract long lengthscale factors are smooth with few zero-crossings thereby averaging out the short lengthscale and more oscillatory factors, whilst the eigenvectors that extract short lengthscale factors are highly oscillatory with many zero-crossings thereby averaging out the long lengthscale and smoother factors. Since the eigenvectors are normalised, a measure of smoothness, i.e. the number of oscillations, and so the lengthscale extracted by an eigenvector, v_i, is

the variation, i.e. $\sum_{j=2}^{N} | v_i(j) - v_i(j-1) |$, where $v_i(j)$ is the j^{th} element of v_i. Given that the prediction extracts the component from **Y** with a particular lengthscale, it might be expected that the longer lengthscale eigenvectors have bigger eigenvalues. A plot with vertical axis the magnitude of the eigenvalue and horizontal axis the variation is shown in Figure 36 for the covariance matrices of $\hat{\mathbf{H}}$. There is an essentially monotonic relationship between the magnitude of the eigenvalues and their variation with the larger eigenvalues having the longer lengthscale. Hence, ordering the eigenvectors for the covariance matrices of $\hat{\mathbf{H}}$ by the magnitude of the eigenvalues, orders them by lengthscale.

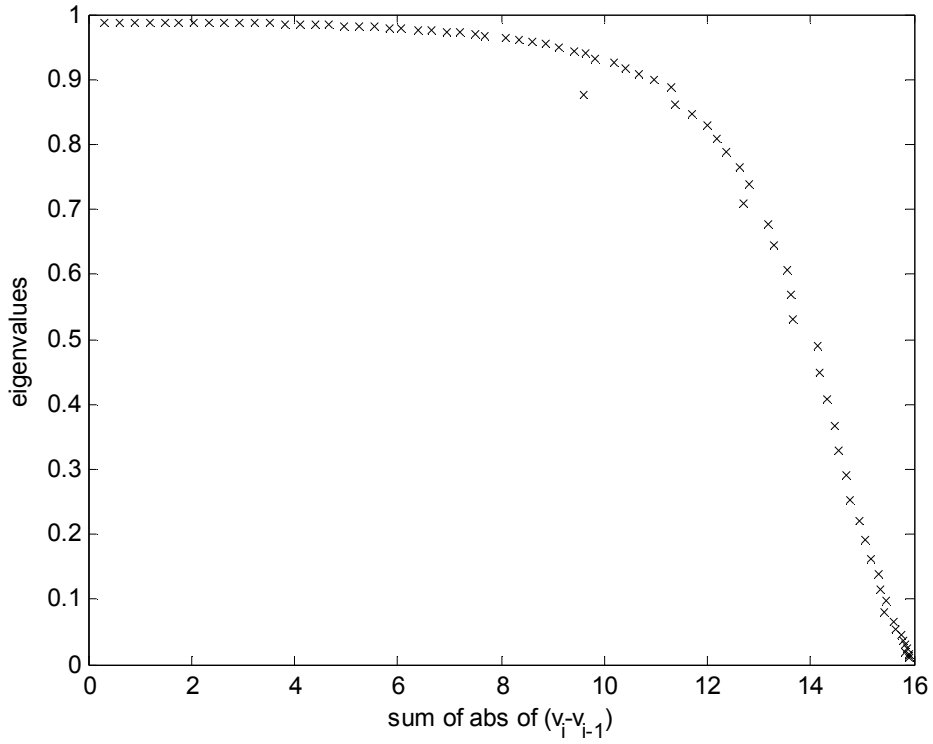


Figure 36 Relationship between eigenvalues and variation or lengthscale of eigenvectors.

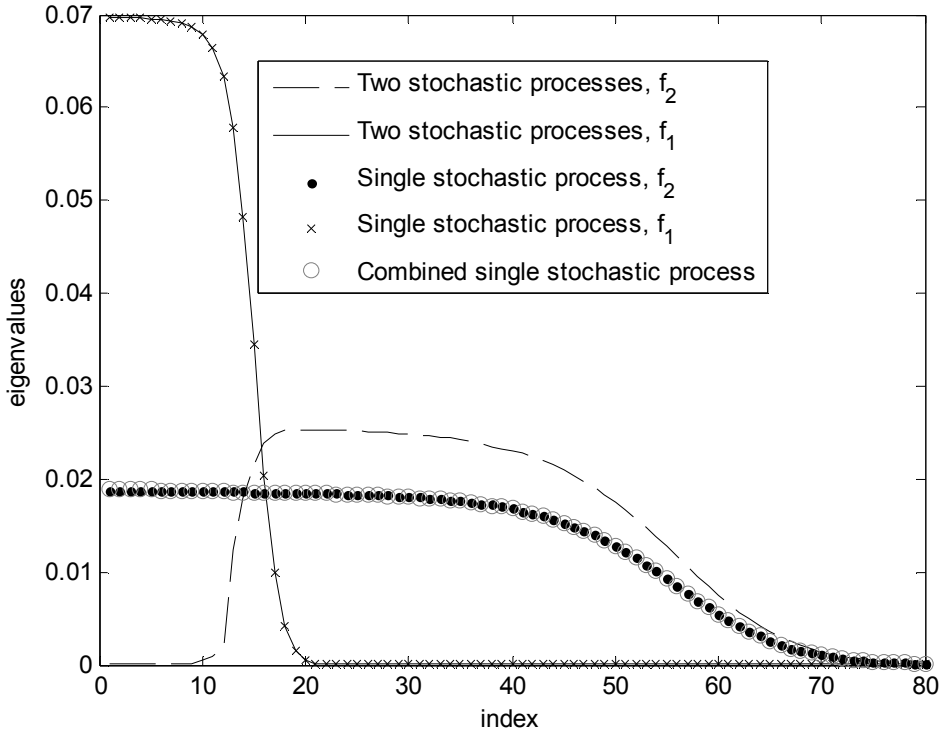


Figure 37 Plots of eigenvalues against indexed points. The posterior for the long lengthscale, **F**, is denoted by f_1 and the posterior for the short lengthscale, **G**, is denoted by f_2.

The eigenvalues for the covariance matrix of $\hat{\mathbf{H}}$ are ordered by magnitude and are plotted in Figure 36, denoted by o. It can be seen that eigenvalues with a broad range of lengthscales dominate. The eigenvalues for the covariance matrices of $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$, $\hat{\mathbf{F}}'$ and $\hat{\mathbf{G}}'$ are ordered by lengthscale; to be precise, they are ordered according to their correlation with the ordered eigenvectors for the covariance matrix of $\hat{\mathbf{H}}$.

In Figure 37, the magnitudes of the eigenvalues are plotted with those for $\hat{\mathbf{F}}$ denoted by ×, those for $\hat{\mathbf{G}}$ denoted by ●, those for $\hat{\mathbf{F}}'$ denoted by the solid line and those for $\hat{\mathbf{G}}'$ denoted by the dashed line. It can be seen that, in the case of $\hat{\mathbf{G}}$, the dominant eigenvalues are similar to those of $\hat{\mathbf{H}}$ but, in the case of $\hat{\mathbf{F}}$, relatively few long lengthscale eigenvalues dominate. Furthermore, in the case of $\hat{\mathbf{F}}'$, the dominant eigenvalues are similar to those of $\hat{\mathbf{F}}$ but, in the case of $\hat{\mathbf{G}}'$, the magnitude of the eigenvalues for the longer lengthscales, i.e. those that dominate $\hat{\mathbf{F}}$ and $\hat{\mathbf{F}}'$, are much reduced. In other words, the transformation has removed the contributions to $\hat{\mathbf{G}}$ at

these longer lengthscales and added them to $\hat{\mathbf{F}}$ thereby meeting the context attribute (a).

4.4.2 Application of Two Gaussian Processes Model

In this section, the application of the two Gaussian processes model to extract information is explored. For clarity, the explanatory variable is always scalar.

In Example 4.5 and Example 4.6, the covariance functions for both Gaussian processes is the standard squared exponential function (17) but with different lengthscale hyperparameters. By construction of these examples, *a priori*, it is known that the nonlinear relationship underlying the measured data consists of two components and all hyperparameters are known. In a specific application, the context may indicate that two components are present and, when they are of different lengthscale as in Example 4.6 and Example 4.8, provide information regarding the lengthscales. When the context does not, the presence of two components may be determined from the data as discussed below using the revised log-likelihood function (29).

The revised log-likelihood function (29) is introduced in Chapter 3.3 to speed up the training procedures. However, not only does the revised log-likelihood function allow faster optimisation, it can also provide additional information when the surface mapping is portrayed as a three-dimensional plot. In particular, the log-likelihood function may be multi-model with several maxima each indicating the presence of a component with different lengthscale.

Example 4.7: Let the nonlinear relationship be $\mathrm{f}(z) = \sin(1.5z) + 0.35\cos(10z)$. 800 data points are measured at 80Hz with additive Gaussian white noise of variance 0.04. The noisy data, together with the two components, is shown in Figure 38. Using a standard single Gaussian Process model with the squared exponential covariance function (17), the revised negative log-likelihood is shown in Figure 39. It is clear from the two minima that there are two lengthscales present, one associated with each of

the sine functions in the nonlinear relationship. Having eliminated the amplitude hyperparameter, the revised negative log likelihood is a function of only the lengthscale hyperparameter, d, and the noise hyperparameter, b. The value of the revised negative log-likelihood function minimised with respect to the noise hyperparameter, b, is plotted against the lengthscale hyperparameter, d, in Figure 40. This two-dimensional plot shows more clearly the presence of the two minima. The hyperparameters are chosen to minimise the revised log likelihood function. With appropriately chosen initial values, the minimisation routine is caused to separately converge on each of the minima. The hyperparameters obtained, corresponding to the long lengthscale minima, are $d = 0.552$, $a = 1.296$ and $n = 0.077$ and, corresponding to the short lengthscale minima, are $d = 20.386$, $a = 0.525$ and $n = 0.070$. Hence, the two components in the data have lengthscale hyperparameter values 0.552 and 20.39, respectively.

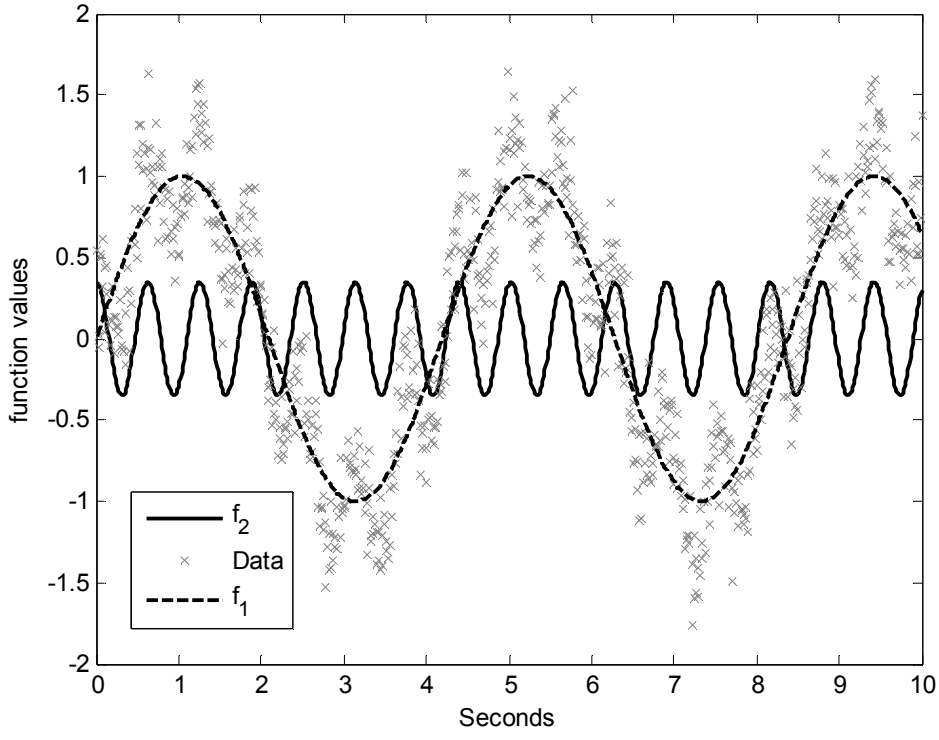


Figure 38 Plots of two sinusoidal functions and sum of the two functions with noise.

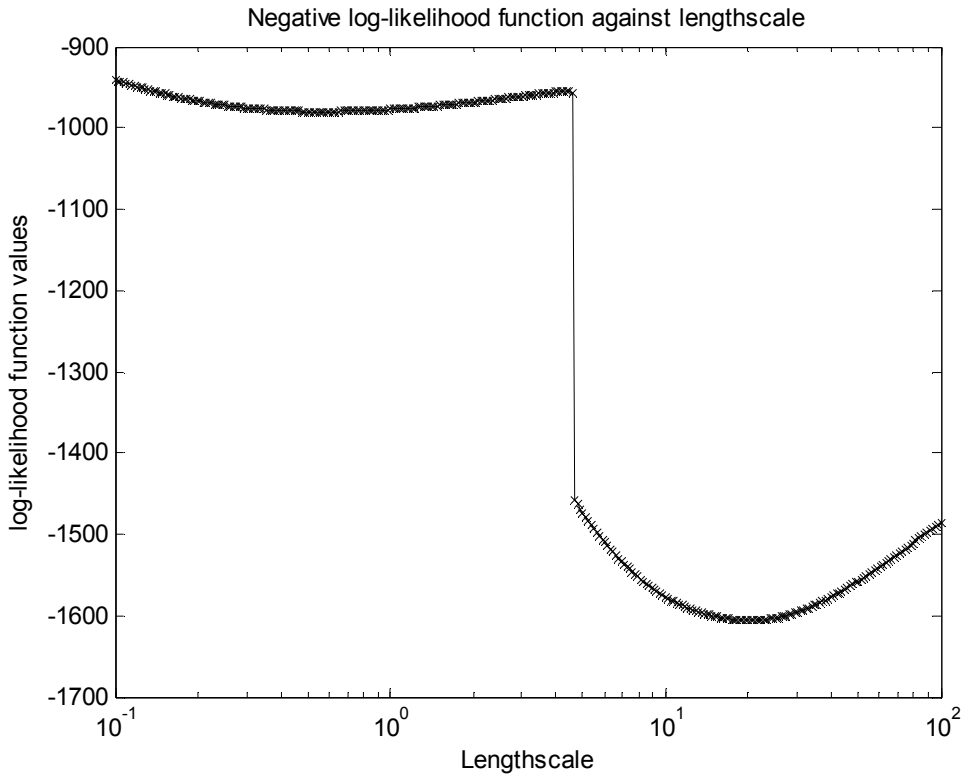


Figure 39 Plot of optimised revised negative log-likelihood function against lengthscale hyperparameter.

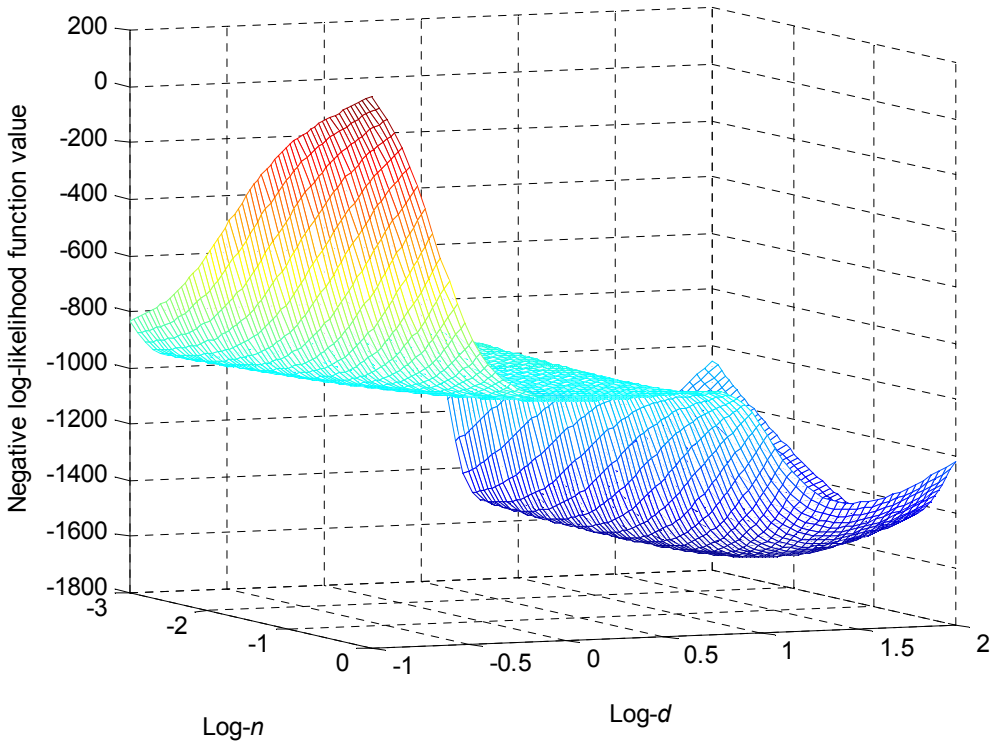


Figure 40 3D plot of revised log-likelihood function against lengthscale and noise variance hyperparameters.

This approach to determine the number of possible lengthscale solution is extremely useful for dataset with explanatory variable that is scalar. However, it cannot be used with all covariance functions. For example, a periodic covariance function of the form

$$C(\mathrm{z}_i,\mathrm{z}_i)=a\exp\left(-\frac{d}{2}\sin^2\left[\pi\lambda\left(\mathrm{z}_i-\mathrm{z}_j\right)\right]\right)+b\delta_{ij} \tag{71}$$

has four hyperparameters and visualising the function mapping in high dimensional space is difficult, or perhaps impossible.

Covariance function is one of the main ingredients in modelling a Gaussian process. Data containing more than one component can be characterised by a compound covariance function. Noise present in the data is assumed to be Gaussian and the components are assumed to be independent. Note that, no attempt is being made here to propagate a Gaussian distribution (or any other distribution) through a nonlinear function.

Suppose that measurements are not of a single function but the sum of several functions with different characteristics; that is, the measured values are $y_i = f_1(z_i) + ... + f_K(z_i) + n_i$. A possible probabilistic description of $h(z) = f_1(z) + ... + f_K(z)$ is by means of the sum of K independent Gaussian processes, $f_z^1, ..., f_z^K$. Let the covariance functions for these stochastic processes be $C_f^1(z_i, z_j), ..., C_f^K(z_i, z_j)$ respectively, then $h_z = (f_z^1 + ... + f_z^K)$ is a stochastic process with covariance function $C_h = C_f^1(z_i, z_j) + ... + C_f^K(z_i, z_j)$, since $f_z^1, ..., f_z^K$ are independent.

Following Chapter 2.4, the prior joint probability distribution for $\mathbf{H} = [h_{1z}, ..., h_{Kz}]^T$ and **Y** is Gaussian with mean zero and covariance matrix

$$E\left[\begin{bmatrix}\mathbf{H}\\ \mathbf{Y}\end{bmatrix}\begin{bmatrix}\mathbf{H}^T & \mathbf{Y}^T\end{bmatrix}\right] = \left[\begin{array}{c|c}\Lambda_{HH} & \Lambda_{HH}\\ \hline \Lambda_{HH} & Q_H\end{array}\right]$$

with $\Lambda_{HH} = E[\mathbf{H}\mathbf{H}^T]$ and $Q_H = \mathbf{B} + \Lambda_{HH}$. Applying partitioned matrix lemma, the posterior joint probability distribution for **H**, conditioned on the data **Y**, remains Gaussian with mean $\overline{M}$ and covariance matrix $\overline{\Lambda}$, where

$$\overline{M} = \Lambda_{HH} Q_H^{-1} \mathbf{Y} \tag{72}$$

$$\overline{\Lambda} = \Lambda_{HH} - \Lambda_{HH} Q_H^{-1} \Lambda_{HH} \tag{73}$$

The prediction for **H** is the mean $\overline{M}$ with confidence interval of two times the standard deviation ($\pm 2\sqrt{diag(\overline{\Lambda})}$). The concept is best illustrated with an example.

Example 4.8 *(Simple Two GPs Example)*. Consider a GP-generated function using the commonly used prior covariance function (74) for a Gaussian process with scalar explanatory variable, *z*. It ensures that the measurements associated with nearby values of the explanatory variable should have higher covariance than widely separated values; *a* is related to the overall mean amplitude and *d* is inversely related to the lengthscale.

$$a \exp\left\{-\frac{d}{2}(z_i - z_j)^2\right\} \tag{74}$$

Assuming that the measurements contain two components of different characteristics, f(z) and g(z), let the covariance function for f_z be (74) with a = 1.8 and d = 2.5, and the covariance function for g_z be (74) with a = 0.95 and d = 120. That is, f_z has a long lengthscale and g_z has a short lengthscale. Also, let the measurement noise be Gaussian white noise with variance 0.04, i.e. $\mathbf{B}_{ij} = b\delta_{ij}$, where δ_{ij} is the Kronecker delta. Gaussian regression is applied to a set of 800 measurements, $\mathrm{y}_i = \mathrm{f}(z_i) + \mathrm{g}(z_i) + \mathrm{n}_i$, sampled at 100Hz, with f(z_i) and g(z_i) the sample values for the stochastic processes f_z and g_z, respectively. The data values, with the prediction error and confidence intervals obtained using (72) and (73), respectively, are shown in Figure 33.

Remark 4.4: In Example 4.5, the probabilistic description for h(z) is by means of a single Gaussian process, h_z, with compound covariance function, $C_{\mathrm{h}} = (C_{\mathrm{f}} + C_{\mathrm{g}})$. An alternative would be by means of a Gaussian process, $\widetilde{\mathrm{h}}_z$, with the covariance function, $\widetilde{C}_{\mathrm{h}}$, of the form (74). A suitable value of the lengthscale hyperparameter d is the same as that for the short lengthscale in Example 4.5, i.e. $\widetilde{\mathrm{h}}_z$ has the same short lengthscale as g_z in Example 4.5, but a suitable value of the amplitude hyperparameter a is larger, i.e. the value maximising the likelihood of the data. This simpler probabilistic description is almost as effective as the probabilistic description with the covariance function C_{h}, since the prediction and confidence interval at any point depend primarily on nearby values rather than remoter values.

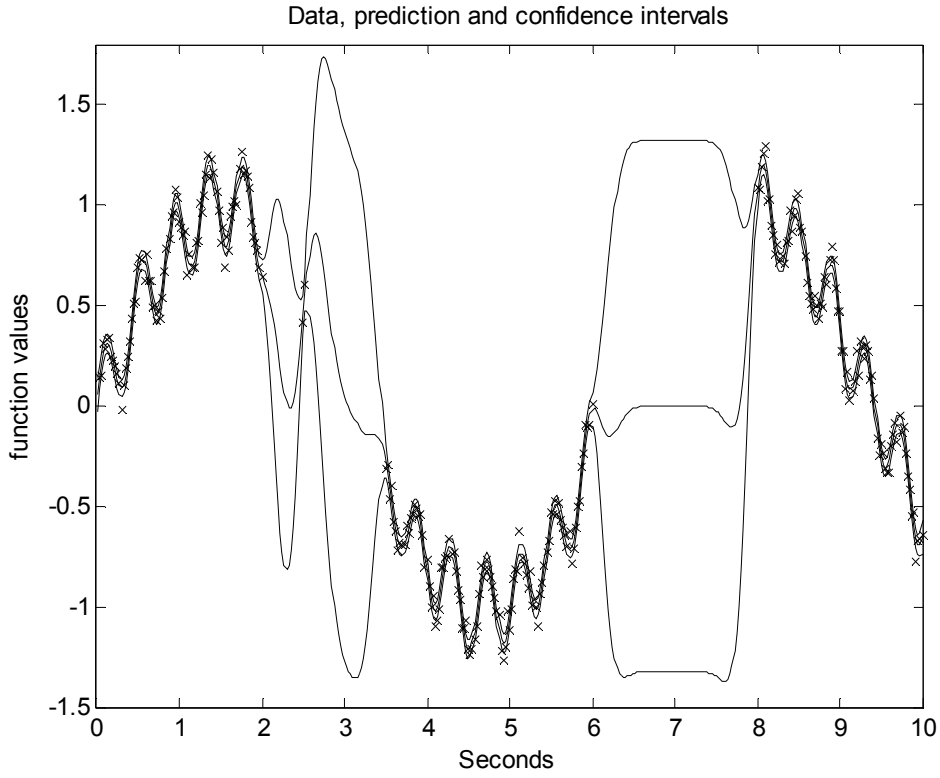


Figure 41 Variable density data, prediction and confidence interval.

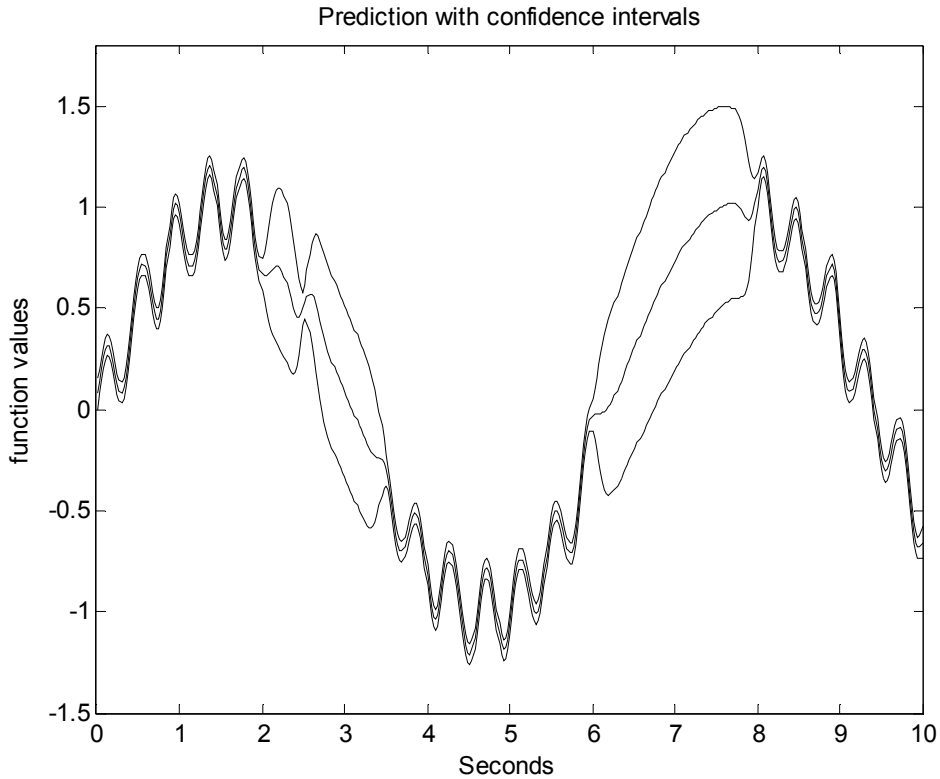


Figure 42 Prediction and confidence interval with long and short lengthscale components.

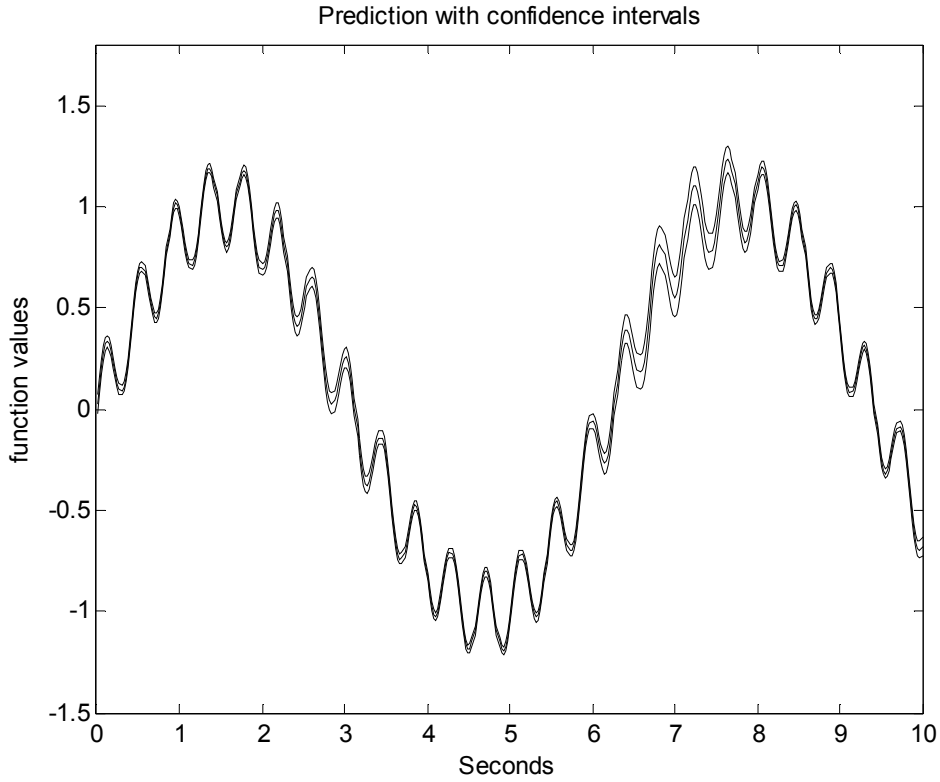


Figure 43 Prediction and confidence interval with long lengthscale and periodical components.

The benefits for prediction using a compound covariance function, such as C_h, become apparent when the density of the data varies. Consider the data in Figure 41. It clearly contains a long lengthscale component and a short lengthscale component. Both are sinusoids. However, there are large gaps in the data between 2 and 3.5 (except for two values near 2.5) and between 6 and 8. Consider the first situation when the covariance function is chosen to be (74), with the hyperparameter d corresponds to the short lengthscale. The prediction and confidence interval obtained are shown in Figure 41. Since it now depends on nearby values, the prediction over the data gaps is poor. Indeed, no prediction is made over the second gap between 6 and 8. Over the data gaps, the confidence interval, reflecting the uncertainty in the prediction, is enlarged. Now, consider the situation when the covariance function is chosen to be similar to that of Example 4.5; that is, it is the sum of two functions, the long lengthscale and short lengthscale. The prediction and confidence interval are shown in Figure 42. Over the data gaps, the prediction has improved due to the inclusion of the long lengthscale component in the covariance function. The confidence interval, reflecting the uncertainty, has reduced considerably. The periodic

nature of the short lengthscale component can be exploited to further improve the prediction over the gaps. A suitable prior covariance function for a periodic Gaussian process with scalar explanatory variable is given by (75).

$$a \exp\left\{-\frac{d}{2}\sin^2\left[\pi\lambda\left(z_i - z_j\right)\right]\right\} \tag{75}$$

Finally, consider the situation when the covariance function is chosen to be the sum of (74) and (75), the former being the long lengthscale component and the latter for the periodic short-term component in the data. The prediction and confidence interval are shown in Figure 43. Over the gaps, the prediction is much improved and the confidence interval much narrower.

The idea here is extendable to measurements having more than two functions, each with a different characteristic, with components represented by a suitable class of covariance functions. Note that, the use of compound covariance function in Gaussian process prior models has been investigated by several researchers (Rasmussen, 1996; Gibbs, 1997; Williams, 1999).

4.4.3 Training Procedures

The implementation of multiple Gaussian process models is dependent on a proper set of training procedures. Two possible pre-emptive approaches may be required before conducting the optimisation stage.

Hyperparameter Initialisation

This section has been covered in Chapter 3.4.

Training Procedure for Multiple Gaussian Processes with Same Explanatory Variable

In the case of independent multiple stochastic processes as outlined in §4.4.4, it is rather interesting to note that the solutions are neither unique nor trivial. Thus, a set of proper training procedures is necessary. It is possible to obtain different solutions given the same data with different training methods. As the solution is not unique in every case, there is a need to explore and provide a standard and improved solution to solve the problem of multiple stochastic processes in a proper and systematic manner.

Given that the data consists of several stochastic processes, it is important to first decide the correct order of training the Gaussian process model. The order of the training procedure is vital to the result of the solution (recall that the solution is not unique). For simplicity, a model of two stochastic processes is used as an illustration. Assume that this model is available; it is important to first decide whether to train the long lengthscale first or the short lengthscale first, alternatively whether the component with periodic feature should be trained first or the non-periodic one first. A proper training procedure should be done in the fashion as laid out in Remark 4.5.

Remark 4.5 (Training procedure for multiple lengthscale Gaussian processes).
Assume that the data contains more than one component, each with a distinct characteristic, the following steps are undertaken to ensure that each component corresponds to a suitable set of hyperparameter values that are adapted from training.

1. Train the data with a simple covariance function[9], consisting of a single term. The set of hyperparameters, which correspond to the component which could not fit into other components, is adapted to maximise the log-likelihood function. For instance, the long lengthscale component, that is not going to fit into the short lengthscale component, should be trained first.
2. Use the adapted hyperparameter values to perform Gaussian regression on the data. Compute the residue.

[9] A simple covariance function mentioned here is not a compound covariance function.

3. Train the residue to obtain hyperparameters that corresponds to next component, which is not going to fit into subsequent remaining components.
4. Repeat step 2 and 3 until the number of trainings corresponding to the number of stochastic processes is achieved.

The procedure outlined above is based on training the data using a single covariance function. In summary, components of the multiple Gaussian processes model that are not going to fit into successive components should be trained first. The residue is then computed after each subsequent optimisation. The purpose of this procedure is to obtain all the hyperparameter values of the multiple Gaussian processes model.

A unified training procedure, with the covariance function being the sum of two or more covariance functions, is useless, such that the resulting hyperparameters do not correspond to their respective components. This is due to the inability of the optimisation routine to interpret the distribution of the contributions from each stochastic process. The predictions are prone to high uncertainties as the Gaussian process lacks the ability to distribute a proper involvement played by each component of the covariance functions.

The set of hyperparameters, corresponding to different components of the Gaussian process models, is achieved with a proper training procedure. These adapted hyperparameters are then used to compute the prediction and the standard deviations of the posterior joint probability distribution.

4.4.4 Extension to General Case Prediction

Theorem 4.1 in §4.4 is extendable to predictions of any data points of the explanatory variable. Consider **F** and **G** to be the respective long and short lengthscale components with scalar predictions $\hat{f}_{\hat{z}}$ and $\hat{g}_{\hat{z}}$, at any possible values of the explanatory variable, z, and vector predictions $\hat{f}_z$ and $\hat{g}_z$ at values of the explanatory variable of the given training data. It follows that the posterior joint probability

distribution of **F** and **G** conditioned on the data **Y** remains Gaussian with mean $\overline{\mathrm{M}}$ and covariance matrix $\overline{\Lambda}$, as given by (76) and (77), respectively.

$$\overline{\mathrm{M}} = \begin{bmatrix} \hat{\mathrm{f}}_{\hat{z}} \\ \hat{\mathrm{g}}_{\hat{z}} \\ \hat{\mathrm{f}}_{z} \\ \hat{\mathrm{g}}_{z} \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathrm{f}_{\hat{z}z}} Q^{-1}\mathbf{Y} \\ \Lambda_{\mathrm{g}_{\hat{z}z}} Q^{-1}\mathbf{Y} \\ \Lambda_{\mathrm{f}_{zz}} Q^{-1}\mathbf{Y} \\ \Lambda_{\mathrm{g}_{zz}} Q^{-1}\mathbf{Y} \end{bmatrix} \tag{76}$$

$$\overline{\Lambda} = \begin{bmatrix} \Lambda_{\mathrm{f}_{\hat{z}\hat{z}}} - \Lambda_{\mathrm{f}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{f}_{z\hat{z}}} & -\Lambda_{\mathrm{f}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{g}_{z\hat{z}}} & \Lambda_{\mathrm{f}_{\hat{z}z}} - \Lambda_{\mathrm{f}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{f}_{zz}} & -\Lambda_{\mathrm{f}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{g}_{zz}} \\ -\Lambda_{\mathrm{g}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{f}_{z\hat{z}}} & \Lambda_{\mathrm{g}_{\hat{z}\hat{z}}} - \Lambda_{\mathrm{g}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{g}_{z\hat{z}}} & -\Lambda_{\mathrm{g}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{f}_{zz}} & \Lambda_{\mathrm{g}_{\hat{z}z}} - \Lambda_{\mathrm{g}_{\hat{z}z}} Q^{-1}\Lambda_{\mathrm{g}_{zz}} \\ \Lambda_{\mathrm{f}_{z\hat{z}}} - \Lambda_{\mathrm{f}_{zz}} Q^{-1}\Lambda_{\mathrm{f}_{z\hat{z}}} & -\Lambda_{\mathrm{f}_{zz}} Q^{-1}\Lambda_{\mathrm{g}_{z\hat{z}}} & \Lambda_{\mathrm{f}_{zz}} - \Lambda_{\mathrm{f}_{zz}} Q^{-1}\Lambda_{\mathrm{f}_{zz}} & -\Lambda_{\mathrm{f}_{zz}} Q^{-1}\Lambda_{\mathrm{g}_{zz}} \\ -\Lambda_{\mathrm{g}_{zz}} Q^{-1}\Lambda_{\mathrm{f}_{z\hat{z}}} & \Lambda_{\mathrm{g}_{z\hat{z}}} - \Lambda_{\mathrm{g}_{zz}} Q^{-1}\Lambda_{\mathrm{g}_{z\hat{z}}} & -\Lambda_{\mathrm{g}_{zz}} Q^{-1}\Lambda_{\mathrm{f}_{zz}} & \Lambda_{\mathrm{g}_{zz}} - \Lambda_{\mathrm{g}_{zz}} Q^{-1}\Lambda_{\mathrm{g}_{zz}} \end{bmatrix} \tag{77}$$

where $Q = \mathbf{B} + \Lambda_{\mathrm{f}_{zz}} + \Lambda_{\mathrm{g}_{zz}}$.

Proof 4.2: To obtain the posterior joint probability distribution, on the condition that the posterior remains independent, appropriate modifications to $\overline{\mathrm{M}}$ and $\overline{\Lambda}$ are made by applying the following transformation,

$$\overline{\mathrm{M}} \to \overline{\mathrm{M}}_0 = T\overline{\mathrm{M}}\,;\ \overline{\Lambda} \to \overline{\Lambda}_0 = T\overline{\Lambda}T^{\mathrm{T}}$$

where

$$T = \begin{bmatrix} I & 0 & 0 & \Lambda_{\mathrm{f}_{\hat{z}z}} Q_{\mathrm{f}_{zz}}^{-1} \\ 0 & I & 0 & -\Lambda_{\mathrm{f}_{\hat{z}z}} Q_{\mathrm{f}_{zz}}^{-1} \end{bmatrix};\ Q_{\mathrm{f}_{zz}} = \mathbf{B}^{*} + \Lambda_{\mathrm{f}_{zz}}$$

Similarly, the estimate of part of the data in **G**, which is explainable by **F**, should be transferred from the prediction for **G** to the prediction for **F**. To ensure that the cross-correlation of the posterior joint probability distribution between **F** and **G** is zero, $\mathbf{B}^{*}$ is simply **B**. The posterior joint probability distribution is a combination of independent Gaussian processes, with mean vector and covariance matrix to be (78) and (79), respectively.

$$\overline{\mathrm{M}}_0 = \begin{bmatrix} \Lambda_{\mathrm{f}_{\hat{z}z}} Q_{\mathrm{f}_{zz}}^{-1}\mathbf{Y} \\ \left(\Lambda_{\mathrm{f}_{\hat{z}z}} + \Lambda_{\mathrm{g}_{\hat{z}z}}\right) Q^{-1}\mathbf{Y} - \Lambda_{\mathrm{f}_{\hat{z}z}} Q_{\mathrm{f}_{zz}}^{-1}\mathbf{Y} \end{bmatrix} \tag{78}$$

$$\overline{\Lambda}_0 = \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix}, \text{ where} \tag{79}$$

$$U = \Lambda_{\mathrm{f}_{\hat{z}\hat{z}}} - \Lambda_{\mathrm{f}_{\hat{z}z}} Q_{\mathrm{f}_{zz}}^{-1} \Lambda_{\mathrm{f}_{z\hat{z}}}$$
$$V = \left(\Lambda_{\mathrm{f}_{\hat{z}\hat{z}}} + \Lambda_{\mathrm{g}_{\hat{z}\hat{z}}}\right) - \left(\Lambda_{\mathrm{f}_{\hat{z}z}} + \Lambda_{\mathrm{g}_{\hat{z}z}}\right) Q^{-1} \left(\Lambda_{\mathrm{f}_{z\hat{z}}} + \Lambda_{\mathrm{g}_{z\hat{z}}}\right)$$

Remark 4.6: The third row of $\overline{\mathrm{M}}$ in (76), and the third row and column of $\overline{\Lambda}$ in (77) are not required to compute the posterior joint probability distribution. It is shown merely for clarity and as a formality.

For general case scenario, Theorem 4.1 can be extended further to explain K independent stochastic processes within a given model, for K > 1.

Theorem 4.2 *(extended version of Theorem 4.1)*: Given that the prior joint probability distribution for K independent stochastic processes, $\mathbf{F}_1,\ldots,\mathbf{F}_\mathrm{K}$ and $\mathbf{Y}$ is Gaussian with mean zero and covariance matrix Λ, the posterior joint probability distribution for $\left[\mathbf{F}_1^\mathrm{T} \quad \cdots \quad \mathbf{F}_\mathrm{K}^\mathrm{T}\right]^\mathrm{T}$ conditioned on the dataset M, and subject to the condition that they remain independent, is Gaussian with mean $\overline{\mathrm{M}}_0$ and $\overline{\Lambda}_0$. The mean is given by

$$\overline{\mathrm{M}}_0 = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 - \Gamma_1 \\ \vdots \\ \Gamma_\mathrm{K} - \Gamma_{\mathrm{K}-1} \end{bmatrix}$$

where $\Gamma_x = \left(\sum_x \Lambda_{\hat{z}z}^x\right)\left(\mathbf{B} + \sum_x \Lambda_{zz}^x\right)^{-1} \mathbf{Y}$, $\forall x \le \mathrm{K}$. The standard deviation, $\overline{\Sigma}_0$, is defined to be the square-root of the diagonal elements of $\overline{\Lambda}_0$.

$$\overline{\Sigma}_0 = \left[diag\left\{ \begin{matrix} \Theta_1 \\ \Theta_2 - \Theta_1 \\ \vdots \\ \Theta_\mathrm{K} - \Theta_{\mathrm{K}-1} \end{matrix} \right\} \right]^{\frac{1}{2}}$$

where $\Theta_x = \left(\sum_x \Lambda_{\hat{z}\hat{z}}^x\right) - \left(\sum_x \Lambda_{\hat{z}z}^x\right)\left(\mathbf{B} + \sum_x \Lambda_{zz}^x\right)^{-1}\left(\sum_x \Lambda_{z\hat{z}}^x\right)$, $\forall x \le \mathrm{K}$. Proof 4.2 is extended to multiple stochastic processes below.

Proof 4.3 (extended version of Proof 4.1): Similarly, the prediction for the contribution from one component, may alternatively, be in part explainable by another suitable component, using a suitable choice of smoothing kernel. Hence it should be appropriate to modify by transferring suitable part of the components of the data to appropriate section of the posterior. For simplicity, denote $\Phi^x = \Lambda^x_{\hat{z}z}\left(\mathbf{B} + \Lambda^x_{zz}\right)^{-1}$ and partition $T = \left[I \mid T_R\right]$, such that

$$T_R = \begin{bmatrix} \Phi^1 & \Phi^1 & \Phi^1 & \cdots & \Phi^1 \\ -\Phi^1 & \Phi^2 & \Phi^2 & \cdots & \Phi^2 \\ 0 & \sum_{k=1}^{2}\Phi^k & \Phi^3 & \cdots & \Phi^3 \\ 0 & 0 & \sum_{k=1}^{3}\Phi^k & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \Phi^{\mathrm{K}} \\ 0 & 0 & \cdots & 0 & \sum_{k=1}^{\mathrm{K}}\Phi^k \end{bmatrix}$$

From the covariance of the posterior joint probability distribution, the correlations between the K components are zero. Note that $\overline{\mathrm{M}}$ is defined to have one row less and $\overline{\Lambda}$ has a row and a column omitted. That is,

$$\overline{\mathrm{M}} = \begin{bmatrix} \Lambda^1_{\hat{z}z} \\ \vdots \\ \Lambda^{\mathrm{K}}_{\hat{z}z} \\ \hline \Lambda^2_{zz} \\ \vdots \\ \Lambda^{\mathrm{K}}_{zz} \end{bmatrix} Q^{-1}\mathbf{Y} \qquad \text{and} \qquad \overline{\Lambda} = \left[\begin{array}{c|c} \overline{\Lambda}_{\hat{z}\hat{z}} & \overline{\Lambda}_{\hat{z}z} \\ \hline \overline{\Lambda}_{z\hat{z}} & \overline{\Lambda}_{zz} \end{array}\right]$$

where $\overline{\Lambda}_{\hat{z}\hat{z}}$ is the cross-correlation between the predicting values of the explanatory variable and itself, $\overline{\Lambda}_{z\hat{z}} = \left(\overline{\Lambda}_{\hat{z}z}\right)^{\mathrm{T}}$ is the correlation between the predicting values of the explanatory variable and those of the measurement data. $\overline{\Lambda}_{zz}$ is the correlation between the values of the explanatory variable of the measurement data. Note that, there is one component less in the definition of $\overline{\Lambda}_{z\hat{z}}$ and $\overline{\Lambda}_{zz}$. The order in which the components are arranged is not of great importance; unlike the order of the training procedure, which is discussed in the following section.

4.5 Case Study

An application based on multiple Gaussian processes model with common explanatory variable is illustrated in this section. The requirement of this application is to identify missing data measured from a nonlinear dynamic system (CATS Benchmark, 2004) using the model.

A 5000-point time-series data, sampled at 1Hz and consisting of five missing gaps, is chosen to be the test data. These gaps are located at the following locations, (981s – 1000s), (1981s – 2000s), (2981s – 3000s), (3981s – 4000s) and (4981s – 5000s). The poor quality of the data can be seen in Figure 44 and the power spectral density of the measurement data is shown in Figure 45. The latter reveals that the data comprises of several components with various frequencies.

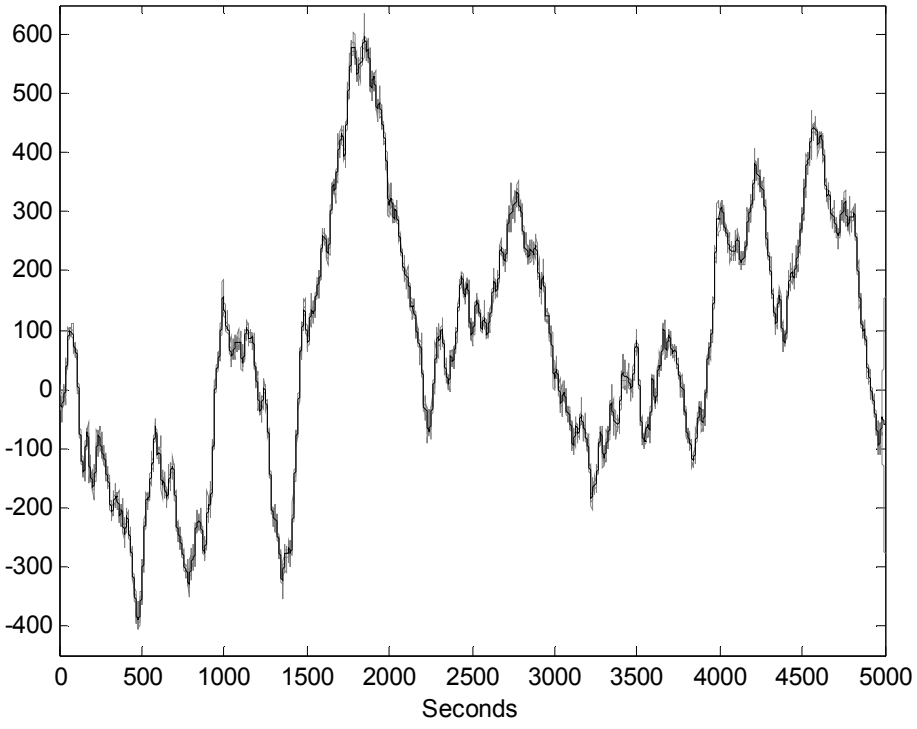


Figure 44 Data and long lengthscale prediction with confidence intervals.

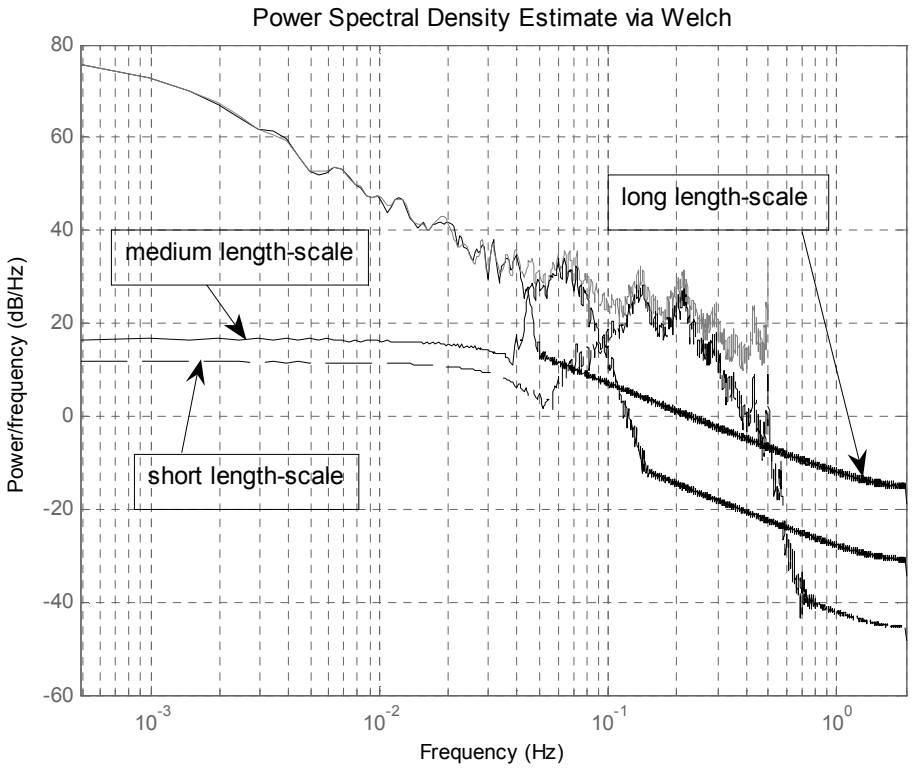


Figure 45 Power spectrums of the data and predictions of three independent components.

Attempts to extract three components from the data have resulted the hyperparameter value of the noise variance, b, to be 35.913. Using the chosen covariance function (74), for f_z, g_z and h_z, the corresponding hyperparameters are a_f and d_f, a_g and d_g, and a_h and d_h, respectively. The covariance for the measurements, y_i, at time t_i, is (80).

$$\mathrm{E}\left[y_i, y_j\right] = b\delta_{ij} + \sum_{k=\{\mathrm{f,g,h}\}} a_k \exp\left[-\frac{d_k}{2}\left(t_i - t_j\right)^2\right] \tag{80}$$

Appropriate training methods are applied to obtain the hyperparameters. Since the time-series data contains missing gaps, the generalised Schur algorithm (see Chapter 3.5.2) is employed to handle this dataset. The prediction of the long lengthscale component with confidence intervals (black lines) is shown in Figure 44, with adapted hyperparameters $a_f = 2.422 \times 10^4$ and $d_f = 0.0038$. A typical section, from 2850s to 3150s, of the medium lengthscale prediction with confidence intervals using $a_g = 83.6391$ and $d_g = 0.0473$, is illustrated in Figure 46. The prediction and confidence intervals of the short lengthscale component, with $a_h = 55.962$ and $d_h = 1.274$, are plotted in Figure 47 from 2900s to 3100s. In addition, the total error estimate and confidence intervals (grey lines) are shown in Figure 48. Note that in the section from

4981s to 5000s, the confidence interval amplifies drastically since the data points are near the edge of the data region and no data is available at those values of the explanatory variable.

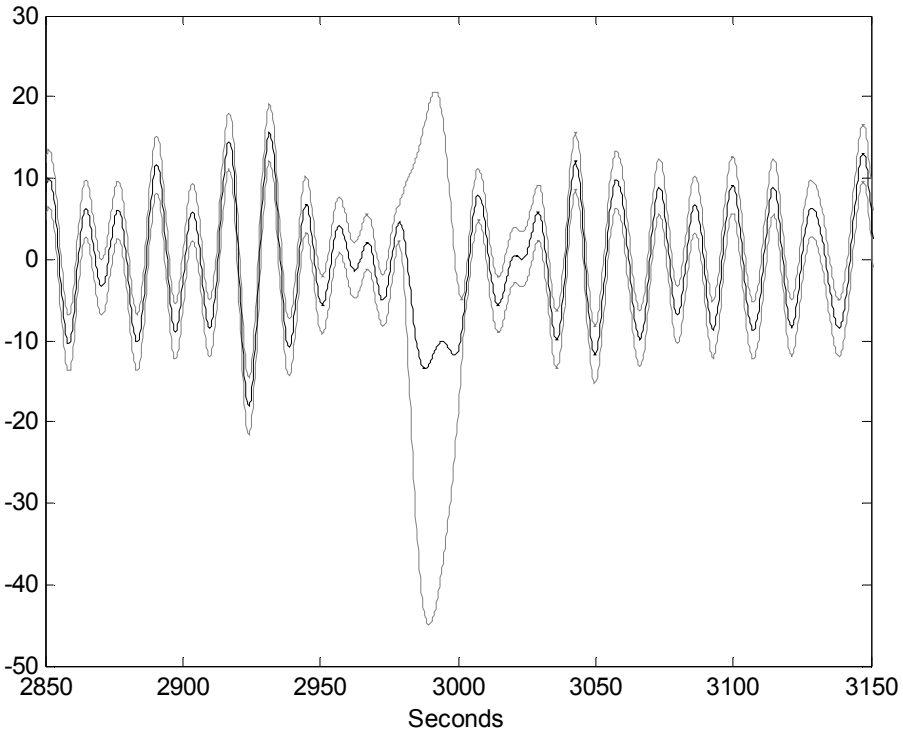


Figure 46 Medium lengthscale component with confidence intervals.

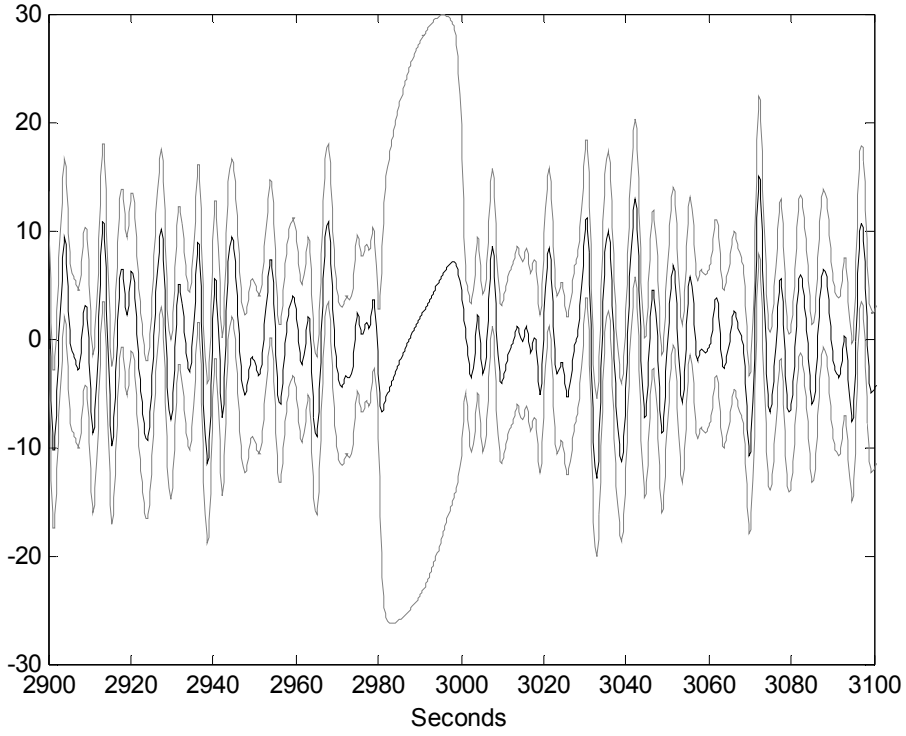


Figure 47 Short lengthscale component with confidence intervals.

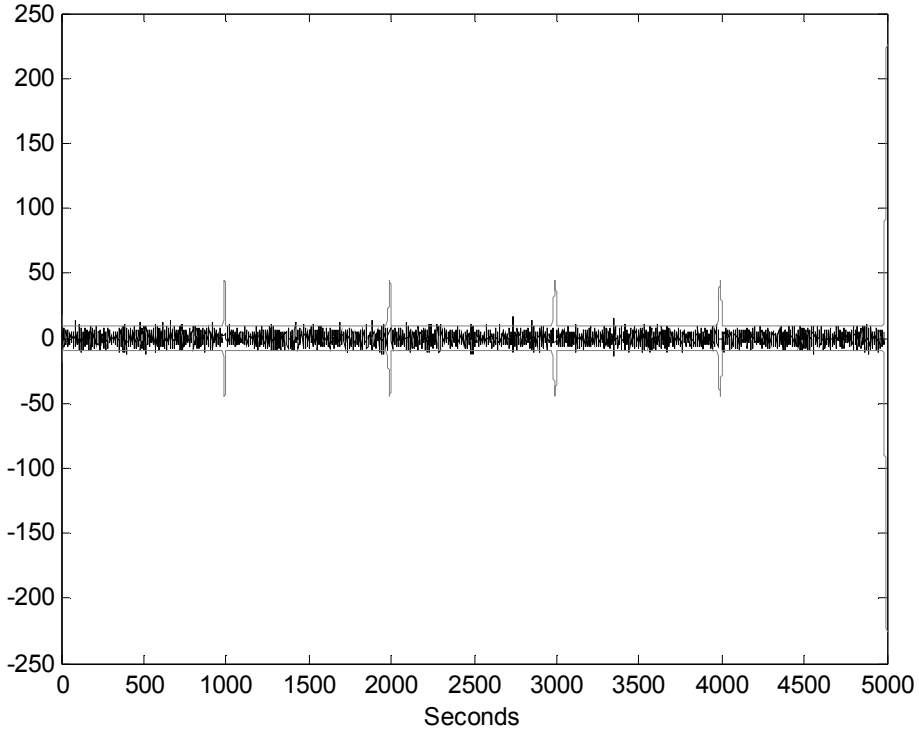


Figure 48 Error estimate with confidence intervals.

Individual components of the data are successfully identified and extracted with the multiple stochastic processes model. The final frequency spectrums of the fits to the individual components are shown in Figure 45.

4.6 Discussion

Methods on extraction of components with different characteristics from the data using Gaussian processes based on models with two or more stochastic processes have been developed. The extracted components of the posterior joint probability distribution are completely independent from each other.

When there is any degree of arbitrariness in the data, the normal Gaussian process model gives a totally unusable prediction due to the arbitrary component. The transformation is then used to select one model from all possible models to remove arbitrariness without affecting the likelihood of the data. For data with different explanatory variables, the choice is made by imposing a condition on an average of the components; this is equivalent to adding a constant to one component and subtracting the same constant from another component, in the case of two Gaussian processes model. The approach does not prejudice the Gaussian processes model and its ability to explain the data in anyway, hence it is perfectly fine to do so. For data with same explanatory variable, the choice is made by using additional information provided within the context of the data. For example, components should be treated as independent and that the short lengthscale component does not contribute to the long lengthscale component, in the case of two Gaussian processes model.

From knowledge of the engineering context, it is clear that some situations are applicable with the multiple Gaussian processes model. For instance, the two Gaussian processes model gives usable predictions when normal Gaussian process models do not allow. This is solely due to the removal of arbitrariness. A conclusion is that Gaussian process modelling cannot be done in isolation without reference and exploitation of the context. In addition, there is no need to quantify improvement with numerous numerical experiments. The standard Gaussian process model works in

some situation when used to extract components, but when it does not work, it is perfectly clear that it does not. Hence, the multiple Gaussian processes model provides an alternative to resolve the issue.

Identification of components using multiple Gaussian processes has been effectively applied on some applications, such as de-trending of data, data with missing gaps, and data with periodic, linear and quadratic features. With this model, the fit of the prediction is much better, as the error bars have shown to be narrower. The application of the multiple Gaussian processes model is applied on the wind turbine data, in the form of a case study in the following chapter.

Chapter 5

Case Study: Identification of Wind Turbine Dynamics Using Gaussian Processes

5.1 Introduction

Gaussian process prior model is one of the many methods for inferring nonlinear dynamic systems from measured data to identify the underlying structure of the nonlinear system. Motivated by the interest in Gaussian process and its capability to perform data analysis on nonlinear dynamic functions, an application of system identification is performed on the wind turbine data to identify the dynamic structure of the machine.

The chosen application is a set of measurements obtained from a wind turbine. These measurements are sampled at 40Hz, for a run of 600 seconds. The set of data includes measurement readings taken from the rotor speed, the wind speed calibration from the anemometer mounted on top of the body, located behind the blades, and the pitch angle values registered on the controller system.

Since explicit calculations of matrix operations of the Gaussian process prior models are limited to medium-scale dataset, fast algorithms developed in Chapter 3 are introduced to handle the large-scale 24,000 points wind turbine data. Given that the noisy measurement data are sampled at fixed intervals, the datasets are simply time-series. As such, the structure of the covariance matrices exhibits some special properties, i.e. Toeplitz, thus having low displacement ranks. Fast algorithms, i.e. the modified Durbin-Levinson's algorithm and the generalised Schur algorithm, are capable of handling such structured matrices with low displacement ranks. Furthermore, the wind turbine dataset contains dynamics that can be characterised in the frequency domain, with the model exhibiting a range of operating frequencies. Identification procedure to train the large-scale data by the use of hyperparameter initialisation is applied beforehand.

It was suggested (Leithead et al., 2003a) that the measurement data contains at least two components, i.e. due to the aerodynamics and drive-train dynamic cross-interference with the electro-mechanical components. Applying the concept of classifying each independent component in the data as a single stochastic process, thereby, subsuming multiple independent stochastic processes, the technique extracts the required lengthscale from the measurement data, ensuring that it remains independent with absolute zero cross-correlation with respect to other components within the data.

Modelling these extracted components into state-space formulation requires an additional step of spatial filtering to be applied on the wind speed data. This is due to the point wind speed measurement being a poor representation of the wind-field. Hence, spatial filtering is required to account for the averaging over the rotor disc. (Note that the spatial filter introduces a small delay on the wind speed data. This is compensated by advancing the spatially filtered wind speed by an equivalent time increment of 0.24 seconds, in this case). A probabilistic description of a single stochastic process is used to predict the fit of the outcome, with relation to the components of the explanatory variable. In the case of the wind turbine data, the rotor acceleration, computed from the first order derivative of the rotor speed, is modelled as a function of the rotor speed, wind speed and blade pitch angle. With the intent to separate the wind speed component from the rotor speed and pitch angle for a possible

controller design, a single stochastic process is insufficient. Due to the uncorrelated relationship between the aerodynamics features for the wind speed and the drive train dynamics of the rotor speed and blade pitch angle, the state equation can be modelled to be the sum of two independent Gaussian processes. This is particularly useful in designing the controller for the wind turbine, such that only the function, which depends on the rotor speed and pitch angle, is required.

Engineered by the Gaussian process prior model, the wind turbine dynamics can be analysed and identified in a probabilistic manner with the use of non-parametric modelling.

5.2 Efficient Algorithms Implementation for Time-Series Data

Two main issues are resolved with the implementation of fast algorithms, developed in §3.5, for Gaussian regression on the large-scale 24,000 wind turbine time-series dataset. Firstly, the $O(N^2)$ memory requirement to store the $N \times N$ covariance matrix is replaced by the introduction of vector-level storage algorithm. Secondly, $O(N^3)$-operations, viz. log-determinant and matrix inversion of the covariance matrix, are reduced to $O(kN^2)$-operations with the implementation of fast algorithms.

To further improve optimisation procedure, algorithm based on hyperparameter initialisation (see §3.4) is applied prior the training. This allows faster convergence since proper initial values for the hyperparameters that are specific to the case are systematically obtained. Hence, nonsensical solutions are suppressed. This additional approach is useful in some cases to prevent any possibility of under-fitting or over-fitting of the data.

5.3 Identification of Wind Turbine Dynamics

Methods and algorithms, developed in the previous few chapters of this thesis, are applied on a physical application, viz. identification of the wind turbine dynamics.

Several identification procedures using Gaussian process prior models are carried out here. The identification procedure of the wind turbine dynamics are classified into two main parts:

1. Phase 1 – *Data cleaning*
2. Phase 2 – *Relationship construction*

Briefly, the wind turbine datasets, consisting of time-series data of the wind speed, rotor speed and the blade pitch angle, are heavily corrupted with noise. Gaussian regressions are applied to these datasets to remove the noise in the first phase of the identification procedure. Models with multiple Gaussian processes with common explanatory variable are applied in this phase.

In the second phase, the predictions obtained from the first phase of the procedure are constructed to form a dynamic relationship between the aerodynamic torque and the functions of wind speed, rotor speed and the blade pitch angle. The requirement here is to confirm the independent relationship between function of the rotor speed and the blade pitch angle, and the function of wind speed. This is a rather important verification because by separating the wind speed components from the rest of the data, it allows engineers to design a more efficient controller based on the knowledge that the other component depends only on the rotor speed and blade pitch angle, both of which can be controlled mechanically or electronically. On the other hand, the wind speed (or rather, wind field that passes through the rotor blades) can not be controlled.

5.3.1 About the Data

The raw wind turbine data consists of the wind speed, rotor speed and the blade pitch angle, specifically, site measurements obtained concurrently from a commercial 1MW wind turbine. The measurement data consists of a run of 600 seconds sampled at 40 Hz and is corrupted by significant measurement noise. The dynamics of the wind turbine, when operated above rated wind speed, are assumed to conform to the simple model as illustrated in Figure 49. The generator torque of the 1MW wind turbine

considered here is kept constant by the controller. The dynamics of the converter are sufficiently fast so that no distinction is required to be made between the demand and achieved generator torque. The generator torque is regulated by means of the blade pitch angle. The controller maintains the generator speed within a small percentage of rated value.

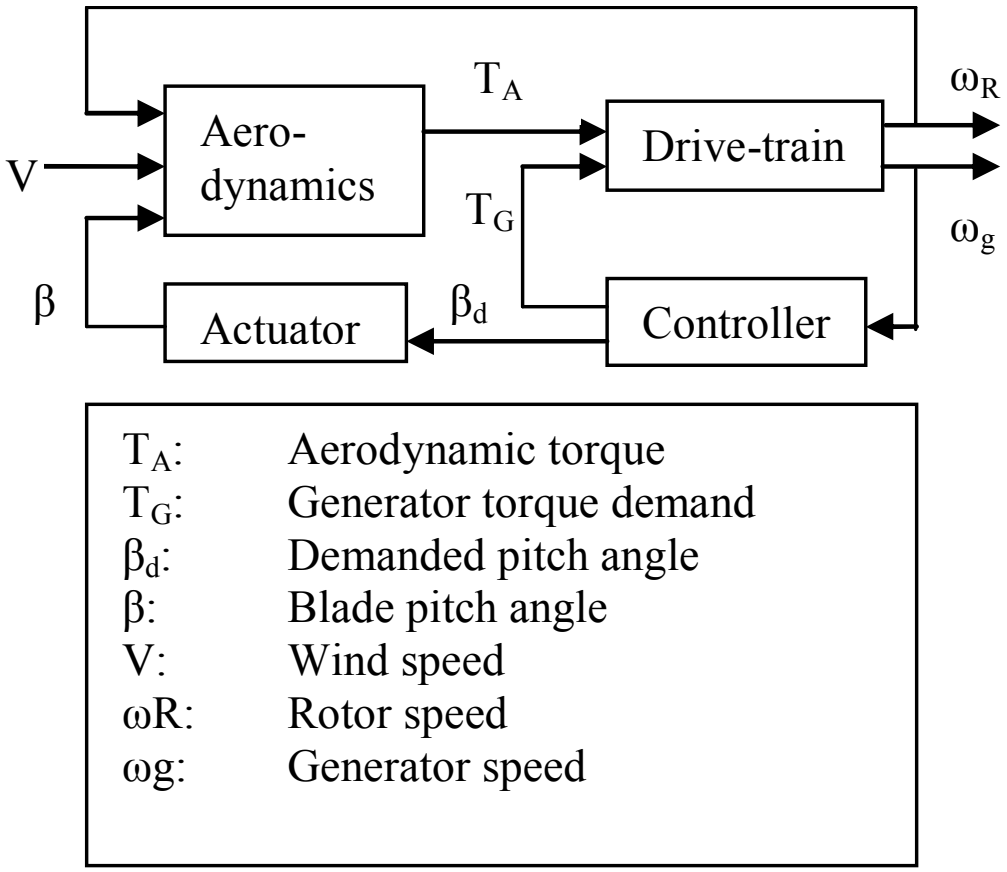


Figure 49 A simple model of the wind turbine dynamics.

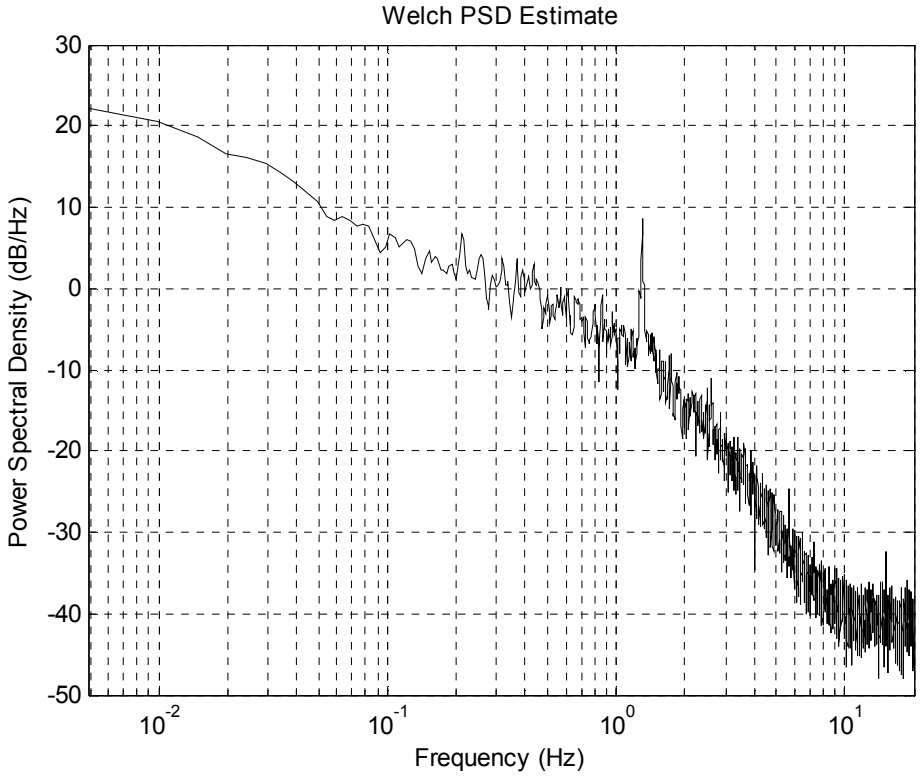
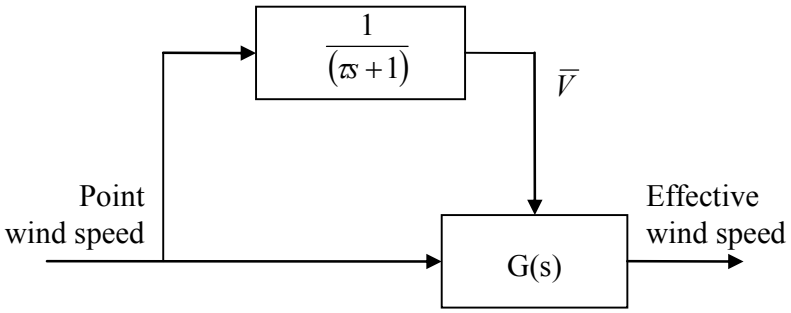


Figure 50 Wind speed spectrum.

The values of the wind speed are obtained from a mean adjusted nacelle anemometer measurement, with a mean value of 17.5m/s, apart from the turbulence intensity of 12.5% observed. The power spectral density function for the wind speed is depicted in Figure 50. As the measurement of the point wind speed results in a poor representation of the wind field, it has to be spatially filtered to account for the averaging over the rotor disc, thus induces a small delay on the wind speed. This can be compensated by an equivalent time increment of 0.24 seconds to the filtered wind speed. The resulting effective wind speed may then be interpreted as the uniform wind speed over the rotor disc such that the power spectral density function of the aerodynamic torque induced by the effective wind speed and the power spectral density function of the aerodynamic torque induced by the non-uniform spatially varying wind field actually experienced by the rotor, are the same for frequencies less than $1\Omega_0$, where Ω_0 is the rated rotor speed. The spatial filter is shown in Figure 51.



$$G(s) = \frac{\sqrt{2} + \sigma(\bar{V})s}{\left(\sqrt{2} + \sqrt{\alpha}\sigma(\bar{V})s\right)\left(1 + \sigma(\bar{V})s/\sqrt{\alpha}\right)}$$

$\alpha = 0.55$

$$\sigma(\bar{V}) = \frac{\gamma R}{\bar{V}}$$

R: Radius of rotor

γ: Wind speed decay factor (1.3)

τ: Time constant (20s)

Figure 51 Spatial filter implemented for wind speed.

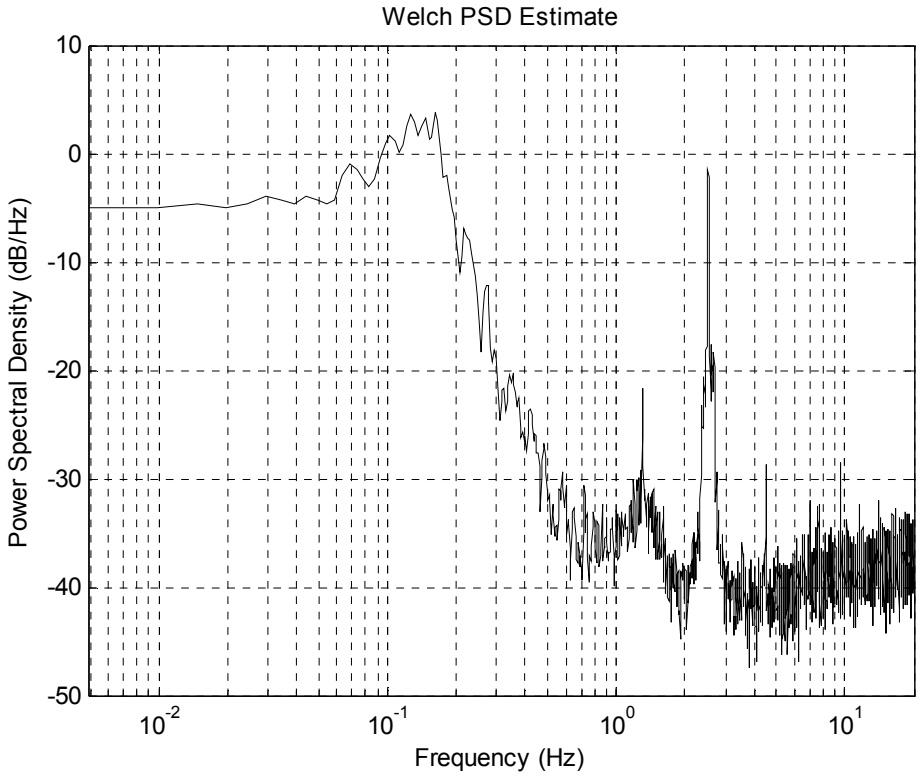


Figure 52 Rotor speed spectrum.

The measured rotor speed is some combination of the measurement of the rotor speed and the generator speed scaled by the gearbox ratio. The effectual result of this mixing is significant at the intermediate and high frequencies but has no consequence

at low frequency; that is, the measured rotor speed and the actual rotor speed are the same at low frequency. The power spectral density function for the rotor speed is shown in Figure 52. Clearly, the data is heavily corrupted with background noise of approximately $3x10^{-3}$. Above a frequency of 4 rad/s, the only features observable above the background noise level are two spectral peaks. The first is due to rotational sampling of the wind field at $3\Omega_0$ and the second is the first drive-train mode enhanced by its close proximity to $6\Omega_0$.

The power spectral density function for the blade pitch angle is shown in Figure 53. Evidently, only the spectral peak at $3\Omega_0$ can be seen.

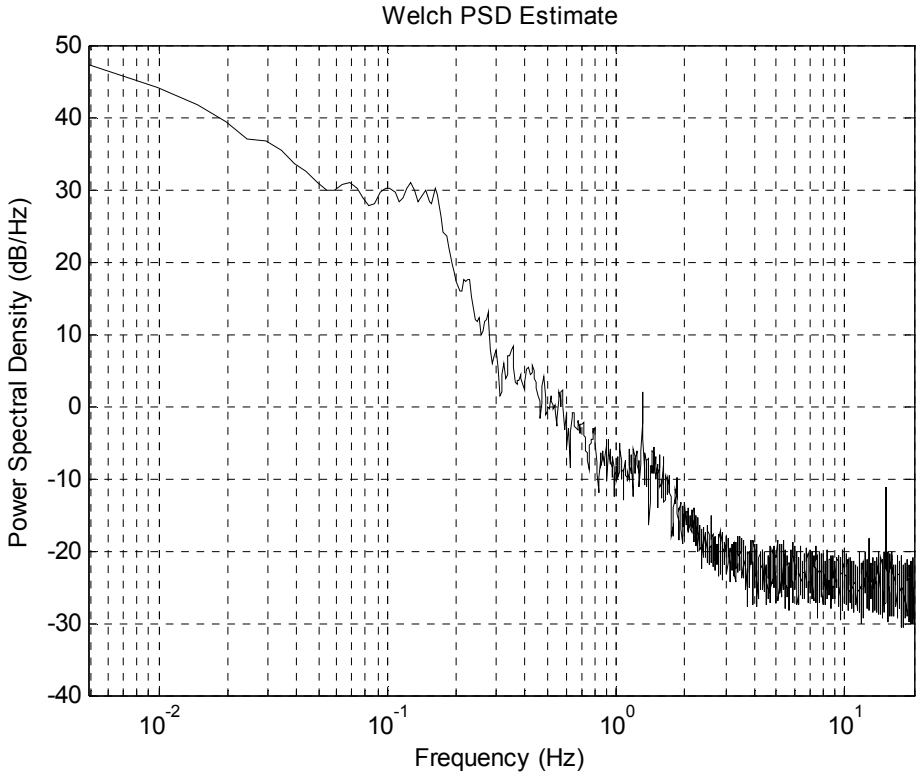


Figure 53 Pitch angle spectrum.

5.3.2 Cleaning Up Raw Data (Phase 1)

In an attempt to perform Gaussian regression on these data, hyperparameter initialisation developed in earlier chapter is applied to ensure that proper initial values are chosen for the optimisation routine.

Rotor speed data

The identification procedure using models with multiple Gaussian processes in §4.4 is applied to the raw wind turbine measurement data. Consider the rotor speed measurement; the data has a long lengthscale component due to the variations in the aerodynamic torque, caused by changes in the wind speed and the pitch angle of the rotor blades, and a short lengthscale component due to the structural and mechanical dynamics of the machine. The two components, from 260s to 280s, can be clearly seen in Figure 54 as can the poor quality of the data. The requirement here is to estimate the long lengthscale component in the rotor speed and the short lengthscale component belonging to the drive-train dynamics of the variable speed wind turbine. In addition, it is of interest to identify the rotor acceleration using Gaussian process prior models, which is required to formulate the aerodynamic torque relationship with the wind speed, rotor speed and the pitch angle in the second phase of the identification procedure. Due to the contact between the tower blade and the aerodynamics of the wind present in the data, small oscillations belonging to the "3p" component, as seen in Figure 52, is induced by the interaction between the structural and electromechanical dynamics. Therefore, there are presently three components, requiring three independent stochastic processes, f_t, g_t and h_t, corresponding to the long, medium and short lengthscale, respectively to be identified. A simple squared exponential covariance function,

$$a \exp\left[-\frac{d}{2}\left(t_i - t_j\right)^2 \right]$$

is chosen to represent each of the three components present in the data, such that a and d are hyperparameters of the covariance function. The measurement noise is assumed to be Gaussian white noise with variance b. It follows that the covariance for the measurement y_i at time t_i is

$$\mathrm{E}\left[y_i, y_j\right] = a_f \exp\left[-\frac{d_f}{2}\left(t_i - t_j\right)^2 \right] + a_g \exp\left[-\frac{d_g}{2}\left(t_i - t_j\right)^2 \right] + a_h \exp\left[-\frac{d_h}{2}\left(t_i - t_j\right)^2 \right] + b\delta_{ij}$$

where δ_{ij} is a Kronecker delta which is one if and only if $i = j$ and zero otherwise. The subscript "f", "g" and "h" on the hyperparameters represent the long, medium and short components, respectively.

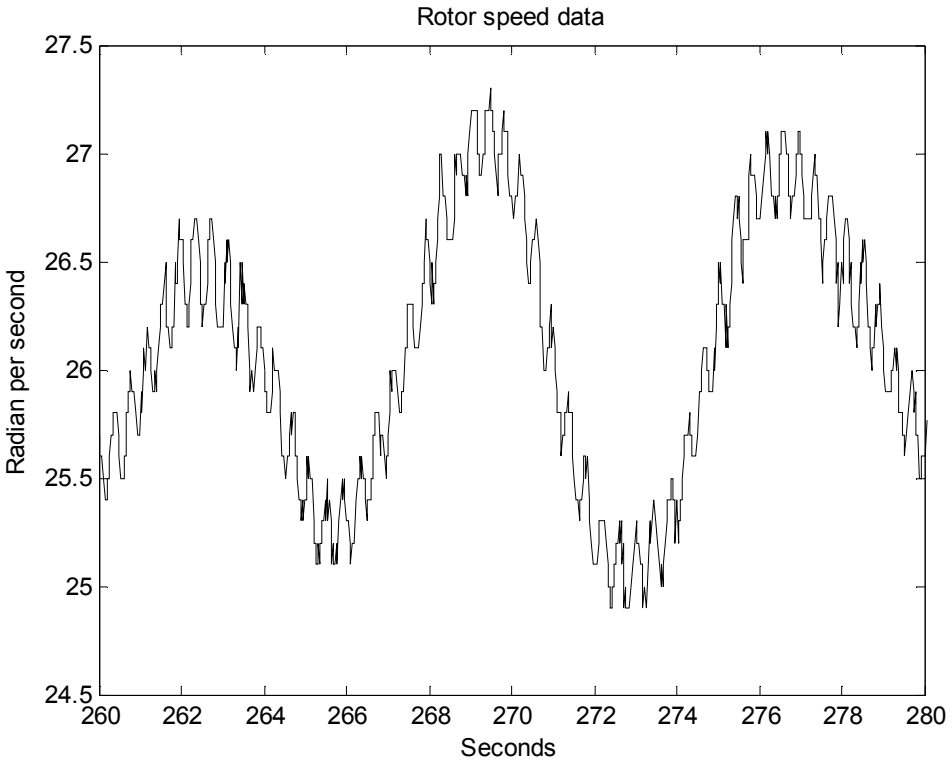


Figure 54 Noisy data of the rotor speed measurement.

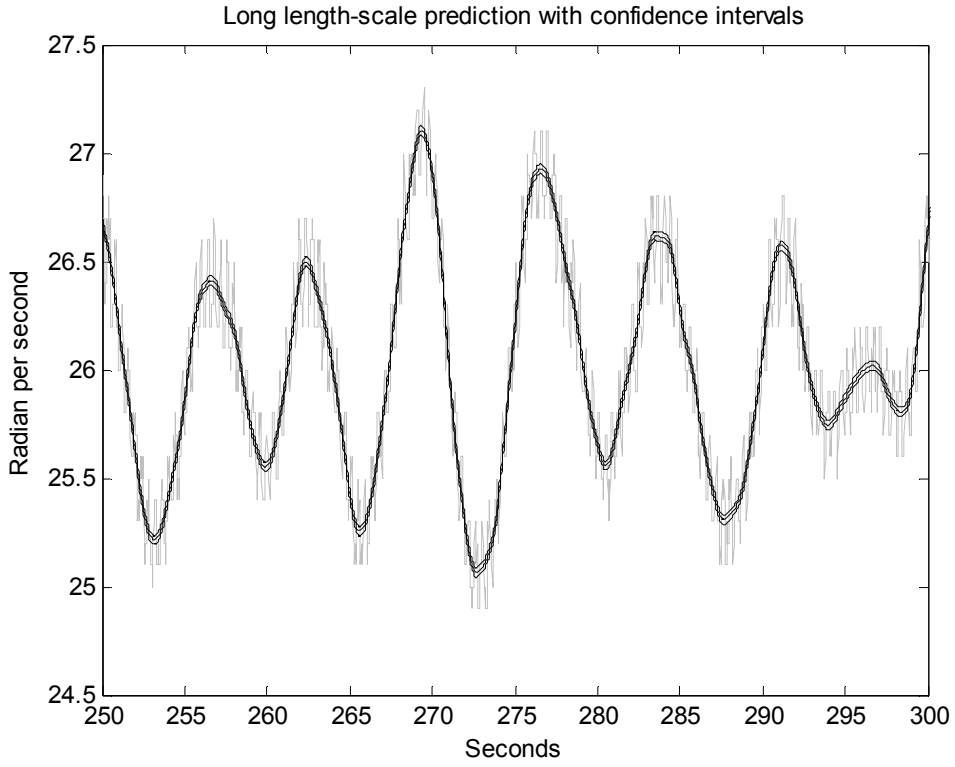


Figure 55 Rotor speed prediction with confidence intervals and noisy data (grey).

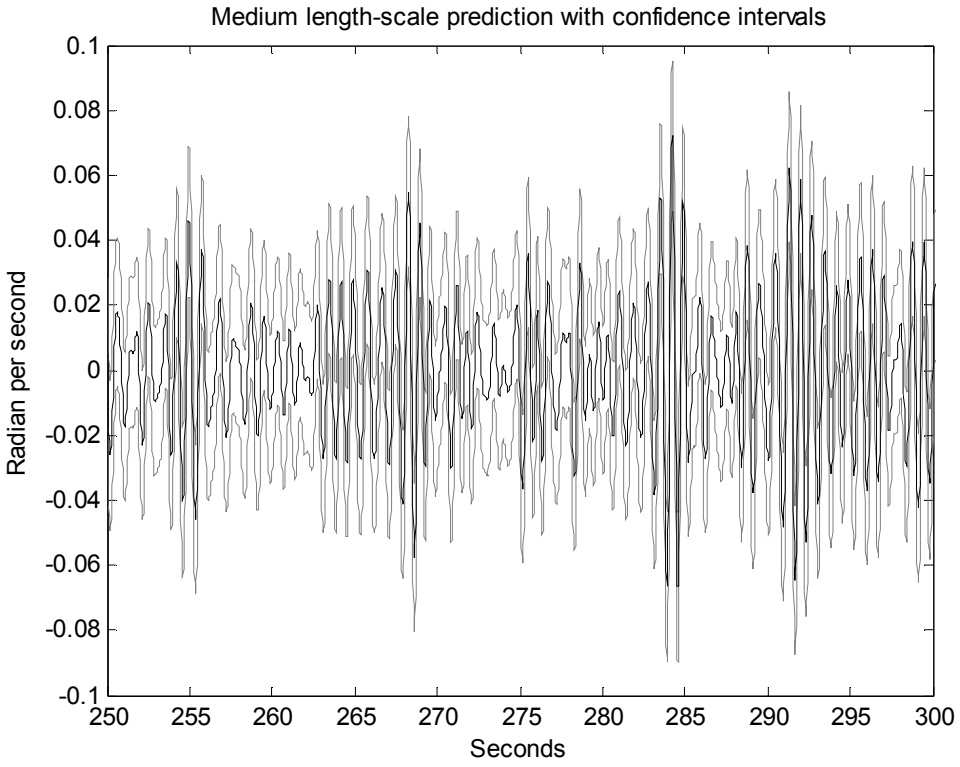


Figure 56 Prediction and confidence intervals of the "3p" component.

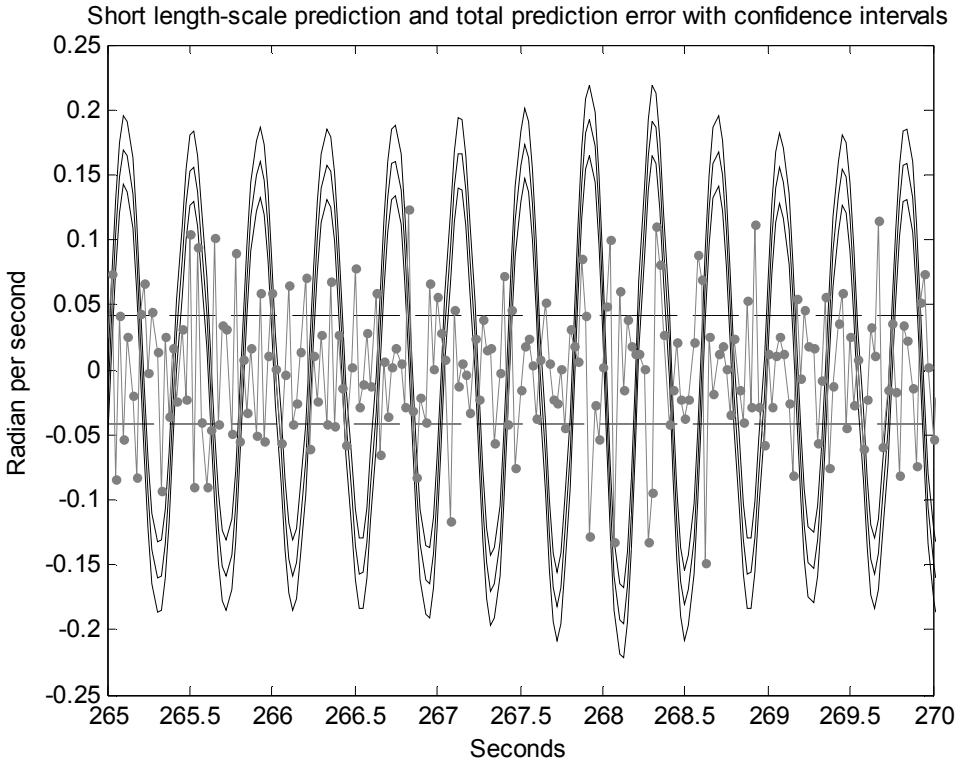


Figure 57 Total error estimate and generator speed prediction with confidence intervals.

Gaussian regression based on model with three stochastic processes is applied to the time-series rotor speed data. The hyperparameters, whilst adapted to maximise the log-likelihood function, are $a_f = 170$, $d_f = 1$, $a_g = 10$, $d_g = 6$, $a_h = 0.776$, $d_h = 25$ and $b = 0.0026$. A section, from 250s to 300s, of the prediction for the rotor speed, viz. the long lengthscale component, together with two times standard deviations is shown in Figure 55. The same section of the prediction for the medium lengthscale component consisting of the "3p" interaction, with confidence intervals is shown in Figure 56. A typical short section of the short lengthscale component, from 265s to 270s, illustrating the generator speed is depicted in Figure 57 (black lines). As it can be perceived, the rotor speed and the generator speed are successfully segregated from each other via the approach using model with multiple Gaussian processes. Finally, the residue (grey lines) and its confidence intervals (dashed lines) are also shown in shown in Figure 57.

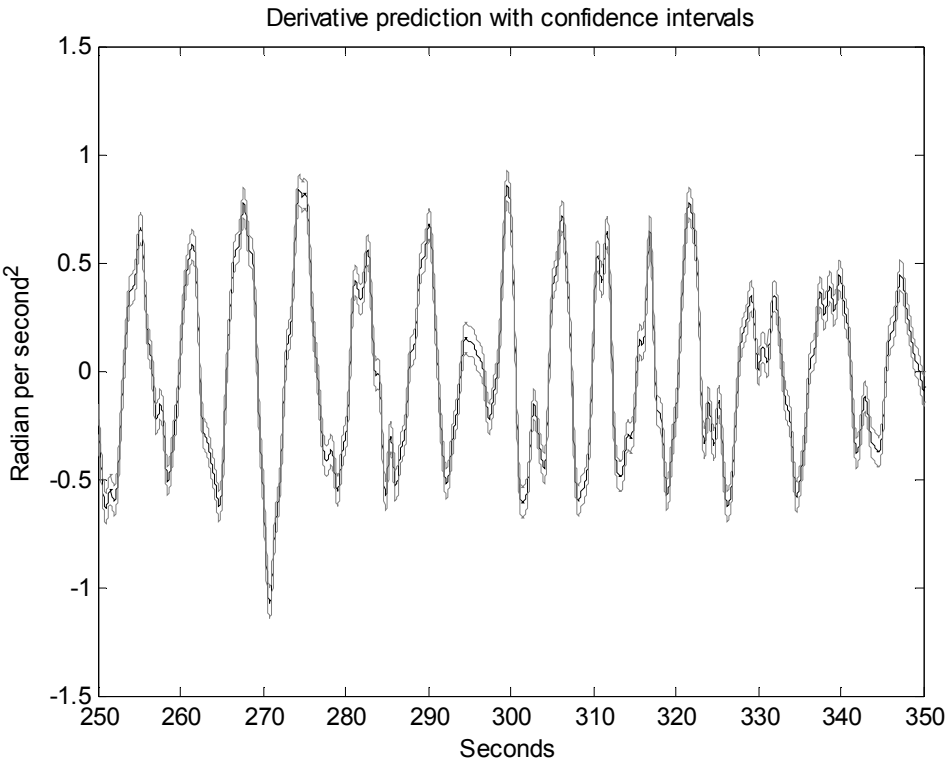


Figure 58 Prediction of the rotor acceleration with confidence intervals.

In addition, the rotor acceleration, which is the derivative of the rotor speed, is also extracted using the derivative stochastic process. The prediction and the confidence intervals of the rotor acceleration, of a section from 250s to 350s, are shown in Figure 58. Because of the implementation of the model with multiple stochastic processes, the confidence intervals obtained are very narrow for the long lengthscale components, i.e. the rotor speed and the rotor acceleration. Model using a single stochastic process will result in a much enlarged confidence interval. This is due to the residue, after extracting the long lengthscale, being represented by shorter lengthscale components that are present in the data, other than noise. In the single stochastic process model, these short components are treated as noise by the Gaussian process. Such representation of the posteriors is very useful in providing a more accurate probabilistic description of the true noise level in the data.

The components of the rotor speed, generator speed and the "3p" interaction are successfully extracted and independently identified by Gaussian regression. Furthermore, the rotor acceleration is also computed. The procedure carried out here is only to illustrate the capability of using models with multiple Gaussian processes to

identify independent components from the datasets. To follow up on Phase 2 in the next section, only the long lengthscale components of the rotor speed and rotor acceleration are of interest. Similarly, only the required components, i.e. the long lengthscale components, will be identified from the wind speed and blade pitch angle data. A model with two Gaussian processes is sufficient.

Wind speed data

Model with two independent Gaussian processes is applied on the raw wind speed data, as shown in Figure 59. The data has a mean wind speed value of approximately 17.5m/s. With the same form of the squared exponential covariance function as the rotor speed for each component present in the data, the covariance for measurement y_i at time t_i is given by

$$\mathrm{E}\left[y_i, y_j\right] = a_f \exp\left[-\frac{d_f}{2}\left(t_i - t_j\right)^2\right] + a_g \exp\left[-\frac{d_g}{2}\left(t_i - t_j\right)^2\right] + b\delta_{ij}$$

where hyperparameters with subscript "*f*" represent the component with long lengthscale and those with subscript "*g*" represent the short lengthscale component. The hyperparameters obtained whilst adapted to maximise the log-likelihood function are $a_f = 1.025\text{x}10^3$, $d_f = 0.27$, $a_g = 0.35$, $d_g = 108.8$ and $b = 0.0024$. The extracted long lengthscale component of the wind speed data is shown in Figure 60. The short lengthscale component can be extracted in the same way as the generator speed. However, since only the wind speed is required, other components are not of interest.

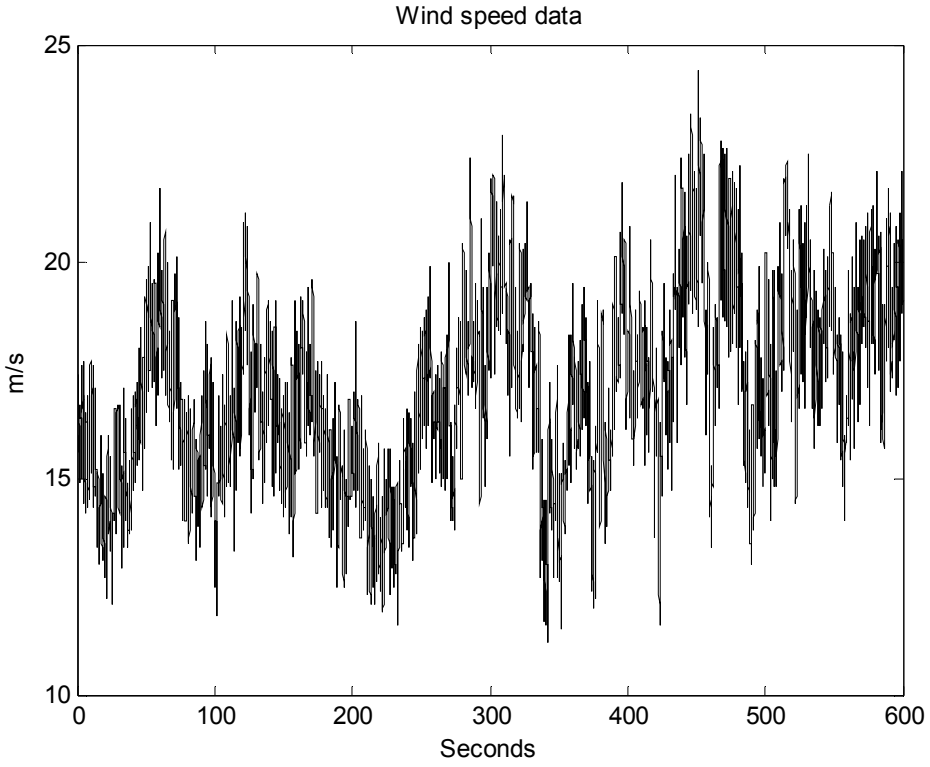


Figure 59 Raw wind speed data.

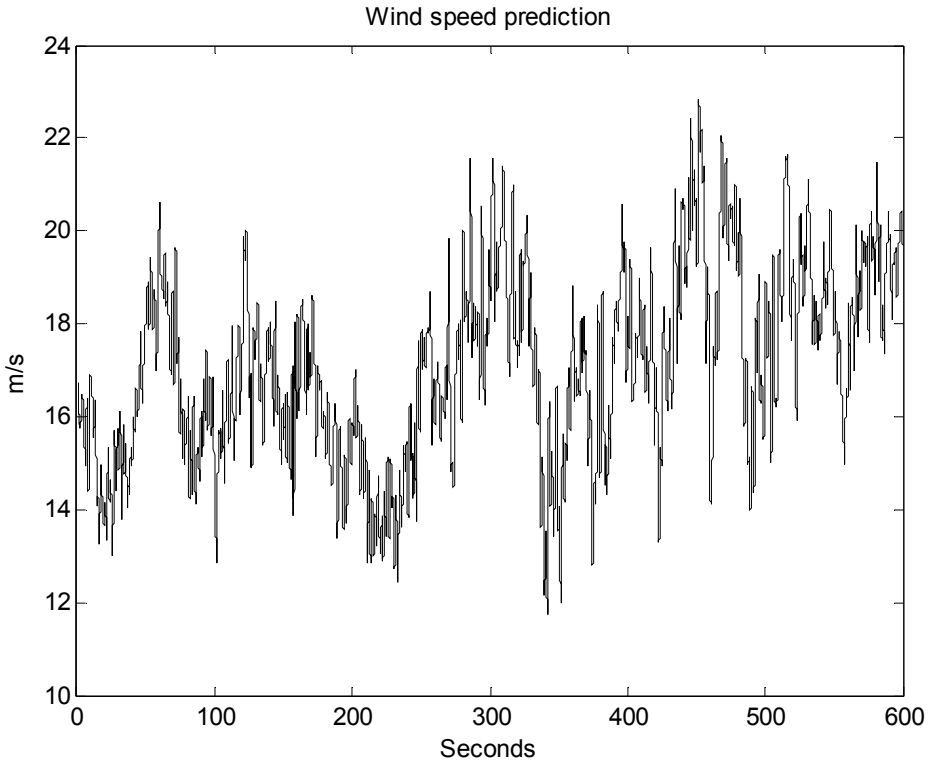


Figure 60 Extracted long lengthscale component of the wind speed data.

The wind speed data consists of the low frequency point measurement wind field taken from the nacelle anemometer reading. Even after applying Gaussian process, the significant turbulence intensity observed still results in a poor representation of the data obtained. Therefore, an additional step to spatially filter the data is essential. This will result in a small delay on the filtered wind speed, which can be compensated by incrementing the data by an equivalent time of 0.24 seconds. The eventual data, as shown in Figure 61, is known as the effective wind speed.

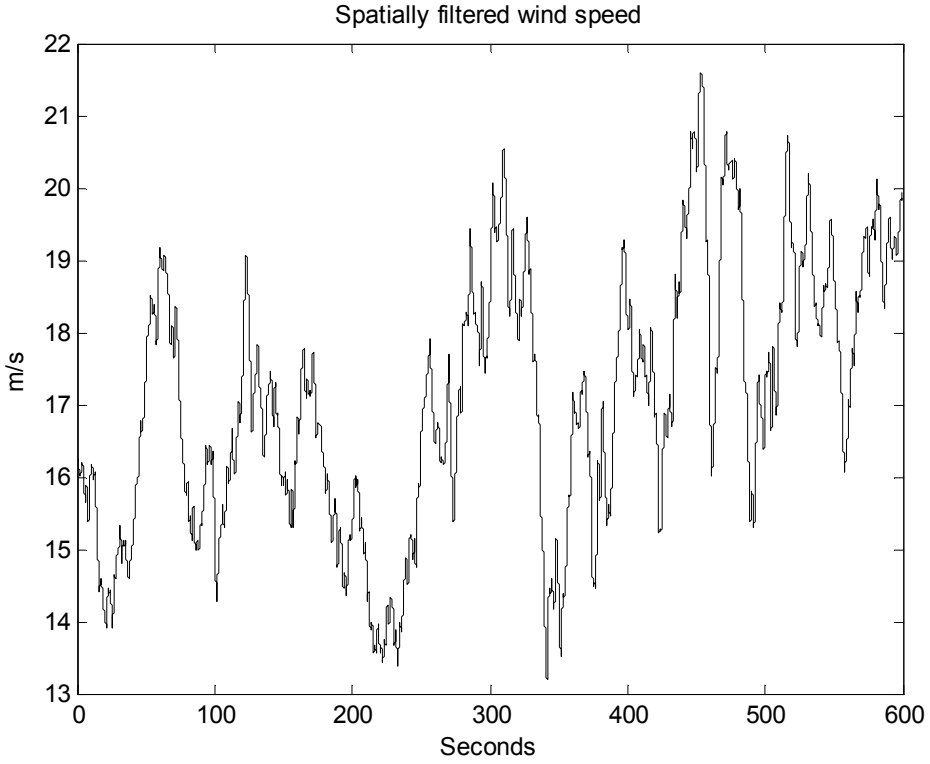


Figure 61 Spatially filtered wind speed data.

Blade Pitch Angle

The extraction of the long lengthscale component from the raw blade pitch angle data is straightforward. Model with two Gaussian processes is applied on the data, with the covariance function similar to that used for the wind speed data.

$$\mathrm{E}\left[y_i, y_j\right] = a_f \exp\left[-\frac{d_f}{2}\left(t_i - t_j\right)^2\right] + a_g \exp\left[-\frac{d_g}{2}\left(t_i - t_j\right)^2\right] + b\delta_{ij}$$

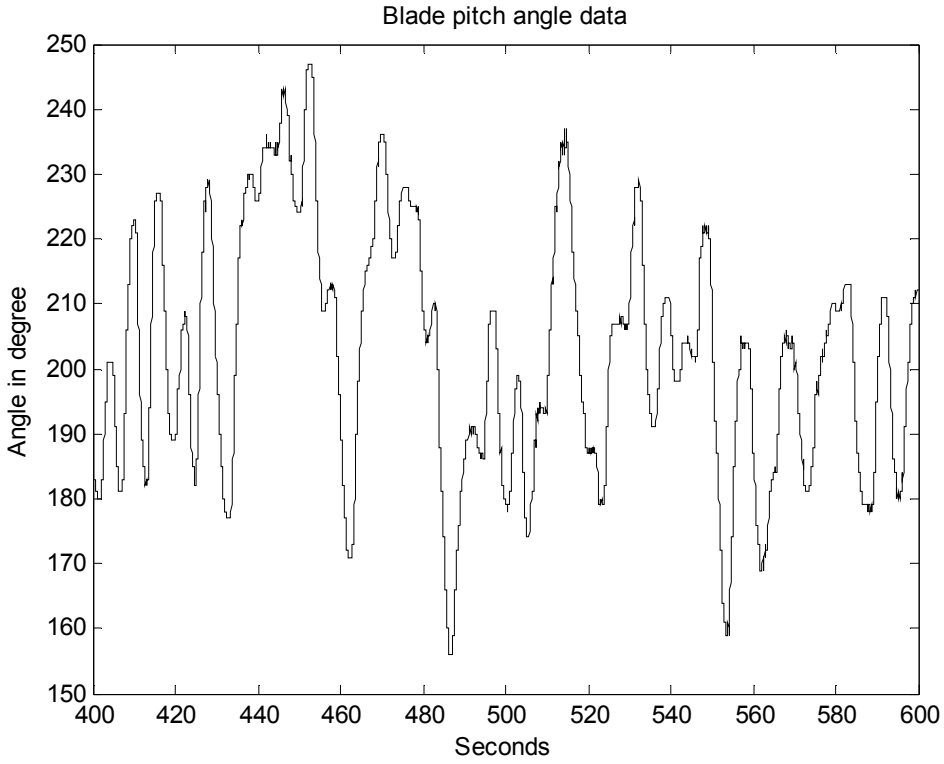


Figure 62 Noisy measurement blade pitch angle data.

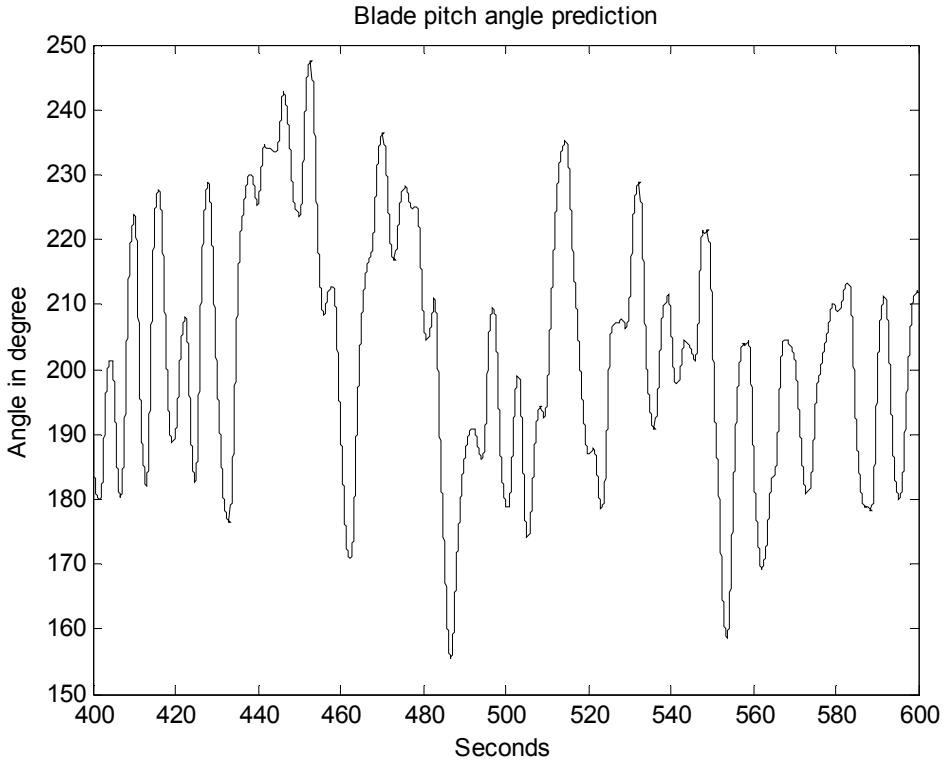


Figure 63 Extracted long lengthscale component of the blade pitch angle data.

A section of the last 200 seconds of the noisy measurement data is shown in Figure 62. Unlike the wind speed and rotor speed data, the short lengthscale component of the blade pitch angle is not obvious. This can be seen from the power spectrum density as shown in Figure 53.

The values of the hyperparameters, adapted to maximise the log-likelihood function, are found to be $a_f = 1.1\text{x}10^5$, $d_f = 0.5$, $a_g = 0.23$, $d_g = 65.3$ and $b = 0.09$. Notice that the prediction for the long lengthscale component, in Figure 63, is now missing those kinks and looks much smoother.

With successful extraction of the required long lengthscale components of the rotor speed (including rotor acceleration), wind speed and blade pitch angle from the raw data, it allows the relationship between the aerodynamic torque and the associated drive-train dynamics to be formulated. This is carried out in the second phase of the identification procedure.

5.3.3 Nonlinear Dynamics Identification (Phase 2)

This section covers the identification of the aerodynamics and drive-train dynamics of the wind turbine machine using Gaussian process prior models. The dynamics is known (Leithead et al, 1999) to consist of a nonlinear component (aerodynamics) and a linear component (drive-train dynamics). It is also known (Leithead et al., 1999; Leithead et al., 2003a) that the aerodynamic torque is considered to be an algebraic function of wind speed, rotor speed and the blade pitch angle, i.e.

$$\mathrm{T_A} = \mathrm{H}\left(\omega_\mathrm{R}, \beta, v\right)$$

where ω_R is the rotor speed, β is the pitch angle and v is the wind speed variable. Furthermore, the aerodynamic torque is proportional to the rotor acceleration $\dot{\omega}_\mathrm{R}$ as given by the equation below.

$$\mathrm{T_A} - \mathrm{T_0} \approx \mathrm{J}\dot{\omega}_\mathrm{R}$$

where $\mathrm{T_0}$ is the generator torque set-point scaled by the gearbox ratio and J is the rotor inertia. This event is noticed at low frequency of the aerodynamic torque. However, it

is not the identification of the aerodynamic torque per se that is of interest, the requirement here is to extract and identify components belonging to the dataset, $\{(\omega_R, \beta, v), \dot{\omega}_R\}$ and prove that the torque is a sum of two independent functions.

Due to the filtering (extraction of long lengthscale components) by Gaussian regression in Phase 1, the data values are now correlated over a time interval of about 1 second. The second phase of the identification procedure requires the filtered data to be sampled further. Instead of using all the 24,000 data points from each extracted set, only a subset of 1,599 data points from each dataset are chosen at random. These sampled data points are obtained by considering every alternate 15 measurements. The purpose of doing so is to ensure that neighbouring data values are no longer correlated. The difference between the effective wind speed and the wind speed, uniform over the rotor disc, that reproduces the actual aerodynamic torque may be considered as further additive disturbance on the former. With the selection of new datasets, this additive disturbance becomes randomised. Any random additive disturbance on the wind speed can be treated as noise on the "measured" aerodynamic torque; thereby preserving the regression formulation. The intensity of the noise on the "measured" aerodynamic torque includes a contribution from this source. Hence, it is known (Leithead et al., 1999) that, during above rated operation, the aerodynamics can be separated into the sum of two functions, the first being a function of the wind speed and the second a function of the rotor speed and the blade pitch angle; that is,

$$\hat{T}_A(\omega_R, \beta, v) = \Phi(\omega_R, \beta) - \Gamma(v) \tag{81}$$

It is also known that the function of the wind speed is almost linear (Leithead et al, 1999) and might possibly be approximated by a quadratic function.

To confirm the separation in equation (81) for the wind turbine, the aerodynamic torque is modelled as the sum of two independent Gaussian processes rather than a single Gaussian process. The first, Φ_z, is a Gaussian process with explanatory variable belonging to the rotor speed and the blade pitch angle, with the covariance function as given in (82).

$$C_{\Phi}\left(\mathbf{z}_i,\mathbf{z}_j\right)=a_{\Phi}\,\exp\left\{-\frac{1}{2}\sum_{k=1}^{2}d_{\Phi_k}\left[\left(\mathbf{z}_i\right)_k-\left(\mathbf{z}_j\right)_k\right]^2\right\};\ \mathbf{z}=\{\omega_{\mathrm{R}},\beta\} \tag{82}$$

The second, Γ_{v}, is a Gaussian process with wind speed as the explanatory variable and covariance function given by (83).

$$C_{\Gamma}\left(v_i,v_j\right)=a_{\Gamma}\,\exp\left\{-\frac{1}{2}d_{\Gamma}\left(v_i-v_j\right)^2\right\} \tag{83}$$

Since the two Gaussian processes are independent, their sum is also a Gaussian process with the covariance function

$$C_{\mathrm{H}}\left(\hat{\mathrm{T}}_{\mathrm{A}_i},\hat{\mathrm{T}}_{\mathrm{A}_j}\right)=a_{\Phi}\,\exp\left\{-\frac{1}{2}\sum_{k=1}^{2}d_{\Phi_k}\left[\left(\mathbf{z}_i\right)_k-\left(\mathbf{z}_j\right)_k\right]^2\right\}+a_{\Gamma}\,\exp\left\{-\frac{1}{2}d_{\Gamma}\left(v_i-v_j\right)^2\right\}$$

The covariance at measurement y_i, is

$$C\left(y_i,y_j\right)=C_{\mathrm{H}}\left(\hat{\mathrm{T}}_{\mathrm{A}_i},\hat{\mathrm{T}}_{\mathrm{A}_j}\right)+b\delta_{ij} \tag{84}$$

where δ_{ij} is the Kronecker delta. The hyperparameters $\left\{a_{\Phi},a_{\Gamma},d_{\Phi},d_{\Gamma},b\right\}$ are then adapted to maximise the likelihood of the data. Gaussian regression using model, with two independent stochastic processes and independent stochastic variables, is applied to the reduced dataset. The optimised values of the hyperparameters are a_{Φ} = 2.66, a_{Γ} = 1.02, $d_{\Phi}^{\omega_{\mathrm{R}}}=1.1\times10^{-3}$, $d_{\Phi}^{\beta}=4.5\times10^{-5}$, $d_{\Gamma}=9.9\times10^{-3}$ and b = 0.0978. The components of the aerodynamics torques, namely $\Phi\left(\omega_R,\beta\right)=\mathrm{E}\left[\Phi_{\mathrm{z}}\right]$ and $\Gamma\left(v\right)=\mathrm{E}\left[\Gamma_{v}\right]$, are depicted in Figure 64 and Figure 65, respectively.

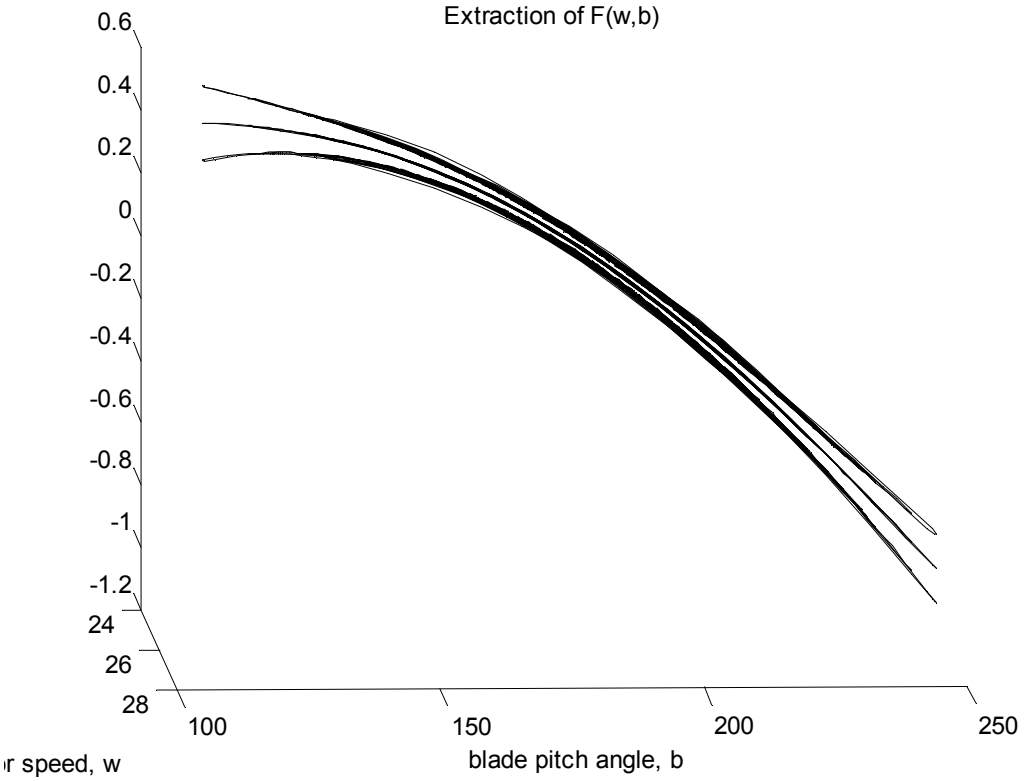


Figure 64 Prediction and confidence intervals of aerodynamic component using squared exponential covariance function.

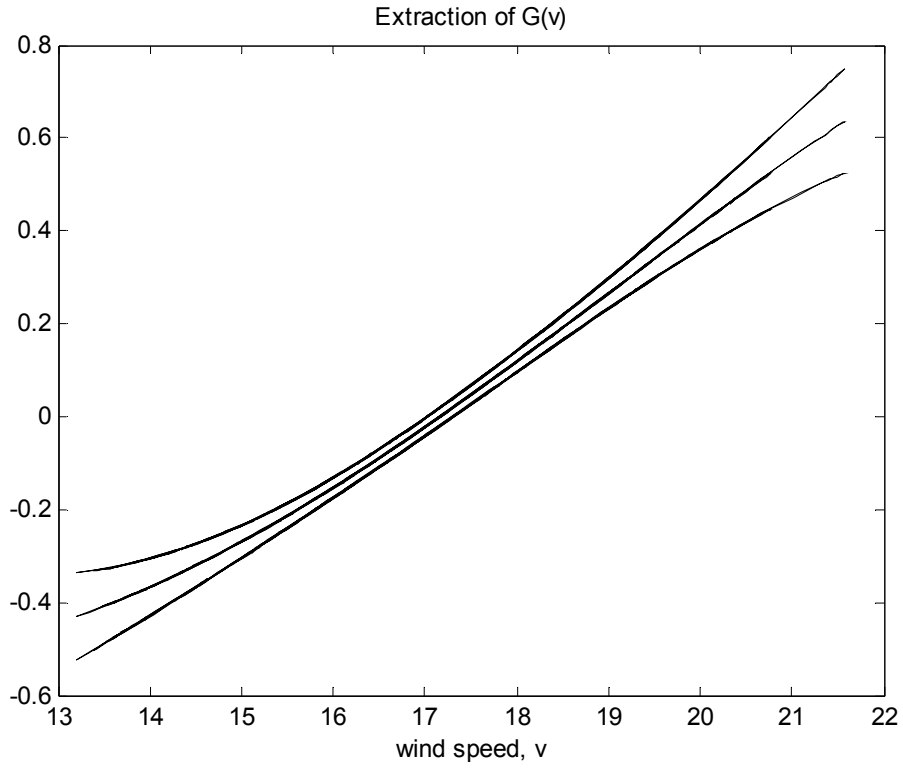


Figure 65 Prediction and confidence intervals of drive-train component using squared exponential covariance function.

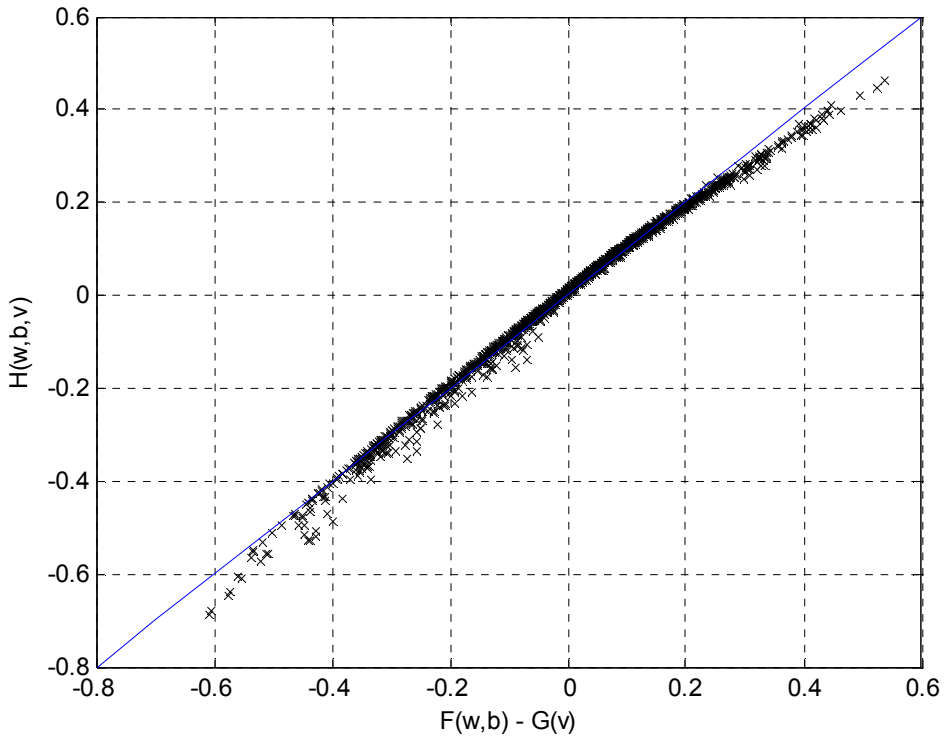


Figure 66 Confirmation of the separation of the aerodynamic torque. The Gaussian process is modelled using squared exponential covariance functions.

Due to the transformation applied to the posterior joint probability distribution, it ensures that equal likelihood can be attributed to either Φ_z or Γ_v. Thus, the confidence intervals for the two predictions are rather narrow. The values for $\hat{T}_A(\omega_R, \beta, v)$ and $T_A(\omega_R, \beta, v)$ at the measured values of (ω_R, β, v) are compared. Figure 66 confirms the separation for the aerodynamic torque into two additive components, as it can be seen that the data points ('x') are lying close to the line of slope passing through the origin.

From the prediction of the drive-train dynamic depicted in Figure 65, it seems that the component is almost linear. A possible representation of the drive-train component is by means of a linear covariance function.

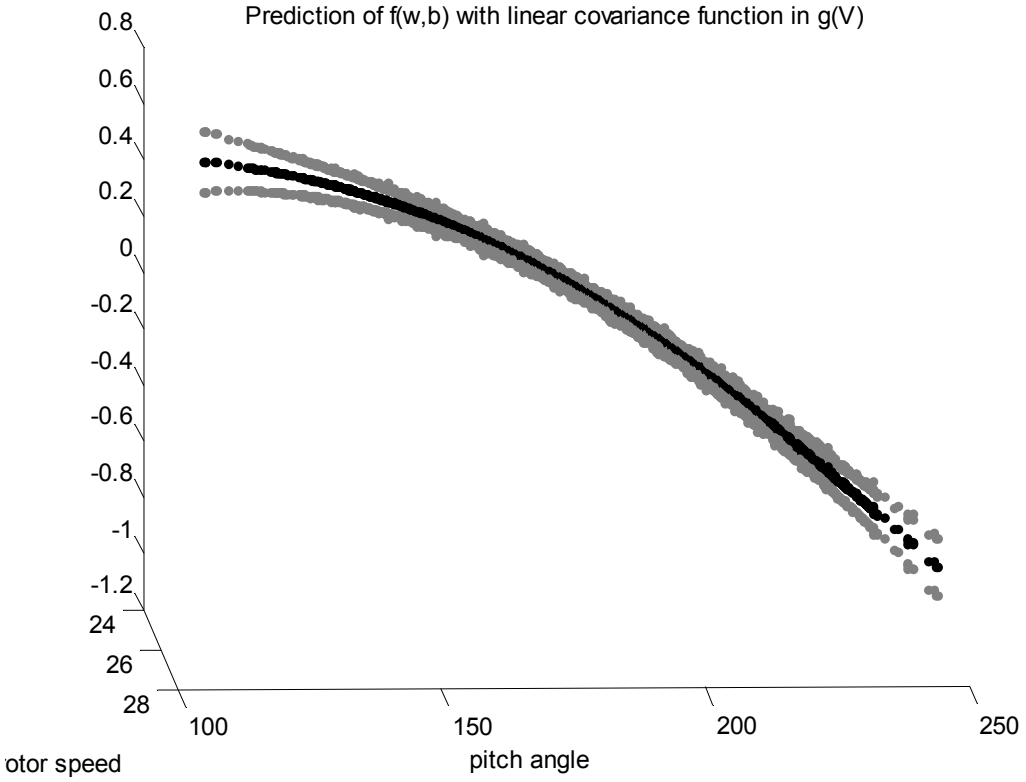


Figure 67 Prediction and confidence intervals of the aerodynamics component with Γ_v using a linear covariance function.

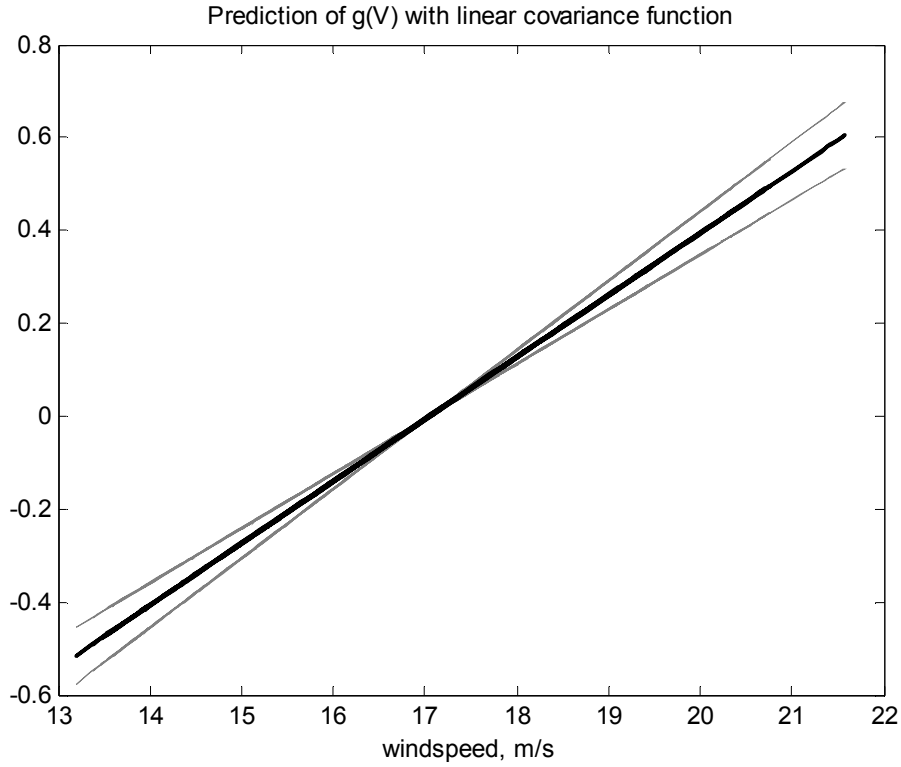


Figure 68 Prediction and confidence intervals of the drive-train dynamic with Γ_v using a linear covariance function.

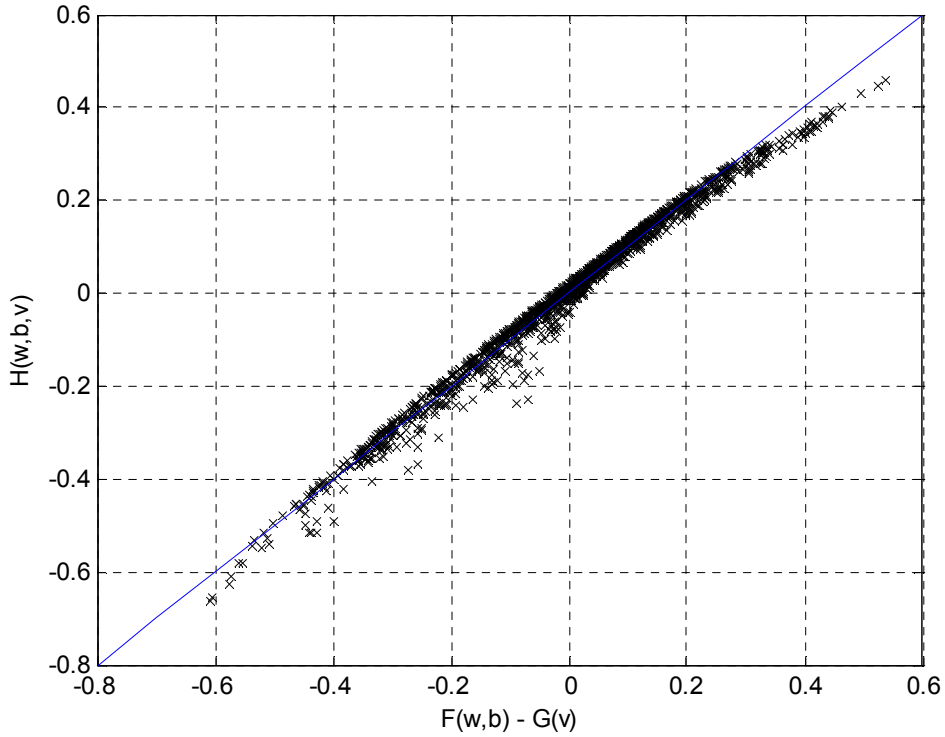


Figure 69 Confirmation of the separation of the aerodynamic torque when Γ_v uses a linear covariance function

Modelling the aerodynamic torque as the sum of two independent Gaussian processes, the first being Φ_z characterised by the same covariance function as before. The second stochastic process is Γ_v, with wind speed as the explanatory variable and covariance function given by (85).

$$C_\Gamma\left(v_i, v_j\right) = w_\Gamma v_i v_j \tag{85}$$

The covariance function C_Γ has a linear feature and is of a non-stationary type. The covariance for the measurements y_i is (84), except with C_H which is to be substituted by $\overline{C}_\mathrm{H}$, i.e.

$$\overline{C}_\mathrm{H}\left(\hat{\mathrm{T}}_{\mathrm{A}_i}, \hat{\mathrm{T}}_{\mathrm{A}_j}\right) = a_\Phi \exp\left\{-\frac{1}{2}\sum_{k=1}^{2} d_{\Phi_k}\left[\left(\mathbf{z}_i\right)_k - \left(\mathbf{z}_j\right)_k\right]^2\right\} + w_\Gamma v_i v_j$$

The set of hyperparameters to be trained is $\left\{a_\Phi, d_\Phi, w_\Gamma, b\right\}$. The predictions and two times standard deviations of the nonlinear component of the aerodynamic torque, $\Phi\left(\omega_R, \beta\right) = \mathrm{E}\left[\Phi_z\right]$, and the linear component of the drive-train dynamic,

$\Gamma(v) = \mathrm{E}[\Gamma_v]$, are shown in Figure 67 and Figure 68, respectively. The adapted values of the hyperparameters are a_Φ = 14.7357, $d_\Phi^{\omega_R} = 2.3\times10^{-4}$, $d_\Phi^{\beta} = 2.084\times10^{-5}$, $w_\Gamma = 0.018$ and b = 0.098. Again, the confidence intervals of the joint probability distribution of the posteriors are narrow within the operating region. The separation of the aerodynamic torque into two additive components is confirmed by Figure 69, illustrating the values of $\hat{\mathrm{T}}_\mathrm{A}(\omega_\mathrm{R},\beta,v)$ and $\mathrm{T}_\mathrm{A}(\omega_\mathrm{R},\beta,v)$ at the measured values of $(\omega_\mathrm{R},\beta,v)$. The agreement is quite good with all the data points ('x') lying close to the slope of the line passing through the origin. Comparing Figure 69 with Figure 66, there is no noticeable distinctive difference of which result is better. The comparison simply shows that the drive-train dynamic has some linear relationship with respect to the wind speed.

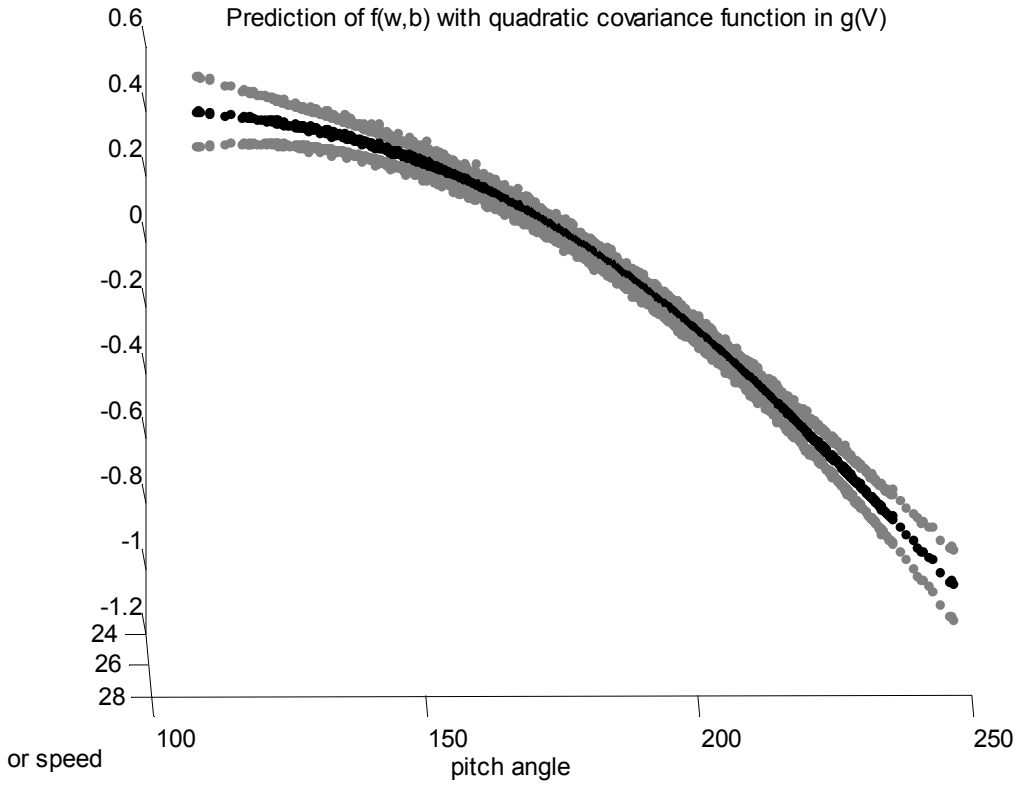


Figure 70 Prediction and confidence intervals of the aerodynamic component with Γ_v using a quadratic covariance function.

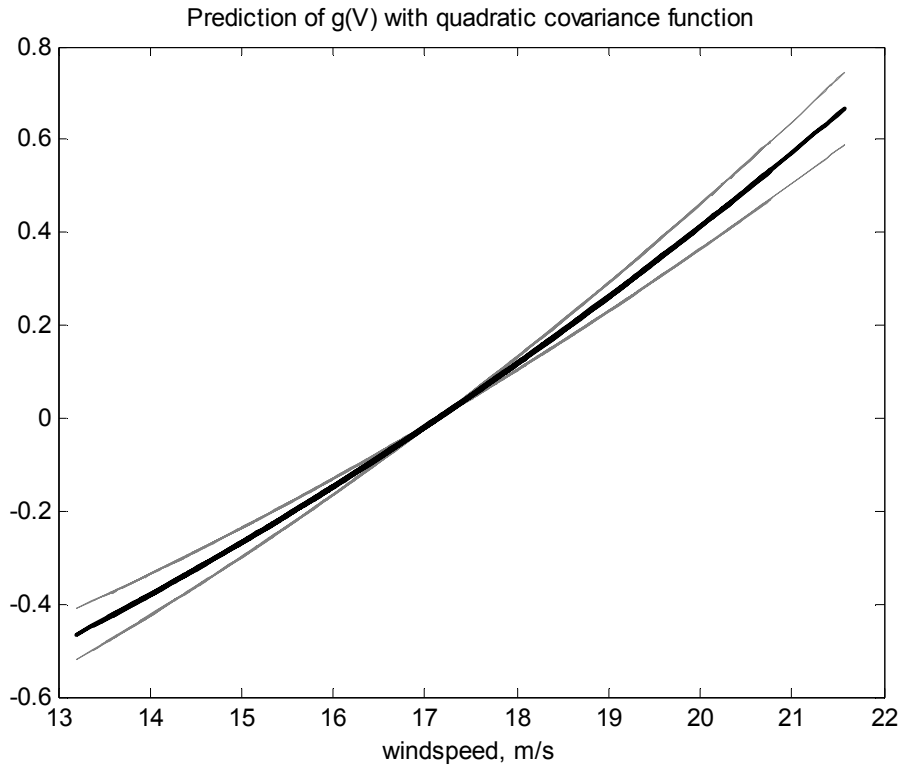


Figure 71 Prediction and confidence intervals of the drive-train aerodynamic component with Γ_v using a quadratic covariance function.

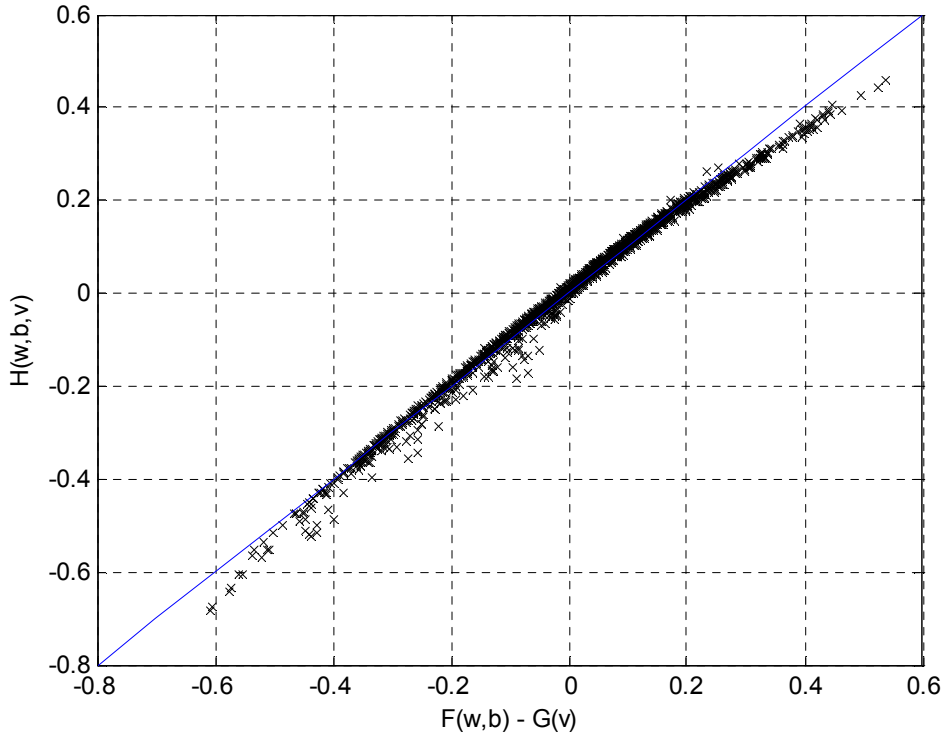


Figure 72 Confirmation of separation of aerodynamic torque when Γ_v uses a quadratic covariance function.

Besides linear analysis of the drive-train dynamic, it might perhaps be possible to model this component by means of a quadratic covariance function. A more detailed analysis of using a quadratic relationship of the drive-train dynamic with respect to the wind speed is discussed in the next sub-section. Here, the aerodynamic torque is modelled as the sum of two independent Gaussian processes, the first, Φ_z, with explanatory variable $\{\omega_R,\beta\}$ characterised by the same squared exponential covariance function, and the second Γ_v, with wind speed as explanatory variable, modelled by a quadratic covariance function defined in (86).

$$C_\Gamma\left(v_i, v_j\right) = w_\Gamma \left(v_i\right)^2 \left(v_j\right)^2 \tag{86}$$

The covariance for the measurements y_i is (84), except with C_H being substituted by $\widetilde{C}_H$, i.e.

$$\widetilde{C}_H\left(\hat{T}_{A_i}, \hat{T}_{A_j}\right) = a_\Phi \exp\left\{-\frac{1}{2}\sum_{k=1}^{2} d_{\Phi_k}\left[\left(\mathbf{z}_i\right)_k - \left(\mathbf{z}_j\right)_k\right]^2\right\} + w_\Gamma \left(v_i\right)^2 \left(v_j\right)^2$$

The set of hyperparameter values obtained by maximising the log-likelihood function are $a_\Phi = 2.51$, $d_\Phi^{\omega_R} = 1.3 \times 10^{-3}$, $d_\Phi^{\beta} = 5.05 \times 10^{-5}$, $w_\Gamma = 1.515 \times 10^{-5}$ and $b = 0.0979$. Similar to previous two models, the predictions and confidence intervals for Φ_z and Γ_v are depicted in Figure 70 and Figure 71, respectively. The separation of the aerodynamic torque into two additive components is illustrated in Figure 72. Once again, the data points ('x') are located close the line that passes through the origin.

5.3.4 Quadratic Function Relationship with Wind Speed

The function of the aerodynamic torque dependent on the wind speed is likely to possess a quadratic relationship. Given that the aerodynamic torque can be separated into two additive functions as shown in (81), an attempt to model the function $\Gamma(v)$ by a non-stationary quadratic covariance function is carried out. It is assumed that the aerodynamic torque has the following relationship,

$$\mathrm{T_A}(\omega_\mathrm{R}, \beta, v) = k v^2 \mathrm{X}(\lambda, \beta) \tag{87}$$

where k and $\mathrm{X}(\lambda,\beta)$ are constant terms and λ is the angular velocity in rad/s of the rotating wind turbine blades. This section focuses primarily on collapsing the domain from the one shown in (81) to

$$\widetilde{\mathrm{T}}_\mathrm{A}(\beta, v) = \Phi(\beta) - \Gamma(v) \tag{88}$$

by mathematically removing the rotor speed ω_R. The purpose of doing so is to allow a visual presentation of a three-dimensional plot, with the aerodynamic torque dependent on only two explanatory variables, i.e. β and v.

Given that the angular velocity remains constant above rated operation, it follows that

$$\lambda = \frac{\omega_\mathrm{R} R}{v} \cong \frac{\omega'_\mathrm{R} R}{v'}$$

where R is the radius of the rotor blade. Based on the equality,

$$v' = \left(\frac{\omega'_\mathrm{R}}{\omega_\mathrm{R}} \right) v \tag{89}$$

It is then appropriate to assume $\mathrm{T_A}(\omega_\mathrm{R}, \beta, v)$ is related to $\mathrm{T_A}(\omega'_\mathrm{R}, \beta', v')$ by different set of values of $(\omega_\mathrm{R}, \beta, v)$. Reformulating (87) by keeping λ and β unchanged, the following relationship is obtained.

$$\frac{\mathrm{T_A}}{v^2}(\omega_\mathrm{R}, \beta, v) \propto \mathrm{X}(\lambda, \beta)$$

Substituting (89) into the equation above, keeping β unchanged and introduce a different value of $\mathrm{T'_A}$ at ω_0 and v_0,

$$\Rightarrow \frac{\mathrm{T_A}}{v^2}(\omega_\mathrm{R}, \beta, v) = \frac{\mathrm{T'_A}}{v_0^2}(\omega_0, \beta, v_0)$$

$$\Rightarrow \mathrm{T_A}(\omega_\mathrm{R}, \beta, v) = \left(\frac{v}{v_0}\right)^2 \mathrm{T'_A}(\omega_0, \beta, v_0)$$

$$\Rightarrow \mathrm{T_A}(\omega_\mathrm{R}, \beta, v) = \left(\frac{\omega_\mathrm{R}}{\omega_0}\right)^2 \mathrm{T'_A}(\omega_0, \beta, v_0)$$

Hence, the ratio of the aerodynamic torques is equivalent to the ratio of the square of the corresponding rotor speeds. From earlier discussions, the aerodynamic torque is directly related to the rotor acceleration. Let v_0 = 17.063m/s be the mean wind speed and ω_0 = 25.9rad/s be the rated rotor speed. Thus, the following dataset $\{\dot{\omega}_\mathrm{R}, (\omega_R, \beta, v)\}$ is altered to be independent of the change in wind speed values, i.e.

$$\left\{\mathrm{T_{A,1}} \propto \dot{\omega}_1; (\omega_1, \beta_1, v_1)\right\} \rightarrow \left\{\dot{\omega}_1\left(\frac{\omega_0}{\omega_1}\right)^2; \left(\omega_1, \beta_1, v_0\left(\frac{\omega_1}{\omega_0}\right)\right)\right\}$$

$$\vdots$$

$$\left\{\mathrm{T_{A,}}_N \propto \dot{\omega}_N; (\omega_N, \beta_N, v_N)\right\} \rightarrow \left\{\dot{\omega}_N\left(\frac{\omega_0}{\omega_N}\right)^2; \left(\omega_N, \beta_N, v_0\left(\frac{\omega_N}{\omega_0}\right)\right)\right\}$$

The values for $y_n = \left\{\dot{\omega}_n\left(\frac{\omega_0}{\omega_n}\right)^2\right\}$, $\forall n = 1, ..., N$, are modelled to be the outcomes of the corresponding values for $\{\beta_n, v'_n\}_{n=1}^N$ where $v'_n = v_0\left(\frac{\omega_0}{\omega_n}\right)$, i.e.

$$f_n(\beta, v') = \Phi(\beta_n) - \Gamma(v'_n); \quad y_n = f_n + \varepsilon_n$$

where ε represents the noise in the dataset. With the formulation of this new relationship, it is apparent that the outcomes can be plotted on a three-dimensional surface as a function of β and v'. Furthermore, the individual functions of $\Phi(\beta)$ dependent on β, and $\Gamma(v)$ dependent on v can also be exploited and plotted on one-dimensional plots.

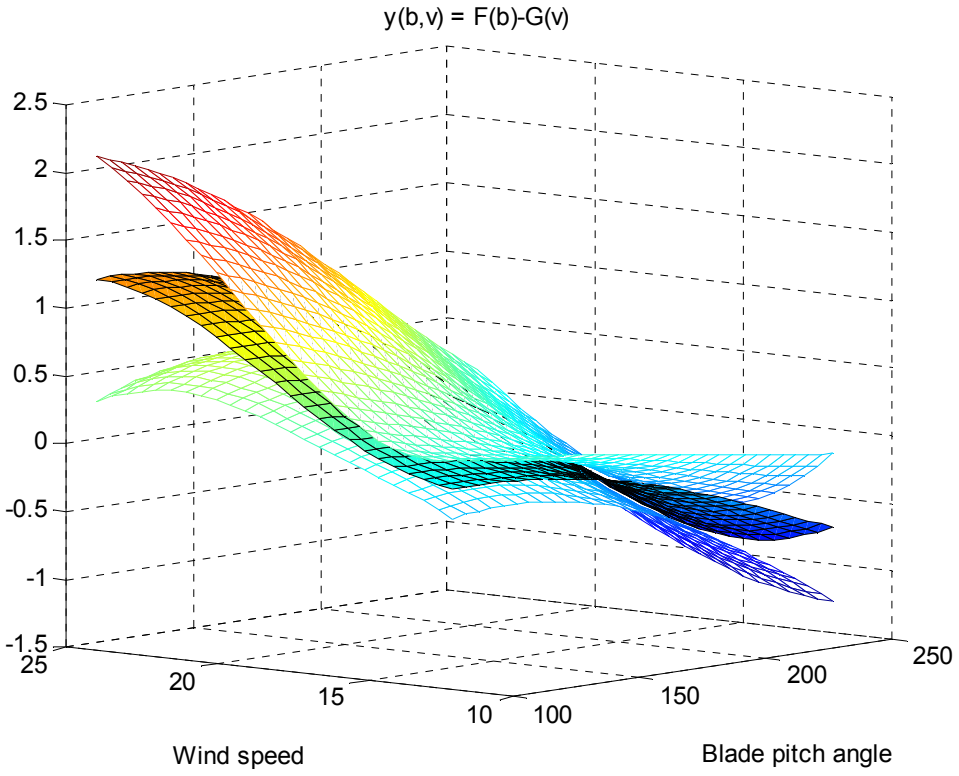


Figure 73 3D plot of scaled aerodynamic torque as a function of blade pitch angle and scaled wind speed.

Firstly, consider the data to be of the form $f_n(\beta, v') = \mathrm{H}(\beta_n, v'_n)$, instead of sum of two functions. Applying standard Gaussian regression with a single stochastic process using the squared exponential covariance function on the dataset, $\{y_n, (\beta_n, v'_n)\}_{n=1}^{N}$, the predictions and confidence intervals are as shown Figure 73. Applying a simple modification to the aerodynamic relationships with wind speed, rotor speed and blade pitch angle allows the domain to collapse from three dimensions into two dimensions. This improves visibility as to how the outcome is likely to vary with respect to changes in the values of the explanatory variable, i.e. β and v'.

However, it is not the scaled aerodynamic torque that is of interest. Instead, the goal is to extract and identify individual functions with different explanatory variables. Hence, a model with two Gaussian processes is implemented in the regression formulation, i.e. $f(\beta, v') = \Phi(\beta) - \Gamma(v')$. Three possible models are formulated; that is, the function $\Gamma(v')$ with respect to scaled wind speed is modelled by the squared exponential, linear and quadratic covariance functions. The use of the squared exponential covariance function allows $\Gamma(v')$ to be adapted with a nonlinear characteristic. In addition, the function of wind speed has shown to exhibit a linear trend and therefore modelling it with a linear covariance function is appropriate. Finally, the theoretical investigation discussed earlier in this section may suggest that $\Gamma(v')$ has a quadratic feature. Thus, a quadratic covariance function is also included in the exploration. The nonlinear aerodynamic component of $\Phi(\beta)$ is modelled using the squared exponential covariance function.

Squared Exponential Covariance Function

The squared exponential covariance function is commonly used in Gaussian regression to model nonlinear functions, which includes the class of all possible linear functions. The predictions and confidence intervals for $\Phi(\beta)$ and $\Gamma(v')$ based on Gaussian regression using model with two stochastic processes are depicted in Figure 74 and Figure 75, respectively. Notice how the drive-train dynamic in the latter figure vary with respect to wind speed. Near the centre of the range of the explanatory variable, between $v' = 16$ m/s to $v' = 20$ m/s, the trend is almost linear. Hence, there is a belief in which $\Gamma(v')$ might be a linear component.

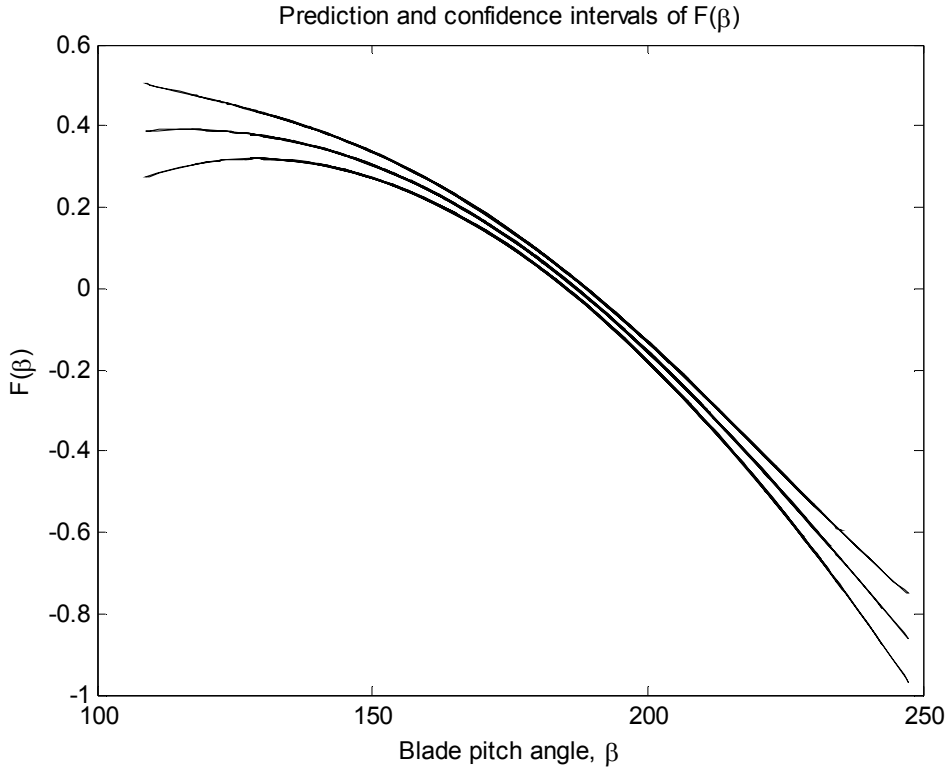


Figure 74 Prediction and confidence interval for $\Phi(\beta)$ with $\Gamma(v')$ using a squared exponential covariance function.

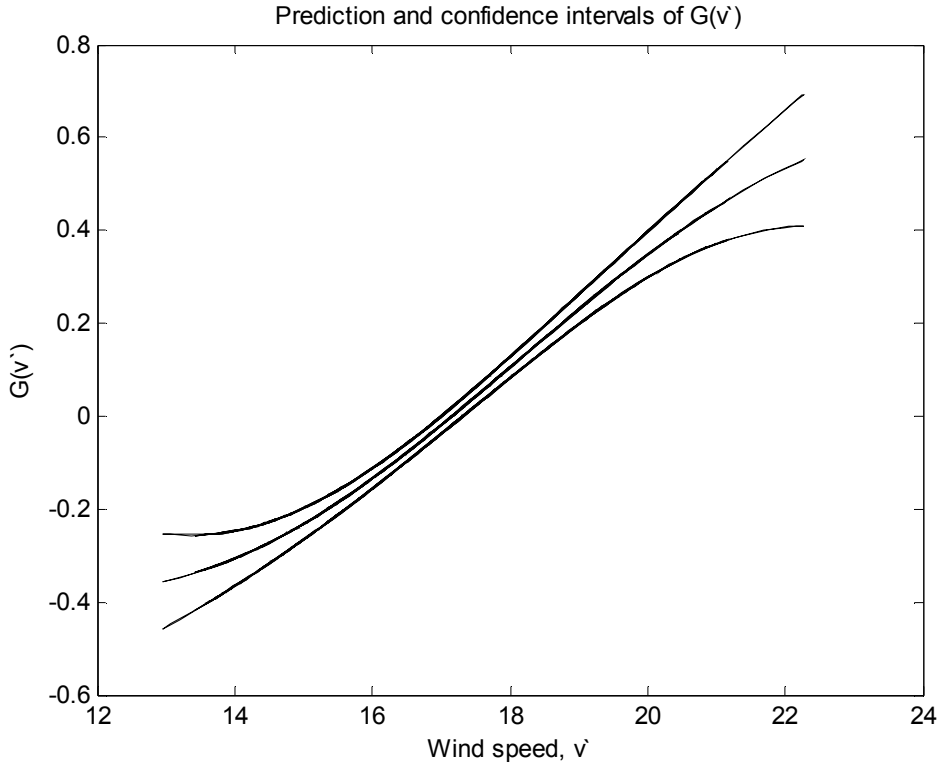


Figure 75 Prediction and confidence interval for $\Gamma(v')$, modelled using a squared exponential covariance function.

Linear Covariance Function

With the component $\Gamma(v')$ modelled by a linear covariance function, the predictions with confidence intervals for $\Phi(\beta)$ and $\Gamma(v')$ are shown in Figure 76 and Figure 77, respectively. The aerodynamic component for $\Phi(\beta)$ in Figure 76 is very similar to that in Figure 74. However, the drive-train dynamic, which is modelled by a linear covariance function, shows a firm straight line passing through zero at the mean wind speed value. The standard deviation is zero at the mean wind speed value and increases linearly towards both ends.

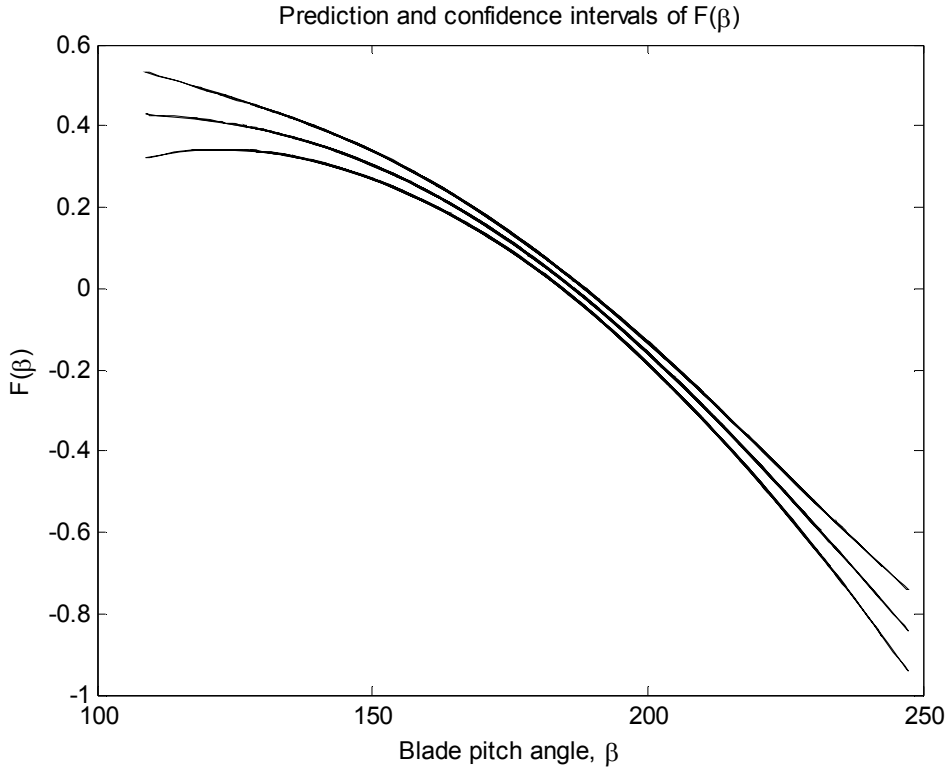


Figure 76 Prediction and confidence interval for $\Phi(\beta)$ with $\Gamma(\nu')$ using a linear covariance function.

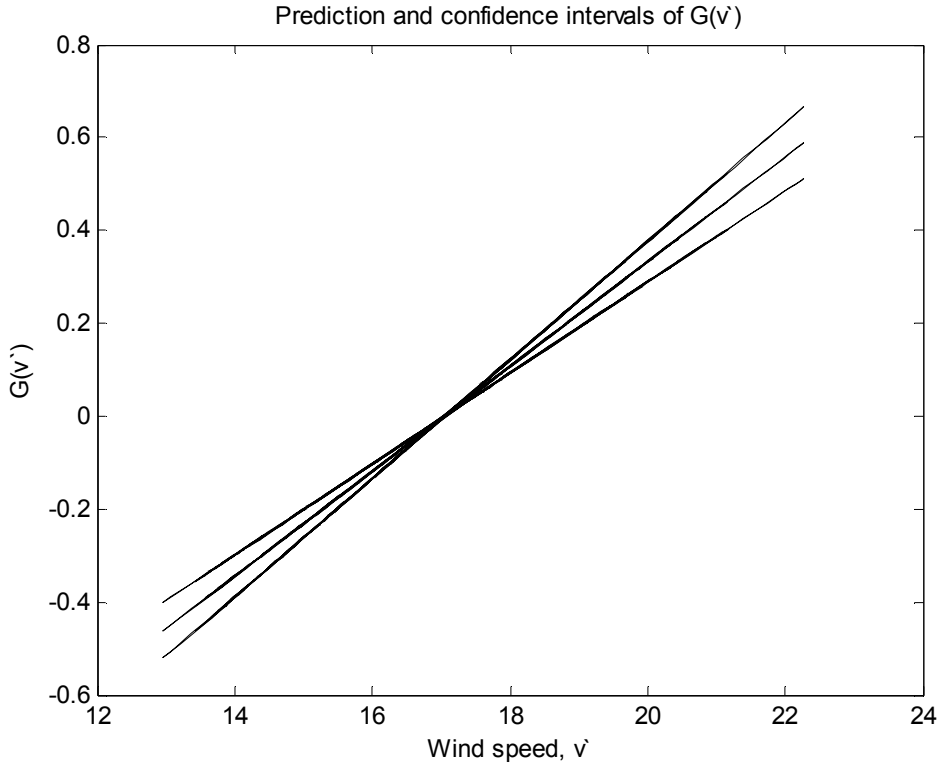


Figure 77 Prediction and confidence interval for $\Gamma(v')$, modelled using a linear covariance function.

Quadratic Covariance Function

From the theoretical explanation, it is known that the aerodynamic torque is a function of the square of the wind speed at above rated operation of the wind turbine. Hence, it is of interest to investigate $\Gamma(v')$ by representing the component with a quadratic covariance function. The predictions and two times standard deviations of $\Phi(\beta)$ and $\Gamma(v')$ are illustrated in Figure 78 and Figure 79, respectively. There is no apparent difference in the aerodynamic component, since it has been modelled by the same covariance function for all three cases. The component of the drive-train dynamic, however, illustrates a rather gentle quadratic feature with respect to wind speed. The standard deviation of the predictions for $\Gamma(v')$ also indicates a quadratic characteristic. It has a value of zero at the mean wind speed and increases away from that mean wind speed value.

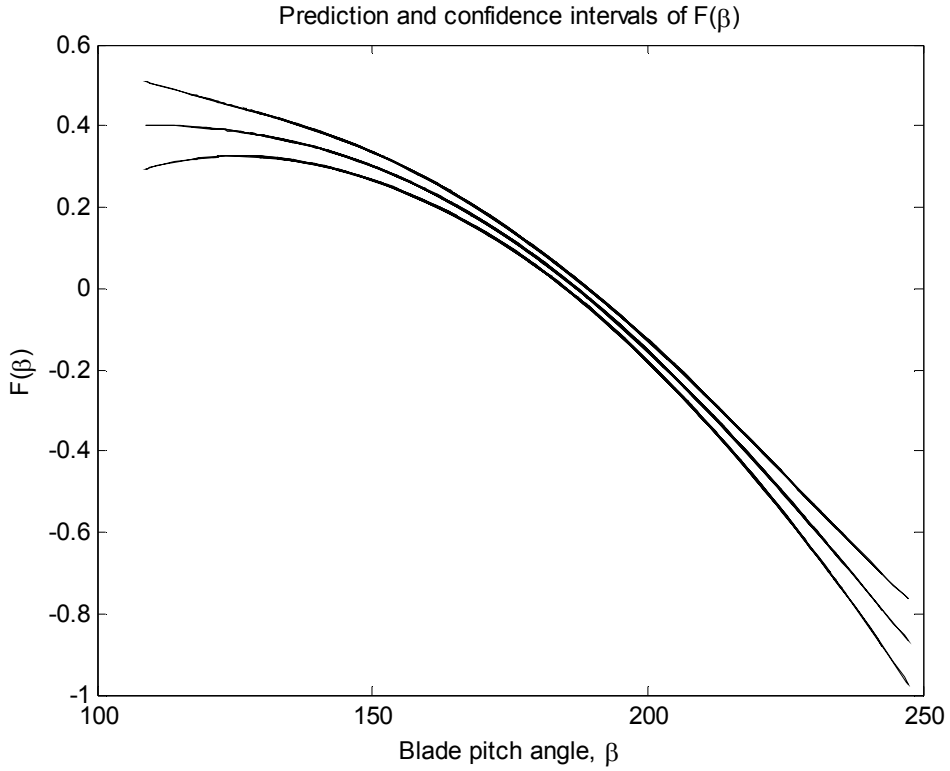


Figure 78 Prediction and confidence interval for $\Phi(\beta)$ with $\Gamma(v')$ using a quadratic covariance function.

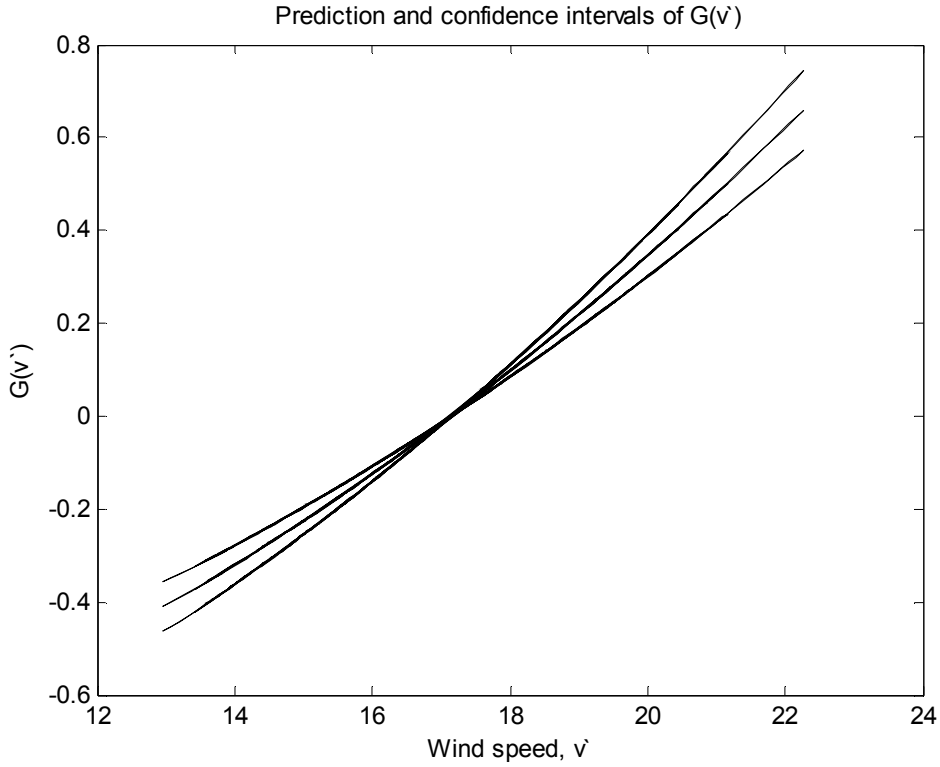


Figure 79 Prediction and confidence interval for $\Gamma(v')$, modelled using a quadratic covariance function.

There is very little to compare between the three cases as the error estimates between them are almost insignificant. However, their confidence intervals might provide more details about the right choice of the model. The standard deviations (1σ) of the prediction for $\Phi(\beta)$ and $\Gamma(v')$ are depicted in Figure 80 and Figure 81, respectively. There is little distinction to make in the former figure, whereas the latter figure shows are more defined illustration. However, it is not justifiable to focus mainly on Figure 81 since the choices of covariance functions are not the same for $\Gamma(v')$. Arguably, the only way to decide is by looking at the standard deviation for $\Phi(\beta)$. Hence, the conclusion that can be drawn is that any of the three covariance functions, i.e. squared exponential, linear and quadratic covariance functions, can be used to model the drive-train dynamics which is a function of wind speed.

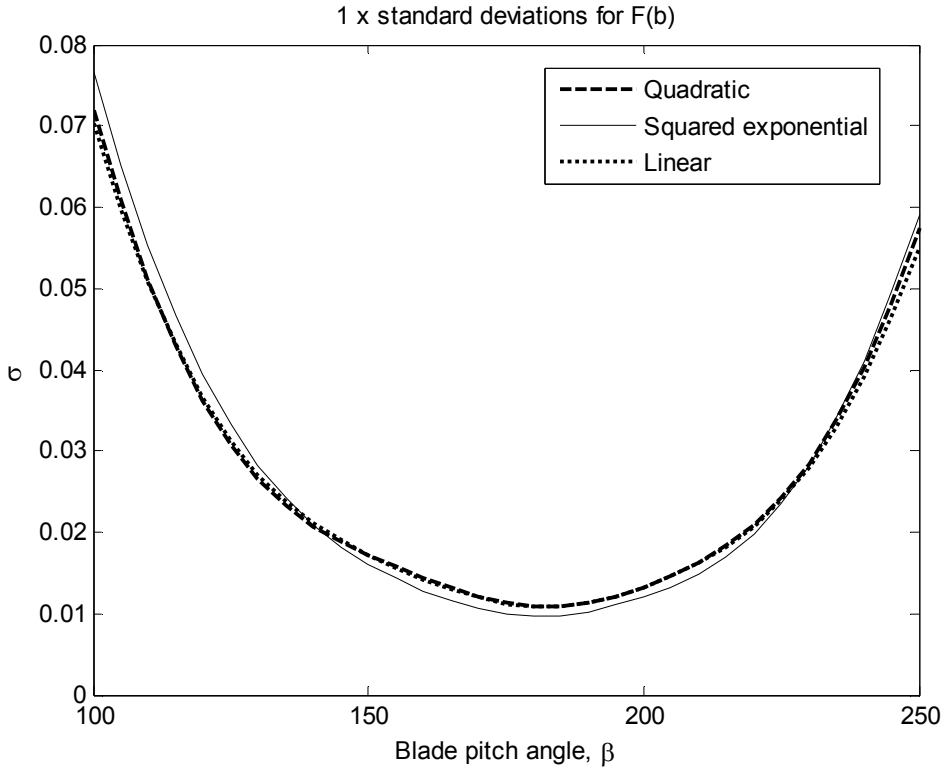


Figure 80 Standard deviations of $\Phi(\beta)$.

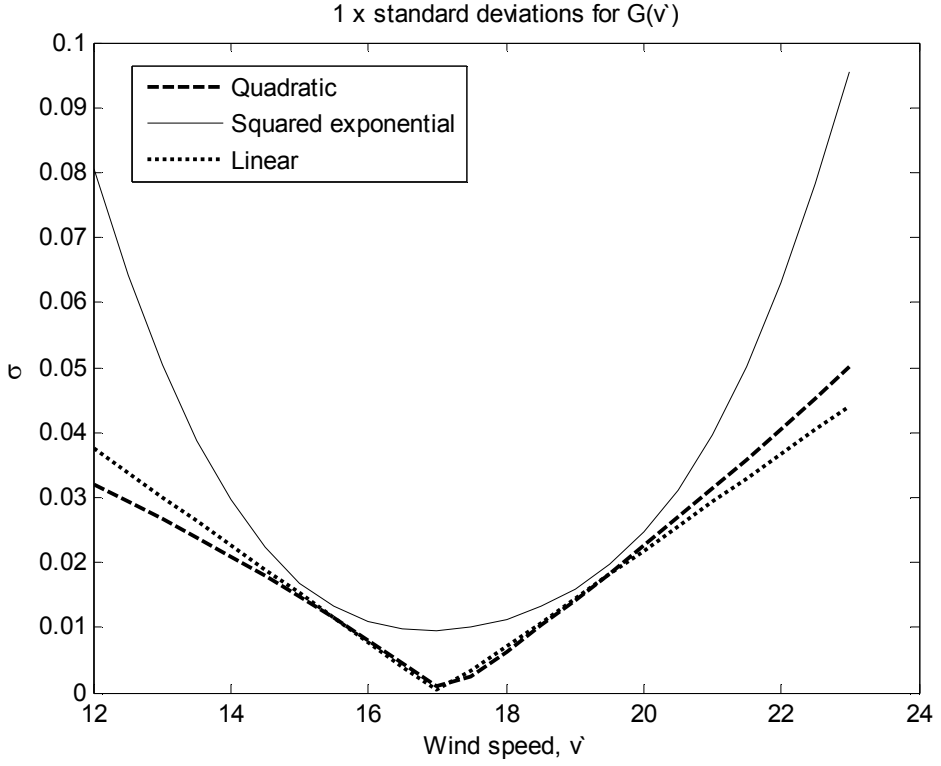


Figure 81 Standard deviations of $\Gamma(v')$.

5.3.5 Verification of Model by Numerical Integration

The experimental assumption that the aerodynamic torque comprises of two additive components is further investigated in this section. Previous analyses have shown good agreement between the values of $\mathrm{T_A}(\omega_\mathrm{R},\beta,v)=\mathrm{H}(\omega_\mathrm{R},\beta,v)$ and the values of $\hat{\mathrm{T}}_\mathrm{A}(\omega_\mathrm{R},\beta,v)=\Phi(\omega_\mathrm{R},\beta)-\Gamma(v)$, however it may not be suffice. This section investigates the separation by the use of numerical integration of the derivative component. If the assumption is true, the result should be close to the posterior joint probability distribution of the fit. To avoid further confusion, squared exponential covariance functions are used in the Gaussian process prior models wherever necessary. This allows the class of linear functions to be included in the stochastic modelling. The following steps are performed:

1. Obtain a Gaussian process fit of $\mathrm{H}(\omega_\mathrm{R},\beta,v)$ using a single Gaussian process, and a fit of $[\Phi(\omega_\mathrm{R},\beta)-\Gamma(v)]$ using the model with two Gaussian processes. The two fits are compared.
2. Obtain a set of operating points for $\{\beta_i,v_i\}$, such that $\{v_i\}\in[14,20]$. The values for $\{\beta_i\}$ are calculated given the values of $\{v_i\}$.
3. Use Gaussian process to compute the values for the derivative of $\partial\Gamma(v)/\partial v$, for all values of $\{v_i\}$.
4. Given the values of $\partial\Gamma(v)/\partial v$, integrate numerically to obtain $\Gamma(v)$ for all values of $\{v_i\}$.
5. Verify the fit of $\Phi(\omega_\mathrm{R},\beta)\equiv\mathrm{H}(\omega_\mathrm{R},\beta,v)+\Gamma(v)$, where Φ and H are computed using Gaussian processes.

Step 1

The predictions of the posterior joint probability distributions for $\mathrm{H}(\omega_\mathrm{R},\beta,v)$ and $[\Phi(\omega_\mathrm{R},\beta)-\Gamma(v)]$ are depicted in Figure 82 on time-series scale, since it is impossible to visualise a data with three-dimensional explanatory variable. The former is denoted by a black line, and the latter is shown using grey dots. Notice that the predictions are rather close to each other. The agreement of the separation looks reasonably good.

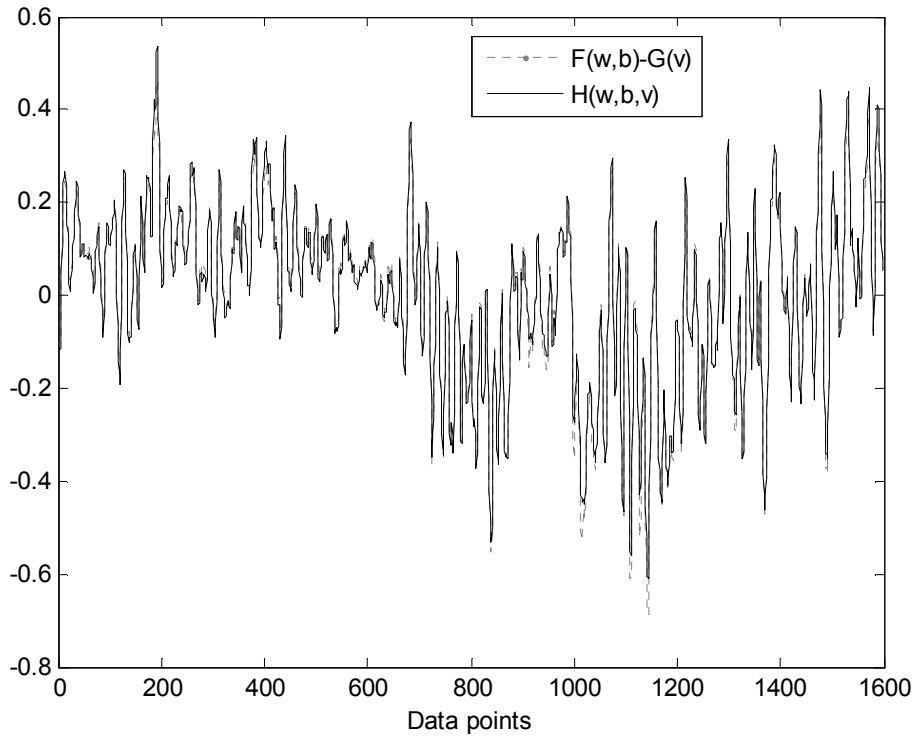


Figure 82 Predictions of the posterior joint probability distributions for $\mathrm{H}(\omega_{\mathrm{R}}, \beta, v)$ and $[\Phi(\omega_{\mathrm{R}}, \beta) - \Gamma(v)]$.

Step 2

The values for the wind speed, $\{v_i\} = \{14, 14.05, ..., 19.95, 20\}$, is obtained. Since numerical integration is involved at a later stage, a reasonable amount of data points is required. Next, ω_0 = 17.063m/s is defined to be the mean wind speed. Let the aerodynamic torque be written as

$$\mathrm{T}_0 = \mathrm{J}\dot{\omega}_0 = \mathrm{JH}(\omega_0, \beta, v)$$
$$\Rightarrow \mathrm{H}(\omega_0, \beta, v) = \frac{\mathrm{T}_0}{\mathrm{J}}$$

where J is the rotor inertia and $\dot{\omega}_0$ is the rotor acceleration. At the mean wind speed operation, $\frac{\mathrm{T}_0}{\mathrm{J}} \triangleq 0$. Thus, $\mathrm{H}(\omega_0, \beta, v) = 0$ for corresponding values of $\{\beta_i, v_i\}$. The values of $\{\beta_i\}$ are solved using any nonlinear solver for the zero-crossing of $\mathrm{H}(\omega_0, \beta, v)$. The values of the blade pitch angle β are plotted against the wind speed values v in Figure 83.

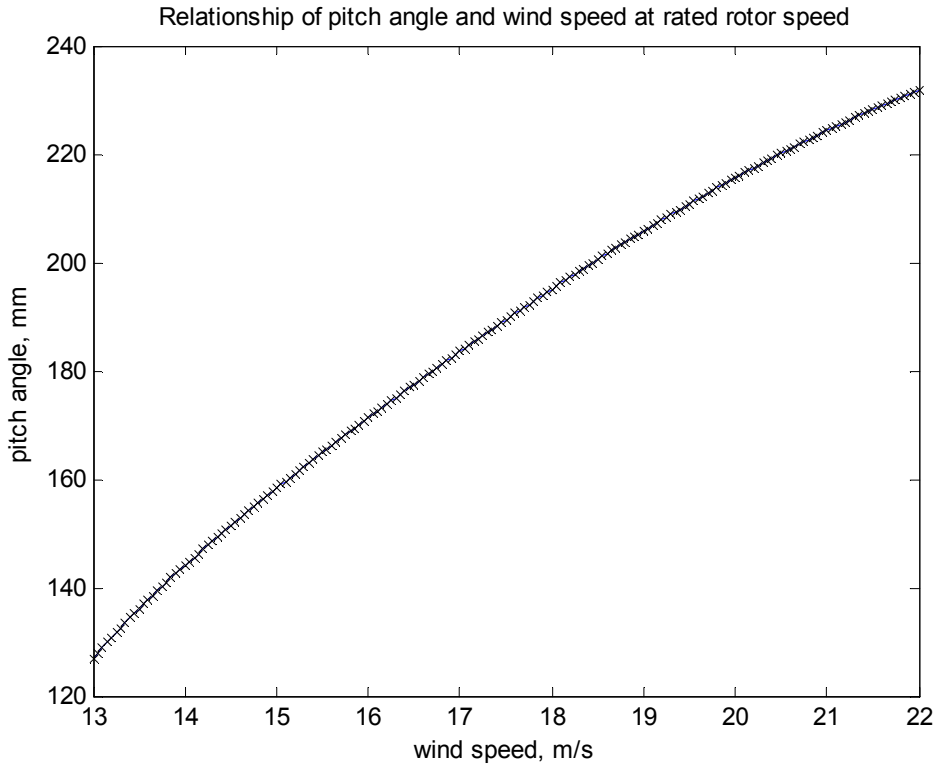


Figure 83 Plot of the blade pitch angle values against wind speed values.

Step 3

The information obtained in Step 2 is now used to determine the values for $\Phi(\omega_R, \beta)$ and $\Gamma(v)$. Assume that $H(\omega_R, \beta, v) = \Phi(\omega_R, \beta) - \Gamma(v)$ is true. Since for ω_0, v is related to β,

$$\therefore \frac{\partial H(\omega_0, \beta, v)}{\partial v} = \frac{\partial \Gamma(v)}{\partial v}$$

Gaussian process prior model is used to evaluate the values of $\partial H(v)/\partial v$ for all values of $\{v_i\}$ from Step 2. In addition, $\partial \Gamma(v)/\partial v$ is also calculated for comparison. The plot of the predictions for the partial derivatives with confidence intervals against wind speed is shown in Figure 84.

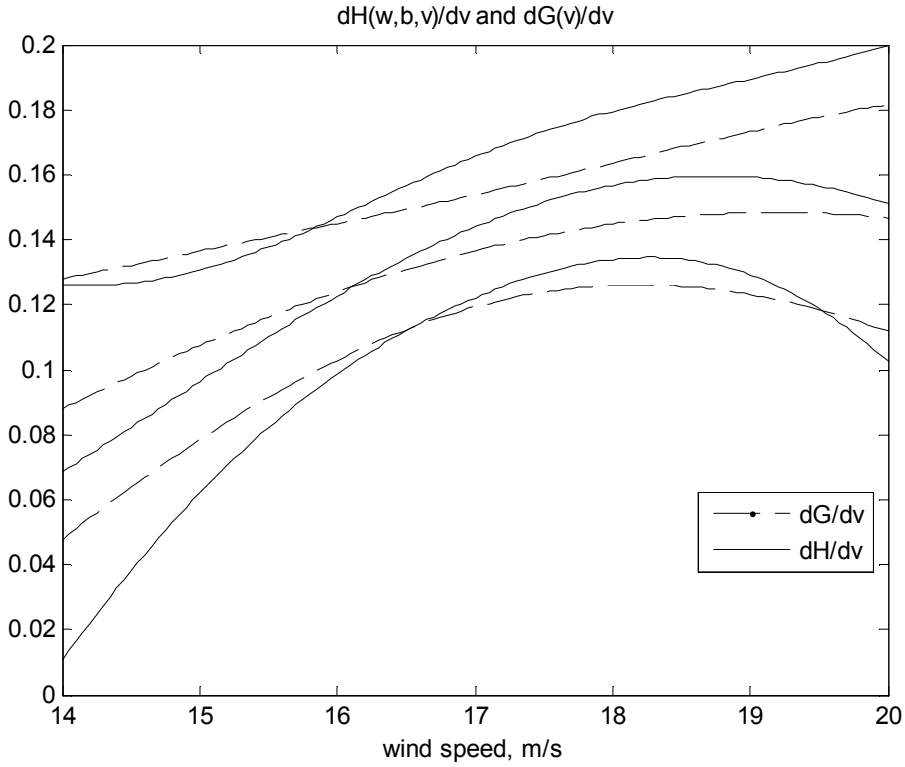


Figure 84 Plot of against $\partial \mathrm{H}(v)/\partial v$ and $\partial \Gamma(v)/\partial v$ wind speed.

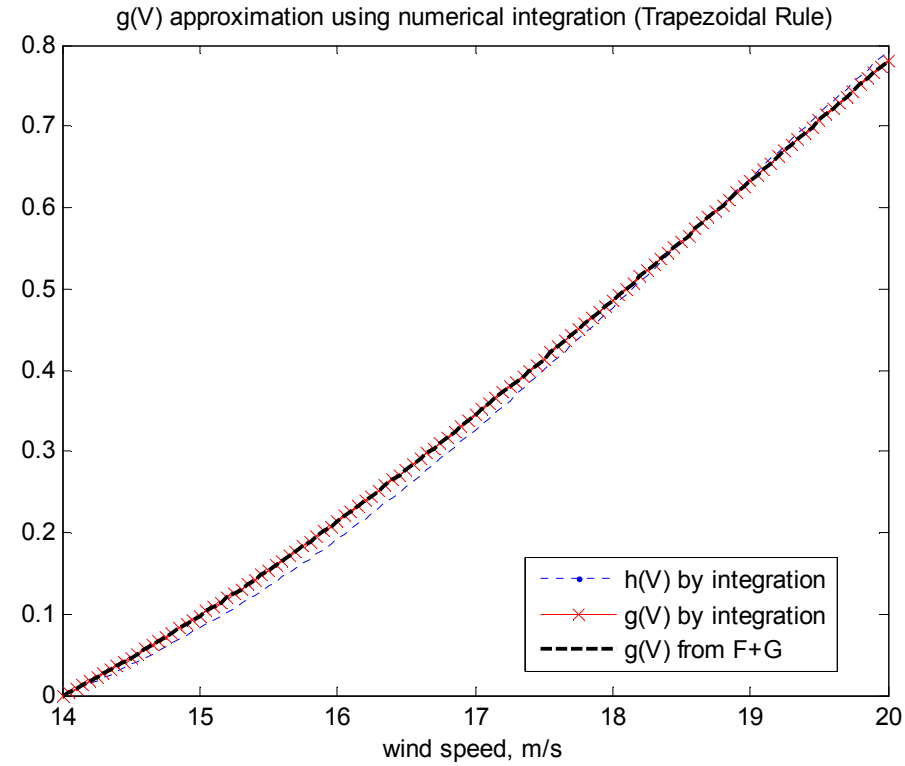


Figure 85 Approximations of $\Gamma(v)$ using numerical integration and prediction of $\Gamma(v)$ using Gaussian process prior models.

Step 4

Given the partial derivatives, $\partial \mathrm{H}(v)/\partial v$ and $\partial \Gamma(v)/\partial v$, obtained in Step 3, the data values are numerically integrated to compute the approximation of the values for $\Gamma(v)$.

$$\text{i.e. } \Gamma(v) \cong \int \frac{\partial \mathrm{H}}{\partial v} dv \text{ and } \Gamma(v) \cong \int \frac{\partial \Gamma}{\partial v} dv$$

Any numerical integration tool, i.e. Simpson's rule, can be used. The Trapezoid rule is chosen for this case. The results are shown in Figure 85. The approximation of $\Gamma(v)$ from $\partial \mathrm{H}(v)/\partial v$ is illustrated by dashed lines, whereas the approximation from $\partial \Gamma(v)/\partial v$ is depicted by crosses. The initial values of $\Gamma(v)$ obtained from model using two Gaussian processes are also shown (dots) in the figure. Apparently, the numerical integration works very well in integrating $\partial \Gamma(v)/\partial v$ as the values are very close together. The result from integrating $\partial \mathrm{H}(v)/\partial v$ is reasonably good, with very little deviation.

Step 5

The fits of $\Phi(\omega_{\mathrm{R}}, \beta) \equiv \mathrm{H}(\omega_{\mathrm{R}}, \beta, v) + \Gamma(v)$ is verified by using the approximated result from Step 4. The data values of $\Phi(\omega_{\mathrm{R}},\beta)$ is compared with the data values from $\mathrm{H}(\omega_{\mathrm{R}}, \beta, v) + \Gamma(v)$ and the comparison is shown in Figure 86. As it can be seen, the agreement is quite good.

From the result obtained, it is almost certain that the aerodynamic torque of the wind turbine machine is a function of two independent additive components; each dependent on a different set of explanatory variables. Thus it is safe to conclude that the aerodynamic torque, $\mathrm{T}_{\mathrm{A}}(\omega_{\mathrm{R}}, \beta, v) = \mathrm{H}(\omega_{\mathrm{R}}, \beta, v)$, consists of separable functions, i.e. $\hat{\mathrm{T}}_{\mathrm{A}}(\omega_{\mathrm{R}}, \beta, v) = \Phi(\omega_{\mathrm{R}}, \beta) - \Gamma(v)$.

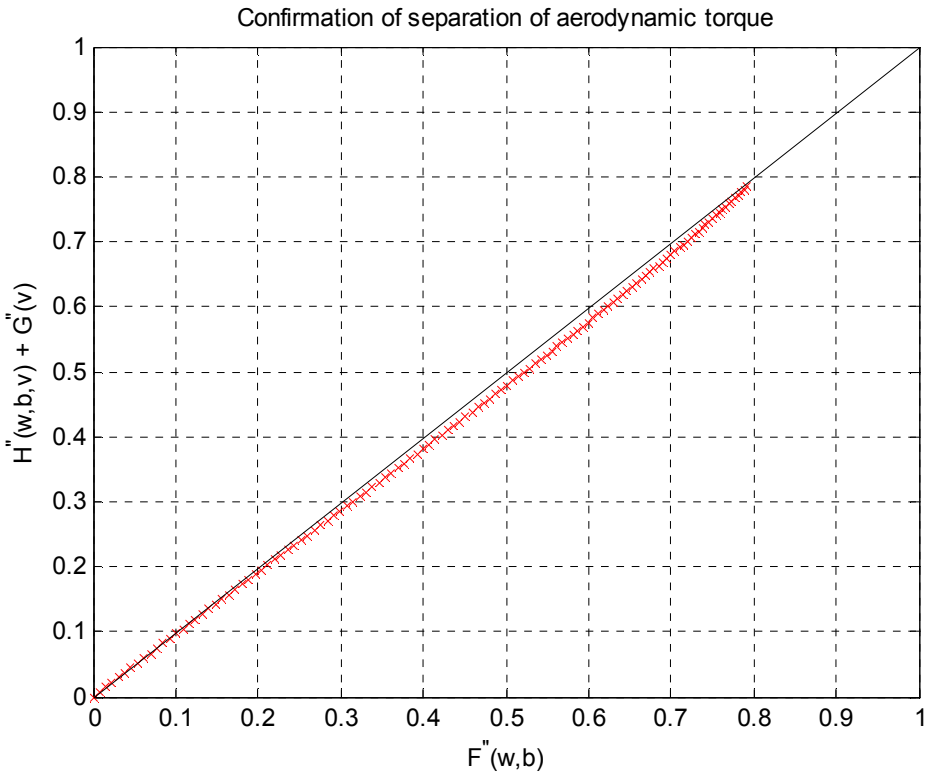


Figure 86 Confirmation of the separation of aerodynamic torque.

5.4 Conclusion

The aerodynamic and drive-train dynamic, in particular the first drive-train mode, are successfully identified for a 1MW variable speed wind turbine from the measured data wholly using Gaussian process prior models within a Bayesian context. The data, consisting of the nacelle anemometer measurement of wind speed, rotor speed and blade pitch angle, is measured whilst the wind turbine is operating normally in above rated wind conditions. The separation of the aerodynamic torque into two additive components, the first being a function of rotor speed and blade pitch angle and the second a function of the wind speed only, is confirmed using various methods. The confirmation enables a direct assessment to be made as to whether the control system on the wind turbine has achieved its design performance.

Chapter 6

State-Space Time-Series Gaussian Process Prior Models

6.1 Introduction

In an engineering context, measurements, intended to inform on a nonlinear relationship, are frequently made sequentially. The data then has two possible interpretations, specifically, as a set of measurements dependent on the same variables as the nonlinear relationship or as a time-series. The particular context, that motivates consideration of these dual nature measurement sets, is the identification of nonlinear dynamic systems.

In discrete-time, the input-output model for a nonlinear dynamic system has the form

$$\mathrm{x}_{i+1} = \mathrm{g}(\mathbf{w}_i) \quad , \quad \mathbf{w}_i = (\mathrm{x}_i, \mathrm{x}_{i-1}, \cdots, \mathrm{x}_{i-k}, \mathrm{r}_{i+1}, \mathrm{r}_i, \cdots, \mathrm{r}_{i-k}) \tag{90}$$

for some $k > 0$, where r_i is the input to the system at time step, i, and x_i is the output. Here, both the input and the output are restricted to being scalars. Typically, when identifying such a system, the response to a known input is measured at a fixed-

interval time sequence, $t_i, i = 1,\cdots,N$; that is, the system (90) is to be identified from the input-out pairs, $\{\mathrm{y}_i, \mathrm{r}_i\}_{i=1}^N$, where the measured output,

$$\mathrm{y}_i = \mathrm{x}_i + \mathrm{n}_i \tag{91}$$

with $\mathrm{n}_i, i = 1,\cdots,N$, consists of additive noise. The known input is assumed to be noise free. The requirement is to determine the nonlinear function, g(**w**), in (90).

Interpreting the measurements to be dependent on **w**, the data set is $\{\mathrm{y}_i, \mathbf{w}_i\}_{i=1}^N$. Although it constitutes a slight abuse of notation, this interpretation here is referred to as the state-space interpretation of the data. It has previously been exploited to apply Gaussian regression to the task of determining g(**w**) in (90) (Rasmussen, 1996; MacKay, 1998; Williams, 1999; Leith, et al., 2000). A standard Gaussian process model, i.e. one based on a single Gaussian process, $\mathrm{g}_{\mathbf{w}}$, with explanatory variable **w**, is employed. The prior is conditioned on the data set, $\{\mathrm{y}_i, \mathbf{w}_i\}_{i=1}^N$, to obtain the posterior, the mean of which is interpreted as usual to be the best fit for g(**w**).

Interpreting the measurements to be dependent on *t*, the data set is $\{\mathrm{y}_i, t_i\}_{i=1}^N$. This interpretation here is referred to as the time-series interpretation of the data. In linear system identification, it has been exploited for a number of reasons through pre-filtering the data (see Chapter 14 in Jung, 1999). However, the time-series interpretation has not been exploited, other than in an ad hoc manner, when applying Gaussian regression to nonlinear dynamic system identification. The purpose of this chapter is to construct Gaussian process models, here referred to as SSTS models, which have both a state-space aspect and a time-series aspect, and to investigate Gaussian regression based on them. Clearly, the state-space interpretation has the greater explanatory power. Indeed, the utility of the time-series interpretation is strongly dependent on the choice of the input, i.e. on experimental design, and in some cases may not provide any added value to the state-space interpretation alone. Nevertheless, the SSTS model should treat both interpretations of the data even-handedly. Furthermore, the explanatory capability of the SSTS model-based Gaussian regression should not be markedly less than the explanatory capability of standard

Gaussian regression. These considerations must inform the construction of the SSTS models.

Dual nature Gaussian process models are developed in §6.2. The dual nature models support the interpretation of data in terms of two general explanatory variables, **w** or **z**, and are more general than the SSTS models in which **w** is defined in conformity with (90) and **z** is restricted to being a scalar with the data measured for a fixed-interval sequence of values. The selection of hyperparameter values for the dual nature models is discussed in §6.3 and the SSTS models are investigated in §6.4. An application based on the SSTS models is illustrated in §6.5, with the introduction of dynamic lengthscale algorithm in §6.6. In Sections 6.2, 6.3 and 6.4, it is assumed that, in the data sets, the values of the explanatory variables have no additive noise. However, only values of y_i are available in practice, not values of x_i, and altering (90) to

$$x_{i+1} = g(\mathbf{w}_i) \quad , \quad \mathbf{w}_i = (y_i, y_{i-1}, \cdots, y_{i-k}, r_{i+1}, r_i, \cdots, r_{i-k}) \tag{92}$$

invalidates this assumption. In §6.7, the SSTS model is altered to cater for this unavailability of values of x_i. Finally, in §6.8 some conclusions are drawn.

6.2 Dual Nature Data Models

In this section, dual nature Gaussian process models are developed. The dual nature models support the interpretation of data in terms of two general explanatory variables, **w** or **z**, and are more general than the SSTS models in which **w** is defined in conformity with (90) and **z** is restricted to being a scalar with the data measured for a fixed-interval sequence of values.

6.2.1 Models with Common Measurement Noise

A model, based on two independent Gaussian processes, for two sets of measurements with common measurement noise, Model 1, is defined below.

Model 1: Let $\mathrm{f}_{\mathbf{w}}$ and $\mathrm{g}_{\mathbf{z}}$ be two independent Gaussian Processes with explanatory variables **w** and **z**, respectively. Suppose N measurements, $\{\mathrm{y}_{\mathrm{f}_i}, \mathbf{w}_i\}|_{i=1}^{N}$, for $\mathrm{f}_{\mathbf{w}}$ and $\overline{N} \le N$ measurements, $\{\mathrm{y}_{\mathrm{g}_i}, \mathbf{z}_i\}|_{i=1}^{\overline{N}}$, for $\mathrm{g}_{\mathbf{z}}$ are available with additive Gaussian measurement noise; that is, $\mathrm{y}_{\mathrm{f}_i} = \mathrm{f}_{\mathbf{w}_i} + n_i$ and $\mathrm{y}_{\mathrm{g}_i} = \mathrm{g}_{\mathbf{z}_i} + \overline{n}_i$, where n_i and $\overline{n}_i$ are Gaussian noise. Furthermore, suppose that the noise on the two sets of measurements is common; that is, $\forall i \le \overline{N}$, $\exists \mathrm{k}_i \le N$ such that $i \ne j \Rightarrow \mathrm{k}_i \ne \mathrm{k}_j$ and $\overline{n}_i = n_{\mathrm{k}_i}$. Let $\mathrm{P} \in \mathrm{P}^{\overline{N} \times N}$ be such that its *ij*-th element is 1, when $j = \mathrm{k}_i$, and 0, when $j \ne \mathrm{k}_i$, then $\mathrm{N}_{\mathrm{G}} = \mathrm{P}\,\mathrm{N}_{\mathrm{F}}$, where $\mathrm{N}_{\mathrm{F}} = [n_1, \cdots, n_N]^{\mathrm{T}}$ and $\mathrm{N}_{\mathrm{G}} = [n_1, \cdots, n_{\overline{N}}]^{\mathrm{T}}$. Note, $\mathrm{PP}^{\mathrm{T}} = I$. It follows that $\mathrm{E}[\mathrm{N}_{\mathrm{G}}\mathrm{N}_{\mathrm{G}}^{\mathrm{T}}] = \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}$ and $\mathrm{E}[\mathrm{N}_{\mathrm{G}}\mathrm{N}_{\mathrm{F}}^{\mathrm{T}}] = \mathrm{P}\,\mathbf{B}$, where $\mathbf{B} = \mathrm{E}[\mathrm{N}_{\mathrm{F}}\mathrm{N}_{\mathrm{F}}^{\mathrm{T}}]$. Let $\mathrm{f}_{\mathbf{w}}$ and $\mathrm{g}_{\mathbf{z}}$ be zero mean and $\Lambda_{\mathrm{F}} = \mathrm{E}[\mathrm{FF}^{\mathrm{T}}]$, where $\mathrm{F} = [\mathrm{f}_{\mathbf{w}_1}, \cdots, \mathrm{f}_{\mathbf{w}_N}]^{\mathrm{T}}$, and $\Lambda_{\mathrm{G}} = \mathrm{E}[\mathrm{GG}^{\mathrm{T}}]$, where $\mathrm{G} = [\mathrm{g}_{\mathbf{z}_1}, \cdots, \mathrm{g}_{\mathbf{z}_{\overline{N}}}]^{\mathrm{T}}$.

Since $\mathrm{f}_{\mathbf{w}}$ and $\mathrm{g}_{\mathbf{z}}$ are independent, the prior covariance matrix for the combined sets of measurements, $[\mathrm{Y}_{\mathrm{F}}^{\mathrm{T}} \quad \mathrm{Y}_{\mathrm{G}}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathrm{Y}_{\mathrm{F}} = [\mathrm{y}_{\mathrm{f}_1}, \cdots, \mathrm{y}_{\mathrm{f}_N}]^{\mathrm{T}}$ and $\mathrm{Y}_{\mathrm{G}} = [\mathrm{y}_{\mathrm{g}_1}, \cdots, \mathrm{y}_{\mathrm{g}_{\overline{N}}}]^{\mathrm{T}}$, is

$$\mathbf{Q}_{\mathrm{FG}} = \begin{bmatrix} \Lambda_{\mathrm{F}} + \mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{bmatrix} \tag{93}$$

Conditioning on the combined data set, the posterior probability distribution for the predicted values, corresponding to $[\mathrm{Y}_{\mathrm{F}}^{\mathrm{T}} \quad \mathrm{Y}_{\mathrm{G}}^{\mathrm{T}}]^{\mathrm{T}}$, remains Gaussian. Using Theorem 6.1 (a), the mean vector is

$$\begin{bmatrix} \hat{\mathrm{Y}}_{\mathrm{F}} \\ \hat{\mathrm{Y}}_{\mathrm{G}} \end{bmatrix} = \mathbf{\Lambda}_{\mathrm{FG}}\mathbf{Q}_{\mathrm{FG}}^{-1}\begin{bmatrix} \mathrm{Y}_{\mathrm{F}} \\ \mathrm{Y}_{\mathrm{G}} \end{bmatrix} = \begin{bmatrix} \mathrm{Y}_{\mathrm{F}} \\ \mathrm{Y}_{\mathrm{G}} \end{bmatrix} - \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}\mathrm{X}^{-1}\left[\Lambda_{\mathrm{F}}^{-1}\mathrm{Y}_{\mathrm{F}} + \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{Y}_{\mathrm{G}}\right]$$

and, using Theorem 6.1 (b), the covariance matrix is

$$\hat{\mathbf{\Lambda}}_{\mathrm{FG}} = \mathbf{\Lambda}_{\mathrm{FG}} - \mathbf{\Lambda}_{\mathrm{FG}}\mathbf{Q}_{\mathrm{FG}}^{-1}\mathbf{\Lambda}_{\mathrm{FG}} = \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}\mathrm{X}^{-1}\begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix} \tag{94}$$

where $\mathbf{\Lambda}_{\mathrm{FG}} = \begin{bmatrix} \Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}} \end{bmatrix}$ and $\mathrm{X} = [\mathbf{B}^{-1} + \Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]$.

Equivalently, the posterior probability distribution for the corresponding estimation errors or noise has mean vector and covariance matrix, respectively,

$$\begin{bmatrix} \hat{\mathrm{N}}_{\mathrm{F}} \\ \hat{\mathrm{N}}_{\mathrm{G}} \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{N}}_{\mathrm{F}} \\ \mathrm{P}\,\hat{\mathrm{N}}_{\mathrm{F}} \end{bmatrix} = \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \mathrm{X}^{-1} \left[\Lambda_{\mathrm{F}}^{-1} \mathrm{Y}_{\mathrm{F}} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{Y}_{\mathrm{G}} \right] \text{ and } \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}^{-1} \mathrm{X}^{-1} \begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix}$$

Hence, the posterior noise remains common. The structure of the covariance matrix, (93), ensures that $\hat{\mathrm{Y}}_{\mathrm{F}}$ and $\hat{\mathrm{Y}}_{\mathrm{G}}$ are strongly related in the above manner.

The covariance matrices, Λ_{F}, Λ_{G} and $\mathbf{B}$, depend on some set of hyperparameters. The negative log likelihood of the data with covariance matrix, (93), is

$$\ln\left(\begin{vmatrix} \Lambda_{\mathrm{F}} + \mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{vmatrix} \right) + \begin{bmatrix} \mathrm{Y}_{\mathrm{F}}^{\mathrm{T}} & \mathrm{Y}_{\mathrm{G}}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \Lambda_{\mathrm{F}} + \mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{Y}_{\mathrm{F}} \\ \mathrm{Y}_{\mathrm{G}} \end{bmatrix} \tag{95}$$

When the values of these hyperparameters are obtained by maximising the likelihood or equivalently minimising the negative log likelihood of the data, (95) should be used. Although the noise on the two sets of measurements is common, Y_{F} and Y_{G}, together, provide more information than Y_{F} alone, since the values of G do not depend on F.

Theorem 6.1: For $\overline{N} \le N$, where $\mathbf{B} \in \mathrm{P}^{N \times N}$, $\Lambda_{\mathrm{F}} \in \mathrm{P}^{N \times N}$, $\Lambda_{\mathrm{G}} \in \mathrm{P}^{\overline{N} \times \overline{N}}$, $\mathrm{P} \in \mathrm{P}^{\overline{N} \times N}$, $\mathrm{Y}_{\mathrm{F}} \in \mathrm{P}^{N}$ and $\mathrm{Y}_{\mathrm{G}} \in \mathrm{P}^{\overline{N}}$, it follows that

(a)
$$\begin{bmatrix} \Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}} \end{bmatrix} \begin{bmatrix} \Lambda_{\mathrm{F}} + \mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{Y}_{\mathrm{F}} \\ \mathrm{Y}_{\mathrm{G}} \end{bmatrix} = \begin{bmatrix} \mathrm{Y}_{\mathrm{F}} - \left[\mathbf{B}^{-1} + \Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{P} \right]^{-1} \left[\Lambda_{\mathrm{F}}^{-1} \mathrm{Y}_{\mathrm{F}} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{Y}_{\mathrm{G}} \right] \\ \mathrm{Y}_{\mathrm{G}} - \mathrm{P} \left[\mathbf{B}^{-1} + \Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{P} \right]^{-1} \left[\Lambda_{\mathrm{F}}^{-1} \mathrm{Y}_{\mathrm{F}} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{Y}_{\mathrm{G}} \right] \end{bmatrix}$$

(b)
$$\begin{bmatrix} \Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}} \end{bmatrix} - \begin{bmatrix} \Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}} \end{bmatrix} \begin{bmatrix} \Lambda_{\mathrm{F}} + \mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{bmatrix}^{-1} \begin{bmatrix} \Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}} \end{bmatrix} = \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \left[\mathbf{B}^{-1} + \Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{P} \right]^{-1} \begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix}$$

(c)
$$\mathrm{P} \left[I - [\mathbf{B}^{-1} + \Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{P}]^{-1} [\Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}} \Lambda_{\mathrm{G}}^{-1} \mathrm{P}] \right] = \Lambda_{\mathrm{G}} \left[\Lambda_{\mathrm{G}} + \mathrm{P} \left[\mathbf{B}^{-1} + \Lambda_{\mathrm{F}}^{-1} \right]^{-1} \mathrm{P}^{\mathrm{T}} \right]^{-1} \mathrm{P}\, \Lambda_{\mathrm{F}} \left[\mathbf{B} + \Lambda_{\mathrm{F}} \right]^{-1}$$

(d) $\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]^{-1}\mathrm{P}^{\mathrm{T}}=\Lambda_{\mathrm{G}}-\Lambda_{\mathrm{G}}\left[\Lambda_{\mathrm{G}}+\mathrm{P}\left[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}\right]^{-1}\mathrm{P}^{\mathrm{T}}\right]^{-1}\Lambda_{\mathrm{G}}$

(e) $\left[I \quad \mathrm{P}^{\mathrm{T}}\right]\begin{bmatrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{bmatrix}^{-1}\begin{bmatrix}I \\ \mathrm{P}\end{bmatrix}=\left[\left(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right)^{-1}+\mathbf{B}\right]^{-1}$

(f) $\left|\begin{matrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{matrix}\right|=|\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,|\left|[\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]^{-1}+\mathbf{B}\right|$

The proofs for Theorem 6.1 are shown below.

$$\begin{bmatrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{bmatrix}^{-1}$$
$$=\begin{bmatrix}\Lambda_{\mathrm{F}}^{-1} & 0 \\ 0 & \Lambda_{\mathrm{G}}^{-1}\end{bmatrix}-\begin{bmatrix}\Lambda_{\mathrm{F}}^{-1} \\ \Lambda_{\mathrm{G}}^{-1}\mathrm{P}\end{bmatrix}\left[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}\begin{bmatrix}\Lambda_{\mathrm{F}}^{-1} & \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\end{bmatrix}$$

Hence,

$$\begin{bmatrix}\Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}}\end{bmatrix}\begin{bmatrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{bmatrix}^{-1}$$
$$=\begin{bmatrix}I & 0 \\ 0 & I\end{bmatrix}-\begin{bmatrix}I \\ \mathrm{P}\end{bmatrix}\left[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}\begin{bmatrix}\Lambda_{\mathrm{F}}^{-1} & \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\end{bmatrix}$$

and

$$\begin{bmatrix}\Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}}\end{bmatrix}\begin{bmatrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{bmatrix}^{-1}\begin{bmatrix}\Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}}\end{bmatrix}$$
$$=\begin{bmatrix}\Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}}\end{bmatrix}-\begin{bmatrix}I \\ \mathrm{P}\end{bmatrix}\left[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}\begin{bmatrix}I & \mathrm{P}^{\mathrm{T}}\end{bmatrix}$$

It follows immediately from (a) and (b) in Theorem 6.1 that

$$\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]^{-1}$$
$$=\mathrm{P}\left[I+[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}$$
$$=\left[I+\mathrm{R}[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\right]^{-1}\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}$$
$$=\Lambda_{\mathrm{G}}\left[\Lambda_{\mathrm{G}}+\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathrm{P}^{\mathrm{T}}\right]^{-1}\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}$$

Hence,

$$\mathrm{P}\left[I-[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]^{-1}[\,\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,]\right]$$
$$=\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,]^{-1}\mathbf{B}^{-1}$$
$$=\Lambda_{\mathrm{G}}\left[\Lambda_{\mathrm{G}}+\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathrm{P}^{\mathrm{T}}\right]^{-1}\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathbf{B}^{-1}$$

and

$$\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,]^{-1}\mathrm{P}^{\mathrm{T}}=\Lambda_{\mathrm{G}}\left[\Lambda_{\mathrm{G}}+\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathrm{P}^{\mathrm{T}}\right]^{-1}\mathrm{P}\,[\mathbf{B}^{-1}+\Lambda_{\mathrm{F}}^{-1}]^{-1}\mathrm{P}^{\mathrm{T}}$$

Hence, (c) and (d) follows immediately. (e) is established as follows

$$\begin{aligned}
&\begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} \Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{bmatrix}^{-1}\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}=\begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix}\left[\begin{bmatrix} \Lambda_{\mathrm{F}} & 0 \\ 0 & \Lambda_{\mathrm{G}} \end{bmatrix}+\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}\mathbf{B}\begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix}\right]^{-1}\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \\
&=\begin{bmatrix} I & \mathrm{P}^{\mathrm{T}} \end{bmatrix}\begin{bmatrix} \Lambda_{\mathrm{F}}^{-1} & 0 \\ 0 & \Lambda_{\mathrm{G}}^{-1} \end{bmatrix}\left[I+\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}\mathbf{B}\begin{bmatrix} \Lambda_{\mathrm{F}}^{-1} & \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1} \end{bmatrix}\right]^{-1}\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \\
&=\begin{bmatrix} \Lambda_{\mathrm{F}}^{-1} & \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1} \end{bmatrix}\left[I-\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix}\left[I+\mathbf{B}(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)\right]^{-1}\mathbf{B}\begin{bmatrix} \Lambda_{\mathrm{F}}^{-1} & \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1} \end{bmatrix}\right]\begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \\
&=(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)-(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)\left[I+\mathbf{B}(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)\right]^{-1}\mathbf{B}(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,) \\
&=(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)\left[I+\mathbf{B}(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)\right]^{-1}=\left[(\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,)^{-1}+\mathbf{B}\right]^{-1}
\end{aligned}$$

In addition, (f) is established as follows

$$\begin{aligned}
&|\,[\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,]^{-1}+\mathbf{B}\,|=|\,\Lambda_{\mathrm{F}}[I+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,\Lambda_{\mathrm{F}}]^{-1}+\mathbf{B}\,| \\
&\quad=|\,\Lambda_{\mathrm{F}}[I-\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}[I+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}]^{-1}\mathrm{P}\,\Lambda_{\mathrm{F}}]+\mathbf{B}\,| \\
&\quad=|\,\Lambda_{\mathrm{F}}[I-\mathrm{P}^{\mathrm{T}}[\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\Lambda_{\mathrm{F}}]+\mathbf{B}\,| \\
&\quad=|\,\Lambda_{\mathrm{F}}-\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}[\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\Lambda_{\mathrm{F}}+\mathbf{B}\,| \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,I-[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}[\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\Lambda_{\mathrm{F}}\,| \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,I-\mathrm{P}\,\Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}[\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}]^{-1}\,| \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}-\mathrm{P}\,\Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,|^{-1} \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,[\Lambda_{\mathrm{F}}-\Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\Lambda_{\mathrm{F}}]\mathrm{P}^{\mathrm{T}}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,|^{-1} \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,[\mathbf{B}-\Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}]\mathrm{P}^{\mathrm{T}}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,|^{-1} \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}-\mathrm{P}\,\Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}\mathrm{P}^{\mathrm{T}}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,|^{-1}
\end{aligned}$$

Hence,

$$\begin{aligned}
&|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}-\mathrm{P}\,\Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}\mathrm{P}^{\mathrm{T}}\,| \\
&\quad=|\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,||\,[\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,]^{-1}+\mathbf{B}\,|
\end{aligned}$$

and

$$\begin{aligned}
&\begin{vmatrix} \Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{vmatrix}=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,|\begin{vmatrix} I & [\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{vmatrix} \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,|\begin{vmatrix} I & [\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}\mathrm{P}^{\mathrm{T}} \\ 0 & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}-\mathrm{P}\,\mathbf{B}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}\mathrm{P}^{\mathrm{T}} \end{vmatrix} \\
&\quad=|\,\Lambda_{\mathrm{F}}+\mathbf{B}\,||\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}-\mathrm{P}\,\mathbf{B}[\Lambda_{\mathrm{F}}+\mathbf{B}]^{-1}\mathbf{B}\mathrm{P}^{\mathrm{T}}\,| \\
&\quad=|\,\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\,||\,[\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\,]^{-1}+\mathbf{B}\,|
\end{aligned}$$

6.2.2 Models for Common Measurements

In §6.2.1, Model 1 for two sets of measurements with common measurement noise is discussed. *Model 2* is defined similarly except that the measurements are also common, specifically

$$\mathrm{Y_F} = \mathbf{Y} \quad , \quad \mathrm{Y_G} = \mathrm{P}\,\mathbf{Y}$$

Whilst the covariance matrix for the posterior probability distribution with Model 2 remains unchanged from (92), the mean vector with Model 2 becomes

$$\begin{bmatrix} \hat{\mathrm{Y}}_\mathrm{F} \\ \hat{\mathrm{Y}}_\mathrm{G} \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{Y}}_\mathrm{F} \\ \mathrm{P}\,\hat{\mathrm{Y}}_\mathrm{F} \end{bmatrix} = \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \left[I - \mathrm{X}^{-1}[\Lambda_\mathrm{F}^{-1} + \mathrm{P}^{\,\mathrm{T}}\Lambda_\mathrm{G}^{-1}\mathrm{P}\,] \right] \mathrm{Y} = \begin{bmatrix} I \\ \mathrm{P} \end{bmatrix} \mathrm{X}^{-1}\mathbf{B}^{-1}\mathbf{Y} \tag{96}$$

By Theorem 6.1 (c), the posterior mean for $\mathrm{Y_G}$ is

$$\hat{\mathrm{Y}}_\mathrm{G} = \mathrm{P}\,\hat{\mathrm{Y}}_\mathrm{F} = \Lambda_\mathrm{G}\left[\Lambda_\mathrm{G} + \widetilde{\mathrm{B}}_\mathrm{G}\right]^{-1}\widetilde{\mathrm{Y}} \quad , \quad \widetilde{\mathrm{Y}} = \mathrm{P}\,\Lambda_\mathrm{F}[\Lambda_\mathrm{F} + \mathbf{B}]^{-1}\mathbf{Y} \tag{97}$$

and, by Theorem 6.1 (d), the covariance matrix for $\mathrm{Y_G}$ is

$$\hat{\Lambda}_\mathrm{G} = \mathrm{P}^{\,\mathrm{T}}\mathrm{X}^{-1}\mathrm{P}\, = \Lambda_\mathrm{G} - \Lambda_\mathrm{G}[\Lambda_\mathrm{G} + \widetilde{\mathrm{B}}_\mathrm{G}]^{-1}\Lambda_\mathrm{G} \tag{98}$$

where $\widetilde{\mathrm{B}}_\mathrm{G} = \mathrm{P}\,[\mathbf{B}^{-1} + \Lambda_\mathrm{F}^{-1}]^{-1}\mathrm{P}^{\,\mathrm{T}}$. From (97) and (98), a natural interpretation of regression based on Model 2 is possible. Firstly, **Y** is filtered by applying regression, based on a prior model with covariance matrices for the stochastic process and noise, Λ_F and B, respectively, to the data, **Y**, to obtain $\widetilde{\mathrm{Y}}$ and $\widetilde{\mathrm{B}}$. Secondly, regression, based on a prior model with covariance matrices for the stochastic process and noise, Λ_G and $\widetilde{\mathrm{B}}$, respectively, is applied to the data, $\widetilde{\mathrm{Y}}$, to obtain $\hat{\Lambda}_\mathrm{G}$.

Since $\mathrm{Y_F}$ and $\mathrm{Y_G}$ are not independent but $\mathrm{Y_G} = \mathrm{P}\,\mathrm{Y_F}$, (95) is no longer the likelihood of the measurements with Model 2. To obtain the values of the hyperparameters with Model 2, the union of the hyperparameter sets for $\mathrm{f_w}$, $\mathrm{g_z}$ and the noise, the likelihood of the data subjected to the constraint, $\mathrm{Y_G} = \mathrm{P}\,\mathrm{Y_F}$, rather than (95) should be maximised. It is determined by constructing Model 3 for the measurements. Model 3 is based on a single Gaussian process with explanatory variable, **w**, and the same set of hyperparameters as Model 2, such that, when conditioned on the measurements, **Y**,

the posterior mean vector is $X^{-1}\mathbf{B}^{-1}\mathbf{Y}$ and the covariance matrix is X^{-1}. In terms of modelling the measured data, Model 3 is, by construction, completely equivalent to Model 2. The values for the hyperparameters, obtained by maximising the likelihood of the measurements for Model 3, are equally applicable to Model 2.

Model 3: Let $h_{\mathbf{w}}$ be a Gaussian process with explanatory variable, **w**; that is, it has the same explanatory variable as $f_{\mathbf{w}}$ in Model 2. Suppose *N* measurements, $\{y_{h_i}, \mathbf{w}_i\}|_{i=1}^{N}$, for $h_{\mathbf{w}}$ are available with the same additive Gaussian measurement noise as in $\{y_{f_i}\}|_{i=1}^{N}$; that is, $y_{h_i} = h_{\mathbf{w}_i} + n_i$. Let $h_{\mathbf{w}}$ be zero mean and $\Lambda_H = E[HH^T]$, where $H = [h_{\mathbf{w}_1}, \cdots, h_{\mathbf{w}_N}]^T$. Suppose a mapping, $\mathbf{z} : \mathbf{w} \mapsto \mathbf{z}$, exists such that $\mathbf{z}_i = \mathbf{z}(\mathbf{w}_{k_i})$, $\forall i \le \overline{N}$. This is always the case, provided $\mathbf{z}_i = \mathbf{z}_j$ whenever $\mathbf{w}_{k_i} = \mathbf{w}_{k_j}$. Further suppose that $Y_H = \mathbf{Y}$, where $Y_H = [y_{h_1}, \cdots, y_{h_N}]^T$.

Consider Model 3 with

$$\Lambda_H = [\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1}$$

The prior covariance matrix for the measurement set, Y_H, is $[\Lambda_H + \mathbf{B}]$. Conditioning on the data, the posterior probability distribution for the predicted values corresponding to Y_H has mean vector and covariance matrix, respectively,

$$\hat{Y}_H = \Lambda_H [\Lambda_H + \mathbf{B}]^{-1} Y_H \text{ and } \hat{\Lambda}_H = \Lambda_H - \Lambda_H [\Lambda_H + \mathbf{B}]^{-1} \Lambda_H$$

As required,

$$\hat{Y}_H = [\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1} \left[[\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1} + \mathbf{B} \right]^{-1} \mathbf{Y} = X^{-1} \mathbf{B}^{-1} \mathbf{Y}$$

and

$$\hat{\Lambda}_H = [\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1} \left[[\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1} + \mathbf{B} \right]^{-1} \mathbf{B} = X^{-1}$$

Hence, the negative log likelihood of the measurements is

$$\begin{aligned} &\ln |\Lambda_H + \mathbf{B}| + Y_H^T [\Lambda_H + \mathbf{B}]^{-1} Y_H \\ &= \ln \left\{ \left| [\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1} + \mathbf{B} \right| \right\} + \mathbf{Y}^T \left[[\Lambda_F^{-1} + P^T \Lambda_G^{-1} P]^{-1} + \mathbf{B} \right]^{-1} \mathbf{Y} \end{aligned} \tag{99}$$

When the values of the hyperparameters for Model 2 are obtained by minimising the negative log likelihood of the measurements, (99) should be used. By Theorem 6.1 (e) and (f),

$$\begin{bmatrix}\mathbf{Y}^{\mathrm{T}} & \mathbf{Y}^{\mathrm{T}}\mathrm{P}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{Y} \\ \mathrm{P}\,\mathbf{Y}\end{bmatrix}=\mathbf{Y}^{\mathrm{T}}\left[\left[\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}+\mathbf{B}\right]^{-1}\mathbf{Y}$$

but

$$\begin{vmatrix}\Lambda_{\mathrm{F}}+\mathbf{B} & \mathbf{B}\mathrm{P}^{\mathrm{T}} \\ \mathrm{P}\,\mathbf{B} & \Lambda_{\mathrm{G}}+\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}\end{vmatrix}=\left|\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\right|\left|\left[\Lambda_{\mathrm{F}}^{-1}+\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}+\mathbf{B}\right|$$

Hence, the negative log likelihood function, (99), differs from the negative log likelihood function, (95), by the term

$$-\ln\left\{\left|\Lambda_{\mathrm{G}}+\mathrm{P}\,\Lambda_{\mathrm{F}}\mathrm{P}^{\mathrm{T}}\right|\right\}$$

In Model 2, the two sets of measurements are common with common noise. Furthermore, suppose that the two Gaussian processes are the same with the same explanatory variable and the same hyperparameter sets; that is, the hyperparameters are the union of the hyperparameter sets for $\mathrm{f}_{\mathbf{w}}$ and the noise. In this special case, for consistency, the model for the measured data, based on two Gaussian processes, should be the same as Model 3, the standard model for the measured data based on a single Gaussian process, with the same explanatory variable and the same hyperparameter set as Model 2.

Suppose the Gaussian processes, $\mathrm{f}_{\mathbf{w}}$ and $\mathrm{g}_{\mathbf{z}}$, in Model 2 and their explanatory variables are the same, i.e. $\mathrm{g}_{\mathbf{z}}|_{\mathbf{z}=\mathbf{w}}=\mathrm{f}_{\mathbf{w}}$, the measurements for both are the same, namely $\{\mathrm{y}_i,\mathbf{w}_i\}|_{i=1}^{N}$, and $\Lambda_{\mathrm{G}}=\Lambda_{\mathrm{F}}=\alpha\mathrm{P}(\Theta)$, where α is the magnitude hyperparameter and the set, Θ, the remaining hyperparameters on which Λ_{F} depends. The noise also remains common. The prior covariance matrix, (93), for Model 2 becomes

$$\mathbf{Q}=\begin{bmatrix}\alpha\mathrm{P}(\Theta)+\mathbf{B} & \mathbf{B} \\ \mathbf{B} & \alpha\mathrm{P}(\Theta)+\mathbf{B}\end{bmatrix} \tag{100}$$

(The hyperparameter dependence of the noise covariance matrix, **B**, is not shown explicitly in (100). However, doing so would make no material difference to the

discussion below.) From (99), the negative log likelihood function of the measured data is

$$\ln\left\{\left|\,(\alpha/2)\mathrm{P}(\Theta)+\mathbf{B}\,\right|\right\}+\mathbf{Y}^{\mathrm{T}}\left[(\alpha/2)\mathrm{P}(\Theta)+\mathbf{B}\right]^{-1}\mathbf{Y} \tag{101}$$

and, from (94) and (96), the mean vector and covariance matrix of the posterior probability distribution for Y_{F} are, respectively,

$$(\alpha/2)\mathrm{P}(\Theta)[(\alpha/2)\mathrm{P}(\Theta)+\mathbf{B}]^{-1}\mathbf{Y} \text{ and } (\alpha/2)\mathrm{P}(\Theta)[(\alpha/2)\mathrm{P}(\Theta)+\mathbf{B}]^{-1}\mathbf{B} \tag{102}$$

Alternatively, suppose the measurements are modelled by Model 3 with

$$\Lambda_{\mathrm{H}}=\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta}),\ \text{where}\ \widetilde{\mathrm{P}}(\widetilde{\Theta})=\mathrm{P}(\Theta)\,|_{\Theta=\widetilde{\Theta}} \tag{103}$$

The prior covariance matrix for Model 3, with (103), is $\widetilde{\mathrm{Q}}=[\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})+\mathbf{B}]$ and the negative log likelihood function of the measured data is

$$\ln\left\{\left|\,\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})+\mathbf{B}\,\right|\right\}+\mathbf{Y}^{\mathrm{T}}\left[\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})+\mathbf{B}\right]^{-1}\mathbf{Y} \tag{104}$$

and the mean vector and covariance matrix of the posterior probability distribution for Y_{H} are, respectively,

$$\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})[\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})+\mathbf{B}]^{-1}\mathbf{Y} \text{ and } \widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})[\widetilde{\alpha}\widetilde{\mathrm{P}}(\widetilde{\Theta})+\mathbf{B}]^{-1}\mathbf{B} \tag{105}$$

Let $\widetilde{\alpha}_M$ and $\widetilde{\Theta}_M$ be the values of the hyperparameters that minimise (104), then $\alpha_M=2\widetilde{\alpha}_M$ and $\Theta_M=\widetilde{\Theta}_M$ minimise (101) and hence, the mean vectors and covariance matrices, (102) and (105) are the same, that is, the posterior probability distributions are the same. The two models are consistent as required.

In Model 2, the predictions, $\hat{\mathrm{Y}}_{\mathrm{F}}$ and $\hat{\mathrm{Y}}_{\mathrm{G}}$, are the same at common points of the explanatory variables; that is, from (97), $\hat{\mathrm{Y}}_{\mathrm{G}}=\mathrm{P}\,\hat{\mathrm{Y}}_{\mathrm{F}}$. However, Model 2 provides an interpretation of the data and predictions in terms of the two explanatory variables, **w** and **z**. Furthermore, both the predictions and the training depend only on the single data set, **Y**, and, provided the training is appropriate, Model 2 is consistent with respect to standard Gaussian regression based on a single Gaussian process. Hence, Model 2 has all the attributes required for the dual nature model. Note that, it is based on two Gaussian processes rather than one. The selection of the hyperparameters for

the dual nature model is discussed in §6.3. An appropriate procedure is proposed that ensures preservation of the dual nature of the model and consistency with respect to standard Gaussian regression for a broad range of circumstances.

6.3 Selection of Hyperparameters

Consider the situation when $\mathrm{P} = I$ and the covariance matrix for the measured values for the dual nature model is $\left[[\Lambda_{\mathrm{F}}^{-1} + \Lambda_{\mathrm{G}}^{-1}]^{-1} + \mathbf{B}\right]$, where $\Lambda_{\mathrm{F}} = a_{\mathrm{F}}\mathrm{P}_{\mathrm{F}}$ and $\Lambda_{\mathrm{G}} = a_{\mathrm{G}}\mathrm{P}_{\mathrm{G}}$. Suppose that the noise hyperparameters are known and that the hyperparameters for Λ_{G} have been chosen to maximise the likelihood of the measured data using the covariance matrix, $[\Lambda_{\mathrm{G}} + \mathbf{B}]$. Furthermore, suppose that the nonlinear relationship underlying the measured data is better explained in terms of the explanatory variable, **w**, than **z**. For almost all sets of data, the likelihood obtained using $[\Lambda_{\mathrm{G}} + \mathbf{B}]$ would be higher than the likelihood obtained using $\left[[\Lambda_{\mathrm{F}}^{-1} + \Lambda_{\mathrm{G}}^{-1}]^{-1} + \mathbf{B}\right]$ for any other choice of hyperparameters for Λ_{G} and any choice for Λ_{F}; that is, the covariance matrix $[\Lambda_{\mathrm{G}} + \mathbf{B}]$ would provide the best explanation for the measured data. Consequently, under these circumstances, selecting the hyperparameters for the dual nature model, by maximising the likelihood of the measured data using the covariance matrix $\left[[\Lambda_{\mathrm{F}}^{-1} + \Lambda_{\mathrm{G}}^{-1}]^{-1} + \mathbf{B}\right]$, would diminish the role of Λ_{F} making $a_{\mathrm{F}} / a_{\mathrm{G}}$ very large. When $\mathrm{P} \neq I$ but the size of the data set, P**Y**, is not overly reduced in comparison to **Y**, the covariance matrix $[\Lambda_{\mathrm{G}} + \mathbf{B}]$ may remain the best explanation for the measured data and $a_{\mathrm{F}} / a_{\mathrm{G}}$ may remain large on selecting the hyperparameters for the dual nature model by maximising the likelihood function.

When $a_{\mathrm{F}} \to \infty$, the prediction $\hat{\mathrm{Y}}_{\mathrm{F}} \to \left[I + \mathbf{B}\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}\mathbf{Y}$, since

$$\hat{\mathrm{Y}}_{\mathrm{F}} = [\Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]^{-1}\left[[\Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]^{-1} + \mathbf{B}\right]^{-1}\mathbf{Y} = \left[I + \mathbf{B}[\Lambda_{\mathrm{F}}^{-1} + \mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}]\right]^{-1}\mathbf{Y}$$

Partition the data set, **Y**, into two orthogonal components, Y_1 and Y_2, such that $\mathbf{Y} = \mathrm{Y}_1 + \mathrm{Y}_2$, where

$$\mathrm{Y}_1 = [I - \mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1}\mathrm{P}]\mathbf{Y} \quad ; \quad \mathrm{Y}_2 = \mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1}\mathrm{P}\,\mathbf{Y}$$

Since

$$\left[I + \mathbf{B}\mathrm{P}^{\mathrm{T}}\Lambda_{\mathrm{G}}^{-1}\mathrm{P}\right]^{-1}\mathbf{Y}$$
$$= [I - \mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1}\mathrm{P}\,]\mathbf{Y} + \mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1}\Lambda_{\mathrm{G}}[\Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\mathbf{Y}$$

It follows that

$$\hat{\mathrm{Y}}_{\mathrm{F1}} = \mathrm{Y}_1 \quad ; \quad \hat{\mathrm{Y}}_{\mathrm{F2}} = \mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1}\Lambda_{\mathrm{G}}[\Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\mathrm{Y}_2$$

Similarly, when $a_{\mathrm{F}} \to \infty$, the prediction $\hat{\mathrm{Y}}_{\mathrm{G}} = \mathrm{P}\,\hat{\mathrm{Y}}_{\mathrm{F}} \to \Lambda_{\mathrm{G}}[\Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\mathbf{Y}$. Because $\mathrm{P}\,\mathrm{Y}_2 = \mathrm{P}\,\mathbf{Y}$, it follows from the definition of Y_2 that

$$\mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1} : \mathrm{P}\,\mathbf{Y} \mapsto \mathrm{Y}_2 \text{ and } \mathrm{P} : \mathrm{Y}_2 \mapsto \mathrm{P}\,\mathbf{Y}$$

Similarly,

$$\mathbf{B}\mathrm{P}^{\mathrm{T}}(\mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}})^{-1} : \Lambda_{\mathrm{G}}[\Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\mathrm{Y}_2 \mapsto \hat{\mathrm{Y}}_{\mathrm{F2}}$$

and

$$\mathrm{P} : \hat{\mathrm{Y}}_{\mathrm{F2}} \mapsto \Lambda_{\mathrm{G}}[\Lambda_{\mathrm{G}} + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}]^{-1}\mathrm{P}\,\mathrm{Y}_2$$

Furthermore, since $\mathrm{P}\,\mathrm{Y}_2 = \mathrm{P}\,\mathbf{Y}$, $\hat{\mathrm{Y}}_{\mathrm{G}} = \mathrm{P}\,\hat{\mathrm{Y}}_{\mathrm{F2}}$. Hence, the prediction of $\hat{\mathrm{Y}}_{\mathrm{F2}}$ from Y_2 and the prediction of $\hat{\mathrm{Y}}_{\mathrm{G}}$ from P**Y** are equivalent; to be precise, the sets of possible prediction pairs, $(\mathrm{Y}_2, \hat{\mathrm{Y}}_{\mathrm{F2}})$ and $(\mathrm{P}\,\mathrm{Y}_2, \hat{\mathrm{Y}}_{\mathrm{G}})$, are isomorphic. The orthogonal prediction, i.e. the prediction of $\hat{\mathrm{Y}}_{\mathrm{F1}}$ from Y_1, simply leaves Y_1 unchanged. In other words, when $a_{\mathrm{F}} \to \infty$, only the correlation with respect to **w** is exploited and not the correlation with respect to **z**. No added utility is derived from the dual nature model in comparison to standard Gaussian regression based on $\mathrm{g}_{\mathbf{w}}$ alone.

The hyperparameter values need to be chosen such that the dual nature of the model is preserved to enable both aspects to be exploited. However, from the preceding discussion, it is clear that a more considered approach to choosing the hyperparameters, than simply maximising the likelihood of the measured data using the covariance matrix $\left[[\Lambda_{\mathrm{F}}^{-1} + \Lambda_{\mathrm{G}}^{-1}]^{-1} + \mathbf{B}\right]$, is required. In addition, the context within which the dual nature model is being applied, must inform the approach to be used. From the applications, from which motivation for the development of the dual nature model arises, see §6.1, the context has the following attributes.

a. The nonlinear relationship underlying the measured data is better explained in terms of the explanatory variable, **w**, than **z**.

b. The benchmark for comparison of the utility of the dual nature model is Gaussian regression based on g_w alone.

c. Typically, though not always, the noise is white Gaussian, i.e. its covariance, **B**, is *bI*.

Within the above context, consider the following procedure for selecting the hyperparameters in the dual nature model of §6.2.2, Model 2.

1. The hyperparameters for Λ_G and **B** are obtained by maximising the likelihood of the subset of data, P**Y**, using the covariance matrix, $[\Lambda_G + \mathrm{P}\,\mathbf{B}\mathrm{P}^{\mathrm{T}}]$.
2. The hyperparameters for Λ_F and any hyperparameters for F not obtained in step 1 are obtained by maximising the likelihood of the data, **Y**, using the covariance matrix, $[\Lambda_F + \mathbf{B}]$.
3. An amplitude hyperparameter *a*, to rescale a_F and a_G is obtained by maximising, with respect to *a*, the likelihood of the data, **Y**, with the covariance matrix, $\left[a[\Lambda_F^{-1} + \mathrm{P}^{\mathrm{T}}\Lambda_G^{-1}\mathrm{P}]^{-1} + \mathbf{B}\right]$.

The rationale for the step 1 determination of the hyperparameters for Λ_G is provided by context attributes (a) and (b). The rationale for the step 1 determination of the noise hyperparameters and their subsequent use in step 2, is provided by the need for a common noise model and context attribute (c). Almost always all the noise hyperparameters are obtained in step 1, e.g. when **B** = *bI*. The values for the amplitude hyperparameters, a_G and a_F, determined in steps 1 and 2, are not necessarily appropriate for the dual nature model, see the discussion in §6.2.2 concerning the case with $\Lambda_G = \Lambda_F$. A rescaling hyperparameter, *a*, such that $a_F \to aa_F$ and $a_G \to aa_G$, is determined in step 3 to account for the interplay between the correlations with respect to **w** and **z**. To examine whether steps 1, 2 and 3

provide sensible hyperparameter values for the dual nature model over a wide range of circumstances, three disparate situations are considered below.

Situation 1: Consider the case in §6.2.2 when the Gaussian processes, $\mathrm{f}_{\mathbf{w}}$ and $\mathrm{g}_{\mathbf{z}}$ and their explanatory variables are the same and P = *I*, i.e. $\mathrm{g}_{\mathbf{z}}|_{\mathbf{z}=\mathbf{w}} = \mathrm{f}_{\mathbf{w}}$ and $\Lambda_{\mathrm{G}} = \Lambda_{\mathrm{F}}$. Applying step 1, the hyperparameter values for Λ_{G} and **B** are those that would pertain for the comparative benchmark, namely, Gaussian regression based on $\mathrm{g}_{\mathbf{w}}$ alone. Applying step 2, the hyperparameter values for Λ_{F} are identical to those of Λ_{G}. Similarly to the discussion in §6.2.2 of the amplitude hyperparameter values, applying step 3, the value of *a* is 2. Hence, in line with context attribute (b), the dual nature model exactly matches the comparative benchmark.

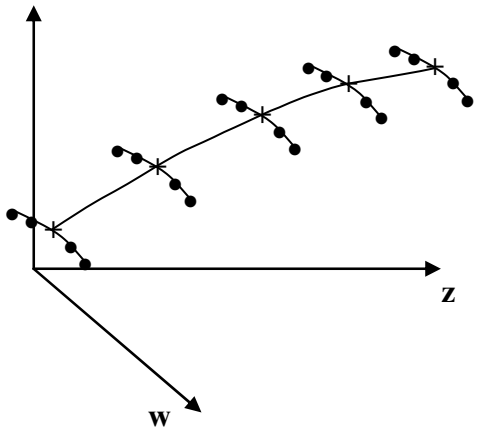


Figure 87 Projection of the data points along the surface.

Situation 2: Consider the situation that the correlations with respect to explanatory variable, **w**, between members of the data subset, $\{\mathrm{y}_{\mathrm{f}_j}, \mathbf{w}_j\}|_{j=k_i, i=1}^{\bar{N}}$, i.e. the subset common to both explanatory variables, are essentially zero. The situation is illustrated by Figure 87 for the case with both **w** and **z** scalar. The members of the common data subset are indicated by × and the remaining data with only explanatory variable, **w**, are indicated by •. The data points, ×, correlate with respect to the explanatory variable, **z**, as indicated by the connecting line. The data points that correlate with respect to the explanatory variable, **w**, to each member of the common data subset form disjoint subsets. These correlations are indicated by the short connecting line sections. The interplay between the correlations with respect to **w** and **z** is, thus, weak.

A natural way to exploit both aspects of the correlation in this situation would be the following. Firstly, apply Gaussian regression with respect to the explanatory variable, **w**, to the measured data. The prediction is

$$\widetilde{\mathrm{Y}} = \Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}} + \mathbf{B}]^{-1}\mathbf{Y} \tag{106}$$

where the hyperparameters for Λ_{F} and **B** are obtained by maximising the likelihood of the data, **Y**, using the covariance matrix $[\Lambda_{\mathrm{F}} + \mathbf{B}]$. The correlation between the additive noise components on $\mathrm{P}\,\widetilde{\mathrm{Y}}$, the subset of predictions corresponding to the common data subset, remains essentially zero; for example, when the noise covariance matrix for the measured data is $\mathbf{B} = bI$, the noise covariance for $\mathrm{P}\,\widetilde{\mathrm{Y}}$ is $\widetilde{\mathrm{B}}_{\mathrm{G}} = \mathrm{P}\,(\Lambda_{\mathrm{F}} - \Lambda_{\mathrm{F}}[\Lambda_{\mathrm{F}} + \mathbf{B}]^{-1}\Lambda_{\mathrm{F}})\mathrm{P}^{\mathrm{T}} \approx \widetilde{b}I$. However, its magnitude is reduced. Secondly, apply Gaussian regression with respect to the explanatory variable, **z**, to $\mathrm{P}\,\widetilde{\mathrm{Y}}$. The prediction is

$$\hat{\mathrm{Y}}_{\mathrm{G}} = \Lambda_{\mathrm{G}}\left[\Lambda_{\mathrm{G}} + \widetilde{\mathrm{B}}_{\mathrm{G}}\right]^{-1}\mathrm{P}\,\widetilde{\mathrm{Y}} \tag{107}$$

where the hyperparameters for Λ_{G} are obtained by maximising the likelihood of the data, $\mathrm{P}\,\widetilde{\mathrm{Y}}$, using the covariance matrix $[\Lambda_{\mathrm{G}} + \widetilde{\mathrm{B}}_{\mathrm{G}}]$. With the same hyperparameters, the predictions, (106) and (107), and the predictions, (97), for the dual nature model would be the same. Given context items (a) and (c), the hyperparameters values for Λ_{F}, Λ_{G} and **B**, obtained as above, would not differ greatly from those obtained by steps 1 and 2. Also, due to the interplay between the correlations with respect to **w** and **z** being minimal here, the value for hyperparameter, *a*, obtained by step 3, would mostly be approximately 1, particularly, when the dimension of $\mathrm{P}\mathbf{B}\mathrm{P}^{\mathrm{T}}$ is much less than the dimension of **B**. Hence, sensible values for the hyperparameters are obtained by steps 1, 2 and 3. Furthermore, in line with context attribute (b), the $\hat{\mathrm{Y}}_{\mathrm{G}}$ prediction in (97) is almost certainly better than the comparative benchmark, since the magnitude of the additive noise on $\widetilde{\mathrm{Y}}$ is smaller than on **Y**.

Situation 3: Consider the situation similar to *Situation 2* except that the correlations with respect to explanatory variable, **w**, between different members of the measured data set are essentially zero. Gaussian regression could be applied as above to obtain

the predictions, (106) and (107). When choosing the hyperparameters for Λ_F and **B** by maximising the likelihood using the covariance matrix, $[\Lambda_F + \mathbf{B}]$, the measured data, **Y**, could be interpreted to consist primarily of noise or, alternatively, almost noise free. In the former interpretation, the outcome would be low lengthscale hyperparameter values for Λ_F and high magnitude for the noise whilst, in the latter, the outcome would be high lengthscale hyperparameter values for Λ_F and low magnitude for the noise. Neither outcome is appropriate. Instead, only the correlations with respect to explanatory variable, **z**, should be exploited and that with respect to explanatory variable, **w**, ignored. The prediction for Y_G becomes

$$\hat{\mathrm{Y}}_G = \Lambda_G \left[\Lambda_G + \mathrm{P}\, \mathbf{B} \mathrm{P}^{\mathrm{T}} \right]^{-1} \mathrm{P}\, \mathbf{Y} \tag{108}$$

where the hyperparameters for Λ_G and **B** are obtained by maximising the likelihood of the data, P**Y**, using the covariance matrix, $[\Lambda_G + \mathrm{P}\, \mathbf{B} \mathrm{P}^{\mathrm{T}}]$. Given context items (a) and (c), the hyperparameters values for Λ_G and **B** thereby obtained and those obtained by step 1 would be the same. The amplitude hyperparameter value for Λ_F, obtained by step 2, would mostly be sufficiently high that $\widetilde{\mathrm{Y}} = \Lambda_F [\Lambda_F + \mathbf{B}]^{-1} \mathbf{Y} \approx \mathbf{Y}$ and $\widetilde{\mathrm{B}} = \Lambda_F [\Lambda_F + \mathbf{B}]^{-1} \mathbf{B} \approx \mathbf{B}$. Also, the value for hyperparameter, *a*, obtained by step 3, would be approximately 1 and the $\hat{\mathrm{Y}}_G$ prediction in (97) is similar to (108). Hence, sensible values for the hyperparameters are obtained by steps 1, 2 and 3. Furthermore, in line with context attribute (b), the $\hat{\mathrm{Y}}_G$ prediction in (97) is almost certainly similar to the comparative benchmark.

6.4 SSTS Models

An *SSTS Model* is a refinement of the dual nature model in §6.2.2. The explanatory variable, **w**, for the Gaussian process, $\mathrm{f}_{\mathbf{w}}$, is scalar and the values of the explanatory variable, at which the measurements are made, are a constant interval sequence; that is, $\{\mathbf{w}_i\}|_{i=1}^{N} = \{i\Delta\}|_{i=1}^{N}$. The *ij*-th elements of the selection matrix, P, are 1, when $j = ((i-1)m+1)$, for $m \geq 1$, and 0 elsewhere; that is, with respect to explanatory

variable, **w**, every *m*-th data point is used. The hyperparameter values are obtained by steps 1, 2 and 3 in §6.3.

Context attributes (a), (b) and (c), in §6.3, pertain to the applications that motivate the development of the SSTS model defined above, see §6.1. Hence, the rather pragmatic adoption of steps 1, 2 and 3 provides sensible hyperparameter values for a broad range of circumstances unlike those values arising from simply maximising the likelihood of the measured data using the covariance matrix, $\left[\left[\Lambda_{\mathrm{F}}^{-1}+\Lambda_{\mathrm{G}}^{-1}\right]^{-1}+\mathbf{B}\right]$ as discussed in §6.3. Since the ratio, $a_{\mathrm{w}}/a_{\mathbf{z}}$, is set by Steps 1 and 2, the dual nature of the model is preserved; that is, the SSTS model treats both interpretations of the data even-handedly. The time-series aspect of the model enables pre-filtering of the data in the time domain, which is to be incorporated into Gaussian regression when applied to nonlinear dynamic system identification. Of course, the choice of Gaussian processes for the time-series aspect of the SSTS model is determined by the type of pre-filtering required. In most cases, the pre-filtering corresponds to the extraction of a component of the data in which case the multiple Gaussian process models developed in Chapter 4 should be used.

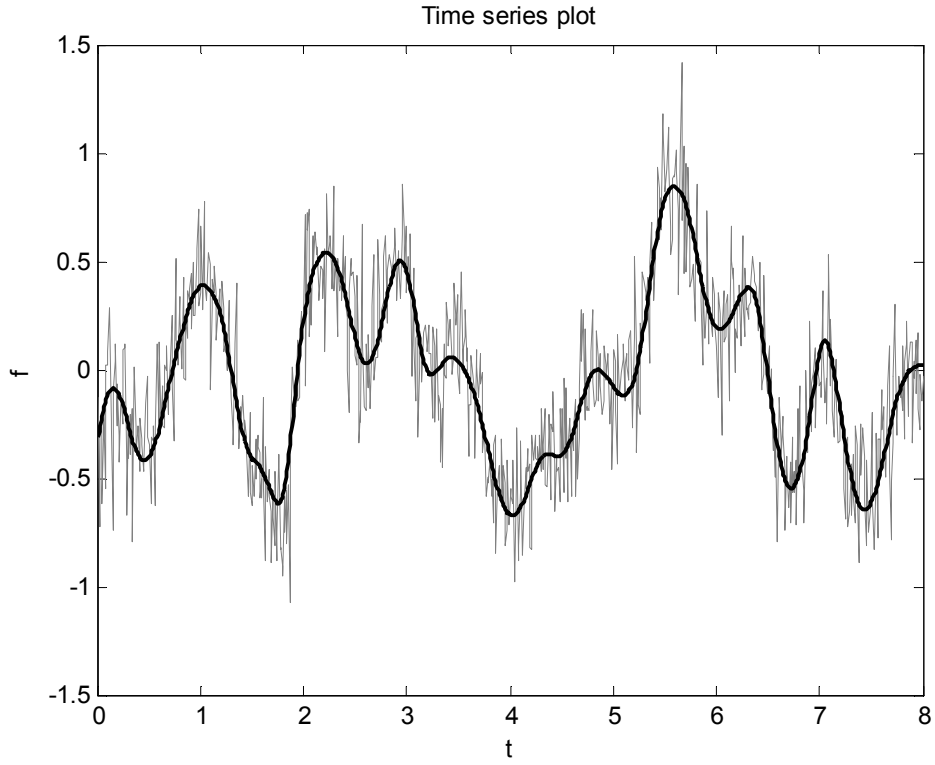


Figure 88 Time-series plots of original noise-free data and noisy measurement.

Example 6.1: The data set consists of 800 measurements at interval, Δ= 0.01, of the scalar explanatory variable w, with additive white Gaussian noise of variance 0.04. The noisy measurements and the noise-free data values are plotted against w in Figure 88. Explanatory variable $\mathbf{z} = (z_1, z_2) \in \mathrm{P}^2$ and its values for the data set are depicted in Figure 89. The noise-free data set is defined by the nonlinear function, $\tanh(z_1)\cos(0.8z_2)$. The covariance functions for $\mathrm{f_w}$ and $\mathrm{g_z}$ are, respectively,

$$C_{\mathrm{f}}(\mathrm{w}_i, \mathrm{w}_j) = \mathrm{E}[\mathrm{f}_{\mathrm{w}_i}, \mathrm{f}_{\mathrm{w}_j}] = a_{\mathrm{w}} \exp\left(-\frac{d_{\mathrm{w}}}{2}(\mathrm{w}_i - \mathrm{w}_j)^2\right)$$

and

$$C_{\mathrm{g}}(\mathbf{z}_i, \mathbf{z}_j) = \mathrm{E}[\mathrm{g}_{\mathbf{z}_i}, \mathrm{g}_{\mathbf{z}_j}] = a_{\mathbf{z}} \exp\left(-\frac{1}{2}(\mathbf{z}_i - \mathbf{z}_j)^{\mathrm{T}} D_{\mathbf{z}}(\mathbf{z}_i - \mathbf{z}_j)\right)$$

where $D_{\mathbf{z}} = \mathrm{diag}\{d_{\mathbf{z}_1}, d_{\mathbf{z}_2}\}$. The covariance matrix for the noise is $\mathbf{B} = bI$.

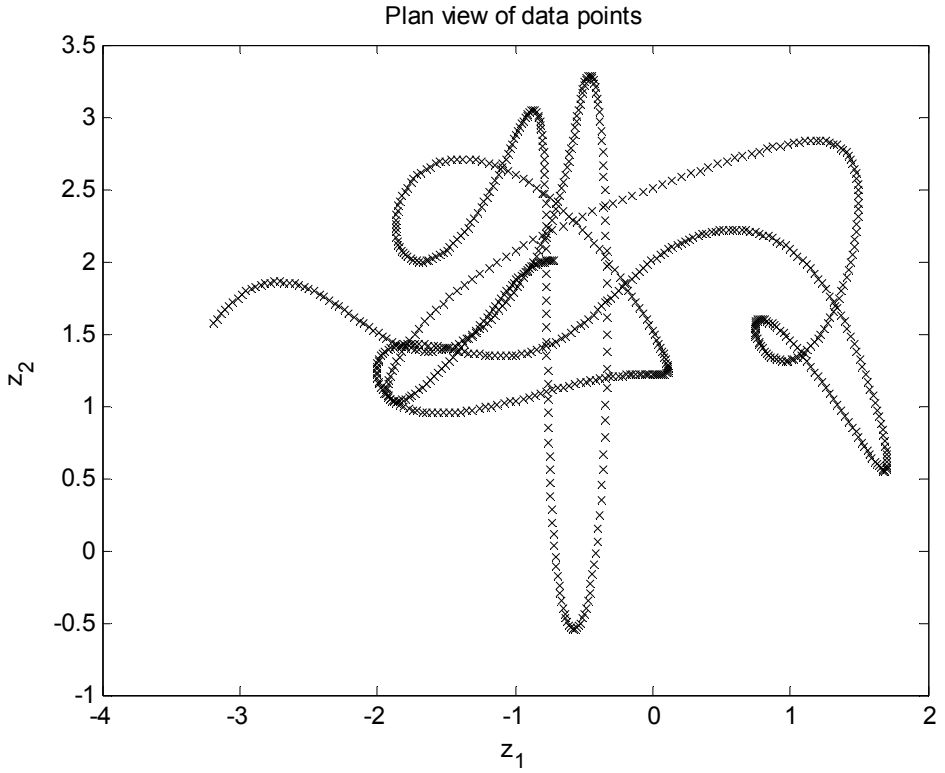


Figure 89 Plan view of data plot.

Before obtaining the hyperparameter values by steps 1, 2 and 3, the ineffectiveness, as discussed in §6.3, of obtaining amplitude hyperparameter values by maximising the likelihood of the measured data, **Y**, using the covariance matrix, $\left[\left[\Lambda_{\mathrm{F}}^{-1}+\Lambda_{\mathrm{G}}^{-1}\right]^{-1}+\mathbf{B}\right]$, is illustrated below.

TABLE V Hyperparameter values for Example 6.1

m	$a_{\mathbf{w}}$	$d_{\mathbf{w}}$	$a_{\mathbf{z}}$	$d_{\mathbf{z}_1}$	$d_{\mathbf{z}_2}$	b	$\ell\ell$
1 (ord=0)	0.1209	22.84				0.0100	1859.72
1 (ord=0)			0.4034	0.3227	0.2829	0.0101	1748.04

TABLE VI Hyperparameter values for Example 6.1

m	$a_{\mathbf{w}}$	$d_{\mathbf{w}}$	$a_{\mathbf{z}}$	$d_{\mathbf{z}_1}$	$d_{\mathbf{z}_2}$	b	$\ell\ell$
1 (ord=0)	5.646×10^5	22.84	0.4369	0.3227	0.2829	0.0100	1746.74
5 (ord=4)	1.170×10^5	22.84	0.4099	0.3073	0.3621	0.0101	1746.35
10 (ord=9)	69.57	22.84	0.4949	0.2692	0.3585	0.0101	1748.79
15 (ord=14)	1.710	22.84	0.4844	0.3068	0.3577	0.0101	1744.78
20 (ord=19)	1.144	22.84	0.4516	0.2117	0.3088	0.0100	1754.06

Table VII Hyperparameter values for Example 6.1

m	a_{w}	d_{w}	$a_{\mathbf{z}}$	$d_{\mathbf{z}_1}$	$d_{\mathbf{z}_2}$	b	$\ell\ell$
1 (ord=0)	3.337	22.84	10.64	0.3227	0.2829	0.0103	1763.64
5 (ord=4)	3.236	22.84	8.921	0.3073	0.3621	0.0103	1762.12
10 (ord=9)	1.117	22.84	4.575	0.2692	0.3585	0.0102	1752.17
15 (ord=14)	0.6514	22.84	1.647	0.3068	0.3577	0.0101	1749.72
20 (ord=19)	0.3490	22.84	2.192	0.2177	0.3088	0.0100	1755.40

Interpreting the data solely in terms of explanatory variable, w, the hyperparameters, (a_{w}, d_{w}, b), are obtained by maximising the likelihood of the measured data, **Y**, using the covariance matrix, $[\Lambda_{\mathrm{G}} + \mathbf{B}]$. The hyperparameter values are listed in TABLE V (the first row) together with the corresponding minimum value of the negative log-likelihood function, $\ell\ell$. The resulting single Gaussian process model is the time-series model. Interpreting the data solely in terms of explanatory variable, **z**, the hyperparameters, $(a_{\mathbf{z}}, d_{\mathbf{z}_1}, d_{\mathbf{z}_2}, b)$, are obtained by maximising the likelihood of the measured data, **Y**, using the covariance matrix, $[\Lambda_{\mathrm{F}} + \mathbf{B}]$. The hyperparameter values are again listed in TABLE V (the second row) together with the corresponding minimum value of $\ell\ell$. The resulting single Gaussian process model is the state-space model. In agreement with the nonlinear relationship, being by definition better explained in terms of the explanatory variable, **z**, rather than, w, the negative log-likelihood in the second row of TABLE V is smaller.

For m = 1, 5, 10, 15 and 20, the lengthscale hyperparameter values, $(d_{\mathrm{w}}, d_{\mathbf{z}_1}, d_{\mathbf{z}_2})$, are obtained by Steps 1 and 2 but the remaining hyperparameters, $(a_{\mathrm{w}}, a_{\mathbf{z}}, b)$, are obtained by maximising the likelihood of the measured data, **Y**, using the covariance matrix, $\left[[\Lambda_{\mathrm{F}}^{-1} + \Lambda_{\mathrm{G}}^{-1}]^{-1} + \mathbf{B}\right]$. The hyperparameter values are listed in TABLE VI together with $\ell\ell$, the negative log-likelihood function. For the reasons discussed previously, $a_{\mathrm{w}} / a_{\mathbf{z}}$ is very large for m = 1, since the covariance matrix, $[\Lambda_{\mathrm{G}} + \mathbf{B}]$, is the better explanation for the data, and remains large until m is large.

For m = 1, 5, 10, 15 and 20, the hyperparameter values, $(a_{\mathrm{w}}, d_{\mathrm{w}}, a_{\mathbf{z}}, d_{\mathbf{z}_1}, d_{\mathbf{z}_2}, b)$, obtained by Steps 1, 2 and 3 are listed in Table VII together with $\ell\ell$. By comparing the values of $\ell\ell$ in Table VII to the value in the second row of TABLE V, it can be

seen that the SSTS models, for all m, explain the data almost as well as the state-space model.

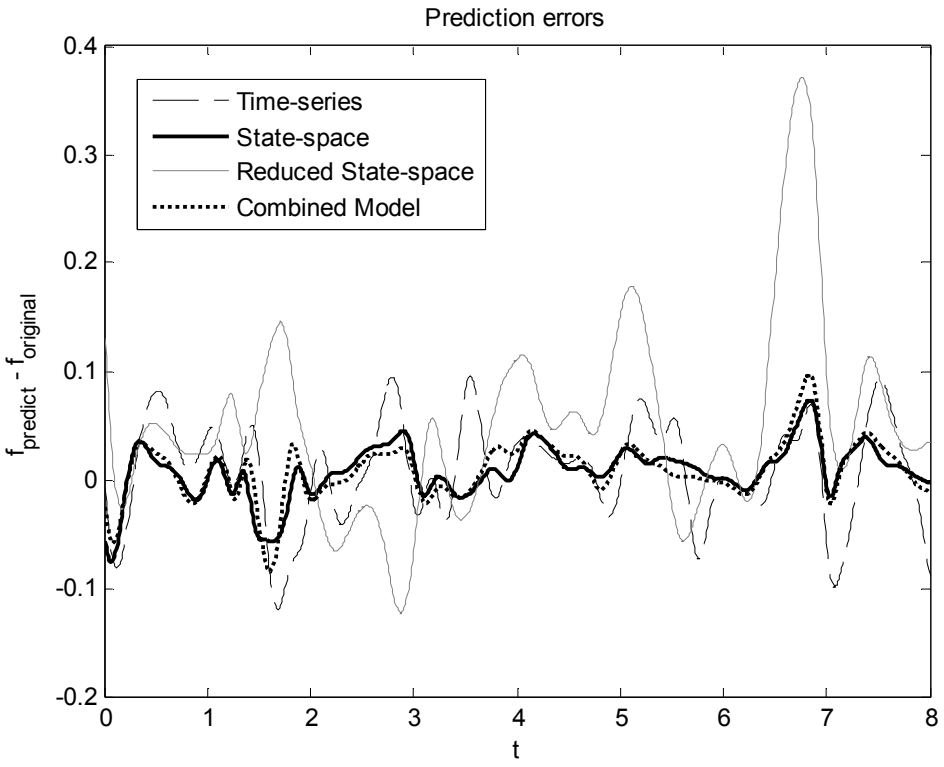


Figure 90 Comparison of various prediction errors.

The prediction errors for the time-series model, the full state-space model with $m = 1$, the reduced state-space model with $m = 10$ and the SSTS model with $m = 10$ are shown in Figure 90 and the corresponding confidence intervals in Figure 91. As would be expected the errors for the full state space model with $m = 1$ are considerably smaller than for the time-series model and the confidence interval for the former is considerably narrower than the latter. The errors for the reduced state-space model with $m = 10$ are very large and its confidence interval is very broad in comparison to both the time-series model and the full state-space model with $m = 1$. However, the errors for the SSTS model with $m = 10$ and its confidence interval are very similar to those of the full state space model with $m = 10$. Although, the number of data points contributing directly to the state-space interpretation has been drastically reduced to 80, each point is correlated to those nearby through the time-series interpretation. Hence, the information from the points omitted from the state-space interpretation is still available.

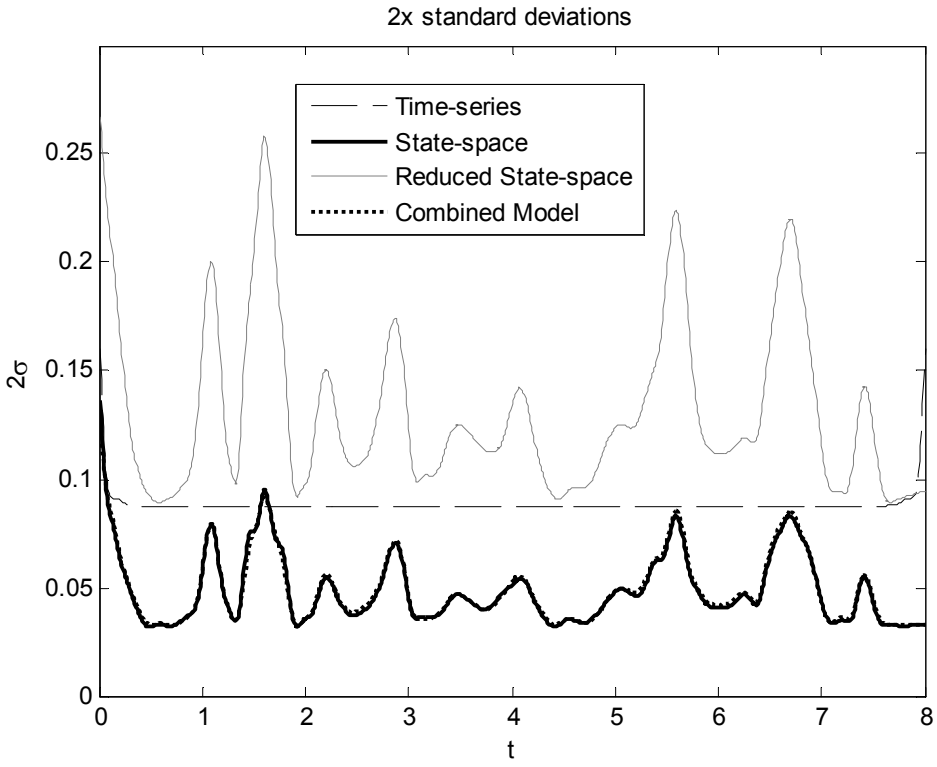


Figure 91 Comparison of various confidence intervals.

The covariance matrices for the time-series aspect of the SSTS model, being Toeplitz-like, are highly structured and so amenable to fast algorithms. When applying Gaussian regression to nonlinear dynamic system identification, data obtained away from equilibrium operating points are very likely to be transitory in nature. Hence, to obtain enough information, the data set often consists of a number of separate segments. Suitable fast algorithms based on Schur decomposition are investigated in Chapter 3. Unfortunately, no equivalent fast algorithms exist for the state-space aspect of the SSTS model.

The feasible size, $\overline{N}_f$, of the data set for a state-space model is considerably smaller, typically an order of magnitude smaller, than the feasible size, N_f, for a time-series model, since fast algorithms based on Schur decomposition exist; that is $\overline{N}_f << N_f$. However, when using an SSTS model, the size of the reduced data set for the state-space aspect might be $\overline{N}_f$ but the size of the full data set for the time-series aspect

might be N_f. Gaussian regression using such an SSTS model can be more accurate than Gaussian regression using a reduced state-space model based on the $\overline{N}_f$ state-space data points; see Example 6.1 in which $\overline{N}_f = 80$ and $N_f = 800$, such that the accuracy is almost identical to Gaussian regression using the full state-space model with 800 data points. The extent, to which expanding the data set in this manner, to improve the accuracy of the model depends on the nature of the data set, see Situations 2 and 3 in §6.3, and so on experimental design. When applying Gaussian regression to nonlinear dynamic system identification, in addition to pre-filtering, the SSTS models can thus be used to increase the size of the data sets and improve the accuracy of the models.

The SSTS model, particularly when combined with the multiple Gaussian process models of Chapter 4 and the fast algorithms of Chapter 3, is proposed as the appropriate model when applying Gaussian regression to nonlinear dynamic system identification. It treats both the state-space and time-series information even-handedly to enable pre-filtering of the data and enables larger data sets to be used to increase the resolution of the models.

6.5 Application of Fast Algorithm

This section explains the vital steps to integrate fast algorithms into the SSTS model. Matrix operations that can be exploited using fast algorithms (Leithead and Zhang, 2005) are highlighted below.

1. $\mathrm{T}^{-1}b$, where $\mathrm{T} \in \mathrm{P}^{N\times N}$ is a Toeplitz matrix and $b \in \mathrm{P}^{N\times 1}$ is a vector.
2. $|\mathrm{T}|$, where $|.|$ is the determinant operator on the Toeplitz matrix, T.
3. $\mathrm{PT}^{-1}\mathrm{P}^{\mathrm{T}}$, where a direct factorisation for T^{-1} is obtained through the use of the generalised Schur algorithm.
4. $\mathrm{PT}^{-1}\mathrm{T}^{-1}\mathrm{P}^{\mathrm{T}}$, where a direct factorisation for can $\mathrm{T}^{-1}\mathrm{T}^{-1}$ be obtained.
5. $\mathrm{PT}^{-1}\Pi\mathrm{T}^{-1}\mathrm{P}^{\mathrm{T}}$, to obtain a direct factorisation for $\mathrm{T}^{-1}\Pi\mathrm{T}^{-1}$ where Π is a Toeplitz matrix.

Following the matrix inversion lemma and matrix symmetry property, the inverse of covariance function (93) can be written as

$$\widetilde{Q}^{-1} = \mathbf{Q}_{\mathrm{FG}}^{-1} = \begin{bmatrix} \overbrace{\Phi_{11}}^{N} & \overbrace{\Phi_{21}^{\mathrm{T}}}^{n_1} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{matrix} \}N \\ \}n_1 \end{matrix}, \text{ where}$$

$$\Phi_{11} = Q_f^{-1} + b^2 Q_f^{-1}\mathrm{P}^{\mathrm{T}}\left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\mathrm{P}\,Q_f^{-1}$$

$$\Phi_{12} = -bQ_f^{-1}\mathrm{P}^{\mathrm{T}}\left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}$$

$$\Phi_{21} = -b\left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\mathrm{P}\,Q_f^{-1} = \Phi_{12}^{\mathrm{T}}$$

$$\Phi_{22} = \left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}$$

such that $Q_f = \Lambda_{\mathrm{F}} + b(1+\varepsilon)I$ and $Q_g = \mathrm{P}\left[\Lambda_{\mathrm{G}} + b(1+\varepsilon)I\right]\mathrm{P}^{\mathrm{T}}$, where $\mathbf{B} = bI$ and ε is a small scalar term introduced to stabilise the matrix inversion operation. To avoid $O(N^2)$ storage-level demand, large intermediate matrices are never explicitly stored, except for $\Phi_{22} \in \mathrm{P}^{n_1 \times n_1}$, which is obtained using the generalised Schur algorithm.

Remark 6.1: The matrix $\Phi_{22} = \left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1} \in \mathrm{P}^{n_1 \times n_1}$ is of small size, thus the inverse of $\left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)$ is done explicitly using any mathematical tool such as MATLAB. To obtain Φ_{22}, the matrix $\left(\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right) \in \mathrm{P}^{n_1 \times n_1}$ is first computed using the generalised Schur algorithm. It is possible to obtain a direct factorisation for Q_f^{-1}, as mentioned in Chapter 3.5.2, therefore only a slight modification to the output of the result from the Schur algorithm is required as P is a projection matrix.

The calculation of the log-likelihood function requires the formulation of these two essential terms, i.e. the log determinant of $\widetilde{Q}$ and $\widetilde{Q}^{-1}\widetilde{\mathbf{W}}$, where $\widetilde{\mathbf{W}} = \left[\mathbf{Y}^{\mathrm{T}} \quad (\mathrm{P}\,\mathbf{Y})^{\mathrm{T}}\right]^{\mathrm{T}}$. Using the following matrix property,

$$\log\left|\widetilde{Q}\right| = \log\left|Q_f\right| + \log\left|Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right|$$

The term, $\left|Q_g - b^2 \mathrm{P}\, Q_f^{-1} \mathrm{P}^{\mathrm{T}}\right|$, is easily computed since it is of small size. The term, $\left|Q_f\right|$, requires the use of fast algorithms (See Chapter 3 for the procedure to obtain the log-determinant of a Toeplitz-like covariance matrix). Therefore, both are easily obtained and hence, $\log\left|\widetilde{Q}\right|$.

Next, let $\left[X^{\mathrm{T}} \,\vdots\, Y^{\mathrm{T}}\right]^{\mathrm{T}} = \widetilde{Q}^{-1}\widetilde{\mathbf{W}}$. It follows that we have

$$Y = \left[Q_g - b^2 \mathrm{P}\, Q_f^{-1} \mathrm{P}^{\mathrm{T}}\right]^{-1}\left(\mathrm{P}\,\mathbf{Y} - b\mathrm{P}\, Q_f^{-1}\mathbf{Y}\right) = \Phi_{22}\left(\mathrm{P}\,\mathbf{Y} - b\mathrm{P}\,\overbrace{Q_f^{-1}\mathbf{Y}}^{*}\right)$$

$$X = Q_f^{-1}\mathbf{Y} - bQ_f^{-1}\mathrm{P}^{\mathrm{T}}\left[Q_g - b^2 \mathrm{P}\, Q_f^{-1} \mathrm{P}^{\mathrm{T}}\right]^{-1}\left(\mathrm{P}\,\mathbf{Y} - b\mathrm{P}\, Q_f^{-1}\mathbf{Y}\right) = \overbrace{Q_f^{-1}\mathbf{Y}}^{*} - b\overbrace{Q_f^{-1}\mathrm{P}^{\mathrm{T}}Y}^{*}$$

The vectors, *X* and *Y*, are efficiently determined using fast algorithms (terms marked with '*' requires the use of fast algorithms) and some matrix-vector operations (linear-algebra). The computation of $Q_f^{-1}\mathbf{Y}$ is straightforward using either modified Durbin-Levinson's algorithm or the modified generalised Schur algorithm. The computation of $Q_f^{-1}\mathrm{P}^{\mathrm{T}}Y$ requires *Y* to be obtained first. A projection matrix P is then applied to *Y* before computing the inverse matrix operation using fast algorithm. These two vectors, *X* and *Y*, are stored for later calculations. It follows that the negative log-likelihood function can be written as

$$L = \frac{1}{2}\left\{\log\left|Q_f\right| + \log\left|Q_g - b^2 \mathrm{P}\, Q_f^{-1} \mathrm{P}^{\mathrm{T}}\right| + \widetilde{\mathbf{W}}\left[X^{\mathrm{T}} \,\vdots\, Y^{\mathrm{T}}\right]^{\mathrm{T}}\right\}$$

Gradient information is often included in optimisation routines to speed up training procedures. As hyperparameters are constrained to be positive scalars, they can be modified to take exponential powers, e.g. $a = e^{\alpha}$, $b = e^{\beta}$ and $d_i = e^{\gamma_i}$, $\forall i = 1,...,K$. The hyperparameters for the SSTS covariance function are $\theta = \left\{\alpha_t, \alpha_s, \beta, \gamma_t, \gamma_{s,k}\right\}$, $\forall k = 1,...,K$ such that subscripts *t* and *s* refer to the time-series and state-space components, respectively. (Refer to Chapter 2.6.1.1 for the definition of these hyperparameters). The gradient is the first order derivative of the log-likelihood function with respect to the hyperparameters. The derivatives of the covariance matrix are simplified as shown

$$\frac{\partial \widetilde{Q}}{\partial \alpha_t} = \left[\begin{array}{c|c} \Lambda_{\mathrm{F}} & 0 \\ \hline 0 & 0 \end{array}\right]$$

$$\frac{\partial \widetilde{Q}}{\partial \alpha_s} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \Lambda_{\mathrm{G}} \end{array}\right]$$

$$\frac{\partial \widetilde{Q}}{\partial \beta} = e^{\beta}\left[\begin{array}{c|c} I & \mathrm{P}^{\mathrm{T}} \\ \hline \mathrm{P} & I \end{array}\right]$$

$$\frac{\partial \widetilde{Q}}{\partial \gamma_t} = \left[\begin{array}{c|c} \partial\Lambda_{\mathrm{F}}/\partial\gamma & 0 \\ \hline 0 & 0 \end{array}\right]$$

$$\frac{\partial \widetilde{Q}}{\partial \gamma_{s,k}} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \partial\Lambda_{\mathrm{G}}/\partial\gamma_{s,k} \end{array}\right]$$

Clearly, the first order derivative of the log-likelihood function can be separated into two terms, i.e. $\mathfrak{l}(\theta_i) = tr\left(\widetilde{Q}^{-1}\frac{\partial \widetilde{Q}}{\partial \theta_i}\right)$ and $\wp(\theta_i) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{Q}^{-1}\left(\frac{\partial \widetilde{Q}}{\partial \theta_i}\right)\widetilde{Q}^{-1}\widetilde{\mathbf{W}}$. For notation simplicity, let $P_t = \frac{\partial \widetilde{Q}}{\partial \gamma_t}$ and $P_{s,k} = \mathrm{P}\,\frac{\partial \widetilde{Q}}{\partial \gamma_{s,k}}\mathrm{P}^{\mathrm{T}} = \frac{\partial}{\partial \gamma_{s,k}}\left(\mathrm{P}\,\widetilde{Q}\mathrm{P}^{\mathrm{T}}\right)$, $\forall k = 1,\ldots,K$. Simplifying the terms for $\wp(\theta_i)$, we have

$$\wp(\alpha_t) = X^{\mathrm{T}}\Lambda_{\mathrm{F}}X$$

$$\wp(\alpha_s) = Y^{\mathrm{T}}\Lambda_{\mathrm{G}}Y$$

$$\wp(\beta) = b\left\{X^{\mathrm{T}}X + Y^{\mathrm{T}}Y + 2Y^{\mathrm{T}}(\mathrm{P}\,X)\right\}$$

$$\wp(\gamma_t) = X^{\mathrm{T}}P_tX$$

$$\wp(\gamma_{s,k}) = Y^{\mathrm{T}}P_{s,k}Y$$

The terms for $\mathfrak{l}(\theta_i)$ are obtained in a slightly more complicated way.

$$\begin{aligned}
\mathfrak{l}(\alpha_t) &= tr\{\Phi_{11}\Lambda_{\mathrm{F}}\} \\
&= tr\left\{Q_f^{-1}\Lambda_{\mathrm{F}}\right\} + b^2 tr\left\{\left(\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\right\} \\
&\qquad -b^2(b+\varepsilon)tr\left\{\left(\mathrm{P}\,Q_f^{-1}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\left(Q_g - b^2\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\right\} \\
&= tr\left\{Q_f^{-1}\Lambda_{\mathrm{F}}\right\} + b^2 tr\left\{\left(\mathrm{P}\,Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\Phi_{22}\right\} - b^2(b+\varepsilon)tr\left\{\left(\mathrm{P}\,Q_f^{-1}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\Phi_{22}\right\}
\end{aligned}$$

$$\mathfrak{l}(\alpha_s) = tr\{\Phi_{22}\Lambda_{\mathrm{G}}\}$$

$$
\begin{aligned}
\mathfrak{\lambda}(\beta) &= b\left(tr\{\Phi_{11}+\Phi_{12}\mathrm{P}^{\mathrm{T}}\}+tr\{\Phi_{21}\mathrm{P}^{\mathrm{T}}+\Phi_{22}\}\right)\\
&= b\left[tr\left(Q_f^{-1}\right)+tr\left\{\left(Q_g-b^2\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\right\}\right]+b^3tr\left\{\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\left(Q_g-b^2\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\right\}\\
&\qquad -2b^2tr\left\{\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\left(Q_g-b^2\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\right\}\\
&= b\left[tr\left(Q_f^{-1}\right)+tr\{\Phi_{22}\}\right]+b^3tr\left\{\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\Phi_{22}\right\}-2b^2tr\left\{\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\Phi_{22}\right\}\\
\mathfrak{\lambda}(\gamma_t) &= tr\{\Phi_{11}P_t\}\\
&= tr\left\{Q_f^{-1}P_t\right\}+b^2tr\left\{\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}P_tQ_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\left(Q_g-b^2\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)^{-1}\right\}\\
&= tr\left\{Q_f^{-1}P_t\right\}+b^2tr\left\{\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}P_tQ_f^{-1}\mathrm{P}^{\mathrm{T}}\right)\Phi_{22}\right\}\\
\mathfrak{\lambda}(\gamma_{s,k}) &= tr\{\Phi_{22}P_{s,k}\}
\end{aligned}
$$

Using the generalised Schur algorithm, direct factorisations for Q_f^{-1}, $Q_f^{-1}Q_f^{-1}$ and $Q_f^{-1}P_tQ_f^{-1}$ are obtained for the computations of $\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)$, $\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}Q_f^{-1}\mathrm{P}^{\mathrm{T}}\right)$ and $\left(\mathrm{P}^{\mathrm{T}}Q_f^{-1}P_tQ_f^{-1}\mathrm{P}^{\mathrm{T}}\right)$, respectively (Refer to Chapter 3.6.3 for the choice of augmentation matrices and algorithms for these computations). Three augmentation matrices are required for the use of the generalised Schur algorithm.

$$
\left[\begin{array}{c|c} -Q_f & I \\ \hline I & 0 \end{array}\right],\quad
\left[\begin{array}{c|c|c} Q_f+b(1+\varepsilon)I & Q_f & 0 \\ \hline Q_f & 0 & I \\ \hline 0 & I & 0 \end{array}\right],\quad
\left[\begin{array}{c|c|c} Q_f+P_t & Q_f & 0 \\ \hline Q_f & 0 & I \\ \hline 0 & I & 0 \end{array}\right]
$$

The Schur complements formulated from these matrices are Q_f^{-1}, $Q_f^{-1}+b(1+\varepsilon)Q_f^{-1}Q_f^{-1}$ and $Q_f^{-1}+Q_f^{-1}P_tQ_f^{-1}$, respectively. Since P is a projection matrix, the essential terms for $\wp(\theta_i)$ and $\mathfrak{\lambda}(\theta_i)$ are now of computable sizes. The first order derivative with respect to hyperparameter, θ_i, is given by

$$\frac{\partial L}{\partial \theta_i}=\frac{1}{2}\mathfrak{\lambda}(\theta_i)-\frac{1}{2}\wp(\theta_i)$$

In Gaussian processes, the posterior joint probability distributions are defined by the mean function and covariance function. Given that the posterior is $\overline{\mathbf{W}}=\Lambda_{21}^{\mathrm{T}}\widetilde{Q}^{-1}\mathbf{W}$, it can be simplified as shown

$$\overline{\mathbf{W}}=\left[\begin{array}{c|c} \Lambda_t & 0 \\ \hline 0 & \Lambda_z \end{array}\right]^{\mathrm{T}}\left[\begin{array}{c} X \\ \hline Y \end{array}\right]=\left[\begin{array}{c} \Lambda_t^{\mathrm{T}}X \\ \hline \Lambda_z^{\mathrm{T}}Y \end{array}\right]$$

where the i^{th} elements of vectors Λ_t and Λ_z, respectively are $\mathrm{E}[y_i \mathrm{f}_t]$ and $\mathrm{E}[y_i \mathrm{g}_z]$. These operations require only matrix-vector manipulations; hence no fast algorithm is necessary. However, the generalised Schur algorithm is employed for the computation of the posterior covariances of the SSTS model,

$$\begin{aligned}\overline{\Lambda} &= \Lambda_{11} - \Lambda_{21}^{\mathrm{T}} \widetilde{Q}^{-1} \Lambda_{21} \\ &= \begin{bmatrix} \Lambda_{\hat{t}} & 0 \\ 0 & \Lambda_{\hat{z}} \end{bmatrix} - \begin{bmatrix} \Lambda_t & 0 \\ 0 & \Lambda_z \end{bmatrix}^{\mathrm{T}} \widetilde{Q}^{-1} \begin{bmatrix} \Lambda_t & 0 \\ 0 & \Lambda_z \end{bmatrix}\end{aligned}$$

where $\Lambda_{\hat{t}}$ is $\mathrm{E}[\mathrm{f}_t \mathrm{f}_t]$ and $\Lambda_{\hat{z}}$ is $\mathrm{E}[\mathrm{g}_z \mathrm{g}_z]$. Although the posteriors are obtainable in either time-series domain or state-space domain, it is paramount to compute both of them. The next objective is to calculate the standard deviation, $\sigma = \sqrt{diag(\overline{\Lambda})}$. Let $\Omega_{tt} = \mathrm{E}[\mathrm{f}_t \mathrm{f}_t]$, $\Omega_{zz} = \mathrm{E}[\mathrm{g}_z \mathrm{g}_z]$, $\Omega_{yt} = \mathrm{E}[\mathbf{Y} \mathrm{f}_t]$ and $\Omega_{yz} = \mathrm{E}[(\mathrm{P}\ \mathbf{Y}) \mathrm{g}_z]$, it follows that the corresponding outcomes are

$$\sigma_t = \left[diag(\Omega_{tt}) - diag\left(\Omega_{yt}^{\mathrm{T}} \Phi_{11} \Omega_{yt}\right) \right]^{\frac{1}{2}}$$

$$\sigma_z = \left[diag(\Omega_{zz}) - diag\left(\Omega_{yz}^{\mathrm{T}} \Phi_{22} \Omega_{yz}\right) \right]^{\frac{1}{2}}$$

where $\sigma = \left[\sigma_t^{\mathrm{T}} \mid \sigma_z^{\mathrm{T}} \right]^{\mathrm{T}}$. The intermediate matrix operations of σ_z are of small dimension and therefore, can be computed explicitly without the need of fast algorithms.

Since the covariance function is of the form (15), σ_t and σ_z can be simplified,

$$diag\{\Omega_{tt}\} = diag\{\mathrm{E}[\mathrm{f}_t \mathrm{f}_t]\} = a_t = e^{\alpha_t} = a_{\mathrm{w}}$$

$$diag\{\Omega_{zz}\} = diag\{\mathrm{E}[\mathrm{g}_z \mathrm{g}_z]\} = a_s = e^{\alpha_s} = a_{\mathbf{z}}$$

$$\begin{aligned} diag\left\{\mathrm{E}[\mathrm{f}_t \mathbf{y}^{\mathrm{T}}] \Phi_{11} \mathrm{E}[\mathbf{y} \mathrm{f}_t]\right\} &= diag\left\{\Omega_{yt}^{\mathrm{T}} \Phi_{11} \Omega_{yt}\right\} \\ &= diag\left\{\Omega_{yt}^{\mathrm{T}} Q_f^{-1} \Omega_{yt}\right\} + b^2 diag\left\{\left(\Omega_{yt}^{\mathrm{T}} Q_f^{-1} \mathrm{P}^{\mathrm{T}}\right) \Phi_{22} \left(\mathrm{P}\ Q_f^{-1} \Omega_{yt}\right)\right\}\end{aligned}$$

The matrix, $\left(\mathrm{P}\ Q_f^{-1} \Omega_{yt}\right)$, is first computed using the generalised Schur algorithm, with the use of the following augmentation matrix,

$$\begin{bmatrix} -Q_f & \Omega_{yt} & I \\ \Omega_{yt}^{\mathrm{T}} & 0 & 0 \\ I & 0 & 0 \end{bmatrix}$$

which provides a fast factorisation for the computations of $\left(\Omega_{yt}^{\mathrm{T}} Q_f^{-1} \Omega_{yt}\right)$ and $\left(\mathrm{P}\ Q_f^{-1} \Omega_{yt}\right)$. σ_t is then obtained with a few additional arithmetic steps.

6.6 Application of Dynamic Lengthscale

Proper training procedures have often been overlooked. In the case of *likelihood maximisation*, hyperparameters are adapted to maximise the likelihood function. However, it may not necessarily be an appropriate approach in some situations, such as the SSTS model. In the context of the SSTS Gaussian process model, great importance is placed upon proper training to ensure that optimisation is optimal, practical and efficient.

Consider that the hyperparameters of the prior model are adapted to maximise the log-likelihood function. These hyperparameters are denoted a_t and d_t for the time-series component, a_s and D_s for the state-space component, where $D_s = diag\left\{d_1^s, d_2^s, ..., d_p^s\right\}$, and b as the noise variance. Assume that data containing m-dimensional input explanatory variable, $\mathbf{z} \in \mathrm{P}^{\,m}$, with measurement output containing noise, $\{\mathbf{z}_i, y_i\}_{i=1}^N$, is sampled at fixed interval. The time-series explanatory variable, $t \in \mathrm{P}$, can effectively be any time-series sequence $\{t_i, y_i\}_{i=1}^N$. Furthermore, in the SSTS model, the time-series component comprises of the full data and the state-space component holds only partial data.

6.6.1 Training SSTS Model

To achieve a balance between accuracy and efficiency of the training procedure, yet maintaining the posterior joint probability distribution of the combined SSTS

Gaussian process prior model, the following method is introduced to train the hyperparameters.

Algorithm 6.1 (Modified training algorithm):

1. Train the hyperparameters $\{a_s, D_s, b_s\}$ for the reduced state-space component using the squared exponential covariance function (17) in a single Gaussian process.
2. Also, train the time-series hyperparameters $\{a_t, d_t, b_t\}$ using the squared exponential covariance function, but with time parameter as explanatory variable.
3. Apply *Algorithm 6.2* to obtain d_{ts}, the dynamic lengthscale hyperparameter.
4. Fix the hyperparameters $\{a_s, D_s, d_{\text{ts}}\}$ and train the remaining hyperparameters of the compound covariance function. Values of the hyperparameters, $\{a_t, b_t\}$, obtained from Step 2 are used as initial values for the optimisation routine.

In doing so, the loss of generality is minimal since the state-space hyperparameters are fixed, which ensures the surface curvature of the nonlinear space mapping is maintained. On the contrary, it is illogical to fix the time-series hyperparameters, since the time-series measurements do not have a unique space mapping feature (the lengthscale characteristics may vary greatly, unlike the state-space map).

6.6.2 Dynamic Lengthscale Algorithm

An algorithm, capable of discriminating between fast-varying and slow-varying regions, is introduced here.

Algorithm 6.2 (Dynamic lengthscale algorithm): The algorithm describes the steps to obtain appropriate lengthscale hyperparameter for the time-series component in the SSTS Gaussian process prior model.

1. Train the time-series data set, $\{t_i, y_i\}_{i=1}^{N}$, using a single Gaussian process with the incorporation of the modified Durbin-Levinson's algorithm (Leithead et al., 2005c), to obtain the lengthscale hyperparameter, d_{ts}.
2. Next, consider a modified time-series data, corresponding to the reduced state-space dataset, i.e., $\{\bar{t}_k, \bar{y}_k\}_{k=1}^{K}$.
3. Obtain the K datasets as follow:
   a. Use d_{ts} to calculate the amount of data required for any contribution to the data point at k, i.e. $\{(\bar{t}_k \pm \Delta\tau), (\bar{y}_k \pm \Delta y)\}$, where $\bar{y}_k \pm \Delta y$ are the output measurements corresponding to the input explanatory variable at $\bar{t}_k \pm \Delta\tau$. Assume that squared exponential covariance function (17) is used throughout the employment of SSTS Gaussian process, let $\exp\left\{-\frac{d_{\text{ts}}}{2}(\bar{t}_i - \bar{t}_j)^2\right\} \geq \chi$, where χ is a fraction of the normalised probability distribution contributed to the data point at k.
   b. Reformulating the inequality, we have $\Delta\tau \leq \sqrt{-2\ln(\chi)/d_{\text{ts}}}$, such that the number of neighbouring data points required is the maximum integer value of $\Delta\tau/\Delta m$, where Δm is the time-series sampling interval.
   c. Hence, $[\{\bar{t}_k - \Delta\tau, \bar{t}_k + \Delta\tau\}, \{\bar{y}_k - \Delta y, \bar{y}_k + \Delta y\}]$ is the new dataset at k.
4. Obtain a list of dynamic lengthscale hyperparameters from training these K sub-datasets, i.e., $\hat{d} = \{d_k^1, d_k^2, ..., d_k^K\}$.
5. The maximum value of $\hat{d}$ is chosen. If this value is less than d_{ts}, then the latter is used; otherwise the former is used to invoke the SSTS Gaussian process prior model.

This algorithm may also be seen as segmenting of the time-series dataset. The key to this algorithm lies in the imposition of the criterion of χ.

Remark 6.2: If $\{\bar{t}_k, \bar{y}_k\}$ is located at the edge of the full time-series dataset, then these remaining fewer data points, $\left[\{\bar{t}_k - \Delta\tau, \bar{t}_k + \Delta\tau\}, \{\bar{y}_k - \Delta y, \bar{y}_k + \Delta y\}\right]$, will be used.

Remark 6.3: A typical choice of χ is about 0.1%. Too large value will induce under estimation of the noise intensity; too small will result in over smoothing of the data, with vital information filtered away.

The *dynamic lengthscale algorithm* is rather swift and efficient since fast algorithm is employed to handle time-series Gaussian processes. With this method, only two hyperparameters are adapted to maximise the log-likelihood function.

6.6.3 Graphical Representation of Dynamic Lengthscale

To substantiate *Algorithm 6.2*, a simple mechanism is introduced to justify the technique of choosing the maximum hyperparameter value from the list of dynamic lengthscales. Two data points, each with distinct level of smoothness, are selected from the time-series dataset for comparison; one from a fast-varying region, and the other from a slow-varying region. The kernels for these two particular data points are evaluated in their respective time-series and state-space domains, i.e. $\Lambda_{21}^{\mathrm{T}} Q^{-1}$ where $Q = \mathrm{E}\left[\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\right] + \mathbf{B}$ and $\Lambda_{21} = \mathrm{E}\left[\mathrm{f}_z \mathbf{Y}^{\mathrm{T}}\right]$ with $\mathbf{B} = b\delta_{ij}$ as the noise covariance matrix, and f_z being the selected data point.

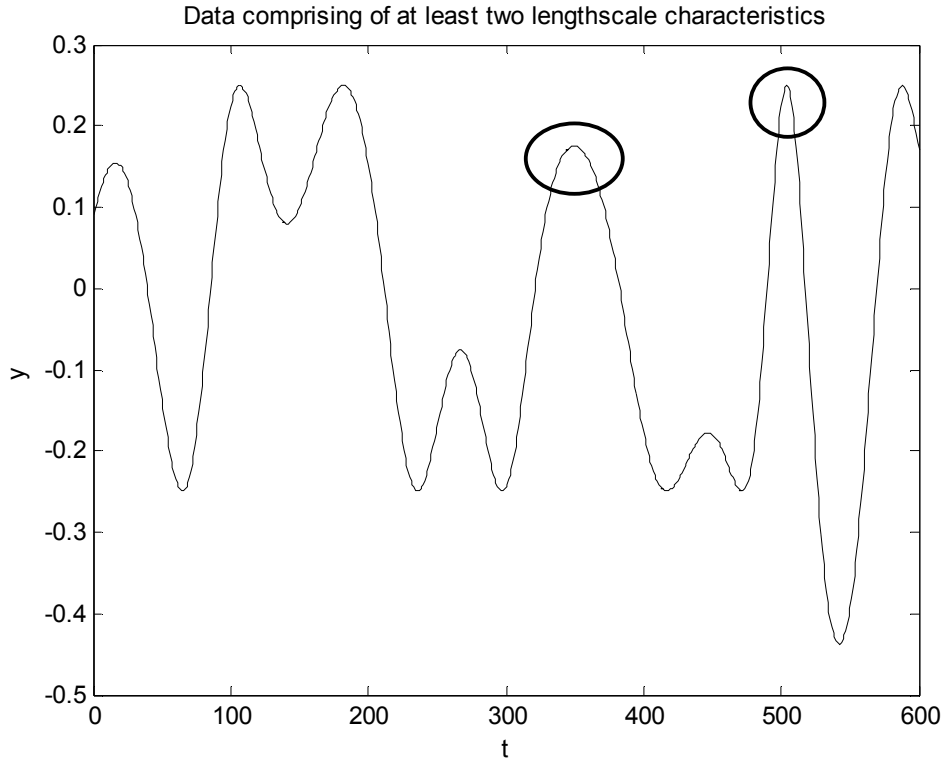
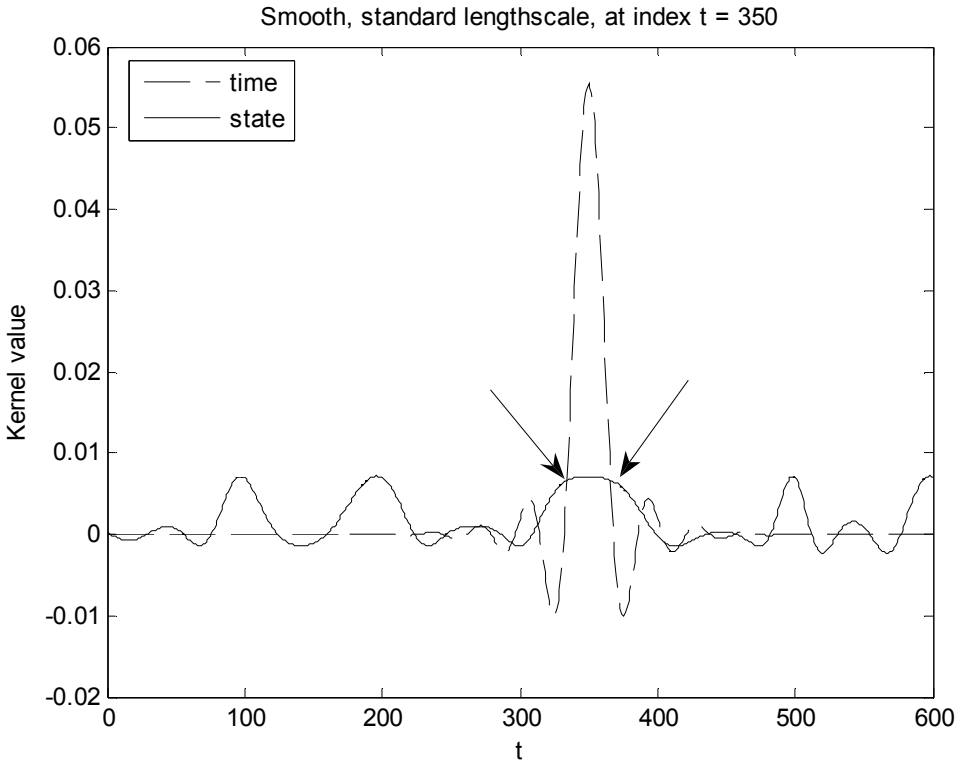


Figure 92 Data representing dynamically-varying lengthscale characteristics. This particularly toy example does not posses a fixed lengthscale.

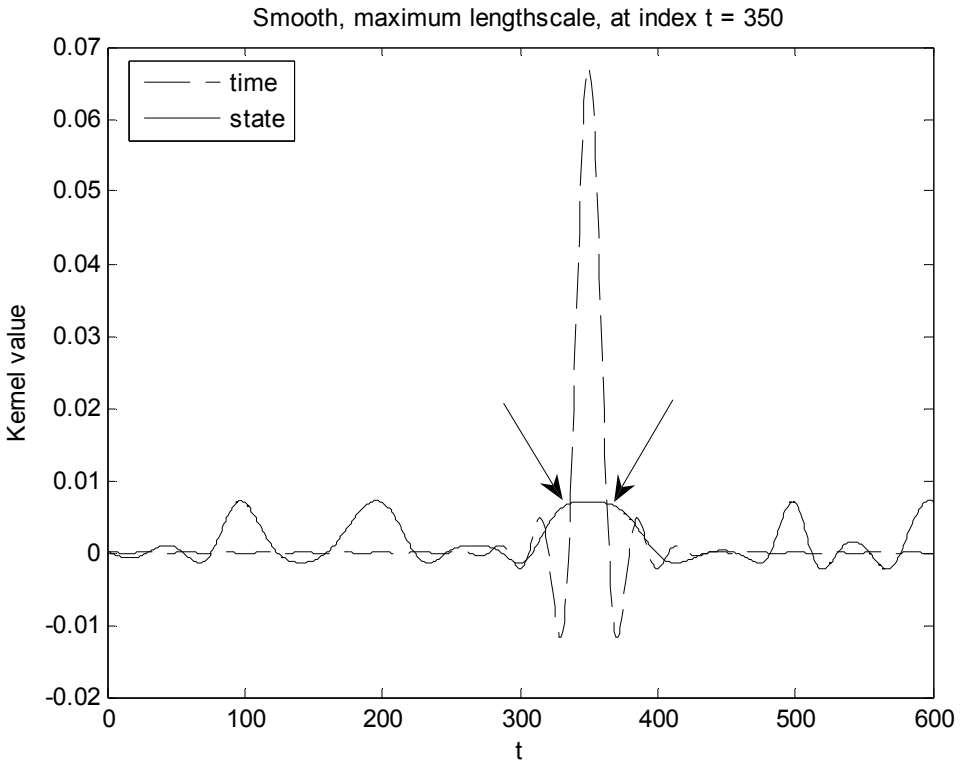
An example is chosen to illustrate this mechanism. A 600-point data is arbitrarily-generated to illustrate its composition of multiple lengthscales. As shown in Figure 92, the data has a point in the fast-varying region and another in the slow-varying region, at i = 504 and i = 350, respectively. Two datasets, i.e. $\{\mathbf{z}_i, y_i\}_{i=1}^N$ and $\{t_i, y_i\}_{i=1}^N$, are available and trained using covariance function (17). The adapted lengthscale hyperparameter for the time-series dataset is d_{ts} = 8.2321. Using the dynamic lengthscale algorithm (*Algorithm 6.2*), the lengthscale hyperparameter obtained for the time-series dataset is d_{dyn} = 13.5665. The smoothing kernels are then computed at these two locations.

Figure 93 shows the slow-varying region whereas Figure 94 illustrates the fast-varying region. For a larger lengthscale hyperparameter value, the width of the kernel is narrower; heavier probabilistic weightings are given to neighbouring points than those further apart. Both plots in Figure 93 show that the widths of the time-series kernels are narrower than that of the state-space kernels at region, i = 350. It means the time-series fits, as characterised by the adapted lengthscale hyperparameters, are

sufficiently smooth to reproduce the data at the slow-varying region. However, in the case of fast-varying region at i = 504, Figure 94a shows that the width of the time-series kernel being wider than that of the state-space kernel, indicating that the smoothing effect is filtering away vital information, i.e. too much averaging is done. With the implementation of *Algorithm 6.2*, Figure 94b demonstrates that the widths of both kernels are almost identical. It follows that the averaging is done moderately with appropriate lengthscale hyperparameter values for the time-series component, ensuring the combined SSTS stochastic process does not trim away important details, i.e. under-fit the data.

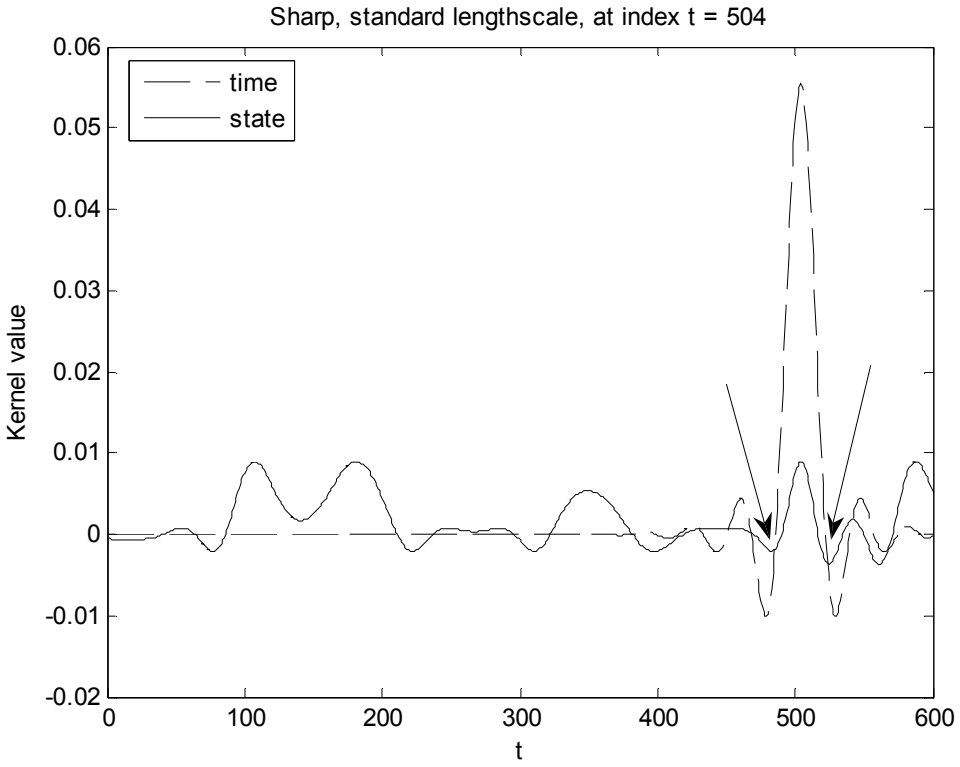


a) Standard approach. The width of the state-space kernel at $t = 350$ is sufficiently wide enough to cover that of the width of the time-series kernel (*see arrows*).

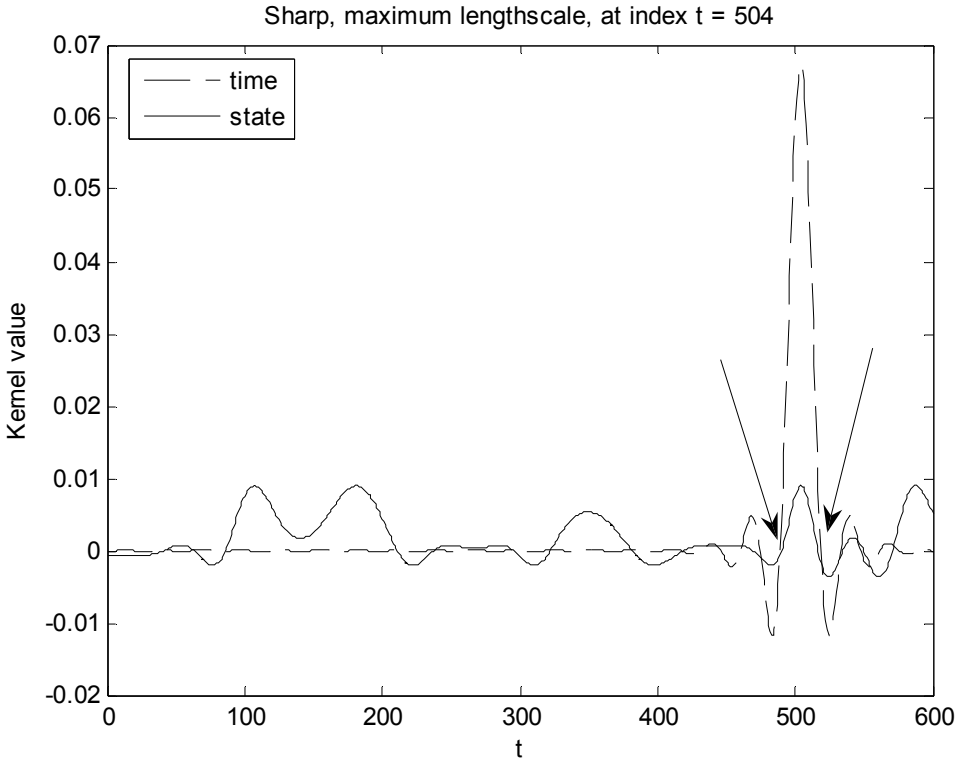


b) Dynamic lengthscale algorithm approach. A larger time-series lengthscale value results in a narrower width at $t = 350$, and therefore narrower than that of the state-space kernel.

Figure 93 Plots comparing state-space and time-series kernels at slow-varying region.



a) Standard approach. The width of the time-series kernel is now wider than that of the state-space kernel, resulting in *sharp corners* being trimmed off (*see arrows*).



b) Dynamic lengthscale algorithm approach. With shorter lengthscale for the time-series component, the width of its kernel is now similar to that of the state-space kernel.

Figure 94 Plots of state-space and time-series kernels at fast-varying region.

Example 5.1 *(Dynamic lengthscale illustration)*. A simple example is shown here to illustrate the improvement from using the *dynamic lengthscale algorithm*. A nonlinear function, $f(z) = \tanh(2z)\cos(1.5z)$, with N = 1,600 measurements is chosen to be the test data. The data points for the explanatory variable, $z \in \mathrm{P}$, depicted in Figure 95 are generated from a Gaussian process, sampled at 160Hz. The feature of the data points for z is smooth such that it simulates a certain input to a dynamic system. The outcome for the explanatory variable is also smooth and is shown in Figure 96. Figure 97 illustrates the relationship between the outcome f and the explanatory variable z. Additive Gaussian white noise n, of variance 0.01, is introduced to the outcome. The noisy measurements, $y \in \mathrm{P}$, are shown in Figure 96 and Figure 97.

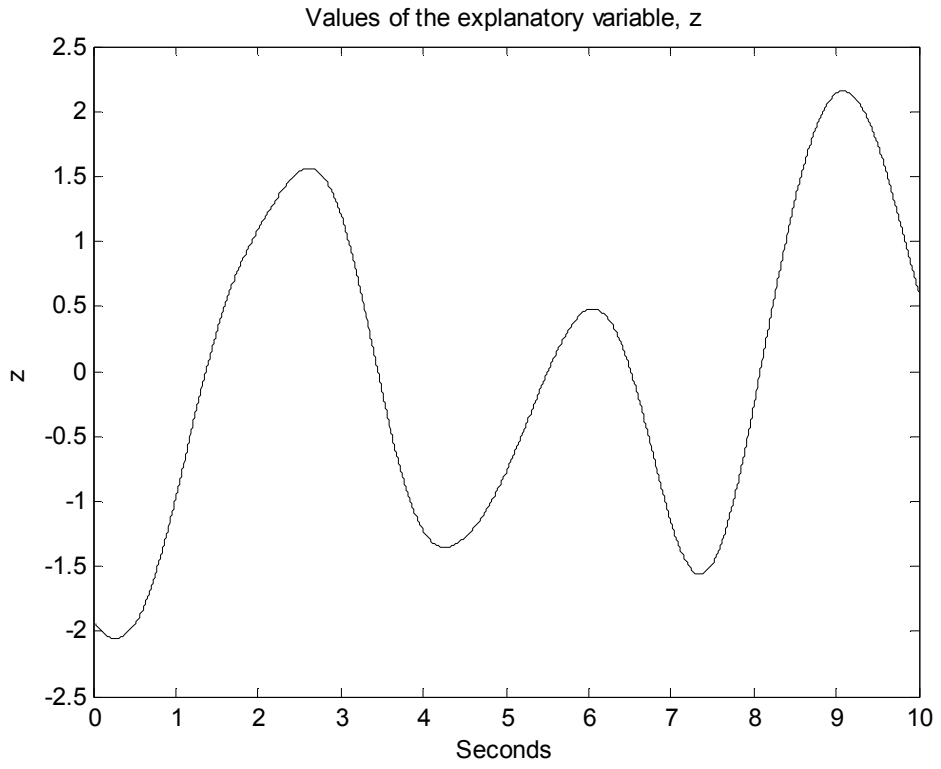


Figure 95 Values of the explanatory variable generated sequentially from a Gaussian process.

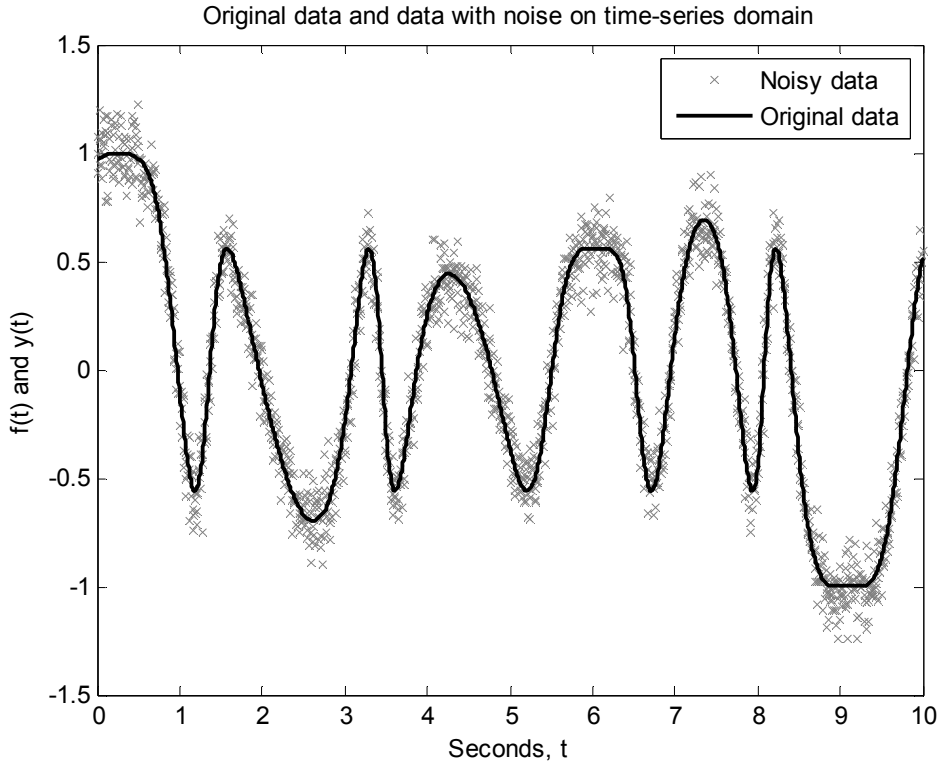


Figure 96 Original and noisy data on a time-series scale.

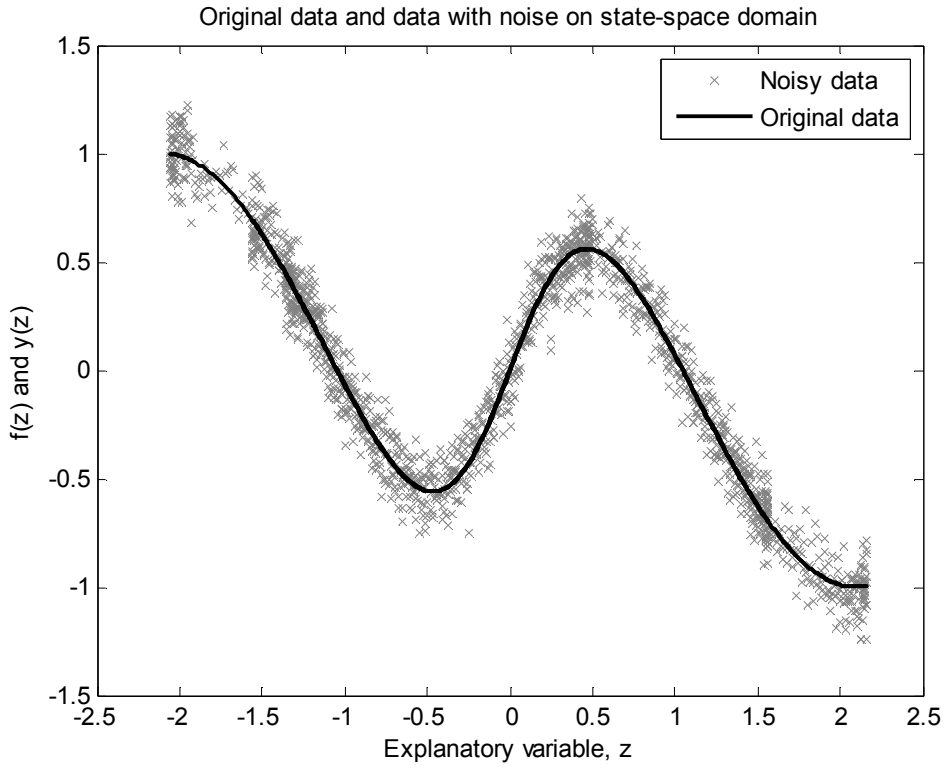


Figure 97 Original and noisy data illustrated on a state-space domain.

Since the purpose of this example is to demonstrate that the *dynamic lengthscale algorithm* is more effective in predictions when using SSTS Gaussian regression, the focus is peculiar to the time-series domain of the outcome. As seen in Figure 96, the dataset $\{t_i, y_i\}_{i=1}^{N}$, $t \in \mathrm{P}$ has the characteristics of dynamically-varying lengthscales. Some regions, i.e., between 3.1s and 3.5s, have fast-changing data points, whereas other regions, i.e. between 8.6s and 9.6s, have relatively slow-changing features. Therefore, the time-series component of the SSTS model does not have a distinctive lengthscale.

SSTS Gaussian regressions are applied on the dataset using two different approaches.

First Approach

The time-series lengthscale hyperparameter, d_{ts} = 19.8, is obtained from training the time-series dataset $\{t_i, y_i\}_{i=1}^{N}$ using standard Gaussian regression.

Second Approach

The time-series lengthscale hyperparameter, d_{dyn} = 32.9, is obtained from the *dynamic lengthscale algorithm* (see *Algorithm 6.2*).

These two time-series lengthscale hyperparameters are fixed values in the compound covariance function during the optimisation routine. The hyperparameters of the state-space component of the compound covariance function are obtained from standard Gaussian regression on the reduced dataset $\{\bar{z}_i, \bar{y}_i\}_{i=1}^{m}$, where $\bar{z}$ and $\bar{y}$ are data points permuted from z and y by taking into account of every 10^{th} data point, respectively. They are a_{s} = 0.397 and d_{s} = 2.5 for both cases, and are fixed during the optimisation routine. Only hyperparameters a_{t} and b are trained to maximise the log-likelihood function. The adapted values are a_{t} = 0.24 and b = 0.0093 in the first approach, and a_{t} = 0.13 and b = 0.0093 in the second approach.

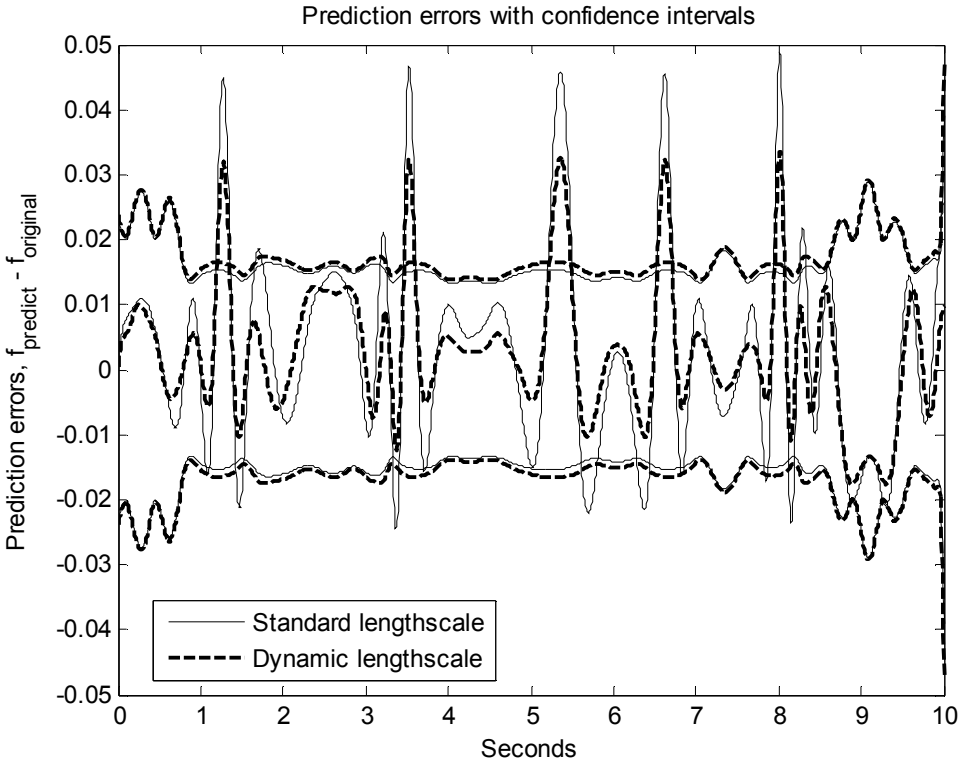


Figure 98 Prediction errors with confidence intervals of SSTS regressions.

The prediction errors and confidence intervals of the posteriors are illustrated in Figure 98. The results from the first and second approaches are indicated by straight and dashed lines, respectively. The confidence intervals are quite similar in both cases. The prediction errors, however, are rather different. The prediction errors from using the first approach are smaller than that of the second approach. Thus, SSTS Gaussian regression using *dynamic lengthscale algorithm* is capable of providing more accurate predictions.

6.7 SSTS Gaussian Regression on Dataset with Noise on Explanatory Variable

This section extends the analysis to application of SSTS Gaussian regression on datasets with input noise.

6.7.1 Modification in State-space Time-series Model

Following Gaussian regression on nonlinear dynamic systems, this section brings forth the next level by investigating the application of SSTS Gaussian process prior models. This combined model is used to exploit the presence of time-series characteristics where standard Gaussian regression is explicitly impossible when encountering large-scale state-space datasets. In the presence of noise on the input measurement, the compound covariance matrix for the SSTS Gaussian process is no longer of the form (93), but a modified one with an additional noise hyperparameter *e* in the state-space component to represent that input measurement noise, as shown here.

$$\overline{Q} = \left[\begin{array}{c|c} Q_t + bI & b\mathrm{P}^{\mathrm{T}} \\ \hline \mathrm{P}\, b & \mathrm{P}\left(Q_s + (b+e)I\right)\mathrm{P}^{\mathrm{T}} \end{array}\right] \tag{109}$$

In the prior assumption, the noise data is assumed to be the same, so the outcome of the time-series measurements has the same noise variance *b*, where **B** = *bI*, as that in the state-space representation. With noise present on the explanatory variable, the noise variance of state-space component is no longer the same as that in the time-series component. Hence, it has to be taken into account by compensating with a hyperparameter *e*. Since the explanatory variable of the time-series component does not contain noise, no additional hyperparameter is required in the time-series component of the compound covariance matrix (109).

Different datasets, including standard Gaussian regression, are investigated in this section.

1. $\mathrm{M}_1 = \{(x_i + n_i), y_i\}_{i=1}^N$, state-space dataset with noise present at the explanatory variable.
2. $\mathrm{M}_2 = \{x_i, (y_i + \tilde{n}_i)\}_{i=1}^N$, state-space dataset with noise present at the outcome.
3. $\mathrm{M}_3 = \{(x_i + n_i), (y_i + \tilde{n}_i)\}_{i=1}^N$, state-space dataset with noise present on both input and output measurements.

4. $M_4 = \{\hat{x}_i, (y_i + \tilde{n}_i)\}_{i=1}^N$, state-space dataset with noise present on the output measurement and with noisy input measurement filtered by Gaussian regression.
5. $M_5 = \{t_i, (y_i + \tilde{n}_i)\}_{i=1}^N$, time-series dataset.
6. $M_6 = \{t_i, (x_i + n_i), (y_i + \tilde{n}_i)\}_{i=1}^N$, with noises present at the explanatory variable and the outcome of the SSTS dataset.
7. $M_7 = \{t_i, \hat{x}_i, (y_i + \tilde{n}_i)\}_{i=1}^N$, with noise present on the output measurement and noisy input measurement filtered by Gaussian regression.
8. $M_8 = \{t_i, x_i, (y_i + \tilde{n}_i)\}_{i=1}^N$, with only noise present on the output measurement of the SSTS dataset.

M_1 to M_5 are datasets using standard Gaussian regression, whereas M_6 to M_8 are applied using SSTS Gaussian processes. The state-space components of the latter datasets are of reduced size; that is, every ten consecutive pairs of state-space measurements are considered from the full dataset. The time-series component uses the full time-series dataset. Standard Gaussian regressions applied on M_1 to M_5, using standard squared exponential covariance function (17), are investigated to ensure that the comparison is justifiable. The reduced datasets for M_1 to M_4 are analysed, i.e. $\tilde{M}_1, \tilde{M}_2, \tilde{M}_3$ and $\tilde{M}_4$. Dynamic lengthscale algorithm, as described in §6.6.2, is incorporated in the training procedure of the SSTS model.

For the experiment in this section, the test data is a nonlinear dynamic function, $y_i = \tanh(0.7x_i)\cos(1.5x_i)$, of size N = 600 with the explanatory variable, x_i, arbitrarily generated such that it is smooth, for i=1,…,N. Gaussian white noise of variance 0.0025 is introduced to measurements where noise is perceived. 20 samples are generated for these 8 datasets ($\tilde{M}_1, \tilde{M}_2, \tilde{M}_3$ and $\tilde{M}_4$ are further obtained from M_1, M_2, M_3 and M_4, respectively), each with a different noise data. Note that, with the SSTS Gaussian process prior model, predictions can be made in time-series domain, state-space domain or both. The hyperparameters are optimised and adapted to maximise the log-likelihood function. The results are reflected in the histograms, illustrating the cumulative prediction errors for the 20 samples, including each and

every data point. The histogram of the prediction errors are shown in Figure 99 and Figure 100.

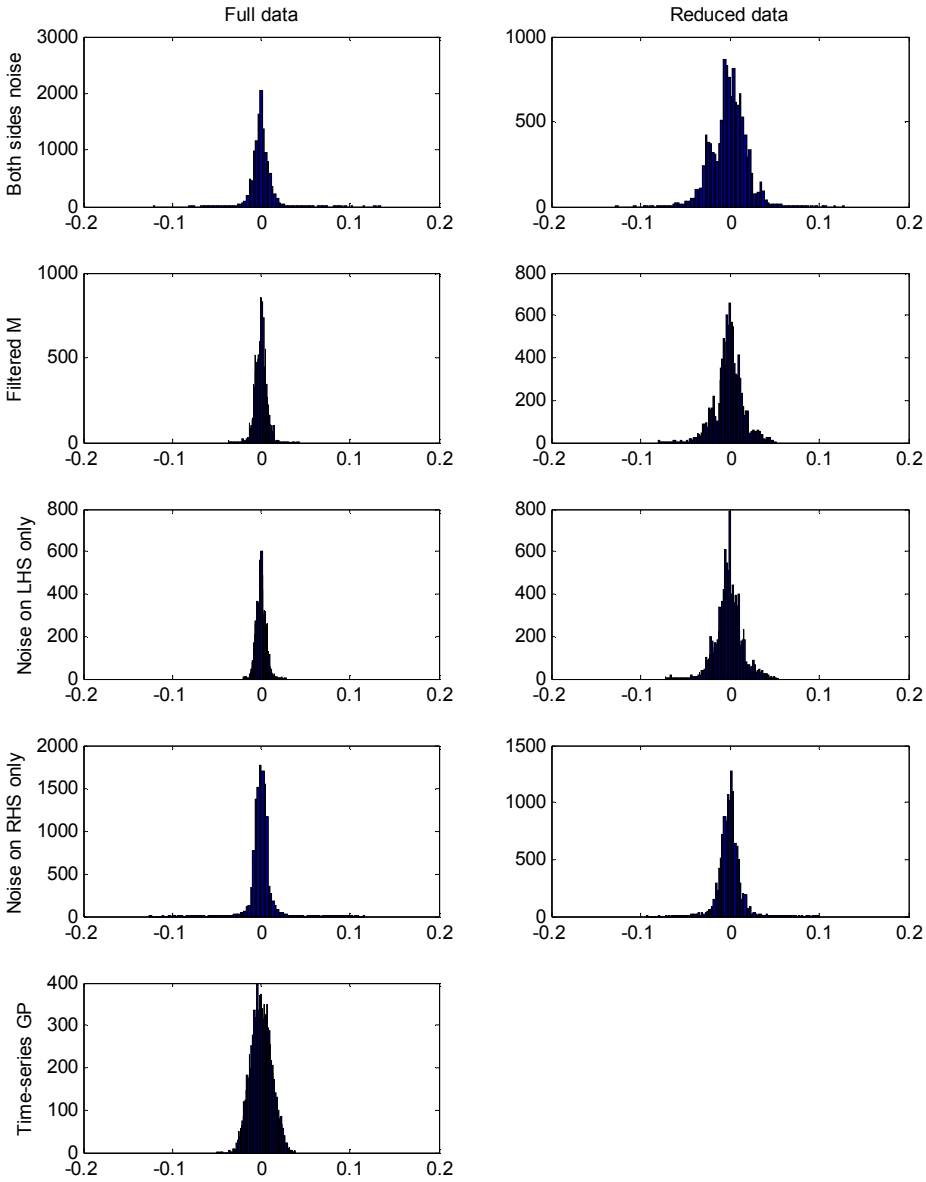


Figure 99 Standard Gaussian process prior models on 5 different datasets. The left column indicates the result obtained using the entire dataset, whereas the right column uses partially reduced datasets.

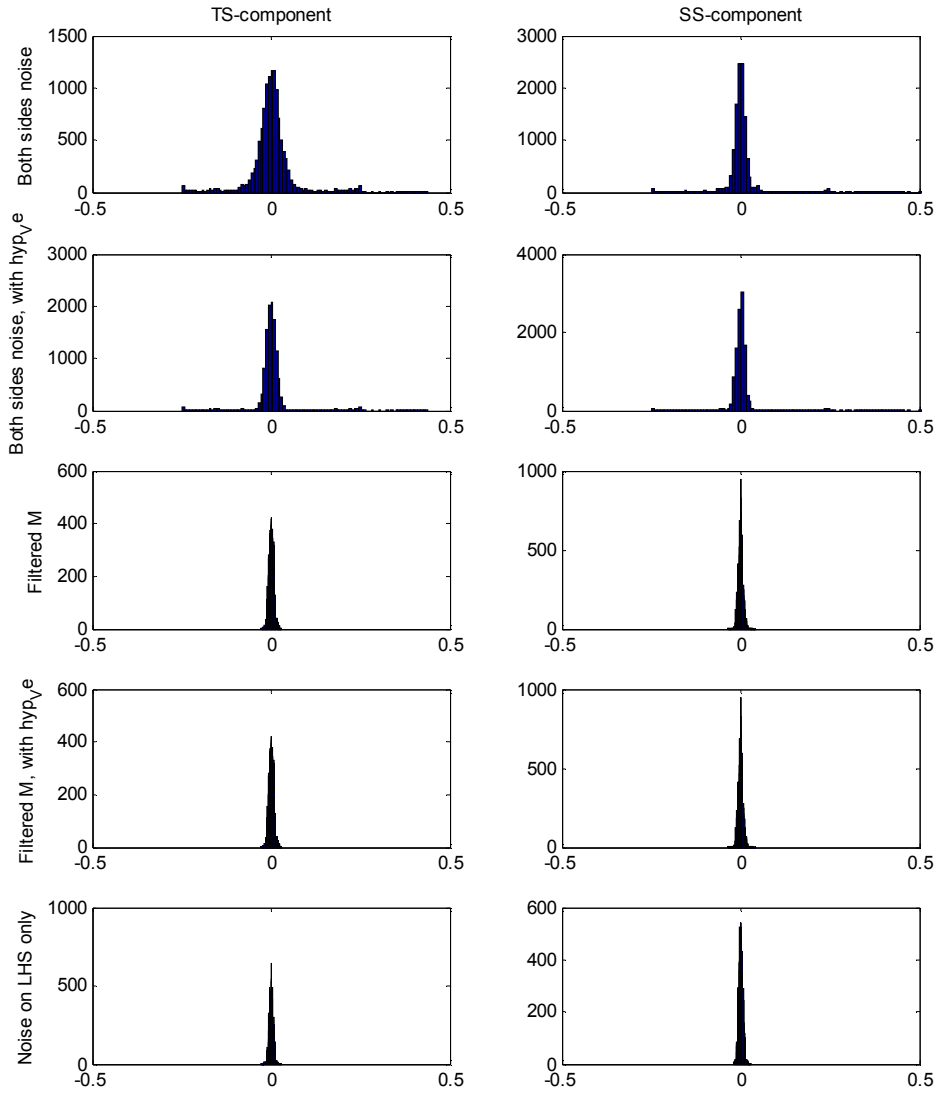


Figure 100 State-space time-series Gaussian process application on various SSTS datasets. The left column indicates prediction made on the time-series domain using information from the state-space component, whereas the right column illustrates prediction made on the state-space domain using the time-series information.

Figure 99 reveals that by using the reduced dataset of the Gaussian regression, it results in a broader spread of the distribution, which is mainly due to the lack of information. A time-series analysis is included to compare Gaussian regression using time-series data and state-space data. The time-series Gaussian regression uses the full dataset. It is clear that the histogram for the time-series Gaussian regression shows that the prediction error is not as good as any of the Gaussian regression applied on the full state-space dataset, despite the former using all the time-series data. In the

case where noisy input measurement has been filtered prior to Gaussian regression, the result is almost as good as though no noise is present at the input measurement, i.e. noise present only at the output measurement. Technically, Gaussian process is robust to the location of the noise, i.e. either in the input or output measurement, or even both. The absence of bias illustrates that Gaussian process is capable of predicting a good fit to the data.

The hyperparameter values from the standard Gaussian regression are tabulated in TABLE VIII. Datasets M_2 and M_4 (as well as corresponding $\widetilde{M}_2$ and $\widetilde{M}_4$) have recorded almost identical values. This is likely due to "pre-filtering" of the noise in the explanatory variable of the dataset before applying Gaussian process; that is, the latter dataset resembles that of the former. Clearly, the values of the noise variance for the four datasets and the time-series dataset, M_5, are identical.

TABLE VIII Mean hyperparameter values from standard Gaussian regression

	Mean values of the hyperparameters		
Dataset	*a*	*d*	*V*
M_1	0.0568	9.9991	0.0009
$\widetilde{M}_1$	0.3312	2.5243	0.0010
M_2	0.5373	1.1146	0.0025
$\widetilde{M}_2$	0.2732	2.1281	0.0025
M_3	0.1425	3.8240	0.0034
$\widetilde{M}_3$	0.2882	2.7862	0.0036
M_4	0.5634	1.0355	0.025
$\widetilde{M}_4$	0.2726	2.1560	0.0025
M_5	0.0415	7.8207	0.0025

Standard Gaussian regression using the squared exponential covariance function is performed on the 20 samples of the 9 datasets. Datasets M_1 to M_4 and $\widetilde{M}_1$ to $\widetilde{M}_4$ are mainly state-space regression, where M_5 is purely time-series Gaussian regression.

The second set of histograms, shown in Figure 100, consists of five different cases applied on datasets M_6, M_7 and M_8. Two of the datasets, M_6 and M_7, undergo SSTS Gaussian process using both classes of covariance functions, (93) and (109), whereas M_8 is only applied using covariance function (93). The SSTS Gaussian process allows prediction to be made in both time-series and state-space domains. Noticeably, the state-space fits for all cases are slightly better than time-series fits. Improvement is

observed for dataset M_6 with the introduction of an additional noise hyperparameter term in covariance function (109), but not for the case of dataset M_7, where the noisy input measurement has been pre-filtered by Gaussian process. The latter shows little or no difference as its value for hyperparameter, *e*, is found to be close to zero, i.e., 10^{-9}. More detail of the hyperparameter values is tabularised in TABLE IX.

TABLE IX Mean hyperparameter values from SSTS Gaussian regression

	Mean values of hyperparameters					
Dataset	a_t	a_s	d_t	d_s	v	e
M_6	0.0225	0.2882	29.1682	2.7862	0.0048	-
M_6^*	0.0225	0.2882	29.1682	2.7862	0.0048	$7.4\text{x}10^{-4}$
M_7	0.0430	0.2726	7.3669	2.1560	0.0024	-
M_7^*	0.0430	0.2726	7.3669	2.1560	0.0024	$3.8\text{x}10^{-10}$
M_8	0.0424	0.2732	7.4567	2.1281	0.0024	-

Superscript (*) on the dataset denotes that this particular dataset uses the modified compound covariance function (109), otherwise the standard compound covariance function (93) is used. SSTS Gaussian regressions are performed on the three datasets. Hyperparameters obtained from the optimisation procedure are shown here. The experiment is repeated using the modified covariance matrix (109) for datasets M_6 and M_7.

The mean values of the hyperparameters from the 20 sample datasets are calculated in TABLE IX. Only datasets M_6 and M_7 include an additional hyperparameter term, *e*, for repeating the experiment using compound covariance function (109). Notice that the values of the hyperparameters of datasets M_7 and M_8 are very similar. Once again, this is because of the "pre-filtering" technique that is performed on the explanatory variable of the noisy dataset before applying SSTS Gaussian regression. Generally, the hyperparameter value of the noise variance of dataset M_6 (and M_6^*) is higher than the rest since the dataset encompassed noises in both explanatory variable and the outcome.

However, it is probably better to use the filtered input measurement data before applying SSTS Gaussian process prior model. With no necessity to introduce a new noise hyperparameter, the combined covariance function (93) is sufficiently justified to perform data analysis using SSTS Gaussian process. With comparison to M_8, the result demonstrates that with pre-filtering, the fit is almost statistically identical to the dataset without having any input measurement noise. Pre-filtering the noise (assuming

noise is Gaussian) in the explanatory variable reduces the uncertainty of the data points for the explanatory variable. The Gaussian process still gives the joint probability distribution of the model given the data; therefore it is acceptable to first "filter" the explanatory variable data before applying the SSTS Gaussian regression.

6.7.2 2D Case

The case of state-space dataset with explanatory variable that is two-dimensional is further explored in this section with the nonlinear dynamic test data chosen to be $y_i = \tanh(0.7x_i + r_i) + \cos(1.5x_i)$, for i=1,…,N, where the data size is N = 1,000. Explanatory variables, $\{x_i\}_{i=1}^N$ and $\{r_i\}_{i=1}^N$, of the stochastic process are essentially smooth Gaussian process-generated functions; that is, functions obtained using the squared-exponential covariance function, (15), with specific hyperparameter values. These hyperparameters are a = 0.1 and d = 28.8 for x and a = 0.1 and d = 48 for r, on a randomly-generated (using a normal distribution, $\mathrm{N}(0,1)$) time-series data within the range, $t \in [0,10]$. The variance of the noise data is 0.0025. 20 samples of the 8 datasets (reduced state-space datasets are obtained from M_1, M_2, M_3 and M_4), as defined in §6.7.1, are obtained, each with different noise sample, and applied using standard Gaussian processes and combined SSTS Gaussian processes. In the case where data reduction is necessary, every ten consecutive data points are, thus considered. The results from the data analysis are plotted in Figure 101 and Figure 102 in the form of histograms. Again, the conclusions are the same and are coherent with those from §6.7.1.

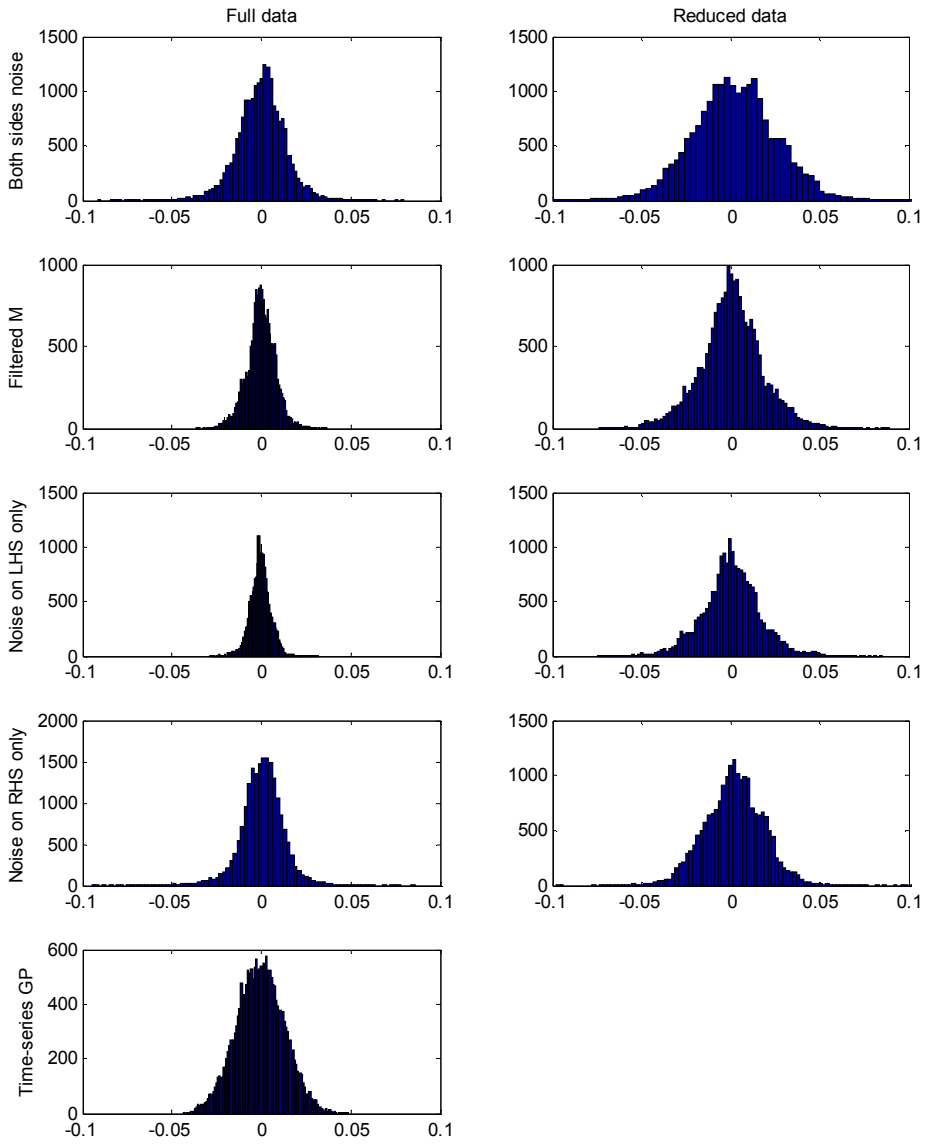


Figure 101 Standard Gaussian process prior models on 5 different datasets. The left column indicates the result obtained using the entire dataset, whereas the right column uses partially reduced datasets. The explanatory variable of the dataset is two-dimensional.

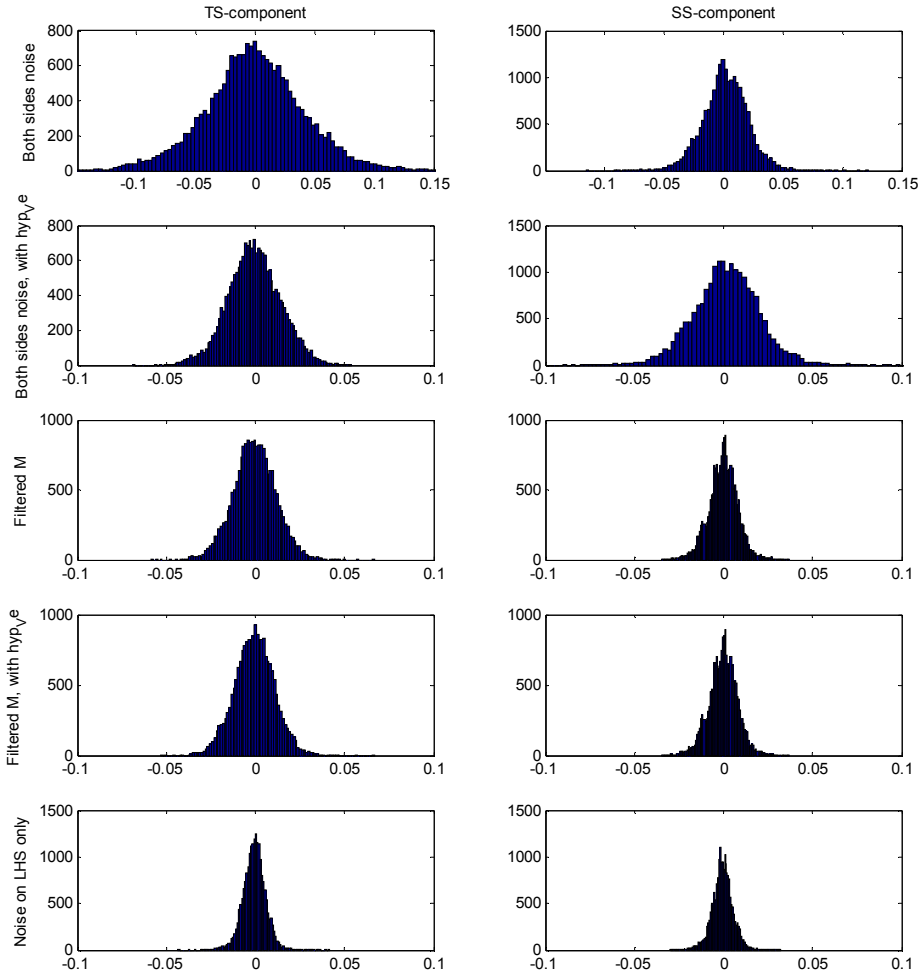


Figure 102 State-space time-series Gaussian process application on various SSTS datasets. The left column indicates prediction made on the time-series domain using information from the state-space component, whereas the right column illustrates prediction made on the state-space domain using the time-series information. The explanatory variable of the dataset is two-dimensional.

6.8 Stochastic Derivatives of SSTS

This section analyses the derivations for derivative observations of the SSTS Gaussian process prior models. Derivative observations can be predicted in both time-series and state-space domains.

6.8.1 Derivative Observations of Combined Gaussian Process

Recently, Solak et al. (2003) demonstrates the capabilities of using Gaussian process prior models to achieve derivative observations directly from the empirical model. Besides function observations, derivative observations are also vital, particularly in identification of nonlinear dynamic systems from experimental data. The derivative stochastic processes are defined in Chapter 2.6.1 from equations (11), (12) and (13).

Differentiation is a linear operator, so the derivative of a combined Gaussian process remains a Gaussian process. The use of derivative observations in Gaussian processes has been described by O'Hagan (1992) and Rasmussen (2003), with some engineering applications (Murray-Smith et al., 1999, Leith et al., 2002). In dynamic systems with large-scale datasets, the application of a derivative Gaussian process can be catalysed by the use of the combined SSTS model. Given that the model consists of two cross-related components, i.e. the state-space domain and the time-series domain, the derivative operations can be performed on any of these domains. The identities (17), (18) and (19) are necessary to form the full covariance matrix (93).

The means and covariances for the first order derivatives are formulated in (110) and (111), respectively, where $\widetilde{Q}$ is the combined SSTS covariance matrix (93).

$$\Lambda'_{\hat{z}z}\widetilde{Q}^{-1}\mathbf{W} \tag{110}$$

$$\Lambda''_{\hat{z}\hat{z}} - \Lambda'_{\hat{z}z}\widetilde{Q}^{-1}\Lambda'_{z\hat{z}} \tag{111}$$

The terms $\Lambda''_{\hat{z}\hat{z}}$ and $\Lambda'_{\hat{z}z}$ are defined as

$$\Lambda''_{\hat{z}\hat{z}} = \left[\begin{array}{c|c} \dfrac{\partial^2}{\partial t_i \partial t_j}(\Lambda_t) & 0 \\ \hline 0 & \dfrac{\partial^2}{\partial z_i \partial z_j}(\Lambda_z) \end{array}\right] \text{ and } \Lambda'_{\hat{z}z} = \left[\begin{array}{c|c} \dfrac{\partial}{\partial t}(\Lambda_t) & 0 \\ \hline 0 & \dfrac{\partial}{\partial z}(\Lambda_z) \end{array}\right]$$

where Λ_t and Λ_z are covariance matrices of the time-series and state-space components for the SSTS model, respectively.

Not all covariance functions are infinitely differentiable. For example, the Matérn class of covariance functions as given in (16) is k-times differentiable if and only if $v > k$. The squared exponential covariance function, on the other hand, is infinitely differentiable, which means the Gaussian process has mean square derivative of all orders, and therefore is very smooth. It follows that the combined SSTS covariance matrix is also infinitely differentiable.

6.9 Application to State-space Time-series Model

The combined SSTS Gaussian process prior model is applied to the following large-scale example. An Intel® Pentium® IV 2.8GHz machine with 512MB RAM is used to carry out this experiment.

Let $\mathrm{f}(\mathbf{z}) = \tanh\left[Az_1z_2 - \sin(Bz_1)\right] - C\sin(Ez_2)\cos(z_1z_2)$ be a smooth function, with A=0.1, B=3, C=0.8 and E=5, on the domain D containing the rectangular grid, $\{-1 \le z_1 \le 1, -2 \le z_2 \le 2\}$. 6,000 training data points are obtained at a sampling rate of 10Hz and mapped on the grid with noise intensity of variance 0.01, as shown in Figure 103. The test data is generated from the state-space function, $y_{n+1} = ay_n + cr_n$, where a = 0.998 and c = 0.0153. The input explanatory variable is $\mathrm{M} = [y_n \mid r_n]$ defined as f(M|z_1=y,z_2=r). Following the training procedure outlined in §6.6.1, the SSTS prior model is applied to this example. The time-series component contains the full measurement data, whereas the state-space component uses a reduced dataset, with samples taken at every 10 measurement, viz. 600 pairs of measurement points are used in the state-space component of the combined model for the optimisation routine. The hyperparameters adapted from maximising the log-likelihood function are a_t=0.27, d_t=1.15, a_s=0.73 $d_{s,1}$=2.9, $d_{s,2}$=5.5 and b=0.0096. This newly constructed method is compared to the standard Gaussian regression using only the state-space information, with prior covariance function (17). For standardisation purposes, the standard Gaussian process for the state-space analysis also uses the reduced 600 pairs of dataset.

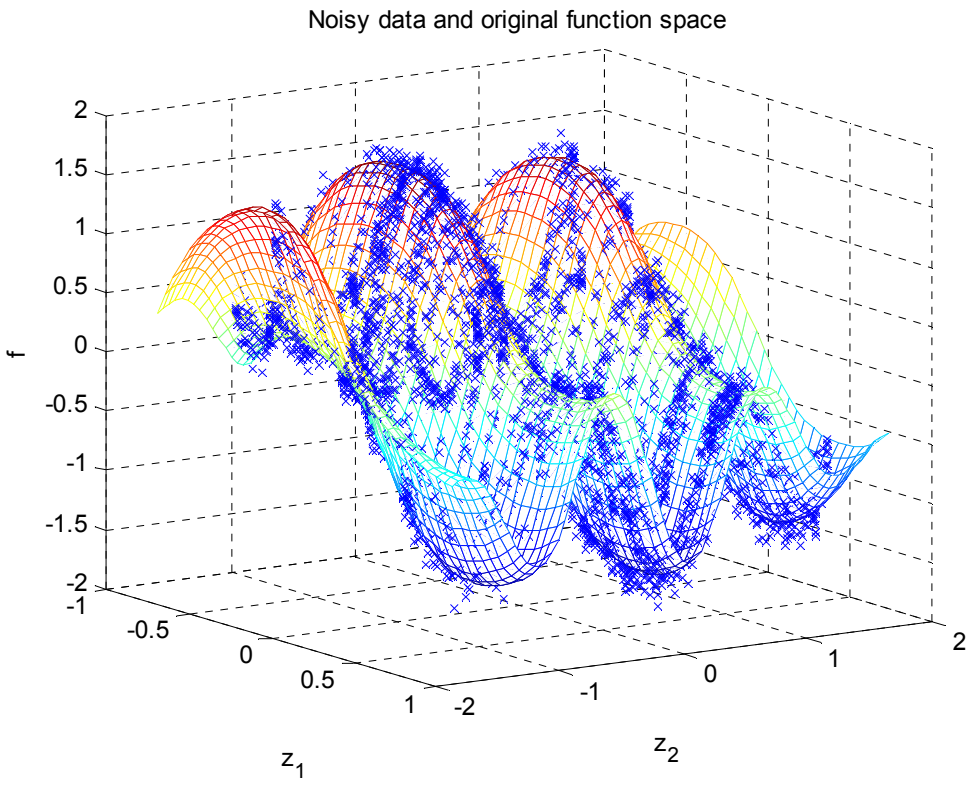


Figure 103 Original space mapping and noisy measurement data.

The total prediction errors based on these two stochastic processes are plotted in Figure 104 and Figure 105. In addition, the data points are also depicted in the figures by grey 'x' markings. Some regions were found to have poorer prediction as shown in Figure 104. Figure 105 illustrates that the prediction errors are smaller in most regions. The prediction errors from the standard Gaussian process applied on state-space dataset are observed to be generally larger than those applied using the combined SSTS Gaussian process model. The respective two times the standard deviations are plotted in Figure 106 and Figure 107. Data points, marked by grey 'x', are also shown in these figures. Small uncertainties are accumulated towards regions with higher measurement data density. The confidence intervals of the combined model, illustrated in Figure 107, show a tighter fit of the data. In contrast, the general Gaussian process model shows a wider confidence interval, thereby indicating a higher uncertainty in the posterior joint probability distribution.

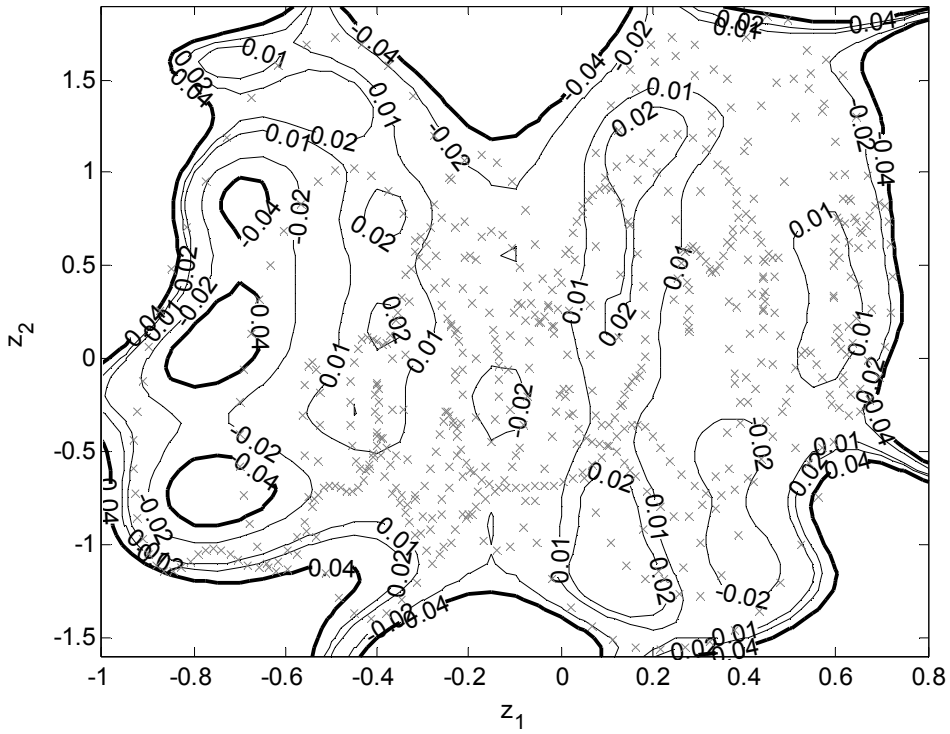


Figure 104 Prediction errors from standard Gaussian process model.

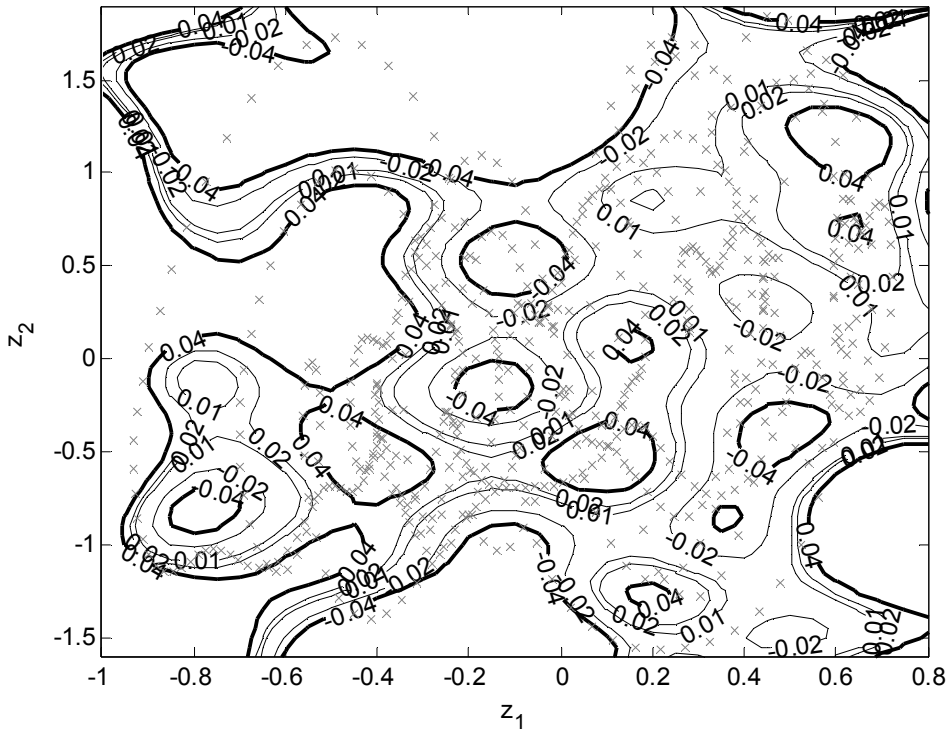


Figure 105 Prediction errors from the state-space time-series model.

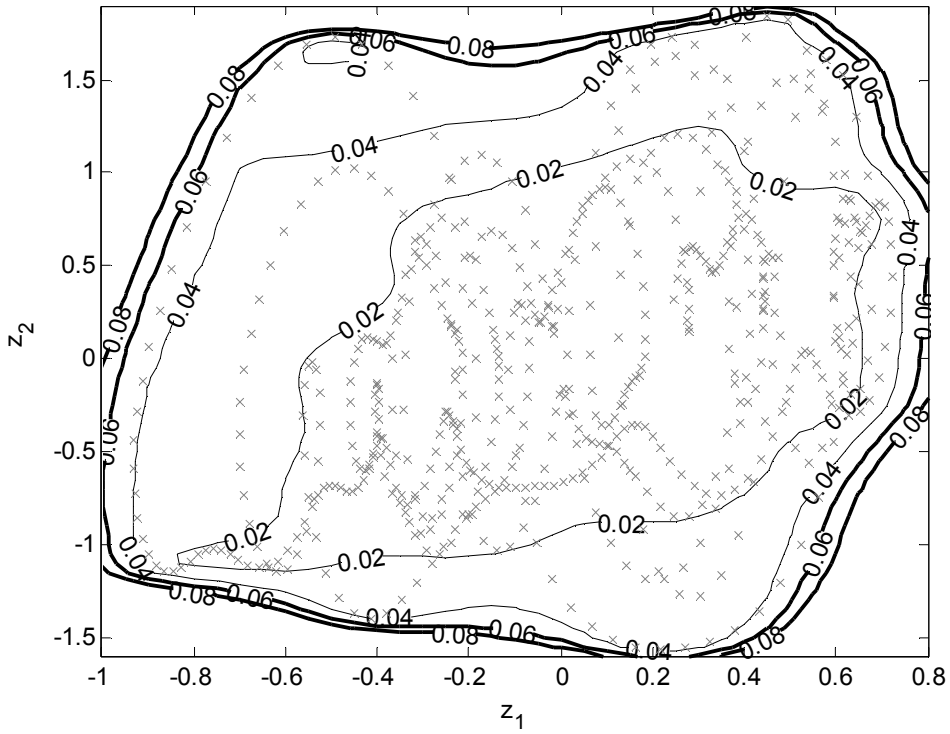


Figure 106 Two times the standard deviation from standard Gaussian process.

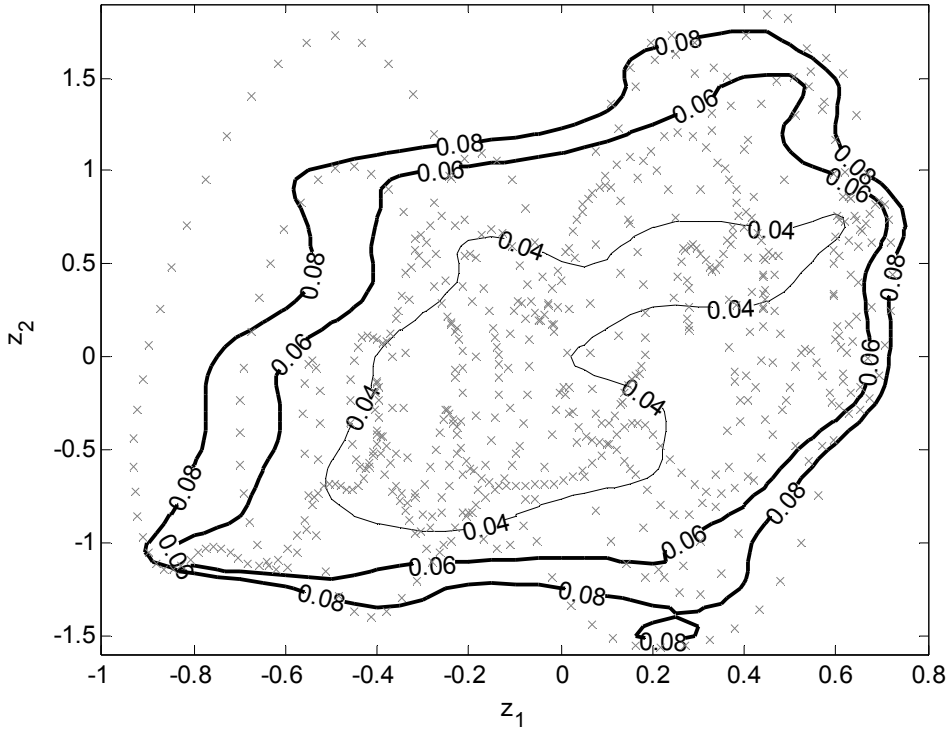


Figure 107 Two times the standard deviation from the state-space time-series model.

6.10 Conclusion

Two main issues are identified and addressed in this chapter. Firstly, following successful individual identification of time-series and state-space data by Gaussian regression, a method to combine both approaches into a single stochastic process is presented. Secondly, large-scale data, which is seemingly impossible for Gaussian process to be applicable, is now possible with the use of combined SSTS model. Without loss of generality, all the measurement data is used in the time-series domain, with the reduced dataset applied in the state-space component.

The limitations of Gaussian processes, with several $O(N^3)$ operations and $O(N^2)$ memory storage requirement, are relieved with the establishment of the modified Durbin-Levinson's algorithm and the generalised Schur algorithm within the Gaussian process for time-series data analysis application. This ensures that all $O(N^3)$ operations are now $O(kN^2)$, where k is very much smaller than N. §6.9 has successfully illustrated the identification of nonlinear dynamic dataset with the employment of the novel SSTS Gaussian process.

Chapter 7

Conclusion

"Only two things are infinite, the universe and human stupidity, and I'm not sure about the former."

- Albert Einstein (1879 – 1955)

This thesis discusses a statistical framework for the identification of nonlinear dynamic systems (Jung, 1999) within a Bayesian context (Bayes, 1763; Box and Tiao, 1973). From the many identification techniques available, ranging from sub-space identification to unsupervised learning of neural networks, Gaussian regression is chosen here for this purpose. Gaussian regression was first proposed several decades ago, but is mainly used by statisticians. It was only during the late 1990s that Gaussian regression began to be applied to engineering applications (Kocijan et al., 2003; Leith et al., 2004). More often, it is considered to be an aspect of machine learning, subsuming other methods. It is now used in many different fields, including statistics, pattern recognition, signal processing, artificial intelligence, data mining and neural networks. Nonlinear dynamic systems are challenging to model, particularly if the form of the underlying nonlinear relationship is unknown. Attempts to model such systems using a non-parametric approach are preferred to avoid any bias that could arise from a parametric approach.

Chapter 2 provides a detailed explanation of the Gaussian regression. As a stochastic process, the Gaussian process is computationally simple since the model is completely defined by the mean function and the covariance function. The choice of covariance function is influenced by the prior information. For example, if the data is known to be periodic, a periodic covariance function can be used for the Gaussian process model. Alternatively, the commonly used squared exponential covariance function is frequently chosen. The mean function and covariance function depend on a set of hyperparameters, whose values during model selection are often obtained either by adapting them to maximise the log-likelihood function (Mardia and Marshall, 1984; Moller, 1993), or by the Monte Carlo methods (Barry and Pace, 1999; Duane et al., 1987). There is little to choose between the two methods, although the former is preferred for large datasets (Gibbs, 1997).

The identification of nonlinear dynamic systems using Gaussian regression must deal with several issues. Since Gaussian regression involves several $O(N^3)$ operations with $O(N^2)$ memory storage requirement, the computational effort is rather expensive. In Chapter 3, modifications to the training procedure for the hyperparameters to speed up the optimisation routines (Seeger et al., 2003; Wu et al., 2001; Yoshioka and Ishii, 2001) are discussed. In addition, some fast and memory efficient algorithms are developed to enable large-scale datasets to be handled. The generalised Schur algorithm (Chandrasekaran and Sayed, 1996, 1999a, 1999b; Kailath, 1999) is implemented in a form that is suitable for Gaussian regression applied to datasets (including gaps) that depend on a scalar explanatory variable with fixed interval between the measurements.

When the nonlinear relationship underlying data consists of two or more additive components with different characteristics, the extraction of these components by Gaussian regression may not be effective since the confidence intervals for the predictions can be excessively broad. The excessively broad confidence intervals are caused by the freedom to add an arbitrary function to one component while subtracting it from the other without changing the likelihood of the data. The key to minimising the confidence intervals is to remove this freedom. Gaussian regression methods to extract the individual components with minimum confidence intervals are developed in Chapter 4. Multiple independent Gaussian process models are used. Two

different cases are considered, the case, when the components have different explanatory variables, and the case, when the components have the same explanatory variable.

The algorithms and methods developed in Chapter 3 and 4 are applied to a case study in Chapter 5. The data set consists of site measurements of rotor speed, blade pitch angle and wind speed for a 1MW commercial wind turbine (Leithead, 1992; Leithead et al., 2003a). The data is sampled at 40Hz for a run of 600 seconds, providing 24,000 data points. The aim is to extract the wind turbine aerodynamics from the data which are heavily corrupted with measurement noise. Firstly, as the data are obtained as time-series, Gaussian regression with time as the explanatory variable is applied to filter the measurements of rotor speed, blade pitch angle and wind speed and to predict the rotor acceleration. In each case, the methods based on multiple stochastic processes developed in Chapter 4 are used to obtain narrow confidence intervals. The fast algorithms developed in Chapter 3 are necessary to handle the large size of the data sets. Secondly, Gaussian regression is applied to the predicted rotor acceleration with the explanatory variable the filtered rotor speed, blade pitch angle and wind speed to extract the wind turbine aerodynamics. As it is only one component of the underlying nonlinear relationship, the other being the wind turbine drive-train dynamics, the multiple Gaussian processes models developed in Chapter 4 are again used.

In Chapter 6, the Gaussian process prior model, to be used when applying Gaussian regression to identify nonlinear dynamic systems, is investigated. In this context, the data set is typically obtained as a time-series but the underlying nonlinear relationship is dependent on some other explanatory variable. Gaussian regression based on a pair of independent Gaussian processes that caters for this dual nature of the data is proposed. One of the Gaussian processes accounts for the correlation in the data with respect to time, the other accounts for the correlation with respect to the explanatory variable underlying the nonlinear relationship. The correlation for the former is high only for those points that are nearby in the data sequence whilst the correlation for the latter is also high for those points widely separated in the data sequence but nearby in terms of the explanatory variable underlying the nonlinear relationship. This aspect of the dual nature Gaussian process model can be exploited to reduce the size of the data,

when considered to depend on the explanatory variable underlying the nonlinear relationship, without any significant loss of information: the confidence intervals for predictions based on the reduced data are very similar to those for the full data set. Consequently, the dual nature Gaussian process models of Chapter 6 in combination with the fast and efficient algorithms developed in Chapter 3 enable larger data sets to be used when applying Gaussian regression to identify nonlinear dynamic systems. Because of the transitory nature of the data when not restricted to being in a neighbourhood of an equilibrium operating point, to obtain sufficient data in a region in the space of the explanatory variable underlying the nonlinear relationship, the data might not consist of a continuous time-series but might contain gaps. However, the algorithms of Chapter 3 extend to data with gaps. Furthermore, the dual nature Gaussian process model of Chapter 6, since it explicitly includes the time-series aspect, enables pre-filtering of the data, particularly, when combined with the multiple Gaussian process prior models of Chapter 4.

This thesis lays the foundation for future applications of Gaussian regression to the identification of nonlinear dynamic systems. Further work following on from that presented here includes the following.

- Modify the dual nature Gaussian process models from Chapter 6 to include the multiple Gaussian process models from Chapter 4. These modifications only apply to the time-series aspects of the model. Pre-filtering of the data can then be undertaken using the multiple Gaussian process models.
- Modify the training and prediction algorithms for the dual nature Gaussian process models from Chapter 6 through using the general approximation methods developed elsewhere (Leithead and Zhang, 2007; Zhang and Leithead, 2007; Quiñonero-Candela and Rasmussen, 2005) to develop faster and more efficient algorithms. This modification only applies to those aspects related to the explanatory variable underlying the nonlinear relationship. It would enable the size of larger data sets to be further increased. Apply Gaussian regression, using the dual nature Gaussian process models with the enhancements suggested above, to nonlinear dynamic systems

identification with realistic data sets and undertake a thorough evaluation of the methodology.

- Develop "grey-box" approaches to nonlinear dynamic system identification using Gaussian regression, for example, gain-scheduled models (see Leithead et al., 1999). Since this entails the separation of the nonlinear component from the linear component, the multiple Gaussian process models of Chapter 4 are appropriate.

Although there remains much to do, significant steps towards a practical methodology for identifying nonlinear dynamic systems based on Gaussian regression have been made in the work reported here.

Appendix A

Some useful mathematical formulae are used in the development of the research work in this thesis. Several standard operations are repeatedly met when dealing with multi-variate probability distributions which are tractable analytically. It is perhaps helpful by summarising some of these mathematical formulas for easy reference.

A.1 Matrix Identities

Given the matrices, **A**, **B** and **C**, the following mathematical expressions are defined.

A.1.1 Determinants of Matrix Expressions

Assume for the case where **A** and **D** are non-singular square matrices, and **B** and **D** are rectangular matrices of appropriate dimensions, then the following relation for the determinant holds; that is

$$\det\left\{\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}\right\} = \left|\mathbf{A}\right| \times \left|\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right| = \left|\mathbf{D}\right| \times \left|\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right|$$

Since the log-determinant is more widely used, it follows that

$$\log\det\left\{\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}\right\} = \log\left|\mathbf{A}\right| + \log\left|\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right| = \log\left|\mathbf{D}\right| + \log\left|\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right|$$

Another special property of the determinant expression is that

$$\left|\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right| = \frac{\left|\mathbf{A}\right|}{\left|\mathbf{D}\right|} \times \left|\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right|$$

A.1.2 Inverse Block Matrix

Let the invertible matrix be partitioned with **A**, **B**, **C** and **D** as shown below, such that **A** and **D** are square matrices, but not necessarily the same size. Then, **B** and **C** are rectangular matrices of appropriate dimensions. Conversely, the inverse of this partitioned matrix can be written as

$$\left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right]^{-1} = \left[\begin{array}{c|c} \left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & -\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \\ \hline -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \end{array}\right]$$

or

$$\left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right]^{-1} = \left[\begin{array}{c|c} \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \\ \hline -\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} & \left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \end{array}\right]$$

where **A** is a n_1 x n_1 matrix and **B** is a n_2 x n_2 matrix.

A.1.3 Matrix Inversion Lemma

Also known as the Woodbury, Sherman & Morrison formula (Press et al., 1992), the matrix inversion lemma states that

$$\left(\mathbf{A}+\mathbf{B}\mathbf{D}\mathbf{C}^{\mathrm{H}}\right)^{-1} = \mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}^{-1}+\mathbf{C}^{\mathrm{H}}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}^{\mathrm{H}}\mathbf{A}^{-1}$$

such that **A** and **D** are square matrices, assuming all the relevant inverses exists. They may not necessarily have the same dimension. $\mathbf{C}^{\mathrm{H}}$ denotes the complex conjugate transpose of the matrix **C**, provided it is a complex matrix, otherwise it is just a transpose operator.

A.1.4 Properties of Matrix Derivatives

Given here are some properties of the derivatives of matrix, **X**.

1. $$\frac{\partial}{\partial\theta}\mathbf{X}^{-1} = -\mathbf{X}^{-1}\left(\frac{\partial\mathbf{X}}{\partial\theta}\right)\mathbf{X}^{-1}$$

2. $$\frac{\partial}{\partial\theta}\log|\mathbf{X}| = tr\left\{\mathbf{X}^{-1}\left(\frac{\partial\mathbf{X}}{\partial\theta}\right)\right\}$$

3. $$\frac{\partial}{\partial\theta}|\mathbf{X}| = |\mathbf{X}|\mathbf{X}^{-\mathrm{T}}$$

4. $$\frac{\partial}{\partial\theta}tr(\mathbf{X}) = I$$

Applying chain rule to derivative operation, if f(**x**) is a scalar function of **x**, which itself is a scalar, then

$$\frac{\partial}{\partial\theta}\mathrm{f}(\mathbf{x}) = tr\left\{\frac{\partial\mathrm{f}(\mathbf{x})}{\partial\mathbf{x}}\left(\frac{\partial\mathbf{x}}{\partial\theta}\right)^{\mathrm{T}}\right\} = tr\left\{\left(\frac{\partial\mathrm{f}(\mathbf{x})}{\partial\mathbf{x}}\right)^{\mathrm{T}}\frac{\partial\mathbf{x}}{\partial\theta}\right\}$$

Appendix B

Some important proofs and derivations that are related to Gaussian process prior models are given here. For example, the derivation of Hessian functions to improve optimisation procedure.

Derivation of Hessian Matrix

It follows from (6) that

$$\frac{\partial^2 \Lambda(\theta)}{\partial\theta_i \partial\theta_j} = \frac{1}{2}\frac{\partial}{\partial\theta_j}\left\{tr\left(Q^{-1}\frac{\partial Q}{\partial\theta_i}\right)\right\} - \frac{1}{2}\mathbf{Y}^{\mathrm{T}}\left[\frac{\partial}{\partial\theta_j}\left\{Q^{-1}\frac{\partial Q}{\partial\theta_i}Q^{-1}\right\}\right]\mathbf{Y}$$

$$= \frac{1}{2}tr\left[\frac{\partial tr\left(Q^{-1}\frac{\partial Q}{\partial\theta_i}\right)}{\partial\left(Q^{-1}\frac{\partial Q}{\partial\theta_i}\right)}\left\{\frac{\partial\left(Q^{-1}\frac{\partial Q}{\partial\theta_i}\right)}{\partial\theta_j}\right\}^{\mathrm{T}}\right]$$

$$-\frac{1}{2}\mathbf{Y}^{\mathrm{T}}\left[\frac{\partial\left(Q^{-1}\right)}{\partial\theta_j}\frac{\partial Q}{\partial\theta_i}Q^{-1} + Q^{-1}\frac{\partial^2 Q}{\partial\theta_i\partial\theta_j}Q^{-1} + Q^{-1}\frac{\partial Q}{\partial\theta_i}\frac{\partial\left(Q^{-1}\right)}{\partial\theta_j}\right]\mathbf{Y}$$

Using the theorem on trace derivative and quadratic products (Golub and Van Loan, 1996), the equation is simplified to

$$\frac{\partial^2\Lambda(\theta)}{\partial\theta_i\partial\theta_j}=\frac{1}{2}tr\left\{\frac{\partial}{\partial\theta_j}\left(\frac{\partial Q}{\partial\theta_i}Q^{-1}\right)\right\}-\frac{1}{2}\mathbf{Y}^{\mathrm{T}}\left[Q^{-1}\frac{\partial^2 Q}{\partial\theta_i\partial\theta_j}Q^{-1}+2Q^{-1}\frac{\partial Q}{\partial\theta_i}\frac{\partial\left(Q^{-1}\right)}{\partial\theta_j}\right]\mathbf{Y}$$

Applying inverse-derivative formulation,

$$\frac{\partial^2\Lambda(\theta)}{\partial\theta_i\partial\theta_j}=\frac{1}{2}tr\left\{\frac{\partial^2 Q}{\partial\theta_i\partial\theta_j}Q^{-1}+\frac{\partial Q}{\partial\theta_i}\frac{\partial\left(Q^{-1}\right)}{\partial\theta_j}\right\}$$

$$-\frac{1}{2}\mathbf{Y}^{\mathrm{T}}\left[Q^{-1}\frac{\partial^2 Q}{\partial\theta_i\partial\theta_j}Q^{-1}-2Q^{-1}\frac{\partial Q}{\partial\theta_i}Q^{-1}\frac{\partial Q}{\partial\theta_j}Q^{-1}\right]\mathbf{Y}$$

$$=\frac{1}{2}tr\left\{\frac{\partial^2 Q}{\partial\theta_i\partial\theta_j}Q^{-1}-\frac{\partial Q}{\partial\theta_i}Q^{-1}\frac{\partial}{\partial\theta_j}Q^{-1}\right\}-\frac{1}{2}\mathbf{Y}^{\mathrm{T}}\left[Q^{-1}\frac{\partial^2 Q}{\partial\theta_i\partial\theta_j}Q^{-1}-2Q^{-1}\frac{\partial Q}{\partial\theta_i}Q^{-1}\frac{\partial Q}{\partial\theta_j}Q^{-1}\right]\mathbf{Y}$$

Maximum Likelihood Estimation

A relationship between linear regression and maximum likelihood estimation (MLE) exists and is shown with the explanation below. A first order system is assumed here.

Linear Regression as MLE

Consider the dataset $\{\hat{y}_n(z_n,r_n)\}_{n=1}^N$ such that the explanatory representations are

$$y_n=az_n+br_n;\ \hat{y}_n=y_n+\varepsilon_n$$

such that noise is represented by ε. Let $\hat{Y}=[\hat{y}_1,...,\hat{y}_N]^{\mathrm{T}}$, $Z=[z_1,...,z_N]^{\mathrm{T}}$, $R=[r_1,...,r_N]^{\mathrm{T}}$ and $E=[\varepsilon_1,...,\varepsilon_N]^{\mathrm{T}}$. Hence,

$$E=\left(\hat{Y}-aZ-bR\right)$$

and E is a Gaussian variable with covariance matrix dI. The likelihood of the data given $\theta=(a,d,b)$ is

$$L=(2\pi d)^{-N/2}\exp\left\{-\left(\hat{Y}-aZ-bR\right)^{\mathrm{T}}\left(\hat{Y}-aZ-bR\right)/(2d)\right\}$$

It follows that the log-likelihood function is

$$LL = -2\ln(L) \approx N\ln(d) + \left(\hat{Y} - aZ - bR\right)^{\mathrm{T}}\left(\hat{Y} - aZ - bR\right)/d$$

The first order derivative of *LL* with respect to *d* is

$$\frac{\partial LL}{\partial d} = \frac{N}{d} - \left(\hat{Y} - aZ - bR\right)^{\mathrm{T}}\left(\hat{Y} - aZ - bR\right)/d^2$$

By equating the derivative to be zero,

$$\frac{\partial LL}{\partial d} = 0$$
$$\therefore d = \left(\hat{Y} - aZ - bR\right)^{\mathrm{T}}\left(\hat{Y} - aZ - bR\right)/N$$

Thus, with this choice for *d*, the likelihood of the data becomes

$$L = \left(\frac{2\pi}{N}\left(\hat{Y} - aZ - bR\right)^{\mathrm{T}}\left(\hat{Y} - aZ - bR\right)\right)^{-N/2} \exp\left\{-\frac{N}{2}\right\}$$

It follows immediately that the likelihood is maximised for the values of *a* and *b* that minimises $\left(\hat{Y} - aZ - bR\right)^{\mathrm{T}}\left(\hat{Y} - aZ - bR\right)$. Consequently, the linear regression is equivalent to maximum likelihood estimation. The solution to the linear regression is

$$\begin{bmatrix} a \\ b \end{bmatrix} = Q^{-1}\begin{bmatrix} Z^{\mathrm{T}} \\ R^{\mathrm{T}} \end{bmatrix}\hat{Y} = Q^{-1}\begin{bmatrix} Z^{\mathrm{T}} \\ R^{\mathrm{T}} \end{bmatrix}Y + Q^{-1}\begin{bmatrix} Z^{\mathrm{T}} \\ R^{\mathrm{T}} \end{bmatrix}E; \qquad Q = \begin{bmatrix} Z^{\mathrm{T}}Z & R^{\mathrm{T}}Z \\ Z^{\mathrm{T}}R & R^{\mathrm{T}}R \end{bmatrix}$$

with $Y = [y_1, ..., y_N]^{\mathrm{T}}$. The covariance matrix for (*a*,*b*) is dQ^{-1}, where *d* is the variance of the noise.

MLE with Input Noise Case

Consider the dataset $\{\hat{y}_{n+1}(\hat{y}_n, r_n)\}_{n=1}^{N}$ and suppose the explanatory representation

$$y_{n+1} = ay_n + br_n;\ \hat{y}_n = y_n + \varepsilon_n$$

is sought. Let $\hat{Y}_{+1} = [\hat{y}_2,...,\hat{y}_{N+1}]^{\mathrm{T}}$ and $E = [\varepsilon_1,...,\varepsilon_N]^{\mathrm{T}}$ such that $\mathrm{T}E = \left(\hat{Y}_{+1} - a\hat{Y} - bR\right)$, where

$$\mathrm{T} = \begin{bmatrix} -a & 1 & 0 & \cdots & 0 \\ 0 & -a & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -a & 1 \end{bmatrix}$$

The likelihood of the data given (a,d,b) is

$$L = \frac{1}{(2\pi d)^{N/2}|P|^{1/2}} \exp\left\{-\left(\hat{Y}_{+1} - a\hat{Y} - bR\right)^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y} - bR\right)/(2d)\right\}$$

where

$$P = \mathrm{T}\mathrm{T}^{\mathrm{T}} = \begin{bmatrix} (1+a^2) & -a & 0 & \cdots & 0 \\ -a & (1+a^2) & -a & \ddots & \vdots \\ 0 & -a & (1+a^2) & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -a \\ 0 & \cdots & 0 & -a & (1+a^2) \end{bmatrix}$$

Similar to previous case, d can be eliminated to yield the likelihood of the data

$$L = \left(\frac{2\pi}{N}\left(\hat{Y}_{+1} - a\hat{Y} - bR\right)^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y} - bR\right)\right)^{-N/2} |P|^{-1/2} \exp\left\{-\frac{N}{2}\right\}$$

Hence, the equivalent technique to linear regression is to choose a and b that maximise the above likelihood function, L. However, note that analytic solution and derivation of the covariance matrix is no longer possible.

It follows that the negative log-likelihood function is

$$LL = -2\ln(L) \approx \ln(\det(P)) + N\ln\left\{\left(\hat{Y}_{+1} - a\hat{Y} - bR\right)^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y} - bR\right)\right\}$$

with its derivative with respect to $\theta = \{a,b\}$ as

$$\frac{\partial LL}{\partial \theta} = tr\left\{\frac{\partial P}{\partial \theta}\right\} + N\left[\Gamma^{\mathrm{T}} P^{-1}\Gamma\right]^{-1}\left\{2\frac{\partial}{\partial \theta}\left(\Gamma^{\mathrm{T}}\right)P^{-1}\Gamma - \Gamma^{\mathrm{T}}P^{-1}\left(\frac{\partial P}{\partial \theta}\right)P^{-1}\Gamma\right\}$$

where $\Gamma = \mathrm{T}\,E = \left(\hat{Y}_{+1} - a\hat{Y} - bR\right)$, and *tr*(.) is the trace operator of a matrix.

b can also be eliminated from the negative log-likelihood function to improve optimisation routine. By equating the derivative of *LL* with respective to *b* to be zero,

$$b = \frac{R^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y}\right)}{R^{\mathrm{T}} P^{-1} R}$$

Substituting the result back into the negative log-likelihood function,

$$LL = \ln(P) + N \ln\left\{\left(\hat{Y}_{+1} - a\hat{Y} - \frac{R^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y}\right)}{\left(R^{\mathrm{T}} P^{-1} R\right)} R\right)^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y} - \frac{R^{\mathrm{T}} P^{-1}\left(\hat{Y}_{+1} - a\hat{Y}\right)}{\left(R^{\mathrm{T}} P^{-1} R\right)} R\right)\right\}$$

Thus, a simple first order formulation for maximum likelihood estimation can be simplified to be dependent on only one parameter, *a*.

Appendix C

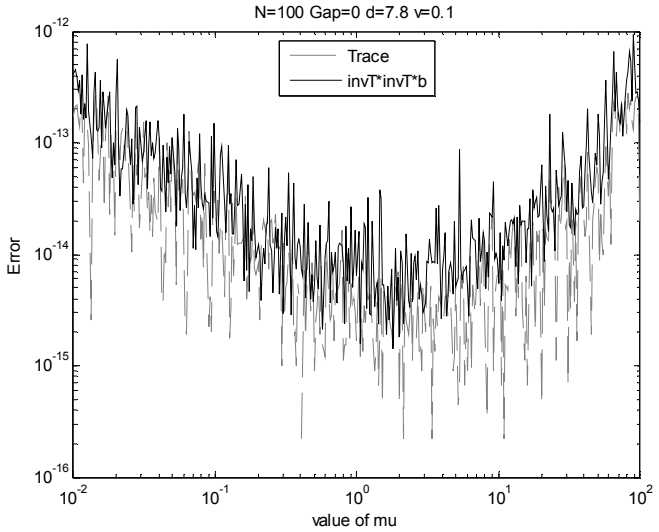
Some experimental results during the investigation of the generalised Schur algorithm that are incorporated into Gaussian regression are listed here.

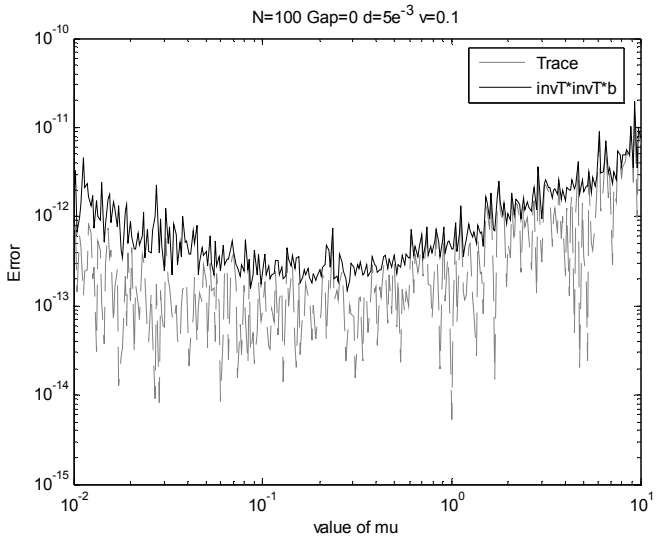
Experimental Results From the Investigation of Coefficient Factor, μ

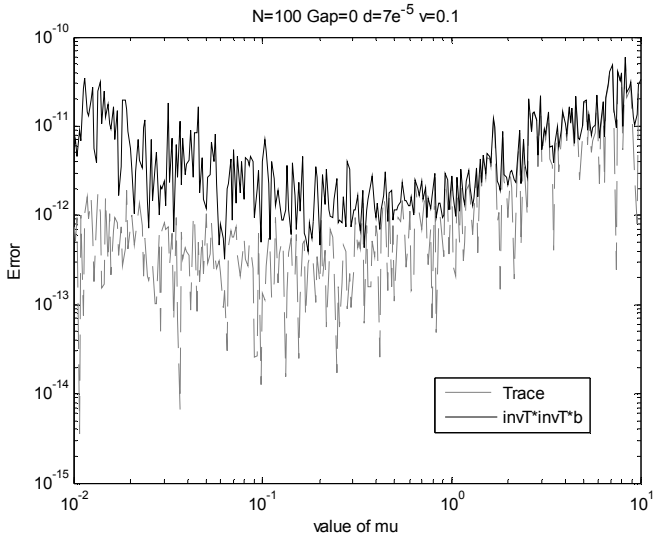
The best value for the convergence factor μ is close to that value of noise variance b. This is illustrated by some experiments.

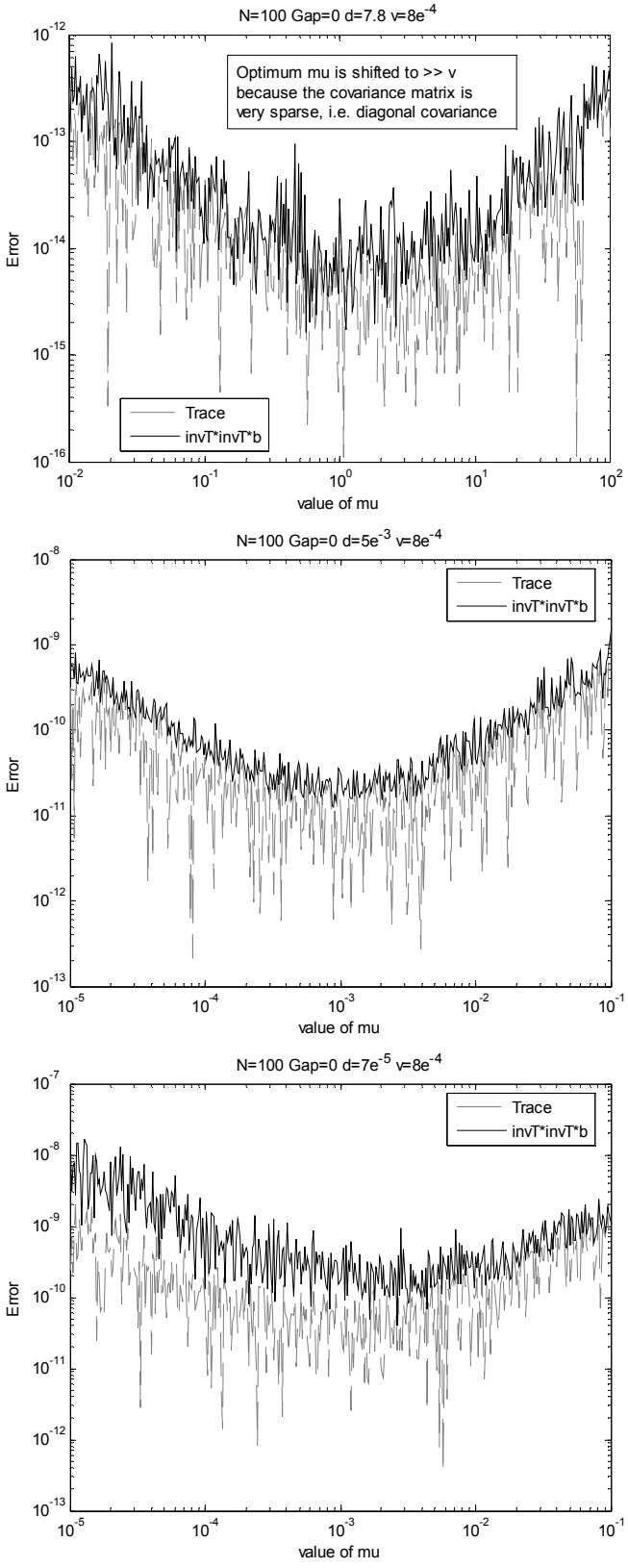
Three cases are presented. Firstly, sets of 100 data points are compared with different hyperparameter values. Secondly, sets of 100 data points with 4 missing gaps are compared with different hyperparameter values. Thirdly, sets of 500 data points are presented. The figures shown are in absolute errors, plotted on logarithmic scales. Note that the variable v is defined to be noise variance b in the following figures.

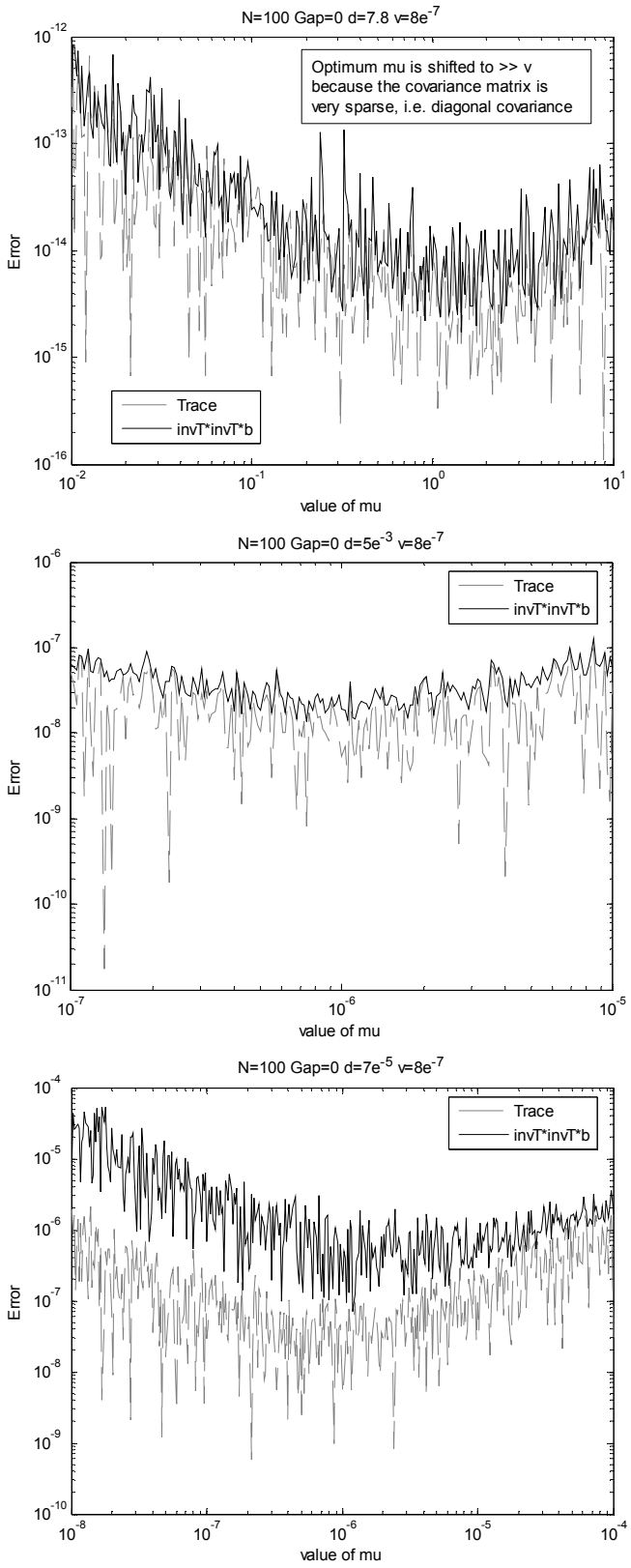
Case 1: $N = 100$, data without gap








N=100 Gap=0 d=7.8 v=8e-4
Optimum mu is shifted to >> v
because the covariance matrix is
very sparse, i.e. diagonal covariance
Trace
invT*invT*b
Error
value of mu
N=100 Gap=0 d=5e-3 v=8e-4
Trace
invT*invT*b
Error
value of mu
N=100 Gap=0 d=7e-5 v=8e-4
Trace
invT*invT*b
Error
value of mu

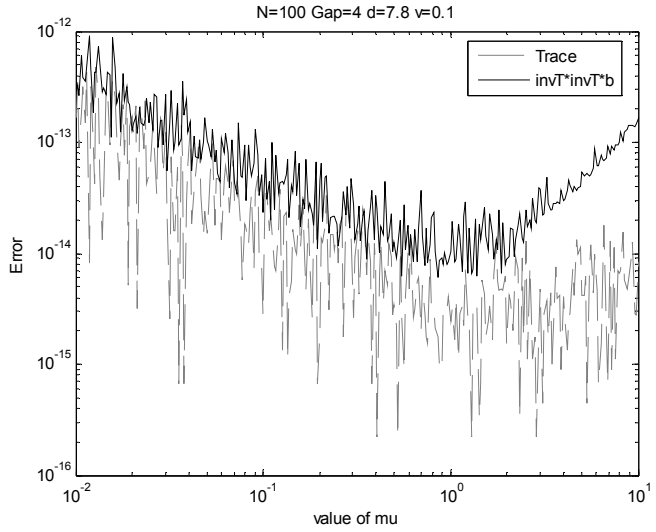


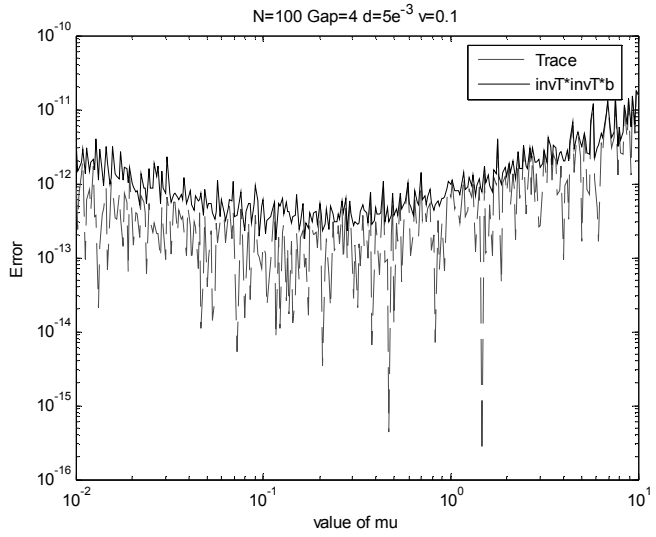
Three lengthscale hyperparameters d = {7.8, $5\text{x}10^{-3}$, $7\text{x}10^{-5}$} are illustrated on the respective first, second and third rows. The noise variance hyperparameter values, v =

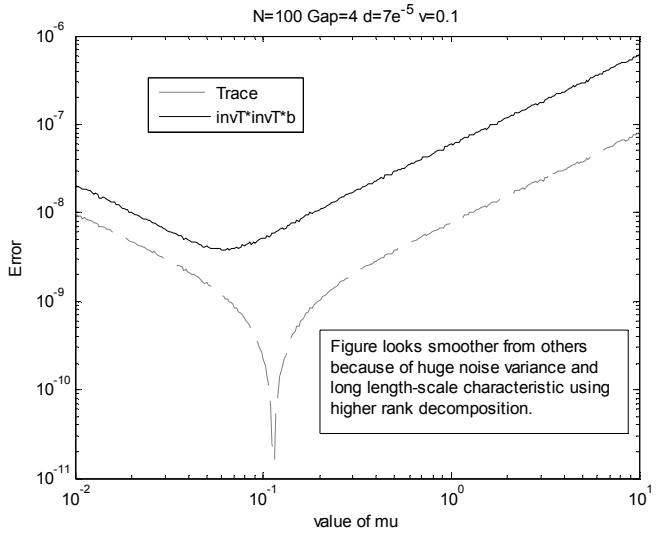
0.1 is shown on the first set of figures, $v = 8\text{x}10^{-4}$ is shown on the second set of figures, and $v = 8\text{x}10^{-7}$ shown on the third set of figures.

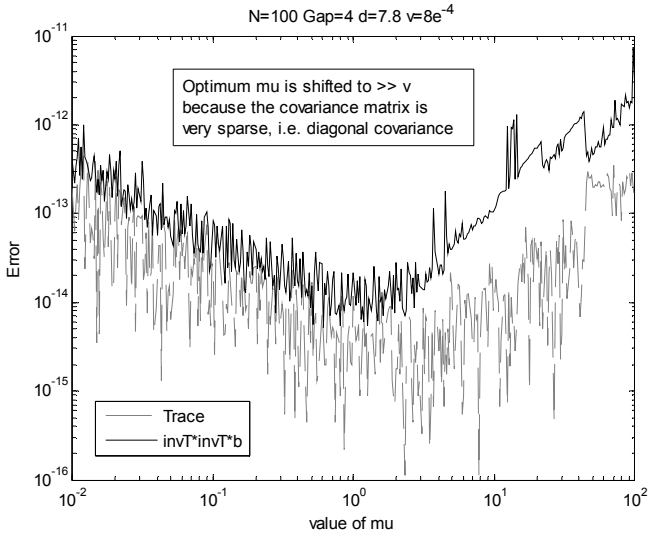
Clearly, a good choice of values of μ is the value of the noise variance, v, except for the case of the top row figures. The lowest absolute error does not correspond to the value of μ because that example has a very sparse covariance matrix. That value of μ corresponds to the lowest eigenvalue of the covariance matrix; that is, approximately 1 in this case (since hyperparameter $a = 1$).

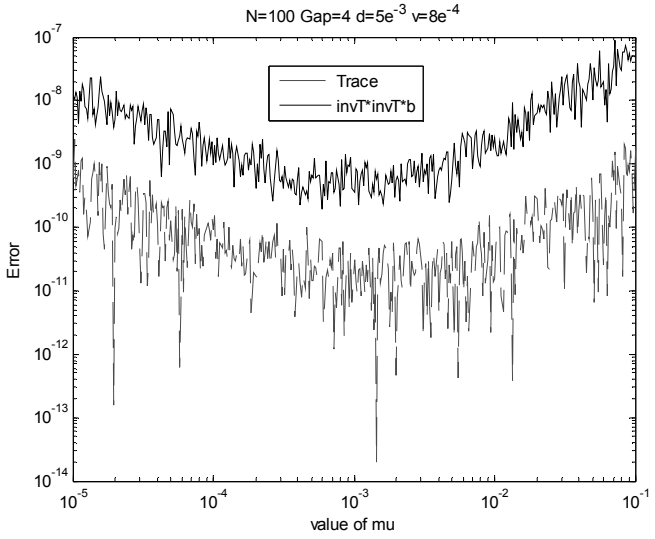
Case 2: $N = 100$, data with 4 missing gaps

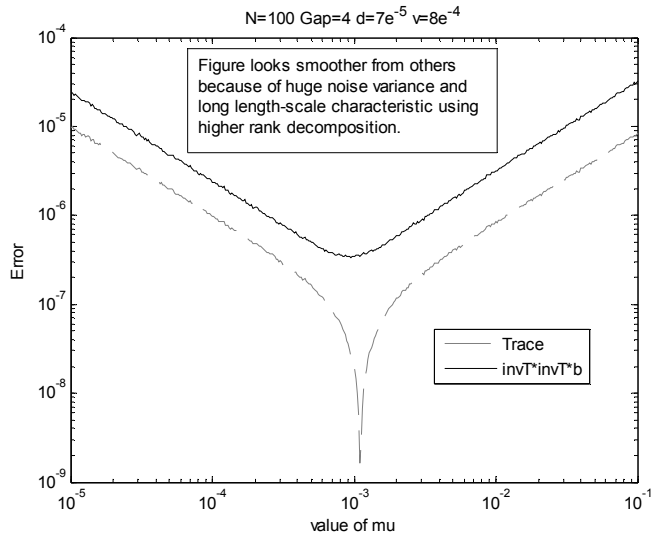


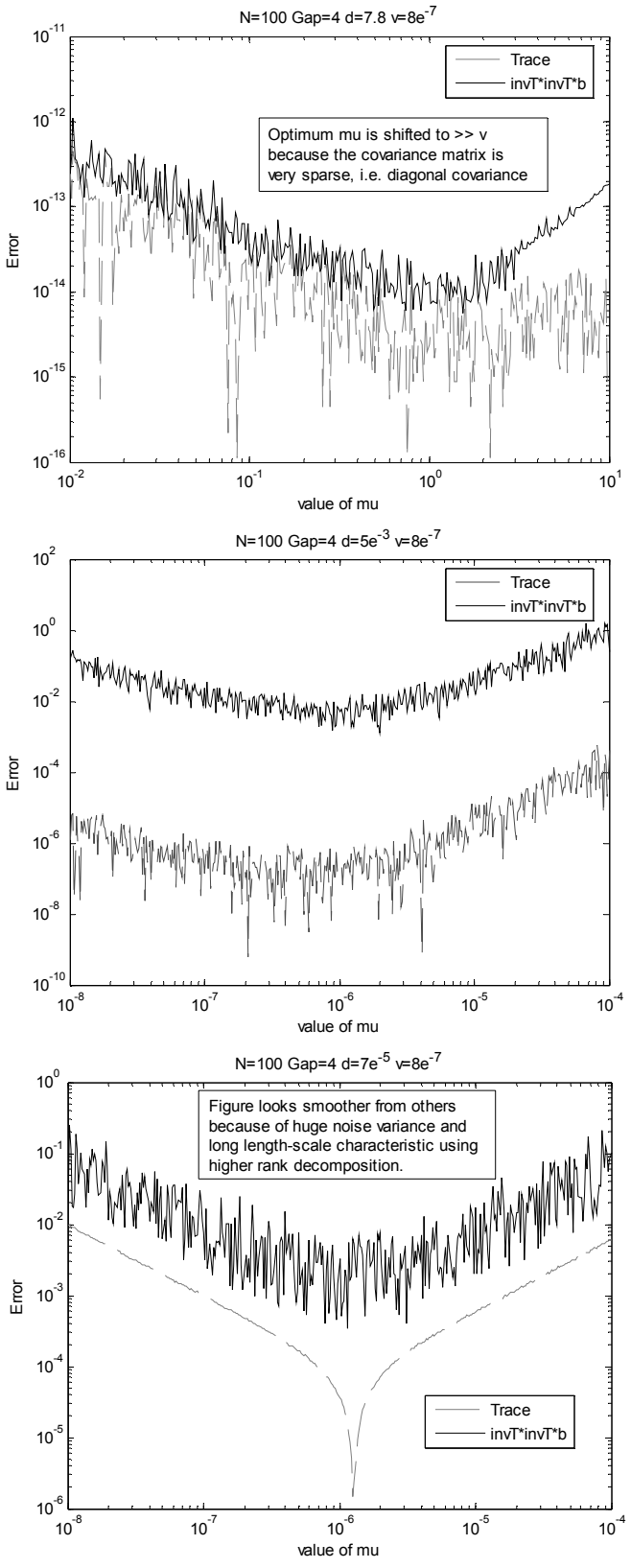





N=100 Gap=4 d=7.8 v=8e-4
Optimum mu is shifted to >> v
because the covariance matrix is
very sparse, i.e. diagonal covariance
Trace
invT*invT*b
Error
value of mu


N=100 Gap=4 d=5e-3 v=8e-4
Trace
invT*invT*b
Error
value of mu

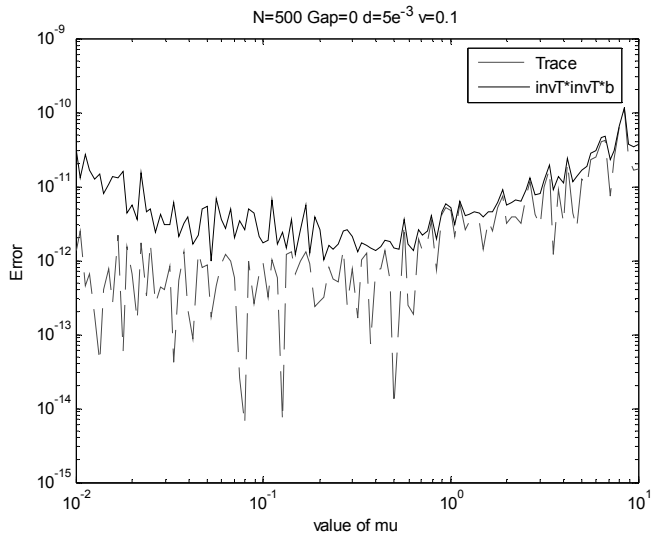

N=100 Gap=4 d=7e-5 v=8e-4
Figure looks smoother from others
because of huge noise variance and
long length-scale characteristic using
higher rank decomposition.
Trace
invT*invT*b
Error
value of mu

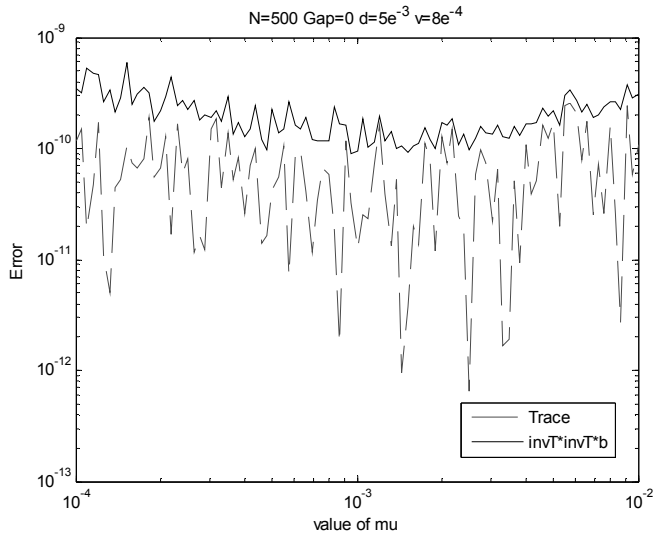


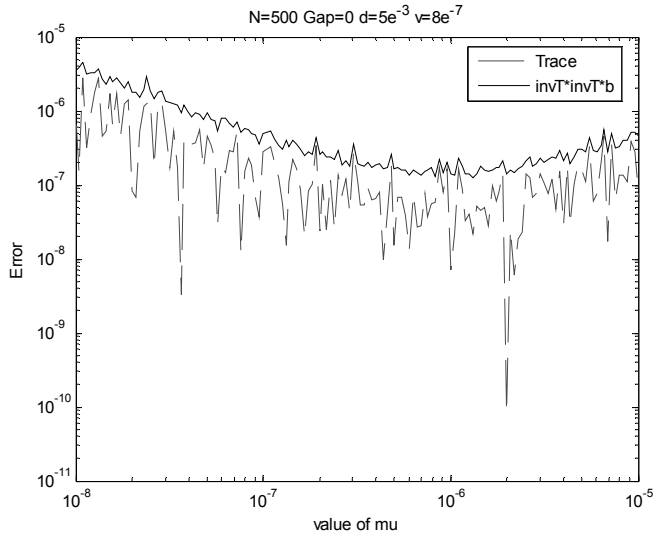
Case 2 illustrates the same scenario as in Case 1 but with dataset having 4 missing gaps. Three lengthscale hyperparameters d = {7.8, 5x10^{-3}, 7x10^{-5}} are illustrated on

the first, second and third rows, respectively. The noise variance hyperparameter values $v = 0.1$ is shown on the first set of figures, $v = 8\text{x}10^{-4}$ is shown on the second set of figures and $v = 8\text{x}10^{-7}$ on the third set of figures. The result is similar to Case 1; that is, the best choice for μ is the noise variance hyperparameter v, except for the top row figures, in which the covariance matrix is sparse. Note the third rows; the smoothness of the slopes is due to the presence of the huge noise variance and long lengthscale characteristics of the datasets.

Case 3: $N = 500$, data without gap







Case 3 investigates the case of larger dataset, i.e. data set with 500 data points. Focusing only on a particular lengthscale hyperparameter, $d = 5\text{x}10^{-3}$, that has a dense covariance matrix, three different noise variance hyperparameters are compared. Apparently, the best choice for μ remains approximately to be that noise variance hyperparameter, v, of the dataset.

Numerical Experimental Results of Function and Optimisation Tests

Below are supporting figures presented for Chapter 3.7.

Table C – I and C – II show the mean relative accuracies of the generalised Schur algorithm for various functions. The former is applied on strictly time-series data, whereas the latter is applied on time-series data with one missing gap. In these two experiments, 100 Gaussian process-generated samples are used for each data size, N, $\forall N = \{100,200,...,1000\}$.

TABLE C – I Relative errors of functions using Schur algorithm (data without gap)

	Relative Accuracies			
Functions	M	M(R)	C	C(R)
$\log\|Q\|$	$0.50\text{ x}10^{-14}$	$0.27\text{ x}10^{-14}$	$0.42\text{ x}10^{-14}$	$0.43\text{ x}10^{-14}$
$Q^{-1}\mathbf{Y}$	$0.53\text{ x}10^{-12}$	$0.60\text{ x}10^{-12}$	$0.35\text{ x}10^{-12}$	$0.40\text{ x}10^{-12}$
$tr(Q^{-1})$	$0.73\text{ x}10^{-13}$	$0.49\text{ x}10^{-13}$	$0.36\text{ x}10^{-13}$	$0.34\text{ x}10^{-13}$
$\mathbf{Y}^{\mathrm{T}}Q^{-1}\mathbf{Y}$	$0.72\text{ x}10^{-13}$	$0.46\text{ x}10^{-13}$	$0.34\text{ x}10^{-13}$	$0.32\text{ x}10^{-13}$
$\mathbf{Y}^{\mathrm{T}}Q^{-1}Q^{-1}\mathbf{Y}$	$0.14\text{ x}10^{-12}$	$0.09\text{ x}10^{-12}$	$0.07\text{ x}10^{-12}$	$0.06\text{ x}10^{-12}$
$\Phi_1 Q^{-1}\mathbf{Y}$	$0.22\text{ x}10^{-9}$	$0.22\text{ x}10^{-9}$	$0.44\text{ x}10^{-11}$	$0.42\text{ x}10^{-11}$
$tr(Q^{-1}\Phi_1)$	$0.11\text{ x}10^{-12}$	$0.12\text{ x}10^{-12}$	$0.06\text{ x}10^{-12}$	$0.04\text{ x}10^{-12}$
$\mathbf{Y}^{\mathrm{T}}Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$	$0.11\text{ x}10^{-10}$	$0.11\text{ x}10^{-10}$	$0.10\text{ x}10^{-10}$	$0.08\text{ x}10^{-10}$
$tr(Q^{-1}Q^{-1})$	$0.02\text{ x}10^{-8}$	$0.10\text{ x}10^{-8}$	$0.02\text{ x}10^{-8}$	$0.01\text{ x}10^{-8}$
$Q^{-1}Q^{-1}\mathbf{Y}$	$0.04\text{ x}10^{-8}$	$0.12\text{ x}10^{-8}$	$0.04\text{ x}10^{-8}$	$0.06\text{ x}10^{-8}$
$tr(\Phi_1 Q^{-1}\Phi_1 Q^{-1})$	$0.12\text{ x}10^{-10}$	$0.02\text{ x}10^{-10}$	$0.17\text{ x}10^{-10}$	$0.12\text{ x}10^{-13}$
$\Phi_1 Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$	$0.54\text{ x}10^{-11}$	$0.45\text{ x}10^{-11}$	$0.68\text{ x}10^{-11}$	$0.47\text{ x}10^{-11}$
$tr(Q^{-1}\Phi_1 Q^{-1})$	$0.77\text{ x}10^{-10}$	$0.50\text{ x}10^{-10}$	$0.85\text{ x}10^{-10}$	$0.42\text{ x}10^{-10}$

M refers to Schur algorithm programmed in MATLAB script, (R) refers to hyperbolic rotation integrated into the Schur algorithm, C refers to Schur algorithm compiled in C code.

TABLE C – II Relative errors of functions using Schur algorithm (data with one missing gap)

Functions	Relative Accuracies M	M(R)	C	C(R)
$\log\|Q\|$	0.72×10^{-13}	0.06×10^{-13}	0.41×10^{-13}	0.15×10^{-13}
$Q^{-1}\mathbf{Y}$	0.14×10^{-10}	0.02×10^{-10}	0.17×10^{-10}	0.02×10^{-10}
$tr(Q^{-1})$	0.58×10^{-12}	0.05×10^{-12}	0.77×10^{-12}	0.13×10^{-12}
$\mathbf{Y}^{T}Q^{-1}\mathbf{Y}$	0.51×10^{-12}	0.05×10^{-12}	0.79×10^{-12}	0.14×10^{-12}
$\mathbf{Y}^{T}Q^{-1}Q^{-1}\mathbf{Y}$	0.10×10^{-11}	0.01×10^{-11}	0.16×10^{-11}	0.02×10^{-11}
$\Phi_1 Q^{-1}\mathbf{Y}$	0.36×10^{-10}	0.34×10^{-10}	0.39×10^{-10}	0.34×10^{-10}
$tr(Q^{-1}\Phi_1)$	0.60×10^{-11}	0.08×10^{-11}	0.62×10^{-11}	0.02×10^{-11}
$\mathbf{Y}^{T}Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$	0.46×10^{-9}	0.05×10^{-9}	0.71×10^{-9}	0.03×10^{-9}
$tr(Q^{-1}Q^{-1})$	0.06×10^{-7}	0.01×10^{-7}	0.14×10^{-7}	0.30×10^{-9}
$Q^{-1}Q^{-1}\mathbf{Y}$	0.15×10^{-7}	0.03×10^{-7}	0.21×10^{-7}	0.02×10^{-7}
$tr(\Phi_1 Q^{-1}\Phi_1 Q^{-1})$	0.24×10^{-9}	0.61×10^{-12}	0.07×10^{-9}	0.78×10^{-12}
$\Phi_1 Q^{-1}\Phi_1 Q^{-1}\mathbf{Y}$	0.36×10^{-10}	0.16×10^{-10}	0.43×10^{-10}	0.16×10^{-10}
$tr(Q^{-1}\Phi_1 Q^{-1})$	0.34×10^{-8}	0.02×10^{-8}	0.36×10^{-8}	0.03×10^{-8}

M refers to Schur algorithm programmed in MATLAB script, (R) refers to hyperbolic rotation integrated into the Schur algorithm, C refers to Schur algorithm compiled in C code.

The numbers of iterations that the Gaussian process optimisation take to converge for the GP Tests (optimisation test) are tabulated in Table C – III and C – IV. The former is being applied on time-series data and the latter on time-series data with one missing gap. In these two tests, 20 samples are used for each data size, N, $\forall N = \{100,200,...,1000\}$.

TABLE C – III Iterations (average) on convergence (data without gap)

Data size	Gradient					Hessian				
N	C	C^R	M	M^R	S	C	C^R	M	M^R	S
100	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2
200	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3	5.3
300	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4
400	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5
500	6.2	6.2	6.2	6.2	6.2	6.2	6.2	6.2	6.2	6.2
600	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6
700	5.8	5.8	5.8	5.8	5.8	5.8	5.8	5.8	5.8	5.8
800	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9
900	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4
1,000	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4	6.4

C refers to optimisation using Schur algorithm programmed in C code. C^R denotes the C code includes hyperbolic rotation. M refers to the optimisation programmed in MATLAB script. M^R denotes that MATLAB script includes hyperbolic rotation. S defines the optimisation routine performed using standard MATLAB commands.

TABLE C – IV Iterations (average) on convergence (data with one missing gap)

Data size	Gradient					Hessian				
N	C	C^R	M	M^R	S	C	C^R	M	M^R	S
100	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9
200	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6
300	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6
400	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5
500	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2
600	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6
700	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7
800	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9
900	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9	5.9
1,000	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7	5.7

C refers to optimisation using Schur algorithm programmed in C code. C^R denotes the C code includes hyperbolic rotation. M refers to the optimisation programmed in MATLAB script. M^R denotes that MATLAB script includes hyperbolic rotation. S defines the optimisation routine performed using standard MATLAB commands.

Appendix D

Linear Analysis

Linear covariance functions have been investigated by several researchers (Rasmussen and Williams, 2006). Stochastic processes using linear covariance function for multiple Gaussian processes model is discussed in this section. It is known that the result from using the standard linear regression model is very similar to the least square regression method (Rasmussen and Williams, 2006).

Proposition *(Linear Regression Model Using Maximum Likelihood Estimation (MLE))*. Given that the data is $y \approx f(\mathbf{x}) + \varepsilon$, where $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{w}$, such that $\mathbf{x}$ is the input vector and $\mathbf{w}$ is the weighting vector. f is the function value whereas y is the observed target value. This is known as the standard linear regression model. It is assumed that Gaussian white noise is the difference between the observed target value and the true function value as given by the distribution, $\varepsilon \sim \mathrm{N}\left(0, \sigma_n^2\right)$, such that the error is simply given by $\varepsilon_i = y_i - f_i(\mathbf{x}),\ \forall i = 1,...,N$. The model together with the noise assumption gives rise to the likelihood of the data. The joint probability distribution of the prior is

$$p(y \mid \mathbf{x}, \mathbf{w}) = \frac{1}{\left(2\pi\sigma_n^2\right)^{n/2}} \exp\left[-\frac{1}{2\sigma_n^2}\left|y - \mathbf{x}^{\mathrm{T}}\mathbf{w}\right|^2\right]$$

By substituting $d = \sigma_n^2$, it follows that the log-likelihood function is written as

$$\begin{aligned}\Lambda &= \log\left\{\frac{1}{(2\pi d)^{n/2}}\exp\left[-\frac{1}{2d}\left|y-\mathbf{x}^{\mathrm{T}}\mathbf{w}\right|^2\right]\right\} \\ &= -\frac{1}{2d}\left(E^{\mathrm{T}}E\right)-\frac{n}{2}\log(2\pi)-\frac{n}{2}\log(d)\end{aligned}$$

where $E = \left(y-\mathbf{x}^{\mathrm{T}}\mathbf{w}\right)$. In the case of linear regression, $f(\mathbf{x}) = \mathtt{a}\mathbf{x}+\mathtt{b}$, where $\mathtt{a}$ and $\mathtt{b}$ are coefficients of the standard linear model. Thus, the negative log-likelihood function simplified to the form

$$\begin{aligned}\overline{\Lambda} &\approx \frac{1}{d}\left(E^{\mathrm{T}}E\right)+n\log(d) \\ &= \frac{1}{d}\left(y-\mathtt{a}\mathbf{x}-\mathtt{b}\right)^{\mathrm{T}}\left(y-\mathtt{a}\mathbf{x}-\mathtt{b}\right)+n\log(d)\end{aligned}$$

It follows that the first order derivative of the negative log-likelihood function is

$$\frac{\partial\overline{\Lambda}}{\partial d} = -\frac{1}{d^2}\left(E^{\mathrm{T}}E\right)+\frac{n}{d}$$

Setting the first order derivative to zero, $d = \frac{1}{n}\left(E^{\mathrm{T}}E\right)$ and substituting it back to the log-likelihood function,

$$\begin{aligned}\overline{\Lambda} &= \frac{n}{\left(E^{\mathrm{T}}E\right)}\left(E^{\mathrm{T}}E\right)+n\log\left[\frac{1}{n}\left(E^{\mathrm{T}}E\right)\right] \\ &= n-n\log(n)+n\log\left(E^{\mathrm{T}}E\right)\end{aligned}$$

Removing redundant constant and coefficients that do not affect the result of maximising the log-likelihood function, a simple formulation for the negative log-likelihood function (112) is available, with $\mathtt{a}$ and $\mathtt{b}$ the parameters to be optimised.

$$\overline{\Lambda} \cong \left(y-\mathtt{a}\mathbf{x}-\mathtt{b}\right)^{\mathrm{T}}\left(y-\mathtt{a}\mathbf{x}-\mathtt{b}\right) \tag{112}$$

Simplification of the Gradient Information

Gradient information is included in the optimisation routine to minimise the negative log-likelihood function (112). The first order derivative information of the negative log-likelihood function with respect to the parameters is shown below.

1. $\frac{\partial \overline{\Lambda}}{\partial \mathrm{a}} = -2\mathbf{x}^{\mathrm{T}}\left(y - \mathrm{a}\mathbf{x} - \mathrm{b}\right)$
2. $\frac{\partial \overline{\Lambda}}{\partial \mathrm{b}} = -2\sum_{i=1}^{N}\left(y_i - \mathrm{a}\mathbf{x}_i - \mathrm{b}\right)$

Though not necessary, the Hessian information is also provided here.

1. $\frac{\partial^2 \overline{\Lambda}}{\partial \mathrm{a}^2} = 2\mathbf{x}^{\mathrm{T}}\mathbf{x}$
2. $\frac{\partial^2 \overline{\Lambda}}{\partial \mathrm{b}^2} = 2N$
3. $\frac{\partial^2 \overline{\Lambda}}{\partial \mathrm{a}\partial \mathrm{b}} = 2\sum_{i=1}^{N}\mathbf{x}_i$

The above information is sufficient for the optimisation procedure to be carried out on the standard linear regression model. This procedure is also widely known as the Maximum Likelihood Estimation (MLE).

Linear Covariance Function

To apply the non-stationary linear covariance function to the Gaussian process prior model, only the definition of that covariance function is required. The prior covariance function for the linear component is given in (113), where $\left\{x_i^k\right\}_{k=1}^{D} \equiv \mathbf{x}_i$.

$$C\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sum_{k=1}^{D} w_k x_i^k x_j^k \tag{113}$$

The hyperparameter, w_k, in the covariance function (113) is adapted to maximise the log-likelihood function of the Gaussian process. The result from the Gaussian process prior model is almost as good as performing the MLE procedure. This is demonstrated by the following example.

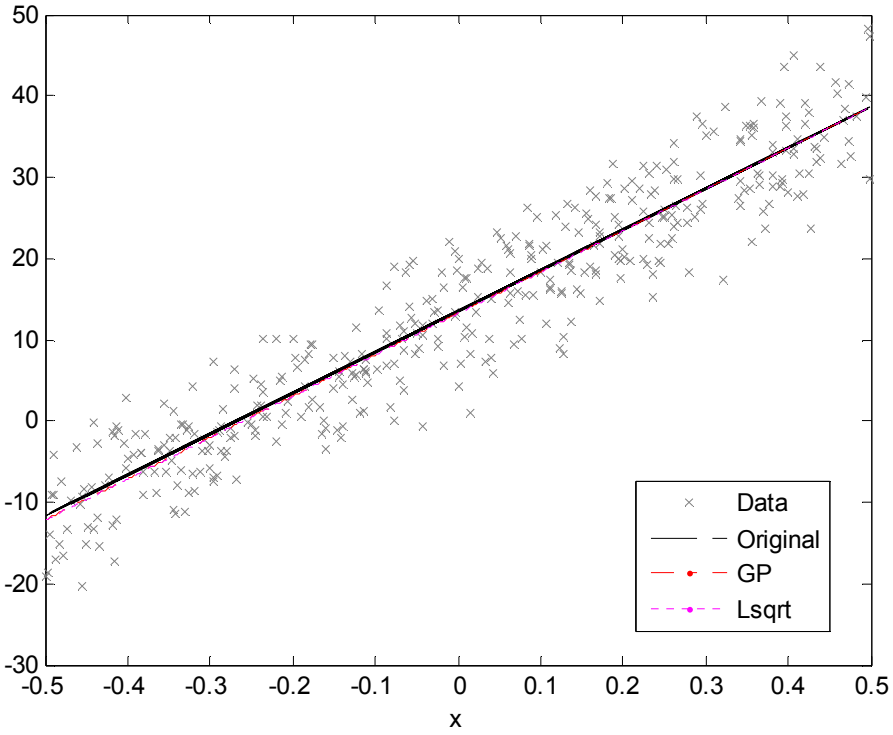


Figure D – 1 Data and predictions using standard linear regression and Gaussian process with linear covariance function.

Example *(Comparing Linear Least Square Regression and GP Using Linear Covariance Function)*. A simple linear function is given by $f_i = ax_i + b$, where a = 50.275 and b = 13.507. Additive Gaussian white noise of variance 26.01 is introduced to the data of 400 measurements, in the range $[-0.5 \le x \le 0.5]$. This is merely a simple polynomial function of first degree which can be solved using standard linear least square regression technique. The noisy data is shown in Figure D – 1. Standard linear least square regression and Gaussian regression are carried out and compared. The predictions are also shown in Figure D – 1.

For the case of linear least square regression, the coefficients obtained are a = 50.8 and b = 13.2. The prediction for the test data is shown by the *dash* line in the figure. In the case of Gaussian regression, the value of the hyperparameter adapted to maximise the likelihood function is $w = 2.58 \times 10^3$. The prediction is shown in the figure by the *dash-dot* line. Despite the huge noise intensity, it can be seen that the results of both approaches are quite coherent.

The derivative observations can be computed directly from the Gaussian process prior model. From the linear covariance function given in (113), the covariance between the derivative observations and the measurement data, and the covariance between the derivative observations and itself are defined in the following equations, respectively.

$$C\left(\mathbf{x}_i, \dot{x}_j\right) = \sum_{k=1}^{D} w_k x_i^k$$

$$C\left(\dot{x}_i, \dot{x}_j\right) = \sum_{k=1}^{D} w_k$$

These two equations provide the correlation necessary for the full probabilistic description of the derivative stochastic process.

Example *(cont.)*. It follows that the derivative observations are predicted using Gaussian process prior models. The fit to the derivative is just a constant since a linear covariance function is used in the Gaussian process. The constant value of the derivative is 50.16, which corresponds closely to the exact value of a = 50.275, defining the true gradient of the line. The variance of the posterior, as a result from the stochastic process, is also a constant value of 0.7791. Compared to the standard linear least square regression, Gaussian regression shows a slightly better prediction.

Quadratic Functions

The quadratic covariance function is another example belonging to the class of non-stationary covariance functions. The joint probability distribution will not be discussed since the same analogy can be found in the previous section (*on Linear Functions*). The quadratic covariance function contains the product of the weighting coefficient and the explanatory variable. A prior covariance function for a quadratic component is written in the form (114), where $\left\{x_i^k\right\}_{k=1}^{D} \equiv \mathbf{x}_i$.

$$C\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sum_{k=1}^{D} w_k \left(x_i^k\right)^2 \left(x_j^k\right)^2 \tag{114}$$

The hyperparameter w_k of the covariance function (114) is adapted to maximise the log-likelihood function of the Gaussian process.

Similar to the linear covariance function, the derivative observation for a quadratic form of the Gaussian process can be computed. The covariance between the prediction point and the measurement point, and the covariance between the prediction point and itself are given by the following equations, respectively.

$$C\left(\mathbf{x}_i, \dot{x}_j\right) = 2\sum_{k=1}^{D} w_k \left(x_i^k\right)^2 \left(x_j^k\right)$$

$$C\left(\dot{x}_i, \dot{x}_j\right) = 4\sum_{k=1}^{D} w_k \left(x_i^k\right)\left(x_j^k\right)$$

The first and second order derivatives with respect to $\mathbf{x}^k$ are essential to the construction of the derivative observation for the stochastic process.

Example *(Gaussian Process Using Quadratic Covariance Function)*. Given a data of 261 measurements from the underlying function, $f = 0.5x^2 - 100$, in the range $\left[-34 \le x \le -21\right]$, a quadratic covariance function is chosen for the Gaussian regression. The data points are selected far away from $x = 0$ because it is of interest to examine the performance of the prediction at different set of data points other than the training data. The noisy data is shown in Figure D – 2 by grey crosses. The noise in the data is additive Gaussian white noise with a variance of 100. In this experiment, Gaussian regression using quadratic function is applied on the data. The prediction with confidence intervals is shown with straight lines. The original function is illustrated by the dashed line and it is observed that the prediction is reasonably good.

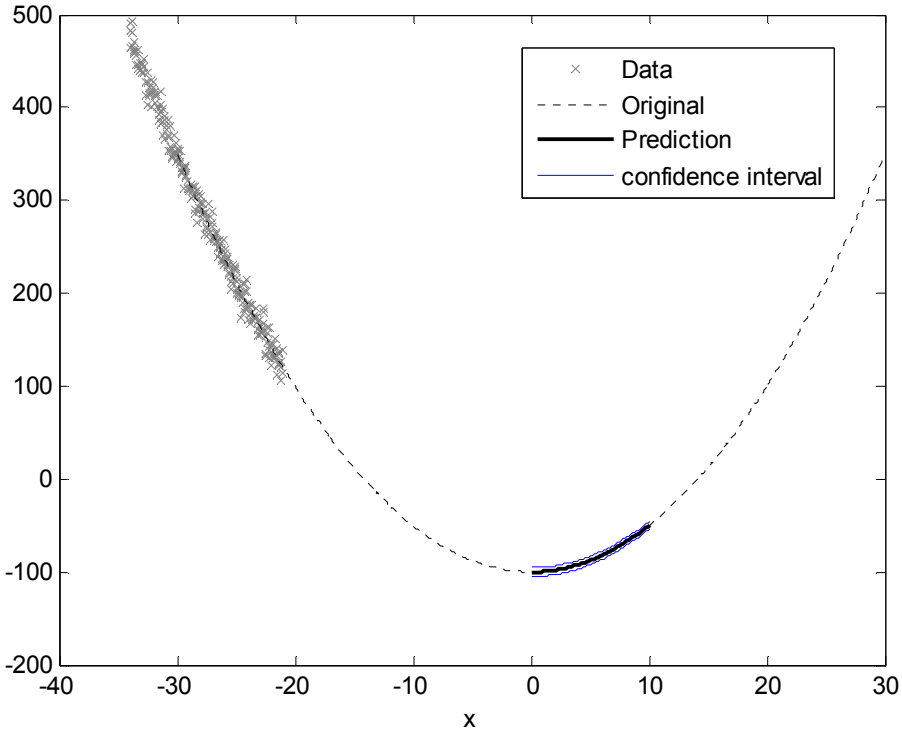


Figure D – 2 Data, prediction and confidence interval using Gaussian process with quadratic covariance function.

The derivative observation of the posterior joint probability distribution is a linear function as shown in Figure D – 3. The confidence interval of the posterior is zero at the origin and increases linearly as it goes along the positive and negative axes away from the origin.

The linear and quadratic covariance functions are widely known as dot product covariance functions (Rasmussen and Williams, 2006). The study of the squared exponential covariance function and the two examples from the class of non-stationary covariance functions is particularly interesting. The former is adaptable to the prior mean regardless if the prior mean is non-zero. The latter covariance functions are more restrictive in their application, particularly having to adhere to the prior assumption that the mean has to be zero.

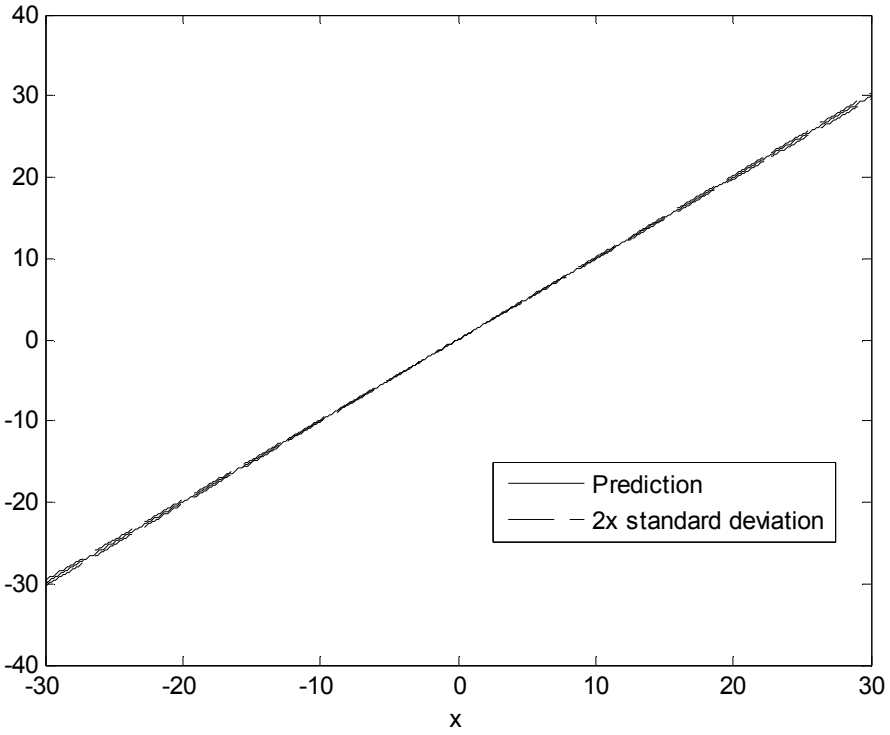


Figure D – 3 Derivative prediction and confidence interval using Gaussian process with quadratic covariance function.

Appendix E

Throughout the thesis, hyperparameters are adapted, conditioned on the data to maximise the log-likelihood function in Gaussian process prior models. In some other cases, iterative-type issues are also encountered, such as finding the solution to locate the zero-crossing problem.

MATLAB Optimisation Toolbox

Most of the optimisation scripts written for the research are based on the tools provided in MATLAB Optimisation Toolbox (MathWorks, 2003). Particularly, large-scale optimisation involving trust-region algorithms are used throughout the research. Some of the scripts allow users to explicitly supply Hessian information. Two of the optimisation functions are highlighted below.

1. *fmincon* – constrained minimisation function of several variables.
2. *fminunc* – unconstrained minimisation function of several variables.

Without user-supplied Hessian information, these functions are able to approximate the second order derivative by finite-differencing.

Appendix F

Several source codes are programmed throughout the development of Chapter 2 to Chapter 6. Due to the massive library of source codes being made, they are stored in a Compact Disc-Read Only Memory (CD-ROM) format (at the back of the thesis). The source codes are also made available at the following website: http://www.hamilton.ie/keith/research.htm.

About The Source Codes

The source codes are mainly written in either MATLAB or C (for the use of MEX-C) language. All source codes in any distribution are subjected to the following copyright license:

Limitation and Liability

I shall not be liable for infringements of third parties rights. In no events, unless required by applicable law, shall I be liable for any direct, indirect, incidental, special, exemplary, or consequential damages of any character including, without limitation, damages for loss of good will, work stoppage, computer failure or malfunction, or any and all other damages or losses, even if advised of the possibility of such damage. Also, I am under no obligation to maintain, correct, update, change, modify, or otherwise support these source codes.

Disclaimer

The source codes is provided “AS IS” without warranty of any kind. The functions that are contained in the source codes are not warranted to meet users’ requirement or that the operation will be uninterrupted or error-free.

Appendix G

Some of the machine specifications are listed here.

Machine A (Desktop System)

The specifications are as follow.

Type	Description	Remarks
Brand	DELL® Computer Corporation	
Model	OptiPlex GX280	
Processor	Intel® Pentium® IV	
Speed	2.8 Gigahertz (GHz)	
Memory	512 Mega Bytes (MB) DDR2 Ram	Double data rate
Cache	L1 – 8kB, L2 – 512kB	
Operating Systems	Microsoft® Windows XP Professional, SuSE Linux 9.3 Professional	Dual boot system
Front Side Bus	800 MHz	
Hard disk	80 GB (7,200 rpm*)	
Graphics	Integrated Intel® i82915G chip onboard	
Optical Media	CD-RW drive	
Other Features	Hyper-threading, MMX, SSE, SSE2	

*rpm denotes number of revolutions per minute.

Machine B (Laptop)

The specification is as follow.

Type	Description	Remarks
Brand	DELL® Computer Corporation	
Model	Inspiron 5150	
Processor	Mobile Intel® Pentium® IV	
Speed	3.06 Giga Hertz (GHz)	
Memory	512 Mega Bytes (MB) DDR2 Ram	Double data rate
Cache	L1 – 8kB, L2 – 512kB	
Operating Systems	Microsoft® Windows XP Professional, SuSE Linux 9.3 Professional	Dual boot system
Front Side Bus	533 MHz	
Hard disk	60 GB (5,400 rpm)	
Graphics	Nvidia GeForce FX Go5200	Dedicated 64 MB
Optical Media	Combo DVD-CDRW drive	
Other Features	Hyper-threading, MMX, SSE, SSE2	

Bibliography

Abramowitz, M. and Stegun, I. A. (1965). *Handbook of Mathematical Functions*. pp. 84-85, Dover, New York.

Alexandrov, N., Dennis Jr., J. E., Lewis, R. M. and Torczon, V. (1998). A trust region framework for managing the use of approximation models in optimization. In *Journal on Structural Optimization*, vol. 15, pp. 16-23.

Adler, R. J. (1981). *The Geometry of Random Fields*. Wiley, Chichester.

Archambeau, C., Conford, D., Opper, M. and Shawe-Taylor J. (2007). Gaussian process approximations of stochastic differential equations. In *JMLR Workshop and Conference Proceedings 1*, pp. 1-16.

Aronszajn, N. (1950). Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, vol. 68, No. 3, pp. 337-404.

Barry, R. P. and Pace, R. K. (1999). Monte Carlo estimate of the log determinant of large sparse matrices. In *Linear Algebra and its Applications*, vol. 289, pp. 41-54.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philos. Trans. Royal Society London*, vol 53, pp. 370-418.

Berman, Simeon M. (1990). A stochastic model for the distribution of HIV latency time based on T4 counts. In *Biometrika*, vol. 77, pp. 733-741.

Bernoulli, J. (1713). *Ars Conjectandi*. Basel: Thurnisiorum.

Box, G. E. P. and Tiao, G. C. (1973). *Bayesian inference in statistical analysis*. Addison-Wesley.

Byrd, R. H., Schnabel, R. B. and Shultz, G. A. (1998). Approximate solution of the trust region problem by minimization over two-dimensional subspaces. In *Mathematical Programming*, vol. 40, pp. 247-263.

CATS Benchmark (2004). Time Series Prediction Competition. In *International Joint Conference on Neural Networks*, http://www.cis.hut.fi/~lendasse/competition/competition.html.

Chandrasekaran S. and Sayed, A. H. (1996). Stabilizing the Generalized Schur Algorithm. In *SIAM Journal Matrix on Analysis and Applications*, vol. 17, No. 4, pp. 950-983.

Chandrasekaran S. and Sayed, A. H. (1999a). Stabilized Schur Algorithms. In *Fast reliable algorithms for matrices with structure*, Kailath, T. and Sayed, A. H. (Ed.), pp. 57-83.

Chandrasekaran S. and Sayed, A. H. (1999b). Fast Stable Solvers for Structured Linear Systems. In *Fast reliable algorithms for matrices with structure*, Kailath, T. and Sayed, A. H. (Ed.), pp. 85-102.

Conn, A. R., Gould, N. I. M. and Toint, Ph. L. (2000). Trust region methods. In *SIAM*. Philadelphia.

Cressie, N. A. C. (1993*). Statistics for spatial data*. Wiley, New York.

Dennis, J. E. and Schnabel, R. B. (1983). Numerical methods for unconstrained optimization and nonlinear equations. In *Prentice Hall*. Englewood Cliffs, New Jersey.

Duane, S., Kennedy, A. D., Pendleton, B. J. and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, vol. 195, pp. 216-222.

Freedman, L. S., Fainberg, V., Kipnis, V., Midthune, D. and Carroll, R. J. (2004). A new method for dealing with measurement error in explanatory variables regression models. *Biometrics*, vol. 60 (1), pp. 172-181.

Gibbs, M. N. (1997). *Bayesian Gaussian processes for regression and classification*. Ph.D. thesis. Cambridge University, U.K.

Gibbs, M. N. and Mackay, D. J. C. (1997). Efficient implementation of Gaussian process. In *Technical report*, Cavendish Laboratory. Cambridge University, U.K.

Gibbs, M. N. and Mackay, D. J. C. (2000). Variational Gaussian process classifiers. In *IEEE Transactions on Neural Networks*, vol. 11, pp.1458-1464.

Girard, A. (2004). *Approximate Methods for Propagation of Uncertainty with Gaussian Process Models*. Ph.D. thesis. University of Glasgow, U. K.

Girosi, F., Jones, M. and Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computations*, vol. 7, No. 2, pp. 219-269.

Goldberg, P. W., Williams, C. K. I. and Bishop, C. M. (1998). Regression with input-dependent noise: a Gaussian process treatment. In *Advances in Neural Information Processing Systems*, Jordan, M. I., Kearns, M. J. and Solla S.A. (Ed.), vol. 10, pp. 493-499, MIT Press, Cambridge.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. 3rd edition. The John Hopkins University Press, Baltimore and London.

Grimmett, G. R. and Stirzaker, D. R. (1992). *Probability and random processes*. Oxford University Press, 2nd edition, Oxford, England.

Jung, L. (1999). *System Identification. Theory for the user*. Prentice Hall, 2nd edition, New Jersey.

Kailath, T. (1999). Displacement structure and array algorithms. In *Fast reliable algorithms for matrices with structure*, Kailath, T. and Sayed, A. H. (Ed.), pp. 1-56.

Kocijan, J., Girard, A., Banko, B. and Murray-Smith, R. (2003) Dynamic systems identification with Gaussian processes. In *4th MATHMOD Vienna, 4th IMACS Symposium on Mathematical Modelling*, vol. 2, pp. 776-784.

Kulhavý, R. and Ivanova, P. (1999). Quo Vadis, Bayesian Identification? In *International Journal of Adaptive Control and Signal Processing*, vol. 13, pp. 469-485.

Laplace, P. S. (1812). Théorie analytique des probabilités. *Paris: V. Courcier*.

Leith, D. J., Leithead, W. E. Solak, E. and Murray-Smith, R. (2002). Divide and conquer identification using Gaussian process priors. In *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 11, pp. 624-629.

Leith, D. J., Heidl, M. and Ringwood, J. V. (2004). Gaussian process prior models for electrical load forecasting. In *Proceedings of the 8th International Conference on Probabilistic Methods Applied to Power Systems*, Iowa States University, pp. 112-117.

Leith, D. J., Murray-Smith, R., and Leithead, W. E. (2000). Nonlinear structure identification: A Gaussian process prior/velocity-based approach. *Control 2000*, Cambridge.

Leithead, W. E. (1992). Effective wind speed models for simple wind turbine simulation. In *Proceedings of the British Wind Energy Conference*, pp. 321-326.

Leithead, W. E., Hardan, F. and Leith, D. (2003a). Identification of aerodynamics and drive-train dynamics for a variable speed wind turbine. In *Proceedings of European Wind Energy Conference*. Madrid, Spain.

Leithead, W. E., Leith, D. J., Hardan, F. and Markou, H. (1999). Global gain-scheduling control for variable speed wind turbines. In *Proceedings of the European Wind Energy Conference*, pp. 853-856.

Leithead, W. E., Neo, K. S. and Leith, D. J. (2005a). Gaussian regression based on models with two stochastic processes. In *Proceedings of the 16th IFAC World Congress*. Prague, Czech Republic.

Leithead, W. E., Solak, E. and Leith, D. (2003b). Direct identification of nonlinear structure using Gaussian process prior models. In *Proceedings of European Control Conference*. Cambridge, U.K.

Leithead, W. E. and Zhang, Y. (2007). $O(N^2)$-operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method. In *Comms in Statistics – Simulation and Computation*, in press.

Leithead, W. E., Zhang, Y. and Leith, D. J. (2005b). Efficient Gaussian process based on BFGS updating and Logdet approximation. In *Proceedings of the 16th IFAC World Congress*. Prague, Czech Republic.

Leithead, W. E., Zhang, Y. and Leith, D. J. (2005c). Time-series Gaussian process regression based on Toeplitz computation of $O(N^2)$ operations and $O(N)$-level storage. In *Proceedings of the 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference (CDC-ECC)*, pp. 3711-3716. Seville, Spain.

Leithead, W. E., Zhang, Y. and Neo, K. S. (2005d). Wind turbine rotor acceleration: Identification using Gaussian regression. In *Proceedings of the 2nd International Conference on Informatics in Control, Automation and Robotics*. Barcelona, Spain.

Lu, J. and Adachi, T. (1989). A new global optimization algorithm using stochastic model and its application to electronic circuit design. In *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 2044-2047. Portland, USA.

MacKay, D. J .C. (1992). Bayesian interpolation. *Neural Computation*, 4(3), pp. 415-447.

Mackay, D. J. C. (1993). Hyperparameters: Optimize, or integrate out? In *Maximum Entropy and Bayesian Methods*, Heidbreder, G. (Ed.), pp. 43-59, Kluwer Academic Publisher. Santa Barbara.

Mackay, D. J. C. (1997). Probable networks and plausible predictions – A review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6, pp. 469-505.

Mackay, D. J. C. (1998). Introduction to Gaussian processes. In *Neural Networks and Machine Learning, F: Computer and Systems Sciences*, Bishop, C. M. (Ed.), pp. 133-165, Springer. Berlin, Heidelberg.

Mardia, K. V. and Marshall, R. J. (1984). Maximum likelihood estimation for models of residual covariance in spatial regression. In *Biometrika*, vol. 71, pp. 135-146.

MathWorks, Inc. (2003). Optimization Toolbox User's Guide, Version 2.3.

Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. In *Neural Networks*, vol. 6, pp. 525-533.

Moré, J. J. and Sorensen, D. C. (1983). Computing a trust region step. In *SIAM Journal of Scientific and Statistical Computing*, vol. 4, pp. 553-572.

Moré, J. J. and Thuente, D. J. (1994). Line search algorithms with guaranteed sufficient decrease. In *ACM Transactions on Mathematical Software*, vol. 20, pp. 286-307.

Murray-Smith, R. and Girard, A. (2001). Gaussian process priors with ARMA noise models. In *Irish Signals and Systems Conference*, pp. 147-153. Maynooth, Ireland.

Murray-Smith, R., Johansen, T. A. and Shorten, R. (1999). On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. In *Proceedings of the 5th European Control Conference*. Karlsruhe, Germany.

Murray-Smith, R., Sbarbaro, D., Rasmussen, C. E. and Girard, A. (2003). Adaptive, cautious, predictive control with Gaussian process priors. In *Proceedings of the IFAC on System Identification*. Rotterdam.

Neal, R. M. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. *Technical Report CRG-TR-97-2*, Dept. of Computer Science, University of Toronto.

Neal, R. M. (1996). Bayesian Learning for Neural Networks. *Lecture Notes in Statistics*, no. 118, New York, Springer.

Neo, K. S, Leithead, W. E. and Zhang, Y. (2006). Multi-frequency scale Gaussian regression for noisy times-series data. *In Proceedings of the 6th bi-ennial UKACC Control Conference, International Control Conference 2006*. Glasgow, U.K.

Nocedal, J. (1992). Theory of algorithms for unconstrained optimization. In *Acta Numerica*, pp. 199-242. Cambridge University Press, U.K.

O'Hagan, A. (1978). On curve fitting and optimal design for regression. *Journal of the Royal Statistical Society B*, 40, pp. 1-42.

O'Hagan, A. (1992). Some Bayesian numerical analysis. *Bayesian Statistics 4*, Bernado, J. M., Berger, J. O., Dawid, A. P. and Smith, A. F. M. (Ed.), pp. 345-363. Oxford University Press, U.K.

O'Hagan, A. (1994). Bayesian Inference. *Kendall's Advanced Theory of Statistics*. Edward Arnold.

Paciorek, C. J. (2003). *Nonstationary Gaussian processes for regression and spatial modelling*. Ph.D. thesis. Department of Statistics, Carnegie Mellon University, Pittsburgh Pennsylvania.

Papoulis, A. (1991). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). *Numerical Recipes in C*. Cambridge University Press, 2nd edition.

Pugachev, V. S. (1967). *Theory of random functions and its application to control problems*. Pergamon Press.

Quiñonero-Candela J. and Rasmussen C. E. (2005). A unifying view of sparse approximate Gaussian process regression. In *Journal of Machine Learning Research*, vol. 6, Herbrich R. (Ed.), pp. 1939-1959.

Rasmussen, C. E. (1996). *Evaluation of Gaussian processes and other methods for non-linear regression*. Ph.D. thesis, University of Toronto.

Rasmussen, C. E. (2003). Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals. *Bayesian Statistics 7*, Bernado, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M. and West, M. (Ed.), pp. 651-659. Oxford University Press, U.K.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for Machine Learning*. MIT Press, Cambridge.

Sambu S., Wallat, M., Graepel, T. and Obermayer K. (2000). Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 241-246.

Sayed, A. H. and Kailath T. (1994). Extended Chandrasekhar Recursions. In *IEEE Transactions on Automatic Control*, vol. 39, No. 3, pp. 619-623.

Sayed, A. H., Kailath, T. and Lev-Ari, H. (1994). Generalized Chandrasekhar Recursions from the Generalized Schur Algorithm. In *IEEE Transactions on Automatic Control*, vol. 39, No. 11, pp. 2265-2269.

Schwaighofer A. and Tresp, V. (2003). Transductive and inductive methods for approximate Gaussian process regression. In *Advances in Neural Information Processing Systems*, vol. 15, Becker, S., Thrun, S. and Obermayer K. (Ed.), pp. 953-960. Cambridge, Massachusetts, MIT Press.

Shi, J. Q., Murray-Smith, R. and Titterington, D. M. (2003). Bayesian regression and classification using mixtures of multiple Gaussian processes. In *International Journal of Adaptive Control and Signal Processing*, vol. 1 17, pp. 149-161.

Seeger, M. (2000). Skilling techniques for Bayesian analysis. In *Technical Report*, Institute for Adaptive and Neural Computation. University of Edinburgh, U.K.

Seeger, M., Williams, C. K. I. and Lawrence, N. D. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Bishop, C. M. and Frey, B. J. (Ed.), Society for Artificial Intelligence and Statistics.

Skilling, J. (1993). Bayesian numerical analysis. In *Physics and Probability*, Grandy Jr., W. T. and Milonni, P. W. (Ed.). Cambridge University Press, U.K.

Smola, A. J. and Barlett, P. L. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems*, vol. 13, Leen, T. K., Ditterich, T. G. and Tresp, V. (Ed.), pp. 619-625. Cambridge, Massachusetts, MIT Press.

Solak, E., Murray-Smith, R., Leithead, W. E., Leith, D. J. and Rasmussen, C. E. (2003). Derivative observations in Gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems*, vol. 15, pp. 1033-1040, MIT Press.

Stein, M. L. (1999). *Interpolation of spatial data*. Springer-Verlag, New York.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. In *SIAM Journal on Numerical Analysis*, vol. 20, pp. 626-637.

Stewart, M. (2003). A superfast Toeplitz solver with improved numerical stability. In *SIAM Journal on Matrix Analysis and Applications*, vol. 25, pp. 669-693.

Vincente, L. N. (1996). A comparison between line searches and trust regions for nonlinear optimization. In *Investigação Operacional*, vol. 16, pp. 173-179.

Wang, Y. and Krishna, H. (1989). On fast and super fast algorithms for solving block Toeplitz systems. In *23rd Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 643-647.

Williams, C. K. I. (1999). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In *Learning in Graphical Models*, Jordan, M. I. (Ed.), pp. 599-621.

Williams, C. K. I. and Barber, D. (1998). Bayesian classification with Gaussian processes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1342-1351.

Williams, C. K. I and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, Touretzky, D. S., Mozer, M. C. and Hasselmo, M. E. (Ed.), MIT Press.

Wu, Z. Phillips Jr., G. N., Tapia, R. and Zhang, Y. (2001). A fast Newton method for entropy maximization in statistical phase estimation. In *Acta Crystallographica*, vol. A57, pp. 681-685.

Yaglom, A. M. (1987). Correlation Theory of Stationary and Related Random Functions, vol. 1, *Basic Results*, Springer Verlag.

Yoshioka, T. and Ishii, S. (2001). Fast Gaussian process regression using representative data. In *Proceedings of International Joint Conference on Neural Networks*, vol. 11, pp. 132-137.

Zhang, Y. and Leithead, W. E. (2005). Exploiting Hessian matrix and trust-region algorithm in hyperparameter estimation of Gaussian process. In *Applied Mathematics and Computation*, 171, pp. 1264-1281.

Zhang, Y. and Leithead, W. E. (2007). Approximate implementation of logarithm of matrix determinant in Gaussian processes. In *Journal of Statistical Computation and Simulation*, in press.